CS 6650 Scalable Dist Systems

Homework Set #1

Late days left : 5

I . Study Chapter 2 Systems Models Coulouris Book

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?

The time taken by the server to respond to a client request entirely depends on the type of request made by the client. It depends on the server how it decides to handle the client request hence the time taken is unpredictable. There might be reasons as to why the server takes time to process the requests :

- The server might be using threads to handle the requests.
- The server might queue the requests and execute them one at a time.
- The server execution time also depends on the complexity of the client request.
- The size of the data that is getting processed might be taking time.

To make the server execute requests in a bounded time

- The client requests can be limited according to the capacity of the server.
- A server with a larger number of processors can be used to accommodate all client requests.
- The server can execute the request in the order received and implement only specific requests in a given time duration.
- The server can prioritize the requests which require a shorter time to execute.

This might lead to high costs and setting time limits on the server can lead to poor performance of the server which might affect the client requests

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?

Service A
Messages are Lost - Failure of Validity : communication omission failure
Messages are Duplicated - Failure of Integrity : arbitrary failure
Messages are Delayed - Since we are considering an asynchronous system delay of messages is not a failure as long as the message gets delivered.
Checksums are applied only on Headers - As checksums are only applied on headers the messages might be corrupted resulting in failure of integrity : arbitrary failure

Service B
Messages are Lost - Failure of Validity : communication omission failure
Messages are Delayed - Since we are considering an asynchronous system delay of messages is not a failure as long as the message gets delivered.(For a synchronous system it will be a failure of timeliness)

Messages are Delivered too Fast - Leading to user not able to process the message : communication omission failure
Delivered messages arrive with correct contents - Achieves integrity

However the system fails the property of validity leading to Service B not being reliable as an asynchronous distributed system.

II.Please refer to the 2 articles (PDF) on Middleware for Distributed Systems. 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 software layer which is present in between the Operating System and the Application.
It is also called Plumbing as it connects two applications together so that data can be easily passed between the pipes. Middleware is used to build applications more efficiently and it allows flow of communication or data. Technology is all about transmission of data, information and services and Middleware works with all these resources. Using middleware helps weather applications to get weather information, helps users submit forms on a web browser etc.

Some uses of Middleware :

- It provides a simple interface for applications to communicate and provides abstraction to hide the underlying network infrastructure and helps the application to focus on its own performance.
- Middleware implements logic based on the client requirement for instance it can detect if the user requests in a particular language say English, it makes sure that the backend provides results in English language.
- It plays an essential role in securing the backend resources and provides encryption, authentication and authorization
- It provides scalability and is essential in providing concurrency, load balancing and transaction management. It also distributes the client request among multiple servers or virtual machines.
- Middleware allows applications to communicate with each other without knowing the physical location of one another which allows greater flexibility in terms of scaling the system.

Heterogeneous elements refers to a system that is made up of different types of hardware or different types of software. For instance different computers might have different processors, different operating systems, different application versions, different models etc.

The multiplayer game mentioned has different heterogeneous elements. The 4 PC's and the game server all together make five systems. These systems have three unique layers and can different components on every layer

- **The Hardware**
  The PC1 might have a different processor than PC2, PC3 or PC4. PC1 might be having an Intel Core processor whereas PC2, PC3 might be having AMD Ryzen 7 and PC4 might be having AMD Ryzen 5. This will result in a different number of cores, clock speed, or instruction set.
  Similarly, all the systems might have different models/brands of the laptop. PC1 might be having a Dell Laptop, PC2 might be having an HP laptop, PC3 and PC4 might have an Acer Laptop.
  The game server is going to have stronger hardware configurations than all the gamers PC's. It might have more RAM, faster processors, more storage etc.
  All the systems might be having a different graphics card like PC1 might be having AMD Radeon whereas others might be having NVIDIA GeForce.

