

EECS 423
Distributed Systems
2004 Fall Semester

Homework 4

Due at the beginning of class, October 28, 2004
No late papers accepted

Test Suite

The objective of this step is to construct a test suite for an e-Parcel server. A test suite is a collection of test sets. Each test case is a sequence of sessions from the same or from different clients. A test case consists of the exact sequence of e-Parcel requests that correct clients could make following RFCv0.2 (to be posted shortly). For example, a test case starts with a HELLO command and ends with a QUIT command. Basically, the test cases should be written in such a way that each session can be cut-and-pasted into an open telnet <e-Parcel server> <port number>. Each test case should also come with a list of expected responses from a correct e-Parcel server. The test cases should be written so that they are independent: the same outcome should ensue independently of whether another test case was previously run. For example, a test case should not assume that a previous run created an e-Parcel stub.

Write 3 test cases. The test cases should be of increasing complexity: the first one should contain no more than 4 commands and the last one should contain at least 10. Make sure to exercise all of the features of the e-Parcel specification across the 3 test cases. Send the test cases and the expected server replies both to the instructor and to the TA by e-mail.

The complete test suite will consist of 33 test cases (3 cases per student times 11 students) and will be posted on the class Web site.

Server Implementation

Implement a single-threaded e-Parcel server. You will re-use this server in future assignments, where you will be required to implement additional e-Parcel features. The server should be “unconditionally compliant” with RFCv0.2 (to be posted shortly). For extra credit, you can implement a pre-threaded or pre-forked version of the e-Parcel server.

Although e-Stub management is necessary and required, focus your project on the communication part, not on sophisticated or efficient data management. Many things need to be *avoided* to accomplish this: for example, you should encapsulate and hide away data management. Another suggestion is to use an off-the-shelf XML-library. The UNIX lab supports libxml, libxml2, and libxstl (C/C++) and J2SE for javax.xml.parsers (Java). You

can ask `help@eecs` to install other libraries and tools such as XML::Parser (for `/usr/local/bin/perl`), PyXML (Python), or SAX (Java). If you ask `help@eecs`, be sure to provide enough lead time for them to do it. Otherwise, you can install the appropriate libraries in your own directories. Note: the use of any of these libraries is only a suggestion to save you time and effort, and it is not required.

Here is a list of things that you need to hand in for this part of the assignment:

- Send to the TA the source code of the e-Parcel server, an executable if available, and instructions to compile and run the server.
- Keep a server running on your machine (or a machine in the lab) until the TA allows you to stop it. The running server will be used by the TA to grade your project. Send to the TA the address and port number of the server, and make sure that the server is generally Internet accessible (no NAT or firewalls).
- Send both to the instructor and to the TA by e-mail any changes to the RFC that you think is needed to implement a correct e-Parcel server.

Amendments

It is possible that the contents of RFCv0.2 will require amendments to this homework assignment. Please, consult the Web site and the mailing list for amendments.