COMP90015 Exam

2021 Semester 1 Demo

Q1 What is DFS, explain challenges involved in designing DFS?

- Distributed File Systems 分布式文件系统
- Challenges 这种没有标准答案（ppt 原话的我尽量只写思路，防止老师查同，5 分的话大概挑 2 条说就好）
  - 当有大(巨)量用户同时发送请求时，我们的服务器要 hold 得住。假如有一亿个用户同时登陆你这个系统，能不能保证不卡？（虽然国外一般没这么大负载，不过面向世界的公司应该有，比如谷歌）（还有就是 DDoS 了）（然后卡不卡是会跟本地文件系统操作对比的）
  - 当特定时期负载量很大，平时则没有这么多需求的时候，如何保证我们的资源（比如服务器）之类的开销不浪费。特殊的时期可能是校招期间，或者考试期间，需要用到 DFS 的人群就多之类的。其实最好理解的情况是双十一。
  - 怎么保证多人同时编辑一个文件不出问题。比如说，如果我们三个人同时在改一个文档的同一行，甚至同一个字，服务器怎么处理。
  - DFS 为了保证系统的可用性，对文件是有做备份实现的，但是如果我们有很多僵尸用户，大量长时间不用的文件和备份会对我们的资源造成占用，但如何识别僵尸用户又是个问题，万一人家就是 2 年改一次文件，或者有的文件就是要长期“供养”的呢？
  - 要支持用户用各种各样的 os 访问，苹果和 windows 还好说，linux 的版本就比较多了，还有用各种手机 os 的，光是了解尽可能多的 os 特性，确保正常使用就是个需要消耗大量精力的活，更何况有的犄角旮旯里出的 os 听都没听说过，但是用户群可能还不小
  - Difficult to achieve the consistency for distributed file systems while

maintaining good performance and scalability 要想保持高性能，一般都是一段时间才刷新一下，但是如果要保持文件的一致性，就要高强度刷新，这又会占用大量资源，让响应变得不那么流畅

- 老生常谈的安全问题
  - 比如需要一个安全可靠的权限系统，如果不是像谷歌，腾讯这种本身业务就很多，账号系统通用的公司，单纯做一个自家的账号系统还是挺吃资源的，如果允许第三方登录，要考虑的就更多了，更何况还有小 b 崽子不小心外泄密码
  - 网络通讯过程加密，如何保证不会被破解

Q2 What is Thread? Compare differences between thread and process. 线程和进程

- A *Thread* is a piece of code that runs in concurrent with other threads. Each thread is a statically ordered sequence of instructions. Threads are used to express concurrency on both single and multiprocessors machines. (From Lecture 3 没有页码的一页，在 14 页后面 & tutorial 3 Q1)
- Threads are light-weight processes within a process. Threads are used to perform parallelism and concurrent execution of independent tasks/operations. (From Lecture 3 p14)
- Thread v.s. Process (from Tutorial 3 Q1 基本说的都是对比进程，线程的优点)
  - Threads share the same address space
  - Context-switching between threads is normally inexpensive
  - Communication between threads is normally inexpensive
  - 进程是执行中的一段程序，而一个进程中执行中的每个任务即为一个线程
  - 一个线程只可以属于一个进程，但一个进程能包含多个线程

Q3 Describe RMI with its components? 远程函数调用

- Remote Method Invocation (RMI) is an extension of the object-oriented programming model. (L6 p2) Each process conatins a collection of objects, some of which can receive both remote and local invocations. Method invocations between objects in different processes are known as RMI, regardless the processes run in the same or different machines. (L6 p3)
- Java RMI is an extention of the Java model to support distributed objects. Methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. (L6 p8)
- components (L6 p9 流程图可见 p10)
  - **Remote reference module** (at client & server) is responsible for providing addressing to the proxy (stub) object
  - **Proxy** is used to implement a stub and provide transparency to the client. Its is invoked directly by the client (as if the proxy itself was the remote object), and then marshal the invocation into a request
  - **Communication module** is responsible for netwoking
  - **Dispatcher** selects the proper skeleton and forward message to it
  - **Skeleton** un-marshals the request and calls the remote object

Q4 What are Sockets?

- Sockets provide an interface for programming networks at the transport layer (Lecture 2 p13)
- A socket is an endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data destined to be sent. (Lecture 2 p16)
- Bound to a local port. Sockets are used for communication between two

networked processes. Each socket is associated with a protocol (UDP or TCP). Acts as programming interface to application code and transport layer. (Tutorial 2 Q1)

Q5 Explain why certificates are used in distributed system?

- A digital certificate is a digital form of identification, like a passport. It provides information about the identity of an entity. It is issued by a Centification Authority (CA), such that the validity of the informatoin in the ceitificate is guaranteed. The issue of distributing Public Key is massive, and the Public Key should be distributed in a scalable and truchful way, when certificate is helpful to solve it. (T9 Q4)

2020 Semester 1

Q1 Discuss the key challenges that one needs to address in the design and development of distributed systems or applications.

(1) Heterogeneity

- Heterogeneous components must be able to interoperate

(2) Distribution transparency

- Distribution should be hidden from the user as much as possible

(3) Fault tolerance

- Failure of a component (partial failure) should not result in failure of the whole system

(4) Scalability

- System should work efficiently with an increasing number of users
- System performance should increase with inclusion of additional resources

(5) Concurrency

- Shared access to resources must be possible

(6) Openness

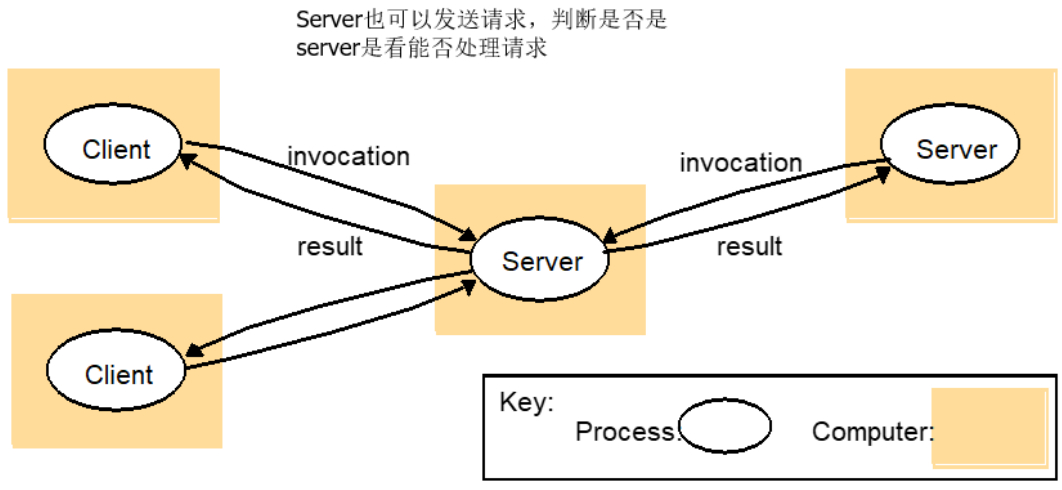
- Interfaces should be publicly available to ease inclusion of new components

(7) Security

- The system should only be used in the way intended

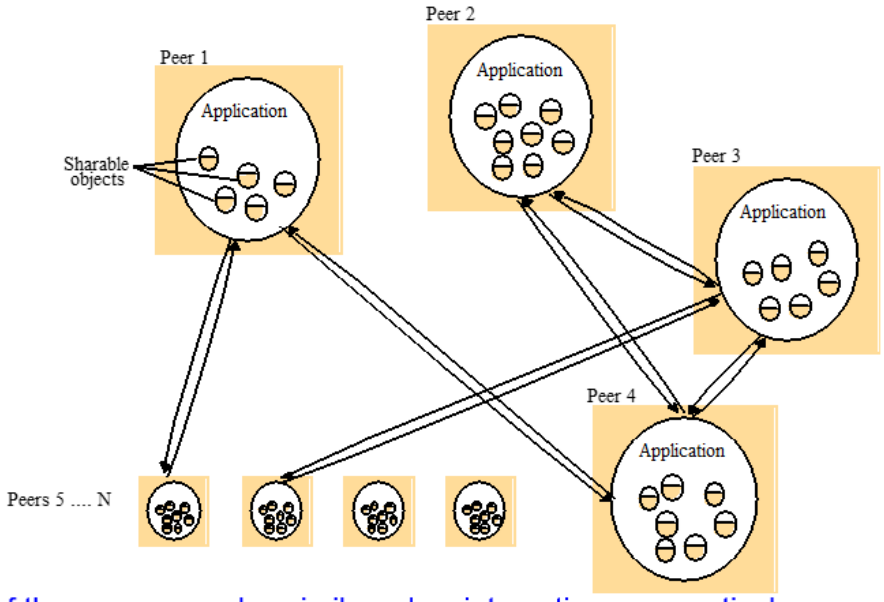
Q2 Discuss any two architectural models for construction of distributed systems.

(1) Client-server model



- Client processes interact with individual server processes in a separate computer in order to access data or resource. The server in turn may use services of other servers.
- Example:
  - A Web Server is often a client of file server.
  - Browser → search engine -> crawlers → other web servers.

(2) peer to peer model



- All of the processes play similar roles, interacting cooperatively as peers to perform distributed activities or computations without distinction between clients and servers. E.g., music sharing systems Napster, Gnutella, Kaza, BitTorrent.
- Distributed "**white board**" – users on several computers to view and interactively modify a picture between them.

Q3 Describe five types of attacks (on processes, communication channels, services) that might occur in the Internet.

- Security Threats - Three broad Classes:
  - Leakage: Acquisition of information by unauthorised recipients
  - Tampering: Unauthorised alteration of information
  - Vandalism: Interference with the proper operation of systems
- Method of Attacks are listed below:
- Eavesdropping - A form of leakage
  - obtaining private or secret information or copies of messages without authority.
- Masquerading – A form of impersonating
  - assuming the identity of another user/principal – i.e, sending or receiving messages using the identity of another principal without their authority.
- Message tampering
  - altering the content of messages in transit
    - *man in the middle attack (tampers with the secure channel mechanism)*
- Replaying
  - storing secure messages and sending them at a later date
- Denial of service - Vandalism
  - flooding a channel or other resource, denying access to others

Q4 What are the advantages of using multiple threads over multiple processes? Explain the difference between a worker pool thread model and a thread-per-object model, including details concerning what concurrency control and queueing is required in each case.

- advantange (Tutorial 3 Q1)
  - Threads share the same address space
  - Context-switching between threads is normally inexpensive
  - Communication between threads is normally inexpensive
- In worker pool thread, the server creates a fixed pool of worker threads to process requests. The module "receipt and queuing" receives requests from sockets/ports and places them on a shared request queue for retrieval by the workers. (L2 p46)
- A thread-per-object model associates thread with each object. An IO thread receives request and queues them for workers, but this time there is a per-object queue. (L2 p47)

Q5 Consider a server process that has a single TCP server socket, bound and listening on port 4444. For this scenario answer the following:

(A.) While listening for an incoming TCP connection on port 4444, can the process also receive UDP packets on port 4444? Explain your answer.

