Improving the Availability and Performance of Network-Mediated Services

Geoff Carpenter (gcc@watson.ibm.com)

Germán Goldszmidt (gsg@watson.ibm.com)

IBM Research

Abstract

We present a client-based dynamic server switching method that improves the availability and performance of network mediated applications. Narwhal provides a local (client-resident) intermediary broker, that can intelligently route the traffic among intermediaries. Its benefits include (1) improved availability of intermediary services, (2) load sharing requests across several intermediaries, (3) bypass intermediaries whenever possible, and (4) remote administrative control enabling the implementation of domain-specific policies to utilize the shared, limited networking resources. For example, interactive data, such as e-commerce traffic can be given higher priority than other non-critical data at the client side. Remote administrative control prevents the "tragedy of the commons" syndrome, where each client tries to locally maximize its private utilization. A shared Narwhal server performs pro-active monitoring and implements a "resource broker". A prototype was implemented for sharing intermediaries that provide SOCKS V4/V5 and HTTP proxy interfaces. Experimental results show that Narwhal provides improved and more consistent performance.

Table of Contents

1. Introduction

Network intermediaries are computational elements that lie along the path of networked interactions. Some intermediaries are generic, and are used by many different applications, while others serve some very specific purposes. Examples of the former are proxy caches, which are used to address generic network performance considerations. Other proxies provide personalized services, such as data anonymity, (e.g., [LPWA]). Application-specific intermediaries have been used for many purposes, such as image distillation, targeted advertising, and web site advising. Several systems (e.g., HTTP Stream Transducers [ASPSHST] and WBI [WBI]) provide toolkits to assist in building such intermediaries. It is our conjecture that the use of both types of network intermediaries for TCP/IP applications will continue to increase, as they provide useful auxiliary services.

Already, many networked applications must access intermediaries in order to reach remote services. For example, most corporations establish security firewalls [CB94] to protect their internal networks from unauthorized access (e.g., from Internet sites). At the same time, internal users need to "cross" these security firewalls to access remote services (e.g., to access public search engines). One common way to enable crossing these firewall boundaries is for client applications (e.g. Web browsers) to access remote services indirectly, via SOCKS servers [SOCKS]. These servers are trusted intermediaries enabled to communicate across the firewalls boundaries with remote Internet services.

Typically, client applications that access remote services across the firewalls are statically configured to use only one such intermediary (e.g., a given SOCKS-server, S1). This can be configured separately for each application or by configuring a common library. For example, the former can be done by inserting the name of the SOCKS server "S1" in the proper field of a browser configuration menu. Installing and configuring a generic library, such as [SOCKSCAP], can do the latter.

In either case, these applications become dependent on the performance and availability of the given intermediary. For instance, if S1 fails, all the client applications using S1 will be unable to communicate with any remote server outside the firewalls. From the end-user perspective, it appears that connectivity to the Internet has been lost. In fact, that is not necessarily the case, since an alternative SOCKS server (S2) may provide the equivalent service at the same time. However, end-users may not understand how to change the bindings to these alternate intermediaries or may not be authorized to do so. Furthermore, manually changing these bindings upon the vicissitudes of an intermediary would be error-prone, inefficient, and tedious.

The static allocation of intermediaries often results in poorly balanced workloads, consequently wasting network resources. For example, one SOCKS server can have a large number of clients simultaneously connecting to it while others are serving only a few clients. The workload across the intermediaries can become skewed and highly unbalanced, potentially causing significant performance degradations due to server overloads. In the event that an intermediary server fails or is overloaded, there is no automatic mechanism for redirecting a client to another intermediary server. Ideally, client applications would be able to automatically switch to an alternate intermediate server in order to receive remote services without interruption in case of service degradation or failure of a network intermediary.

