Squash: Design and Implementation of a Large Scale HTTP Gateway and Masquerader

Behdad Esfahbod   Hossein Safy Allah
behdad@behdad.org   hossein@bamdad.org
Computing Center
Sharif University of Technology

July 4, 2003

Abstract

Paying for information on the web is a usual scenario these days. There would not be much problem if each user is going to pay for her own needed information, but most of the times, organizations are going to pay for information needed by they users. These organization themselves may charge the higher academic body for the what they spent in the process. Setting access rights and accounting for each single user's access to each information database consumes a huge amount of time and human resources. In this paper we focus on the solution we have designed and implemented to overcome the problem, by setting up a central access point that both users and information databases would interact with. After discussing the design and its benefits, we focus on the implementation practice we have done, the base components used, and scalability issues, and finally propose some of the ways that the system can be developed to overcome the current limitations.

1 Introduction

The growth of Internet and the web technologies in the recent years, has affected many different aspects of today's people life. From regular mail, newspapers, magazines, television, radio, and telephone, to shopping, computer games, and business, all have reformed to make proper use of the Internet. Among them, what has affected the academic environments, is the great pool of information and knowledge shared by people all over the world, and available through the wires connecting them to the cloudy network.

Information available on the net can be divided in two main categories. The first category is those information distributed among different websites and servers that most of them can be reached with modern rich search engines in a minute or two. The second category is the information gathered in central information databases (infobases) which contain a huge amount of information in a wide variety of topics and age, and available and searchable through the website representing the infobase. Infobases usually contain the latest technology and information on topics they cover, by providing up to date research papers, journals, and conference proceedings. These are the essentials that an academic environment should be fed to keep itself updated and produce new science. The number of infobases is greatly limited comparing to the providers of the first category. Moreover, the infobases that a small environment may be interested in are usually limited to a few.

2 Problem Definition

In the common experience of using infobases, there are a few *infobases*, lets say up to a hundred ones, and a greater number of *users* that want to use the infobases. Here we contrast on the users from a geographical region, e.g. a country. Each user is a member of some academic environment that may be a university, a research laboratory, or another kind of department. We call the users of each of these environments a *group*. The process of requesting and receiving information is mostly done on top of the common web technology named the Hyper Text Transfer Protocol (HTTP).

Infobases sell information to the user on a per request basis, or to larger groups, based on the number of needed resources (articles). The later is the preferred way for the infobases, and will break the price many times!

Authentication is done in one of two ways, or both of them: username/password and IP-based.

The traditional way of acquiring information from infobases is shown in Figure 1, that each user sends a request for what she needs to the infobase, and the infobase responses with the requested information. There may be three different scenarios:

1. Each user buys her own information. Usually there is no need for authentication, as she enters payment information each time she buys some document.

2. Each group pays for a large number of documents, and then allows its users to get what they need. Au-

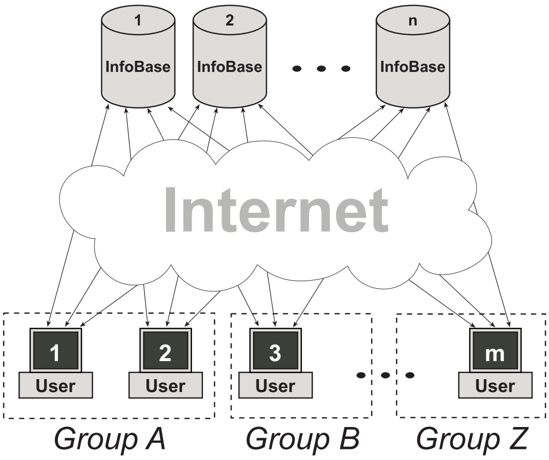

Figure 1: Traditional way of acquiring information from infobases

thentication is needed, to allow users from the group, and deny others, from using group's resources. The group should ask each infobase to set up some IPs and/or username/passwords, for users of the group.

3. The whole region pays for a larger number of documents, and then the groups are allowed to use their quota. This way a single entity communicates with the infobases, but the same entity should communicate with groups, to gather each group's IP addresses and username/password pairs, and send them all to the infobases.

The main problems with these methods are:

- Users must authenticate on each infobase separately, in case of username/password authentication.
- In case of group or region registration, many users should share a single password, as infobases do not register many username/password pairs for a single customer. Sharing passwords, means that a password leak is quite probable, and then the password for many users would change, and users should be notified of the new password.
- Same document may be bought several times by different users of the same, or different groups.
- There is a trade-off between the number of documents a customer pays, and the price of a single document. The difference in price not ignorable in any sense, as a single document for an unregistered user may cost around $30, but the same document, when accessed via a registered region of around 100 groups, will cost as low as $2! On the other hand, when the number of users that a customer has, grows, it may take up to months that the single users or groups, report their username and passwords pairs, and IP addresses, and the customer reports them to each infobases, and the infobases put them in effect. This delay can consume some months of a twelve month membership period.

