An MPLS based architecture for differentiated Web service

Radu Dragos & Martin Collier
Dublin City University, School of Electronic Engineering
Dublin , Ireland
e_mail: {dragosr, collierm}@eeng.dcu.ie

Abstract: **The WWW has been the preferred technology used to provide information and e-services over the Internet. However, one major limitation of Web servers is that they treat all requests equivalently, without using priority schemes or different levels of service. Various methods to introduce such differentiated service are possible.**

MultiProtocol Label Switching (MPLS) can be an useful tool to extend the quality-of-service (QoS) capabilities and Traffic Engineering from the core of the network to the end-systems.

In the context of a QoS enabled network with MPLS capabilities, we propose an MPLS based solution to provide different levels of Web service. We have designed and simulated a Web switching architecture for next-generation QoS enabled IP networks, based on a Linux implementation of MPLS.

1. INTRODUCTION AND MOTIVATION

The number of services offered over the Internet has dramatically increased in the last few years. The WWW service no longer consists in merely displaying the static content from a web server to a web browser. New and more traffic- and CPU-intensive services are widely used nowadays. Services like search engines, file servers, multimedia streaming, application servers, database servers and e-commerce servers, not only require higher bandwidth, but also stretch to the limit the computational performance of servers. Addressing this problem requires new architectures such as Web clusters, and new technologies such as Web switching. The current available solutions do not offer end-to-end quality-of-service (QoS) or are cost prohibitive [1].

This section describes the motivation for providing different levels of Web services. The paper continues in section 2 with a short presentation of an MPLS based load balancing architecture for Web switching. In section 3 we propose a framework for guaranteed Web services and we present experimental results in section 4. The paper is concluded in section 5.

*1.1 **Web server availability***

Popular Web sites receive ten times more requests than three years ago, from 115 million page views per day in June 1998 to 1.5 billion page views per day in September 2001 (e.g., [2]). The value is around 17361 hits per second and may increase over the next few years. The rate of incoming requests is higher than what a single server could handle in a timely manner. Consequently, Web service providers face technical challenges in meeting this burgeoning demand and must deploy new techniques in order to satisfy the continuously increasing number of requests.

The typical solution to the problem of over-congested Web servers is **clustering of Web servers (server farms)**. This technology has to transparently divert the client's request to the optimal server. The technique consists in grouping the servers in so called server clusters and adding a new and transparent service to the network, which is responsible for distributing the requests uniformly among the servers [3].

Successful administration of server clusters requires the use of specialized mathematical and engineering theories like queuing theory and load balancing techniques. The most important goal from the Web service provider's point of view is to uniformly balance the workload among the servers within the cluster.

*1.2 **Differentiated Web services***

The Internet is undergoing a transformation from a best-effort network to an end-to-end QoS architecture. The research effort is concentrated toward the core of the network. The Internet Engineering Task Force(IETF) projects such as Integrated Services [4] , Diffserv [5], Resource Reservation Protocol [6] or MPLS [7] are concerned more about improving the performance of the network capabilities rather than the capabilities of end-systems. Consequently, QoS aware servers can dramatically improve the overall performance of the network.

With Diffserv or MPLS, packets are classified and marked to receive a particular per-hop forwarding behavior on nodes along their path. Traffic can be aggregated into traffic trunks. Various policies can be applied in a different manner for disjunct classes of traffic. Classes of users or applications can be maintained and their traffic can be shaped in a different manner.

Our proposal is to extend the QoS capabilities of the core network to the network boundaries. The quality of Web service (QoWS) can be improved if Web servers are QoS aware. Moreover, different levels of Web service can be delivered from HTTP servers.

The following sections present an architecture for providing differentiated Web services in an MPLS aware network.

1.3 *Guaranteed Web services*

We consider the problem of providing two classes of differentiated Web service. The approach can be used when a Web service provider (WSP) conceives a plan for serving more than one class of users. The two classes comprise privileged users and best-effort users. Here are few possible scenarios for distributing users among the classes:

- Paying customers versus non-paying customers;
- Intranet users versus Internet users;
- Professors versus students.

Various terms can be used to specify the two classes:

- Privileged and best-effort;
- High-priority and low-priority;
- Foreground and background;
- Premium and basic.

The rest of the paper will refer to the two classes as premium services and basic services.

