The Resource Pooling Principle

ABSTRACT

Since the ARPAnet, network designers have built localized mechanisms to pool resources, often without understanding the broader implications. We argue that resource pooling is the only practical way to satisfy ever-increasing demands for resilient Internet service. Further we argue that this is best achieved by harnessing the responsiveness of end systems.

1 Introduction

There is a silent revolution that is reshaping the Internet. The seed for this revolution was planted with the original design of statistical multiplexing through packet switching. As demands on the Internet grew, network operators began to use traffic engineering, and sites began to be multihomed. These practices are examples of *resource pooling*—making a network's resources behave like a single pooled resource, as illustrated in Figure 1. The aim is to increase reliability, flexibility and efficiency; and the method is to build mechanisms for shifting load between the various parts of the network. Now end systems are taking a major role in pooling network resources, for example in peer-to-peer applications and in load-balancing by Google and Akamai across globally-distributed server farms.

Each party has implemented its own resource-pooling mechanisms independently. Unfortunately, these can cause stress to other parts of the architecture, and fight against each other. For example, network operators can no longer operate their networks as if traffic is open-loop, because the traffic matrix reacts on the timescale of a few RTTs. But if resource-pooling end systems could balance their load across multiple paths in the right way—the right reaction to the right congestion signals from the network—then traffic would quickly move away from congested links in favor of uncongested links, which is the goal of traffic engineering.

We believe the Internet needs a new architectural principle, in the same mould as the end-to-end 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 the these mechanisms do not conflict with each other.* This paper is a call to arms for a new way of thinking about routing and congestion control, based on the resource pooling principle. We believe that the natural evolution of the Internet is to harness the responsiveness of multipath-capable end systems, to solve the problems and limitations of the current piecemeal approaches to resource pooling.

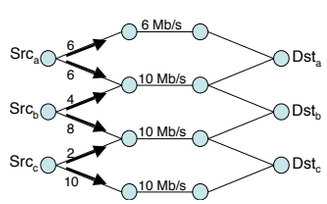

Figure 1. Pooling capacity. Even though no flow has access to all four links, the entire 36 Mb/s capacity can be shared equally, and each multipath flow achieves more than it could over a single path.

The key benefit of resource pooling is robustness, but there are many other benefits too; for example, wireless mobile devices would see particular advantages. Such devices frequently have multiple radios (GPRS, WiFi, bluetooth, etc.) but each individual channel is intermittent and has variable capacity. If traffic could rapidly be shifted between the different channels, depending on RF conditions, then the device would have faster and more reliable overall performance.

In this paper we describe the attempts to-date to do resource pooling (Section 2) and the reasons for believing it is of such great importance. We will then explain the motivations for implementing resource pooling in a particular general way at the transport layer (Section 3), as opposed to other possible alternatives. Finally we will attempt to outline a research agenda of questions that need to be answered before we can fully benefit from resource pooling, and metrics by which we might consider architectural changes to be a success. Some parts of the solution are well understood; in the last few years researchers have developed models of how systems might respond, 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 a great deal of work.

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

Packet switching is the most fundamental concept in the Internet architecture. Computer communication is inherently bursty; the ideal service would transfer finite

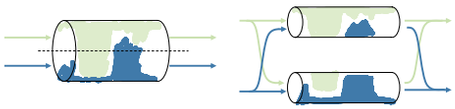

Figure 2. Pooling capacity (a) between "circuits" on a single link, and (b) across links. In both cases pooling improves the ability to cope with bursts or unexpected traffic patterns; the latter case achieves robustness to link failure.

sized files in zero time. 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.

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.

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

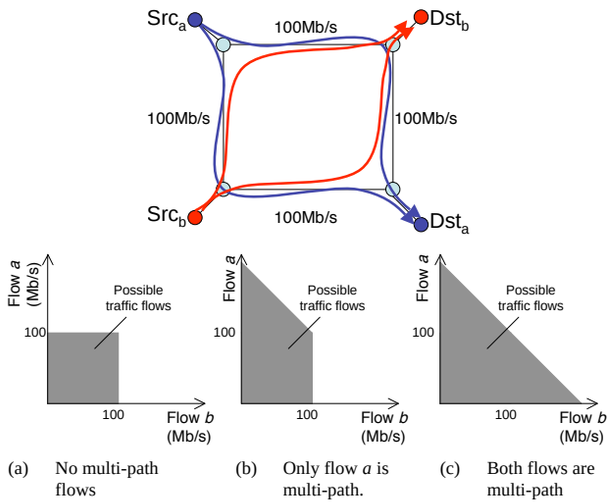

Figure 3. Resource Pooling increases the set of traffic matrices that can be served.

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.

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 [9, 11].

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 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[**?**], with a growth rate that is worrying ISPs and router vendors[13]. About half the entries are for so-called more-specific prefixes[12]; 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[12], 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[12]. 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 [10]. A single ISP can tweak OSPF to give faster reconvergence [5], and MPLS 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[14]. 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].

