Request Routing in Cache Meshes

Christian Grimm, Marc Neitzner, Helmut Pralle, Jens-S. Vöckler[1]

University of Hanover
Institute for Computer Networks and Distributed Systems
Schloßwender Straße 5
D-30159 Hannover

{grimm,pralle,voeckler}@rvs.uni-hannover.de

Abstract

The current document concerns the inclusion of routing informations into neighbor cache selection algorithms. The background for this work is derived from experiences in the DFN Caching Project. During this project, 10 cache servers were installed at the central nodes of the German Broadband Research Network (B-WiN) as an integral part of the backbone infrastructure.

The fact that B-WiN contains several exchange points to other networks aggravates an intelligent and effective domain name based routing of requests within this cache hierarchy. Especially exchange points in Frankfurt and Hanover to other national networks - all containing webservers in the .de domain - demonstrate the insufficient possibilities in common implementations of neighbor selection algorithms.

The proposed solution consists of an extended caching software to gather routing informations by whois requests. These informations can be evaluated in the cache selection algorithm. Starting with basic design issues, the main part of this paper is built by a detailed description of an extended neighbor selection algorithm. It is concluded by the description of a full implementation in the squid cache software [1].

Keywords: request routing cache mesh DFN B-WiN

Table of Contents

1 Introduction

This section starts with an introduction into the DFN Caching Project. The current conception and into the reasons for employing a cache. Afterwards the technical environment and administrational coordination in the current project are detailed.

1.1 The DFN Cache Project

To counter the effects of increasing WWW traffic in B-WiN with overloaded links and congested networks the project *Conception of a Caching Infrastructure in the German Research Network* was initiated in August 1996 by DFN Association. The main goals of the project were to

- coordinate a caching infrastructure among sites connected to B-WiN,
- clarify the potential of a top level cache hierarchy in B-WiN, and
- establish a mesh of top level cache servers as an ingetral part of the infrastructure in the backbone of B-WiN (the DFN cache service).

From August to December 1996, 10 SUN Ultra-2 server were installed at the central nodes of B-WiN and connected to their respective Cisco router. The free Squid software was selected to run on these machines as a caching process. In January 1997, all DFN Caches were ready for operation and announced to the interested user group of peer cache administrators.

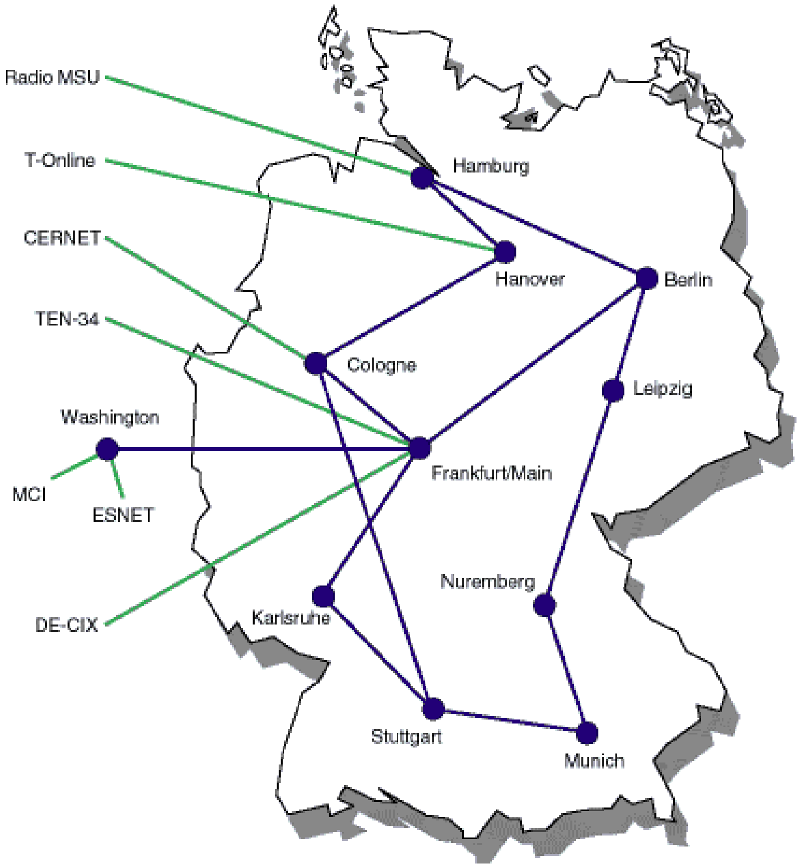

Figure 1: Map of German Reserach Network

Figure 1 shows a map of B-WiN (blue lines). The DFN cache servers are located at the 10 cental nodes across Germany. All B-WiN nodes inside Germany participate in a ring structure employing 34 Mbps ATM technology. A few additional cross connections of the same capacity add a partial mesh structure. The capacity of some selected links was updated to 155 Mbps during the last months. The interconnections with other networks were significantly improved by dedicated high capacity links (green lines). At the writing of this document the B-WiN includes

