Prometeo internals

Simone Tellini and Renzo Davoli

Department of Computer Science - University of Bologna,
Mura Anteo Zamboni, 7, I40127 Bologna, Italy
{*davoli, tellini*}*@cs.unibo.it*

Abstract

This paper introduces Prometeo[1], a multi-function, modular and extensible proxy server created as part of one the author's thesis work. We will discuss the needs that this project was meant to address: mainly the lack of an application with the aforesaid features, combined with native IPv6 support and ease of administration. Prometeo also provides a C++ framework which simplifies the development of networking applications. The design of Prometeo's will be described, starting with an overview of its components and modules and commenting on the most significant parts of the implementation. Then we will focus on the main issues considered during the development of the project, comparing the adopted solutions with those of other state-of-the-art packages like Squid [1]. Finally we will discuss new ways of improving Prometeo's performances and scalability.[2]

1 Introduction

Proxies are important components of large, heterogeneous networks: they're often found on the frontier of private LAN's or corporate networks to allow their users to access Internet resources in a controlled manner - for instance, forcing them to obey to corporate policy. Caching proxies also help to optimize the available resources, reducing the traffic generated by the users. Another class of proxies enables interoperability between applications, translating on the fly from one protocol to another (for example, from NNTP to POP3). Proxies can also be used for special purposes, for instance to allow visually challenged people to browse the web [2] or to improve the management of networked games [3].

2 Motivations

There are plenty of proxies for almost every service, thus one may wonder where the need of another product comes from. Problem is, the vast majority of the existing packages were aimed at solving a specific goal such as providing a certain service (e.g. http) or adding a special or missing feature to an existing client-server setup (e.g. stripping banners from web pages before feeding them to the browser, or adding TLS/SSL encryption). This implies that a system administrator who need to setup different proxy services is bound to install several packages, each one of them with its own management rules and its idiosyncrasies.

[1]Available under the GPL license on sourceforge. See http://prometeo-proxy.sourceforge.net/

[2]This work has been partially supported by the FIRB project named WebMinds of the Italian Ministry for Education, University and Research and by the 6NET E.U. project.

Feature		Prometeo	Squid	WWWOFFLE	Apache	Ziproxy	Tinyproxy	SuSE Proxy-Suite	Frox	DeleGate	Stunnel	TLSWRAP	WinGate
Open Source/Free Software		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	
Modular Design		Y			Y								Y
Easy to extend		Y											n/a
Easy to use		Y				Y	Y		Y			Y	Y
IPv6 support		Y		Y		Y							?
Transparent proxy support		Y	Y				Y	Y	Y				Y
Support for multiple protocols		Y								Y	Y		Y
Remote Administration		Y		Y						Y			Y
DNS cache		Y	Y							Y			Y
FTP proxy	RFC-959 compliance	Y				n/a	n/a	Y	Y	Y	n/a	Y	Y
	SSL/TLS wrapper	Y				n/a	n/a			Y	n/a	Y	
HTTP proxy	HTTP/1.1 support	Y	Y	Y	Y			n/a	n/a	Y	n/a	n/a	Y
	Cache	Y	Y	Y	Y			n/a	n/a	Y	n/a	n/a	Y
	Gzip/deflate compression	Y		Y		Y		n/a	n/a	Y	n/a	n/a	
	Connection cache towards origin servers	Y	Y					n/a	n/a		n/a	n/a	
	Filters	Y	Y	Y				n/a	n/a		n/a	n/a	Y
	Host remapping	Y			Y			n/a	n/a	Y	n/a	n/a	
POP3 proxy	POP3 service	Y								Y			Y
	SpamAssassin support	Y											
SSL Tunneling		Y								Y	Y		Y
TCP Tunneling		Y								Y			Y

Table 1: Prometeo features compared to those of other existing applications

Prometeo main idea was to provide the administrator with an equivalent of the inetd daemon for proxies: a single application able to serve different kind of services through the use of plug-ins. This solutions gives several benefits:

- it simplifies the administration and maintenance of the proxies: services can be started, stopped, (un)installed or updated independently using the same tool;
- resource optimizations: many common functions are included in the framework which is shared by every module implementing a single service;
- Prometeo's framework lets the developer to focus on the logic of the service she wants to implement, making it quite easy to add new services to the package.

Moreover, we wanted an application which supported IPv6 networks natively and that could help to interconnect IPv4 and IPv6 networks allowing IPv4-only clients to access IPv6 servers or vice-versa.