- 不行，同一个端口 port 只能同时搭建一个协议 protocol 对应的 socket

(B.) While the server is using the TCP connection on port 4444 for sending command messages to the client, can the same server process use another TCP connection on port 2000 for sending control messages to the client? Explain your answer.

- 可以，只要程序通过多个线程管理这 2 个 TCP 协议的 socket，这 2 个协议的通讯是互不干扰的，这也正是端口的意义所在

(C.) Is it possible for the server process to receive 5 concurrent TCP connections from clients on the same port? Explain your answer.

- 可以，只要我们有至少 5 个线程，一个线程接收一个客户端的链接，就可以实现题目中的情景

(D.) Is it possible for a client to connect from TCP port 4444 to the server process? Explain your answer.

- 可以，当客户端的 4444 端口未被占用的时候，可以将 socket bind 到 4444 端口发送请求

(E.) To reduce the load of a server process, can another server process be added to the same host with a TCP socket bound on port 4444? Explain your answer.

- 不可以，一个 host+port 组合只能跑一个 socket *（补充）*如果想减负，可以选择负载均衡（load balance）算法，开放多个端口，将接收到的请求匹配到不同端口上（这门课应该没讲这个概念，所以具体怎么实现咱们也可以不说，甚至可以不提负载均衡）

Q6 What is an idempotent operation? Some of the primitive operations for a typical flat file service interface for a Distributed File System are shown below (UFID stands for Unique File Identifier). Which of the following primitives of the interface are not idempotent? Explain your answer.

(A.) Read(UFID, i, n): Reads up to n items from position i in the file.

(B.) Write(UFID, i, Data): Writes the data starting at position i in the file. The file is extended if necessary.

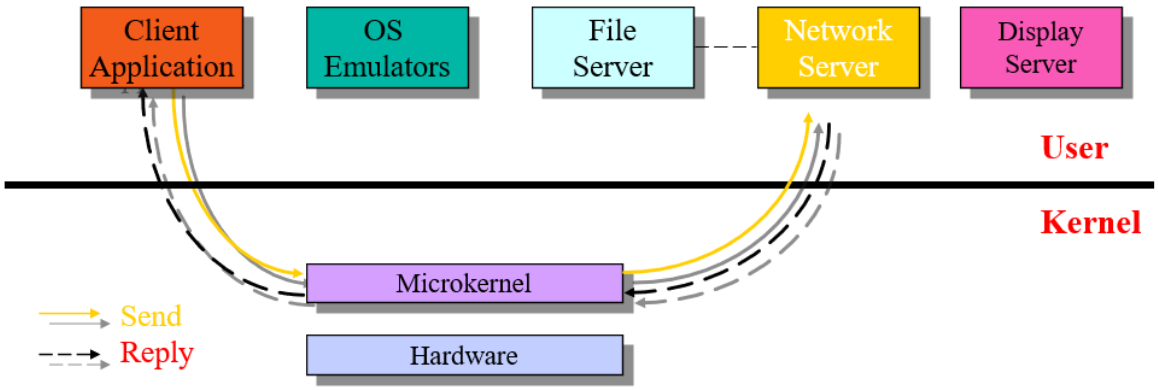
(C.) Create(): Creates a new file of length 0 and returns a UFID for it.

(D.) Delete(UFID): Removes the file from the file store/system.

- 幂等运算其实指的就是，其重复执行的效果与它仅执行一次的效果相同。例如向集合当中添加了一个元素的操作是幂等操作，因为它每次的执行对于集合都是一样的。然而，向序列中添加一个项就不是幂等操作，因为每次执行都扩展了这个序列
- 除了 Create，其他都是幂等操作，即允许使用至少一次的 RPC 语义,Client 可能在没有收

到应答的情况下重复调用。而重复执行 Create 会每次生成一个新的文件

Q7 Discuss the architecture of a microkernel-based operating system. Comment on how well this architectural model supports the creation of extensible operating systems.



- Compared to monolithic, microkernel design provides only the most basic abstractions

(address space, threads and local IPC)

- All other system services are provided by servers that are dynamically loaded precisely on those computers in the DS that require them.

- Clients access these system services using the kernel’ s message-based invocation mechanisms

Comment : (L5 p18)

- A relative small kernel is more likely to be free of bugs than one that is larger and complex
- Extensibility and its ability to enforce modularity behind memory protection boundaries

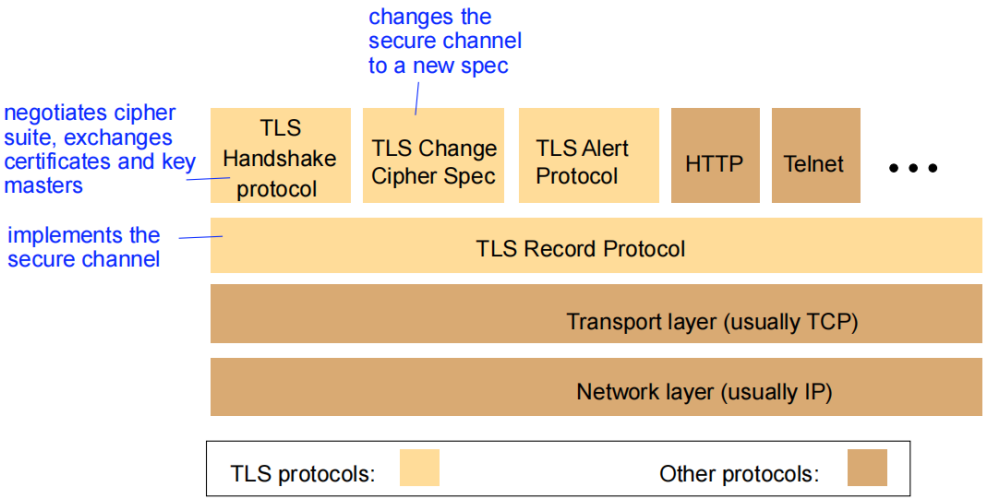
Q8 Why is symmetric encryption used for session encryption, rather than asymmetric encryption? Explain how asymmetric keys are used in digital signatures.

- symmetric 即 secret，asymmetric 即 public
- 可以用来进行加密和解密运算，确保信息传递的安全性/隐秘性，而如果是使用 public key，很容易导致加密解密算法被破解，信息被解读泄漏(leakage)
- public key 可以用于验明身份(certificates)，也可以在 digital signatures 中用于 decrypts the signature

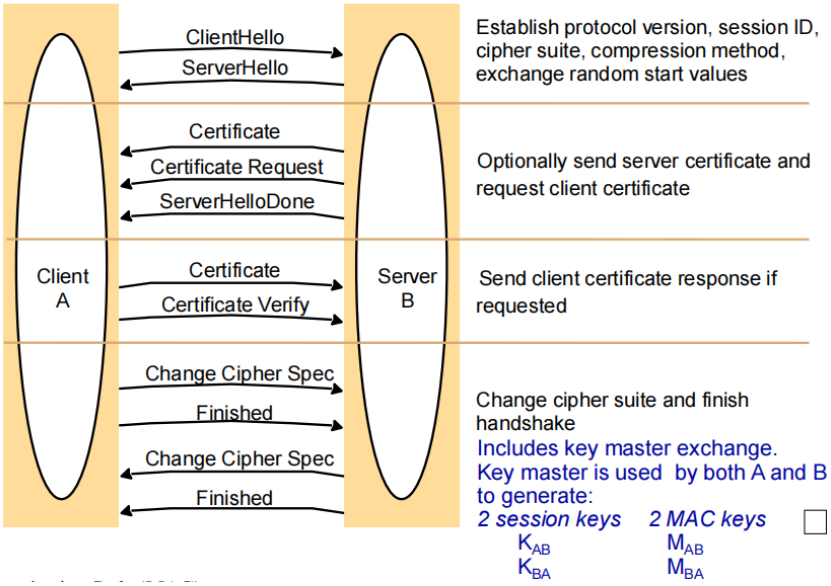
Q9 Discuss the secure socket layer (SSL) with Transport Level Security (TLS) protocol stack architecture and its components.

TLS 由两层组成。一层是 TLS 记录协议层，该层实现了一个安全通道，用来加密和认证通过任何面向连接的协议传输的消息；另一层是握手层，包含 TLS 握手协议和两个其他相关协议，它在客户和服务器之间建立一个 TLS 会话（即一个安全通道）。这两层通常都是用客户和服务器应用层的软件库实现的

TLS protocol stack



TLS/SSL handshake protocol (Handshake is performed over an existing connection)



TLS handshake configuration options

Component	*Description*	*Example*
Key exchange method	the method to be used for exchange of a session key	RSA with public-key certificates
Cipher for data transfer	the block or stream cipher to be used for data	IDEA (International Data Encryption Algorithm)
Message digest function	for creating message authentication codes (MACs)	SHA (Secure Hash Algorithm)

Q10 Describe the types of navigation schemes that can be used for name resolution in Domain Name Systems.

DNS 能够处理递归导航和迭代导航两种情况。当与名字服务器联系时，解析器指定需要何种形式的导航。然而，名字服务器并不一定实现实现递归导航，因为其会占用服务器线程，这意味着其他请求会被延迟。

迭代导航：在解析一个名字的时候，客户将改名字送到本地的服务器中，该服务器对其进行解析。若本地名字服务器有这个名字，则返回结果，否则会建议另一个能够提供帮助的服务器对其进行解析，直到改名字被找到，或者没被找到....

递归导航：客户只与一个服务器打交道，如果未储存改名字，则服务器与另一个存储改名字的服务器联系，此服务器将会解析该名字，一直递归下去，直到名字被解析

Q11 Discuss the model architecture of a distributed file system. Illustrate how comprehensive it is by comparing it to the NFS implementation.

It contains three components:

- Flat file service:
  - Concerned with the implementation of operations on the contents of file. *Unique File Identifiers* (UFIDs) are used to refer to files in all requests for flat file service operations. UFIDs are long sequences of bits chosen so that each file has a unique among all of the files in a distributed system.
- Directory Service:
  - Provides mapping between *text names* for the files and their UFIDs. Clients may obtain the UFID of a file by quoting its text name to directory service. Directory service supports functions needed to generate directories and to add new files to directories.
- Client Module:
  - It runs on each computer and provides integrated service (flat file and directory) as a single API to application programs. For example, in UNIX hosts, a client module emulates the full set of Unix file operations.
  - It holds information about the network locations of flat-file and directory server processes; and achieve better performance through implementation of a cache of recently used file blocks at the client.

DFS:It is reliable, fault tolerant, highly available, location transparent

Q12 Write a simple Java RMI program that demonstrates the invocation of remote object services. Implement a service which offers dictionary services. It should support 3 operations/services

(a) "ADD" which adds a new word and its meaning to the dictionary;

(b) "SEARCH" which returns the meaning of a word passed as an argument.

(c) DELETE which deletes word passed as an argument from the dictionary.

