Intern

COMP90007 2021 Semester 1
Tutorial 01

1. Provide a definition of a Distributed System

1. Provide a definition of a Distributed System

- A system in which hardware or software components located at networked computers communicate and coordinate their actions only by passing message [Coulouris]

- A collection of independent computers that appears to its users as a single coherent system [Tanenbaum]

2. Briefly explain the difference between a computer network and a distributed system.

2. Briefly explain the difference between a computer network and a distributed system.

A Computer Network: Is a collection of spatially separated, interconnected computers that exchange messages based on specific protocols. Computers are addressed by IP addresses.

A Distributed System: Multiple computers on the network working together as a system. The spatial separation of computers and communication aspects are hidden from users.

3. List three reasons for using a distributed system.

3. List three reasons for using a distributed system.

- Economy (cost effective)
- Reliability (fault tolerance)
- Availability (high uptime)
- Scalability (extendible)
- Functional Separation (Modularity)

The main motivation to build and use distributed systems is Resource Sharing

- Hardware Resources (Disks, printers, scanners etc.)
- Software Resources (Files, databases etc)
- Other (Processing power, memory, bandwidth)

4. Briefly explain four consequences when using distributed systems, i.e. issues that arise that are not present otherwise.

4. Briefly explain four consequences when using distributed systems, i.e. issues that arise that are not present otherwise.

- Concurrency
- Heterogeneity
- No Global Clock
- Independent Failures

Java IDE

IDE - Integrated Development Environment

Used to facilitate development, options available:

- Eclipse (supported by this subject)
- IntelliJ
- Netbeans

You can use any IDE of your choice

Quick Eclipse Demo

- Create a new Eclipse project
- Add a JAR file (internal / external)
- Build an executable jar file

Create a new Eclipse Project

http://www.tutorialspoint.com/eclipse/eclipse_create_java_project.htm

Build an executable jarfile

- Export a Jar
- Execute Program using Jar

Add a JAR file (internal / external)

Command Line Arguments

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 02

Today’s Agenda

- Questions/ discussion on Sockets, UDP, TCP
- Code demonstration
    - Sockets /client server – UDP and TCP

Sockets

Q1. Briefly discuss three aspects of the Socket interface.

Sockets

Q1. Briefly discuss three aspects of the Socket interface.

- Bound to a local port.
  - Socket address = IP + Port number
- Sockets are used for communication between two networked processes.
  - sending and receiving data
- Each socket is associated with a protocol (UDP or TCP).

- Acts as programming interface to application code and transport layer
- Socket handle is mostly like file handle

Q2. Briefly explain three possible failures that can happen when using UDP for communication.

Q2. Briefly explain three possible failures that can happen when using UDP for communication.

- Data Corruption.
- Omission failures (No guaranteed delivery).
- Order.

Q3. Briefly explain aspects of TCP that address issues not addressed by UDP.

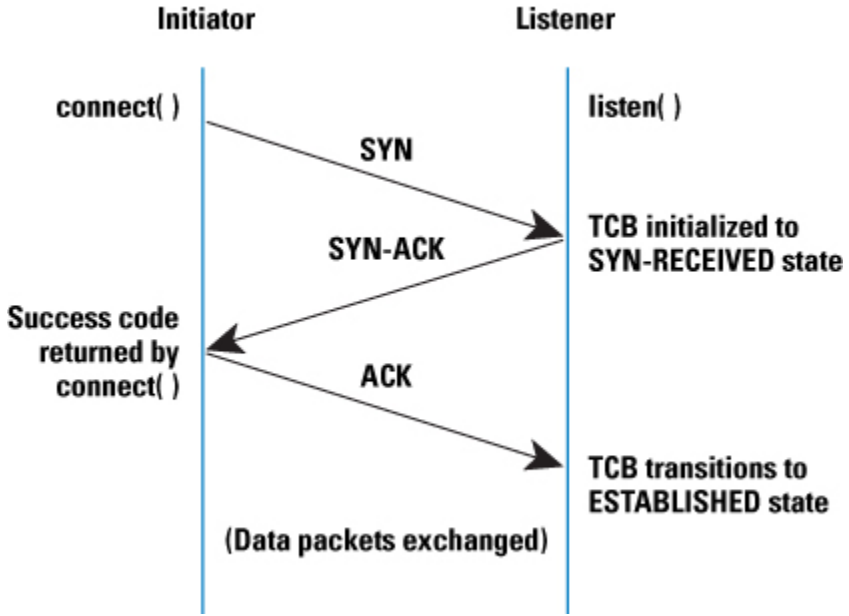
Q3. Briefly explain aspects of TCP that address issues not addressed by UDP.

