Principles for End-to-End Failure Masking

David G. Andersen, Rohit N. Rao
MIT Laboratory for Computer Science
{dga,rohit}@lcs.mit.edu

Abstract

Outages, long-term congestion, and misconfigurations plague Internet service availability. This paper argues that an end-to-end recovery approach augments traditional approaches such as BGP multihoming. The end-to-end approach is effective even when other solutions cannot be used. We present five design principles for protocols and applications that vastly increase their resilience against network failures. We demonstrate the utility of this approach using a resilient Web proxy, in day-to-day use at our lab, which provides highly reliable Web access.

1 Introduction

While Internet access speeds and the number of connected hosts have increased at dramatic rates, the end-to-end availability of services from clients has hardly improved [3, 9, 14]. As one example, our lab at MIT (LCS), a rather network-savvy and sizable organization, was disconnected from the Internet five times during June 2003 for a total of about seven hours. This experience is common; many studies have shown that client-service availability is often under 99%. Network outages, persistent congestion, and misconfigurations are among the dominant causes of low availability.

Service providers traditionally improve availability by deploying additional network links between sites, employing routing protocols like OSPF to take advantage of this redundancy. Customers as well are increasingly adding redundant Internet links, because many failures are believed to occur near the edges of the network. Larger customers traditionally use BGP multihoming to take advantage of link redundancy.

Peer-to-peer networks are often targeted at lower-end users who don't have the resources for BGP multihoming. Even when they can, end-to-end failures persist despite the use of many of these network-layer techniques: the failures for LCS come from a BGP-multihomed environment. End-to-end reachability is more than the sum of functioning links and routing. Equipment failures, firewall (mis)configurations, and deliberate rate-limiting can cause one application's traffic to be dropped while the rest of the network appears to function perfectly. Ultimately, *the only reliable test of whether an Internet path will work for a particular application is to attempt to use the path.* This observation is a direct application of the end-to-end argument [18]. While overlay-based techniques like RON improve end-to-end availability [1], existing techniques do not seem to scale to the sizes required for peer to peer networks.

When sites are multihomed, such that they have path redundancy in reaching the rest of the Internet, can end-to-end techniques the Internet, can end-to-end techniques improve client-service without sophisticated network routing techniques?

The right set of end-to-end techniques for solving this problem is not obvious—if indeed they exist at all. Solving the problem, however, could improve peer to peer systems' resilience to a large class of failures that occur near network edges. In conjunction with good network routing, these techniques could considerably improve availability.

Using an end-to-end approach, we designed a resilient Web proxy that uses a "backup" commercial DSL link to recover from failures of MIT's main Internet connection. From our experience with this system, we articulate five design principles for improving end-to-end availability. Because the principles focus on reachability between end-systems, we believe they apply equally to peer to peer systems and client-server systems. The principles capture common design wisdom, but many systems omit one or more of them.

The five principles are:

(i) Explicitly advertise multihoming and/or multiple servers—even if also using BGP.
(ii) Let applications choose which network link to use.
(iii) Support session resumption.
(iv) Maximize independence between naming and service.
(v) Redundantly transmit selected packets to avoid exceptional delays.

In the rest of this paper, we discuss the goals of our work (Section 2), our experience operating a resilient Web proxy (Section 3), how our principles can be used to engineer better end-system protocols (Section 4), and outline a research agenda that might help realize the potential gains suggested by our approach (Section 5).

2 Goals

The specific goals this work hopes to meet are:

- Mask last-mile link failures

- Enable local and wide-area fail-over
- Provide faster fail-over than that provided by BGP
- Allow running connections to fail-over to new links without interruption (“zero-click fail-over”)
- Provide these features with no network support

These goals are motivated in part by three real-world network scenarios. As Internet connectivity becomes cheaper, using multiple network links becomes more attractive for both network administrators and researchers. In many situations, however, it is difficult to effectively use these links.

Multiple home Internet links. Many students at LCS have multiple Internet links. Most students have a high-speed DSL or cable modem link, and can either reach a neighbor’s network through 802.11 wireless, or are participating in the LCS Grid Roofnet project. Often, neither connection is particularly reliable, and the time-to-repair for residential links is quite high. These users would benefit from arranging “peering” arrangements with their neighbors such that they use each other’s networks if their own fails. These users might often share excess capacity with their neighbors.

Adding Internet Connections. As organizations grow, they may wish to add new Internet connections without renumbering or giving up their old IP addresses. For example, LCS is getting a fast new Internet connection separate from MIT’s, but MIT policy prohibits labs from announcing their subnets via BGP. Users don’t want to give up their MIT addresses, but would like to load balance and fail-over to the new connection easily. The lab uses BGP to route over this connection, but external policies render BGP fail-over impossible without renumbering.

