Simulation of Multi-port Server, Error Detection Techniques and Network Games

Type: Group
Note: I encourage both inter and intra group discussion. Each member of a group must contribute in the development process, otherwise you may not be able to meet the deadline.
Due: 27, Sep 2016. Partial evaluation on Sep 20.

Module 1: Multiport Server

Configure a server to listen to the following ports and the application to which it will be listening is also given.

- 3100: Fibonacci Series
- 3200: Factorial
- 3300: IsPrime
- 3400: IsPerfectSquare
- 3500: Random number generation
- 3600: Chat
- 3700: Text file Transfer
- 3800: Dummy

Description: A client who wishes to connect to the server can connect to any of the ports. (i) Create 8 clients and make the first 6 listen to any one of the ports 3100 to 3500. (ii) Client 7 connects itself to 3500 and ask for a random number x in the range [1..8] (except 7). Further, on receiving x, Client 7 performs a chat with Client x. Client 8 is initially connected to port 3800. Suppose the value of x is 8, the server informs 7 and 8 that both will perform 'chat' soon. On seeing server's message, both clients (7 and 8) close the current binding and bind itself to port 3600, and performs a chat. The server records the chat communication in a text file. Once the chat is over, the server informs again 7 and 8 that it must switch to port 3700. Further, the server broadcasts the 'chat communication text file' to clients 7 and 8 through Port 3700.

Module 2: Error Detection Techniques

Simulation of VRC, LRC, CheckSum, CRC-8 and CRC-12.
Description: For all detection techniques, generate a random binary string of size 512 and compute the overhead bits as per the scheme. Let the message to be transmitted is x. Choose 10 bits in x randomly and corrupt those 10 bits (flip their values) before transmission and let the modified message be x'. On receiving x', the receiver performs the appropriate detection scheme to check whether there was an error in transmission. Simulate the above task for 100 sample inputs each of size 512. Through simulation, assess which encoding scheme is good. Also, do the task by choosing 20 random bits.

Module 3: Network Games

`Think of a Number Game:`
Two clients c_1 and c_2 are connected a server and they both wish to play this game. The sample sequence is given below for implementation purpose. Ensure the code is generic in nature wherever possible. The values given below must not be hard coded as it is only for explanation purpose.

At C_1	At C_2
C_1 thinks some random positive integer, say x. C_1 brings x to a single digit y and performs $z = y * 9$, brings z to a single digit, say z'. z' is confidential and it is not disclosed to C_2. NOTE: if you bring 12345 to a single digit, you get $1+2+3+4+5 = 15, 1+5 = 6$. C_1 sends 'I am ready to play'	C_2 gets 'ready to play' message. C_2 gives a series of computational tasks to C_1.
hline	C_2 sends Add 6 to z'
C_1 gets 'Add 6' message, performs $z' = z' + 6$. C_1 sends 'I am done, what next'	
	Perform: Muliply 15 with z'
C_1 performs, $z' = z' * 15$, C_1 sends 'I am done, what next'	
...	...
	C_2 gives some sequence of tasks (add, sub, mul, factorial). After a while, C_2 says, i wish to end the game.
	Based on the above sequence C_2 guesses the value of final z' and sends the guess to C_1.
On receiving the guess from C_2, C_1 compares the guess and z'. C_1 sends the message ' I am surprised, how did you guess z' '	
Shall we play again	

Note: Use your creativity to model this problem. The sequence given above is just a guideline. You can see the above sequence as a chat and messages (computational instructions) are typed at the terminal.

`Friend/Enemy Game:` In any group of 6 people, there exist 3 mutual enemies or 3 mutual friends.

At C_1	At C_2
C_1 claims that the above claim is true. Sends the message 'claim is true' to C_2.	
	C_2 says ' I do not see the claim, can you prove by example'.
C_1 sends the graph G_1 on 6 vertices to C_2	
	C_2 verifies G_1 and says 'Perfect, not convinced, send the next graph'
C_1 sends G_2	
	C_2 verifies G_2 and says 'Perfect, not convinced, send the next graph'
...	...
C_1 eventually sends all $2^{\binom{6}{2}}$ to C_2	
	'I am convinced' well done C_1.

Note: Use your creativity to model this problem. The above sequence can be modeled as a chat and chat messages can be automated instead of typing at the terminal.

`Pumping Lemma:`

Objective: To convince a client that a given language is non-regular through chat. The following sequence can be modeled as a chat and chat messages are typed at the terminal. Refer to Automata Theory text to know more about pumping lemma.

At C_1	At C_2
C_1 claims $a^n b^n$ is regular and sends this to C_2	
	C_2 says, NO. C_2 generates a string z, satisfying the 'constraints' and sends it to C_1.
	C_2 instructs C_1; decompose $z = uvw$ such that $\|uv\| \leq n$ and $\|v\| \geq 1$. Any decomposition satisfying the constraint is fine.
C_1 sends the decomposition to C_2	
	C_2 chooses an integer $i \geq 0$ and show that $uv^i w \notin L$. This implies that L is non-regular. Sends this message to C_1 along with the value of i.
C_1 says, 'i am convinced', L is non-regular	
C_1 gives the next language a^p where p is prime to C_2	
the games continues	C_2 wins again