- a dedicated 90 Mbps link to the US internet with an own point of presence in Washington D.C.,
- a gateway to the european research network backbone TEN-34 at 45 Mbps,
- a gateway to most commercial providers in Germany (DE-CIX) at 34 Mbps,
- a dedicated gateway to the largest commercial provider in Germany (T-Online) at 20 Mbps and
- access to several other national research networks, e.g. in Russia or to the science network of the People's Republic of China.

1.2 Conception

The initial configuration in the DFN cache mesh started out by distributing various toplevel domains evenly all over the ten DFN caches. For example, the DFN cache in Frankfurt was configured as a dedicated cache for the .com domain and the DFN cache in Berlin was configured for the domains .cz, .hu, .pl, .ru, .si, .sk, .su an .tr. The idea behind this concept was to achieve a sort of load balancing and traffic among the DFN caches. Soon, several essential drawbacks and the necessity of an improved conception became obvious [2].

In October 1997, a redesign of the DFN Cache Mesh was established. Within the new conception, respect was given to the fact that about 50% of all WWW traffic originated from webservers in the .com domain. To keep the new configuration simple, all DFN Caches were evenly distributed between the two logical domains .com and !.com. Starting at Cologne as a !.com cache,

A further benefit from the new scheme was the avoidance of some parent relationships among the DFN caches. All caches within one group only maintain a sibling relationship to their neighbours. Thus the number of hops from an enduser to the origin site was reduced to two levels. Currently, no interconnections between the two groups of caches exist. Figure 2 illustrates the different levels of the current cache hierarchy in B-WiN.

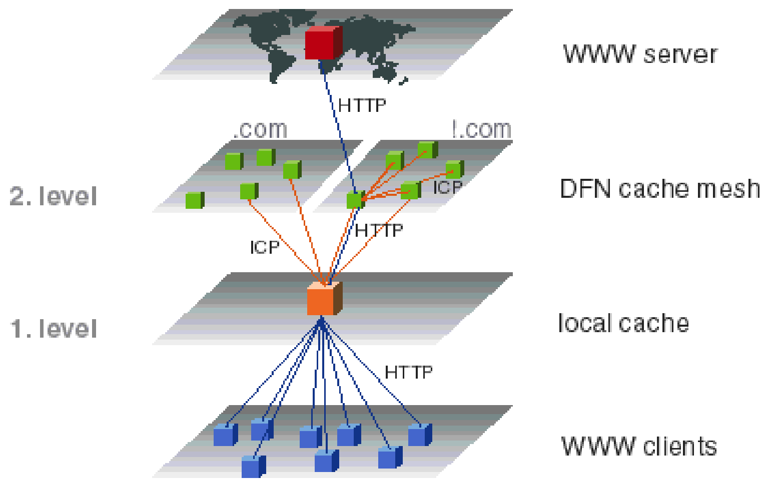

Figure 2: Current cache hierarchy in B-WiN

2 Motivation for Cache Routing Improvements

Though the current DFN cache mesh has a well balanced and stable condition today, several limitations can be observed. One major issue is caused by the lack of routing capabilities in current cache implementations. Actually, most of the routing mechanisms in cache software like Squid is based on domain names. There is no question that this approach is very pragmatic, because extracting and parsing of webserver names from URLs is easy to implement. To illustrate the motivation for cache routing improvements, this section shows the routing situation in B-WiN and the resulting limitations for a domain based cache routing.