- **Connection oriented:** The communicating processes establish a connection before communicating. The connection involves a connect request from the client to the server followed by an accept request from the server to the client.

- **Message sizes:** There is no limit on data size applications can use.
- **Lost messages:** TCP uses an acknowledgment scheme unlike UDP. If acknowledgments are not received the messages are retransmitted.
- **Flow control:** TCP protocol attempts to match the speed of the process that reads the message and writes to the stream.
- **Message duplication or ordering:** Message identifiers are associated with IP packets to enable the recipient to detect and reject duplicates and reorder messages in case messages arrive out of order.

Q4. List the steps involved at the client and at the server to establish a TCP stream socket connection.

Q4. List the steps involved at the client and at the server to establish a TCP stream socket connection.



UDP vs TCP

UDP vs TCP

UDP: User Datagram Protocol

- Provides a message passing abstraction.
- Is the simplest form of connectionless Interprocess Communication (IPC).
- Transmits a single message (called as datagram) to the receiving process.
- Example use cases- Video stream (real time applications), DNS, NTP (query response)

TCP: Transmission Control Protocol

- Provides an abstraction for a two-way stream (called as packets).
- Streams do not have message boundaries.
- Stream provide the basis for producer/consumer communication.
- Data sent by the producer are queued until the consumer is ready to receive them.
- The consumer must wait when no data is available.
- Example use cases – http, ftp

Client - Server Demo

1. UDP
    - Client and Server
2. TCP
    - Interactive Client and Server

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 03

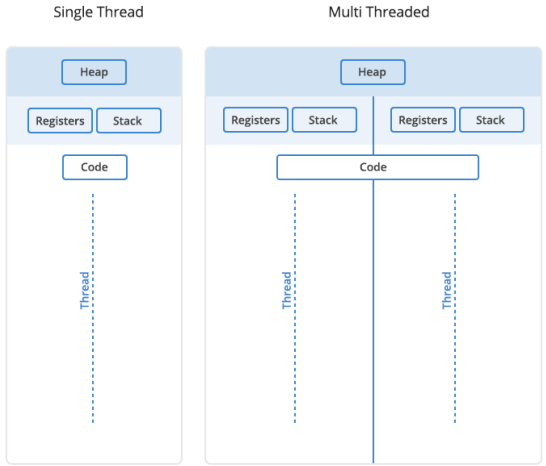
Today’s Agenda

- Quickly go through the thread slides
- Concept/question discussion
    1. What is a thread and life cycle of a thread
    2. Synchronous access to shared resources
    3. Comparison of worker pool multi-threading architecture with the thread-per-request architecture
- Code demonstration of thread Sleep, Join and Synchronization and Multithreaded Server and Client

Q1. What is Thread?

Q1. What is Thread ?

- 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.

