Chapter 12

AUTONOMIC ADMISSION CONTROL FOR NETWORKED INFORMATION SERVERS

Giuseppe A. Paleologo
Mathematical Sciences Department, IBM T.J. Watson Research Center
PO BOX 218, Yorktown Heights, NY 10598
gappy@us.ibm.com

Nicholas Bambos
Department of Management Science and Engineering
Terman Engineering Center, Stanford University, CA 94305
bambos@stanford.edu

Abstract

While the issue of enabling performance guarantees on the Internet has been the subject of intense research in recent years, the problem of enabling QoS guarantees in edge servers has received relatively little attention. The need for QoS guarantees is already present in today’s Internet: while most backbones operate at a low level of utilization, web servers are often congested and are the main cause for the delay experienced by the end user. Here, we present a novel approach to admission control and resource allocation of sessions in edge servers. The model we adopt is quite general and its implementation does not depend on the type of application supported by the server (e.g., http or SSL). We model the system as a single server accessed by \( N + 1 \) users. \( N \) users have a lower bound on QoS, while one “super-user” aggregates the best-effort traffic to the server. The control rules admit a simple interpretation. Admitted classes “track” a target delay which is slightly smaller than their lower bound. The choice of a conservative target protects them from the performance degradation caused by the arrival of candidate classes into the system. On the other hand, candidate classes follow a “slow start” mechanism, similar to the update rule for TCP Reno. The intuitive rationale for this choice is similar to that of congestion-control algorithms: by increasing their priorities slowly, the candidate classes do not degrade the QoS of the admitted classes below their upper bounds. This resource allocation algorithm enjoys several attractive properties: it is measurement-based, since it only relies on the measurement of each class’ delay during a busy cycle; it is de-
centralized, since each class updates its priority based on local information; and finally it is closed-loop, while most admission control schemes are open-loop. As a consequence, the algorithm does not require signaling to admit a new class.

Based on the above assumptions, we prove that the control scheme is asymptotically correct in the following sense: i) for small values of the constant $\epsilon$ and large values of $\alpha$ the average delay of the admitted classes is always less than their required bounds; ii) if the candidate classes are admissible, i.e., there exist a set of priorities for the server such that the average delay of both admitted and candidate classes are less than their upper bounds, then the candidate flows will be admitted.

Keywords: Admission Control, Autonomic Computing, Quality of Service, Queueing Systems

1. Introduction

While the issue of enabling performance guarantees on the Internet has been the subject of intense research in recent years, the problem of enabling QoS guarantees in edge servers has received relatively little attention. The need for QoS guarantees is already present in today's Internet: while most backbones operate at a low level of utilization, web servers are often congested and are the main cause for the delay experienced by the end user. Of particular interest is the design and implementation of algorithms that enable service guarantees for multimedia applications. For this class of applications it is important to guarantee bounds on delay and/or loss rate; at the same time, audio and/or video streaming applications tax system resources and can easily lead to congestion. The issue arises whenever

- two or more users share the same server;
- each user requires lower bounds on one or more QoS metrics, which must be guaranteed during the time he accesses the shared resources; and
- the perceived QoS is influenced by the presence of other users and their required lower bounds.

In principle, it is possible to imagine a centralized "omnipotent resource manager" (or ORM)\(^1\) that has the ability to:

1. receive the desired service requirements from the users and the current state of utilization from the server;

2. know the exact relationship between the amount of resources allocated to each user and the resulting QoS level, and decide whether adequate spare resource capacity exists to admit new users; and

3. allocate resources within the server based on the information gathered in the first two steps.