The Resource Pooling Principle

Damon Wischik
University College London
d.wischik@cs.ucl.ac.uk

Mark Handley
University College London
m.handley@cs.ucl.ac.uk

Marcelo Bagnulo Braun
UC3M, Madrid
marcelo@it.uc3m.es

This article is an editorial note submitted to CCR. It has NOT been peer reviewed. Authors take full responsibility for this article's technical content. Comments can be posted through CCR Online.

ABSTRACT

Since the ARPAnet, network designers have built localized mechanisms for statistical multiplexing, load balancing, and failure resilience, often without understanding the broader implications. These mechanisms are all types of *resource pooling*, which means *making a collection of resources behave like a single pooled resource*. We believe that the natural evolution of the Internet is that it should achieve resource pooling by harnessing the responsiveness of multipath-capable end systems. We argue that this approach will solve the problems and limitations of the current piecemeal approaches.

Categories and Subject Descriptors: C.2.1 [Computer communication networks]: Network architecture and design

Keywords: resource pooling, traffic engineering, load balancing, statistical multiplexing, multipath

1 Introduction

There is a silent revolution that is reshaping the Internet. It began with the original design of statistical multiplexing through packet switching. As demands on the Internet grew, network operators began to use traffic engineering to balance load, and sites began to be multihomed for improved resilience against failures. The revolution is that now end systems are involved in managing wide-area traffic patterns, for example peer-to-peer applications, or load balancing by Google and Akamai across globally-distributed server farms.

These mechanisms are all examples of *resource pooling*. The general concept is that the network's resources should behave as though they make up a single pooled resource; the aims are to increase reliability, flexibility and efficiency. Figure 1 illustrates full pooling of link capacity and partial pooling of reliability in the face of link failure. In Section 2.1 we list attempts to date at resource pooling in the Internet.

Different parties have devised mechanisms for partial resource pooling, and implemented their mechanisms independently. This has caused stress to other parts of the network. Sometimes the different mechanisms fight against each other: for example, network operators can no longer perform traffic engineering within their networks as though the traffic were fixed, because end-system reactions mean that the matrix can change on the timescale of a few RTTs. In Section 2.2 we list some of the problems with the way that resource pooling is done today.

This paper is a call to arms for a new way of thinking about routing and congestion control. We believe that the natural next step in the evolution of the Internet is to harness the responsiveness of end systems to achieve resource pooling. If end systems could spread their load across multiple paths in the right way, with the right reaction to the right congestion signals from the network, then traffic would quickly move away from congested or failed links in favor of uncongested links. In Section 3.1 we explain the end-system multipath architecture, and in Section 3.2 we explain how it would achieve resource pooling—robustness against failure, load balancing, and flexibility in the face of bursts of traffic—while avoiding the problems and limitations of current piecemeal approaches. Wireless mobile devices in particular can benefit from pooling the capacity of their various radio channels (GPRS, WiFi, bluetooth, etc.) in environments where one channel on its own is too intermittent.

Many questions still need answering before we can fully benefit from resource pooling. In Section 4 we attempt to outline a research agenda, and metrics by which we might judge the success of our proposed architectural changes. Some parts of the solution are well understood: in the last few years researchers have developed models of how multipath-capable end-systems might respond to congestion signals, and what the potential benefits might be. But to go from these theoretical models to actual changes in operating system stacks and ISP routing systems will require much work.

We conclude in Section 5 by extrapolating from our study of transport-layer resource pooling a general architectural principle, the *resource pooling principle*: "Resource pooling is such a powerful tool that designers at every part of the network will attempt to build their own load-shifting mechanisms. A network architecture is effective overall, only if these mechanisms do not conflict with each other."

