Sfwr Eng/Comp Sci 4F03 (Winter 2017) Programming Assignment 3

1 Assignment

1.1 Problem

You are to design a program which will utilize UDP sockets and XDR remote procedure call (RPC) to construct a string S in parallel using N threads. S will be broken down into M segments of length L where the characters of each segment satisfy a property F.

1.2 Program Parameters

When "make" is called within the PA3 folder, your project should generate 3 programs:

- *./server_verify*, the application has no input parameter,
- *./server_append* < *host_name* >, the application has no input parameter,
- *./client F N L M C0 C1 C2 host_name1 host_name2*, the first 7 input parameters are similar to those in PA1 and PA2, and the two last ones are host names of append and verify servers, respectively

1.3 Specification

The specification for the program will be the same as PA1 and PA2 with the following additions (see Figure 1):

- Your *client* program will consist of n threads.
- The client uses a remote procedure RPC_InitAppendServer to send *f, L, M, C0, C1, C2, host_name2* parameters to the append server, and RPC_InitVerifyServer to send *N, L, M* parameters to the verify server.
- On receiving the parameters, the servers setup themselves. Moreover, the verify server also initializes a UDP socket to receive S from the append server.
- The client launches N local threads.

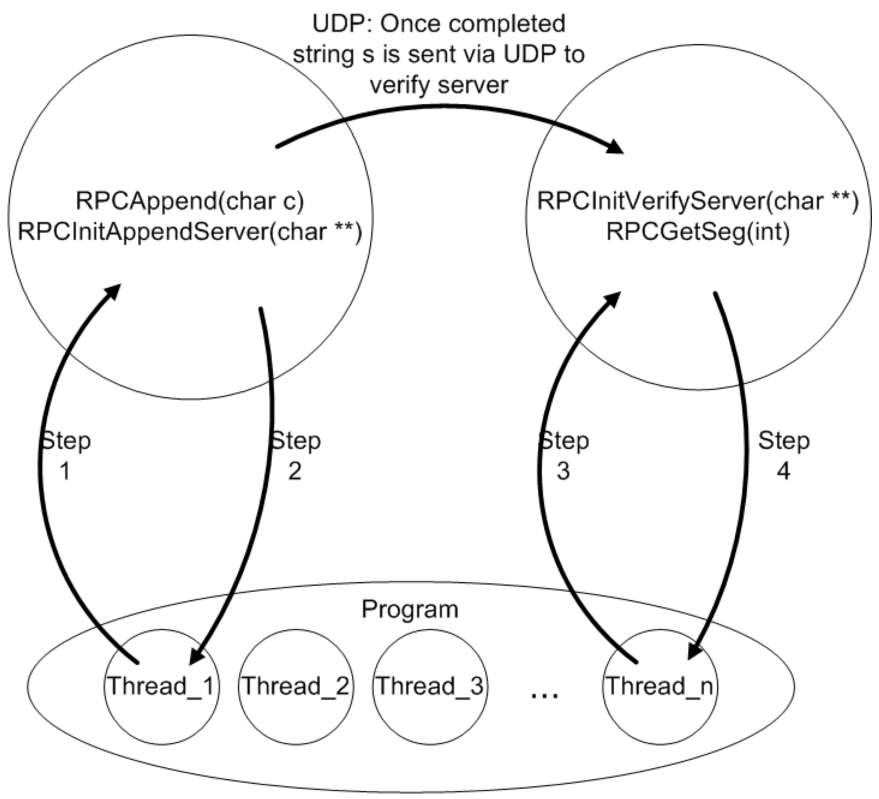
UDP: Once completed string s is sent via UDP to verify server
RPCAppend(char c)
RPCInitAppendServer(char **)
RPCInitVerifyServer(char **)
RPCGetSeg(int)
Step 1
Step 2
Step 3
Step 4
Program
Thread_1
Thread_2
Thread_3
...
Thread_n

Figure 1: Program flow diagram

- Each thread will try to append its assigned character to S by calling a remote procedure RPC_Append.
- RPC_Append will either return 0, if appending was successful, or -1 when S is complete.
- When S is fully constructed, the append server will send S to the verify server via UDP socket. Moreover, each client's thread that receives -1 from the append server will then call another remote procedure RPC_GetSeg from the verify server to retrieve a segment to verify. If the segment satisfies the property f, then a local counter is incremented.
- Each thread will continue to call RPC_GetSeg() until RPC_GetSeg returns a string starting with a "-", indicating that no more segments are available.
- The threads will then perform a summation reduction of the local counter, and the master thread will output the string S and the count to terminal.

- Both remote procedures should run on `moore`. Client threads should be able to run on any machine in the CAS network.

1.4 Example

Please see Figure 1 for reference:

1. Thread_1 sleeps for a random amount and then tries to append "a" to S by calling PRC_Append("a").
2. It continues to do so until it receives a -1.
3. Thread_n has received value -1, and now calls PRC_GetSeg(n).
4. Since this is the first time this call was performed, RPC_GetSeg will setup a UDP socket to receive S from the append server. It then sends the first segment back to Thread_n.
5. Thread_n verifies the segment, and then calls PRC_GetSeg(0) again to get the next available segment.
6. If Thread_n receives a '-', it knows to perform a summation reduction with all threads.
7. The program then prints out the string as well as the total count of valid segments.

1.5 Grading

- Up to 40% for generating S using N threads with RPC_Append.
- Up to 50% Above + verify server gets string S from append server via UDP.
- Up to 60% Above + checking segments serially using RPC_GetSeg.
- Up to 70% Above + checking segments using N threads.
- Up to 100% Above + for generating S using N threads and enforcing that all segments will satisfy property F (check is 100%).
  - This also entails a check when the program starts weather segments of length L and alphabet size N are capable of satisfying property F.
  - If the check fails, inform user that other parameters need to be selected.

1.6 Submission

Your solutions must be submitted by **11:59pm on Monday March 20** in the provided SVN folder (see below)

All your source files and makefile should be located in PA3.

Please ensure that the program will work on the department machines.