A second problem faced by these client applications is that many of their communications are often unnecessarily being routed via intermediaries. Consider the following 2 examples:

A Web browser may be statically configured to access a given SOCKS server for all remote services, including those services that are "inside the firewall". That is, there are services that are local to the network domain and hence do not require access via a SOCKS server, but the client host is not aware of this. As another example, an HTTP request for a dynamic page may be routed to an HTTP cache, increasing the load on the cache without any benefit. Such requests create unnecessary load on the intermediaries, which result in additional delays for all interactions.

A third problem that Narwhal attempts to address is the lack of configurable policies to allocate constrained resources. For example, on a bandwidth-constrained network, the download of a large news file may significantly slow all other interactions via the network. In particular, real-time applications, such as live broadcasts of multimedia streams may be unnecessarily delayed, since the current Internet infrastructure is lacking in effective priority mechanisms. By controlling the pace of requests and acknowledgments of IP packets at the clients, Narwhal provides a mechanism to address this priority problem.

The main goal of Narwhal is to improve the performance and availability of networked applications by dynamically accessing multiple intermediary servers. Each Narwhal-enabled client request is transparently directed to one of the intermediary servers to obtain the requested service. If an intermediary is not functioning as desired, the client request is automatically rerouted to another intermediary.

The rest of this paper is organized as follows: Section 2 describes the capabilities of the Narwhal Client Agent. Section 3 outlines the management of Narwhal components. Section 4 briefly outlines some of the performance improvements that we obtained using Narwhal. Section 5 describes related work and Section 6 concludes.

2. Narwhal Client Agent Capabilities

Narwhal Client Agents (NCAs) provide a rich set of capabilities that improve the performance and reliability of firewall-aware applications. Some of its capabilities are:

- Avoidance of slow or failed intermediaries
- Protocol conversion
- Bypass of intermediaries when appropriate
- Prioritization of traffic based on type
- Use alternative intermediaries based on type of requested data

Avoidance of Poorly Performing Intermediaries

The avoidance of intermediaries that are poorly performing or, in the worst case, completely inoperable, is an obvious benefit provided by NCAs. Wasting time trying to use a slow or failed server when faster alternatives exist is clearly non-optimal, so the point will not be belabored; however, the process of selecting an alternative service provider is interesting. NCAs are typically configured with a collection of intermediaries that provide equivalent functionality. A given collection of functionally equivalent intermediaries is further broken down into tiers of intermediaries with similar performance. The ranking of tiers is important because utilization of a slower server results in a degradation of performance. This is illustrated in the graph of Figure 1 below:

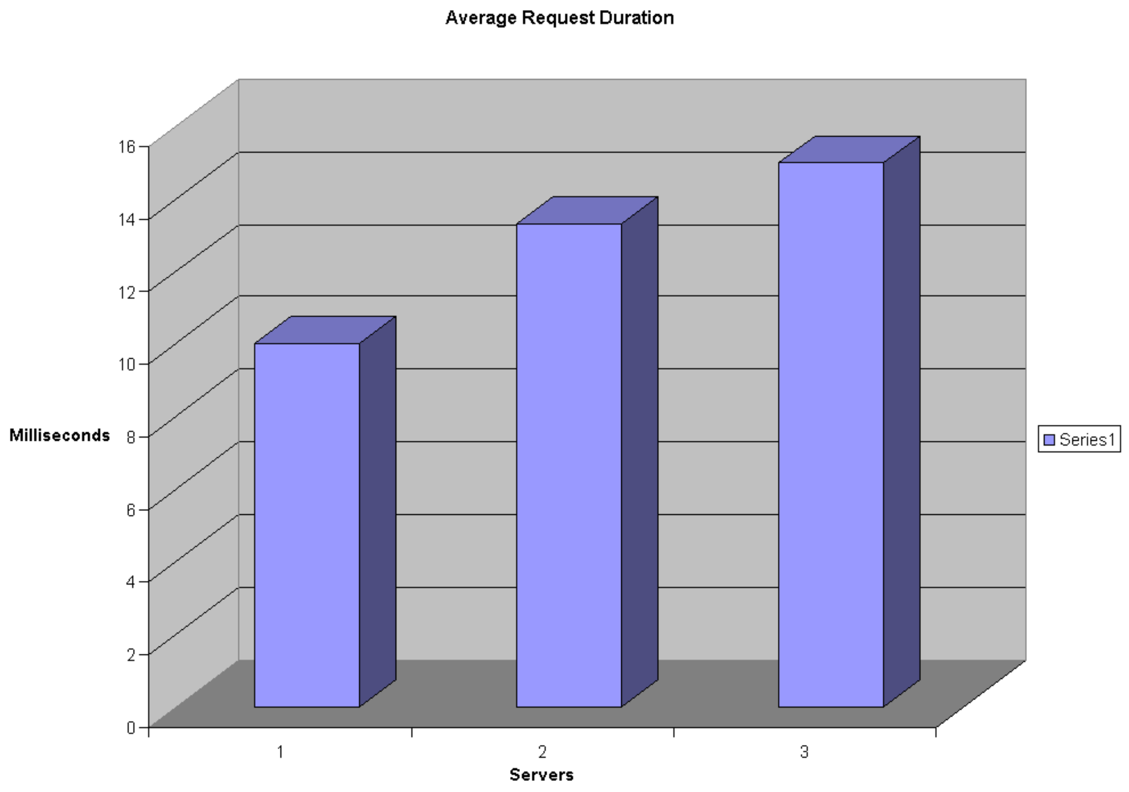


Figure 1

Only intermediaries in the topmost tier are candidates for utilization. NCAs are permitted to autonomously change the tier assignment of an intermediary when a performance-related failure is detected. This alteration in tier ranking results in the NCA avoiding the use of the poorly performing intermediary. When an NCA makes a local determination to alter a tier ranking, it also makes this information available to the Narwhal management applications. The management applications in turn can instruct other NCAs to avoid the use of the indicated intermediary. This proactive action allows a discovery by one NCA to benefit all of its peers.

Bypass of Intermediaries

Many companies have internal networks that are protected from access by outside entities. This is often achieved using a firewall. Applications that need to access content outside the corporate network typically must traverse the firewall using some protocol (e.g., SOCKS, HTTP proxy, etc.). On the other hand, these same applications should not use a firewall if they are accessing content within the internal corporate network as doing so is both inefficient and introduces unproductive load on resource-constrained gateways. Solutions like auto proxy attempt to address this problem in an application-specific fashion.

Traffic Prioritization

Users that are connected to the network using slow dialup links are rarely influenced by operational, yet slow intermediaries as their extremely limited connection bandwidth provides the dominant constraint on their realized performance. NCAs have the ability to prioritize traffic based on its type (as well as other factors). As an example, this permits a file transfer to be given a lower priority than interactive HTTP browsing or telnet sessions. This permits interactive applications to remain responsive while time-insensitive data transfers make use of the communications link when it is otherwise idle.

Use of Alternate Types of Intermediaries

NCAs have the capability of protocol conversion, that is, utilizing a different protocol than that used by the requesting application. This feature enables very useful capabilities, such as the use of alternate types of intermediaries. This is illustrated by the following scenario. A large multinational has a single web server farm that provides employees with access to corporate news, human resource information, etc. Regardless of an employee’s physical location, any access to the corporate web pages results in a hit against the single site and generates a lot of traffic across the oceans. If an employee’s host runs a NCA, requests for pages that have a high probability of being in an HTTP proxy cache can be sent to a nearby cache. If the incoming request was SOCKS-based this will require a protocol conversion. The net effect is to take load off the corporate web site and reduce the bandwidth consumed on inter-site communication lines by using the internal caches local to each country. In addition, the load is reduced on the caches themselves since they only receive requests for data that they have a high probability of having had cached.

