MPLS-based Request Routing

Arup Acharya, Anees Shaikh, Renu Tewari, Dinesh Verma
IBM T.J. Watson Research Center
Hawthorne, NY 10532
{arup,aashaikh,tewarir,dverma}@watson.ibm.com

Problem Scenario

Current high-volume, high-availability data centers and content distribution networks use clusters of Web servers or caches to achieve scalability and reliability. In such environments a front-end dispatcher (or router) directs incoming client requests to one of the server machines. The request-routing decision can be based on a number of criteria, including server load, client request, or client identity. Dispatchers may be broadly categorized into two types: layer-4 dispatchers which base decisions on TCP and IP headers alone, and layer-7 dispatchers which use application layer information (e.g., HTTP headers) to direct clients. Content-based routing, for example, requires the dispatcher to terminate the incoming connection, examine the higher-layer headers, and create a new connection with the appropriate server (or transfer the incoming connection). As Web application requirements evolve, there will be a need for more sophisticated request routing, which implies that examining only transport-layer headers provides insufficient functionality. But layer-7 dispatchers, while more sophisticated, suffer from limited scalability and performance since they must perform connection termination and management for a large number of clients. Ideally, a dispatcher should provide the flexibility of layer-7 forwarding, while maintaining performance levels comparable with layer-4 hardware switches. In this synopsis we describe an approach that, when combined with an intelligent client-side proxy, can implement a dispatcher using commercial, high-performance, off-the-shelf switching hardware, while also providing the flexibility of a content-based router.

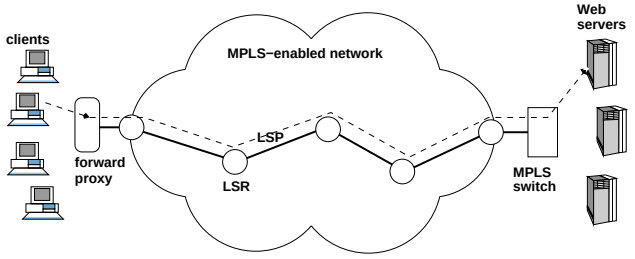

Figure 1: MPLS-based request routing architecture

Solution Overview

We leverage the growing migration of networks to Multiprotocol Label Switching (MPLS) to provide flexible routing and sophisticated traffic engineering capabilities. MPLS provides a circuit-switching service in a hop-by-hop routed network by grouping related packets using a shared, fixed-size label, allowing the packets to be routed and treated the same way in the network. Though standard usage of MPLS involves establishment of arbitrary label-switched paths (LSPs) for forwarding particular classes of traffic, MPLS labels do not have built-in semantics, in contrast to port numbers or IP addresses, for example. We take advantage of this flexibility by mapping application-layer information onto labels to enable high-performance Web switching, rather than using labels to express routing and forwarding policies. Figure 1 shows the overall architecture, which consists of an MPLS-enabled client-side proxy, an MPLS network, and a server cluster fronted by a dispatcher which includes a software controller and a standard MPLS switch.

The proxy is responsible for applying labels to packets belonging to client connections. The label applied by the proxy is used at the dispatcher to choose which server should handle the request. We exploit the label stacking feature of MPLS, which allows an inner label to be independent of any outer labels used to route the request through the MPLS network. The mapping of client connections to labels is communicated by the dispatcher to the proxy using a persistent control connection. Depending on the mapping, the dispatcher can support a variety of functions without having to terminate TCP connections, including content-based routing, client-server affinity, client-specific service differentiation, and server load balancing.

This approach has several key advantages over competing solutions that use a front-end Web switch. First, it removes the main bottleneck where all TCP connections are terminated and application-layer information is examined. Second, it lends itself to realization in a standard off-the-shelf MPLS switch, obviating the need for specialized, layer-7 Web switch hardware. Finally, using this approach, many of the functions typically available only with specialized hardware are provided at a much improved price-to-performance ratio. Unlike conventional schemes, this technique assigns some of the dispatch function to the proxy which requires that the proxy is MPLS-aware and that it implements the protocol to receive mapping information from the dispatcher.

Deployment of our scheme relies on wide deployment of MPLS in core networks and access networks, which is supported by reports from several large ISPs and equipment vendors. Also, proxies at the edges of the network must be modified to implement MPLS and support the protocol to communicate with dispatchers in server installations. This is appropriate in settings where proxies and Internet data centers are under a single administrative control, e.g. most large ISPs offer Web hosting service. Similarly, a large corporation with many branch offices and business partners could use the scheme to improve the performance and capacity of its Intranet/Extranet Web servers.

Scalability is a concern if we assume that every forward proxy accessing a web site participates in this scheme. Instead, we expect that in many cases a relatively small set of proxies contributes the bulk of the traffic to a site, and it is feasible to modify only those proxies. Another scenario that justifies proxy participation is one in which the Web hosting provider wants to provide service level agreements to particular clients. In this case the hosting provider requires proxy participation from those clients as part of the agreement.

For additional information and references please see the project web site at `http://www.research.ibm.com/mplsws`.