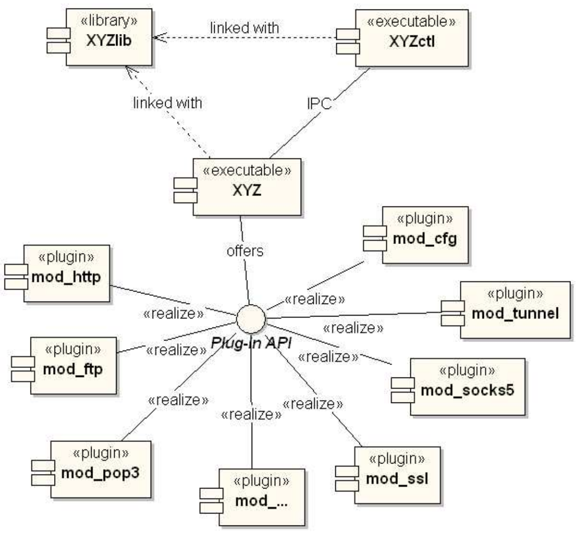

Figure 1: Prometeo components

Table 1 shows a comparison of Prometeo's main features against those of other available applications: as you can see, we tried to gather the most important features of several packages into a single, integrated package.

Two are the products which mostly share Prometeo's philosophy and goals:

- WinGate, a commercial proxy suite for Microsoft Windows platform developed by Qbik and marketed by Deerfield.com. It's main drawback is that it's a closed-source application and that seemed an unacceptable constraint to us.
- DeleGate, an open-source project started in 1984 by Yutaka Sato and grown thanks to the contribution of many programmers around the world. It too provides a lot of different proxy services in a single product, although it has a monolithic design rather than a modular one. Moreover, it's doesn't support IPv6 at all.

Among the others, we can't help mentioning Squid [1], perhaps the most well- known and widely used Web caching proxy for POSIX systems. Squid has become an industry standard used by many corporations or Internet Service Providers thanks to its robustness and scalability.

3 Design overview

Prometeo is composed by several components, as shown in figure 1.

The core of the system is formed by the Prometeo main executable, which contains the framework shared by the modules: other than the normal housekeeping functions, it offers a centralised access to the configuration, logging, access control, storage access and so on.

The core uses a plugin architecture to offer its services: every plugin implements a proxy for a different protocol or implements additional features - mod_cfg, for instance, offers a web-based control panel, providing a comfortable way of configuring and managing the whole application, other plugins included.

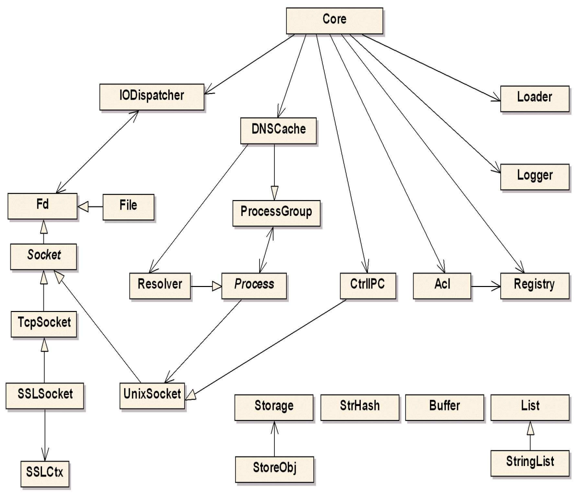

Figure 2: an overview of the main classes composing Prometeo's framework

Promlib provides the asynchronous I/O file and network functions. It adds an abstraction layer between the application logic and the standard BSD-socket interface, encapsulating all the required functions in a set of C++ classes. Prometeolib supports IPv6 in a fairly transparent way: the programmer won't have to worry about the differences between IPv4 and IPv6 unless he requires so. This library can also be used in other projects without many changes, making life easier expecially whenever asynchronous operations are required.

Prometeoctl is an utility program which allows to administer the system from command-line or from scripts.

4 Implementation

Figure 2 shows an overview of the most significant classes included in Prometeo's framework. In the following paragraphs we'll give a brief description of them and focus on the most interesting aspects of the implementation.

4.1 Core

Core is the main object of Prometeo's executable: it holds pointers to the instances of the shared objects which are available to every module.

It also provides a fork() wrapper which should used whenever a module needs to spawn a child process. The biggest problem with using the system call directly is that the children inherit the file descriptors of the parent. Suppose that one of them is a listening socket: the parent wouldn't be able to close it and bind another socket to the same port, because it would be kept in use by the child. The fork() wrapper remedies to the problem by asking every module to free any resource which shouldn't be passed on to children.

4.2 Main utility classes

Loader deals with all the functions connected to the management of plugins.

Logger at the moment is merely a syslog-wrapper. It's only goal is to enable a quick transition to other logging systems, should the need arise.

