

Constructing a DNS-Based Client Redirector for Generic Load Balancing

Marie Charisse L. Gascon, Clifford Ian G. Lim, William Yu, Pierre Tagle, PhD

Authors

Department of Information Systems and Computer Science, Ateneo de Manila University
632-4266001

cha@chasys.net, cliffkins@yahoo.com, wyu@ateneo.edu, ptagle@ateneo.edu

ABSTRACT

This project aims to provide an optimized load balancing solution for delivering content to geographically dispersed users. This involves the construction of a DNS-based server-side application which will direct clients to the nearest or most accessible content mirrored across servers in worldwide locations. Publicly available BGP data and self-collected availability information will be assessed to achieve this client redirection. The solution must be sufficiently general to support any hosted site and deliver a variety of content (live or on-demand).

Categories and Subject Descriptors

C.2.3 [Computer-Communication Networks]: Network Operations – *network management*.

General Terms

Performance, Management.

Keywords

Content Delivery Networks, Load Balancing, Redirection, Border Gateway Protocol, Domain Name System

1. INTRODUCTION

1.1. Background and Problem

The internet is a non-centralized network of computers allowing for the movement of small, “bite-sized” units of information (packets). Packet switching is therefore necessary for the packet to reach its destination, going from server to server across a multitude of connections and ISPs, hopping from one broad network to another depending on connectivity. However, these packets often have to contend with clogged networks and high-traffic connections before it reaches its destination [20], made even worse by local content providers availing of foreign hosting services (refer to M. Paraz’s and W. Yu’s study, *Philippine Internet Content Performance Metrics*, for a case in point) [16].

Mirror sites have been tapped as an indispensable solution to such

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference ’04, Month 1–2, 2004, City, State, Country.
Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

content delivery problems, as it brings the data closer to the client’s location and ensures increased service availability, reduced network traffic and faster download speeds. [30] However, the client needs to be told which mirror server is nearest and most accessible. The central concern becomes the very process of routing a client to the shortest path to - or most accessible source of - his/her requested data, given a myriad of mirror servers hosting similar content.

By combining server replication, data caching and load balancing, routing optimization thus improves content availability by precisely redirecting clients to the nearest service replica in a broad network. Organizations running high-volume websites will also be able to “deliver consistent, responsive service to visitors” [15] and save on bandwidth. In the end, the growth and increasing congestion of the internet necessitates a method of providing web services that “accepts the reality of bad traffic management in the Internet” [28].

1.2. Context

The industry standard for such server replication, data caching and load balancing efforts involves the provision of integrated **Content Delivery Network** (CDN) services.

Succinctly, XCache Technologies defines a Content Delivery Network as a collection of “caching servers deployed around the world to form a network.” [26] Alternatively, Lisa Phifer refers to a CDN as “an overlay network of customer content, distributed geographically to enable rapid, reliable retrieval from any end-user location”. [19] The CDN is composed of three main components:

- 1) “[**Hardware**] caching [and content replication is used] to push replicated content close to the network edge... Stored content is kept current and protected against unauthorized modification.” [19]
- 2) “[**Global load balancing** [or redirection] ensures that users are transparently routed to the ‘best’ content source.” [19]
- 3) The CDN is also **monitored** to ensure network effectiveness and efficiency. “Customer-accessible traffic logs [produced by monitoring agents also] enable data mining for marketing and capacity planning.” [19]

The actual caching devices (caching servers) are basically computers with preinstalled software, and are different from the actual web server. When a client requests content from the CDN, a caching server may first have to retrieve them from the web

server. Once the content is cached in the CDN server closest to the request, however, they are served directly to end users without any further server communication. These individual CDN servers may not communicate with each other. A server-side routing/load-balancing module transparently redirects clients to the nearest cache, and the caching servers themselves only correspond with the web server(s) as the need arises.

1.3. Motivation

Content Delivery Networks can cost a bundle because customers are leveraging someone else's network investment. Rates can be specified per Megabit or Gigabit served, clients can pay monthly rates ranging in the thousands of dollars, or even avail of high-end, value-added features for hundreds of millions of dollars. Routing optimization becomes a huge investment unavailable to the general computing public. General, open-source solutions must be developed for organizations or individuals wishing to avail of the CDN's benefits (essentially, load balancing) without the exorbitant cost.

1.4. Objectives and Scope

The project aims to provide a generic and open-source load balancing solution which transparently connects clients to the "best" available caching server. Thus, of the processes usually involved in the setup, deployment and execution of Content Delivery Networks (Caching and Content Replication, Load Balancing, Security, Logging), the project will solely focus on **load balancing**. It does so by constructing a generic and intelligent DNS-based redirection module capable of being configured on top of existing networks. It will be independent of web server technology used and content type, making the service useful not only for streaming media providers but for any service available on the internet that relies on DNS.

The installation of physical caching servers and subsequent content replication and synchronization will not be part of the project's scope. The project shall be implemented on the main DNS server, with little configuration on the remote cache servers.