- **The Operating System**
  All PC's might be using different OS. PC1 might be using MacOS whereas PC2,PC3, PC4 might be using Windows. The Game server might be running on Linux. Within different OS they might also be using different versions of OS. Like PC2 might be using Windows, PC3 might be using Windows 10 and PC4 might be using Windows 11.

- **The Game Application**
  The PC's might be using different versions of the same game. Some gamers might have not updated to the latest version while some might have updated. The players might have different settings for their applications.

After all these various reasons and different heterogeneous components at different layers. All the five systems are still playing the game without any issues. Here Middleware comes into picture. It provides an abstraction layer due to which the gaming experience of the players is not affected. It helps by communicating with the other PC's without the need to know about the system configurations of the other players. It also provides security and scalability which enhances the user experience.

III.From Chapter 3 Coulouris BookNetworking. Answer the following questions using explanation and diagrams as needed:
3.1 A 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
Connection setup time (TCP only): 5 ms
Data transfer rate: 10 Mbps
MTU: 1000 bytes
Server request processing time: 2 ms
Assume that the network is lightly loaded.]

i) Connectionless - UDP
5 + 1600/10000 + 2 + 5(5 + 10000/10000) = **36.16 ms**

ii) Connection-oriented - TCP
5 + 5 + 1600/10000 + 2 + 5(5 + 10000/10000) = **41.16 ms**

iii) Server process is in the same machine
5 + 1600/40000 + 5 + 50000/40000 = **11.29 ms**

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

File Transfer protocol is used for transferring files between the client and server. The client can request or send files to the server. This is achieved using either User Datagram Protocol (UDP) or using Transfer Control Protocol (TCP).

UDP is a connectionless protocol which means that there is no connection between the sender and the receiver for file transfer. The process is faster but it not reliable the sender or the receiver cannot make sure if the messages were delivered across each other or if the messages have gone missing.

TCP is a connection oriented protocol and a connection is set up between the sender and the receiver before transferring any files or data. It is comparatively slower compared to UDP but the sender and receiver can make sure that the packets have been delivered or if they are missing.

UDP could be used if there are lesser errors and files can be largerand the process will be less time consuming, But on the Internet this is not possible. This makes TCP more reliable for file transfer.

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.)

Simple Client - Server Architecture
The diagram referred to explains the layered architecture of a distributed system. A client sends a request to the server the server receives the request and processes it meanwhile the client waits for the response. This can be achieved by the means of a simple connectionless protocol when the network is reliable as it is in many local-area networks. It does provide efficiency but problem arises when messages are lost, corrupted or if there is a failure. We can let the client resend the message but this will not be a possible solution as there might be cases when the request is processed twice.

Two-Tiered Architecture

The organizations was as such that only two machines were present: a client machine and a server machine. According to this architecture everything is handled by the server and the client machine is only a graphical interface. The architecture can also be divided into three layers and these layers can be distributed on among different machines :

- User Interface layer
- Processing layer
- Data layer

The figure focuses on only two machines, the server and the client and this is called the two-tiered architecture.

Figure 2.16(a) - In this organization the terminal-dependent part of the UI is present on the client machine while the application runs on remote servers and also control the presentation of their data. The client machine displays the UI and also handles the user input while the application is stored and processed on remote servers.
Examples - Social media applications like Twitter, Facebook use this architecture
Remote Desktop applications like Virtual Network Computing (VNC) uses this architecture.

Figure 2.16(b) - In this diagram the entire user interface software is placed on the client side. In this architecture the client software does no processing and only handles what is necessary for the application's interface. The entire communication is carried out via an application specific protocol.
Examples - Mobile applications like e-commerce apps : Amazon, InstaCart etc.
Rich Internet Applications like Silverlight.