Cheap Backup Links. The lab has a DSL line that often remains up when its other network connections fail. We would like our services to transparently fail-over to the DSL line when necessary, but given our large traffic volume, the DSL line isn’t useful for load balancing.

3 Case Study

Several current systems achieve some of the foregoing goals, but none achieves them all. Most of the shortcomings could be corrected relatively easily, though they may require changes to both client and server software. To understand how a system might achieve these goals, we present one detailed example of a resilient Web proxy that improves Web access, and briefly examine other systems here and in Section 6 to see how they meet our goals.

3.1 Resilient Web Proxy

We have deployed a modified version of the Squid cache based on the five principles from Section 1. The proxy has been in day-to-day use at our lab for almost three months.

The machine on which the proxy runs has two IP addresses, one assigned by MIT, and one assigned by the DSL provider. FreeBSD `ipfw` firewall rules direct all packets originating from its MIT address to the MIT router, and packets from the DSL address to the DSL router. This lets the proxy choose the network link over which it sends packets.

DNS handling. Most domains are configured with at least two name servers, and the root-level servers have 13. To perform DNS queries, the proxy selects a random nameserver to contact using each link, and returns the first reply it receives. This permits the proxy to avoid either the failure of one of its links or the failure of a DNS server with no additional delay. The DNS proxy is based on the `dnscache` program.

HTTP handling. For HTTP requests, the proxy attempts to open one TCP connection over each of its outbound links, rotating through the list of server addresses returned by the DNS. The proxy uses the first connection that returns, and immediately closes the other. Like the DNS handling, this redundancy allows rapid masking of either client-link failures or a single server-link failure if the server is multihomed and advertises its redundancy.

The proxy provides all of our goals except fail-over of running connections. Supporting fail-over requires changes to the server, and can be accomplished through the use of TCP migrate [20], SCTP [4] or other connection-migration mechanisms. We plan to implement one or more of these mechanisms in the future, but, fortunately, most Web connections are relatively short, so a failure mid-transfer is less likely than with other protocols.

3.2 Evaluation

The proxy has weathered eleven major network outages, for nearly ten hours of downtime, and was able to provide service during all but the first—when we realized we had not implemented a reliable DNS server.

The proxy has served 2.3 million requests, and has established about 630,000 TCP connections while doing so. It has fetched about 4GB of useful data from the network, and served 5% more bytes via caching. The dual-open approach has transmitted about 36 megabytes of “extra” SYN and FIN/RST packets. The DNS server appears to have added an additional 10 megabytes or so, but this number may be overstated because of an early bug in our DNS server. The total overhead from duplicating requests has been about 1%. 86% of HTTP requests have been fulfilled over the MITnet network, but this number is biased by user behavior—more users use the proxy when MIT’s network is behaving poorly.

Table 1 summarizes the improvements from the proxy over a two day period in July. Aside from avoiding last-mile outages, the most significant benefit of the proxy was to reduce the effective loss rate for DNS packets and SYN packets. Since a loss of either of these packets results in a 3+ second

Metric	MIT	DSL	Both
11,833 DNS lookups			
Lost DNS	4.5%	4%	1.3%
Slow DNS (> 250ms)	1.8%	6.3%	0.5%
DNS RTT	91ms	142ms	70ms
22,141 TCP connections			
Lost SYN	0.33%	0.10%	0%
Unreachable Server	2.3%	1.5%	1.3%
SYN RTT	52ms	74ms	49ms

Table 1: Loss rates and round-trip times for requests handled by the resilient proxy. Lost SYNs count only connections where the SYN ACK was eventually received.

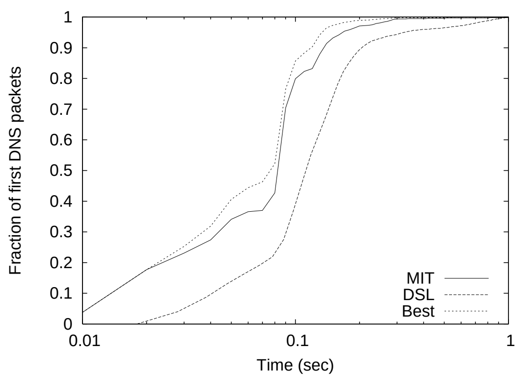

Figure 1: CDF of RTTs for DNS packets with and without the proxy. Many of the low-RTT samples are for local MIT name resolutions, which the proxy cannot speed up.

timeout, masking these losses avoids imposing excessive delays on the user.