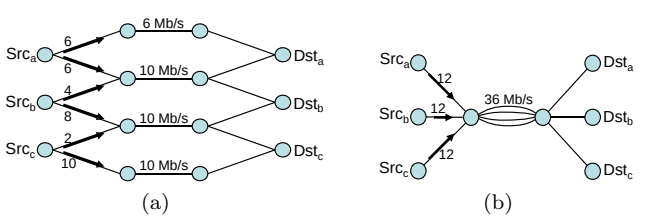

Figure 1. Pooling capacity: the entire 36 Mb/s capacity in (a) can be shared fairly and each flow can achieve more than it could over a single path, as though there were load balancing over the four links in (b). **Partial pooling of reliability:** each flow in (a) is more robust against failure of one of the four bottleneck links than if it only had access to a single path, but not quite as robust as in (b).

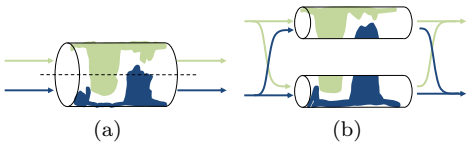

Figure 2. Pooling capacity (a) between "circuits" on a single link, and (b) across links. In both cases pooling improves the ability to cope with bursts or unexpected traffic patterns; the latter case achieves robustness to link failure.

2 Current practice of resource pooling

The main benefits of resource pooling are, in order,
(1) increased robustness against component failures;
(2) better ability to handle localized surges in traffic;
(3) maximized utilization.
In this section we list the mechanisms that end users and network operators have *already devised* for obtaining these benefits. We also list some of the problems which have arisen because of the piecemeal way in which these mechanisms were developed: BGP scaling, bad interactions between network and user mechanisms, inadequate robustness, and missed opportunities.

2.1 Goals and mechanisms

Statistical multiplexing through packet switching is the most fundamental concept in the Internet architecture. Computer communication is inherently bursty; ideally we would be able to transfer any size chunk of data in a single burst. Packet switching allows a finite capacity link to be used at maximum efficiency, and it achieves this through resource pooling in two ways. Figure 2a illustrates the first type of resource pooling: whereas circuit switching would prevent a burst of traffic on one circuit from using spare capacity on other circuits, packet switching allows the entire link capacity to be used whenever there is demand. Buffers are the second type of resource pooling: they allow the capacity at one time period to be pooled with capacity at the next. Regular TCP congestion control also allows this type of pooling, spreading transfers out over time.

Robustness through dynamic alternative routing. In the telephone network, operators including BT and AT&T use dynamic alternative routing: if a link fails or is overloaded, calls are automatically routed along an alternative set of links. In effect the capacities of different paths are pooled, and so the network as a whole has greater resilience to link failure and to unexpected surges in traffic: see [6] for a striking example of resilience in a multiparented network much like BT's network today.

Resource pooling via multipath routing is the *only* way the phone network can achieve high reliability, greater than the reliability of the individual switches and links. Internet routing is already more dynamic than telephone routing, and provides resilient service despite poorly coordinated growth. However, the demand for a robust Internet has been steadily growing as it has become an essential part in business, and also due to the rise in interactive applications such as VoIP and gaming. The slow convergence of conventional Internet routing isn't really up to this challenge, and the robustness that might come from resource pooling provided by load-dependent routing has proved elusive and remains the holy grail of routing systems.

Robustness and load balancing through multihoming. A site may have links to two or more network providers; as long as one of them remains up then the site still has connectivity. In other words the reliability of the links to individual network providers is pooled. BGP can be used for this: a non-aggregatable prefix is announced to the world via each provider, drawing traffic to the site via both paths.

The site may achieve more fine-grained capacity pooling, rather than coarse-grained reliability pooling, by routing some nodes via one link and some nodes via the other. BGP can also be used for this: extra more-specific prefixes are advertised to a subset of the nodes.

Reliability and capacity pooling can also be achieved at the transport level, for example by SCTP. If the end-systems are multihomed, with different addresses from each network provider, then SCTP will choose which address to use at each end and it can switch over in the case of failure or poor throughput.