Write both server and client programs.

```
// IDictionary.java
package remote
import java.rmi.Remote;
import java.rmi.RemoteException;
```

```
public interface IDictionary extends Remote {
	public Result add(String word, String meaning) throws RemoteException;
	public Result search(String word) throws RemoteException;
	public Result delete(String word) throws RemoteException;
}
// Result.java  结果包装类
package remote

public class Result {
	private int statusCode;
	private String message;

	public Result() {}
	public Result(int statusCode, String message) {
		this.statusCode = statusCode;
		this.message = message;
	}

	public int getStatusCode() { return statusCode; }
	public String getMessage() { return message; }
	public void setStatusCode(int statusCode) {
		this.statusCode = statusCode;
	}
	public void setMessage(String message) {
		this.message = message;
	}

	public String toString() {
		return String.format("Message{statusCode=%d, message=%s}", statusCode, message);
	}
}
// Dictionary.java
package server

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

import remote.IDictionary;
import remote.Result;

public class Dictionary extends UnicastRemoteObject implements IDictionary {
	private Map<String, String> database;

	protected Dictionary() {
```

```
        database = new HashMap<>();
    }

    @Override
    public Result add(String word, String meaning) throws RemoteException {
        if (database.containsKey(word)) {
            return new Result(-1, "word already existed");
        }
        database.put(word, meaning);
        return new Result(1, "success");
    }

    @Override
    public Result search(String word) throws RemoteException {
        if (!database.containsKey(word)) {
            return new Result(-1, "word does not existed");
        }
        return new Result(1, database.get(word));
    }

    @Override
    public Result delete(String word) throws RemoteException {
        if (null == database.remove(word)) {
            return new Result(-1, "word does not existed");
        }
        return new Result(1, "success");
    }
}
// Server.java
package server
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

import remote.IDictionary;

public class Server {
    public static void main(String[] args) {
        try {
            IDictionary remoteDictionary = new Dictionary();

            Registry registry = LocateRegistry.getRegistry();
            registry.bind("Dictionary", remoteDictionary);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
}
```

墨学模拟卷 1

Q1 Briefly discuss the key challenges that one needs to address in the design and development of distributed systems or applications.

- Heterogeneity
  - Heterogeneous components must be able to interoperate
- Distribution transparency
  - Distribution should be hidden from the user as much as possible
- Fault tolerance
  - Failure of a component (partial failure) should not result in failure of the whole system
- Scalability
  - System should work efficiently with an increasing number of users
  - System performance should increase with inclusion of additional resources
- Concurrency
  - Shared access to resources must be possible
- Openness
  - Interfaces should be publicly available to ease inclusion of new components
- Security
  - The system should only be used in the way intended

Q2 Draw a high level architecture diagram for each of the following architectural models and briefly explain each diagram:

i. Client/Server

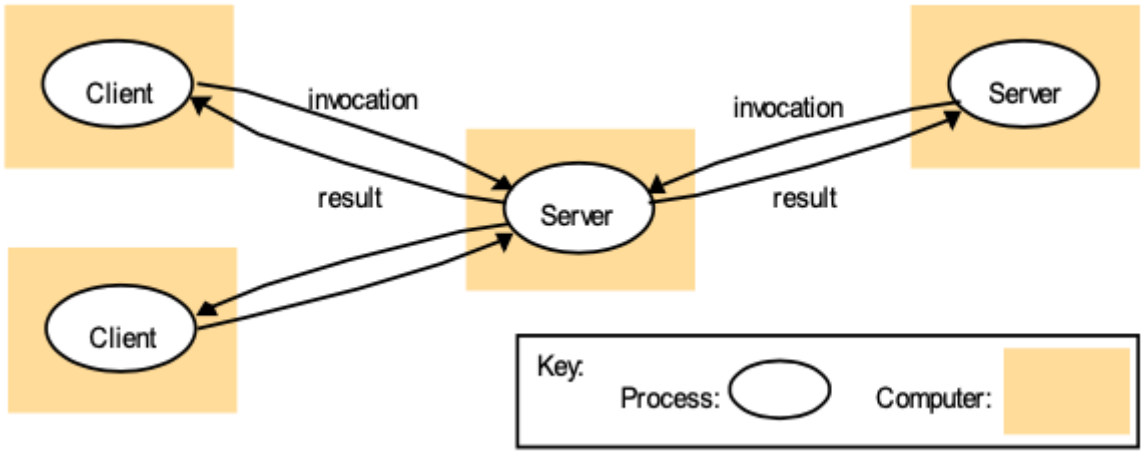
ii. Peer-to-Peer

iii. A service provided by multiple servers

iv. Proxy server

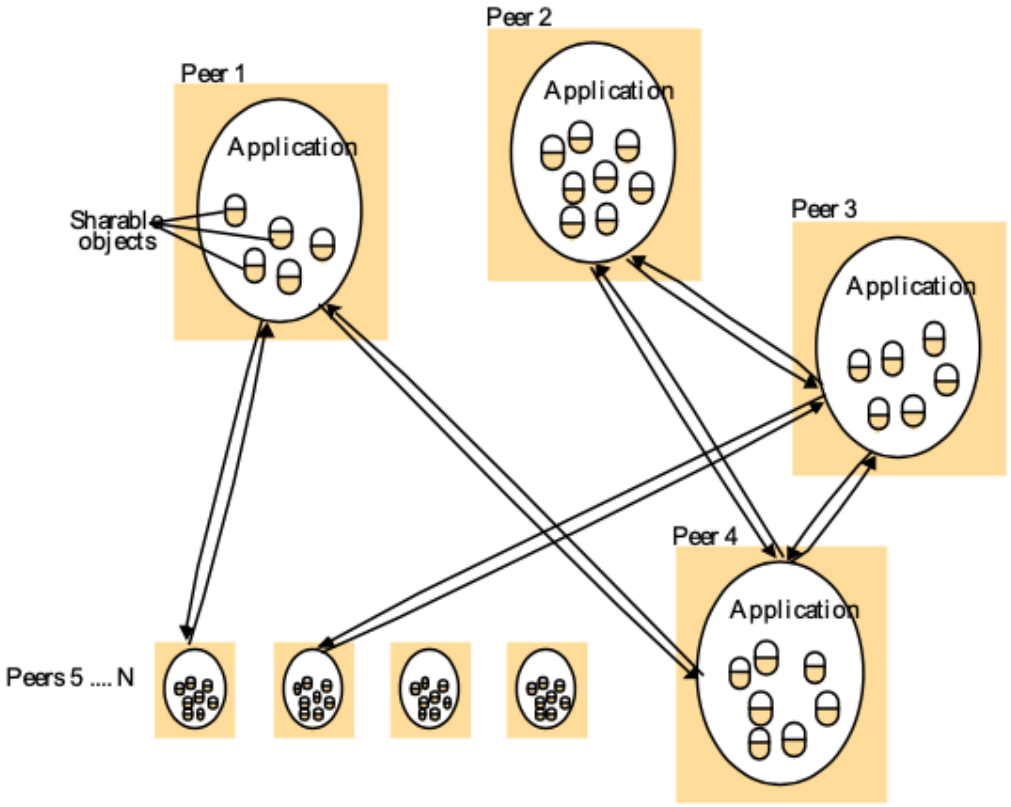
v. Web applets

- Client/Server



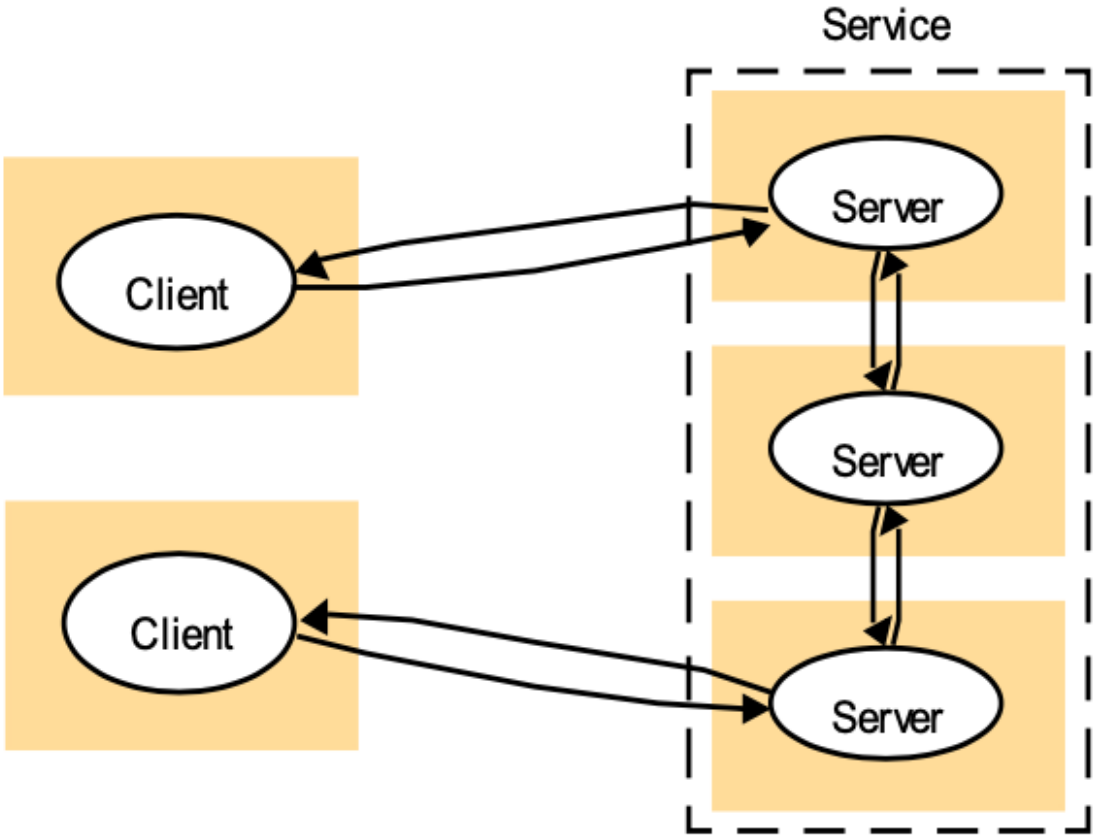
Client processes interact with individual server processes in a separate computer in order to access data or resource. The server in turn may use services of other servers.

- Peer-to-Peer



All of the processes play similar roles, interacting cooperatively as peers to perform distributed activities or computations without distinction between clients and servers. Distributed "white board" – users on several computers to view and interactively modify a picture between them.

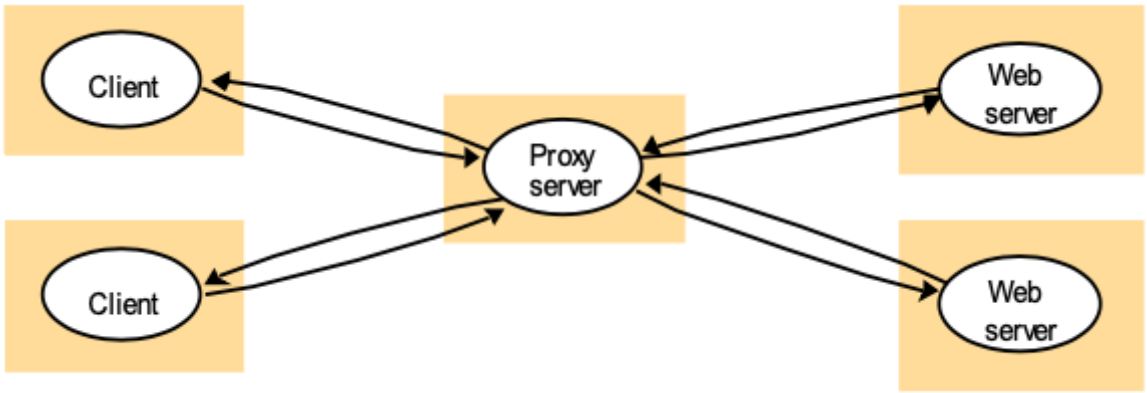
- A service provided by multiple servers



Services may be implemented as several server processes in separate host computers.

