Design and implementation of a Layer-7 MPLS-based Web Switching Architecture

A. Mancuso[1]*, E.S.G. Carotti*[1]*, J.C. De Martin*[2]*, A.R. Meo*[1]
[1]*DAUIN/*[2]*IEIIT-CNR – Politecnico di Torino*
c.so Duca degli Abruzzi, 24
10129 - Torino – Italy
{*carotti*|*demartin*|*meo*}*@polito.it*

Abstract

During recent years web servers evolved from providing simple, static content to offering different services and a variety of dynamically generated pages and objects. Consequently, scalability and load-balancing have emerged as main requirements for modern web farms. A common solution is based on placing a Web Switch in front of the web servers; the switch acts as a dispatcher that redirects user requests according to desired criteria. In this paper we describe a novel web switching architecture, based on the MPLS technology and on Open-Source software. The switching decision is primarily made considering Layer-7 information, thus achieving flexible content-based routing to the most appropriate server. State information from the web servers - such as, for example, load average and resource availability - is considered as well. The architecture here proposed has been implemented as a free software project using MPLS-enabled Linux workstations.

Keywords

Web switching, MPLS, Web Farm, Layer-7 switching, content-based routing

1. Introduction

As the use of residential broadband connection increases, more users connect to the Internet to access services like, for example, home-banking or e-commerce. As a consequence, the need for larger bandwidth and more computing power is increasing to handle user requests quickly and efficiently. To meet these growing requirements many solutions have been proposed and adopted, such as for example: various form of caching, mirroring and clustering of servers.

A common solution is to build a web farm as a cluster with a front-end which chooses the proper web server for each request according to a predefined policy.

Dragos *et al.* [4] implemented a load-balancing architecture based on Multi-Protocol Label Switching (MPLS, RFC 3031 [6]), which is a fast-growing technology [1] allowing for faster switching performance with respect to IP, since only a fixed-length label switch has to be performed on each packet. Dragos's technique, however, does not choose the web server according to the specific resource requested since it uses only Layer-4 information. Yang and Luo showed in [3] that it is convenient to partition the web farm according to the kind of resources offered to optimize the performance of a distributed

web server system. For this purpose a Layer-7 switch needs to be employed, i.e. a switch that routes requests based on the content requested.

In this paper, we discuss the design and the implementation of a novel MPLS-based Web Switch using Layer-7 information, i.e., the switching decision is taken with respect to the availability of specific resources on a certain web server. We took our cue from an article discussing Web Switching over MPLS [2] where the high-level architecture of an MPLS-based Web Switch was sketched. We detailed the architecture by designing and developing all the different modules needed to implement a Layer-7 Web Switch using only Open-Source software such as MPLS-enabled [5] Linux.

2. Open Web Switch Architecture

The main goal of this work is to design an architecture that could be used as a model to implement an MPLS-based Layer-7 Web Switch. Thus, the proposed architecture defines the subsystems, their functionalities and their mutual interaction, leaving to the implementor the choice of the specific MPLS switch, which could be a standard commercial MPLS-switch or even a low-cost MPLS Linux-based system; different switching policies can be easily implemented.

The proposed architecture is: **Open**, since it can be implemented using only Open-Source software and the design itself is open, meaning that anybody can adapt it to particular needs; **Distributed**, because while other load-balancers or content-routers centralize all the functionalities on a single device, OWS makes a clear distinction between the forwarding logic and the physical dispatching of the traffic to the web farm, distributing them on different devices at different network locations; **Modular** as each subsystem in the architecture is composed of several interacting software modules.

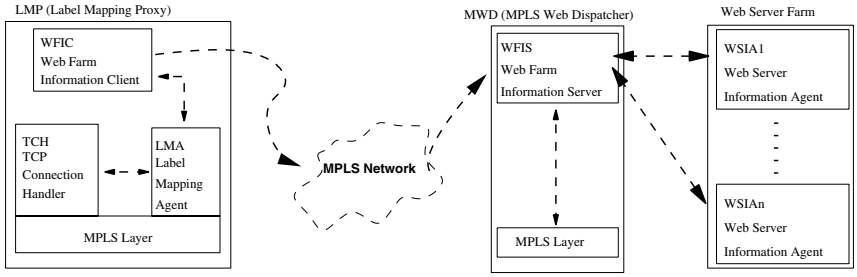

Figure 1: The main modules of the proposed architecture.

The Open Web Switching (OWS) architecture, depicted on Figure 1 is composed of a cluster of web servers and two other subsystems, which are the **Label Mapping Proxy** (LMP) which binds the appropriate MPLS label to the HTTP requests according to Layer-7 headers, and the **MPLS Web Dispatcher** (MWD) which dispatches incoming flows to the more suitable web server in the farm according to the label previously attached. The dispatching logic is thus distributed among the LMP which performs "logical dispatching" and the MWD which performs "physical dispatching".