Load balancing through traffic engineering. Network operators engineer traffic to balance load within their networks. For example, they might have several different MPLS labels corresponding to different possible paths from a given ingress point to an egress point, and they could label flows so as to balance traffic in the network.

Figure 3 shows the potential of multi-path load balancing: the network is able to handle a larger range of traffic matrices. In Figure 3, it is as if there is a single link of capacity 200 Mb/s, and any traffic matrix is admissible if the total arrival rate is less than 200 Mb/s. That is, the network achieves resource pooling. The constraint "total traffic $\leq$ 200 Mb/s" is called a *virtual resource constraint* since it is a constraint that does not correspond directly to any physical resource. Virtual resources of this sort were introduced by [13, 17].

Resource pooling results in a more flexible network, more resilient to localized surges in traffic, and capable of functioning well even when the actual traffic matrix differs substantially from that envisaged by the network designers. A secondary benefit is that operators can get higher utilization out of their infrastructure.

Interdomain traffic engineering. ISPs also use BGP to shift traffic flowing between networks. In part this achieves resilience against failure, by using resource pooling of the same sort as described for multihoming. In part it is a crude mechanism for coping with surges in traffic, by using manually tuned resource pooling of the same sort as described for internal traffic engineering.

Content distribution networks. Akamai, Google and others use huge numbers of redundant servers distributed in multiple data centers worldwide. Using DNS load-balancing and hardware load balancing at each data center, they pool the CPU cycles, bandwidth and reliability of these servers. If a hot spot develops, traffic can be moved to less overloaded servers or network links. If a server dies, it is just dropped from the pool. Such CDNs can move large amounts of traffic from one part of the Internet to another, independently of any traffic engineering performed by upstream ISPs.

Peer-to-peer content dissemination, as in BitTorrent, pools the instantaneous upload capacity of many network nodes. It also pools capacity over time, in a similar way to buffers

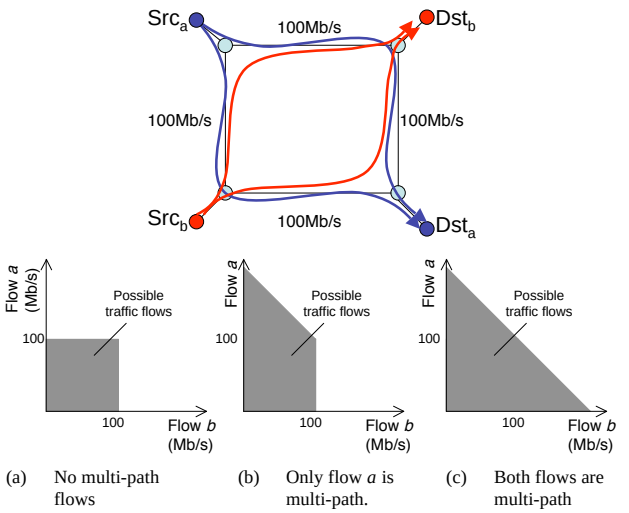

Figure 3. Resource Pooling increases the set of traffic matrices that can be served.

pooling link capacity over time. Furthermore, BitTorrent selects peers in part according to the throughput achieved; this moves flows away from congestion hotspots, performing a type of load-balancing which, as we have discussed, is a form of resource pooling. BitTorrent's rarest first algorithm also pools for reliability at the chunk level, preferentially copying chunks of data that are less common so as to pool the unreliable storage of the nodes in the session.

2.2 Problems with current mechanisms

Scalability of Interdomain Routing. The global BGP routing table contains about 300,000 entries and is updated up to a million times a day[10], with a growth rate that is worrying ISPs and router vendors[20]. About half the entries are for so-called more-specific prefixes[19]; these are routes for which there is also a less specific aggregate route that should in principle be sufficient to reach the same destination. About 44% are associated with multihoming[19], injecting a more specific network prefix via each of their providers. About 40% of all the more specific routes can be associated with traffic engineering techniques used to modify the normal BGP routing behavior[19]. A key reason why ISPs do this to move a subset of traffic from an overloaded path towards one with available capacity.