2. RELATED WORKS

NetAirt is at present the closest version of the current project available to the computing public. It is part of the Globule project, an open-source module for the Apache Web server which simulates both the Content Delivery Network's replication and redirection services. According to the Globule website, Globule "allows a given server to replicate its documents to other Globule servers. Clients are automatically redirected to one of the available replicas." [7] NetAirt is simply Globule's actual redirector module, and was developed by Michal Szymaniak (for his masteral-level thesis at Freedom University, Amsterdam). Its redirection mechanisms include both HTTP and DNS redirection, and its redirection policies include both round-robin and AS-path length. This means that cache servers are chosen either by cycling among the available mirrors (round-robin), or by analyzing or consulting graphs processed from existing Border Gateway Protocol (BGP) data.

The **Border Gateway Protocol** is an inter-Autonomous System routing protocol - that is, it allows traffic to be transmitted across

networks which have their own routing policy (an Autonomous System) through the exchange of network reachability information between BGP speaking systems or routers. Note that routers are typically connected to two or more routers or local area networks, and it is the presence of these connected routers which keep the Internet connected. The BGP protocol is further documented in the Request for Comments article 1771 (RFC1771, <http://www.ietf.org/rfc/rfc1771.txt>).

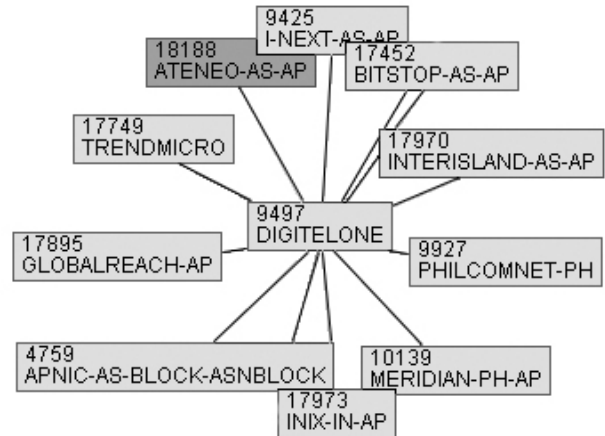


Figure 1: Autonomous System Peering (from Java Autonomous System Path Visualization, <http://lab.verat.net/Jasppi/>). To view a map of the Philippine Internet, refer to G. Ledesma's "Mapping the Philippine Internet" [13].

The AS-path length policy therefore involves determining which cache belongs to an Autonomous System nearest the client's own Autonomous System. The cache belonging to the AS with the shortest number of "hops" to the client's own AS will be chosen.

It must be noted, however, that basing redirection policies on BGP data includes more than just "counting the number of hops" between connections. Since BGP supports "*policy-based routing*" – that is, routing based on "non-technical reasons (for example, political, organizational, or security considerations)" – redirection policies can actually take advantage of bilateral agreements or connections made between ISPs. This feature makes BGP the "leading exterior routing protocol of the Internet". [10]

NetAirt allows users to easily define which policy to be utilized on a per-host basis by appending site-defining directives to Apache's main configuration file (httpd.conf).

Despite Globule's apparent significance, the project will not simply parrot its DNS redirection and AS-path length policies. First, the project endeavors to be web-server independent, capable of being implemented on most any network infrastructure relying on DNS technology. This makes the project extremely lightweight and flexible without sacrificing the complexity that comes with analyzing Internet-wide BGP data. Existing organizations using the "most popular implementation of DNS today" [1], BIND (The Berkeley Internet Name Domain), will for instance be able to preserve their BIND zone data files instead of converting them into NetAirt directives. Second, the project also endeavors to

factor in cache server availability (as will be discussed) in order to support the AS-path length policy.

It must be mentioned that Szymaniak employs a binary tree of prefixes in order to convert client IP addresses to their corresponding Autonomous System Numbers (ASNs): “Each node contains a prefix, which is common to all prefixes in its descendants. Also, each node contains an ASN for addresses matching the prefix, or 0 if the prefix is too short to establish the ASN”. [25] He then employs a graph of Autonomous Systems (each node is marked by the corresponding ASN) to map out the entire internet. For a particular client ASN, a breadth first search is conducted to find the nearest cache server ASN. Data structures close to these will be employed for this project, as Szymaniak reports reasonable lookup times for his methodology (see [25]).

3. METHODOLOGY

3.1. Project Description

The project’s primary deliverable is the **content director**, a server-side application or “hack” implementing the intelligent load balancing module. This will include all necessary pre-processing applications required to parse and incorporate data the content director will use. For the application to be sufficiently general in nature, it will be built on top of DNS (The Domain Name System) and a BIND-like (The Berkeley Internet Name Domain) name server implementation offered as open-source.

The **Domain Name System** “is a global network of servers that translate host names... to numerical IP addresses... which computers on the [Inter]net use to communicate with each other” [29]. This basic service is called “name resolution”. BIND is a Unix-based “implementation of the Domain Name System (DNS) protocols and provides an openly redistributable reference implementation of the major components of the Domain Name System”, namely, the DNS Name Server, DNS Resolver Library, and tools for the proper operation of the DNS Server [6]. By building on top of these basic technologies, the application will be able to perform load balancing by “resolving” the most appropriate IP addresses for individual requests (through the open-source, BIND-like name server), guided by definite redirection policies. These redirection policies will be the core concern of the content director. Redirection criteria include:

- 1) **Distance (BGP):** Clients are routed to the netographically closest caching server by analyzing BGP data.
- 2) **Availability (Ping):** Clients are routed to the first available caching server which responds to the request.

