The following paper was originally published in the
Proceedings of the Twelfth Systems Administration Conference (LISA '98)
Boston, Massachusetts, December 6-11, 1998

Automatically Selecting a Close Mirror Based on Network Topology

Giray Pultar
giray@coubros.com

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Automatically Selecting a Close Mirror Based on Network Topology

Giray Pultar – giray@coubros.com

ABSTRACT

The content of many popular ftp and web sites on the Internet are replicated at other sites, called ‘‘mirrors’’; typically, to decrease the network load at the original site, to make information available closer to its users for higher availability; and to decrease the bandwidth requirements these sites place on long-haul network connections, such as international and backbone links.

Even though the success of mirroring depends heavily on the selection of a good mirror, there are very few methods to pick a good mirror: i.e., a mirror ‘‘close’’ to its user based on network topology.

This paper describes a method and two tools developed to locate a ‘‘close’’ mirror among replicated copies of a network service such as ftp, www, irc, streaming audio by utilizing network topology information based on autonomous systems. Routing information from the Internet Routing Registry is combined with information about the location of mirrors to generate mirroring tables, similar to routing tables, which are used to identify a ‘‘close’’ mirror, where ‘‘close’’ is defined as traversing the minimum number of autonomous systems.

The tools are avaliable via anonymous ftp from ftp.coubros.com .

Background Material

Mirroring

‘‘Mirroring’’ is the replication of the content of a site (such as a web site or an ftp site) at a different site called a ‘‘mirror.’’

The content at the original site is replicated at the mirror, typically once a day, by utilizing a mirroring program, such as `mirror` [6].

In most cases, the locations of the mirrors are advertised at the original site, and users of the site are asked to choose a mirror ‘‘close’’ to themselves. These mirrors are called ‘‘public mirrors,’’ because their existence is advertised at the original site; and users of the original site are allowed to use any of these mirrors. Sites are mirrored at public mirrors to decrease the network and server load at the original site and to make the content available to the public even if the original site is down.

In some cases, the location of a mirror is not advertised at the original site. Such a mirror is called a ‘‘private mirror,’’ because its existence is not advertised at the original site; and the mirror is intended for a subset of the users. Sites are mirrored at private mirrors to decrease the network load at the mirror location (based on the assumption that the content will be requested by the users at the mirror location more than once) and to make the content available to the users of the private mirror even if the original site is down.

In this paper, we are only concerned with public mirrors.

Autonomous Systems

Section 2.2.4 of RFC 1812 [3] gives the definition of an autonomous system:

‘‘An Autonomous System (AS) is a connected segment of a network topology that consists of a collection of subnetworks (with hosts attached) interconnected by a set of routes. The subnetworks and the routers are expected to be under the control of a single operations and maintenance (O&M) organization. Within an AS routers may use one or more interior routing protocols, and sometimes several sets of metrics. An AS is expected to present to other ASes an appearance of a coherent interior routing plan, and a consistent picture of the destinations reachable through the AS. An AS is identified by an Autonomous System number.’’

Figure 1 depicts a hypothetical internet made up of three autonomous systems. Autonomous system 1 (AS1) has two routers; AS2 has three and AS3 has two routers. There are external network connections between AS1 and AS2; and between AS2 and AS3. This hypothetical network will be used throughout this paper.

Exterior Gateway Protocols

Section 7.3.1 of RFC 1812 [3] states: ‘‘Exterior Gateway Protocols are utilized for inter-Autonomous System routing to exchange reachability information for a set of networks internal to a particular autonomous system to a neighboring autonomous system.’’

In our hypothetical world, for example, AS1's router would tell AS2's router that it can reach all the networks inside AS1 (such as the network containing client 1). Similarly AS2's router would tell AS3's router that it can reach all the networks inside AS2, (such as the network containing Server 1). Moreover, this AS2 router, would also tell AS3 that it can reach the networks connected AS1.

With the exchange of this information, AS3 would learn that it can reach Client 1 via AS2. The protocols used for exchanging this information are called External Gateway Protocols.