Figure 2.16(c) - In this figure a part of the application is moved to the front- end. This is useful when some parts have to be executed on the client side of the machine for performance or for user experience reasons.
Examples - Online forms - The front end checks for validation and then it is sent to the backend to store in the database.
Word Processor - The front end handles the formatting, input and editing whereas the backend handles

the spelling check and grammatical mistakes(Grammarly Application)

Figure 2.16(d) - This is a popular architecture. The client server her handles most of the application and all operations on files and databases are handled on the server side.
Examples - Banking applications where the users enters the amount to be withdrawn and the application contacts the server where the database is looked up to process the transaction.
Point Of Scale Systems - Here the client machine handles the input like the customer's groceries and the server handles the inventory, items in stock etc.

Figure 2.16(e) - This is a popular architecture too. In this architecture the local disk of the client machine contains some data which might have to be used frequently by the client and fetching data from the server everytime won't be a feasible option
Examples - Web Browser - A huge cache is built on the local disk where the client can easily go and fetch the recently visited web pages. These pages load faster and quickly and do not take a lot of time.
Mobile Applications also store a cache of the data on local disk and it can be easily accessed even if the user is not online.

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

A port is an array of communication. It is a virtual point where the network connections on a computer begin and end. It is a numerical label that is assigned to a service that runs on a computer in a network. Ports help computers to identify different services being offered. For example web services, gaming services email services all use different ports. A port is a 16-bit unsigned number ranging from 1 to 65535**.** Ports numbered from 0 - 1023 are known as well known ports. Ports from 1024 to 49151 are registered ports and private ports range from 49152 to 65535.

Some of the well known ports are :

Port Number	Usage	Application
20	File Transfer Protocol (FTP) Data Transfer	Used for transferring data and files in FTP. It is also used by Trivial File Transfer Protocol(TFTP) , which is a simpler version of FTP. It is also used for peer-to-peer file sharing programs.
21	File Transfer Protocol (FTP) Command Control	It is used for control connection of FTP. Prior to establishing a connection on FTP a control connection on port 21 has to be established. Which is used for sending and receiving between the client and the server. The actual file transfer happens on port 20 while port 21 mangoes the session.

22	Secure Shell (SSH)	This port provides secure encrypted communications and is used for SSH protocol. It manages the devices remotely. A secure connection is set up on port 21 and then the client and server can exchange data securely.
23	Telnet - Remote login service, unencrypted text messages	Telnet is an insecure protocol in comparison to SSH protocol. The communication between the client and server is not secure. It is also used for the remote access of devices.
25	Simple Mail Transfer Protocol (SMTP) Email Routing	This port is used for sending and receiving email messages between the server and the client. Once to hit send on that email a port the SMTP server is set on port 25 and the mail is then sent to the server from where it reaches the recipient's mail id.
53	Domain Name System (DNS) service	Every web address has a domain name. The DNS is basically responsible for converting the domain name to the IP address. Whenever you enter a website name on the search bar a request is made to DNS on port number 53 where the name is searched up for and converted into IP address.
80	Hypertext Transfer Protocol (HTTP) used in World Wide Web	This is one that is commonly heard of. This is used for transferring data between the server and web client on port 80. The client can send an HTTP request like a GET, POST or DELETE request which will be fulfilled by the server.
110	Post Office Protocol (POP3) used by email clients to retrieve email from a server	This port is basically used to retrieve email messages from the email server. Once the connection is set up on port 110. The user can then enter the username, password for authentication and then retrieve the email messages from the server.

119	Network News Transfer Protocol (NNTP)	This port is used mainly for news articles and news information. When the client wants to access the usenet news articles, once the connection to the NNTP server is set up on port 119 the user can send commands to get details of his /her choice or post news details. NNTP is an older protocol and it is replace by IMAP and SNMP.
123	Network Time Protocol (NTP)	This is used for time accuracy of all the devices. The client can set up a connection to NTP on port 123 to get the exact details of the time so that all the devices are set up on the current time on the server.

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.