Figures 1 and 2 show the distribution of response times for packets that were successfully received. (Due to caching in the proxy, we can only determine the response time for the first query packet sent while resolving a given name.) We believe that the greater improvement in DNS reachability and round-trip times is a result of better network dispersity of DNS servers vs. Web servers. Our measurements probably *undercount* TCP problems, because TCP connection initiation depends on successful DNS resolution. If both DNS and TCP fail, we observe only the DNS failure.

4 Principles

Based on our experience with the reliable proxy and other protocols (e.g., DNS and SFS), we present five design principles that facilitate end-to-end reliability. While many of these principles seem common-sense, most systems do not incorporate all of them.

Explicitly advertise server multihoming. Service providers should explicitly announce some of their re-

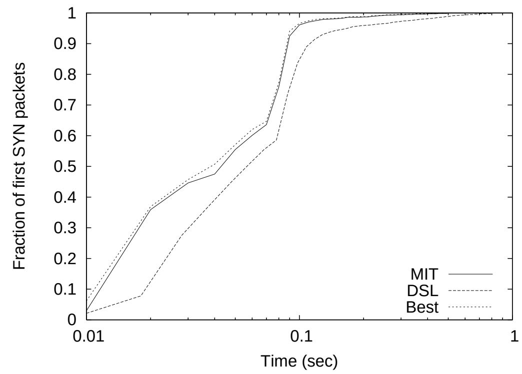

Figure 2: Round-trip times for connections with no lost SYN packets.

dundancy by making multiple addresses available per service name, instead of hiding them all behind network-layer multihoming. These addresses can be one host through multiple networks, or multiple hosts through multiple networks if application constraints permit. This advertising gives clients a chance to make a server choice based on end-to-end reachability. Systems with explicit advertisements include DNS's multiple NS records and MX records for SMTP [16]. Between multiple MX hosts and application-level retransmission of messages, it is no surprise that SMTP is a highly resilient system. Similarly, many peer-to-peer systems implicitly perform these tests when attempting to connect to another peer, but explicit testing of multiple addresses would provide individual peers with enhanced connectivity.

Announcing multiple DNS address records permits fail-over and simple load-balancing, but it is insufficient to express policies such as "back-up link." While trusting clients to always obey such suggestions is foolish, there are many cases in which these hints from the server benefit the client. These hints are often implemented via dynamically updated short-TTL DNS replies, but this technique conflicts with our fourth design principle by tightly coupling service availability with naming availability, as discussed below.

The naming service should also distinguish between different servers and different addresses for the same server for applications in which all requests should go to the same server.

The proxy applies this principle by attempting to connect to two of the advertised addresses for a website at a time, cycling through the entire list.

Let clients choose which network link to use. Source routing is often disabled for security or performance reasons, and overlay routing requires an overlay. One practical way to accomplish this—used in the resilient Web proxy—is to assign multiple IP addresses to clients and use policy routing to send outbound packets down different paths depending on their source address. This strategy fits the tactics available to small multihomed clients, and aligns with the host-based multihoming suggested for IPv6 [10] and SCTP [4].

Support session resumption. Clients should be able to disconnect from the network, change their attachment point, and resume their session as they left off [21]. Minimizing connection state, as in the DNS and HTTP protocols, facilitates this part of the design—but HTTP itself serves as a warning of the dangers of assuming transactions will always be short and idempotent, as more long-running services such as audio playback are layered atop it. Even small requests can be problematic, as numerous Web pages that instruct "Do not double click or you may be charged twice!" testify.

Resumption can be strengthened if *content* is also decoupled from network location, as with Content Delivery Networks, but is not always possible.

The SFS [12] filesystem transparently reconnects its TCP connections if they break, performing a new lookup of the remote host address. Because it uses NFS, some of SFS's operations can be erroneously replayed during session re-establishment. The Web proxy relies on HTTP's small state, but should be improved as noted in Section 3.

Maximize independence between naming and service. Most Internet services rely on the DNS for naming, and as a result, many clever systems dynamically update the DNS to provide service fail-over, load balancing [19], client redirection [23], and so on. The danger of these tricks is that they typically use short-TTL DNS responses. In the limit, these tactics require one DNS query per connection, a function better performed by session initiation packets.

If DNS records can be extensively cached and replicated, transient DNS failures will have a much smaller effect on service availability. DNS failures are unfortunately frequent, even for large companies; e.g., the 2001 DNS failure that rendered Microsoft's website inaccessible [6]. Long-lived naming allows caching clients to plan for name server failures.