It is beyond the scope of this document to describe the details of exterior gateway protocols, such as BGP, and the reader is referred to [2].

The Internet Routing Registry – IRR

There are three components to understanding the motivation for the internet routing registry (IRR) [4] : exchange points, policy routing and route servers.

Exchange Points

The Internet is a collection of interconnected networks managed by different network providers. When a host is connected to the internet; it is connected to one of the networks that make up the Internet. The fact that a host on the Internet can reach any other host on the internet is the result of agreements between the network providers and the connections between the different networks that make up the internet.

If each provider had to connect to every other provider on the Internet directly, the number of network connections required would grow with the square of the number of providers; and therefore would be impractical.

The solution to this problem was the creation of exchange points (aka network access points). Each network provider gets connected to an exchange point, and can then communicate with all other providers connected to the same exchange point. In this scenario; the number of network connections required grows linearly with the number of network providers.

There are currently several exchange points in the US, such as MAE-EAST, MAE-WEST, PAIX, etc. For more information about exchange points, the reader is referred to [5].

Policy Routing

As described in the previous section, there are several network providers that are connected to each exchange point. However, this does not mean that each provider is willing to exchange traffic with all other providers at the same exchange.

When a provider does not want to send traffic to another provider, they must modify the routing tables on their routers that connect to the exchange point to direct this traffic to a different provider.

Generating routing tables, not only based on network connectivity, but based on the preferences or policies of a provider is called ‘‘policy routing.’’

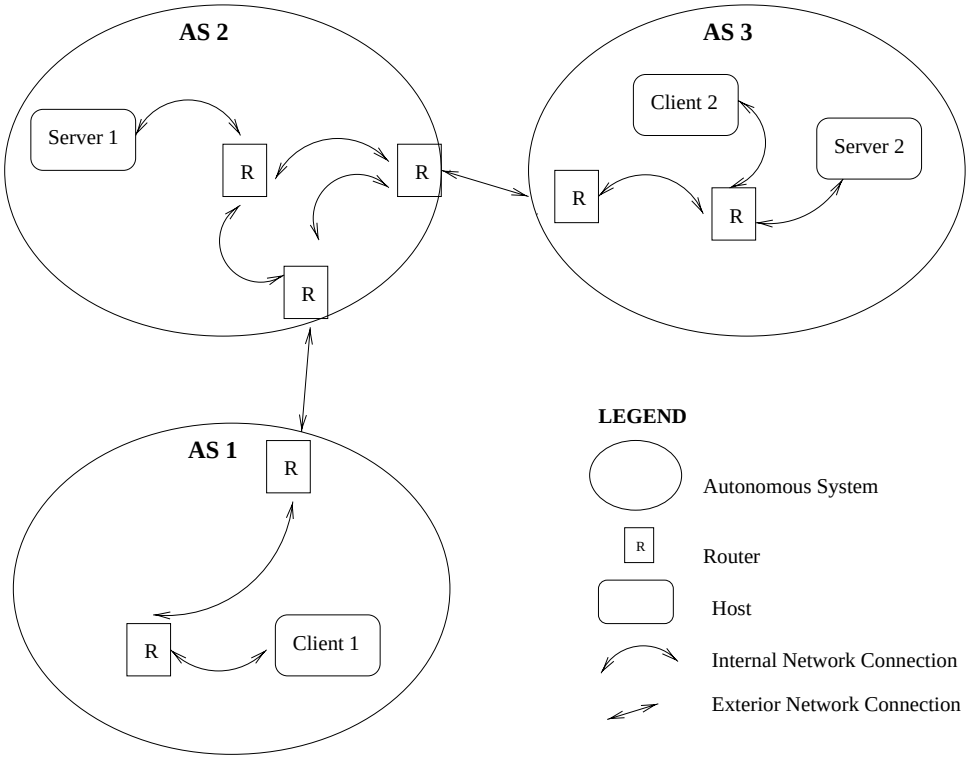

Figure 1: A hypothetical internet with 3 autonomous systems.