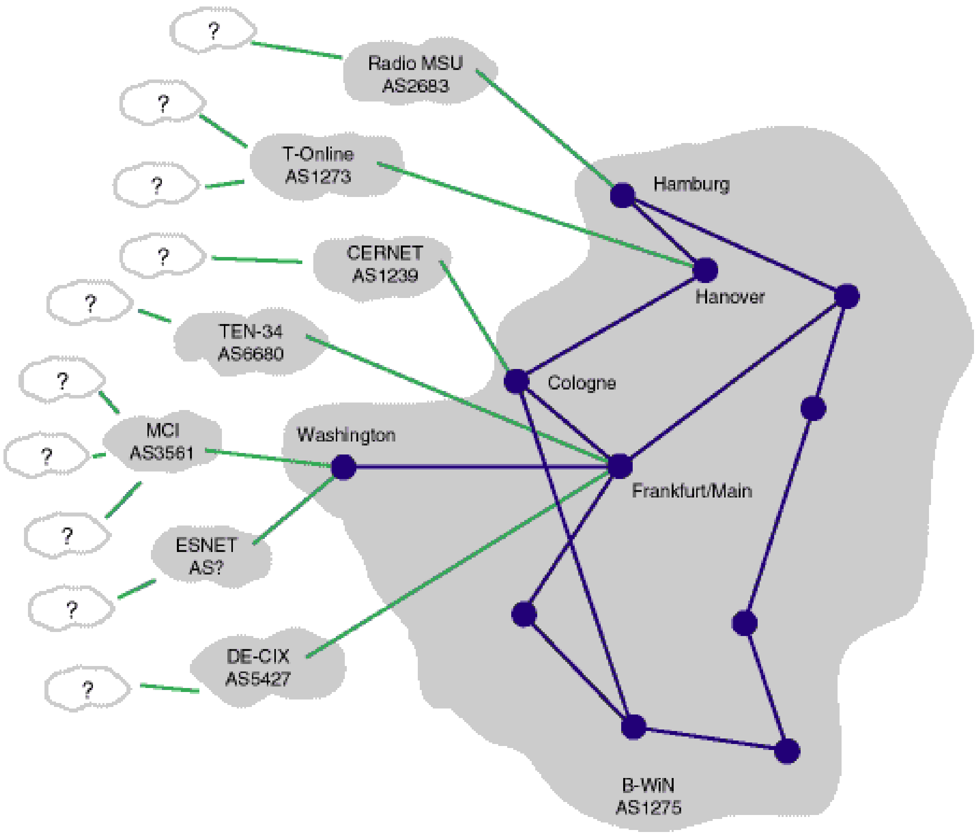

Figure 2: B-WiN and neighboring Autonomous Systems

Figure 3 shows B-WiN and its neighboring networks. In contrast to figure 1, here attention is directed to the exchange points and AS numbers. A deeper look shows that german commercial networks are connected to B-WiN at two different locations, first DE-CIX in Frankfurt and T-Online in Hanover. It is obvious that lots of webservers in the .de domain are located behind both exchange points. Because of the limitation to **domain** based cache routing, it is impossible to set up a dedicated DFN cache in Hanover to serve for all objects in the T-Online network and a dedicated DFN cache in Frankfurt to cache objects behind DE-CIX. However, this setup would provide the most effective way to preserve bandwith and decrease delay. The scenario does not only occur in DFN cache mesh and its special configuration. It is adoptable to all larger networks with more than one exchange point and a multi level cache mesh.

To accomplish a more intelligent routing of cache requests, knowledge about network number and AS number relations has to be introduced into cache selection algorithms. Therefore, further research into real cache **routing** algorithms is needed. The following sections describe our current position in this area, starting at basic design issues up to a working implemention of squid software.

3 Gathering Routing Informations

At the beginning of this work, two major principles of gathering routing informations were discussed:

- listening to Border Gateway Protocol (BGP) messages and
- using Whois services.

Both methods have certain advantages and shortcomings. This section briefly illustrates the most relevant points on BGP versus Whois. In the following paragraphs, an explanation is given why BGP was ruled out early in experimentation and a decision was made towards whois.

3.1 Border Gateway Protocol

The Border Gateway Protocol [3], [4] is mainly used between routers of different networks. The purpose of BGP is to exchange informations about network information and related AS information. BGP traffic can be characterized as a current stream of data which consist of update messages. There is no request/response scheme in BGP like in common client/server protocol implementations. The opposite is true, BGP is based on an active sending and passive listening processes.

The investigations into BGP showed some fundamental arguments not to follow this approach any further. The major reasons are:

1. **Router configuration**
   To send BGP messages to a cache, the routers have to be configured at least with the cache ip address. This work may be affordable for single caches, its not reasonable in large networks with lots of caches. Generally, the router administrator or the NOC has to be contacted to enable this functionality.

