Northeastern University

CS 6650 Scalable Distributed Systems

Homework Set #1 [100 points]

Name: Nisarg Patel
Email: patel.nisargs@northeastern.edu

GOAL: An introductory understanding of Dist Sys and Java client server programming.

INSTRUCTIONS: ***Please provide clear explanations I your own sentences, directly answering the question, demonstrating your understanding of the question and its solution, in depth, with sufficient detail. Submit your solutions [PDF preferred]. Include your full name. Do not email the solutions.***

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

Answer the folowing questions using explanation and diagrams as needed:

2.11
2.14

Answer:

2.11:

Since there is a simple server carrying out all the client requests, it is not possible to set a limit on the time taken to respond to a client's request as there is no limit to the number of client requests coming to the server. Using any algorithm, one cannot limit time if the rate of client requests is unknown. For example, if the requests are added in a queue, then the response time is dependent on the number of requests already in the queue, and could not be limited. Even with the use of threads, the server would not be able to handle a large number of concurrent requests in the given time limit.

If the server really needs to execute requests within a bounded time, then server also needs to limit the rate of client requests. One way to do this is to limit the number of clients itself. If more clients are required than more processors are required in the server. Since this method would limit the number of clients, it does not make the server scalable. In the other case, the cost of the server would increase drastically. Thus, both options would question practicality of such server.

2.14:

Service A:

1) Since messages may be lost, this communication service exhibits omission failure. This makes the service not satisfy the validity property.
2) Duplicated messages(delivered more than once), this is arbitrary failure. This makes the service not satisfy the integrity property.
3) Since checksum is applied only to header, the message body could become corrupted, again arbitrary failure. This makes the service not satisty the integrity property.

Since both validity and integrity is not satisfied, Service A is not a reliable communication service.

Service B:

1) Since messages may be lost, this communication service exhibits omission failure. This makes the service not satisfy the validity property.
2) Since messages can be delivered too fast and recipient cannot handle it, leading to dropped messages, it is again a (receiver) omission failure. This makes the service not satisfy the validity property.

Service B satisfies integrity but not the validity property. Thus, Service B is not a reliable communication service. But since it exhibits only omission failures, it is possible to build a service that masks these failures and thus making that service reliable.

Since both the services are asynchronous, they do not exhibit time failures.

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 ilustarte this (e. g. MacOs on PC1; Linux on PC 2 etc.) and then explain how middleware helps here.

Answer:
Middleware is a software layer that help manage the complexity and heterogeneity inherent in distributed systems and provides a common programming abstraction across a distributed system. This helps the programmers by relieving them from tedious and error-prone programming along with masking heterogeneity of networks, hardware, operating systems and programming languages. The goal of middleware is to create system independent interfaces for distributed applications. More uses of middleware are as follows:

1) As discussed above, it shields developers of distributed systems from low-level, tedious, and error-prone platform details, like socket-level network programming.
2) It amortizes software lifecycle costs by leveraging previous development expertise and capturing implementations of key patterns in reusable frameworks.
3) It provides a consistent set of higher-level network-oriented abstractions to simplify the development of distributed systems.
4) It provides a wide array of developer-oriented services, such as logging and security.

The layers of middleware are as follows:

1) Applications
2) Domain Specific Middleware Services
3) Common Middleware Services
4) Distribution Middleware
5) Host Infrastructure Middleware
6) Operating Systems and Protocols
7) Hardware Devices

