Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp212-216)

Integrating Network QoS and Web QoS to Provide End-to-End QoS

Wang Fei   Wang Wen-dong   Li Yu-hong   Chen Shan-zhi
State Key Lab of Networking and Switching ,
Beijing University of Posts & Telecommunications,
Beijing 100876,
CHINA

Abstract: To provide end-to-end QoS guarantees, it is not sufficient to provide QoS in the network layer or in the web servers alone. This paper analyzes and studies web QoS and network QoS, and proposes a scheme which considers both network QoS and web QoS. The scheme can provide the end-to-end QoS.

Keyword: Network QoS, Web QoS, MPLS,Diffserv

1 Introduction

With the evolvement of the E-commerce, the Internet is undergoing substantial changes from a communication and browsing infrastructure to a medium for conducting business and selling services. These changes place the Web server at the center of the E-commerce infrastructure with increasing requirement s for providing service differentiation and performance assurance. So web servers must have mechanisms and policies for establishing and supporting QoS. In addition, to provide end-to-end QoS guarantees, it is not sufficient to provide QoS in the network layer or in the web servers alone.

In this paper, we propose a scheme which considers both network QoS and web QoS in order to provide the end-to-end QoS.

The organization of this paper is as follows. In section 2, we describe our web server with QoS model. In section 3, we discuss network QoS. We propose a framework integrating network QoS and web QoS in section 4 and conclude in section 5.

This work was supported by the National Natural Science Foundation of China (No. 60472067), the National 863 Program of China (No. 2005AA 121630) and NCET-04-0113.

2 Web Server with QoS Model

Web servers are increasingly serving dynamic web pages.They run servlets, scripts, beans etc to generate web pages. In order to provide guaranteed response, this processing must be done according to some QoS. In our model, each HTTP request determines the QoS by the connection manager[1].

The connection manager uses request classification policy to determine the QoS to be provided to the HTTP request. Once the required QoS is determined, an admission control decision has to be made. Requests are then queued up in the appropriate queue. There are priority classes, and the request is placed in one of them. Once queued, the processing of these requests depends upon the scheduler[2]. When the reply is ready to be transmitted back, the appropriate DiffServ marking is set on the socket to be used. This marking indicates to the underlying the network QoS that is expected by the transmitted packets. Figure 1 depicts the procedure[3].

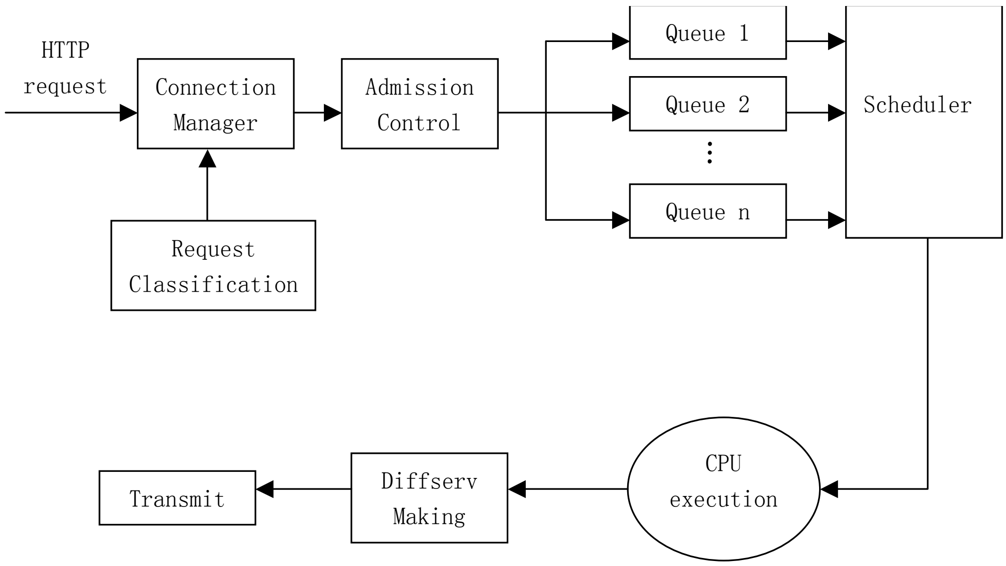

Fig. 1 Web Server with QoS Model

2.1 Connection Manager

The connection manager intercepts all requests and classifies the requests and places the requests on the appropriate queue. Since all requests must be accepted by the connection manager it is essential that the connection manager runs frequently enough to keep request queues full. If the connection manager does not run frequently enough worker processes may execute requests from lower classes because all the requests from higher classes have been processed even though there may be top-class requests waiting to be accepted. This results in a server that is more “fair” but may allocate server processes to lower-priority work, and thus make it difficult to quickly respond to newly arriving premium requests, thereby violating the preferential treatment policy.