Our research takes into consideration services that are more CPU-intensive than static Web content delivery is.(e.g. CGI scripts or database queries). Those types of requests are characterized by a smaller arrival rate but require a longer CPU processing time at the server side. Hence, there is a higher probability of multiple simultaneous requests contending for CPU time slots. Experimental results for a Web server presented in section 4 show that the execution time increases linearly with the number of tasks being processed in parallel and may overload the server.

The result is an execution time greater than what an user is willing to wait for. Moreover, the client (human user or software program), reaching a time-out threshold may retry the request and by that the server will receive even more requests. A continuously increasing number of incoming requests will cause undesired congestion. Hence, the number of active connections per server must be restricted below a predetermined threshold.

The first two approaches to keep the server within it's running parameters are:

- Dropping all the incoming requests that exceed a maximum limit.
- Queueing the requests and waiting for free server slots.

Neither one of the above solutions are acceptable for a client expecting guaranteed service delivery. Consequently, when a server is not capable of serving the clients in a timely manner, more than one server is required.

Clustering Web servers is a common approach for improving the performance of Web content hosting [8], [9]. Using a cluster of servers introduces a new problem: load balancing across the servers.

Another issue in providing guaranteed Web services using multiple servers are Web traffic bursts. Since Web requests do not arrive with a constant arrival rate, the Web service provider has to be prepared for the worst situation and that is the highest possible arrival rate. This may occur when all the clients initiate requests in a short time interval at peak times (i.e. all the clients arrive at work in an interval between 7.55 am and 8.05 am and access the database server to retrieve an certain information required before beginning their work). Therefore, a WSP has to reserve enough resources in order to satisfy the maximum possible number of requests.

Web traffic is best characterized by heavy tailed probabilistic distributions such as Pareto or Weibull [10]. With such an arrival distribution, heavy traffic (greater than 90% of maximum number of requests) is more likely to occur in less than 10% of the time. For the rest of the time (i.e. the rest of the day) the server will be underutilised and thus uneconomic.

Our approach proposes to fill the idle times with low priority Web requests from another class of users providing a secondary class of services (basic services)as in figure 1.

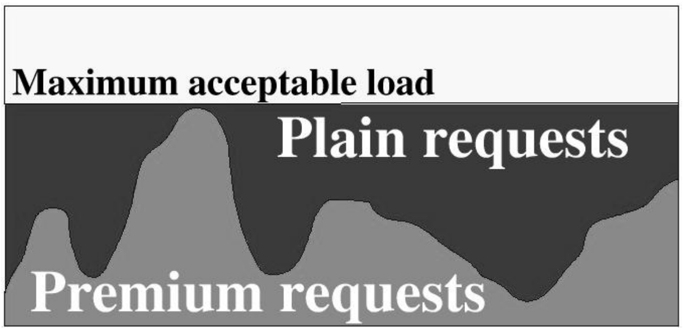

Fig. 1
TWO CLASSES OF WEB REQUESTS ON AN HTTP SERVER

2. MPLS BASED WEB SWITCHING

There are various alternatives to Web traffic load balancing [1]. We consider the MPLS approach.

The Internet is a connectionless network. Nevertheless, the WWW architecture, uses the HTTP application layer protocol to deliver information. Moreover, HTTP is based on TCP layer 4 protocol which is a connection-oriented protocol. Meanwhile, MPLS is a connection-oriented protocol. Therefore, a natural approach to load balance the HTTP flows is to map them into MPLS LSP's. The idea is to use different labels to specify the flows for each server across the cluster.

The first proposal for the use of MPLS for Web routing was presented in a IBM research report [11]. Since MPLS provides better mechanisms to support QoS routing than the legacy IP [12], [13], [14],it can more elegantly support such functions as:

- **content-based routing**
- **client affinity**
- **different classes of service**
- **load balancing**

The technique proposed in [11] requires that the dispatcher will maintain a table of associations between labels and the associated server weight. The dispatcher will then send a tuple $< \{L_1, W_1\}, \{L_2, W_2\}...\{Ln, Wn\} >$ to a proxy server situated in front of MPLS ingress nodes using a dedicated signaling protocol.

The approach proposed in [1], reduces the load of the dispatcher and the need for a dedicated signaling protocol. It also reduces the complexity of the solution by eliminating the proxy nodes used by [11] at the client side. Moreover, the functionality of an MPLS powered Web switching technique is preserved. The framework was implemented using cost-effective PC based MPLS switches.