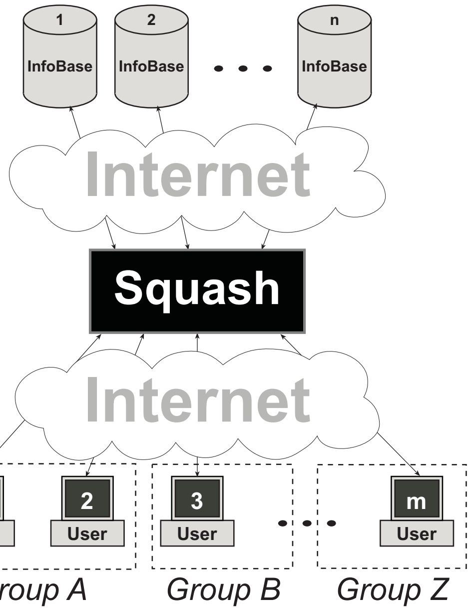

Figure 2: Proposed way of acquiring information from infobases

3 Our Solution

To overcome the problems mentioned, we have designed a central system as a access-point and gateway for users of the whole region to access to the infobases. This method is shown schematically in Figure 2.

Some characteristics of the system, not shown in the figure are listed below:

- Squash hides the real user from the infobase. Infobases just see the Squash system requesting documents.
- Users authenticate on the Squash system.
- Squash is responsible for authentication on the infobases.
- Squash manages users' permissions, and does the accounting.
- Squash manages a cache.
- Squash's configuration is done via web, and each user or group administrator, can configure her settings online. The configurations take effect in a few hours.

There are many ways the groups and the users benefit from this design, most important ones are:

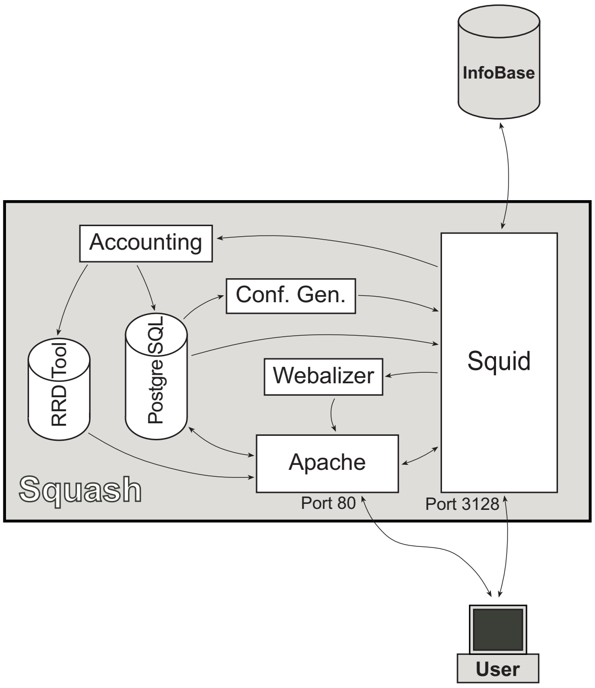

Figure 3: Internal components of the Squash system

- Users need to authenticate once on the Squash system, from then, they can access all infobases they are allowed, without any authentication.
- The cost of a document is minimized, as the whole region is registering as a single customer.
- The delay is minimized, as the user side configuration is done via web by each user herself, or group administrator. The infobase side is also done very fast, as a single username/password, or IP address suffices.
- An efficient cache can be set up to reduce the total cost. As all the traffic is coming from a few number of infobases, caching can be efficient.
- The region can set up a fast Internet connection to the infobases, and users and groups can benefit from fast local backbones connecting them to the Squash system.

4 Implementation

The Squash system is implemented on a personal computer using the RedHat Linux operating system. A powerful PC with a tailored version of the servers can handle thousands of request a second. The internal components of Squash are briefly described in this section, and schematically shown in Figure 3.

4.1 Apache HTTP Server with PHP

PHP is used as the web programming language. The web application is responsible authenticating users for configuration purposes, and also as a portal to the infobases. The system administrator, group administrator, and end users, can login and set their configuration here. The entry point of the site is served as HTTP, it then identifies the user from her IP address, or otherwise, redirects her to the HTTPS secure pages to login.