These more specific routes tend to be more volatile than average because they expose edge-link failures directly to the whole world, increasing BGP churn.

Slowness of failure recovery. The most serious limitation of current resource pooling techniques is that they are not very good at recovering from network failure. Demanding applications ideally want no interruption at all, but realistically might be willing to accept an RTT or so of disruption. Is it possible adapt techniques from the telephone network to increase the resilience of the Internet? We will argue in Section 3 that transport-layer resource pooling will make this possible.

The current network does respond to failures, but slowly. BGP responds by re-routing traffic although convergence times may be measured in minutes [16]. A single ISP can tweak OSPF to give faster reconvergence [5], and MPLS

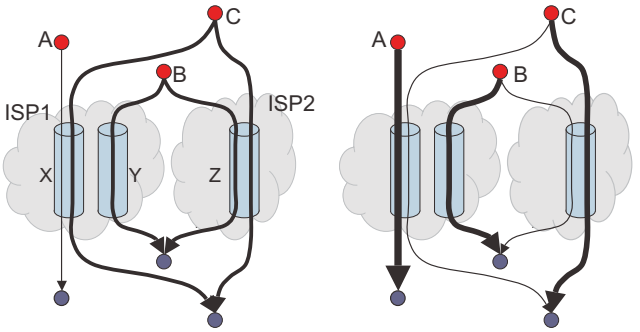

Figure 4. Traffic engineering works better when end systems and network operators cooperate. If there is a surge in traffic on route A, and if the other flows B, and C use multiple paths, then they could (with the right feedback from the network) re-balance themselves so that they do not use the congested link at ISP1. Even though traffic moves to link Y at ISP1, ISP1 could not on its own achieve this by local measures.

and link-layer path restoration can provide still faster convergence, but these are not end-to-end mechanisms so they cannot address the full range of end-to-end failure modes. Other end-to-end mechanisms for fault tolerance have been proposed such as the REAP protocol [3] or HIP[21]. However, in all these cases only one path is used at a time and because they are network layer protocols it is challenging to identify failures in a transport layer agnostic way, resulting in response times that are measured in seconds [3].

Failure is not the only cause of path change. Current mobility mechanisms [11, 21] perform handovers by abruptly changing from one attachment point to another, resulting in a sudden burst of traffic on a new path. While techniques like bi-casting[18] have been proposed, they seem overkill for regular traffic, since the data rate doubles while bi-casting. We will argue that transport-layer resource pooling can naturally deal with path change in a congestion-friendly way.

Bad interactions between users and network. Peer-to-peer applications such as BitTorrent do load-balancing based on their own selfish criteria. Often the result is positive: BitTorrent preferentially retrieves data along uncongested paths. If the Internet used congestion pricing, BitTorrent would come close to optimizing for cost. However, with the current charging models for peering, there can be a substantial mismatch between minimal congestion paths and minimal cost paths [2]. If the ISPs and the end-systems have different metrics for 'congestion', it can be shown that the cost of anarchy (i.e. the degradation in performance due to conflicting load-shifting) can be arbitrarily high [23, 1].

If the load-shifting mechanisms of end systems and of ISPs could be aligned, then the Internet as a whole would be better at traffic engineering. Figure 4 shows a scenario where multipath-capable end systems can (with the right feedback) achieve a better traffic engineering outcome than the ISPs can by themselves.

3 An end-system multipath architecture

In this section we will explain the proposal for multipath routing, and how it solves the problems listed in Section 2. We will go into some detail, so that it does not seem too far-fetched that responsive end-systems might be harnessed to achieve network-wide resource pooling.