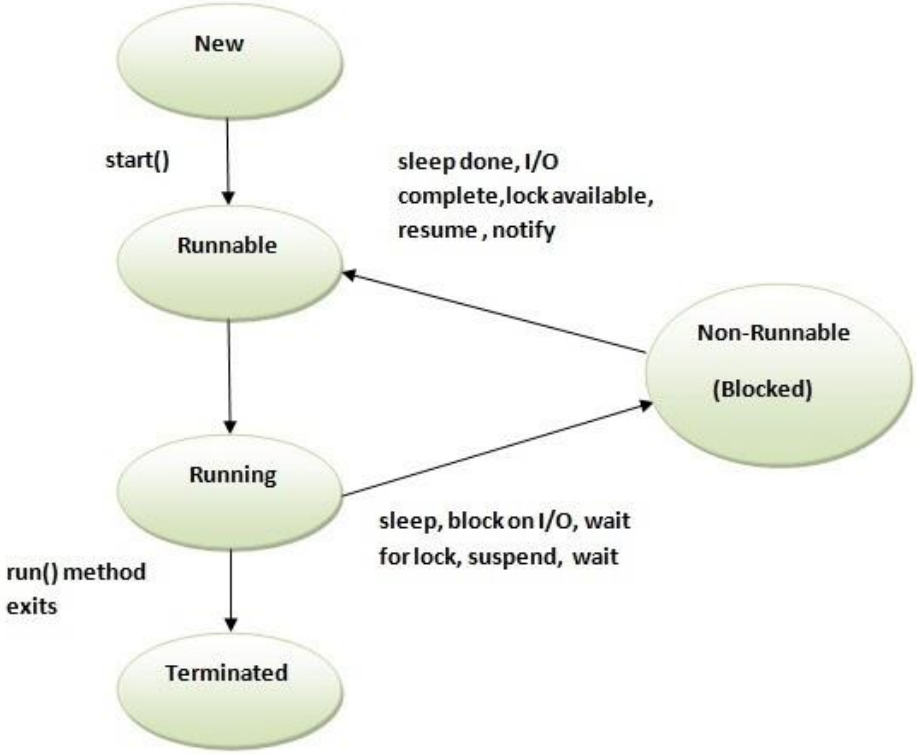


Thread vs Process

- Advantages of thread based parallelism
  - Threads share the same address space.
  - Context-switching between threads is normally inexpensive.
  - Communication between threads is normally inexpensive.

Q3. What is life cycle of a thread?

Thread Lifecycle



Q4. What do you mean by synchronous access to shared resources and how can we achieve it?

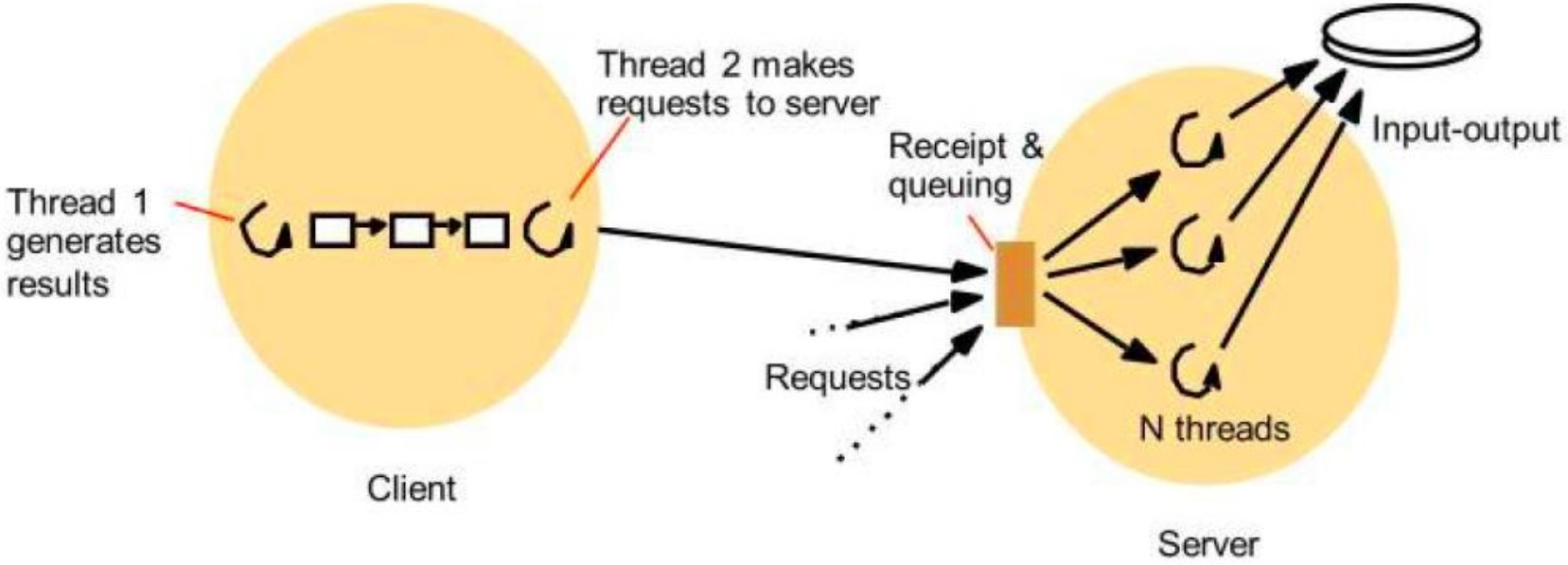
Q4. What do you mean by synchronous access to shared resources and how can we achieve it?

- If one thread tries to read the data and other thread tries to update the same data, it leads to inconsistent state.
- This can be prevented by synchronising access to the data.
- Use “synchronized” to methods or objects:

```
public synchronized void update()
{
                    …
}
```

Q5. Compare the worker pool multi-threading architecture with the thread-per-request architecture.