Route Servers

There are two main functions performed by a router:

1. routing packets: receiving packets and sending these packets out based on IP addresses and the router's routing tables
2. routing table management: communicating with neighboring routers and updating routing tables and calculating new routes based on changes in network connectivity

''Route servers'' are computers (typically UNIX based general purpose computers, as opposed specialized hardware such as routers) designed to offload the routing table management function from the routers, so that they can focus on routing packets. There are typically two or more route servers at each exchange point. These route servers communicate with all of the routers at the exchange point, collect connectivity information and route advertisements, and prepare routing tables for all of the routers based on the policies of each network provider. The routers at the exchange point, communicate with the route server and receive their routing tables specifically prepared for them.

To prepare routing tables for each router at the exchange, the route servers need to know the policies of each Internet Service Provider to honor their preferences. The Internet Routing Registry (IRR) is a collection of routing policies of each ISP to be used by route servers in preparing the routing tables.

The information that makes up the IRR is currently stored in five different databases, provided by five different organizations. For more information on the IRR, see [4] .

The databases in the IRR are stored in the RIPE-181 format [8] and contain objects that describe network providers, their contacts, autonomous systems and routes. The objects of interest to this paper are the ''route'' object, and the ''autonomous system'' object.

The ''route'' object defines a route in the CIDR format, and its originating autonomous system.

```
route: 198.87.45.0/24
origin: AS3333
[...]
```

This tells us that the address space 198.87.45.0-198.87.45.255 originates in AS3333.

The ''autonomous system'' object defines an autonomous system and a list of its neighboring autonomous systems and what routes it is willing to advertise, and what routes it is willing to accept from those autonomous systems. For example, see Listing 1.

This object tells us that autonomous system AS1104 connects to AS1213 and is willing to accept route advertisements from AS1213 for all routes that are registered in the IRR with an origin of AS1213. It is also willing to accept any route from AS1755. In advertising routes, AS1104 will advertise all the routes it knows about to AS1213; and all the routes registered as originating from AS1213 and itself from AS1104. It is likely that this autonomous system connects to the Internet via AS1104, and AS1213 connects to the Internet via this autonomous system. (However, this is not conclusive, as there may be other entries in the IIR that describe how AS1213 is connected.)

Existing Methods

There are relatively few existing methods of selecting a ''close'' mirror.

- **User selection:** This is the method that is most commonly used. The user is given a list of all the mirrors and is asked to choose one which they think is ''close'' to them.
- **Geographical:** This method is a slightly improved version of the ''user-selection'' method. The user is, again, given a list of all the mirrors along with their geographical locations; and is asked to choose one which they think is ''geographically close'' to them (e.g., `www.gnu.org`).
- **Connect everywhere:** This method is really not a method for selecting a ''close'' mirror; but making a site ''closer'' to its users. The idea is to get connected to many countries and many exchange points, so that the server is ''close'' to everyone (e.g., `www.digisle.net`).
- **Domain name:** This is one of the two existing methods found that can automatically select a close mirror. It tries to match as many labels as possible from the fully qualified domain names of the server and the client. For example, a client at `joe.domain.com` would use `ftp.domain.com` if there were such a mirror, since the `domain.com` portion of the

```
aut-num: AS1104
as-in: from AS1213 100 accept AS1213
as-in: from AS1755 150 accept ANY
as-out: to AS1213 announce ANY
as-out: to AS1755 announce AS1104 AS1213
[...]
```

Listing 1: Route acceptance example.

names match (e.g., `www.perl.com/CPAN`).

- **Router based:** The other method to automatically select a close mirror is by using information from routers at the mirror sites. For example, Cisco's Distributed Director product [7] uses Cisco's Director Response Protocol (DRP) to find a close mirror. This method requires that the all the routers at the mirror sites implement DRP, and therefore use Cisco IOS.

Initial Ideas

Following are some of the ideas generated in formulating solutions to the problem of ''finding a close mirror.''

Client-side Selection

One of the first issues looked at was the advantages and disadvantages of finding a ''close'' mirror at the client side.