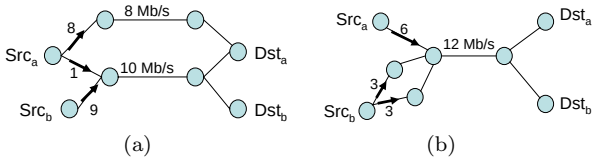

Figure 5. Fair rate allocations. These allocations would not be achieved by running TCP independently on each subflow.

3.1 The proposal

We propose a multipath-capable TCP which works as follows. Assume that either the sender or the receiver or both is multihomed, and that it has been assigned several IP addresses. In the initial handshake, the sender and the receiver exchange all their IP addresses. Then the multipath TCP sets up multiple subflows, from some or all of the sender's IP addresses to some or all of the receiver's IP addresses. These subflows will each use window-based congestion control, though the congestion windows will be coordinated as described below. Reliable delivery will be implemented over the set of subflows, so that if a packet is dropped on one subflow it may be resent on another; this improves reaction time to timeouts. Congestion on a path can be indicated by ECN marks. Multipath TCP is incrementally deployable—it is a drop-in replacement for TCP—but there are still practical issues and other limitations which we discuss in Section 4.

The congestion windows of the subflows should be coupled, for example as described by [12, 7]. They suggest that the window sizes should increase as per standard TCP Reno, and that when a subflow receives a drop or an ECN mark it should cut its window size, but that the cut in window size should be proportional not to its current window but rather to the aggregate rate of the sender. This has the outcome of allocating rates fairly to end users, as illustrated in figures 1 and 5. Also, when there is a failure on one path, the traffic on that path should not be immediately transferred to other paths, since that is likely to result in network-wide instability; instead, the window on the other paths should gradually ramp over multiple round trip times.

Adding multipath support to TCP is so obvious that it has been re-invented many times [9, 8, 22, 4], and multihoming is built into SCTP, though no protocol that *simultaneously* uses multiple paths has ever been standardized let alone widely deployed. Why is there not more multipath at the transport layer? Perhaps because it has not been understood that multipath lets end systems solve network-wide resource pooling problems, and because the issues with current mechanisms are only now becoming pressing enough to fix.

3.2 The benefits

Better response to failure and other path change. When multiple paths are used simultaneously, the failure of a path can be regarded as an extreme case of congestion. Multipath TCP will react by diverting the traffic through paths that are still working and have available capacity. The response time can be on the order of a round-trip-time, with retransmissions routed through alternative working paths. Moreover, where resilience is critical, redundant information can be used across the multiple paths so that even if packets in flight are lost due to path failure, enough information is

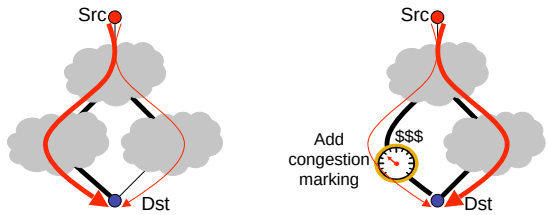

Figure 6. Resource pooling can load balance between links used for multihoming without the need for more specific routing prefixes.

delivered to the destination via the remaining paths.

Multipath TCP provides smooth handovers in make-before-break mobility scenarios which will be commonplace when multiple radios are used. Both old and new paths can be used simultaneously and the traffic distribution adapts to the available rate for each path, making a smooth handover without suddenly dumping traffic onto a new path and causing congestion.

The improved response to path changes is important to end users, since they will obtain better resiliency, but it is also a motivation for network operators. Path change events would behave in a more congestion-friendly manner, requiring less spare capacity to handle bursts.

Fast simple traffic engineering. Multipath TCP allows the network operator to ECN-mark traffic on congested or expensive links, causing it to shift automatically to other paths (Figure 6). This achieves traffic engineering on timescales of RTTs, using only local information, and without loading BGP with more-specific routes. Network operations will be simplified, since instead of manually tweaking OSPF or BGP, much of the matching of load to spare capacity would be automatic.