A Complete Example

Use of nearly all of a NCA’s capabilities can yield a fault-tolerant service that makes efficient use of available CPU and bandwidth resources. Consider an example that involves the use of annotation services. These services provide an HTTP proxy interface. When a request for an item of data is received, they retrieve it and, when appropriate, rewrite the retrieved data by adding additional content. When a NCA is used on the end user machine, several improvements can be made:

1. Multiple annotation service providers can be supported, which easily allows such typically CPU-bound applications to handle much more load.

2. The annotation service is bypassed when the requested content is of a type that is not handled by the service (e.g., images). This reduces the load imposed the annotation service.

3. If the content is cacheable, the request can be directed to an HTTP proxy cache. This provides an opportunity to reduce the load on firewalls and destination HTTP servers.

4. If the destination is reachable directly, the firewall is not used and a connection is made directly to the target HTTP server.

Note: even if it is not possible to deploy NCAs on end-user machines, a single NCA can be used as the published interface to the annotation service and the same benefits listed above will be achieved. This demonstrates that although NCAs are intended for use on client-machines, they can be deployed in a very effective server-side-only role.

The annotation service itself should use its own NCA to gain the usual benefits (fault-tolerance, avoidance of poor-performing firewalls, bypassing the firewall whenever possible) as well as the intelligent utilization of the HTTP proxy cache. In a similar fashion, the HTTP proxy cache should also use an NCA to gain the performance and fault-tolerance benefits.

From this discussion, it can be seen that NCAs can be deployed in three modes:

1. As a sophisticated intermediary for client applications.

2. As a load-sensitive, fault-tolerant intelligent front-end for a collection of servers that can bypass them when needed.

3. As a back end portal for intermediaries, this provides them much the same functionality as in mode 1.

3. Remote Management of Narwhal Components

NCAs are actively managed by Narwhal management applications. Every Narwhal component can be monitored and managed remotely. Since NCAs are intended to be deployed on each host in an enterprise and be actively managed, this creates a potential scaling problem. Conventional network management systems can perform real-time monitoring but are limited in the number of entities that can be actively monitored. The scaling problem is avoided by focussing on a relatively small number of important network elements, such as routers and servers. In contrast, a complete Narwhal deployment expects the ability to monitor and actively manage every NCA.

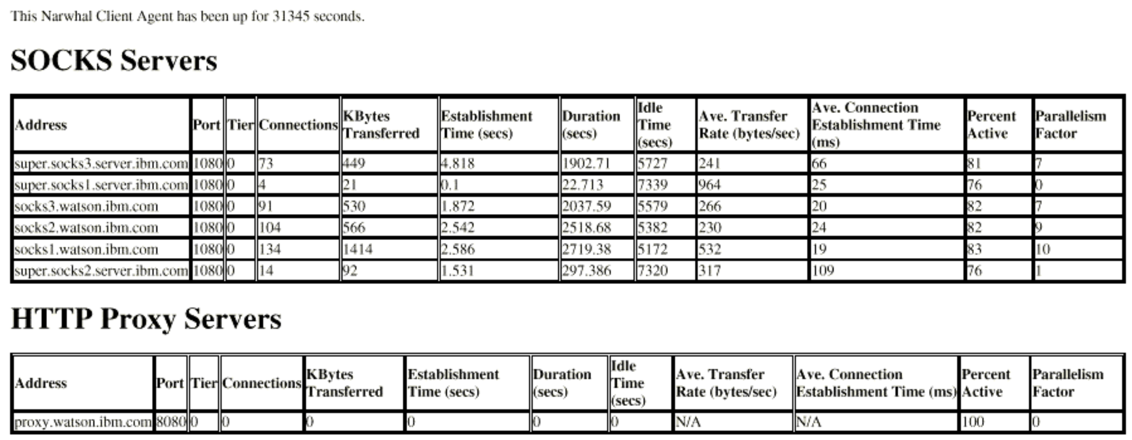
TEMP — The Enterprise Management Protocol

Narwhal components use The Enterprise Management Protocol [TEMP] and its associated architecture to implement the infrastructure for remote management. TEMP is a management protocol and associated infrastructure, currently under development by IBM Research, that enables secure real-time monitoring and management of every host and application in an enterprise. Some of its attributes include:

- TEMP agents are self-describing. This permits generic management applications to determine appropriate variables to monitor and their associated thresholds, as well as enabling the use of generic WWW browsers to display application-specific operator GUIs.