The first issue with a client side solution is that the software must be installed at the client to perform the mirror selection. This problem has recently been somewhat resolved by client side scripting and virtual machine technologies, such as Java and Javascript. Loading software on to the limited number of mirrors is easier than loading software onto the clients.

We assume that the server, already contains a list of its public mirrors. (This is probably a valid assumption in most cases on the Internet today.) It would be unreasonable to expect the client to have a priori knowledge of mirrors for all sites. Even if the client could look up a list of mirror sites, this information would most likely be coming from one of the mirrored sites, anyhow.

Implementing a client side solution would make sense if there was any information that clients already possessed which was necessary to make the ''closest'' decision; but was also very difficult to transmit to the server side. The only piece of information that comes to mind that the server does not already possess is the IP address of the client; however, the transmission of this information is very simple.

Based on the software loading problem and the location of data already available to make a decision, in our opinion, the method of finding a ''close'' mirror must work on the server side.

Traceroute

One of the tools that comes to mind when talking about route paths on the internet is `traceroute`. What kind of mechanism can we implement using traceroute?

One possibility is to have each mirror run a traceroute to the client, and among themselves decide which one has the shorter path. There are two problems with this approach:

1. All the mirrors need to be contacted, and work together to make a decision. As the number of mirrors increases, this will take increasingly more traffic/time.
2. This method will give the paths from a server to the client. However, the paths on the internet are not always symmetric. That is, the path taken from a client to a server is not the same as the path taken from the server to the client. Assuming that most of the data transfer is going to be from the server to the client, it would be inappropriate to look at the path in the opposite direction of the bulk of the data transfer.

The other possibility is to run traceroute from the client to all the mirrors. Again, as the number of mirrors increases, the time/traffic to make a decision will increase with the number of mirrors.

Based on the scalability issue and asymmetric nature of the internet, would be an inappropriate mechanism; and that the mechanism we find must be scalable: the time/traffic requirement must not grow with the number of mirrors.

The Solution

The method being implemented in the tools presented in this paper is to use the routing information available from the sources that provide autonomous system path information (such as the Internet Routing Register) to determine, a priori, a ''close'' mirror, that is the mirror reachable by traversing the minimum number of autonomous systems, for all IP addresses by building a mirroring table.

This mirroring table can then be used to make decisions as to which mirror is ''close'' to a client with a given IP address.

Some of the assumptions and implementation issues relating to this solution are discussed in the ''Discussion'' section.

Source Address	Mirror address
8.0.0.0/8	http://www.us.domain.com/public/tool
28.10.0.0/16	http://www.domain.de/publik/
130.36.0.0/24	http://www.us.domain.com/public/tool
130.36.128.0/28	http://www.domain.de/publik/

Table 1: Mirroring table.

Mirroring Table

The mirroring table is a table that specifies which mirror clients should use based on their IP address. This table is similar to routing tables: routing tables specify which interface should be used for sending packets and the next hop, based on the destination IP address of packets. A mirroring table specifies which mirror should be used, based on the IP address of the client.

As an example, consider a primary site at `http://www.us.domain.com/public/tool` (shown as Server 1 in AS 2 Figure 1) and a mirror at `http://www.domain.de/publik/` (shown as Server 2 in AS3 in Figure 1).

Furthermore, let's assume based on the network topology information, we generate a mirroring table, of which a portion looks like Table 1.

The source address is specified using the CIDR syntax [9]. According to this table, client 1 at 8.2.3.4 (in AS 1) would be directed to use the primary site (server 1), based on the first line. On the other hand, Client 2 at 130.36.130.22 (in AS 3) would be directed to use the mirror (Server 2) according to the last line of the table.

Generating the Mirroring Table

The IRR contains information about network routes, and route advertisements that ISP's are willing to send and accept from other ISP's.

One can visualize the information in the routing registry as a graph, by considering the autonomous systems as nodes, and the sending and and the willingness accept route advertisements as arcs between these nodes.

Once represented as a graph, the problem of finding a ''close'' mirror gis reduced to applying Dijkstra's one-to-all shortest path algorithm for each mirror to generate AS paths from each AS to the AS containing the closest mirror.