Multipath TCP helps with the problem shown in Figure 4, since the multipath flows will automatically move away from congestion hotspots. This is better traffic engineering than can be accomplished by an ISP on its own. It is an interesting open question what new sorts of traffic management tools might emerge in a world where most flows are congestion-sensitive multipath.

Multihoming without stressing BGP. The use of multiple aggregatable addresses to achieving multipath forwarding will naturally give the benefits of multihoming without imposing costs on BGP. The same is true of HIP, Shim6 and SCTP—but multipath TCP is more resilient than these approaches because it uses multiple paths simultaneously, and it better for traffic engineering because it is sensitive to congestion.

Mobile hosts with multiple radios are an extreme case of multi-homing, and multipath TCP is an ideal solution. As access links come and go, active connections gain and lose addresses as they are learned via DHCP. Traffic is split between these addresses, balanced naturally according to congestion, or indirectly according to link cost using ECN to migrate traffic away from high-cost paths. The one addition that is needed is a home agent, à la Mobile IP, to bootstrap connections. Unlike Mobile IP though, the home agent can remain in the connection's address set, but only be called upon in transient moments when the other addresses all fail, typically because both ends move simultaneously.

4 Research agenda

How can ISPs do traffic engineering? Network operators still need the ability to control traffic on their networks—how can they avoid bad interactions with multipath congestion control at the end systems? We have suggested that ECN marks will allow the network to influence the end-system control loop. The network might also do traffic engineering at a slower timescale, e.g. by per-flow routing. It seems reasonable to expect that this will not interfere with the end systems.

We will need new tools for traffic engineering that anticipate how end systems will shift their load. The notion of a bottleneck link has to be replaced by a bottleneck 'generalized cut', i.e. a set of links across which end systems do load balancing and which collectively form a bottleneck [17, 13]. These generalized cuts may span several different networks, which makes traffic engineering much more complicated. Figure 4 shows the limitations of traffic engineering via local measures.

What are the design issues with multipath TCP? A key question is whether to use a sequence space per subflow or one sequence space across all subflows. The former, while requiring extra sequence numbers in each packet, is more in line with the existing TCP protocol; as TCP SYN and FIN flags occupy sequence space, a per subflow sequence space is needed for them to be cleanly acked. Acks are also simpler, as the cumulative ack can serve its role effectively[1] without being critically dependent on selective acknowledgments. Also for pragmatic reasons, the more each subflow looks like a standalone flow to middleboxes, the lower the deployment barriers.

Application-limited connections also present a challenge, as these cannot use their full congestion window. Such connections can get worse throughput than a single-path TCP if they open too many subflows to maintain a good ack clock. They also present a stability challenge because when a subflow fails they will often have spare window available on the remaining subflows (bulk-transfer connections will not); so cwnd validation is needed to prevent abrupt rate increases.

The TCP receive window can also interact badly with multipath connections. If two subflows share the receive window, and the slow subflow has one packet in its congestion window while the other has the remaining packets, then the loss of the packet on the slow subflow will cause the faster subflow to stall because the receive window fills up. This situation calls for a more aggressive movement of traffic away from congested paths than the theory suggests, which in turn could cause instability problems.

There is an issue of when to start additional subflows. TCP's ack clock really needs four packets in flight to be able to fast retransmit, so there is little point in starting subflows for very short transfers—unless resilience is critical in which case multiple paths might be used to send multiple redundant copies.

What traffic mixes support resource pooling? Very short flows cannot respond to congestion and so cannot help with resource pooling. Do long-lived flows make up enough traffic that, when they move out of the way of short flows, resource pooling is maintained? Indeed, might there be enough traffic *today* from peer-to-peer applications for resource pooling, if their congestion control and peer selection were done correctly?

Is TCP the right place to support resource pooling? The point of implementing multipath in TCP is that this is a single point of change, with direct access to the congestion control loop where we may implement a well-understood load-balancing mechanism. Is this the right layer, and what are the alternatives?