2. **Amount of data**
   A cache receiving BGP messages has to deal with an additional amount of data. It has to collect as much as possible information out of these data. It is foreseeable, that a the majority of this data is of little use to the cache and its operation.

3. **No specified requests**
   BGP doesn't provide a request/response scheme. Thus it is not possible to request a router for specific routing informations on a given webserver address. A cache has to wait for the appropriate information in a BGP message stream. The listener has little or no influence on arrival.

4. **Security considerations**
   Security considerations might prevend router operators from configuring numbers of BGP listeners in their routers. No further investigations on this point was done in this paper, but compared to DNS and other services, spoofing of BGP messages might be an issue.

3.2 Whois Service

In contrast to BGP, whois is an ASCII based client/server protocol [5],[6]. Several whois servers throughout the Internet provide network number to AS number resolution among other informations. Acccess to most of these servers is free. Whois does fit our needs more appropriately. With simple requests whois servers can be asked for the required data. No extra admins or operators have to be consulted.

Of course, whois also has some drawbacks. It is not essential for the operation of a network to have a valid whois database entry. In other words, whois entries are on a voluntary basis. Thus the quality of whois databases depends highly on the willingness of network operators to provide their information. Also, whois database engines vary from server to server. There is almost no communication or even automatic database balancing among whois servers. Requests to different servers showed that the RIPE whois server (whois.ripe.net) provides a rather complete view on european networks. However, the MERIT whois server (whois.ra.net) has a better view of networks in the US.

4. Introduction into Design Principles

Assume a request for a document on `www.sun.de` at a cache. The essence of AS routing is to find the next neighbour cache closer to the origin site. In order to follow this decision, a cache has to obtain some knowledge about the path to the destination.

To get the path to the origin site, that is routing information, the tool `traceroute` can be used. Some modifications to traceroute allow - with the help of a whois server - the output of the associated autonomous system numbers. Table 1 displays the output from such a trace.

```
cache $ traceroute -A -h whois.ra.net www.sun.de.
  ...
 1  gate-s5-1-IP5.rrzn.uni-hannover.de (130.75.5.250) [AS1275] ...
 2  BWINgate.rrzn.uni-hannover.de (130.75.1.253) [AS1275] ...
 3  Uni-Hannover1.WiN-IP.DFN.DE (188.1.4.5) [] ...
 4  ZR-Hannover1.WiN-IP.DFN.DE (188.1.4.1) [] ...
 5  ZR-Koeln1.WiN-IP.DFN.DE (188.1.144.26) [] ...
 6  ZR-Frankfurt1.WiN-IP.DFN.DE (188.1.144.34) [] ...
 7  cix-frankfurt1.WiN-IP.DFN.DE (188.1.164.14) [] ...
 8  DE-CIX.DE-CIX.ecrc.net (194.31.232.33) [AS5427] ...
 9  e5-1-r1-FFM2.ecrc.net (195.27.83.105) [AS1273] ...
10  hssi9-0-r2-MUC.ecrc.net (194.59.191.101) [AS1273] ...
11  fddi9-0-r1-MUC.ecrc.net (194.59.191.1) [AS1273] ...
12  W3-GW.ECRC.DE (194.112.80.70) [AS1273] ...
13  www.sun.de (194.221.99.176) [AS1273] ...
```

Table 1: traceroute to `www.sun.de`.

The authors are aware that the repeated injection of trace PDUs into foreign networks is excessive. For the time being and the sake of our discussion, let us assume that we may safely inject such PDUs without penalty. Indeed, the algorithms shown at a later time try to avoid such PDUs as much as possible.

After tracing the currently active route, the cache has to simplify the vast amount of data in order to arrive at an educated decision. There are two levels of abstractions applicable.

1. The first level of abstraction determines the network class of the hosts and routers found. More often than not a traversed backbone contains routers in the same network class number. Thus the network path is shorter than the original host trace. Subnetting need not be feared, because networks are allocated at least in network class sized in chunks to an autonomous system.

The determining of the network class from a host address is a simple AND calculation at almost no cost for a host.

2. The next level of abstraction is more difficult, as the networks are further condensed into the autonomous systems. Each traversed networks is thus shrunk into a single point along a path of AS.

   To determine the AS number of a given network, `whois` lookups are necessary. As shown in table X, it is not always possible to look up the AS number for any given network. Depending on the algorithm chosen later on, an educated guess can in some situations provide the correct number.