1) Given the multiplayer game having 4 player PCs and a Game Server. Suppose Player1 is playing on Windows, Player2 is playing on MACOS, Player3 is playing on Android and Player4 is playing on PS5, each of these will have different hardware devices and Operating systems. Middleware helps to connect these different operating systems with the Game server creating a seamless gameplay.
2) Host Infrastructure Middleware eliminates the peculiarities of different operating systems and help maintain networked applications via low-level OS programming APIs. For example, JVM, .NET.
3) Distribution middleware enables clients to program distributed systems much like stand-alone applications by invoking operations on target objects without hard-coding dependencies on their location, programming language, OS platform, communication protocols and interconnects, and hardware. Example, CORBA and RMI. This would make the multiplayer game playable from any type of inputs, like Player1 using Keyboard and player 4 using a controller.
4) Common middleware services define higher-level domain-independent services that allow developers to concentrate on programming business logic. Example: CORBA, EJB. Regarding multiplayer game, it would be the login/signup to save progress of the game.
5) Domain-specific middleware services are tailored to the requirements of domains. For the multiplayer game domains, it would have services related to the gameplay and concurrency.
6) Same Application can be written in different languages. For the Multiplayer Game example, PS5 would have a different executable whereas Windows would have another. Building middleware would make the multiplayer game language and program independent.

III. From Chapter 3 Coulouris Book **Networking** [20 points]

Answer the folowing questions using explanation and diagrams as needed:

3.1

3.7 part (ii) ONLY (*i.e.,* FTP)

Answer:

3.1:

i. UDP

- Number of send packets = 1
- Number of receive packets = (5000/1000) = 5
- Latency for 1 send/recieve packet = 5ms
- Transfer time for send packet = (200 x 8) / (10 x 10^6) s
  = 0.00016 s = 0.16 ms
- Transfer time for 1 receive packet = (1000 x 8) / (10 x 10^6) s
  = 0.0008 s = 0.8 ms
- Server request processing time = 2 ms

Thus approximate total time for UDP = 1x(5+0.16) + 2 + 5x(5+0.8) ms
= 5.16 + 2 + 29 ms
= 36.16 ms

ii. TCP

– Connection setup time = 5 ms

Otherwise the time to send and receive the packets would be approximately the same as that taken by UDP.
Thus approximate total time for TCP = 5 + 36.16 = **41.16 ms**

iii. Same machine
Since the server process is in the same machine, the data transfer rate would be much faster than the earlier one. Also single packet can be used in both send and receive. Le the data transfer rate be 100 Mbps, then:

- Transfer time for send packet = (200 x 8) / (100 x 10^6) s
= 0.000016 s = 0.016 ms
- Transfer time for receive packet = (5000 x 8) / (100 x 10^6) s
= 0.0004 s = 0.4 ms

Thus approximate total time for same machine = 5 + 0.016 + 2 + 5 + 0.4 ms
= 12.46 ms

3.7 (ii) FTP:
File transfer would require a large amount of data to be transmitted. For this, connectionless communication would be the better option. But if the error rates are high in UDP, the connection-oriented communication would be better. FTP therefore sends data using TCP/IP connection.

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.) [15 points]

Answer:
The diagram represents the client-server organizations in a simple two-tiered architecture: a client machine and a server machine. A distributed application can be divided into 3 layers: 1) User Interface Layer, 2) Processing Layer, 3) Data Layer. These layers can be distributed in five different ways among the two machines, by cutting through different layers and distribute the top part for client to handle and bottom part for server to handle. These 5 possible organizations are as follows:

1) Terminal dependent part of User Interface on client. Other UI and layers on Server. Web applications like Youtube and Facebook can be thought of such organization.
2) Entire User Interface on Client, Processing and Data layer on the server. Google Maps, Drive app can be thought as this organization.
3) Part of application on client. Eg: Google Docs, Appointment booking client-server app which requires form filling, basic info can be checked on client side.
4) Most application on client side, all operations on database on server side. Multiplayer games like Clash of Clans or Banking apps like Chase are examples of it.
5) Data present also on client side. Mobile applications like WhatsApp and Gmail can be an example of such organization which also stores the data in the local storage.

Two examples of architectural models:
1) Client: Web Browser, ATM Machine
2) Server: Youtube server, Bank Server
3) User Interface: web page of www.youtube.com, the display of atm screen
4) Application: The click and playing of videos, the transaction process in ATM
5) Database: Youtube's and Bank's database. Some examples of database include MySQL, SQLite, MONGODB, etc.