Example: Cluster based Web servers and apps such as Google, parallel databases Oracle

- Proxy server

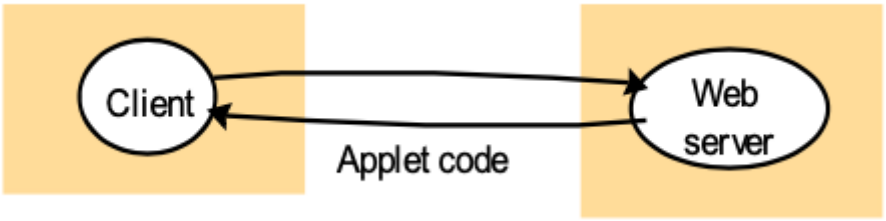


Web proxy servers can operate at client level to improve performance and reduce communication costs for frequently accessed data.

Caching can even be used for dynamic data (such as a google search). This reduces the load on the web servers and improves the performance for end users by reducing the time taken for a dynamic request.

- Web applets

a) client request results in the downloading of applet code



b) client interacts with the applet

Applets downloaded to clients give good interactive response.

Mobile codes such as Applets are potential security threat, so the browser gives applets limited access to local resources (e.g. NO access to local/user file system).

Similar to client-server model but client takes on more responsibility

As code is executed locally the application has good responsiveness.

Need to be careful running such code as client can be exploited. From a server perspective, they cannot control the client environment so there are integrity issues there as well.

Q3. What is the "no global clock" issue in a distributed system?

For this issue, we have two variants of the interaction model.

Describe these two interaction models.

Each computer in a DS has its own internal clock, which can be used by local processes to obtain the value of the current time.

Therefore, two processes running on different computers can associate timestamp with their events.

However, even if two processes read their clocks at the same time, their local clocks may supply different time. This is because computer clock drifts from perfect time and their drift rates differ from one another.

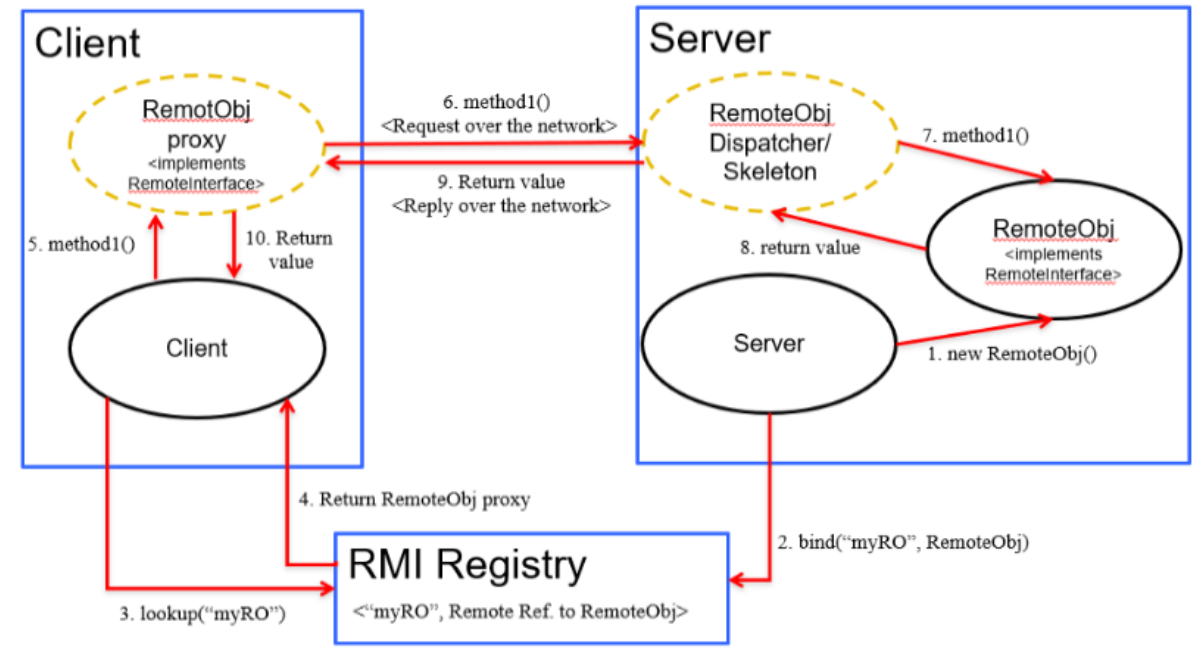
Even if the clocks on all the computers in a DS are set to the same time initially, their clocks would eventually vary quite significantly unless corrections are applied.

- Synchronous DS – hard to achieve:
  - The time taken to execute a step of a process has known lower and upper bounds.
  - Each message transmitted over a channel is received within a known bounded time.
  - Each process has a local clock whose drift rate from real time has known bound.
- Asynchronous DS: There is NO bounds on:
  - Process execution speeds
  - Message transmission delays
  - Clock drift rates.

Q4. Describe three invocation semantics: Maybe, At-least-once, At-most-once. What is the different between Maybe and At-most-once?

- Maybe: The remote procedure call may be executed once or not at all. Unless the caller receives a result, it is unknown as to whether the remote procedure was called.
- At-least-once: Either the remote procedure was executed at least once, and the caller received a response, or the caller received an exception to indicate the remote procedure was not executed at all.
- At-most-once: The remote procedure call was either executed exactly once, in which case the caller received a response, or it was not executed at all and the caller receives an exception. The difference is that the result of Maybe is unknown to the caller, while the caller in At-most- once can clearly know whether the current execution status is executed (1 time) or not executed (0 time).

Q5. Draw a flowchart to briefly represent the remote object invocation process of Java RMI.



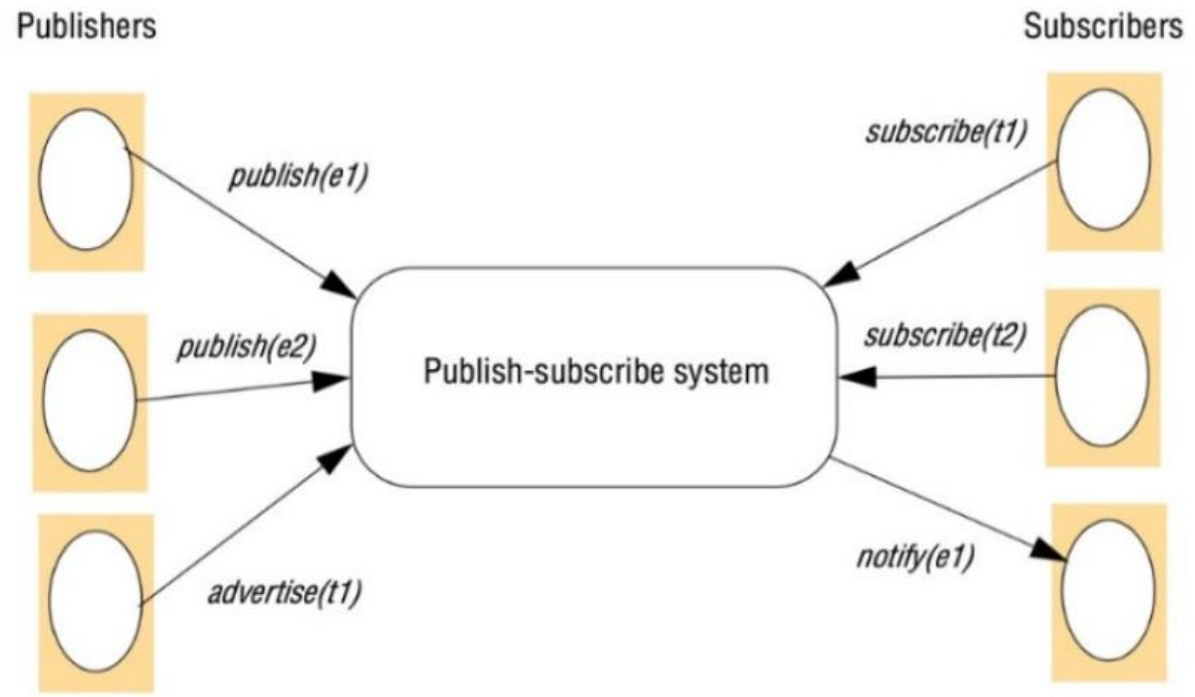
Q6. Explain the following aspects of a publish subscribe system:

i. **Event**

ii. **Notification**

iii. **Subscriber**

iv. **Publisher**



- Event
  - A change of state that is of interest.
- Notification
  - Information regarding an event that is sent to a subscriber.
- Subscriber
  - Express interest in particular events and get notified when they occur.
- Publisher
  - Publish/send structured events to an event service.

Q7. Describe the worst case assumptions and design guidelines for a distributed system. List and briefly explain the threats and attacks that the distributed system may suffer.

- Worst case
  - Interfaces are exposed
    - DSs made up of processes with open interfaces
  - Networks are insecure
    - Messages sources can be falsified.
  - Limit the lifetime and scope of each secret
    - Passwords and keys validity – needs to be time restricted.
  - Algorithms and code are available to hackers
  - Attackers may have access to large resources
  - Minimise the trusted base.
- Security Threats - Three broad Classes:
  - Leakage: Acquisition of information by unauthorised recipients
  - Tampering: Unauthorised alteration of information
  - Vandalism: Interference with the proper operation of systems
- Method of Attacks
  - Eavesdropping - A form of leakage
    - obtaining private or secret information or copies of messages without authority.
  - Masquerading – A form of impersonating
    - assuming the identity of another user/principal – i.e, sending or receiving

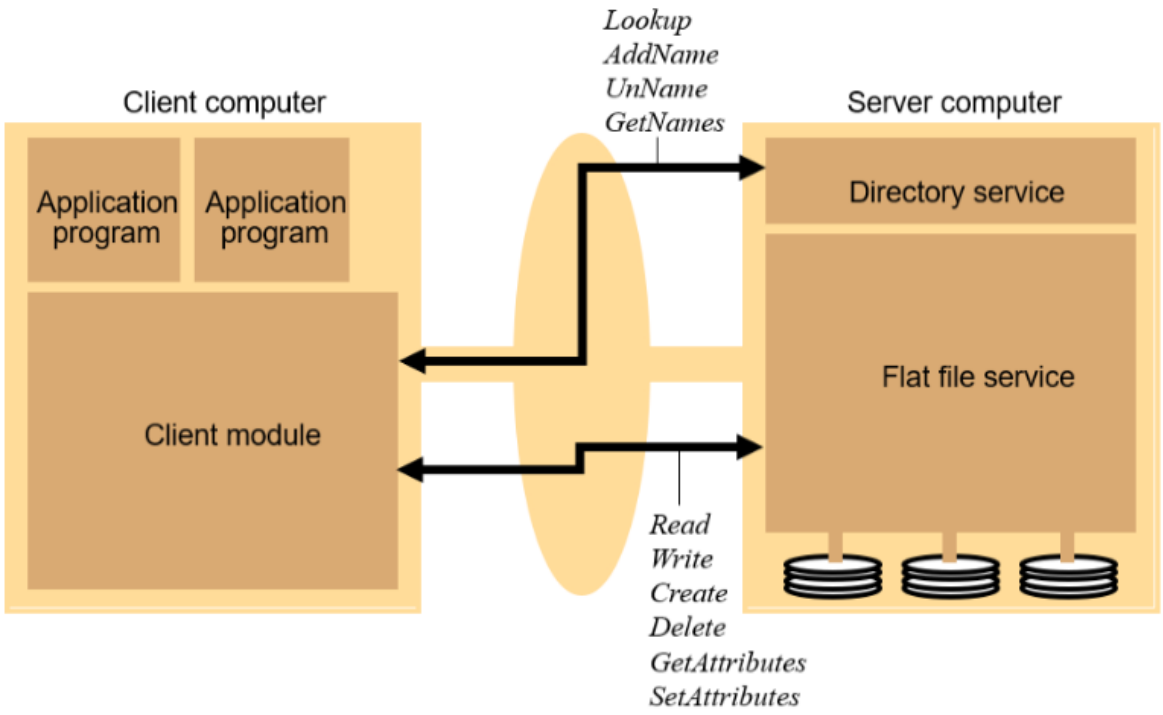
messages using the identity of another principal without their authority.