Registry is the configuration access interface used everywhere across the package. Currently, it stores the settings in a tree structure stored on disk as an XML file.

CtrlIPC handles the communication with Prometeoctl and implements all the commands made available by the latter.

IODispatcher manages the asynchronous I/O operations. As the name implies, its main job is to dispatch the incoming events to the right handler.

4.3 DNSCache

DNSCache handles the resolution of hostnames in an asynchronous way. The current implementation uses a configurable number of helper processes to perform the actual lookup.

The cache can be shared by every module, giving Prometeo an advantage over a solution composed by different applications.

4.4 ProcessGroup

Along with the Process abstract class, ProcessGroup is used to quickly implement helper processes and processes caches.

These two classes offer methods to communicate between parent and child process without having to worry about the IPC details. The implemented message passing protocol is very simple but adequate for most uses; if required, it can be easily extended overriding a couple of methods. In the current implementation it's based Unix sockets.

Both classes are meant to be sub-classed: the ProcessGroup descendant will populate itself with specialised Process descendants when instantiated (as DNSCache) or on-demand (as mod_ftp, for one).

These classes are ideal to implement a process cache: when they need to satisfy a client request, they look for an idle process in the group. If available, it will be used to process the request. Otherwise, if the maximum number of children has not been reached yet, a new process will be spawned and added to the group.

ProcessGroup is able to dispose periodically those processes which have been idle longer than a certain time, in order to free resources when they're not needed.

4.5 ACL

This class manages user authentication and access control lists.

User authentication is implemented using BSD-Auth on BSD systems and PAM [4] elsewhere. Both of them let the system administrator to choose the authentication method that best suits his environment.

4.6 Storage

The Storage and StoreObj classes provide an abstraction layer to the storage medium. For simplicity, the current implementation uses the OS file-system, although this isn't an optimal solution for high-load proxies, since the I/O overhead can be quite elevated [5].

5 Modularity

All the services offered by Prometeo are implemented as modules. As already mentioned, modules can be independently configured, loaded, unloaded or upgraded. It's also possible to provide specialised services using the same plugin with different configurations on different ports.

In the following sections we'll give a brief description of the most interesting aspects of the currently available modules.

5.1 mod_http

Most of the time spent developing Prometeo has been dedicated to this module, since HTTP is indeed the most used protocol [8]. The module implements an HTTP 1.1 [11] caching proxy.

It has been designed keeping the following points in mind:

- asynchronous operations: everything is performed in the same process, in an effort to minimize latency and to simplify the access to common resources, such as the cache, since no locking mechanism is required;
- low latency: to guarantee a prompt reply to client requests, the full index of the cache is kept in memory, while the actual cache objects are loaded only when needed;
- bandwidth optimization: if the server or the client supports either gzip or deflate content encoding data is transferred in a compressed format; moreover, concurrent requests for the same resource are satisfied using a single server connection, thus avoiding unnecessary transfers;
- standard compliance: mod_http tries to comply with all the recommendations found in [11] regarding proxy servers: for instance it correctly handles persistent connections separately with its clients and the origin servers, it understand the cache control headers and so on.

Data compression is especially useful when clients are connected to the proxy using a slow link, such as a dialup connection or GPRS [9].

5.2 mod_ftp

This module provides an FTP [12] proxy whose strength points are:

- IPv6 support: not only it understands the IPv6 protocol extensions [13], it allows IPv4 clients to access IPv6 servers as well.
- SSL/TLS support [14]: mod_ftp is optionally able to secure communications with origin servers, even if the client doesn't support SSL/TLS. It's also possible to disallow access to unsecured origin servers.

5.3 mod_cfg

Unlike the other modules, mod_cfg doesn't implement a proxy. Instead, it offers a web-based configuration interface which can be used to configure the whole Prometeo application, including the other modules.

In fact, every module provides an XML fragment describing its configuration options. mod_cfg exploits these information to build an user-friendly interface on-the-fly.

6 Extensibility

A key aspect of Prometeo's design is extensibility: adding a new service is a very simple job. The framework already deals with most of the low-level or tedious parts which are needed by any proxy, such as logging, authentication or caring about IPv4/IPv6 differences.

The programmer should only need to focus on the logic of the proxy she needs. Looking at the source code will show how the code of the modules is simple. For instance, mod_ftp is just about 1500 lines of code (comments included), against the 4600 lines of SuSE Proxy Suite.

mod_http is just 4500 lines of code, which is not bad considering that WWWOffle [16] is made up of 29000 lines of code offering more or less a comparable range of features of Prometeo (although Prometeo is more scalable). As a side note, Squid is about 56000 lines of code, but it offers a number of features (SNMP, ICP...) still not comparable to mod_http.