While BGP data does not change frequently, network traffic changes all the time. Peak hours may elicit heavy network traffic in particular areas, or hosts within an Autonomous System may go down for maintenance. Including availability as a redirection criterion allows the program to account for such changes by measuring individual cache server responsiveness.

3.2. Architecture and Design

The content director is further subdivided into two distinct components: a standalone pre-processor, and the actual “hacked” DNS name server. The pre-processor basically parses existing BGP routing information and produces two data structures. The

name server utilizes these data structures to determine the nearest and most accessible cache server for a given client IP address.

3.2.1. Pre-Processor

The pre-processor’s primary task is to parse BGP routing information and produce two sets of data.

The first set of data produced is a PATRICIA trie of network prefixes (a prefix pertains to a 32-bit IPV4 address followed by a slash and the size of the network in bits, as in “4.0.0.1/8”) mapped to Autonomous System Numbers. This will be used by the name server to find an ASN number for a given client IP address.

A trie “is an ordered tree data structure that is used to store an associative array where the keys are strings... [and] all the descendants of any one node have a common prefix of the string associated with that node” [17]. The PATRICIA trie (which stands for “Practical Algorithm to Retrieve Information Coded as Alphanumeric”, and is simply a radix-2 trie), additionally eliminates unary nodes (those with only one child) by merging them with their parents. [24] This keeps the trie height as low as possible. The project relies on such a trie for IP-to-ASN mappings because it “is the data structure most commonly used for routing table lookups. It efficiently stores network prefixes of varying lengths and allows fast lookups of containing networks.” [14] Berkeley Unix’s FreeBSD and 4.3 Reno, for instance, use radix tries to handle their IP routing tables [22, 23]. Various studies on the PATRICIA trie’s performance also indicate reasonable lookup times. Szpankowski reports that the average lookup time for a PATRICIA trie of n records “asymptotically becomes $(1/h) \ln n + O(1)$ ” [24], where h is the entropy of the alphabet (randomness of the language). Reyes reports that the worst case lookup time may be $O(n)$, where n is the length of the longest prefix. [22]

Table 1: Average time complexity comparison between PATRICIA trie and other common data structures, where n is the number of records in the data structure and h is the entropy of the alphabet [8, 22]

Data Structure	Average Time
PATRICIA trie	$(1/h) \ln n + O(1)$
Binary Search Tree	$O(\log n)$
AVL Tree	$O(\log n)$
Array or Linked List	$O(n)$
Hashtable	$O(1)$

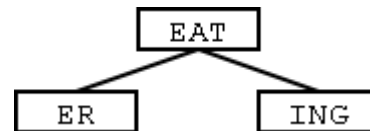


Figure 2: Strings in a PATRICIA trie (figure from [22])

It must be noted that while a Hashtable promises an $O(1)$ retrieval time [9], it is incapable of finding a “best match” with only part of the keys. This is vital since exact client IP addresses will not always be reflected in the BGP data – one must instead defer to the network prefix closest to the client’s IP address.

The second set of data produced is a Hashtable of ASNs mapped to a list of cache server ASNs, sorted by increasing distance.

As the pre-processor conducts its parsing, an undirected graph of Autonomous Systems is already being produced based on the BGP connectivity information. Nodes are labeled by ASN, and peer ASes have edges connecting their nodes. The ideal objective is to develop a graph which reflects the complete topology of the entire internet. However, the graph will only be as “complete” as the source of the BGP routing information.

The graph is then analyzed to produce a Hashtable which maps each ASN in the graph to an ordered list of cache server ASNs – that is, a list of ASes each cache server belongs to, starting with the one nearest the AS in question all the way to the one farthest. The program does this by conducting a breadth-first search for each ASN in the graph, appending cache server ASNs to the list as it finds them. Such ASNs are recognized as having cache servers because the preprocessor also refers to the zone data files and keeps a list of cache IP addresses mapped to their corresponding ASNs. This eliminates the need to perform searching algorithms with each new client request, and the listing of nearest cache servers can be retrieved in $O(1)$ time.

Note that this BGP routing information is not obtained automatically by the pre-processor - rather, the user must retrieve MRT-formatted BGP data himself and call on the pre-processor to sort out the data. The user may choose to obtain the data through routing software (GNU Zebra) installed in his organization’s own machines, or download them from publicly available sources, such as The University of Oregon’s Route Views project. (Refer to <http://www.routeviews.org>)

It is possible to produce both these data structures because BGP route tables contain “a list of routes to prefixes, each of which is associated with the IP address of the peer router that has offered the route, and the AS-path for this route” [25]. However, it must be noted that BGP has certain limitations (discussed in later sections of the paper), and the pre-processor does not test for BGP data integrity or identify malformed data. It is sufficient to recognize these limitations and provide an alternate policy when BGP fails.

3.2.2. Hacked DNS Name Server

An open-source DNS name server was modified to integrate the project’s actual redirection criteria with existing DNS name resolution.