In this paper we are interested in using MPLS to provide **different classes of service** to clients, based on service level agreements or other administrative factors. The MPLS approach can provide different FECs for different classes of service. Packets can be labeled corresponding to the type of the class (gold, silver or bronze). If servers have different processing performances, the gold-labeled packets can then be switched to the best performing server. Label stacking also can be used to define and identify hierarchical classes of service.

This approach presumes that the ISP providing the Web service already uses an MPLS enabled network. All the ISP's devices are MPLS capable. The clients for the Web service do not have to implement MPLS since the ingress of the ISP's administrative domain will be the ingress of an autonomous MPLS domain as well. The solution involves the use of a front-end dispatcher and a Web server farm as in Figure 2.

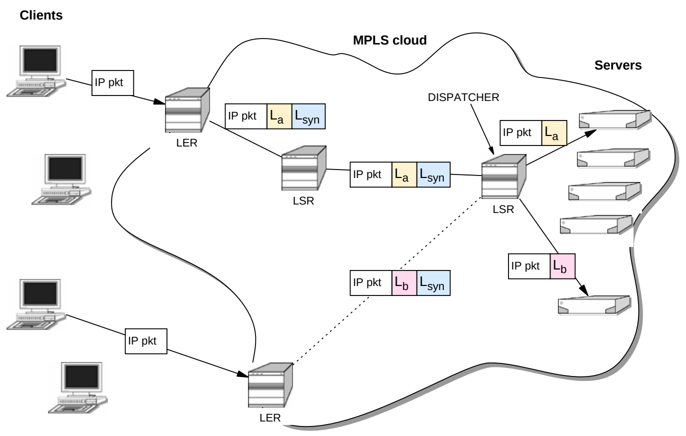

Fig. 2
A FRAMEWORK FOR MPLS WEB SWITCHING

The main advantage of this approach is that the edge routers will associate the requests to servers in a distributed manner. Consequently, the dispatcher will perform as an ordinary MPLS switch with an added load-balancing function.

3. GUARANTEED WEB SERVICES FRAMEWORK

3.1 *Alternatives*

Most commonly used techniques to provide differentiated Web service are done at the server site [8], [15]. Those technologies modify the server in order to treat differently the packets belonging to different classes.

- **User-level approach:** implies modifying the Web server program by adding a process scheduler that decides the order in which requests be serviced [15].
- **Kernel-level approach:** The kernel of the operating system is modified so that request priorities are mapped into process priorities.

The above alternatives can be used to provide differentiated levels of Web services but both of them require modifications within the operating system and/or the Web server. Moreover, neither can exploit the features of a QoS enabled network.

The **network approach** to differentiated Web services optimally exploits an traffic engineered network. Using a fast switching protocol (MPLS), our proposal will provide guaranteed Web services by load balancing the classes of requests among a cluster of web servers. The technique is described in section 3.2.

3.2 *Dynamic weighted load balancing*

This section depicts a framework for QoWS using the MPLS load balancing technique described in section 2.

We previously mentioned that for guaranteed web services we need to reserve enough resources (i.e. Web servers) for the maximum possible Web traffic. We may also want to use the idle times for basic traffic. In this approach we use an MPLS aware dispatcher in front of the Web cluster to switch the traffic according to a load balancing algorithm. Here we present the main issues of this approach.

1) Traffic classification: Since we use two classes of services we have to be able to map the requests to the correct class. Here we take advantage of an MPLS enabled network by using labels to classify the traffic. At the edge of the MPLS cloud, the requests are labelled according to their class of service. The dispatcher can identify the classes of requests based on the MPLS label value.

2) Traffic estimation and load distribution: To simplify the analytic model, we assumed that the server could generate the response in a time t_1 (e.g. 10 seconds) for each request if a single request is processed at one time. Empirical tests proved that the execution time of a CGI script increases linearly with the number of concurrent executions. Therefore, we can consider $t_x = a * x + b$, where x is the number of simultaneous processes and a and b are parameters that depend on the CPU speed and process complexity but can be previously estimated for a certain system and a specific request.

The agreement between WSP and client can specify that the requests must be served in a time less than t_{max}. This means that no more than $x_{max} = t_{max}/a - b$ requests should arrive simultaneously at one server. Then we can use x (number of active connections) as a parameter to control the execution time of a process.

Considering a number of c clients, the maximum number of simultaneous requests is c. In a Web cluster with n servers, with an ideal load balancing algorithm, the requests are equally distributed along servers. Therefore each server will encounter maximum number of c/n requests. We have then:

$$\frac{c}{n} \leq x_{max} \Rightarrow n \geq \frac{c}{x_{max}} \Rightarrow n \geq \frac{c}{\frac{t_{max}}{a} - b} \qquad (1)$$

The number of servers (n) can then be predicted using formula 1.