7 Resource optimization

7.1 Caching the connections

mod_http is able to cache connections: HTTP/1.1 persistent connections are always requested when dealing with an origin server. After receiving the requested resource, the idle connection is kept in a connection cache for a limited amount of time (15 seconds in the current implementation, as the default timeout for idle connections in the Apache web server). During this time span, when a client requests another resource from the same server, it will be served using an already established connection. As noted in [7], connection caches can be very useful to reduce latency.

7.2 Requests aggregation

When mod_http receives a new request for a resource that is already been transfered from an origin server to a client, it will serve it without establishing a second connection to the server: it will immediately send the data available to the client and as soon as new data arrivers, it will be forwarded to all the registered clients. The implementation in Prometeo is similar to that of Squid.

Adding requests aggregation to an HTTP proxy proved to be a simple task, yet very useful to save on connections especially when there's a peak of requests for a popular resource.

7.3 Processes cache

Prometeo uses for many of its modules (such as mod_ftp, mod_ssl, mod_pop3...) a cache of child processes which can be used to process an incoming request. The idea has been inspired by the behaviour of the Apache server [15], although Prometeo's framework offers a generalized implementation which helps to create custom process caches very easily.

A processes cache helps to reduce the overhead associated with the creation of a new child and the setup of an IPC channel with the parent and it's particularly effective when the proxy is been stressed.

Another interesting aspect of using a process cache is that if a child process crashes, it's damages to the rest of the system will be limited, in the general case. A crashed process will be replaced with a newly spawned one as soon as it's needed.

8 About scalability

Initial tests have shown that mod_http performs as well as Squid in terms of latency and throughput (requests per second). More accurate benchmarks are required to give a final judgement. Still, we've noted some interesting points:

- logging requests to file inflicts a considerable penalty both in terms of latency and throughput. Disabling the log, Prometeo has been able to serve more than twice the number of requests served when logging is active with less than half of the latency. Squid doesn't have a way of disabling the log, other than redirecting it to /dev/null, which doesn't seem to help much. It's also worth noting that Squid writes log files directly, while Prometeo passes through syslog: further tests should investigate whether using a secondary machine to receive the log information via syslog's network support could help the proxy performances. Unfortunately, disabling the log isn't always feasible, as it might be required for security reasons.
- the use of persistent connections and the connection cache seems to have a great impact on throughput, while it is negligible on latency. The tests seem to be in line with [7] findings, showing a 4x increase in the number of requests satisfied per second.

8.1 Exploiting multiple CPU's

Currently, both Prometeo's mod_http and Squid operate using a single process. The main advantage of this solution is that it doesn't require locking mechanisms to access the cache, thus simplifying the code.

On the other hand, a single process can't receive any benefit being run on a SMP machine.

Apart from mod_http, the other modules of Prometeo are implemented using a process per client. This way, each concurrent process can be scheduled on a different processor.

8.2 Hierarchical caching

One of the biggest scalability advantages of Squid over Prometeo is the support for hierarchical caches: if your proxy does not have an object on disk, its default action is to connect to the origin web server and retrieve the page. In a hierarchy, your proxy can communicate with other proxies (in the hope that one of these servers will have the relevant page).

In large networks, a single proxy server may not be enough to satisfy all the requests. Adding more servers helps to keep the individual load under control, while increasing the overall number of clients that can be served at once. Squid efficiently implements different inter-cache communication protocols, thus being able to maintain latency at a low level even with big cache hierarchies.

Recently new approaches to the scalability problem of web proxies have been proposed [10]. It would be worth to test how well these proposals behave in a real environment.

9 Use cases

In this section we'll see a couple of example showing how Prometeo is currently being used reporting the experiences of its users.

9.1 Interconnecting IPv4 and IPv6 networks

At the CS Department of University of Bologna, Prometeo has been used for more than 5 months as default proxy for the IPv6 research project, proving to be robust and reliable.

The proxy is connected to the Internet and to the 6bone network. Its main goal is to make accessible any server, be it on the IPv4 or IPv6 network, to any client. Also, if there were the need of setting up some web servers on native IPv6 hosts, Prometeo could be used to make them accessible from the Internet acting as a front-end server and mapping the requests to the correct machine.

9.2 Making use of special features

A feature that has encountered a good success among the users is the support for Spam Assassin which has been integrated into mod_pop3.