   The `whois` lookup is expensive in time and consumed bandwith. Therefore, all lookup results must be cached. Due to the basic assumption that a network does not change an AS very often, a long TTL can be safely attached to its AS cache entry.

The result of the different abstractions is shown in figure 4. Of the previously thirteen hops to `www.sun.de`, only seven networks are passed. Furthermore, the networks can be condensed into just three AS: the AS of the requester, a traversed AS along the way, and the destination AS.

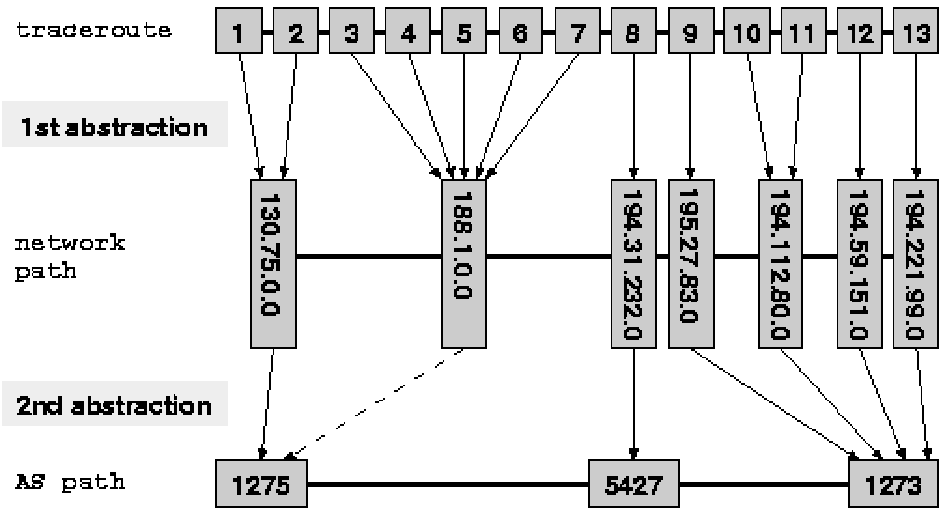

Figure 4: abstractions while getting AS path between cache and origin.

As usual, all the principals for caching still apply. With the knowledge of networks and AS numbers, the next cache along the way can be selected. If there is no closer cache, the origin site might be queried directly. Since the next cache along the way is to be selected, this will be one closer to the border into the next AS, or the first one within a bordering AS. Finding previously unknown caches in foreign networks looks desirable, too.

Caches farther away are of little or no interest, because they might defeat the caching principals. Usually, you set up a cache on your site of a slow link; compare the guidance advice in EU project DESIRE *web caching system design checklist* [7] - a direct result of the locality principle.

If a trace if followed to its origin site, the resulting AS path is the foundation of the proposed algorithms. Please keep in mind that from combining the common parts of different resulting AS paths, a depth first search tree could be constructed which represents the cache's view of the internet. The Internet as whole would be viewed as a graph of meshed AS!

Due to the nature of ever changing routes, temporarily down router and other obstacles, the tree can never be up-to-date. At this stage, measurements and experiences are missing, but it is safe to assume that the time-to-live (TTL) of tree links need to be exceedingly low.

Therefore, the complete capture of an AS path to the origin site is beyond feasibility. Rather, all implemented approaches shown later in this paper limit their knowledge to their bordering AS. The external connections to the local AS can easily be checked and don't change very often.

5. Implemented Approaches

In order to reduce the amount of whois lookups, all networks are associated with their respective AS number and cached. This scheme allows for a very simple AS based routing. Furthermore, the AS number of the bordering AS used to reach the origin site is also associated with the destination AS number. The additional structure allows for a second, more sophisticated routing by AS numbers.

1. **selection by destination AS numbers**

   An elementary algorithm with little overhead tries to determine the next cache or caches just by the AS number of the origin site. Only one AS resolution for the origin *network* is needed for each request not already in the AS cache. No additional trace PDUs will be generated with this approach.

   The major drawback of this solution is the manual configuration of each origin. Unconfigured AS must use the original neighbour selection scheme. Still, for small networks a *negative* entry (anything but AS such and such) also might be feasible. For example, most hierarchical cache meshes can obtain documents from webservers within the own AS faster by direct access.

   The manual AS-based selection is sucessfully implemented.