- In contrast to the scalar data types and the requirement for lexicographic ordering of SNMP, TEMP provides a programmer-friendly rich set of data types including sparse and associative arrays, sets, floating-point numbers, etc.
- Multiple TEMP agents can reside on a single host without requiring the use of sub-agent technologies like the SNMP DPI [RFC1228].
- Support for multiple transport protocols. Protocol-specific TEMP components can interact with TEMP components in incompatible protocol domains (e.g., IPX-only TEMP entities can interact with IP-V4/V6-only TEMP entities).

Figure 2 shows a view of current statistics of an NCA:


This Narwhal Client Agent has been up for 31345 seconds.

SOCKS Servers

Address	Port	Tier	Connections	KBytes Transferred	Establishment Time (secs)	Duration (secs)	Idle Time (secs)	Ave. Transfer Rate (bytes/sec)	Ave. Connection Establishment Time (ms)	Percent Active	Parallelism Factor
super.socks3.server.ibm.com	1080	0	73	449	4.818	1902.71	5727	241	66	81	7
super.socks1.server.ibm.com	1080	0	4	21	0.1	22.713	7339	964	25	76	0
socks3.watson.ibm.com	1080	0	91	530	1.872	2037.59	5579	266	20	82	7
socks2.watson.ibm.com	1080	0	104	566	2.542	2518.68	5382	230	24	82	9
socks1.watson.ibm.com	1080	0	134	1414	2.586	2719.38	5172	532	19	83	10
super.socks2.server.ibm.com	1080	0	14	92	1.531	297.386	7320	317	109	76	1

HTTP Proxy Servers

Address	Port	Tier	Connections	KBytes Transferred	Establishment Time (secs)	Duration (secs)	Idle Time (secs)	Ave. Transfer Rate (bytes/sec)	Ave. Connection Establishment Time (ms)	Percent Active	Parallelism Factor
proxy.watson.ibm.com	8080	0	0	0	0	0	0	N/A	N/A	100	0

Figure 2

Narwhal Management Applications

With the ability to be actively managed, NCAs are able to provide additional capabilities that increase their usefulness. As described earlier, when a poorly performing intermediary is discovered, an NCA makes this information available to the Narwhal management system. The management system in turn can inform the other NCAs of the discovered failure, which yields the benefit that none of peer agents needs waste time rediscovering the poorly performing intermediary. In a similar fashion, a scheduled outage of an intermediary (e.g., for maintenance) can be invisible to end-users if the NCAs are told to avoid using the intermediary ahead of time.

Because the Narwhal management system is able to remotely control individual NCAs, resources can be reserved on over-burdened intermediaries. This is done by reducing the number of NCAs that are permitted to use the intermediary in question. The selection of which client hosts remain able to use the intermediary is a non-trivial task and will not be discussed in detail here. It involves making predictions of a client’s resource utilization based on historical data. Resource reservation can help ensure that a demonstration for a customer goes smoothly or an email gateway is able to contribute free cycles while being protected from being overloaded.