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

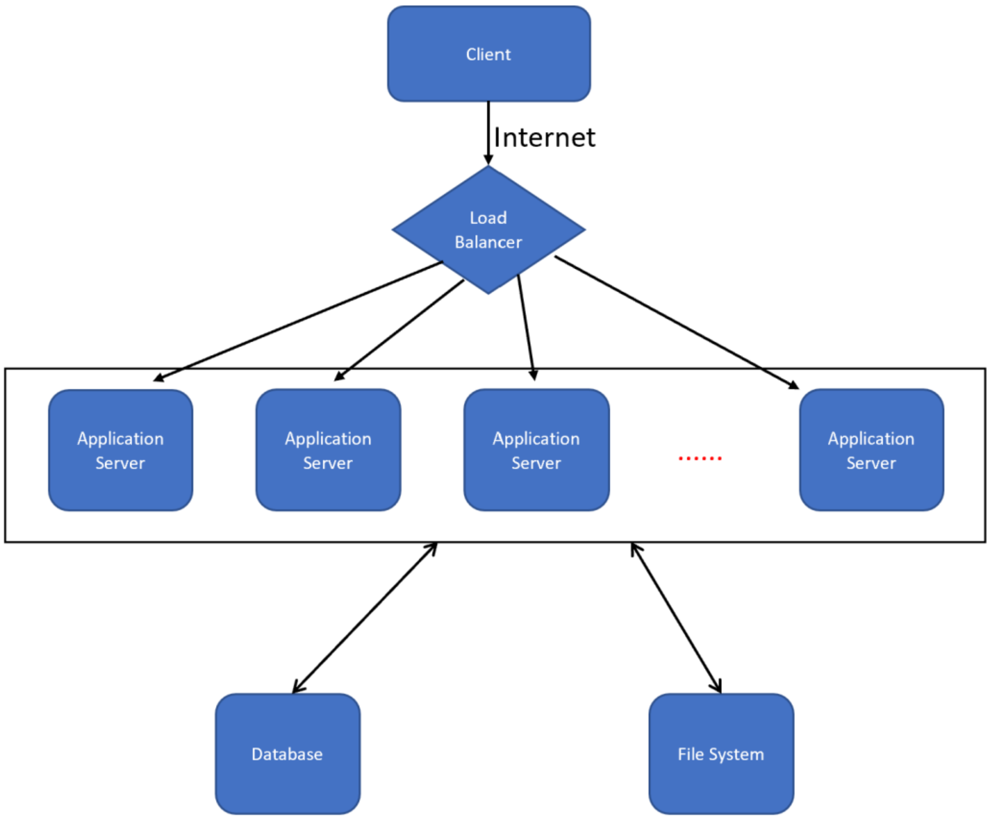
Answer:
A port is a numeric identifier associated to a network protocol that receives or transmits data for a specific service. Only a single service can listen or send data to a particular port number at a time.
Some well known ports are:
80 – For HTTP – Establishes a request/response connection on Internet
443 – For HTTPS – User authentication and encryption connection on Internet
20/21 – For FTP – For File Transfer and manipulation
22 – For SSH – Securely connect to a remote network
110 – For POP3 – Post Office Protocol – Supports email synchronization
143 – For IMAP – Internet Messaging Access Protocol – Supports email synchronization
25 – For SMTP – Simple Mail Transfer Protocol – Sends email in TCP/IP network

---

V. Study Dist Sys Design Goals.pdf. Consider the architecture of Twitter. What do the Goals and Transparencies desicribed in this paper mean in the context of Twitter? Why are they important? Explain with a diagram. [15 points]

Answer:
A twitter system needs to efficiently store and access all the tweets. From the Designing Twitter pdf, we see that there are approximately 100M/86400s = 1150 new tweets every second. Thus the system has to be read heavy. This would require multiple application servers to serve these requests with load balancer for traffic distribution to prevent overloading of a server. An efficient database is required in backend that can store all the new tweets and file storage to store photos and videos. Both, the database and the file storage needs to support a large number of requests.