2. **selection by border AS numbers**

   With a limited knowledge of the path towards the origin site, a semi automatic selection of a cache or caches closer to the origin can be generated. This first scheme still applies. Thus the request starts by issuing a `whois` request for the origin site - if not already found in the AS cache.

   In the next step, the cache starts to send trace PDUs until the *own* AS is left. It is mandatory to stop at the borders of the own AS, otherwise the wrath of the foreign net-admins will descend for sure. As shown previously, a complete AS path to the origin site is not necessary. Rather, the border AS number (BAS#) is associated with the origin site AS number (DAS#). The resulting *path snippets*, some kind of partial path, can be kept long enough to rule out excessive traces.

   With this approach, only the best caches for all bordering autonomous systems need to be configured. AS beyond are automatically associated with the cache along the way.

   The semi-automatic selection is currently being implemented.

Further possibilites are not being implemented, and they will be discussed in the second *possibilities* section as a look-out and things-to-do.

6. Implementation Issues

The integration into the squid cache software needed modifications in the dispatcher. Otherwise, squid was kept intact as expected from an extension to its services. Squid itself was model for some of the chosen approaches.

6.1 Integration into Squid's dispatcher

Whenever a clients request an URL from a cache, and the cache experiences a local MISS, the URL causes a hostname resolution. The squid internal resolver tries to obtain the name from the IP cache within the squid process. If the host is not in the IP cache, the request will be queued with one of the external dnsserver processes.

In the next step, the AS based selection scheme is inserted before the original neighbour selection scheme. The original scheme is kept as a fallback for several cases either not caught or not configured in the AS based selection scheme. It constitutes the default selection scheme.

The AS based selection scheme works with both algorithms described earlier. It will return a possibly empty set of eligible caches, which are supposed to be closer to the origin site. The scheme itself enlists the help of external helper processes.

6.2 Mechanics of the external helper

The whois queries for AS resolution take considerable time. All in all, the effects are similar to DNS resolving, but there are several drawback when compared to DNS:

- There is no hierarchical structure to `whois` base servers.
- Recursive queries have to be client implemented.
- There seems little cooperation between the different `whois` servers.
- There are much fewer `whois` servers.
- There do not exist any *cache-only* `whois` server, though index server provide forward knowledge.
- During busy times, long lookup delays are to be expected.

The querying instances are externalized like the `dnsserver` process. The resulting `asserver` process is shown in figure 5. Due to the fact that squid needs the network class to AS number map for the first AS selection scheme, the cache has to be kept within the squid process. This *AS cache* is similar to the IP cache from the DNS part of squid.

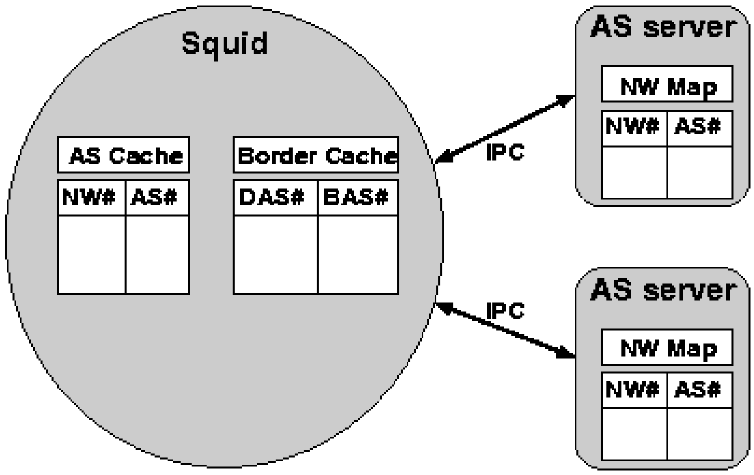

Figure 5: Integration of AS servers into Squid.

The non-blocking AS servers are started externally during the startup phase of squid. The first approach shown here uses an unfortunate partitioning of functionalities, thus each AS server needs a map of network classes to AS numbers, too. Nevertheless, all results reported from any external AS server will be cached within the squid process.

Please mind that the NW map does not constitute a cache: there is no lifetime or expiry, as we expect the AS server lifetime a sufficient guard on the NW map entries. A radix tree allows for compact representation with fast retrievals.