The module implements a simple POP3 proxy which optionally can filter emails through Spam Assassin's spamd daemon: this feature is mostly useful to those users which cannot change the mail server setup because it's not under their control and thus cannot filter spam as it arrives. Also, the support for transparent proxy[3] makes mod_pop3 use very comfortable, since it doesn't require changes in the clients' configuration.

An ISP is currently evaluating the use of this module to offer a spam filtering service to some of its customers without having to modify the rest of its mail servers setup.

[3]At the moment of writing, transparent proxy and IPv6 support are mutually exclusive on GNU/Linux.

10 Improvement ideas

We reckon that some aspects of Prometeo are not optimal, mostly due to the strict time constraint which have condition the development process.

We propose some ideas which would improve Prometeo's performances on large networks or add interesting features.

10.1 Replacement policies

As described in [6], LRU is not the optimal replacement policy for a caching web proxy. It should be fairly easy to implement the proposed LRV policy in mod_http.

10.2 RAM-based cache

Nowadays servers can be fitted with huge amounts of memory for a relatively small price. For instance, a dedicated proxy machine with 4 GB of memory would gain a lot in terms of performances if it could avoid using the disks. At the same time, 4 GB should be more than enough to host a web caching proxy for a large network or several kind of proxy for a smaller one.

Prometeo can be easily modified to use memory for all its storage needs: it's only necessary to rewrite the Storage and StoreObj classes.

The only drawback with placing the cache in memory is that a machine reboot would reset the cache. We believe that the benefits are worth the risk though, also considering that reboots should be rare and mostly due to hardware problems, given that Prometeo has reached a satisfying level of stability.

10.3 Differentiated/dynamic filters

mod_http could enable different filters according to the type of the client which submits the requests.

For instance, if the client is using a narrow link, as in the GPRS case, it could scale down or resample images to save bandwidth.

This could be implemented using proxy-authentication and a database of users' preferences.

10.4 Load balancing module

An interesting module that could be added to Prometeo is a software load balancer. It could be useful to use Prometeo as a front-end to a group of servers which need to sustain very high loads.

References

[1] Squid Internet Object Cache, [online] http://www.squid-cache.org/

[2] Hironobu Takagi, Chieko Asakawa, “Transcoding proxy for non visual web access”, *Proceedings of the fourth international ACM conference on Assistive Technologies*, 2000, pages 164-171

[3] Martin Mauve, Stefan Fischer, Jörg Widmer, “A generic proxy system for networked computer games”, *Proceedings of the first workshop on Network and system support for games*, 2002, pages 25-28

[4] V. Samar, R. Schemer, “Unified login with Pluggable Authentication Modules (PAM)”, *Open Software Foundation RFC 86.0*, October 1995

[5] J. C. Mogul, “Speedier Squid: A case study of an Internet server performance problem”, *Login: The USENIX Association Magazine*, 1999, vol. 24, no. 1, pages 50-58

[6] Luigi Rizzo, Lorenzo Vicisiano, “Replacement Policies for a Proxy Cache”, *IEEE Transactions on networking*, April 2000, vol. 8, no. 2, pages 158-170

[7] Ramón Cáceres, Fred Douglis, Anja Feldmann, Gideon Glass, Michael Rabinovich, “Web proxy caching: the devil is in the details”, *ACM SIGMETRICS Performance Evaluation Review*, December 1998, vol. 26, issue 3

[8] Martin Arlitt, Rich Friedrich, Tai Jin, “Workload Characterization of a Web Proxy in a Cable Modem Environment”, *Proceedings of the eleventh international conference on World Wide Web*, May 2002, pages 25-36

[9] Bruce Zenel, Dan Duchamp, “A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment”, *Proceedings of the third annual ACM/IEEE international conference on Mobile computing and networking*, 1997, pages 248-259

[10] Markus J. Kaiser, Kwok Ching Tsui, Jiming Liu, “Self-organized Autonomous Web Proxies”, AAMAS’02, 2002, pages 1397-1404

[11] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, “Hypertext Transfer Protocol - HTTP/1.1”, *RFC 2616*, June 1999

[12] J. Postel, J. Reynolds, “File Transfer Protocol (FTP)”, *RFC 959*, October 1985

[13] M. Allman, S. Ostermann, C. Metz, “FTP Extensions for IPv6 and NATs”, *RFC 2428*, September 1998

[14] Ford-Hutchinson, Carpenter, Hudson, Murray & Wiegand, “Securing FTP with TLS”, Draft 09, April 2002, [online] http://www.ford-hutchinson.com/~fh-1-pfh/ftps-ext.html

[15] Apache Software Foundation, [online] http://httpd.apache.org

[16] Andrew M. Bishop, [online] http://www.gedanken.demon.co.uk/wwwoffle/index.html