SQUASH: DESIGN AND IMPLEMENTATION OF A LARGE SCALE HTTP GATEWAY AND MASQUERADER

G. A. MONTAZER, Ph.D.
Tarbiat Modarres University
Tehran, I. R. of Iran
Corresponding Author:
montazer@modares.ac.ir

B. ESFAHBOD, M.S.
Computer Engineering
Tarbiat Modarres University
Tehran, I. R. of Iran

H. SAFI ALLAH, M.S.
Computer Engineering
Tarbiat Modarres University
Tehran, I. R. of Iran

Abstract - Information available on the web can be divided in two main categories. The first category is those information distributed among different websites and servers. The second category is the information gathered in central information databases (infobases) which contain a huge amount of information and are available and searchable through the website representing the infobase. Infobase owners sell information to the users on per request basis, or to larger groups, based on the number of needed resources. Authentication is donc in one of two ways, or both of them: username/password and IP-based. The first approach has so many practicial problems so, in this paper a new approach has been proposed based on central system as an access-point and gateway for users to receive and retrieve their information using Squash technology. The architecture and advantages of this novel method have been discussed and the way of implementation has been described.

Keywords - Information, Squid, Squash, Infobase, Authentication.

INTRODUCTION

The growth of the Internet and web technologies in recent years has affected many different aspects of ourlives. 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 [1]. 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, 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 are 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 compared to the providers of the first category. Moreover, the infobases that a small environment may be interested in are usually limited to a few.

In the common experience of using infobases, there are a few infobases, lets say up to a hundred, and so the greater the 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, etc. 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).

Infobase owners sell information to the user on a per request basis, or to larger groups, based on the number of needed resources (e.g. articles). The latter 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 [2].

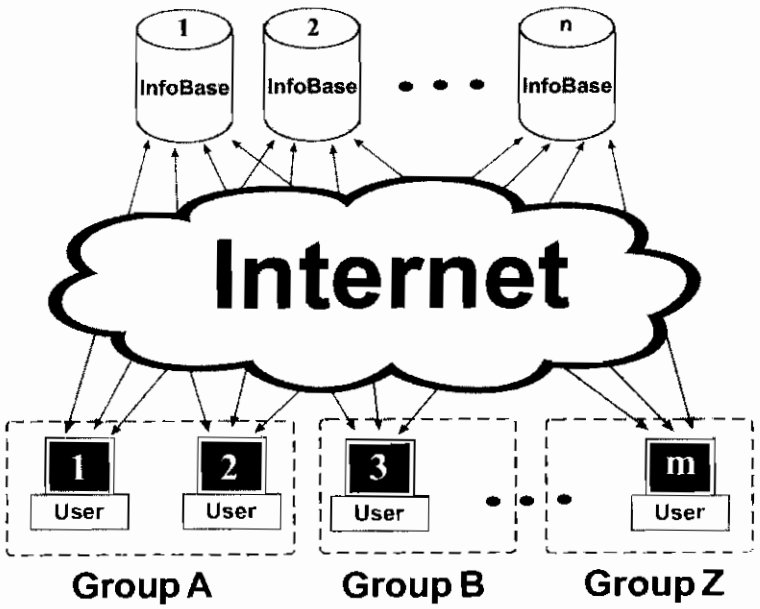

Figure 1: Traditional way of acquiring information from infobases.

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

1. Each user buys his/her own information. In this case, there is no need for authentication, as he/she enters payment information each time the user buys some documents.
2. Each group pays for a large number of documents, and then allows its users to get what they need. In this situation authentication is needed to allow users from the

group, and deny others, from using the 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. In 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 [4]:

- 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 is 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 single users or groups, report their usernames/passwords and IP addresses, and the customer reports them to each infobase, and the infobases put them in effect. This delay can consume some months of a twelve month membership period.

THE NEW APPROACH

To overcome the problems mentioned, we have designed a central system as an access-point and gateway for users of the whole region to access to the infobases. This method has been shown schematically in Figure 2. Some characteristics of the system, not shown in Figure 2 are listed below:

- Squash hides the real user from the infobase. Infobase just sees the Squash system requesting for a document.
- 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 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, the most important of which are:

- 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. In the traditional scenarios, it cannot be, as the infobases are only a few servers in the whole Internet, and caches are filled up with lots of other stuff.
- The region can set up a fast Internet connection to the infobases, and users and groups can benefit from local backbones connecting them to the Squash system, e.g. the fiber-optic backbone of the TCI in Iran that all universities are connected to.