Before the name server actually becomes ready to accept client requests, it first loads the trie and Hashtable produced by the pre-processor into memory, as well as the BIND-format resource record or zone files specifying the details for each cache server.

With all the data structures finally in place, the name server concerns itself with the following:

1. It listens to client requests and retrieves the IP address and ASN (using the PATRICIA trie) of the client.
2. It retrieves the ordered list of nearest cache servers’ ASNs (using the Hashtable), given the client’s own ASN.
3. It finally orders the resource records according to the list of nearest cache servers retrieved, and sends these to the client as a response.

The redirection process is completed by considering cache server availability. The availability policy relies on average Round Trip Times supplied by “pinging”. PING sends an ICMP_echo_request packet (ICMP stands for Internet Control Message Protocol) from the origin computer to port 7 – the “echo port” - of the target host. The response from the target is then called an “ICMP_echo_reply”. Each packet contains either “32 or 64 bytes of data and 8 bytes of protocol reader information”. [27] The total time elapsed from sending the request to receiving a response from the remote host is thusly called the Round Trip Time. In this way, PING provides an accurate measurement of performance in the network layer (OSI Layer 3).

The name server will keep track of Round Trip Times for all of its cache server records according to a specified frequency, listing them in order of decreasing performance. This list will serve as a basis for slightly re-ordering the response records provided by the BGP analysis, increasing the robustness of the name server’s load balancing efforts.

This procedure becomes imperative since there shall be no other monitoring mechanisms deployed on the remote cache servers to provide the name server with network traffic or server health statistics. The name server should have the means to independently monitor from its own vantage point, and do so through standard protocols deployed across the internet. Pinging therefore becomes a viable option. Since the ICMP packet is supported by “almost every network-capable operating system” [27], it allows both the name server and the cache servers to communicate whatever their platforms. If the cache servers are running, replies to the name server are almost guaranteed.

The administrator will be able to specify whether the name server relies on the distance policy alone (BGP analysis), on the availability policy alone, or on both (BGP analysis followed by slight re-ordering based on the PING statistics).

3.3 Implementation

The program was written entirely in Java (JDK 5) on a Linux-based environment.

The pre-processor parses MRT-based BGP data directly (Multi-Threading Routing Toolkit routing information export format). [3] It then makes use of 3rd-party, open-source data structure libraries written in Java to ensure optimal performance. The PATRICIA trie implementation was taken from Konstantin Knizhnik’s “PERST” library (unlicensed), an embedded object-oriented database for applications promising high performance. [18] The graphing libraries were taken from Barak Naveh’s “JGraphT” (GNU Lesser General Public License), which also promises high performance for graphs with millions of vertices and edges. [11]

As previously mentioned, the program also modifies an existing open-source Java name server called “JNamed” (BSD License), a sample program of the DNS Java library, “dnsjava”. [5]

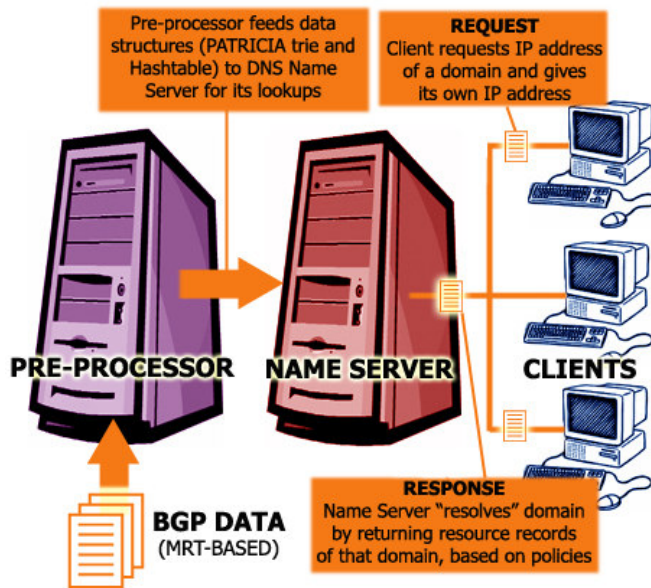


Figure 3: Program Architecture

The actual mechanics of redirection based on the distance or BGP policy is as described in the architecture section. Redirection based on availability, however, requires a more detailed explanation.

The name server routinely monitors individual cache server responsiveness by “pinging” each cache server a set number of times and ranking these servers according to Average Round Trip Time (the number of times the name server pings per checkup for each cache server and the duration between pings for all caches can be configured). The varying degrees of availability of the cache servers are then subdivided into categories that the administrator specifies. If none are indicated, the following ten categories are used:

Table 2: Default Categories of Availability maintained by Availability Policy based on Average Round Trip Time

Average Round Trip Time	Availability
< 100 milliseconds	High-High
< 200 milliseconds	High-Medium
< 400 milliseconds	High-Low
< 700 milliseconds	Medium-high
< 1000 milliseconds	Medium-Medium
< 1300 milliseconds	Medium-Low
< 1500 milliseconds	Low-High
< 1700 milliseconds	Low-Medium
>= 1700 milliseconds	Low-Low
Ping Fails	Unreachable

The name server maintains a list of cache server records ranked according to such categories of availability, and the list is updated with each new check. Unreachable cache servers will be flagged and will not be included in the list at the time of checking. This list is kept separate from the BGP analysis’ own lists of nearest cache servers. Note that the category names are entirely cosmetic (used only for purposes of explanation); records are ranked numerically within the DNS name server depending on the total

number of categories the user specifies, if such information is available.