- Message tampering
  - altering the content of messages in transit: man-in-the-middle attack (tampers with the secure channel mechanism)
- Replaying
  - storing secure messages and sending them at a later date
- Denial of service - Vandalism
  - flooding a channel or other resource, denying access to others

Q8. Explain what is a digital certificate, including what is the basic technique used to create a digital certificate, and what is a certificate chain.

- A digital certificate is a document containing a statement signed by a principal.
- It binds information together, like a principal’ s id with its public key, and is digitally signed by a certificate authority.
- Sara creates a digital signature by encrypting a digest of the data to be signed with her private key.
- Certificate chain: A series of certificates where each certificate's signature is authenticated by the subsequent certificate

Q9. Describe the file service architecture and its three main components: flat file service, directory service, and client module.



- Flat file service:
  - Concerned with the implementation of operations on the contents of file. Unique File
  - Identifiers (UFIDs) are used to refer to files in all requests for flat file service operations. UFIDs are long sequences of bits chosen so that each file has a unique among all of the files in a distributed system.
- Directory Service:
  - Provides mapping between text names for the files and their UFIDs. Clients may obtain the UFID of a file by quoting its text name to directory service. Directory service supports functions needed to generate directories and to add new files to directories.

- Client Module:
  - It runs on each computer and provides integrated service (flat file and directory) as a single API to application programs. For example, in UNIX hosts, a client module emulates the full set of Unix file operations.
  - It holds information about the network locations of flat-file and directory server processes; and achieve better performance through implementation of a cache of recently used file blocks at the client.

墨学模拟卷 2

Q1. List at least three aspects of distribution transparency and give relevant examples.

- Access transparency
  - Access to local or remote resources is identical
  - E.g. Network File System
- Location transparency
  - Access without knowledge of location
  - E.g. separation of domain name from machine address.
- Failure transparency
  - Tasks can be completed despite failures
  - E.g. message retransmission, failure of a Web server node should not bring down the website.
- Replication transparency
  - Access to replicated resources as if there was just one. And provide enhanced reliability and performance without knowledge of the replicas by users or

application programmers.

- Migration (mobility/relocation) transparency
  - Allow the movement of resources and clients within a system without affecting the operation of users or applications.
  - E.g. switching from one name server to another at runtime; migration of an agent/process from one node to another.
- Concurrency transparency
  - A process should not notice that there are other sharing the same resources
- Performance transparency:
  - Allows the system to be reconfigured to improve performance as loads vary
  - E.g., dynamic addition/deletion of components, switching from linear structures to hierarchical structures when the number of users increase
- Scaling transparency:
  - Allows the system and applications to expand in scale without changes in the system structure or the application algorithms.
- Application level transparencies:
  - Persistence transparency
    - Masks the deactivation and reactivation of an object
  - Transaction transparency
    - Hides the coordination required to satisfy the transactional properties of operations

Q2. What is middleware? What is the main purpose of middleware? List three middleware technologies that we have learned in class.

A layer of software whose purpose is to mask heterogeneity present in distributed systems and to provide a convenient programming model to application developers.
RMI/RPC, Request-Reply Protocol, Marshalling and external data representation: CORBA/JSON/XML

Q3. In the failure model of a distributed system, there are three types of failures. Briefly describe these three types of failures. Which type of failure do you think is the most difficult to deal with, and why?

Omission failure refer to cases where the process or communication channel fails to perform a requested action.

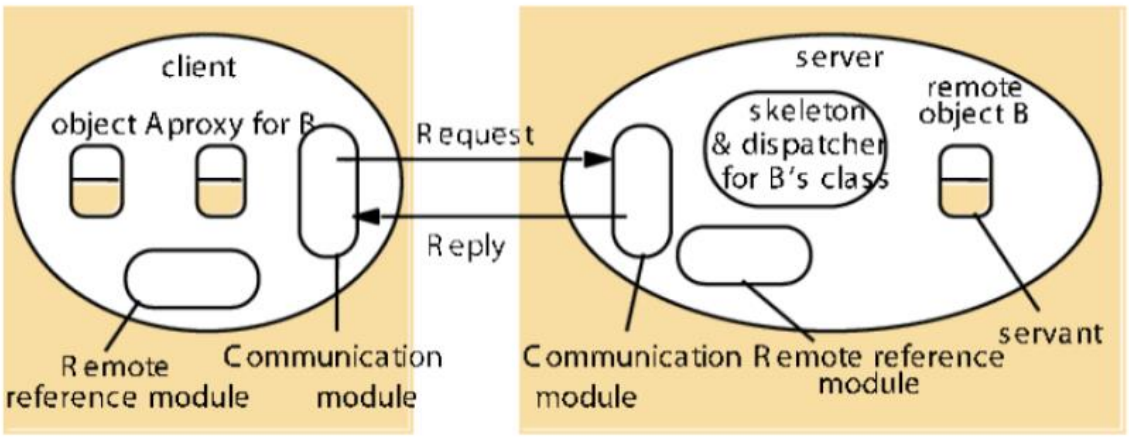
Class of failure	*Affects*	*Description*
Fail-stop	Process	Process halts and remains halted. Other processes may detect this state.
Crash	Process	Process halts and remains halted. Other processes may not be able to detect this state.
Omission	Channel	A message inserted in an outgoing message buffer never arrives at the other end's incoming message buffer.
Send-omission	Process	A process completes a *send,* but the message is not put in its outgoing message buffer.
Receive-omission	Process	A message is put in a process's incoming message buffer, but that process does not receive it.
Arbitrary (Byzantine)	Process or channel	Process/channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commit omissions; a process may stop or take an incorrect step.

Timing failures are related to synchronous messages, where a set bound [clock drift, message ack, process execution time] exceeds defined bounds.

Class of Failure	*Affects*	*Description*
Clock	Process	Process's local clock exceeds the bounds on its rate of drift from real time.
Performance	Process	Process exceeds the bounds on the interval between two steps.
Performance	Channel	A message's transmission takes longer than the stated bound.

Arbitrary failures are the most tricky to deal with, where any type of error can occur. Corrupt data, unexpected responses, etc.

Q4. Describe the structure of RMI and the role of each component. You may draw diagrams to help with your description.



- Communication Module
  - The two cooperating communication modules carry out the request-reply protocol
  - It uses three fields from the message:
    - Message type
    - Request ID
    - Remote object reference
  - It is responsible for implementing the invocation semantics
  - The server module queries the remote reference module to obtain the local reference of the object and passes the local reference to the dispatcher for the class
- Remote Reference Module
  - Creates remote object references

    - Has a remote object table that records the correspondence between local object references in that process and remote object references (system-wide)
    - The remote object table contains an entry for each:
        - Remote object reference held by the process
        - Local proxy
    - Entries are added to the remote object table when:
        - A remote object reference is passed for the first time
        - When a remote object reference is received and an entry is not present in the table
    - For example, when a request message arrives, the table is used to find out which local object is to be invoked
- Proxy
    - Plays the role of a local object to the invoking object. There is a proxy for each remote object which is responsible for:
        - Marshalling the reference of the target object, its own method id and the arguments and forwarding them to the communication module
        - Unmarshalling the results and forwarding them to the invoking object
- Dispatcher
    - There is one dispatcher for each remote object class. Is responsible for mapping to an appropriate method in the skeleton based on the method ID
- Skeleton
    - Is responsible for:
        - Unmarshalling the arguments in the request and forwarding them to the servant
        - Marshalling the results from the servant to be returned to the client

Q5. Answer the following questions about Java RMI:

i. **Can objects be created remotely? Explain your answer.**

ii. **Explain the difference between a local invocation and a remote invocation. If two Java Virtual Machines are on the same physical machine, and invocation is made between them, is this local or remote?**

iii. **What is a remote reference? Explain how an object obtains a remote reference.**

i. Objects cannot be directly created remotely, i.e. with the new command, but a servant factory object can be used to indirectly create an object on a remote machine.

ii. A local invocation takes place between objects within the same Java virtual machine, whereas a remote invocation takes place between objects on different Java virtual machines. If the two Java virtual machines are on the same physical machine, it is still considered a remote invocation.

iii. A remote reference is a reference to an object that is not within the local Java virtual machine. An object can obtain a remote reference by accessing the registry process, knowing the objects name.

Q6. What are the advantages of message queues compared to traditional sending and receiving? What problems can be caused by using a message queue system in a distributed system?

- Advantages:

- Decoupling
  - One of the shortcomings of the traditional model is that the coupling between systems is too strong. With the message queue system, messages are written to the message queue, and the components that need the message subscribe from the message queue by themselves, and the system does not need to make any changes.
- Asynchronous
  - Non-essential business logic runs asynchronously to speed up response.
- Peak clipping
  - When the amount of concurrency is large in the traditional mode, a large number of requests impact the database at the same time, which can easily cause abnormal database connection. With the introduction of a message queue, the system can gradually pull messages from the message queue according to the amount of concurrency that the database can handle.

- Problems:
  - The availability of the system is reduced: if the message queue hangs up, the system will also be affected.
  - The complexity of the system increases: many additional issues need to be considered, such as consistency, reliable transmission, messages must not being repeatedly consumed, and so on.

Q7. Compare the monolithic and microkernel and explain their respective advantages and disadvantages.

- Monolithic:
  - + Better application performance: Relative efficiency with which operations

can be invoked is high because even invocation to a separate user-level address space on the same node is more costly.

- - Massive: It performs all basic OS functions and takes up in the order of megabytes of code and data
- - Undifferentiated: It is coded in a non-modular way (traditionally) although modern ones are much more layered.
- - Intractable: Altering any individual software component to adapt it to changing requirements is difficult.

- Microkernel:
  - Compared to monolithic, microkernel design provides only the most basic abstractions. All other system services are provided by servers that are dynamically loaded precisely on those computers in the DS that require them. Clients access these system services using the kernel’ s message-based invocation mechanisms.
  - + A relative small kernel is more likely to be free of bugs than one that is larger and complex.
  - + Extensibility and its ability to enforce modularity behind memory protection boundaries.
  - - Performance is a trade-off: The additional information transmission and invocation procedure affect the execution efficiency.