There may be cases where resource pooling has to be done lower in the stack. For example, if most of the traffic on a link is short flows (mice, no elephants), and each flow is from a different end system, then no end system is present for long enough to learn about the state of the network. In this case only the network has enough information to do load balancing.

On the other hand, applications like BitTorrent and http have an additional degree of flexibility: they can choose from multiple servers. How might we leverage this flexibility, and how should it be coupled with congestion control, in such a way that these applications can best contribute to network-wide resource pooling?

TCP is unsuitable for applications such as VoIP that need precise control over timing. Although multiple paths might improve the robustness of these UDP applications, it also forces them to increase their jitter buffer so that the overall latency is that of the slowest subflow. This may be worse than using just one subflow. The main constraint is that these flows must not continuously switch all their traffic to the lowest latency path—or they risk congesting it, raising its latency, then moving away again in a perpetual oscillation. In the end there is no one-size-fits-all multipath solution. No lower layer can satisfy all applications, and no higher layer is sufficiently generic, which pushes us towards transport-layer resource pooling.

What topologies support resource pooling? Resource pooling requires flows to have access to several routes, and we described a simple mechanism for route choice based on multiple addresses for multihomed end systems. This raises two questions: is this amount of path diversity enough to obtain resource pooling? and what sort of network support might give better options, such as to flows between single-homed end systems?

Some theory suggests it may suffice to give a small amount of choice, e.g. two paths per flow, to achieve resource pooling through load balancing[14]. However this theory requires that the choices be random; if they are non-random then many more choices may be needed. In the Internet, the available paths for multipath routing will be conditioned by the network topology. Are they so non-random that the benefits of resource pooling will very be limited?

If multiple addresses do not give good enough path diversity, what are the alternatives? For example end-systems could set a DiffServ codepoint, and routers could use this to select one of several routing trees. How then should the routing system choose routes—for example, is it more important to offer short routes or to offer disjoint routes? Where is it more important to give diversity, at the edge or in the core?

How can we quantify the benefits? The questions we have asked about topology and traffic mix need some way to measure the effectiveness of resource pooling. What are good metrics? One possibility is to measure how well the

[1] To see how hard it is to work without a cumulative ack, see [15]

network can cope with traffic surges. For example, we could define the surge factor $s(x)$ to be the amount by which the traffic matrix has to be scaled down in order to accomodate a surge of x Mb/s on some specific source-destination pair. The better the network is at resource pooling, the more easily it can accomodate a surge, and the smaller $s(x)$.

What is the impact on BGP? Multipath TCP will mean that there is automatic traffic engineering by the end systems, so network operators will have less need to use BGP more-specific prefixes, and this should relieve the strain on routing tables, both in size and in churn rate. There needs to be further quantitative study of whether these claimed benefits will actually materialize.

What are the economic consequences? Consider a multihomed site using multipath TCP: its operator can see in detail how congested the paths are through each ISP, and may choose to terminate the contract with the most congested. This should foster competition.

ISPs may offer different levels of service (different RTT, different level of congestion), and multipath clients could move their traffic to whichever ISP suits their application needs. In this way, clients have a simple mechanism for revealing their quality-of-service preferences, which might foster a differentiated market.

Might the benefits of competition and service differentiation be felt throughout the Internet, not just at the first-hop ISP? Multipath-capable hosts will use paths based on end-to-end performance, which hints that there might be network-wide economic consequences.

5 Conclusion

Aspects of resource pooling are and always have been fundamental part of the Internet. But an unprincipled approach and a failure to recognize the generality of the concept have resulted in a myriad of piecemeal solutions that don't work together. We have argued that by harnessing the responsiveness of end systems in the most generic way possible—by coupling congestion control with multipath routing—the Internet will have a simple, flexible and powerful mechanism for resource pooling.