In addition, the DNS name server will also attempt to contact the cache servers through common services or ports specified by the administrator. The time it takes to establish a socket connection with each of these service ports are added to the average Round Trip Time if such connections were established. If no connection is established by the time the attempt times out (this timeout is also configurable), the port is ignored and the cache server does not become unreachable solely because of the failed attempt.

This allows the program to determine reachability information for various services at the application level, compensating for the network-layer-only monitoring provided by pinging. Thus, if the HTTP server of a particular cache is clogged with requests, for instance, the average Round Trip Time of that cache server increases even if a normal ping on the echo port replies in a timely manner. Note that the ports the name server contacts for the availability policy are also externally configurable.

If the administrator opts to combine the distance or BGP policy with the availability policy, the program reorders the cache server records within each category according to the BGP analysis. Response records will thus be ordered from high to low availability - and within each category of availability, according to the BGP data. The availability policy’s ranking system prevents an over-shuffling of records when results between both policies are integrated.

4. RESULTS

Functional testing of the redirection policies was conducted by the program authors to ensure program correctness.

4.1. Functional Testing of Distance Policy

Two sets of tests were conducted in order to verify program correctness on both tight and wide network configurations. The BGP data utilized for both tests was taken from the University of Oregon Route Views Project on Dec. 25, 2004 (rib.20041225.1456).

Resource records in the name server’s zone data files were configured to simulate the deployment of cache servers in definite networks. The name server was deployed on the local host and queries to the name server (using varying test IP addresses) were also made from the same machine. In Linux, *dig* (Domain Information Groper) can be used for such DNS queries. In order to verify that the test IP address indeed belongs to a particular ASN, queries were also made to a whois server (whois.cymru.com).

The first test was based on the following AS connectivity information and cache server configuration (note that other connecting ASes irrelevant to the results – that is, those which connect one cache server to another via a longer path - were removed):

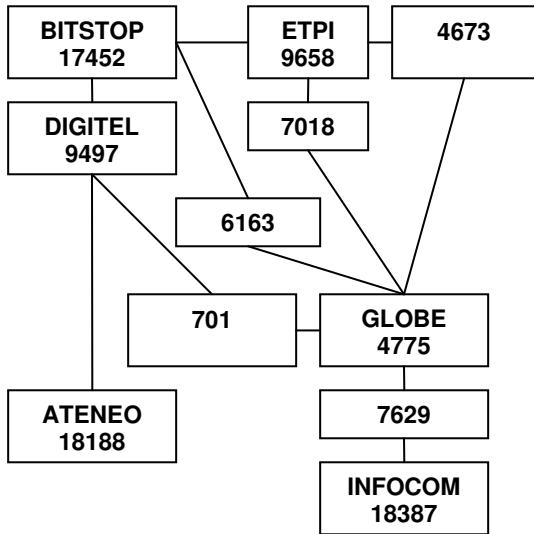


Figure 4: AS Connectivity Map for Distance Policy Test 1

Table 3: Configured “Mock” Cache Servers for Distance Policy Test 1

Cache IP Address	ASN	Server Name
203.167.103.210	9658	ETPI
202.138.128.72	9497	DIGITELONE
203.177.3.42	4775	GLOBE
203.172.21.253	18387	INFOCOM
202.91.161.133	17452	BITSTOP

The name server was queried using test IP addresses belonging to the deployed cache servers – ETPI, DIGITELONE, GLOBE, INFOCOM and BITSTOP. The list of resource records returned by the name server reflected ASes of increasing distance from the test IP addresses (as shown in Table 4). This confirms that the distance policy works based on the BGP information.

Table 4: Results of Distance Policy Test 1 - for the list of returned ASNs, each ASN is followed by a value in brackets pertaining to the number of hops from the client to the ASN in question. The last column shows whether the returned list of records was correct or as expected.

Test/Client IP Address	ASN of Client	Returned List of ASNs	Correct / Expected?
203.167.102.74	9658 (ETPI)	9658 [0] 17452 [1] 4775 [2] 9497 [2] 18387 [4]	Yes
202.138.180.141	18188 (Ateneo)	9497 [1] 17452 [2] 4775 [3] 9658 [3] 18387 [5]	Yes
203.172.11.25	18387 (Infocom)	18387 [0] 4775 [2] 17452 [4] 9658 [4] 9497 [4]	Yes
202.91.163.18	17452	17452 [0]	Yes

	(Bitstop)	9658 [1] 9497 [1] 4775 [2] 18387 [4]	
203.177.23.57	4775 (Globe)	4775 [0] 17452 [2] 9658 [2] 9497 [2] 18387 [2]	Yes

A second test was conducted on a more widely distributed network using the following AS connectivity information and cache server configuration:

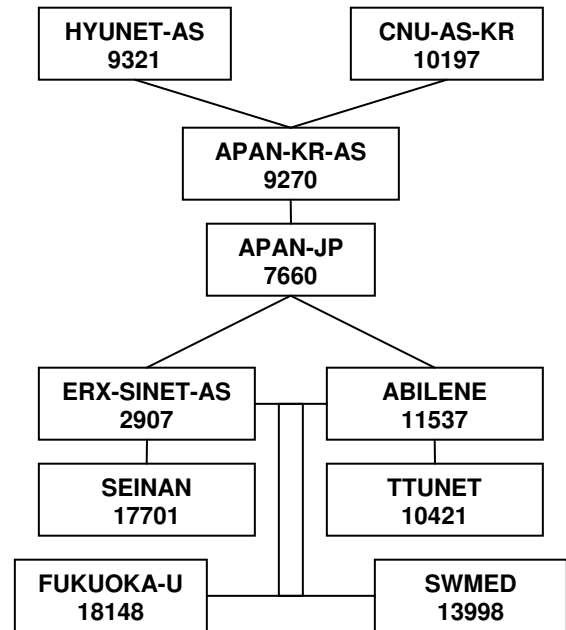


Figure 5: AS Connectivity Map for Distance Policy Test 2

Table 5: Configured “Mock” Cache Servers for Distance Policy Test 2

Cache IP Address	ASN	Server Name
129.118.0.1	10421	TTUNET
168.131.0.1	10197	CNU-AS-KR
160.23.0.1	17701	SEINAN

The name server was queried using test IP addresses belonging to various ASes. The list of resource records returned by the name server reflected ASes of increasing distance from the test IP addresses (as shown in Table 6). This once again confirms that the distance policy works based on the BGP information.

Table 6: Results of Distance Policy Test 2 – for the list of returned ASNs, each ASN is followed by a value in brackets pertaining to the number of hops from the client to the ASN in question. The last column shows whether the returned list of records was correct or as expected.

Test/Client IP Address	ASN of Client	Returned List of ASNs from Name Server	Correct or Expected?
166.104.192.1	9321 (Hyunet)	10197 [2] 10421 [4] 17701 [4]	Yes
133.100.0.1	18148 (Fukuoka)	17701 [2] 10421 [4] 10197 [4]	Yes
129.112.0.1	13998 (SWMED)	10421 [2] 17701 [4] 10197 [4]	Yes

4.2. Functional Testing of Availability Policy

Functional testing of the availability policy will only be successful if the following goals are met in the course of the test: 1) the name server pings all available cache servers and sets up socket connections with them for each port specified; 2) the name server ranks the cache server records according to the categories of availability given in the configuration file; and 3) if the BGP policy is also active, the list of cache servers returned to the client must be reordered according to the rankings of the availability policy. Based on seven continuous tests on physical cache servers, it can be reasonably concluded that the availability policy works.

Table 8 illustrates the collected Average Response Times and assigned rankings after pinging and connecting to each cache server on various ports. Table 9 illustrates the subsequent reordering of the list returned by the distance or BGP policy, based on the availability policy.

Table 7: Categories of Availability used in the Availability Policy Test – the Average Response Time (ART) column pertains to the maximum time a connection can take and still belong to the ranking.

Category Rank	ART (in milliseconds)
0	100
1	200
2	400
3	700
4	1000
5	1300
6	1500
7	1700

Table 8: Results of Availability Policy Testing – the table lists the collected Average Response Times (in milliseconds) and assigned rankings (in parenthesis) after pinging and connecting to each cache server on the specified ports. This shows that the availability policy is functioning correctly.

Test #	1	2	3	4
Cache A	101.26(1)	100.87(1)	101.09(1)	204.90(2)
Cache B	566.66(3)	566.66(3)	566.66(3)	239.29(2)
Cache C	96.43(0)	94.56(0)	96.01(0)	193.06(1)
Cache D	111.90(1)	110.67(1)	120.13(1)	361.63(2)

Cache E	103.82(1)	95.91(0)	95.17(0)	191.66(1)
Test #	5	6	7	
Cache A	233.92(2)	205.05(2)	204.03(2)	
Cache B	210.05(2)	208.7(2)	207.04(2)	
Cache C	192.93(1)	252.37(2)	192.17(1)	
Cache D	385.38(2)	365.99(2)	338.59(2)	
Cache E	191.00(1)	252.75(2)	191.43(1)	

Table 9: Final reordering of the list of cache server records returned to the client – the table illustrates how the distance policy’s results are sufficiently adjusted to meet the constantly changing server responsiveness information offered by the availability policy. The returned list of cache servers are sorted from most appropriate (left) to least appropriate (right). The letters A, B, C, D and E pertain to cache servers.