Once we know which mirror each AS should use, we can enumerate all the routes originating from each AS to generate the mirroring table.

Optionally, the mirroring table can be processed through a route aggregation algorithm, to decrease the number of entries in the mirroring table.

Directing the Clients

The ease of directing a client to a ''close'' mirror for a given protocol depends on whether the protocol supports redirection or not. For example, the http protocol supports the ''Location:'' header which can redirect its client to any other URL.

A http redirector, would take the IP address of the client, look it up in the mirroring tables, find a ''close'' mirror, and return the ''Location:'' header with that mirror.

For protocols that do not support redirection, one possibility is to implement a modified DNS server. The server would be use the IP address of the resolver requesting a name resolution as the client address to find a ''close'' mirror. For an example of a modified name server, see [10]

Implementation

The mkmirrortable tool

The `mkmirrortable` tool takes two or more arguments and generates a mirroring table on stdout (in the format shown above). The arguments are:

- the name of a mirror file, in a /etc/hosts format with addresses in the first column, and a label (such as its hostname, or its URL) in the second column and
- name of IRR database(s) file, in the RIPE-181 format [8].

The execution time of `mkmirrortable` depends on the length of the IRR database files. On the current IRR database, on a Pentium 200 processor, it takes several minutes to generate a mirroring table.

The closest.cgi Tool

The `closest.cgi` is a simple Common Gateway Interface (CGI) perl script, that redirects a web browser to a mirror. This is accomplished via the ''Location: '' header returned by the cgi program.

The client IP address is delivered to `closest.cgi` via an environment variable. The tool then opens the mirroring table generated by the `mkmirrortable` program and searches for all entries that match the IP address of the client. The route with the longest prefix(the most specific route) is selected and its mirror is returned to the web server via the ''Location:'' header.

Discussion

There are several assumptions that have been made in order to use autonomous systems to find a ''close'' mirror and in our implementation.

Definition of "close"

Earlier, we defined a ''close'' mirror, as the mirror that is reachable by traversing the minimum number of autonomous systems. However, this definition is rather arbitrary (and rather self serving for the purposes of this paper).

There is some validity to this definition: The autonomous system paths can be thought of as a simplification of all routes on the Internet. It would be very difficult to map all the routes in the entire Internet. Even then, using this map and doing calculations based on such a map would be extremely difficult. The autonomous system map is a representative derivative.

Moreover, in general, crossing autonomous systems crosses organizational boundaries, and there are monetary costs in crossing organizational boundaries.

Even though the users are not directly aware of it, there are costs associated with crossing network providers: there are the operational costs of exchange points, as well as charges imposed by larger network providers to use their networks. (Of course there are some exceptions, where network providers agree to exchange traffic with each other for free.)

To improve this definition of ‘‘close,’’ one would have to look at the reasons that a site or its users wish to use mirrors.

In most cases, the user does not care which mirror is used to deliver the service, but would like to be connected to a ‘‘up’’ mirror with high bandwidth and/or low latency depending on the service.

The owners of the site, on the other hand, may wish to redirect users to mirrors, to conserve bandwidth at their own site, or to manage the distribution of users among mirrors.

From an Internet architecture perspective, mirrors can serve two purposes: to decrease the overall bandwidth used on the internet, and to avoid some paths becoming overloaded.

A better definition of ‘‘close’’ would need to take into account the purposes of all of these parties.

AS Paths Represent Network Routes

We have assumed that the network route from a client to a mirror will pass through the autonomous system path that is constructed by looking at the autonomous systems that claim to originate the routes for the client and the mirror.

This is most likely true for most routes in the Internet. Almost all network providers represent their network as autonomous systems, and present a coherent view of their network to all other networks at the exchange points. Moreover, since the routing tables at the exchange points are built from the Internet Routing Registry, the coherent view presented by each network provider is stored in the IRR.

An exception to this would be a private network connection between two network providers that does not pass through an exchange point, and is not represented in the IRR.