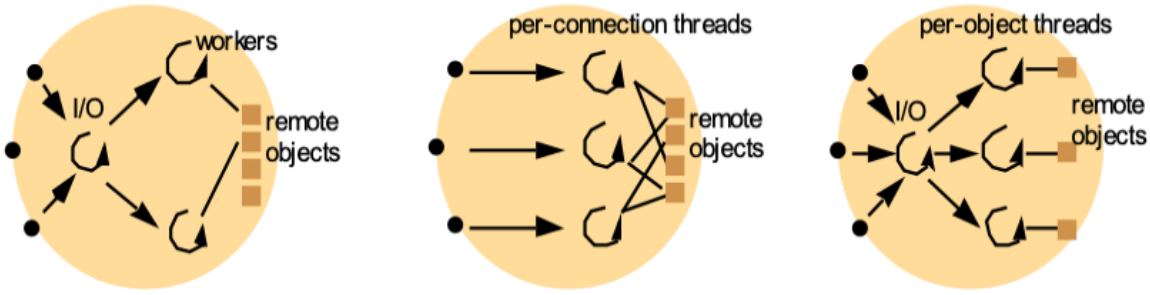
Q8. Explain the following multithreaded server architectures:

i. Thread-per-request

ii. Thread-per-connection

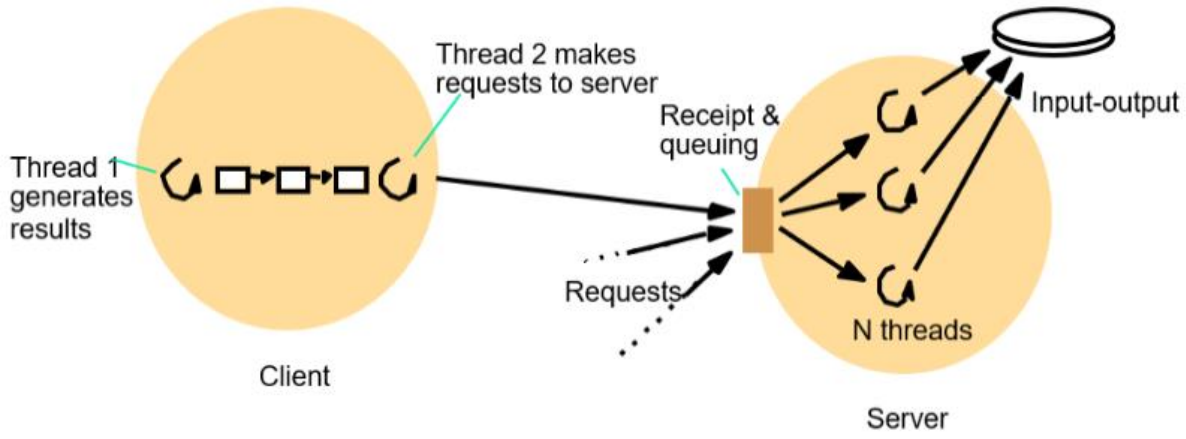
iii. Thread-per-object

iv. worker-pool



a. Thread-per-request b. Thread-per-connection c. Thread-per-object

- Thread-per-request
  - IO Thread creates a new worker thread for each request and worker thread destroys itself after serving the request.
- Thread-per-connection
  - Server associates a Thread with each connection and destroys when client closes the connection. Client may make many requests over the connection.
- Thread-per-object
  - Associates Thread with each object. An IO thread receives request and queues them for workers, but this time there is a per-object queue.
- worker-pool

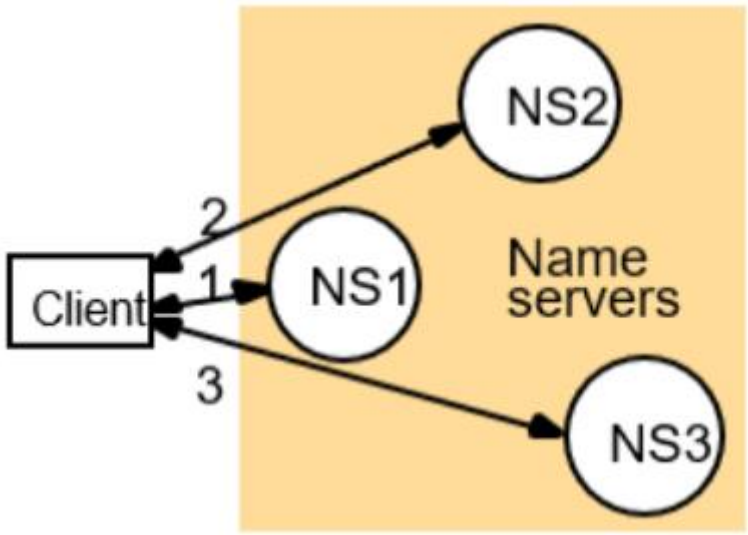


In worker-pool architectures, the server creates a fixed pool of worker threads to process requests.

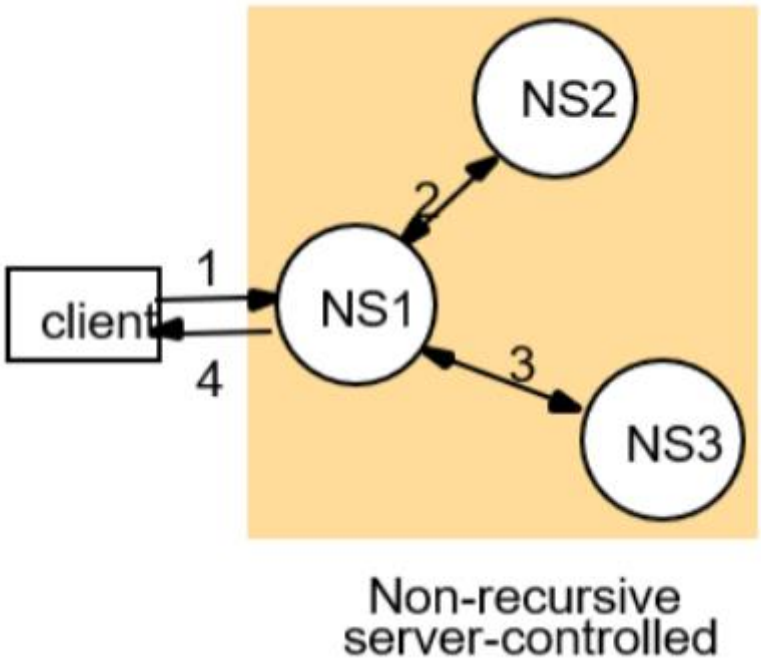
The module "receipt and queuing" receives requests from sockets/ports and places them on a shared request queue for retrieval by the workers.

Q9. Show the navigation procedure of Iterative/Non-recursive/recursive navigation. Discuss the difference between them.

Iterative:

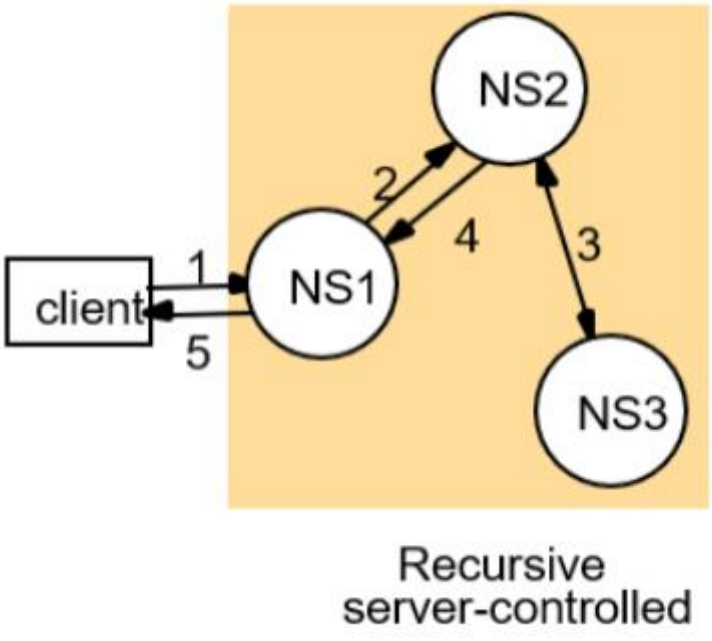


Non-recursive:



Non-recursive
server-controlled

Recursive:



Recursive
server-controlled

- Iterative - client contacts name servers ns1-ns3 itself until it can resolve address
- Non-recursive server controlled
  - name server contacts other ns until it can resolve address

    - it is performed by the first server
    - the server bounces back the next hop to its client
- Recursive server controlled
    - name server passes request down chain, is unaware of who eventually resolves IP
    - it is performed by the naming server
    - the server becomes like a client for the next server
    - this is necessary in case of client connectivity constraints

墨学模拟代码题

Q1

Now we have a simple distributed chat room with Client-Server architecture and TCP connection. The IP address of our server is 10.182.76.43. When the service starts, our server will continue to monitor requests from all clients, and the listening port is 8080.

Suppose we now have a client with an IP address of 10.30.12.10 who wants to access the server. After the connection is established, the client will send a string "hello" to the server, and the server will confirm whether the first message from the client is "hello". If it is, the server will reply "ACK". If not, it is considered an invalid connection, and the server will reply with the string "NAK" and take the initiative to disconnect.

Please design the two classes: ChatServer and ChatClient respectively to implement the above process.

Q2

Suppose we need to design an Order class for an online trading platform to ensure the thread safety of multi-threaded concurrent order information. There are three methods implemented the Order class:

a) String getOrderDetails() - Get the detailed information of the current order, including the order ID, order creation time and order status (whether the payment is completed), and the result is returned as a String.

b) void updatePaymentStatus(Boolean isPaid) - Update the payment status of the order, true means successfully paid, false means payment has not been completed.

c) String readOrderHistory() - Get the current payment history of the current order. Since the query is the record at the current moment and will not modify any information, this method is idempotent, that is, thread-safe.

Please design the Order class based on the above description.

Note: You don't need to declare any variables or design the actual implementation of the method, just declare the method correctly.

Q3

Implement a simple Java RMI program: The server will provide a WorldClock service, allowing the client to obtain the time in the specified time zone, that is, allowing the client to call the following method:

```
LocalDateTime getLocalDateTime(String zoneId);
```

To implement RMI, the server and client must share the same interface. We define a WorldClock interface, the code is as follows:

```
public interface WorldClock extends Remote {
    LocalDateTime getLocalDateTime(String zoneId) throws RemoteException;
}
```

The next step is to write the implementation class of the server, because the call method getLocalDateTime() requested by the client will eventually return the result through this implementation class. The implementation class WorldClockService code is as follows:

```
public class WorldClockService implements WorldClock {
    @Override
    public LocalDateTime getLocalDateTime(String zoneId) throws RemoteException {
        return LocalDateTime.now(ZoneId.of(zoneId)).withNano(0);
    }
}
```

Now, write client and server code to implement RMI calls.

Note: you may use the following code - worldClock.getLocalDateTime("Asia/Shanghai"); - to get the local time of Shanghai by RMI.