This proposal is not the last word in resource pooling. There are likely to be cases where better results can be obtained by involving the application layer, and cases which call for load-shifting functionality to be built into the network. We believe it is time for a new architectural principle, extrapolated from what we have seen of resource pooling in the Internet, to guide future development. We call it the *resource pooling principle*:

Definition. Resource pooling means making a collection of networked resources behave as though they make up a single pooled resource. The general method of resource pooling is to build mechanisms for shifting load between various parts of the network

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 where resources are distributed, we should assume that parties will find ways to achieve robustness, flexibility and high utilization by pooling those resources.

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

Corollary. Network architects should consider how their designs create new resources which might be pooled, or create new ways to pool existing resources. They should ensure that there are ways to achieve resource pooling which do not cause problems in other systems in the network.

Acknowledgements

We are grateful for helpful discussions with Iljitsch van Beijnum, Bob Briscoe, Costas Courcoubetis, Philip Eardley, Lars Eggert, Robert Hancock, Costin Raiciu, and others in the EU-funded Trilogy project; also with Mike Harrison, Frank Kelly, and Devavrat Shah.

References

[1] D. Acemoglu, R. Johari, and A. Ozdaglar. Partially optimal routing. *IEEE Journal of selected areas in communications*, 2007.
[2] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P users cooperate for improved performance? *CCR*, 2007.
[3] A. de la Oliva, M. Bagnulo, A. Garcia-Martinez, and I. Soto. Performance analysis of the REAchability Protocol for IPv6 Multihoming. In *Proc. NEW2AN*, 2007.
[4] Y. Dong, D. Wang, N. Pissinou, and J. Wang. Multi-path load balancing in transport layer. In *Proc. 3rd EuroNGI Conference on Next Generation Internet Networks*, 2007.
[5] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure. Achieving sub-second IGP convergence in large IP networks. *CCR*, 2005.
[6] R. Gibbens, F. Kelly, and S. Turner. Dynamic routing in multiparented networks. *IEEE/ACM Trans. Networking*, 1993.
[7] H. Han, S. Shakkottai, C.V. Hollot, R. Srikant, and D. Towsley. Overlay TCP for multi-path routing and congestion control. *IEEE/ACM Trans. Networking*, 2006.
[8] H.-Y. Hsieh and R. Sivakumar. pTCP: An end-to-end transport layer protocol for striped connections. In *Proc. ICNP*, 2002.
[9] C. Huitema. Multi-homed TCP. Internet draft, IETF, 1995.
[10] G. Huston. http://bgp.potaroo.net.
[11] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775, 2004.
[12] F. Kelly and T. Voice. Stability of end-to-end algorithms for joint routing and rate control. *CCR*, 35(2), April 2005.
[13] F. P. Kelly. Loss networks. *Annals Appl. Prob.*, 1991.
[14] P. Key, L. Massoulié, and D. Towsley. Path selection and multipath congestion control. In *Proc. IEEE Infocom*, May 2007.
[15] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion control without reliability. *Proc. SIGCOMM*, 2006.
[16] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. *ACM Trans. Networking*, 2001.
[17] C. N. Laws. Resource pooling in queueing networks with dynamic routing. *Adv. Appl. Prob.*, 1992.
[18] K. El Malki and H. Soliman. Simultaneous Bindings for Mobile IPv6 Fast Handovers. Internet draft, IETF, 2005.
[19] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 address allocation and the BGP routing table evolution. *CCR*, 35(1), 2005.
[20] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, 2007.
[21] R. Moskowitz and P. Nikander. Host Identity Protocol (HIP) architecture. RFC 4423, 2006.
[22] K. Rojviboonchai and H. Aida. An evaluation of multi-path transmission control protocol (M/TCP) with robust acknowledgement schemes. *IEICE Trans. Communications*, 2004.
[23] T. Roughgarden and É. Tardos. How bad is selfish routing? *Journal of the ACM*, 2002.