Considering the Twitter architecture which is also a distributed system. Twitter is a social media platform where the users tweet and post messages. These tweets are stored in a database in the backend and are delivered to servers all around the world. The architecture of Twitter is as such that it can handle multiple users and tweets. The system is composed of multiple layers such as web servers, application servers, data storage systems, message queues, load balancers etc. All these layers work together to provide services to users, while hiding the underlying network components.
The basic goals are properties that a distributed system is supposed to have. Basic goals mentioned in the paper are :

- Transparency
- Scalability
- Dependability
- Performance
- Flexibility

Explaining these basic goals in terms of Twitter application

Transparency
Transparency is the ability of the system to hide the underlying components of the system for the user and the application due to which the system occurs as a single unit even though
It has various components. As for Twitter we are able to view, use twitter as a single application without
Worrying about what goes in the backend. There are different forms of transparency :

Access Transparency
The user can access the tweets, the posts, the messages on Twitter in the same manner whether they are logged in from their mobile devices, computer devices or tablets. Users have the same experience and access on different devices.

Location Transparency
We as users are unaware about the location of the servers and where all the tweets
and data is stored in the database. There are load balancers which distribute the load on the servers and maintain availability.

Migration Transparency
All the resources like servers, databases etc. can be migrated to different place and there is no
need to change the address of these resources. This allows the system to adapt to changing
Conditions and to handle failures and this does not affect the user experience.

Replication Transparency
There are numerous copies of a users tweet and data this replication transparency is to handle failures. These copies are not visible to the users they are only able to see their tweets and tweets posted by other people.

Failure Transparency
There can be failures in the system and the users are unaware about these. This is because the system Ensures redundancy and replication to handle failures. There are times when the user might face some delay or issues due to failures.

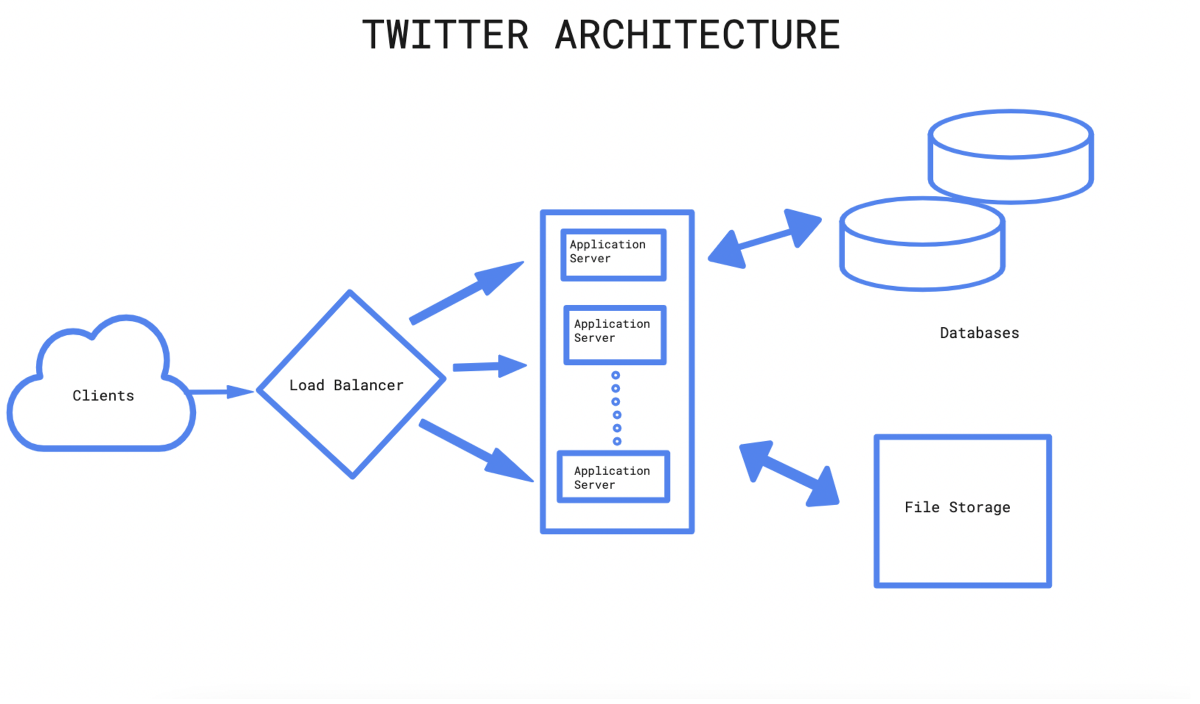
Concurrency Transparency
The systems use load balancers to ensure that the resources are being shared efficiently and
no matter how many users exist the performance of the system is not affected in any way.