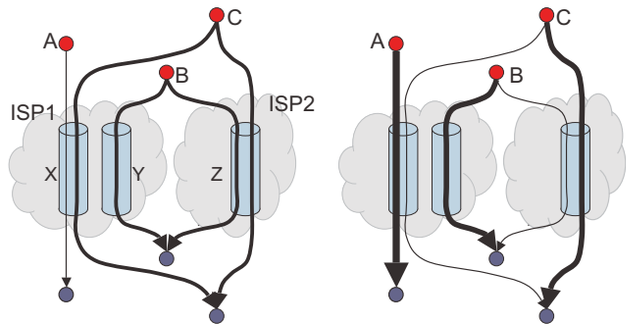

Figure 4. Traffic engineering works better when end systems and network operators cooperate. If there is a surge in traffic on route A, and if the other flows B, and C use multiple paths, then they could (with the right feedback from the network) re-balance themselves so that they do not use the congested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

Failure is not the only cause of path change. Current mobility mechanisms [7, 14] 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[8] 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 [**?**]. 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 [**?**, **?**].

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.

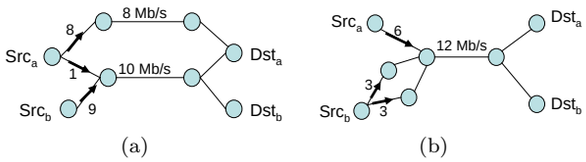

Figure 5. Fair rate allocations. These allocations would not be achieved by running TCP independently on each subflow.

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 [**?**, **?**]. 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 [**?**, **?**, **?**, **?**], 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 delivered to the destination via the remaining paths.

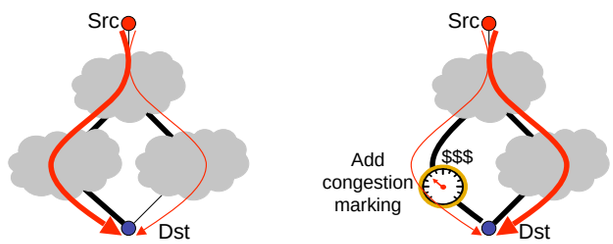

Figure 6. Resource pooling can load balance between links used for multihoming without the need for more specific routing prefixes.

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 [11, 9]. 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 when a subflow fails because, unlike bulk-transfer connections, they often will have spare window available on the remaining subflows resulting in abrupt rate increases unless cwnd validation is done.

The TCP receive window can also interact badly with multipath connections. If two subflows share the receive window such that one of them has one packet and the other has the remaining packets in its congestion window, then the loss of the packet on the slow subflow will cause the faster subflow to stall because the receive window fills up. Under such circumstances this argues for a more aggressive movement of traffic away from the 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

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

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?

There is theoretical support for the idea that it is enough to give a small amount of choice, e.g. two paths per flow, to achieve resource pooling through load balancing[**?**]. 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 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

Resource pooling is and always has been a fundamental requirement of the Internet. But an unprincipled approach has resulted in a myriad of poorly conceived attempts to provide the needed functionality, and we need a new principle by which to judge these attempts. Our argument can thus be briefly summarized as:

Observation 1: Resource pooling is 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: 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.

References

[1] L.A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2), March 2003.

[2] David D. Clark. The design philosophy of the DARPA Internet protocols. *ACM Computer Communication Review*, 1988.

[3] A. de la Oliva, M. Bagnulo, A. Garcia-Martinez, and I. Soto. Performance analysis of the REAchability Protocol for IPv6 Multihoming. In *Proc. Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN)*, September 2007.

[4] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 5(5), September 2000.

[5] P. Francois, C. Filsfils, John Evans, and Olivier Bonaventure. Achieving sub-second IGP convergence in large IP networks. *ACM SIGCOMM Computer Communication Review*, 35(3), July 2005.

[6] Richard J. Gibbens, Frank P. Kelly, and Stephen R. E. Turner. Dynamic routing in multiparented networks. *IEEE/ACM Transactions on Networking*, 1993.

[7] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[8] H. Soliman K. El Malki. Simultaneous Bindings for Mobile IPv6 Fast Handovers. Internet-Draft draft-elmalki-mobileip-bicasting-v6-06, Internet Engineering Task Force, July 2005. Work in progress.

[9] F. P. Kelly. Loss networks. *The Annals of Applied Probability*, 1(3), August 1991.

[10] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. *ACM Transactions on Networking*, 9(3), jun 2001.

[11] C. N. Laws. Resource pooling in queueing networks with dynamic routing. *Advances in Applied Probability*, 24(3), September 1992.

[12] Xiaoqiao Meng, Zhiguo Xu, Beichuan Zhang, Geoff Huston, Songwu Lu, and Lixia Zhang. Ipv4 address allocation and the bgp routing table evolution. *SIGCOMM Comput. Commun. Rev.*, 35(1):71–80, 2005.

[13] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984 (Informational), September 2007.

[14] R. Moskowitz and P. Nikander. Host identity protocol (hip) architecture. RFC 4423, Internet Engineering Task Force, May 2006.