Client A	
BGP	A, E, C, B, D
Test 1	C, A, E, D, B
Test 2, 3	E, C, A, D, B
Test 4, 5, 7	E, C, A, B, D
Test 6	A, E, C, B, D
Client B	
BGP	B, E, C, A, D
Test 1	C, E, A, D, B
Test 2, 3	E, C, A, D, B
Test 4, 5, 7	E, C, B, A, D
Test 6	B, E, C, A, D
Client C	
BGP	C, E, A, B, D
Test 1	C, E, A, D, B
Test 2, 3	C, E, A, D, B
Test 4, 5, 7	C, E, A, B, D
Test 6	C, E, A, B, D
Client D	
BGP	D, C, E, A, B
Test 1	C, D, E, A, B
Test 2, 3	C, E, D, A, B
Test 4, 5, 7	C, E, D, A, B
Test 6	D, C, E, A, B
Client E	
BGP	E, A, B, D, C
Test 1	C, E, A, D, B
Test 2, 3	E, C, A, D, B
Test 4, 5, 7	E, C, A, B, D
Test 6	E, A, B, D, C

4.3 Program Performance

Program performance was tested on an AMD Athlon XP-M 1500 machine with 1.33 GHz of processing speed and 256 MB of RAM. The operating system utilized was Fedora Core 3.

Average processing time for the pre-processor (using varying subnet filters), and average loading and lookup times for the name server are listed in Table 10. The BGP data used was taken from the University of Oregon’s Route Views Project on Dec. 25, 2004 (rib.20041225.1456), and is 404.6 MB in its uncompressed form.

Table 10: Performance times for the different component processes. A “/n filter” means that the pre-processor was configured to exclude subnets with values larger than n. Times are measured in hours (h), minutes (m), seconds (s), and milliseconds (ms) where applicable.

Process	Remarks	Ave. Time
Pre-processor	/8 filter	0h 8m 10s
	/16 filter	0h 10m 36s
	/24 filter	1h 52m 37s
	No filter	1h 50m 3s
Name Server	Loading	0h 3m 26s
	Lookup / Resolution	0h 0m 0s 215ms

5. RECOMMENDATIONS

Based on the test results, it can be reasonably concluded that the content director is able to perform what was intended of it: to redirect clients to the most appropriate cache servers for them based on BGP data and individual cache server responsiveness. However, it still remains to be seen whether the program indeed minimizes network traffic, saves on bandwidth, and ultimately increases content delivery speeds. These can only be verified by obtaining performance metrics on a live network.

Despite the numerous advantages of anchoring redirection policies on BGP data, its apparent limitations must nevertheless be mentioned.

BGP’s effectiveness as a redirection policy is limited by its ignorance of link speeds and traffic congestion information. In other words, BGP data merely reflects which BGP speakers are willing to route packets to each other, without conveying the quality of that connection. In this way, it may happen that a two-hop route to a certain cache server may actually be more congested than a four-hop route to another cache server in real time, and it would have been better to point the client to the latter.

Because BGP is an interdomain routing protocol, the program also fails to consider the internal routing mechanisms within each Autonomous System. In the same way that a link between two Autonomous Systems may be congested, the internal routing protocols used by a single Autonomous System also determines whether a packet is optimally delivered to the next destination in the route.

Finally, the BGP protocol is also susceptible to the action of “aggregators” – collections of interconnected Autonomous Systems identified by a single, overarching Autonomous System. A two-hop route may actually mask ten smaller hops as the packet reaches an “aggregated” Autonomous System.

The apparent weaknesses of BGP are mitigated by a more direct and manual form of host-to-host communication: pinging. However, the availability policy also has its own limitations. First, ping packets may be given low priority while traveling across routers and switches in the internet, or may even be blocked. This reduces the accuracy of internet traffic or cache server performance measurement since average Round Trip Times computed become greater than they actually are. Second, the program’s availability policy provides limited monitoring on the application level. However, checks on common service ports were

undertaken in order to compensate for this. Third, pinging from the name server only gives an account of network traffic relative to the name server and the caches. Information regarding server load is not collected from the client’s unique vantage point.

Despite these limitations, it must nevertheless be noted that the Border Gateway Protocol is still the “de facto standard interdomain routing protocol”. [21] The fact remains that BGP-speaking systems advertise all autonomous systems on the path to a destination address. “As a result of this full advertising, a node that receives more than one possible path to a destination can, without ambiguity, choose the best path.” [2] In the end, its robustness, scalability, support for policy-based routing (routing due to non-technical reasons), and support for classless interdomain routing (the use of any network prefix, not just /8, /16 and /24) make it a significant asset in server load balancing. [4]

The project’s flexibility allows for extensive future development. New redirection policies can easily be appended by “hacking” into the name server and adding new lookup sources. Frederick Echevarria and Ceres Parlade’s (Ateneo de Manila University Undergraduate Project) project, in particular, collects network latency information by sending recursive UDP packet trains. Such an application can form the basis for a latency redirection policy.

Furthermore, the current project may be further modularized by creating a general redirector framework. Such a framework should allow various redirection policies to be merely “plugged in” without adjusting the DNS name server’s code. The redirector framework may also be divorced from any one particular name server, or implemented in such a way that it may be attached to popular name servers such as BIND.