4.2 XPage Database Oriented Web Application Design System

The XPage system by *Mohammad Reza Mahjourian* is tailored to suite our needs. XPage is used to design the web user interface. XPage is a tool to generate web pages from XML meta definitions.

4.3 PostgreSQL Relational Database

The database server is used to store login information, access permissions, and accounting data. The web application is responsible for viewing, editing, and deleting this data. PostgreSQL is used as it has native Unicode support which is needed for Persian computing.

4.4 Squid HTTP Proxy Server

Squid is the engine of the Squash system, and the reason it is named so. Here Squid is set up as HTTP proxy, cache, authenticator, and anonymizer. User just set her browser's proxy setting to point to this Squid, also enters the username and password, if any. After that, Squid gets the request from user, hides the user from the request, checks that the user has permission to the requested infobase by looking up the username/password in the PostgreSQL database, if yes, redirects the request to the infobase, gets the response, and send it to the user, and logging the transaction. As a cache, Squid may respond to the user from cached data, instead of requesting from the infobase.

4.5 Webalizer HTTP Log Monitor

Webalizer is set up to monitor the log file from Squid, and create reports on usage information of different databases, in different hours and days, and from different regions.

4.6 RRDTool Round-Robin Database

Round-robin databases are built for each user accessing to each infobase, later RRDTool can draw usage graphs for past twenty-four hours, or past six months for a user, a group, or the whole system, accessing an specific infobase, or total usage.

4.7 Accounting Module

This module, written in C++ for efficiency matters, reads the log file from Squid, sums up each user's usage and imports this data in PostgreSQL and RRDTool databases. The Cron daemon is responsible for running the module every five minutes. Million lines of log file can be processed in a few seconds.

4.8 Configuration Generation Module

Last, but not the least, is the configuration generation module, written with PHP, extracts the access permission settings from the PostgreSQL database, and builds access lists and rules for the Squid server. Finally, Squid is requested to reread its configuration file, for the settings to take effect. The Cron daemon is responsible for running this module once a night.

5 Scalability

When planning to redirect the load from many academic environments from one single gateway, the main problem that sooner or later we should solve is scalability. Although the narrow bandwidth of the connections in Iran seems to be the bottle-neck of such a system, the performance of the proxy server may become the problem soon.

Three components can be affected by very high load: Apache, PostgreSQL, and Squid. To overcome the problem, each of them can be served on its own powerful server. PostgreSQL cannot be installed on a distributed environment easily, so using a more powerful machine (with more RAM) is recommended. Apache and Squid can be installed on several machines, and a load balancer would redirect requests to the least busy machine on demand. This schema solves the problem of scalability completely.

Another break-point is that when the access permission tables become very large, the Squid server may become slow, as it should check every single request, with a list of access definitions. The rule of the thumb is that a few thousands of access lists is okay, but more than that can cause a real problem. The problem rises from the fact that Squid's access model is not designed for extensive user-based permission handling. A solution that solves the problem efficiently is to implement an standalone authorization module as a firewall. Requests are redirected to the firewall, which after authorizing the access, transparently redirects the request to the Squid server.

6 Conclusion

We first described the problem of many users wanting to acquire information from some information databases. Then we focused on the traditional ways of doing so, and discussed the most important problems with these methods. After that we presented our design for a central gateway to the infobases, and described how it can reduce the effort and cost of accessing to infobases greatly. In a country-wide scale, this difference in net cost means many more informative documents can be pushed into the academy, that hopefully results in greater research to be done.

We then showed that how the designed system is implemented using cheap hardware and Free Software systems. Because of using many powerful and publicly available components, the source code for the whole project is very small, resulting in very low implementation and maintenance costs. Finally, the scalability of the system is discussed.

Acknowledgement

We wish to thank all the people that helped us in the design and implementation phases, specially Behnam Esfahbod, Mohammad Reza Mahjourian, and Hamid Nazerzadeh.

References

Most references used in this project have been the users and programmers manual of the used components, means Apache, PHP, XPage, PostgreSQL, Squid, Webalizer, and RRDTool manuals that all are available on the Internet, and shipped with common Linux distributions. Some valuable books have been used in the implementation phase:

Tanenbaum, Andrew S., *Computer Networks*, 3rd ed, Prentice Hall, 1996.

Matthew, Neil and Richard Stones, *Beginning Linux Programming*, Wrox Press, 1999.

Matthew, Neil, Richard Stones et al, *Professional Linux Programming*, Wrox Press, 2000.

Mourani, Gerhard, *Securing and Optimizing Linux: RedHat Edition*, OpenDocs Publishing, 2000.