

EECS 423  
Distributed Systems  
2004 Fall Semester

## Homework 6

Due at the beginning of class, December 2, 2004  
No late papers accepted

### Server Implementation

A new RFC (RFCv4) is posted. It contains minor modifications of the protocol. Modify your server code accordingly.

### Crashing Client Implementation

The objective is to implement a client that is subject to crash failures. As per Figure 7-1 in the textbook, a crash failure implies that the client “halts, but it is working correctly until it halts”. Additionally, the RFC states that “either end-point MAY close or reset the connection asynchronously”. The crashing client is similar to a telnet window in which you can paste test cases but you should be able to specify that:

1. The client will stall forever, or
2. The connection is to be closed, or
3. The connection is to be reset.

For example, the crashing client could generate commands such as:

```
MAI
```

or

```
ACCEPclose
```

or

```
DELETE vx111@po.cwru.edu EPP/0.1  
<stureset
```

The exact way the reset or close connection are implemented is your choice. For example, you could have command line arguments for stalling, reset, and close. After the connection is closed or reset the client application should exit. You may want to look at the file `nonblock/tcpcli03.c` for an example of connection reset.

The crashing client is introduced for testing purposes only: a very streamlined client would do. Ideally, a crashing client barely copies from standard input into a socket descriptor (and

from a socket descriptor to standard output) plus providing support for crash failures. Send the source code of the crashing client to the TA only.

### **Test Cases**

Write test cases of requests that could be generated by a crashing client which is otherwise conditionally compliant. Follow the same format, number of test cases, and other directions as in assignment 4. Send your test cases to both the TA and the instructor.

### **Server Implementation**

The purpose of this step is to make the protocol and your server implementation more robust to crashing failure.

1. Inspect the RFC and isolate issues, if any, that would occur during the interaction with a crashing client. If you do find problems, document them in your report, along with an explanation and a proposal to fix them. The fix should be cut-and-paste into the RFC. In general, you are allowed to add features to the protocol only if they are absolutely necessary to fix a problem. Selected fixes will be integrated in the authoritative RFC. Send your report to both the instructor and the TA.
2. Inspect your server implementation and isolate issues that would occur during the interaction with a crashing client. Document them in your report, along with an explanation, and your solution to the problem. Modify the server accordingly. Send your server implementation to both the instructor and the TA.
3. Test your server implementation against your crashing client and test cases.