Many people use laptop computers in the office as well as at home or while traveling. Mobility can create several problems. When at the office, the laptop needs to use the local intermediaries to gain access to the Internet. Some Internet Service Providers allow corporate users have both secure access to the corporate LAN as well as direct access to the Internet [DUALACCESS]. In such a situation, the use of a firewall is inappropriate. A third situation arises when an employee travels to a different site that is part of a corporation. A multi-national corporation provides the most dramatic example, but the situation occurs even in small companies with multiple sites. If an employee travels overseas, the settings for intermediaries that were appropriate in her office become extremely inappropriate on the other side of an ocean. They may not even function because gateway-imposed filtering based on the source address of a request may render access to the Internet inoperable. Hosts running NCAs can automatically receive appropriate configuration data for their current location. The net result is that a host using a dialup link or the Dynamic Host Configuration Protocol [RFC1541] on a LAN can essentially just plug in and be up and running in any location. No configuration action needs to be taken by the user, no questions need be asked by a visitor to obtain address for local intermediaries.

4. Narwhal Client Agent Performance

A common concern raised by individuals newly exposed to Narwhal technology is a belief that NCAs must introduce overhead that ends up actually slowing performance when the intermediaries are operating perfectly. While Narwhal does indeed introduce some overhead, the overhead is small and the benefits provided compensate for the slight overhead.

We tested the effect of Narwhal in a conventional environment by retrieving a web page and a set of nine GIF images from a given Internet site. Three production SOCKS servers and a NCA configured to use said servers were queried for the 10 files. A series of 400 trials were performed and the resulting plot of the data appears below:

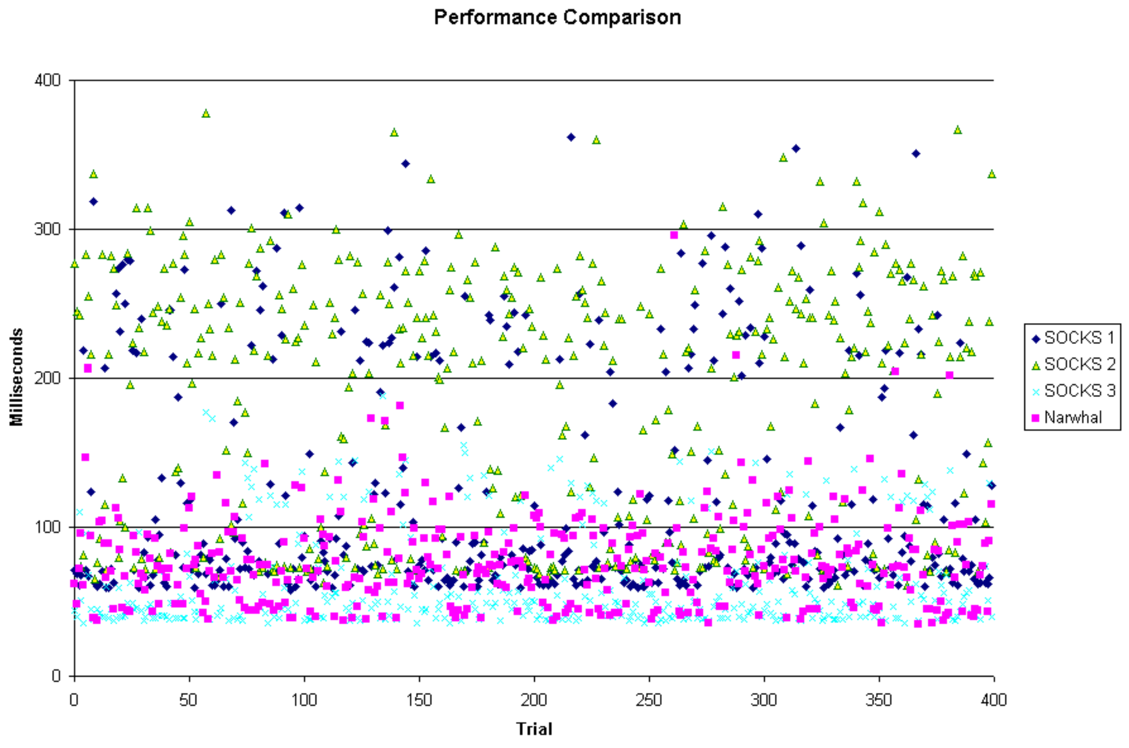


Figure 3

There are two major points illustrated by the graph. The first is that performance of each of the distinct SOCKS servers constantly degrades and improves as time progresses. In other words, a server that has the best performance at time *t* may end up having the worst performance at time *t*+*n*. The second point illustrated by the graph is that the performance obtained by Narwhal is typically in either first place or a close second. Consequently, Narwhal's performance in the presence of no failures is more consistent than that realized by a static choice of SOCKS server.