2016 Semester 1

Q1

(a) [5 marks] Consider a large computer lab, e.g. like those found on a univer- sity campus. The computer lab allows users to login to any computer using their username and password and makes various software and other resources available that are specific to that user. List and briefly explain, with respect to this example, five challenges of distributed systems.

(b) [5 marks] Consider the provisioning of general purpose applications in the cloud, where a web browser is used to connect to and operate the application. For example, an instance of a spreadsheet application can run on the cloud and a user can connect to it via their web browser to work on spreadsheet data. What type of distributed system architectural model best describes this? Give a reason for your answer. Give three reasons that support the use of this architectural model. Give an example of an application for which the architectural model is not suitable.

Q2 [5 marks] In your first project you implemented a multi-server system that broad- cast various kinds of messages between the servers to provide the service. Answer the following questions related to your first project:

(a) In the project each subsequent server could connect to only one existing server at startup. Now consider allowing each subsequent server to connect to mul- tiple existing servers at startup. What major problem would this cause, as- suming the protocol does not change in any other way? What changes to the protocol/implementation can you suggest that could fix this major problem? Briefly explain your suggestion.

(b) Explain the failure model that was assumed in the first project with respect to all components and interactions.

Q3

(a) [3 marks] Briefly explain three fundamental aspects of a communication chanel that are important from the point of view of a distributed system.

(b) [2 marks] What is meant by a *timing failure*? Briefly explain an example of a distributed system where a timing failure would have significant impact.

Q4

(a) [3 marks] State three differences between UDP and TCP communication.

(b) [2 marks] What is a benefit of XML over JSON format? What is a benefit of JSON over XML format?

(c) [5 marks] Explain maybe semantics, at-most-once semantics and at-least-once semantics, in terms of a client sending a request to a server. What level of semantics does the RR protocol achieve? What level of semantics does Java RMI enforce?

Q5

(a) [5 marks] Consider a "music popularity" application that runs on mobile de- vices. Each user's mobile device reports the name of the song that is currently playing on the device, to a central server. The central server ranks the songs in terms of popularity across all devices and makes the results known back to all users, so that the most popular songs can be seen by everyone. Choose either tuple space, message queue or publish/subscribe paradigm and explain how the application could be implemented using your chosen paradigm. Draw an architectural diagram to aid your explanation.

(b) [5 marks] Consider implementing a distributed shared memory middleware. Describe what kind of protocol and messages that you would use, and why. Discuss the operating system support that your implementation would require.

Q6

(a) [4 marks] Explain the difference between a worker pool multi-thread model and a thread-per-connection model, including details concerning what concur- rency control and queueing is required in each case.

(b) [1 marks] Explain the difference between virtualization and emulation.

Q7

(a) [2 marks] Some distributed systems, such as secure shell, require the user to explictly trust a public key provided by the server, when connecting for the first time. Name the security risk that this presents and explain why this is a security risk.

(b) [3 marks] Explain what is meant by challenge-response and give an example of its use.

Q8

(a) [5 marks] Describe the caching policy used by NFS at the client. What parameters are used? What checks are done, in what order and why?

(b) [2 marks] Explain how DNS can be used to distribute the load of incoming requests over a set of servers.

(c) [3 marks] Consider an application where resource objects are stored across a multi-server system. As a requirement of the system, given a name of an object, it is critical that the object needs to be accessible as fast as possible. Describe a name space and name format that would be best to address this requirement and explain why. Is your name format an example of a pure name or not?

2014 Semester 2

Q1

(a) [1 marks] When considering the definition of a distributed system, a computer network in unto itself is often not said to be a distributed system. Why is this the case?

(b) [1 marks] What is the difference between availability and reliability?

(c) [3 marks] Instead of having an non-shared printer attached to each computer in a computer network, a single printer can be used and shared among all of the computers.

i. Is this an example of a distributed system? Explain.

ii. Explain a reason for, and a reason against this approach.

Q2 [5 marks] List and briefly explain four general approaches to failure handling in a distributed system. Give an example for each approach.

Q3

(a) [4 marks] What is an architectural model? In your explanation include the important aspects or steps when developing an architectural model.

(b) [4 marks] What is a fundamental model? In your explanation discuss three aspects of distributed systems that are described using a fundamental model.

(c) [2 marks] Explain the following classes of failure:

i. **Fail-stop**
ii. **Crash**
iii. **Omission**
iv. **Arbitrary**

Q4 [3 marks] Consider a server process that has a single TCP server socket, bound and listening on port 4242.

(a) While listening for incoming TCP connections on port 4242, can the process also receive UDP packets on port 4242?

(b) Assuming that each client is connecting from a different host, what operating system aspect limits the number of concurrent client connections that the server process can handle?

(c) Assuming the server process has 5 concurrently open connections from clients how many ports does the server process require? Explain your answer.

(d) Is it possible for a client to connect from port 4242 to the server process? Explain your answer.

Q5 [2 marks] What is a benefit of XML over JSON format? What is a benefit of JSON over XML format?

Q6 [5 marks] Explain the exchange protocol used in the first project. Draw an inter- action diagram and explain the messages that are sent. What was the purpose of using a counter?

Q7

(a) [2 marks] Discuss two advantages and two disadvantages of using high level middleware (e.g. Java RMI middleware) compared to socket APIs in a distributed system.

(b) [4 marks] Describe the Tuple Space paradigm. Use a diagram and include a description of the API used in a Tuple Space.

(c) [4 marks] Describe the Publish/Subscribe paradigm. Use a diagram and include and include a description of the API used in a Publish/Subscribe system.

Q8

(a) [4 marks] Discuss centralized and decentralized architectures for distributed load management. Critically compare them.

(b) [1 marks] Explain the difference between virtualization and emulation.

Q9

(a) [3 marks] Explain what is a digital certificate and what is a certificate chain

(b) [3 marks] Give three worst-case assumptions when designing a secure system.

Q10

(a) [5 marks] Describe the caching policy used by NFS at the client. What parameters are used? What checks are done, in what order and why?

(b) [2 marks] Explain how DNS can be used to distribute the load of incoming requests over a set of servers.

(c) [1 marks] What is meant by pure name?

(d) [1 marks] What is meant by unification?

2014 Semester 1

Q1

(a) [4 marks] List and briefly explain four reasons why resource sharing is beneficial.

- Reduces cost by allowing a single resource for a number of users, rather than a identical resource for each user.
- Facilitates interactions among users, e.g. through a shared file system.
- Increases capacity, e.g. by allowing otherwise unused local disk space to be used remotely.
- Increases availability, e.g. through redundancy of resources.

(b) [1 marks] In the context of distributed systems, what is meant by the term independent failure?

- An independent failure is where the failure of a part of a distributed system can occur obliviously to the operation of the remaining parts of the system.

Q2

(a) [4 marks] For each of the following distributed system challenges, explain why it is a challenge, giving an example for each case:

- Transparency
    - Transparency is a challenge because sometimes the application or user is required to make sensible decisions at the lower, otherwise transparent, layer.

E.g. it is sometimes better to tell the user about Internet failure, rather than make such failure transparent, so that the user can perhaps fix the problem.

- Security
  - Security is a challenge because not all security problems are known in advance - the attacker is continuously trying new forms of attacks. E.g. the Heartbleed bug was "discovered" only recently and presumably was being used to attack some systems.
- Heterogeneity
  - Heterogeneity is a challenge because supporting a number of different platforms requires more development, including more debugging. E.g. supporting both Android and iOS adds extra development effort.
- Openness
  - Openness is a challenge because defining an API that is flexible and easy to use, as well as ensuring that implementations adhere to it, requires significant effort. E.g. different browser vendors will interpret HTML in different ways and JavaScript engines may not work the same.

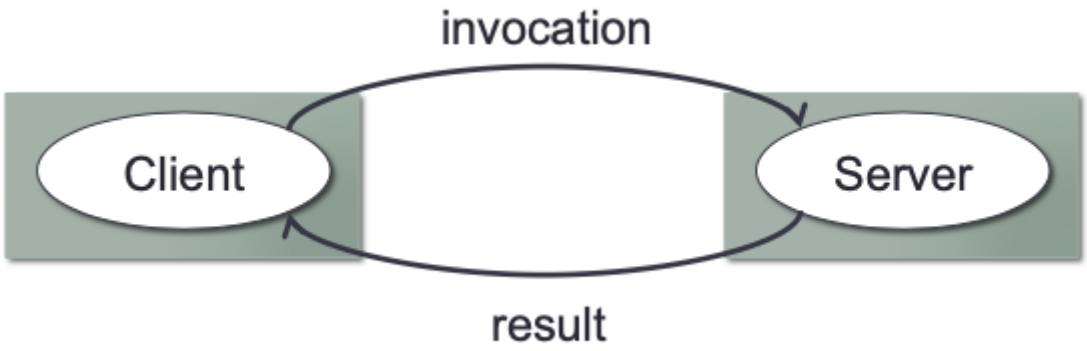
(b) [1 marks] What is middleware in the context of distributed systems?

- Middleware is a software layer that sits between the application and the platform, that provides distributed services of some kind, usually transparently to the application.

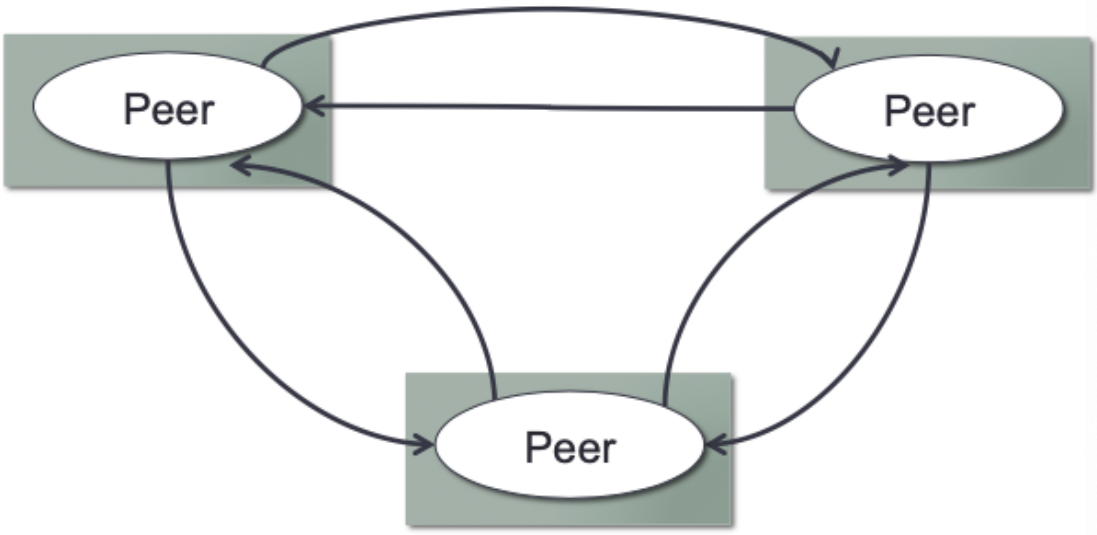
Q3

(a) [4 marks] Draw a high level architecture diagram for each of the following architectural models and briefly explain each diagram:

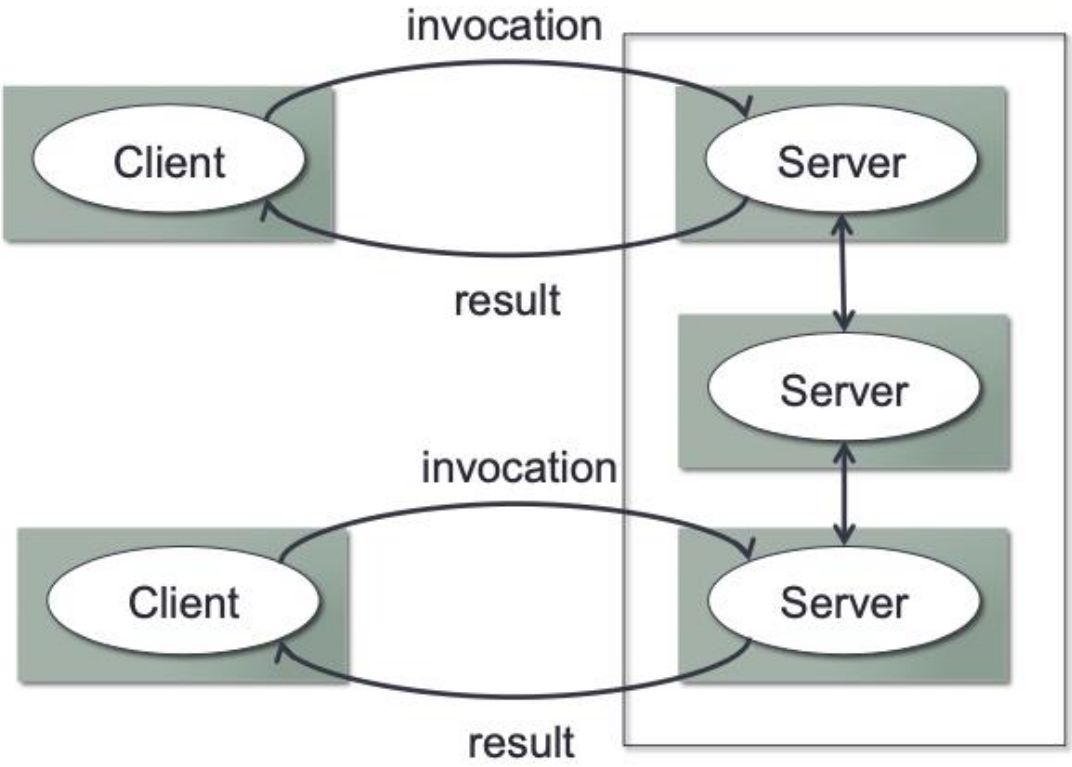
1. **Client/Server**



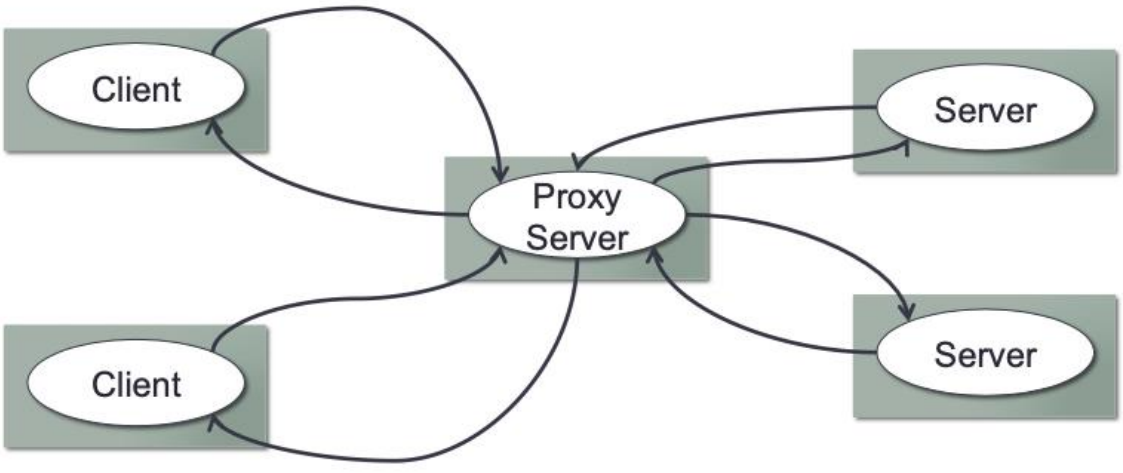
2. **Peer-to-Peer**



3. A service provided by multiple servers



4. Proxy server



(b) [4 marks] Draw an interaction diagram to explain the RRA protocol. Explain the diagram. Explain what problem(s) the RRA protocol solves that the RR protocol does not.

- [The diagram should include the messages, as well as the timers used and the fact that the server will cache results.] The RRA protocol addresses the problem of recomputing responses, where the server can properly discard results when it receives an acknowledgement. The RR protocol does not allow this. 这题应该不考，这学期查不到

(c) [2 marks] Consider a client/server system where the server is sending a stream of temperature readings to the client. Each temperature reading is a 32bit floating point number. Design a format for the stream, using either a well known data representation, or your own representation. Explain your design exactly, so that a third person could implement it without ambiguity.

- The stream can a binary stream over TCP. The 32 bit floating point numbers can be written in big endian format, i.e. 4 bytes. The start of the stream can be a single unsigned byte that represents how many 32 bit floating point numbers will follow. If this byte is 0 then the stream is to be closed. An arbitrary length stream can be constructed using this format.

Q4

(a) [4 marks] Answer the following questions about Java RMI :

i. Can objects be created remotely? Explain your answer.

- Objects can not be directly created remotely, i.e. with the new command, but a servant factory object can be used to indirect create an object on a remote machine.

ii. Explain the difference between a local invocation and a remote invocation. If two Java Virtual Machines are on the same physical machine, and invocation is made between them, is this local or remote?

- A local invocation takes place between objects with the same Java virtual machine; whereas a remote invocation takes place between objects on different Java virtual machines. If the two Java virtual machines are on the same physical machine, it is still considered a remote invocation.

iii. What is a remote reference? Explain how an object obtains a remote reference.

- A remote reference is a reference to an object that is not within the local Java virtual machine. An object can obtain a remote reference by using the Registry server, knowing the objects name.

(b) [2 marks] Explain the following aspects of a publish subscribe system:
我记得这个设计模式这门课今年也没讲

i. Event

- An Event is some change of state that is of interest.

ii. Notification

- is information regarding an event that is sent to a subscriber.

iii. Subscriber

- is an entity that is waiting for Notifications.

iv. Publisher

- is an entity that sends Notifications to Subscribers.

Q5

(a) [2 marks] Explain what is a *persistent asynchronous invocation* and describe an example application that would benefit from this technique.

- A persistent asynchronous invocation is one that will continue to be attempted over a long period of time if errors occurs. E.g. sending an SMS from a mobile phone in the presence of connection problems.

(b) [2 marks] Explain what is meant by *process migration*. Explain two major complications with process migration.

- Process migration is check-pointing a running process on a local machine, move the processes address space to a remote machine, and re-starting the process on the remote machine. Two complications are that the local and remote machine may not share the same instruction set, and the process may have references to resources that are bound to the local machine.

(c) [4 marks] Apart from actual network delay, list and briefly explain four factors that contribute to the delay incurred when making an RMI call.

- system transitions, when the process makes a system call to initiated the RMI call
- marshalling/unmarshalling, when converting objects to a format for transmission
- buffering, copying data between the process and the system and perhaps internally within the system
- dispatching, where the object at the remote machine needs to be looked up in order to call its remote interface

(d) [6 marks] Considering your second project, answer the following questions: 这题显然不考

i. **Draw an architectural diagram that shows the threads used in a possible implementation, the main resources and the interactions between the threads and resources. Briefly explain the diagram, including an explanation of each thread and resource, and any concurrency control needed.**

[diagram needs to focus on what is asked for, be clear and consistent] Class Discussion

ii. **Considering the overall performance of the application, what would you say is the main limiting factor to obtaining a good quality stream of images? Explain your answer.**

The biggest problem for obtaining a good quality stream of images is the bandwidth required; a better encoding method would help a lot here.

Q6

(a) [3 marks] List and briefly explain three worst-case asssumptions when designing a secure system.

- All communications between processes can be copied, modified and retransmitted. Attackers can obtain information that they should not and can pretend to be a legitimate party.
- All of the source code is known to the attacker. Knowing the source code can help the attack discover vulnerabilities.
- The attacker has unlimited computing resources. Encryption can eventually be broken.

(b) [4 marks] Explain what is a *digital certificate*, including what is the basic technique used to create a digital certificate, and what is a *certificate chain*.

- A digital certificate is a document that binds information together, most usually an entities identification with its public key, and is digitally signed by an entity. A digital signature is created by an entity by taking the relevant information and encrypting a digest of it with that entities private key, from a public/private key pair. A series of certificates where each certificate's signature is authenticated by the subsequent certificate is a certificate chain, finishing at a self-signed certificate.

(c) [2 marks] Briefly explain the infamous *Heartbleed Bug*, as discussed in lectures. What popular security system did it affect and basically how did it work?

- The Heartbleed bug was found in the OpenSSL implementation. A client sending a heartbeat packet could fake how many bytes it sent, to fool the server into copying data (possibly sensitive) from its buffer into the return packet.

Q7 [3 marks] Consider a distributed file system that implements file replication. What is file replication and what benefit(s) does it have? What problem(s) can occur in a system that uses file replication? Explain your answers.

- File replication is where the same file is stored on different nodes of the distributed system and it is kept synchronised, i.e. changes at one node are reflected at all nodes. It can provide fault tolerance and increased performance. However changes can lead to inconsistency between the replicas. This can take time and network overhead to overcome.

Q8 [3 marks] Before DNS, a single file was used to store all name information for computers on the Internet. This file was downloaded by everyone on the Internet on a daily basis, from the well known host that provided it. List and briefly explain three problems with this approach, that prompted people to develop DNS.

- The well known host became a bottleneck, as the Internet grew to include many millions of nodes that needed the file.
- Organizations wanted to administer the Internet names in their own networks.
- A more general naming service was required, rather than one that just looks up Internet addresses.

Q9 [5 marks] Consider an image-resizing service. Clients submit an image to the service along with a resize factor and the service returns the image scaled by the resize factor. The service is expected to act in real time – e.g. for images of less than 1 megabyte in size the service should respond in less than 500ms for 99% of the requests. Also, the system should provide 99.9% availability in a month, meaning that the system is rarely unavailable. Using your knowledge of distributed systems, propose an overall approach for this system, draw an architecture diagram, and explain the protocol that you propose. Your approach should address the common challenges of a distributed system and specifically the challenges suggested above. Explain how it does so. Your answer should be less than one page.

[Emphasis is on the architecture of the system that is best suited to provide availability and low response times. E.g. a multi-server architecture, perhaps with layers, using DNS to load balance. Description should included the protocol with details about the messages. E.g. TCP might be used to initially upload the image (since images can be several megabytes) to an I/O server in a multi-server system,

and then I/O server could use a shared file system with a second layer of CPU servers that resize the image, and then the I/O server can respond on the TCP socket with the resized image; finally closing the socket.]