HW3 – Communication Subsystem Implementation

Estimated time: 24-36 hours

Objectives

- Continue to apply principles related to protocol design
- Explore architectural design choices that will lead to good modularization, encapsulation, abstraction, coupling, and cohesion for a communications subsystem
- Become more familiar with testing techniques
- Become more familiar with using infrastructure in the cloud

Overview

This assignment consists of three parts: design, implementation, and unit testing. For the first part, you will design a conversation-oriented subsystem that all your processes can use for UDP and TCP communications. It should not have to know the details of any specific protocol, but be able to process messages for any conversation. You will document your design so they the salient details are communicated effectively. UML classes, interaction, state chart, and activity diagrams are good chooses for communicating architectural and detail designed.

For the second part, you will implement an initial version of your subsystem, and a few key components of your main applications. However, choose wisely which components you implement now. Focus on those that will reduce your overall risk and help you refine your requirements, communication protocols, and communication subsystem. For HW3, the communication subsystem needs to be "mostly" functional, but not "fully" function. For example, it does not yet need to support highly reliable communications, yet. A good starting point for this part of the assignment is to implement your message class hierarchy.

For the third part, you will create executable unit test cases that thoroughly the core functionality of your communication subsystem.

As with HW1, your final execution environment will be virtual machines in a cloud environment. You must be able to run whatever parts you have completed on AWS EC2 instances.

Instructions

1. Study sample architectures
2. Study design principles and practices, like abstraction, encapsulation, modularity, programming to an interface, dependency inversion, as well as design patterns, like Strategy, Template Method, and Factory Method.

3. Brain storm design alternatives and consider which best fit your development environment and functional requirements. Remember that your communication subsystem should be included in all the programs that comprise your distributed application.
4. Document your design using UML class diagrams, interaction diagrams, state charts, and/or activity diagrams. See sample architecture
5. Implement your message class hierarchy and verify with executable test cases
6. Implement most of your communication subsystem and verify with executable test cases. You do not need to implement all aspects of reliability, nor do you need to implement any security at this point.
7. Start the programs for at least two processes, and implement at least one conversation to help test your communication subsystem design and implementation
8. Implement 15%-25% of your application, with a focus on the riskiest features
9. Run on virtual machines in a cloud, i.e., AWS

Submission Instructions

Zip up your design documents and entire implementation workspace into an archive file called CS5200_hw3_<*fullname*>.zip, where fullname is your first and last names. Then, submit the zip file to the Canvas system

Grading Criteria

Criteria	**Max Points**
A well-thought out, appropriate, and documented architectural design	30
A good implementation of your message classes, with thorough executable unit test cases	30
A good initial implementation of a reusable communication subsystem	30
Thorough executable unit tests for key components in the communication subsystem	30
Implementation of 15-25% of the distributed application's functionality, with light system testing in a cloud environment	30
TOTAL MAX POINTS	150