12.1 Introduction

In this chapter we examine the performance characteristics of the latest innovation in client/server technology—Web technology. Unlike the traditional client/server systems discussed in Chap. 11, each Web client typically makes high-frequency, short-term accesses to a relatively small number of servers.

First, we examine some elementary mistakes made in the course of taking Hypertext Transfer Protocol (HTTP) server performance measurements. Based on the queueing theory of Chaps. 4 and 5 and PDQ, we uncover the cause of these mistakes. Next, we analyze the performance a Web-based middleware architecture, which will require the introduction of two novel techniques to calibrate PDQ against the available performance data:

1. the introduction of “dummy” PDQ nodes to account for unmeasured latencies
2. a load-dependent PDQ node to account for the overdriven roll-off observed in the throughput data

These two techniques are extremely important for constructing realistic PDQ performance models. The reader might like to review the concept of load-dependent servers presented in Sect. 8.5.11.

12.2 HTTP Protocol

The HTTP is a Web protocol that uses the TCP/IP Internet transport protocol. The files are usually resident on remote file servers distributed across the internet. The protocol model is very simple. A client machine establishes a connection to the remote server machine, then issues a request. The server processes that request, returns a response with the requested data, and generally closes the connection.
Listing 12.1. getHTML.pl script

```perl
#! /usr/bin/perl
# fetch HTML from a URL

use HTTP::Request::Common qw(GET);
use LWP::UserAgent;
use POSIX;

# Set up and issue the GET ...
$url = "http://www.perfdynamics.com/";
my $ua = new LWP::UserAgent;
my $request = new HTTP::Request('GET',$url);
$request->content_type("application/x-www-form-urlencoded");
printf("%s
", $request->as_string);

# Print the result ...
my $result = $ua->request($request);
if (!$result->is_success) { print $result->error_as_HTML; }
printf("%s
", $result->as_string);
```

The request format in HTTP GET is straightforward, as listing 12.1 shows. It uses the powerful LWP (Library for WWW access in Perl) module. Running the script produces the result shown in listing 12.2.

The first line in the result specifies an object (an HTML file in this case), together with the name of an object to apply the method to. The most commonly used method is GET, which asks the server to send a copy of the object to the client. The client can also send a series of optional headers in RFC-822 format. The most common headers are Accept, which informs the server of object types that the client can accommodate, and User-Agent, which reveals the implementation name of the client. The response from the remote server starts at line 4.

The HTML belonging to the actual web page starts at line 18 in listing 12.2. The rest of the response, down to </HTML>, has been elided for brevity.

Responses start with a status line indicating which version of HTTP is running on the server together with a result code and an optional message. This is followed by a series of optional object headers; the most important of these are Content-Type, which describes the type of the object being returned, and Content-Length, which indicates the length. The headers are terminated with a blank line. The server sends any requested data, and drops the connection. HTTP transfers exhibit a common access pattern. A client requests a hypertext page, then issues a sequence of requests to retrieve any icons (connected by Web hyperlinks) referenced on the first HTML page. Once the client has retrieved the icons, the user will typically select a hypertext link to follow.