6. REFERENCES

- [1] Albitz, Paul and Liu, Cricket. *DNS and Bind, 4th ed.* O’Reilly and Associates, Inc., USA, 2001.
- [2] Black, Uyles. *TCP/IP and Related Protocols, 3rd ed.* McGraw-Hill, USA, 1998.
- [3] Blunk, L. and others. “MRT routing information export format”, Multi-Threading Routing Toolkit Homepage. <http://www.mrtd.net/mrt_doc/mrt-draft-00.html> [13 December 2004]
- [4] “Border Gateway Protocol (BGP)”, Cisco Systems, Inc. <http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bgp.htm#1020549> [28 December 2004]
- [5] DNSJava. <<http://www.xbill.org/dnsjava/>> [13 December 2004]
- [6] DNS, BIND, DHCP, LDAP and Directory Services. Bind9.net. <<http://www.bind9.net/>> [18 October, 2004].
- [7] *Globule: An Open-Source Content Distribution Network.* Globule Website. <www.globule.org> [10 December, 2004].
- [8] Goodrich, Michael and Tamassia, Robert. *Data Structures and Algorithms in Java, 2nd ed.* John Wiley & Sons, Inc., New York, 2001.

- [9] "Hash Table", National Institute of Standards and Technology. <<http://www.nist.gov/dads/HTML/hashtab.html>> [12 December 2004]
- [10] Hunt, Craig. *TCP/IP Network Administration, 2nd ed.* O'Reilly and Associates, Inc., USA, 1997.
- [11] JGraphT. <<http://jgraph.sourceforge.net/>> [13 December 2004]
- [12] Krishnamurthy, Balachander, Wills, Craig and Zhang, Yin. *On the Use and Performance of Content Distribution Networks*, AT&T Labs Research Papers. <<http://www.research.att.com/~yzhang/papers/cdn-imw01.pdf>> [18 October, 2004].
- [13] Ledesma, G. *Mapping the Philippine Internet using the Border Gateway Protocol*. BS Computer Science 2002-2003 Undergraduate Research Project, Ateneo de Manila University, 2003.
- [14] Morrison, Donald R. "PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric", *Journal of the ACM* Volume 15, Issue 4, ACM Press, New York, USA, 1968. pp. 514 – 534. (Actually accessed through: Py-Radix: A Radix Tree for Python, <<http://www.mindrot.org/py-radix.html>> [11 December 2004])
- [15] Ozer, Jan. *Speedy Delivery*, PC Magazine. <<http://www.pcmag.com/article2/0,1759,1174272,00.asp>> [18 October, 2004].
- [16] Paraz, M. A. and Yu, W. S. 2002. *Philippine Internet Content Performance Metrics*. Sixth International Symposium on Parallel Architectures, Algorithms, and Networks (ISPAN2002), May 2002.
- [17] "Patricia Trie". The Free Dictionary. <<http://encyclopedia.thefreedictionary.com/Patricia%20trie>> [11 December 2004]
- [18] PERST. <<http://www.garret.ru/~knizhnik/perst.html>> [13 December 2004]
- [19] Phifer, Lisa. *Content Delivery Networks: Emerging Opportunity for Service Providers*. ISP Planet Website. <http://www.isp-planet.com/technology/cdn_connection.html> [18 October, 2004].
- [20] Priestman, Chris. *Web Radio: Radio Production for Internet Streaming*. Reed Educational and Professional Publishing Ltd, Great Britain, 2002.
- [21] Quoitin, B., Uhlig, S., Pelsser, C., Swinnen, L., and Bonaventure, O. "Interdomain traffic engineering with BGP". <<http://www.info.ucl.ac.be/people/OBO/papers/commag-may2003.pdf>> [26 December 2004]
- [22] Reyes, R. *Optimized IP to ISO3166 Country Code Mapping in C#*, The Code Project. <<http://www.codeproject.com/csharp/iptocountry.asp>> [11 December 2004]
- [23] Sklower, Keith. *A Tree-Based Packet Routing Table for Berkeley Unix*. <http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/HTML/gated_html/radix.html> [11 December 2004]
- [24] Szpankowski, Wojciech. "Patricia Tries Again Revisited", *Journal of the ACM*. Volume 37, Issue 4, ACM Press, New York, USA, 1990. pp. 691 – 711.
- [25] Szymaniak, Michal. *A DNS-based Client Redirector for the Apache HTTP Server*. Masteral Thesis, Vrije Universiteit, July 2002.
- [26] *Untangling the Web of Caching Alternatives*. XCache Technologies Website. <<http://www.xcache.com>> [18 October, 2004].
- [27] "What is PING?", Indiana University Knowledge Base. <<http://kb.indiana.edu/data/aopu.html?cust=203559.38244.131>> [30 December 2004]
- [28] Willis, David. *The Content-Delivery Edge*, Network Computing. <<http://www.networkcomputing.com/1103/1103colwillis.html>> [18 October, 2004].
- [29] Windbigler, Kristin. *Exploring the Domain Name Space*. WebMonkey: The Web Developer's Resource. <<http://webmonkey.wired.com/webmonkey/webmonkey/geektalk/97/03/index4a.html>> [18 October, 2004].
- [30] Zegura, Ellen, *Performance of Mirror Servers*. <http://www.cc.gatech.edu/classes/AY2001/cs7001_fall/projects/zegura05.html> [18 October, 2004]