The following is the diagram of simple design of twitter system:



The Goals and Transparencies described in the pdf along with their importance are as follows:

1) Access Transparency: Twitter stores some tweets or images in cache for faster access. An older tweet can be accessed from this cache from local storage and new tweets from remote DB. User is unaware of from where the tweet is accessed.
2) Location Transparency: User is unaware where the tweets are stored in remote DB. It can be anywhere and on any server.
3) Migration Transparency: Multiple servers can move the resources without the knowledge of user. For the user, it is the same tweets that he would be shown if resources are not moved.
4) Replication Transparency: Multiple copies of the tweets, photos and videos is present in different servers for faster access of the data. User is unaware of this.
5) Failure Transparency: If a single server or component fails, user would not know it as the user would be able to see the same tweets coming from a different server that also has a copy of that resource. This would lead to increase in load but user experience will not be affected.
6) Concurrency Transparency: Multiple users can use twitter together, many of them accessing same database. The user is still able to perform actions normally.
7) Scalability – Based on the Designing Twitter pdf, twitter has around 1150 new tweets per second and 325K read requests, the system has to be scalable. The application servers allow horizontal scalability. A large number of resources means Geography growth is also satisfied. The distribution of data also improves administrative scalability.
8) Dependability – Consistency means in twitter context that user can see the tweets and photos/videos properly. No corrupted data is being displayed. Security means that private

information of users should not be displayed to any user without their consent. Fault tolerance means that even if a server fails, user should be able to see the tweets.

9) Performance – Performance in the context of twitter would mean that user is able to see or refresh tweets without much lag.
10) Flexibility – Twitter system is flexible, since with new updates, many new features and rules are available for users and users having latest versions can use twitter in similar way, which means the system is extensible, open and interoperable.

VI. From Chapter 1 Coulouris Book

1.12 A server process maintains a shared information object such as the BLOB object of Exercise 1.7. Give arguments for and against allowing the client requests to be executed concurrently by the server. In the case that they are executed concurrently, give an example of possible 'interference' that can occur between the operations of different clients. Suggest how such interference may be prevented. Pg. 22 [15 points]

Answer:
If client requests are executed concurrently by the server, the throughput is not limited which it would be if the shared resource would take one client request at a time. But with concurrent executions, several threads would be accessing the shared object at a time, and any operations on the object can lead to conflicts (possible interference) and produce inconsistent results.

Taking an example of bank transactions, the shared object being the bank account. Suppose two tries to deposit in same bank account at the same time. Client C1 depositing $100 and client C2 depositing $200. Let the bank account originally contain $1000.
One possible interference can happen because of the following sequence of operations:

1) C1 thread reads the shared object ($1000).
2) C2 thread reads the shared object ($1000).
3) C1 adds $100 to its read value and updates the bank account ($1100).
4) C2 adds $200 to its read value and updates the bank account ($1200).

Thus, the bank account shows $1200 instead of the correct value $1300.

Such interference can be prevented by synchronization of the operations in a way that makes data consistent. One such technique is to use semaphores.

1.13 A service is implemented by several servers. Explain why resources might be transferred between them. Would it be satisfactory for clients to multicast all requests to the group of servers as a way of achieving mobility transparency for clients? Pg. 23 [10 points]

Answer:
Resources might be transferred between servers in a system. This is to allow the load balancing among different servers. Resources can also be transferred to utilize storage. It also helps to make the system scalable.

If every client multicast the requests to many servers, the load on the network increases a lot and each server must process that request even if server can do nothing for that request. It is same as post office making copies of the same mail and sending to every other post office. And every post office must process the letter once even if it cannot deliver to the address. Thus, it is not satisfactory to multicast the requests to group of servers to achieve mobility transparency for clients.

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: revres 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]

Answer:
TCPServer.java
TCPClient.java