Scalability
Scalability allows the system to handle growing numbers of users and resources and does
not affect the functionality of the system. There is no loss of performance of the system or increased Complexity. Whenever there are more number of users or tweets added the system scales itself up by adding more servers, databases and resources required. Adding of resources does not affect the user Experience

Dependability
Dependability focuses on providing services even if there is a failure. It ensures
high availability, fault tolerance, and disaster recovery. Twitter ensures that users don't face issues even when there is a failure. The techniques used to get dependability are redundancy, replication
and data backup. Replication builds multiple copies of the tweets and other messages. Redundancy make sure that the system uses load balancers whenever failure occurs it can easily shift to another Resource without affecting the user experience. The system always performs data backup so whenever a disaster occurs, all the data can be retrieved easily. It also keeps monitoring for system Failures and performs recovery actions.

Performance
Performance refers to the system providing fast and reliable services to the users. The architecture of Twitter is as such it provides high performance , efficient use of resources and minimises the user requests response time. The system uses load balancing, caching, asynchronous processing and Optimizes data storage to achieve high performance it also monitoring and tuning tools to measure the performance of the system.

Flexibility
Flexibility refers to the system’s ability to to adapt to the changes in requirement and be easily modified without affecting functionality. Twitter is designed to handle new features, multiple users, higher number of tweets and changing requirements. Flexibility is achieved using Microservice architecture, conterization , modularity etc.



These goals and transparencies are important to ensure that the system provides an efficient service to its users. The Twitter users do not face any difficulties while using the application. They also help to ensure that the system can adapt to changing requirements and handle larger number of users and tweets.

VI. Answer the following questions using explanations 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

The design issues related to server ports are as follows :

- **Naming and identification of port by client**

  The servers should assign a unique port number to each service it provides so that the that clients can correctly identify the correct port to use for a given service. The ports should be location independent so that they can offer the same service at different time zones. The client can use a name binder to map the textual name of the service to the port.

- **Port registration and security**

  The ports should be registered with the central registry which makes it easier for the client to search and connect to the port. The server should take security measures to prevent unauthorized access.

- **Port scalability**

  The ports should be able to handle a larger number of client requests. It should allow concurrent connections and scale the port in response to larger connections

- **Access to Local identifiers**

  When the operating systems allows the use of local names to refer to ports it becomes an issue when a server creates a non-public port for a client to send messages as the local name is meaningless to the client.

4.15 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.

i)There may be multiple senders.
The sender should attach a unique sequence number for each message sent and the receiver should check the sequence number of the message received and identify accordingly.

ii) Generally only a small proportion of messages are dropped.
The receiver should keep track of the messages received and should identify the missing messages and request the sender to resend the missing message. The sender in turn should store all the messages sent so the sender can resend the missing message. The sender sends/retransmits the message as a unicast datagram.

iii) Recipients may not necessarily send a message within any particular time limit.
This point refers to the fact that we cannot entirely depend on a n acknowledgement of sending and receiving messages as messages are not going to be sent in any particular time limit. If there are no acknowledgements the sender will be holding all the sent messages. So to overcome this the solutions are :
1) The sender discards the stored messages after a given time limit.
2) Occasional acknowledgements from the receiver can be piggy backed on messages that are already sent.