Each client is assumed to direct all its requests to the logically closest LMP system which assigns the proper MPLS label. Since MPLS labels can be stacked, the label assigned by the LMP will not be removed until the last node (the MWD) is reached and thus, it can be used to identify the web server or the resource. This way the Layer-7 headers are analyzed and the switching decision is taken only at the front-end proxies thus eliminating the bottleneck of a centralized switch.

The MPLS cloud between the proxies and the dispatcher is completely unaware of the meaning of the MPLS label assigned to each flow by the LMP, and simply routes the requests to the MWD system. The web server in the back-end does not have access to the original HTTP header and identifies the requested resource using the original MPLS label, assigned by the front-end. As a consequence the system is transparently and implicitly attributing a semantic meaning to MPLS labels while it is using them to perform optimal web traffic dispatching.

2.1 Web Farm

Each web server in the farm needs to send its state information to the MWD. This is the task of the Web Server Information Agent (WSIA), which interacts with the MWD, sending for example, server load or resource availability, according to the desired policy (load-balancing, partitioning of the web farm and resource allocation). At least the following information has to be sent: the **IP Address** of the web server: each web server has an associated IP address, which is needed to serve client's requests directly back without going through the Web Switch; the **List of Resources** which is needed for the switching decision.

The OWS architecture does not put any constraint on the dispatching policy so, for example, current load average could be sent and taken into account if load balancing functionality is desired.

2.2 MPLS Web Dispatcher

The MPLS Web Dispatcher (MWD) is the subsystem which dispatches user requests to the web servers according to the associated label and which also receives the Web Farm information to be propagated to the front-end proxies from the WSIA.

The MWD keeps a table containing the mapping between user requests (HTTP URNs) and web servers. This table is used in the switching decision so that each incoming flow is forwarded to the proper web server.

The Web Farm Information Server (WFIS) is the only software module (apart from the MPLS forwarding logic) which runs on the MWD subsystem. This module performs two tasks: first, it gathers and distributes all the information needed to make the dispatching decision and to bind MPLS labels to user requests; second, it creates and keeps up-to-date the MPLS switching table.

The MWD can be implemented on any platform with an MPLS forwarding layer and a control processor to establish the communication with the WSIA and the proxies. For this purpose a commercial MPLS switch could be used; we chose to use a Linux workstation with a kernel patched to support MPLS [5].

2.3 Label Mapping Proxy

The main task of the Label Mapping Proxy (LMP) is to receive user requests, analyze the HTTP headers and bind MPLS labels so that each request is dispatched to the proper Web Server. LMP works mostly like an HTTP proxy and it is also responsible of handling low-level TCP details. As such, it can be chosen by the user or transparently deployed by the internet provider.

It is composed of three modules: the TCP Connection Handler (TCH), the Label Mapping Agent (LMA) and the Web Farm Information Client (WFIC).

The task of the TCH module is to handle low-level TCP and HTTP details, such as terminating TCP connections and handling the HTTP protocol. This two steps are closely related because to actually perform Layer-7 switching it is necessary to get the HTTP header which is not sent before a regular TCP connection has been established, i.e., the three-way handshake is performed.

The Label Mapping Agent (LMA) binds user requests to the corresponding MPLS labels. This module decides which is the proper back-end server for every incoming request, attaching the corresponding MPLS label to all the packets belonging to the request.

Last but not least, the WFIC gathers all the information coming from the WSIAs on the dispatcher and redistributes them to the Label Mapping Agent module.

3. Conclusions and future work

In this paper we discussed the design and implementation of a Layer-7 MPLS-based Web Switching Architecture. The main technical goals of the proposed design are: making an efficient Layer-7 Switch, removal of the single point of TCP connection termination, and exploitation of some MPLS features.

Future work includes the definition of a High Availability schema for the MWD subsystem, which would increase the global availability and reliability of the system as an experimental campaign to measure the performance.

References

[1] Arup Acharya. Multi-protocol label switching (MPLS). *http://www.research.ibm.com/mplsws/publications/mpls1.pdf*, August 2000.

[2] Arup Acharya, Anees Shaikh, Renu Tewari, and Dinesh C. Verma. Web Switching using MPLS. *MPLS World News*, January 1997.

[3] Chu-Sing Yang and Mon-Yen Luo. A content placement and management system for distributed Web-server systems. In *Proceedings of the 20th International Conference on Distributed Computing Systems*, pages 691–698, 2000.

[4] Radu Dragos, Sanda Dragos, and Martin Collier. Design and implementation of an MPLS based load balancing architecture for Web switching. In *Proceedings of 15th ITC Specialist Seminar*, Wurzburg, Germany, July 2002.

[5] J. Leu. MPLS for Linux. *URL: http://mpls-linux.sourceforge.net*.

[6] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031: Multiprotocol Label Switching Architecture, January 2001.