6.3 Neighbour selection by AS numbers

The basic algorithm underlying the AS neighbour selection scheme works in the following way:

1. A request triggers the AS neighbour selection scheme. The network class of the request is checked for in the AS cache. If found, the next step is executed. Otherwise an idle external AS server is asked for the associated AS number. If all AS servers are busy, the network request is queued. Either way, squid will continue with the original neighbour selection scheme for the current request in order to yield speedy answers.

2. If the AS number is found within the AS cache, a trace might be started, depending on several conditions.

    - If the AS number is configured in neighbour selection scheme 1, the associated cache or caches are returned to the dispatcher. No trace is necessary.
    - If the AS number has an associated border AS number, the cache(s) associated with this border AS will be returned to the dispatcher. This constitutes scheme 2.
    - If the border AS number is not in the cache, a trace needs to be initiated.

    If a trace is started, the request will be processed by the original neighbour selection scheme, because the trace will take time.

3. If a third request goes to the same network class as a previous one, all necessary data from the previous steps might be in the AS cache and border cache. Thus the request can be processed by the AS neighbour selection scheme without any lookups or traces.

4. If a fourth request goes to a not yet known network, which is later found to be in a known AS, the AS number is stored in the AS cache. A subsequent request for this network can *then* be answered by the AS neighbour selection scheme.

The entries into the AS cache can have several states similar to the IP cache entries, *pending* and *unresolvable* among them. Since AS numbers are not expected to change often, the entries in the AS cache can have a long lifetime. Further reduction in `whois` lookups can be obtained by storing the AS cache during squid termination and restoring the entries during squid initialization. The latter might even be used for the internal map of the AS server.

Further savings are possible by pre-initialization of the AS cache and AS server map during startup with all networks of the own AS number. Thus, the AS server needs almost no extra lookups during its traces. On one hand, the initilization can catch changes in the network, if the `whois` information were up-to-date. On the other hand, only the `ra.net` server(s) seems to be able to generate this kind of information. For larger autonomous systems, the resulting information is of considerable size, and might contain many exotic entries never encountered with caching.

The border cache associtions resemble in a way the *path snippets* shown earlier. Therefore, only a small TTL is allowed for this information. On the other hand, the connections to other AS usually do not change rapidly. Furthermore, only the shortest possible element of a path is kept, thus allowing for larger TTLs.

The radix tree for the network map provides rapid access to its contents. A maximum depth of 24 for class C networks can be expected. Due to the digital search nature of radix trees, a periodical scan for leaves with the same information seems feasible. Such leaves can be combined into the next inner node, reducing the memory consumption and speeding up lookups.

7. Open Issues and Things to do

The current section looks at possible extensions to the chosen selection schemes and different views. Some ideas are introduced, but proven unfeasible. Figure 6 shows the abstract relation between five autonomous systems used for illustration purposes.

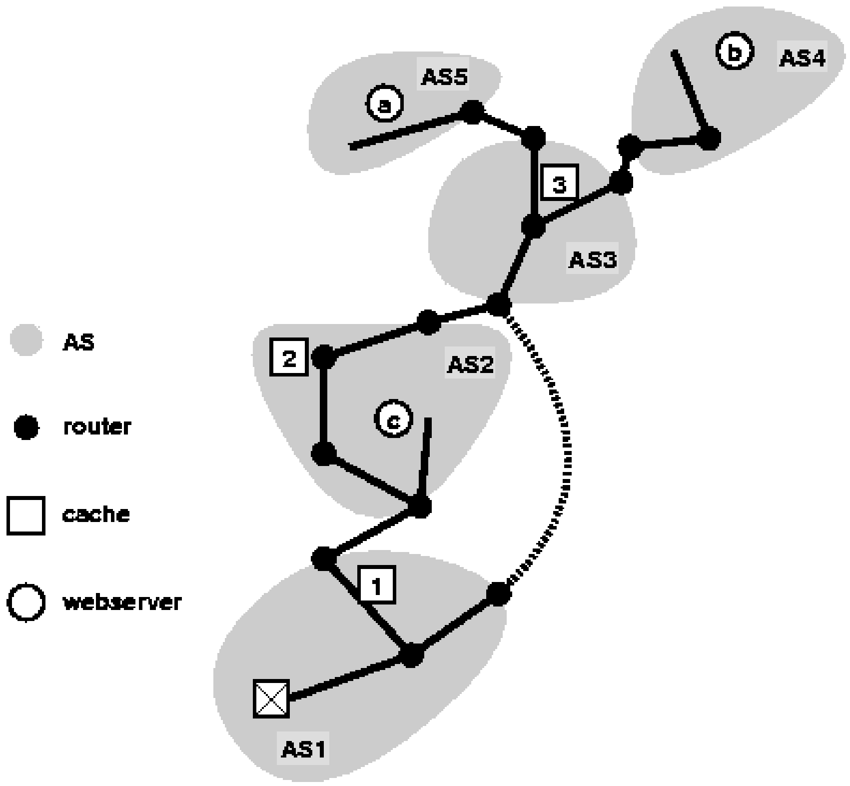

Figure 6: sample network for discussion purposes.

