Northeastern University

CS 6650 Scalable Dist. Systems

Homework Set #1 [100 points]

Late days left : 4

I. Study **Chapter 2 Systems Models** Coulouris Book [10 points]

Answer the following questions using explanation and diagrams as needed:

2.11 Consider a simple server that carries out client requests without accessing other servers. Explain why it is generally not possible to set a limit on the time taken by such a server to respond to a client request. What would need to be done to make the server able to execute requests within a bounded time? Is this a practical option?

Performance of communication channels is affected by the following: latency, bandwidth and jitter. The interaction between processes has delays in communication, the accuracy is limited by these delays and by the difficulty of maintaining the same notion of time. In addition, failure occurs in computers or in the network that connects the computers. Also the openness exposes the distributed systems to attack by external and internal agents. It is difficult to arrive at realistic values of process execution time, message delay and clock drift rates in a distributed system, and design based on the chosen values will bot be reliable.

Modelling an algorithm as a synchronous system may be useful in a real distributed system. It is possible to use timeouts to reflect failure of a process. Synchronous distributed systems can be built with know resource for which they can be guaranteed sufficient processor cycles and network capacity, and for processes to be supplied with clocks with bounded drift rates. Distributed systems such as the Internet are useful in terms of asynchronous distributed systems.

2.14 Consider two communication services for use in asynchronous distributed systems. In service A, messages may be lost, duplicated or delayed and checksums apply only to headers. In service B, messages may be lost, delayed or delivered too fast for the recipient to handle them, but those that are delivered arrive with the correct contents. Describe the classes of failure exhibited by each service. Classify their failures according to their effects on the properties of validity and integrity. Can service B be described as a reliable communication service? page 67, page 71

Server A: omission failures: messages may be lost; timing failures: messages delayed; and arbitrary failures: messages duplicated. Validity is compromised because message in the outgoing message buffer is not delivered to the incoming message buffer. Integrity is compromised because messages may be duplicated.

Server B: omission failures: messages may be lost; timing failures: message delayed or delivered too fast for the recipient to handle them. Validity is maintained because messages that are delivered arrived with the correct contents. Integrity is compromised because messages may be lost. Because reliable communication is defined in terms of validity and integrity, server B cannot be described as a reliable communication service. Unless the system can tolerate occasional messages loss and handle messages delivered at a high rate.

II. Please refer to the 2 articles (PDF) on Middleware for Distributed Systems. [10 points]
See Fig 1. layers of Middleware

What is middleware for Dist. Systems/ What are its uses (list and explain).
Consider a multiplayer game as a Dist. Sys. It is played on 4 players PCs and a Game Server. What are the heterogeneous elements among these 5 systems at each layer? Give examples to illustrate this (e. g. MacOS on PC1; Linux on PC 2 etc.) and then explain how middleware helps here.

Middleware is a class of software technologies designed to help manage the complexity and heterogeneity inherent in distributed systems. It sits between the application and the underlying operation systems, networks, and hardware. The uses are: bridge the gap between application programs and the lower level hardware and software infrastructure, ensuring parts of applications are connected and interoperate; providing reusable services that can be composed, configured and deployed to rapidly and robustly create distributed systems by integrating components from multiple technology suppliers.

The heterogeneous elements in the example of 4 players PCs and a Game Server at each layer include: operating systems, different PCs may run various operating systems such as MacOS, Windows, Linux. Hardware: such as processors, graphics cards, and memory capacities. Network: different network protocols. Software: different game software versions across PCs. Middleware helps provide a uniform interface for communication and interaction between these heterogeneous elements, ensuring the game operates smoothly. It abstracts away the complexities of the distributed environment.

III. From Chapter 3 Coulouris Book **Networking** [20 points]
Answer the following questions using explanation and diagrams as needed:
3.1 client sends a 200 byte request message to a service, which produces a response containing 5000 bytes. Estimate the total time required to complete the request in each of the following cases, with the performance assumptions listed below:
i) using connectionless (datagram) communication (for example, UDP);
ii) using connection-oriented communication (for example, TCP);
iii) when the server process is in the same machine as the client.
[Latency per packet (local or remote, incurred on both send and receive): 5 ms = 0.00008333 minutes
Connection setup time (TCP only): 5 ms = 0.00008333 minutes
Data transfer rate: 10 Mbps = 1.25 MBps = 75000 bytes/minute
MTU: 1000 bytes
Server request processing time: 2 ms =0.00003333 minutes
Assume that the network is lightly loaded.]
pages 82, 122
i) connectionless communication has no connection setup time.
Time to send request = (Request size / Data transfer rate) + Latency
(200 / 75000) + 0.00008333 = 0.00268333 minutes

Time to receive response = (Response size / Data transfer rate) + Latency
(5000 / 75000) + 0.00008333 = 0.06708333 minutes
Total time = Time to send request + Server processing time + Time to receive response
0.00268333 + 0.00003333 + 0.06708333 = 0.0698 minutes

ii) connection oriented communication has connection setup time.
Total time = Connection setup time + Time to send request + Server processing time + Time to receive response
0.00008333 + 0.00268333 + 0.00003333 + 0.06708333 = 0.06988333 minutes

iii) the server process is in the same machine as client, the latency is 0, and no connection setup time.
Total time = Time to send request + Server processing time + Time to receive response = 0.0026 + 0.00003333 + 0.0666 = 0.06923333 minutes

3.7 **part (ii) ONLY (*i.e.,* FTP) Compare connectionless (UDP) and connection-oriented (TCP) communication for the implementation of each of the following application-level or presentation-level protocols:**
ii) file transfer (for example, FTP);

TCP: transmission control protocol is connection oriented, it provides reliable ddelivery of arbitrarily long sequences of bytes via stream based programming abstraction, ensuring all data sent will be delivered correctly and in the same order at the destination. TCP also handles congestion control and provides mechanisms for retransmitting or corrupted data. For file transfer, TCP is a suitable choice when complete and accurate delivery of file is important.