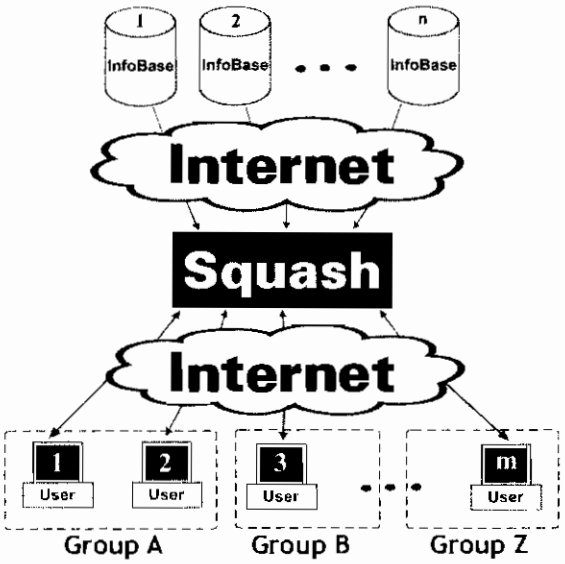

Figure 2: Proposed way of acquiring information from infobases.

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.

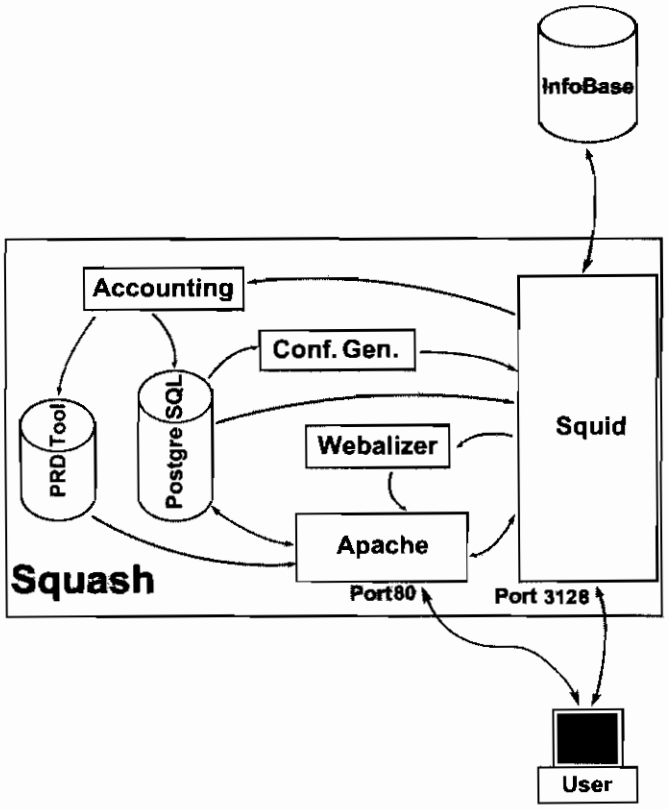

Figure 3: Internal components of the Squash system.

- APACHE HTTP SERVER WITH PHP EXTENSION

PHP is used as the web programming language. The web application is responsible for authenticating users for configuration purposes, and also as a portal to the infobases [3]. 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 its IP address, or otherwise, redirects it to the HTTPS secure pages to login.

- XPAGE DATABASE ORIENTED WEB APPLICATION DESIGN SYSTEM

The XPage system 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.

- POSTGRESQL RELATIONAL DATABASE SERVER

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.

- SQUID HTTP PROXY SERVER

Squid is the engine of the Squash system, and that is the reason why it is named so. Here Squid is set up as HTTP proxy, cache, authenticator and anonymizer. Users just set her their 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 sends 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.

- 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.

- RRDTOOL ROUND-ROBIN DATABASE GRAPHER

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

- 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 the PostgreSQL and RRDTool databases. The Cron daemon is responsible for running the module every five minutes. Millions of lines of log file can be processed in a few seconds.

- CONFIGURATION GENERATION MODULE

The last part of the new system, 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.

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 it is recommended to use a more powerful (dual processor) machine 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 may be 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 OK, but more than that can cause a real problem. The problem rises from the fact that Squid access model is not designed for extensive user handling and user-based permission handling. A solution that solves the problem efficiently is to implement an standalone authorization module as a firewall, that loads the whole permissions database in memory, for a million users and hundreds of infobases, it can be done in at most 100MB of memory. Requests are redirected to the firewall, which after authorizing the access, transparently redirects the request to the Squid server. This method is not implemented yet.

CONCLUSIONS

We first described the problem of many users wanting to acquire information from some information databases, called infobases. Then, we focused on the traditional ways of doing so, and discussed the most important problems with these methods. After that, we presented a new approach for a central gateway to the infobases, and described how it can reduce the effort and cost of accessing to infobases greatly, by reducing the cost of a single document, and the cost of access through the Internet. In a country-wide scale, this difference in net cost means that 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 could be implemented using cheap hardware and open source 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.

Scalability is the first question about this system, which we discussed the ways it could be solved effectively. Other ways to overcome this problem are still under research.

REFERENCES

[1] Markus, K., "Uses and Users of Information", available online at <http://www.rit. Com/internet>, 2002.

[2] Neil, M. and Stones, R., "Beginning Linux Programming", Wrox Press, 1999.

[3] Neil, M., Stones, R., et al., "Professional Linux Programming", Wrox Press, 2000.

[4] Mourani, G., *Securing and Optimizing Linux,* RedHat Edition, OpenDocs Publishing, 2000.

[5] Tanenbaum, A. S., *Computer Networks,* 3rd ed.; Prentice Hall, 1996.