2.2 Request Classification

A key requirement to web server with QoS is the ability to identify and classify the incoming requests of different classes. There are several ways to classify requests. These classification mechanisms can be divided into two categories, user class-based or target class-based. User class-based classification characterizes requests by the source of the request; target class-based classification classifies by the content or destination of the request[4].

2.2.1 User class-based classification

The client IP address is used to distinguish one individual client from another. This method is the simplest to implement. However, the client IP address can be masked due to proxies or fire walls, so this method has limited application.

HTTP cookies, a unique identifier sent to the browser, can he embedded in the request to indicate to which class the client belongs. For example, a subscription to a particular service is implemented as a persistent cookie. A cookie can also be used to identify a session that has been established for session-based classification.

Browser plug-ins can also embed special client identifiers in the body of each HTTP request. Such a plug-in could be downloaded by clients who have paid for a subscription to premium service.

2.2.2 Target class-based classification

The URL request type or filename path can he used to classify the relative importance of the request. In this case the sender of the request is irrelevant. Content can be classified as mission-critical, delay-sensitive, or best-effort. This would allow e-commerce purchase activities, for example, to have higher priority than browsing activities.

Destination IP addresses can he used by a server when the server supports co-hosting of multiple destinations (Web sites) on the same node.

2.3 Admission Control

When the server processing rate falls behind the client demand rate, the server becomes unresponsive to both premium and basic classes. To protect the server from high client loads some requests must be rejected. Naturally, basic requests rather than premium requests should be rejected first, and existing sessions should be maintained. Admission control of basic requests is triggered when the server starts to be loaded.

2.4 Request Scheduling

After requests are classified according to one of the above classification schemes and admitted by admission control, the server must actually realize different service levels for each class of requests. This is done by selecting the order of request execution. Workers are autonomous processes that select requests to process based on the scheduling policy. The scheduling policy may depend on queue lengths. Worker processes may be able to execution requests from any class, or, to reserve capacity for higher-class processes, they may be restricted to executing premium-class traffic. Below we outline several potential policies[4].

Strict priority schedules all higher-class requests before lower-class requests even when low-priority requests are waiting.

Weighted priority schedules a class based on its weighted importance. For example, one class will get twice as many requests scheduled if its class weight is twice another's.

Shared capacity schedules each class to a set capacity, and any unused capacity can be given to another class. The class may also have a minimum reserve capacity that cannot be assigned to another class.

Fixed capacity schedules each class to a fixed capacity that cannot be shared with another class.

Earliest deadline first establishes schedules based on the deadline for completion of each request. This can be used to give predicted response time guarantees.

3 Network QoS

IETF has done a lot of work in standardizing mechanisms to provide QoS for IP networks. By now two QoS architectures have been specified: Integrated Services (IntServ) and Differentiated Services (DiffServ). The IntServ approach can not be deployed in large-scale Internet backbones. So we adopt DiffServ as the basis of QoS mechanism. Our framework is based on the below assumptions.

3.1 Assumptions

(1) Internet backbone is MPLS network supporting DiffServ.
(2) Access networks are configured as DiffServ network.
(3) Internet backbone has enough resources to supply QoS guarantee.

3.2 Diffserv over MPLS

RFC3270[5] specifies a solution for supporting the Diffserv Behavior Aggregates whose corresponding PHBs are currently defined over an MPLS network. This solution also offers flexibility for easy support of PHBs that may be defined in the future. This solution relies on the combined use of two types of LSPs: E-LSP and L-LSP.

A.E-LSP(EXP-Infered-PSC LSP)

LSPs which can transport multiple Ordered Aggregates, so that the EXP field of the MPLS Shim Header conveys to the LSR the PHB to be applied to the packet (covering both information about the packet's scheduling treatment and its drop precedence).