While this dependency is reduced through careful engineering for highly popular sites such as those provided by Akamai [8], the goal of a system designer should be to *increase* the cacheability of their name lookups, not to minimize it. Caching improves reliability and interactivity; even a single lost DNS query can add measurable overhead to the delay a user experiences browsing a Web page [5].

By explicitly and in parallel attempting to contact multiple addresses for a service, the resilient Web proxy allows service providers to implement fail-over and load balancing without resorting to uncacheable DNS records.

Redundantly transmit selected packets to avoid exceptional delays. Bandwidth becomes cheaper over time, but the speed of light doesn't change much. Clients and servers should judiciously duplicate some data—such as SYN packets and DNS lookups—to avoid multiple-RTT timeouts with high user impact.

Kademlia's system-wide concurrency parameter [11] exemplifies this principle. Its use of parallel queries during query resolution both avoids failed or delayed nodes and provides the system with a pool of performance data with which to better route future queries. The resilient Web proxy initiates two TCP connections in parallel, using only the one that returns first. This greatly reduces the impact of lost SYNs or failed servers (Figure 1). Similarly, it performs low-overhead DNS resolutions in parallel to reduce the impact of losing these critical packets.

5 Implementation Challenges

From our experience with the resilient Web proxy, we have identified a few systems and network challenges that stand between good design and implementation.

The programmer overhead of most of these modifications is not large. Web browsers already initiate multiple connections to Web servers for content retrieval, and peer to peer clients already support asynchrony in client connections. Implementing the changes in our Squid proxy and DNS resolver required the addition of under 100 lines of code in each. Three major challenges that remain are:

Legacy transport protocols. A complete implementation of our principles requires application support for reconnection (à la SFS) or an improved transport protocol like TCP Migrate [20] or SCTP [4]. Without these we cannot achieve zero-click failover. Improved transport alone, however, is not sufficient to achieve our reliability goals, and we wish to provide many of our benefits with existing transport protocols.

Applications must control initiation. Applications, with the help of the OS, must be more involved in the initiation of network connections to cope appropriately with connection failures. As an example of why an enhanced connection interface is necessary, consider the standard Berkeley Sockets `connect` system call. Connect contacts a single server, and either succeeds or fails. TCP retransmits SYN packets with an initial conservative timer of 3 seconds [15]. This inflexible interface forces programmers to either waste (considerable) user time, or open connections too aggressively.

Servers must accept overhead. Sending extra connection initiation packets raises the possibility of unacceptable server overhead. Fortunately, the trend in today's server designs is to minimize the cost of session initiation; this trend bodes well for redundancy schemes that rely on duplication of these packets. Techniques like SYN Cookies [2] reduce the overhead of processing connection initiation packets. DNS servers can handle hundreds to thousands of queries per second, and the service is so easily replicated that even handling the DNS for a highly popular site is relatively easy.

6 Related Work

Several studies and products demonstrate that providing clients with a choice of paths to the server increases their reliability. Yip extended the indirect routing ideas of RON [1] to fetching Web pages indirectly through a third host [24], and Akamai's Akarouting [13] behaves similarly. RouteScience [17] uses end-to-end measurements to select outbound routes for networks with multiple Internet links.

Commercial products like Stonesoft's "Multi-Link Technology" implement one technique we study, sending multiple outbound SYN packets to a server [22]. As we've shown, these products would be even more effective with wider use of the design principles this paper encourages.

The Smart Clients approach argues that load balancing and server selection should be implemented in the clients for better flexibility and performance [25]. The major difference between this work and ours is that we believe naming should be static and cacheable. Smart Clients, in contrast, moves server selection into its name resolver. Dykes et al.'s comparison study of server selection [7] concludes that picking the first Web server to respond to a SYN is among the most effective tactics, and can help reduce pathological delays by 7% to 20% over random selection.

Content Distribution Networks (CDNs) like Akamai [23] improve the reliability and speed of content delivery by locating caches nearer to clients. They do rely on the DNS to direct clients to servers (Section 4), but in all other ways, our principles facilitate the job of a CDN by helping to improve the last-mile connection between the client and the CDN cache, and between the caches and the origin servers.

7 Conclusion

Actual checks of client-server reachability provide an end-to-end check that the path provided by network routing protocols is actually suitable for an application's use. Enabling applications to choose between multiple paths and servers gives them the ability to pick the best path from the application perspective, avoiding network failures that interrupt end-to-end connectivity.

To allow clients to make this determination, this paper presented five design principles that together can improve client-service availability using only end-to-end techniques. These techniques can be used even when sophisticated routing techniques cannot, but also work in concert with improvements to the underlying link and routing technologies to provide improved service availability.