5. Related Work

Similar problems have been addressed by both server-side solutions, and application-specific solutions. A server-side solution does not address the same problem since it does not have any control over the clients. An application-specific solution can address a similar problem for a given intermediated application at the cost of increased complexity in the application and the intermediary servers. For a more detailed discussion of server-side solutions, and DNS-based solutions, see [GH97]. Our solution is different in that it is client-based, does not require any modifications to the intermediaries or target servers and can support an arbitrary set of networked applications.

Smart Clients and IntelliWeb

The Smart Client [SmartClients] project implemented distinct client-side Java applets targeted to support specific client applications (telnet, ftp, and chat). These applets in turn reissue requests against co-resident Director applet that maintains some state about server performance. Based on the information currently known to the director applet, a connection is established to a particular service. How the director applet actually obtains state information normally involves modifications to the services. The implementation choice requires one to run a web browser to access services and consequently eliminates the generality of the approach.

IntelliWeb [IntelliWeb] is a commercialized offering of the Smart Clients work and more clearly illustrates the problems associated with this approach. IntelliWeb’s applets require a "relaxed" Java security model for the downloaded applet to access other sites. Thus, it requires "Signed Applets" to which the user must grant access privileges. The server-related state information comes from ping times, inferred physical location of the server and the server load. Server load can only be obtained if the service was modified to provide this information to the director applet. If one considers the global Internet, in practice this information will never be available. IntelliWeb requires that the user determine in advance his own physical location as well as that of the servers with which he will be interacting. This is infeasible outside of laboratory experiments. Finally, the only realistically obtained item of information is ping (round-trip packet travel) times, which are neither a indication of actual server load nor a technique that scales to more than a handful of clients.

High-Availability Web Access (HAWA)

HAWA [HAWA] is a client-side applet-based approach for organizing and accessing Web services via group names. It does not provide support for generic applications. It supports four different access modes. HAWA's same-site retry is equivalent to the inherent behavior of most browsers: a failed connection attempt is retried again in case the outage was caused by a transient overload. HAWA's serial retry is the conventional approach to fault-tolerance (use an alternative service when the primary choice has failed), with the nuance that the redundant services do not have to be identical. This is effectively illustrated with an example of a stock quote: many sites provide access to stock quotes and while the visual layout may be different, the desired information content is the same. HAWA's final two modes issue parallel queries and spray requests to several services in parallel. Unfortunately, such greedy use of shared network resources makes these modes unacceptable for large number of users.

Auto Proxy

Early Web browsers provided the ability to make either direct connections to a destination or use an intermediary (like a SOCKS server or HTTP proxy cache). This was done by entering an exclusion list of networks that were directly accessible. There were two basic problems with this approach. The first was that the feature was incorrectly implemented in some browsers and consequently it did not work. This was a showstopper problem for users of such browsers. The other issue was that each browser had to be configured individually. This was a problem regardless of how many excluded networks needed to be entered, but was essentially infeasible for sites with nearly 200 internal networks.

Auto proxy was the next solution to the problem: a small Javascript is automatically retrieved from a well-known location upon startup of the browser. The script represents an algorithm that will determine what intermediary (if any) should be used based on the destination IP address of a request. Auto proxy eliminates the two big flaws associated with the earlier exclusions list approach: it works and does not require individual users to configure their browsers with lists of networks. However, there are some noteworthy problems with auto proxy:

- The auto proxy script is obtained once at startup of the browser and thus cannot be used for distributing changes (e.g., to use a different firewall during scheduled maintenance).
- It typically generates twice the DNS traffic compared to the same browser making connections without using auto proxy.
- Many auto proxy implementations are limited to a small set of protocols and have no effect on other applications, like email, news, or ftp.
- While the auto proxy script can be custom-generated by the web server when it is requested by a particular web browser, there is no provision for dealing with mobile users. The server generating the auto proxy script would have to be omniscient, essentially an impossible task in a large multinational, as well as be able to determine everything about the client based on its IP address.