B.L-LSP(Label-Only-Infered-PSC LSP)

LSPs which only transport a single Ordered Aggregate, so that the packet's scheduling treatment is inferred by the LSR exclusively from the packet's label value while the packet's drop precedence is conveyed in the EXP field of the MPLS Shim Header or in the encapsulating link layer specific selective drop mechanism (ATM, Frame Relay, 802.1).

In E-LSP, LSR may obtain TOS(type of service) of packet from header label of MPLS direct. It doesn't need to modify the structure of existing ILM(Incoming Label Map) and FTN(FEC To NHLFE). So we adopt the mode of E-LSP.

4 The Integrating Network QoS and Web QoS

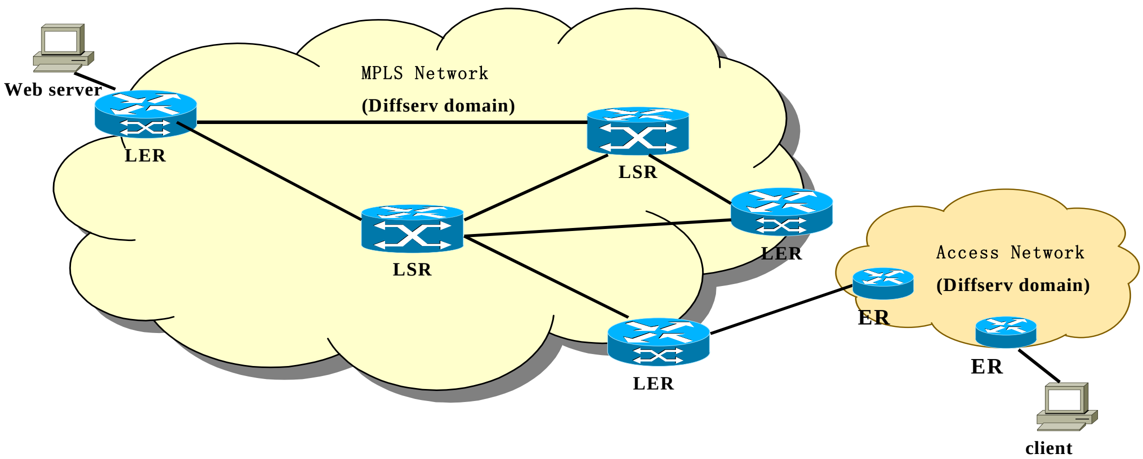

Fig. 2 the QoS framework integrating network QoS and web QoS

Note:

ER — Edge Route; LER — Label Edge Route; LSR — Label Switch Route;

The web server determine the reply Diffserv making. LER classifies the traffic and aggregates into BA, and mark by DSCP in the backbone network. Then packet is transmitted by E-LSP. When the traffic leave the backbone network, it enter access network through SLS negotiation between access network and backbone network. ER of access network reclassifies the traffic and aggregates into BA, and mark by DSCP. Core routers transmit packet through PHB related per BA. The reply is sent to the client finally.

5 Conclusions

In this paper, we propose a scheme which integrates network QoS and web QoS to provide the end-to-end QoS. In our model, each HTTP request m determines the QoS by the connection manager on the basis of request classification policy. Once the required QoS is determined, an admission control decision has to be made. Requests are then queued up in the appropriate queue. When the reply is ready to be transmitted back, the appropriate DiffServ marking is set on the socket to be used. This marking indicates to the underlying the network QoS that is expected by the transmitted packets.

In future work, we will use the system parameters to integrated network QoS and web QoS.

Reference:

[1] N. Bhatti and R. Friedrich. Web server support for tiered services. IEEE Network, September/October 1999.64~71

[2] V.R. Subramaniam and C.K. Tham. Integrating Network QoS to Provide Differentiated Service in a Web Server. IEEE,2002.

[3] C.K. Tham and V.R. Subramaniam. Integrating Web Server and Network QoS to Provide End-to-End Service Differentiation. IEEE,2002.

[4] C. Lin, Z.G. Shang and F.Y. Ren. QoS of Computer Network. TsingHua Publishing Company,2004

[5] F.L. Faucheur and B. Davie, et al. Multiprotocol Label Switching Support of Differentiated Service. RFC 3270,2002