References

[1] ANDERSEN, D. G., BALAKRISHNAN, H., KAASHOEK, M. F., AND MORRIS, R. Resilient Overlay Networks. In *Proc. 18th ACM SOSP* (Banff, Canada, Oct. 2001), pp. 131–145.

[2] SYN Cookies. http://cr.yp.to/syncookies.html, 1996.

[3] CHANDRA, B., DAHLIN, M., GAO, L., AND NAYATE, A. End-to-end WAN Service Availability. In *Proc. 3rd USITS* (San Francisco, CA, Mar. 2001), pp. 97–108.

[4] COENE, L. Multihoming issues in the Stream Control Transmission Protocol. http://www.ietf.org/internet-drafts/draft-coene-sctp-multihome-04.txt, June 2003.

[5] COHEN, E., AND KAPLAN, H. Prefetching the means for document transfer: A new approach for reducing web latency. In *INFOCOM (2)* (2000), pp. 854–863.

[6] DELIO, M. Microsoft's web still tangled. *Wired News* (Jan. 2001). http://www.wired.com/news/business/0,1367,41423,00.html.

[7] DYKES, S. G., ROBBINS, K. A., AND JEFFERY, C. L. An empirical evaluation of client-side server selection algorithms. In *INFOCOM (3)* (2000), pp. 1361–1370.

[8] JUNG, J., SIT, E., BALAKRISHNAN, H., AND MORRIS, R. DNS Performance and the Effectiveness of Caching. In *Proc. Internet Measurement Workshop* (Nov. 2001).

[9] LABOVITZ, C., AHUJA, A., AND JAHANIAN, F. Experimental study of Internet stability and wide-area network failures. In *Proc. FTCS* (Madison, WI, June 1999).

[10] LINDQVIST, K. Multihoming road-map for IPv6. http://www.ietf.org/internet-drafts/draft-kurtis-multi6-roadmap-00.txt, Feb. 2003.

[11] MAYMOUNKOV, P., AND MAZIÈRES, D. Kademlia: A peer-to-peer information system based on the XOR metric. In *Proc. IPTPS* (Mar. 2002).

[12] MAZIÈRES, D., KAMINSKY, M., KAASHOEK, M. F., AND WITCHEL, E. Separating key management from file system security. In *Proc. SOSP* (December 1999), pp. 124–139.

[13] MILLER, G. Overlay routing networks (Akarouting). `http://www-math.mit.edu/~steng/18.996/lecture9.ps`, Apr. 2002.

[14] PAXSON, V. End-to-End Internet Packet Dynamics. In *Proc. ACM SIGCOMM* (Cannes, France, Sept. 1997), pp. 139–152.

[15] PAXSON, V., AND ALLMAN, M. *Computing TCP's Retransmission Timer*. Internet Engineering Task Force, Nov. 2000. RFC 2988.

[16] POSTEL, J. B. *Simple Mail Transfer Protocol*. Internet Engineering Task Force, August 1982. RFC 821.

[17] RouteScience. Whitepaper available from `http://www.routescience.com/technology/tec_whitepaper.html`.

[18] SALTZER, J., REED, D., AND CLARK, D. End-to-end Arguments in System Design. *ACM Transactions on Computer Systems 2* (November 1984), 277–288.

[19] SCHEMERS, R. J. lbnamed: A Load Balancing Name Server in Perl. In *LISA IX* (Monterey, CA, Sept. 1995).

[20] SNOEREN, A. C., AND BALAKRISHNAN, H. An end-to-end approach to host mobility. In *Proc. ACM Mobicom* (Boston, MA, Aug. 2000), pp. 155–166.

[21] SNOEREN, A. C., BALAKRISHNAN, H., AND KAASHOEK, M. F. Reconsidering Internet Mobility. In *Proc. HotOS VIII* (Schloss-Elmau, Germany, May 2001).

[22] STONESOFT. Multi-link technology white paper. `http://www.stonesoft.com/files/products/StoneGate/SG_Multi-Link_Technol%ogy_Whitepaper.pdf`, Oct. 2001.

[23] Akamai. http://www.akamai.com, 1999.

[24] YIP, A. NATRON: Overlay routing to oblivious destinations. Master's thesis, MIT, Aug. 2002.

[25] YOSHIKAWA, C., CHUN, B., EASTHAM, P., VAHDAT, A., ANDERSON, T., AND CULLER, D. Using smart clients to build scalable services. In *Proc. USENIX Technical Conference* (Jan. 1997).