A NCA's ability to bypass an intermediary coupled with active management by the Narwhal management system provides all of the functionality of auto proxy and none of its limitations.

6. Conclusions

Narwhal provides a significant amount of useful functionality for end-users. This functionality ranges from the mundane (fault-tolerance using alternate intermediaries) to the sophisticated (automatic configuration of mobile hosts, intelligent routing of requests, traffic prioritization). A Narwhal Client Agent can be usefully deployed on an individual's host without requiring the deployment of a supporting infrastructure. It does not require any modification to client code. The only change needed is a reconfiguration to point at an NCA instead of using a given intermediary. In practice, NCAs provide near optimal performance. When performance over time is considered, an NCA evens out the varied performance exhibited by a single server. Consequently, even in perfectly functioning environments, NCAs provide a noticeable benefit. A complete Narwhal system includes active management of NCAs. This system permits profiles of individual client machine behavior and intermediary performance to be obtained without introducing additional test loads that influence the resulting data. Active management enables proactive avoidance of failed intermediaries, automatic configuration of client machines (a significant benefit for mobile users) and bandwidth reservation.

7. References

[GH97] NetDispatcher: A TCP Connection Router, German Goldszmidt and Guerney Hunt, IBM Research, RC8691, 1997

[LPWA] Consistent, Yet Anonymous, Web Access with LPWA, Eran Gabber, Phillip B. Gibbons, David M. Kristol, Yossi Matias, and Alain Meyer. Communications of the ACM, February 1999, Vol 42. No. 2.

[RFC1228] G. Carpenter, B. Wijnen, Simple Network Management Protocol Distributed Program Interface, http://info.internet.isi.edu/in-notes/rfc/files/rfc1228.txt

[RFC1541], R. Droms, Dynamic Host Configuration Protocol, http://info.internet.isi.edu/in-notes/rfc/files/rfc1541.txt

[DUALACCESS] Dual Access FAQs, http://w3.ipinfo.ibm.com/html/body_dualfaq.html

[HAWA] HAWA: A Client-side Approach to High-Availability Web Access, Yi-Min Wang, P. Emerald Chung, Chih-Mei Lin, and Yennun Huang, http://poster.www6conf.org/poster/736/F.html.

[WBI] Intermediaries: New Places For Producing And Manipulating Web Content, Rob Barrett, Paul P. Maglio, WWW7 Conference, http://wwwcssrv.almaden.ibm.com/wbi/intermediaries.html.

[IntelliWeb] http://www.intelliweb.net/index.html

[SmartClients], Using Smart Clients to Build Scalable Services, Chad Yoshikawa, Brent Chun, Paul Eastham, Amin Vahdat, Thomas Anderson, and David Culler", Usenix 97, January 1997, http://now.cs.berkeley.edu/SmartClients/usenix97.ps.

[ASPSHST] Application-Specific Proxy Servers as HTTP Stream Transducers, Charles Brooks, Murray S. Mazer, Scott Meeks, and Jim Miller, http://www.w3.org/Conferences/WWW4/Papers/56/.

[CB94] Firewalls And Internet Security - Repelling The Wily Hacker, William R. Cheswick and Steven M. Bellovin, Addison Wesley, 1994.

[SOCKS] http://www.SOCKS.nec.com/

[SOCKSCAP] http://www.SOCKS.nec.com/SOCKScap.html

[TEMP] The Enterprise Management Protocol, Unpublished IBM Report, Geoff Carpenter, 1998.