Different AS sizes

We have made an implicit assumption that all ASes are of the same size, by ignoring the size of ASes. We calculate the number of ASes traversed without regard to how large they may be.

In reality, the size of autonomous systems vary greatly. A better approach would be to try to estimate the size of each AS, and try to minimize the total sum of the sizes of the ASes traversed.

AS Cost of Routes Ignored

One of the pieces of data available in a AS definition in the IRR is a cost associated with routes learned from neighboring autonomous system.

In this implementation, this cost is ignored. It would be difficult to take this into account however, since the cost is only relative to other neighbors for the same AS, but has no significance when going across ASes.

Asymmetric Internet and Direction of Data Flow

Earlier in this paper, we talk about the asymmetric nature of the Internet. We have assumed that the direction of data flow is from a mirror to the client; and have ignored that there is some data flow from the client to the mirror. It is possible to think of services where there is equal or more data flowing from the client to the server (e.g., sending e-mail).

A better method for choosing a ‘‘close’’ mirror would look at the data flow requirements in both directions to try to find an optimal path.

Future Directions

This section discusses several possibilities for extensions to this work.

Empirical Evaluation

An extension to this paper and a well contained project would be to implement the mirror selection method presented in this paper at several sites, and collect data on the selections made. This data can then be analyzed to see how useful this method is and how valid the assumptions being made are.

Client-side Solutions

Another approach would be to look at solutions where the client is involved in the selection of a ‘‘close’’ mirror, either solely at the client, or in cooperation with a server. This could be implemented in a browser downloadable language such as Java or Javascript.

Dynamic Route Information

This implementation uses static information in the Internet Routing Registry. Using dynamic routing information by peering with routers using BGP-4 would make it possible to take into account link states between providers in choosing a mirror.

Modified DNS Server

For services that do not support automatic redirection, using the domain name service as a redirector may be possible. A modified DNS server could respond differently depending on the client. The IP address returned by the nameserver would be the address of a ‘‘close’’ mirror to the client that requested the address. This would work based on the assumption that the resolver requesting the address is in the same autonomous system as the client requesting the service.

Migration to RPSL

There is a plan in the Internet Registry to migrate to a different route specification language called RPSL. The `mkmirrortable` program would need to be extended to use RPSL instead of RIPE-181.

Using Latency and Bandwidth Availability Metrics

A better ‘‘mirror’’ selection, from the client’s perspective would be to find a mirror with a low latency and/or the highest possible bandwidth. Even better, would be to be able to reserve the bandwidth from a specific mirror.

Conclusion

This paper describes a method and presents two tools developed to locate a ‘‘close’’ mirror among replicated copies of a network service by utilizing network topology information based on autonomous systems.

The author hopes that this breakthrough will be embraced by most ftp and web service providers, and that he does not have to ever ‘‘choose his own’’ close mirror again :-) .

References

[1] Huitema, Christian *Routing in the Internet,* ISBN 0-13-132192-7, Prentice Hall, 1995 Englewood Cliffs, New Jersey 07632.

[2] *RFC1771 – A Border Gateway Protocol 4 (BGP-4)*.

[3] *RFC1812 – Requirements for IP Version 4 Routers*.

[4] http://www.merit.edu/radb/docs/irr.html .

[5] http://www.isi.edu/div7/ra/ .

[6] http://sunsite.org.uk/packages/mirror/ .

[7] http://www.cisco.com/warp/public/751/distdir/dd_wp.htm .

[8] Tony Bates, Elise Gerich, Laurent Joncheray, Jean-Michel Jouanigot, Daniel Karrenberg, Marten Terpstra, Jessica Yu, *Representation of IP Routing Policies in a Routing Registry*, http://www.merit.edu/radb/docs/ripe-181.html .

[9] *RFC1519 – Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*.

[10] Schemers, Roland J. III, ‘‘lbnamed: A Load Balancing Name Server in Perl,’’ *LISA ’95*. http://www.usenix.org/publications/library/proceedings/lisa95/ .

[11] *RFC2280 – Routing Policy Specification language*, ftp://ftp.isi.edu/in-notes/rfc2280.txt .