UDP: user datagram protocol is a connectionless protocol. It sends packets without establishing connection or checking if they are received. It is faster than TCP, it does not guarantee the delivery of packet. It can be used in situation where speed is more important than accuracy.

IV. Study **Chapter 13 Java Socket Programming** from this book **Object-Oriented Programming with Java: Essentials and Applications** Authors Buyya, Selvi and Chu [2009] Tata McGraw Hill [PDF posted in Lecture 1 Folder]

13.22 In your own words, explain what the Tannenbaum text book's Layer cake cut diagram is about (pg. 78 Figure 2.16: Client-server organizations in a two-tiered architecture), providing 2 examples each for the 5 architectural models shown (e. g. a Web app, A client-server app, YouTube etc.) [10 points]

Two -tiered architecture involves client machines with user-interface programs and server machines handling processing and data. User interface can be a terminal dependent part on the cient or the entire user interface software. Parts of the application may move to the client side for tasks such as validation or editing. Most of the applications run on client machine, with server support for file or database operations. Client might store part of the data locally, like web browser visited pages.

Examples: a) thin client: banking system, cloud gaming platforms

b) entire UI software on client: Outlook, remote control desktop application.
c) part of the application on client: online MS form (validating on client side before submit), Word with local editing and other features are server-based
d)most application running on client: finance software, project management tools
d)local disk includes some data: web browser, streaming like Netflix

13.23 What is a port? List some well-known ports and explain the applications associated with them. [5 points]

A port is represented by a positive 16-bit integer value and is used by TCP and UDP protocols to map incoming data to a particular process running on a computer.
ftp 21/tcp
telnet 23/tcp
smtp 25/tcp
login 513/tcp
http 80/tcp,udp
https 443/tcp,udp
User-level process/services generally use port number value >= 1024.

V. **Study Dist. Sys Design Goals.pdf. Consider the architecture of Twitter. What do the Goals and Transparencies described in this paper mean in the context of Twitter? Why are they important? Explain with a diagram.** [10 points]

Transparency: providing a seamless experience where users interact with the service as if it were a single, unified platform, regardless of the underlying complex distributed system.
- Access Transparency: Users can tweet, retweet, follow, and message without knowing where these services are hosted
-Location Transparency: The location of servers managing tweets is not known to users
-Replication Transparency: Users are unaware of the multiple copies of tweets that improve load balancing and fault tolerance
-Failure Transparency: Users do not notice when individual components fail because the system is designed to handle such failures

Scalability: Twitter handle a massive number of users and tweets without suffering a noticeable loss of performance. This involves scaling up resources as user demand increases.

Dependability: Twitter takes explicit measures to increase reliability, it requires consistency, security and fault tolerance.

Performance: Twitter delivers tweets and updates in real-time

Flexibility: Twitter's system is extensible and adaptable to new features and changes in user behavior.

From Chapter 4 Coulouris Book

VI. Answer the following questions using explanation and diagrams as needed. No implementation needed.

4.2 A server creates a port that it uses to receive requests from clients. Discuss the design issues concerning the relationship between the name of this port and the names used by clients. page 148 [5 points]

The operating system can dynamically assign a port number for client side sockets. A port can only be used by one service at a time. Clients need to know the correct port number to connect - to a service. Well-known ports might be targeted for attacks.

4.15 [10 points]

Outline the design of a scheme that uses message retransmissions with IP multicast to overcome the problem of dropped messages. Your scheme should take the following points into account:

i) There may be multiple senders.

ii) Generally only a small proportion of messages are dropped.

iii) Recipients may not necessarily send a message within any particular time limit. Assume that messages that are not dropped arrive in sender order. page 173

Use a sequence number to make sure each message sent by senders is uniquely identified, which allows recipients and senders to keep track of which messages were sent successfully received. Sender maintains a retransmission timer for each message sent. If timer expires before acknowledgment is received, the sender can assume message was dropped. Also in the case that only a small portion of messages are dropped, the scheme can use selective acknowledgement meaning that recipients send acknowledgements only for the last consecutive messages. If retransmitted messages arrive out of order, the recipient needs to buffer out of order message until it receives all earlier messages.

VII. **Java Socket Programming implementation** [25 points]

The goal of this assignment is to implement a TCP client and server. You can use Java. Your TCP or UDP client/server will communicate over the network and exchange data.

The server will start in passive mode listening for a transmission from the client. The client will then start and contact the server (on a given IP address and port number). The client will pass the server a string (eg: "network") up to 80 characters in length.

On receiving a string from a client, the server should: 1) reverse all the characters, and 2) reverse the capitalization of the strings ("network" would now become "KROWTEN").
The server should then send the string back to the client. The client will display the received string and exit.

Example
Starting the server:
Assume that you started a server on machine 128.111.49.44, listening to port number 32000. The syntax should look like the following:
csil-machine1> server 32000 <enter>
(in this line, “server” will be replaced by one of the names given below in the Submission Section)

Starting the client:
csil-machine2> client 128.111.49.44 32000 <enter>

(in this line, “client” will be replaced by one of the names given below)

Enter text: This is my text to be changed by the SERVER <enter>

Response from server: reveres EHT YB DEGNAHC EB OT TXET YM SI SIHt
csil-machine2>

At this point (after receiving one line to be reversed), the server and client should both exit.

[Credits: Prof. K. C. Almeroth UCSB]