The crossmarked cache is the asked for a document on webserver b. With regards to figure 6, cache 1 should be selected. If AS 2 is the only bordering AS, cache 1 would be selected in almost all situations. The possible exception are webservers from AS 1, which can be reached faster by direct access.

As cache 1 is selected, the request moves one level closer to the origin site. The current dynamic selection scheme exits its traces as soon as AS 2 is reached. Cache 2 would be configured on cache 1 as parent for everything *through* AS 2. But keep in mind that webservers like c are often located way off the backbone routes. We are talking networking dimensions, of course. If cache 1 would trace requests to its destination, it would create the flood of trace PDUs. The effects and drawback to such an approach were described earlier.

Let us return to the introductory *distant* cache selection. As this paper shows, a selection by AS numbers is more likely to produce results than the selection by toplevel domain. Distant caches could be configured using the AS selection scheme 1. Although such a manual configuration is easy to do, it is typing intensive and error prone. Looking at figure X, how does cache 1 get to know about cache 3 without manual configuration?

- One possible solution is an extension of ICP [8],[9] where caches announce their willingness to accept requests for certain AS numbers. The approach is similar to the SDP protocol used for multimedia conference announcements. Similar ideas, though not for AS numbers, were discussed at the TERENA TF-Cache meeting in 9’97.

Multicast should be used and a low bandwidth employed with similar strict rules as described in SAP. Care must be taken against cache announcement spoofing attempts.

If the cache announces its own AS number, it will behave like an accelerator for a large server set. Since there are no immediate benefits visible to commercial ISPs, the idea is deemed stillborn.

- Another approach might extends ICP in a way that caches talk to each other and exchange certain willingness information. With ACLs, the discussion might or might not be opened for outside caches.

8. Conclusion

The last paragraphs dwelve into ideas which are rather academic at this point of the project. The current conclusion is that the configuration of distant caches goes against the principal of localility. A more robust operation can be obtained, if each cache only knows a little, and at maximum its bordering autonomous systems.

9. References

[1] D. Wessels et. al.
Squid Internet Object Cache, Apr. 1998
http://squid.nlanr.net/

[2] C. Grimm, H. Pralle, J.-S. Vöckler
Load and Traffic Balancing in Large Scale Cache Meshes, to be published

[3] Y. Rekhter, T. Li
RFC 1771*A Border Gateway Protocol 4 (BGP-4)*, Mar. 1995
ftp://nic.merit.edu/internt/documents/rfc/rfc1771.txt

[4] Y. Rekhter, P. Gross
RFC 1772*Application of the Border Gateway Protocol in the Internet*, Mar. 1995
ftp://nic.merit.edu/internt/documents/rfc/rfc1772.txt

[5] C. Weider, J. Fullton, S. Spero
RFC 1913*Architecture of the Whois++ Index Service*, Feb. 1996
ftp://nic.merit.edu/internt/documents/rfc/rfc1913.txt

[6] P. Faltstrom, R. Schoultz, C. Weider
RFC 1914*How to interact with a Whois++ mesh*, Feb. 1996
ftp://nic.merit.edu/internt/documents/rfc/rfc1914.txt

[7] *Web caching System Design Checklist*, Mar. 1997
http://www.uninett.no/prosjekt/desire/arneberg/you.html

[8] K. Claffy, D. Wessels
RFC 2186 *Internet Cache Protocol (ICP), version 2*, Sep. 1997
ftp://nic.merit.edu/internt/documents/rfc/rfc2186.txt

[9] K. Claffy, D. Wessels
RFC 2187 *Application of Internet Cache Protocol (ICP), version 2*, Sep. 1997
ftp://nic.merit.edu/internt/documents/rfc/rfc2187.txt

---

[1] This work is supported by the German Academic Network Organisation (DFN-Verein) with funds from the Federal Ministry of Education, Science, Research and Technology (BMBF).