Let $S = \{s_1, s_2, ...s_n\}$ be the set of n Web servers. Our proposal uses 2 disjunct subsets of S for the two classes of requests: $S_h = \{s_1, s_2, ...s_i\}$ for high-priority requests and $S_l = \{s_i + 1, s_i + 2, ..., s_n\}$ for low-priority requests, where $0 < i \leq n$, $S_h \cap S_l = \emptyset$ and $S_h \cup S_l = S$. The role of the dispatcher is to balance the load among the subsets and map premium traffic and basic traffic to S_h and S_l respectively.

Two major conditions should be satisfied by the system. The number of premium requests per server must be less than the number of basic requests per server in order to always provide better services to the premium clients ($x_{premium} < x_{basic}$). The other condition is to keep the number of basic connections under a threshold for the situation in which in a short time interval, the set of servers S_l should accommodate incoming premium requests:

$$x_{threshhold} > x_{basic} > x_{premium} \qquad (2)$$

Variations of request arrival rates from both sources (premium and basic) will trigger modifications in both S_h and S_l. Consequently, we can define two events as follows:

E1 The first event is characterized by increasing the number of servers in S_h and respectively decreasing the number of servers in S_l. This is due to $\mid S \mid = \mid S_h \mid + \mid S_l \mid$. E1 is triggered at the arrival of an premium request when by accepting the incoming request the system may not accomplish the fundamental conditions mentioned in equation 2.

E2 The second event is characterized by decreasing the cardinality $\mid S_h \mid$ and increasing $\mid S_l \mid$. E2 is triggered at the arrival of an basic request by the following occurrence:

$$x_{threshhold} \gg x_{basic} \gg x_{premium} \qquad (3)$$

This is when the premium traffic is light and the system is underutilised. Hence, we can restrict the premium requests to a smaller subset of servers and by that, allowing a greater number of basic requests to be served.

When the premium traffic is heavy the dispatcher can refuse new incoming basic requests and if necessary even broke running basic connections to free resources for the burst of premium requests.

The proposed solution was simulated and the results are analyzed in section 4.2

4. Simulation results

4.1 Relation between execution times and the number of concurrent requests

Our first analysis was concerned with the behavior of an overloaded Web server. For this experiment we used a system with an AMD K-6 cpu (233MHz) and 64MB RAM running the Linux operating system and Apache[16] Web server. We created a CPU intensive CGI script that executes at the server side in approximately $t_1 = 10$ seconds. We incremented the number of concurrent requests and we analyzed the request execution times. The results in figure 3 shows that the time increases by a linear equation: $t = a * x + b$. x represents the number of simultaneous requests while a and b depend on the system characteristics and the CGI script's computational complexity. We also found a limit for the number of simultaneous connections. Over this limit the server is overloaded and the incoming connections will be dropped.

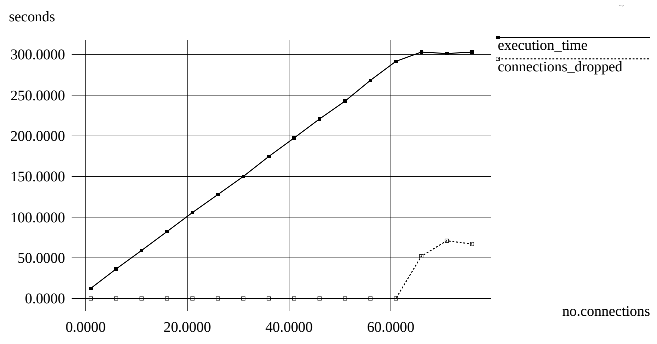

Fig. 3

Execution times for concurrent connections

4.2 Adaptive load balancing distribution

For the second experiment we simulated the model for providing guaranteed Web services. We considered two sources of traffic: premium traffic and basic traffic. The requests arrival is modelled by a heavy tailed probabilistic distribution of inter-arrival times (Pareto). Execution times are given by the results from the previous experiment: $t = ax + b(a = 5, b = 5)$. Premium traffic requests arrive with inter-arrival times given by Pareto distribution with an mean arrival rate $\lambda = 1$ connections/second and basic requests with a mean of $\lambda = 2$ connections/second. The load is dynamically distributed among the 8 servers within the cluster. The subsets of servers dedicated for the two classes of request had initially the cardinality $\mid S_h \mid= 1$ and $\mid S_l \mid= 7$.

