CPSC4780

Assignment 1

Due: February 11

The total is 25 marks.

Written questions (Due at the beginning of the class)

(1) (3 marks) *Transparency* is one of the challenges we face when designing a distributed system. We have discussed many transparencies, such as location transparency and failure transparency. For each transparency, give an example and discuss why it is important to achieve it.

(2) (3 marks) Discuss the issue of scalability in a distributed system and explain why it is so important to the design and implementation of a distributed system. You can use the Internet as example to make your arguments, even though it is not a true distributed system.

(3) (6 marks) Consider blocking and non-blocking send and receive. Usually, a non-blocking operation can similar a blocking operation. For instance, non-blocking receive can simulate a blocking receive.

(a) Suppose that you use non-blocking receive to simulate a blocking receive. Provide the pseudo code for the case of single threading and for the case where you can use multi-threading, respectively.

(b) If a system does not provide a non-blocking send, can you simulate it by a blocking send? If so, show the pseudo code. If not, explain why. Discuss your answer for the case of a single thread and for the case where you can use multiple threads, respectively.

(4) (4 marks) Answer the following questions related to Mach.

(a) Usually when a task sends out the receive right of one its ports to other tasks, the task does not have the receive right to the port any more. Explain why.

(b) Explain why the transfer of receive rights is more complicated than the transfer of send rights. You need to discuss your answer in the context of inter-machine transfer and intra-machine transfer. Propose your solution to the situation.

Programming question (Due at 11:59pm on February 11)

(5) (9 marks) Use Java sockets and multithreading to achieve a dynamic client/server paradigm, i.e., a thread is created by the main thread only when there is a new request.

The question is as follows. Suppose that a server maintains a huge disk, which is divided into tracks. A client can make a *read* request to a track, which will be handled by the

server. For each such a request, the server creates a new thread and let it transfer the data on the track back to the client. You can assume that all the requests are *read*.

Design a client/server paradigm to handle the situation. Your server program should be called server.java while your client program should be called client.java.

When you are done, send them to me through your e-mail account. The subject of your e-mail should be: “Assignment 1, your last name”. The body of your e-mail should contain the instructions as how to compile and execute your program.