The adaptive algorithm used the *E1* and *E2* events to maintain the premium services response times below a value of 50 seconds and the basic services response time above the times for the premium requests. Part of the basic requests were dropped at peak times but more basic connections were accepted when the servers were lightly loaded as shown in figure 4.

A higher arrival rate for basic requests will not affect the premium services while the conditions 2 are satisfied. The simulation proves that two classes of services can be delivered using a load balancing architecture with different weights for the two classes. Moreover, the premium service can be guaranteed since the cluster was prepared to serve the maximum possible number of requests and basic requests will always make way for premium requests.

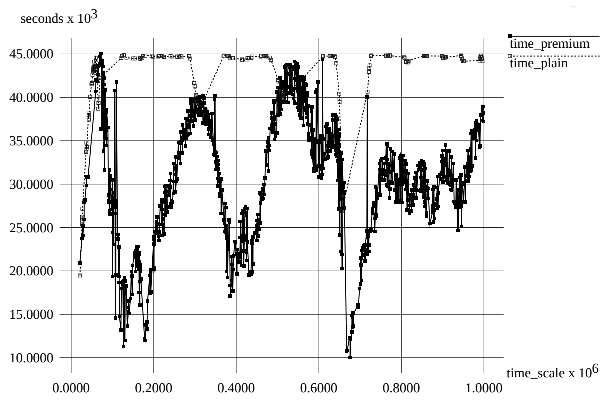

Fig. 4
EXECUTION TIMES FOR PREMIUM AND BASIC REQUESTS

5. CONCLUSIONS

In this paper we described a challenge faced by today's Web service providers. The traffic through the Web sites increases along with the number of clients and the number of services offered. In this context, separating the clients in classes of priority can improve the performance of a Web content hosting site. Economical parameters may also impose a differentiation between potential classes of clients. By providing a solution which uses MPLS, we assume efficient interaction with the favored protocol for high-speed QoS aware networking in today's Internet.

REFERENCES

[1] Radu Dragos, Sanda Dragos, and Martin Collier. Design and implementation of an MPLS based load balancing architecture for Web switching. To apear in the Proceedings of 15th ITC Specialist Seminar, Würzburg, Germany, July 2002.

[2] Yahoo. Investor Relations. URL: http://www.yahoo.com/info/investor.

[3] Eric Dean Katz, Michelle Butler, and Robert McGrath. A scalable HTTP server: The NCSA prototype. *Computer Networks and ISDN Systems*, 27(2):155–164, 1994.

[4] J. Wroclawski. The Use of RSVP with IETF Integrated Services. Technical Report RFC2210, IETF, September 1997.

[5] S. Blake, D. Black, and M. Carlson. An Architecture for Differentiated Services. Technical Report RFC2475, IETF, December 1998.

[6] A. Mankin, F. Baker, and B. Braden. Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement Some Guidelines on Deployment. Technical Report RFC2208, IETF, September 1997.

[7] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. Technical Report RFC3031, IETF, January 2001.

[8] Valeria Cardelini, Emiliano Casalicchio, and Michele Colajanni. A Performance Study of Distributed Architectures For The Quality of Web Services. *Proceedings of the Hawai'i International Conference On System Sciences*, January 2001.

[9] Huican Zhu, Ben Smith, and Tao Yang. Scheduling Optimization for Resource-Intensive Web Requests on Server Clusters. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 13–22, 1999.

[10] Paul Barford and Mark Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In *Measurement and Modeling of Computer Systems*, pages 151–160, 1998.

[11] A. Acharya, A. Shaikh, R. Tewari, and D. Verma. Scalable Web Request Routing with MPLS. Technical Report RC 22275, IBM Research Report, December 2001.

[12] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. Technical Report RFC2702, IETF, September 1999.

[13] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in Internet. Technical Report RFC2386, IETF, August 1998.

[14] Sanda Dragos, Radu Dragos, and Martin Collier. Bandwidth Management in MPLS Networks. URL: http://telecoms.eeng.dcu.ie/symposium/papers/B1.pdf.

[15] Lars Eggert and John S. Heidemann. Application-Level Differentiated Services for Web Servers. *World Wide Web*, 2(3):133–142, 1999.

[16] Apache Web Server. Apache Web Site. URL: http://www.apache.com.