Worker pool architecture



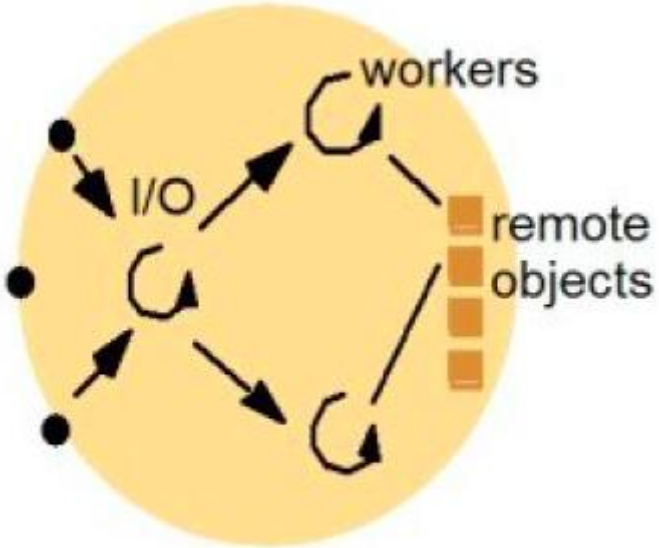
Worker pool architecture

The server creates a fixed number of threads called a worker pool. As requests arrive at the server, they are put into a queue by the I/O thread and from there assigned to the next available worker thread.

Server creates worker pool ➜ request comes in, put into a queue ➜ assigned to an available worker thread.

Useful in highly concurrent system

Thread-per-request



a. Thread-per-request

Thread-per-request architecture

Thread created for each request, when the request is finished, the thread is deallocated.

Code Demonstration

- Multi-threading in Java – Synchronization
  - In java, multithreading can be implemented in two ways
    - Extending Thread class
    - Implementing Runnable interface
- Multithreaded Server and client with Sockets

End of Tutorial

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 05

Today’s agenda

- Discussion / Q & A about Assignment 1
- Inter Process Communication (IPC) data formats
- Code Demonstration : JSON format
- Glance at a research paper

Assignment 1 Q & A

External Data Representation and Marshalling

- Data structures in programs are flattened to a sequence of bytes before transmission
- Different computers have different data representations- e.g., number of bytes for an integer, floating point representation, ASCII vs Unicode. Two ways to enable computers to interpret data in different formats:
  - Data is converted to an agreed external format before transmission and converted to the local form on receipt
  - Values transmitted in the senders format, with an indication of the format used
- **Marshalling:** Process of converting the data to the form suitable for transmission
- **Unmarshalling:** Process of disassembling the data at the receiver
- **External data representation:** Agreed standard for representing data structures and primitive data

Extensible Markup Language (XML)

A **markup** language is a textual encoding representing data and the details of the structure (or appearance)

XML is:

- a markup language defined by World Wide Web Consortium (W3C)
- tags describe the logical structure of the data
- is extensible - additional tags can be defined
- tags are generic - unlike HTML where tags give display instructions
- self describing - tags describe the data
- tags together with namespaces allow the tags to be meaningful
- since data is textual, it can be read by humans and platform independent
- since data is textual the messages are large causing longer processing and transmission times and more space to store

XML Elements and Attributes

Element

- consists of data surrounded by tags - e.g. <name>Smith</name>
- elements can be enclosed within elements - e.g. elements with the tag “name” is enclosed within the elements with tag “person”. This allows hierarchical representation.

Attributes

- a start tag may optionally contain attributes (names and values) - e.g. id="12345678"

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

JavaScript Object Notation (JSON)

JSON is a lightweight data-interchange format.

JSON is a syntax for storing and exchanging data.

JSON is an easier-to-use alternative for XML

It is based on the subset of JavaScript Programming Language

It is text based and completely language independent

IPC Data formats– JSON vs XML

- JSON is lightweight thus simple to read and write.
- JSON supports array data structure.
- JSON files are more human readable.
- JSON has no display capabilities .
- Provides scalar data types and the ability to express structured data through arrays and objects.
- Native object support. Similarities between JSON

- XML is less simple than JSON.
- XML doesn't support array data structure.
- XML files are less human readable.
- XML provides the capability to display data because it is a markup language.
- Does not provide any notion of data types. One must rely on XML Schema for adding type information.
- Objects have to be expressed by conventions, often through a mixed use of attributes and elements.

IPC Data formats– JSON vs XML

JSON serialization produces shorter strings than XML, reducing the amount of data transmission and improved performance

JSON:

```
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML:

```
<employees>
    <employee>
        <firstName>John</firstName> <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName> <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName> <lastName>Jones</lastName>
    </employee>
</employees>
```

Demo time!

- IPC with JSON

Glance at a research paper

- Understanding what it takes to build a good Ditsributed System
  - https://queue.acm.org/detail.cfm?id=2482856

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 06

Today’s Agenda

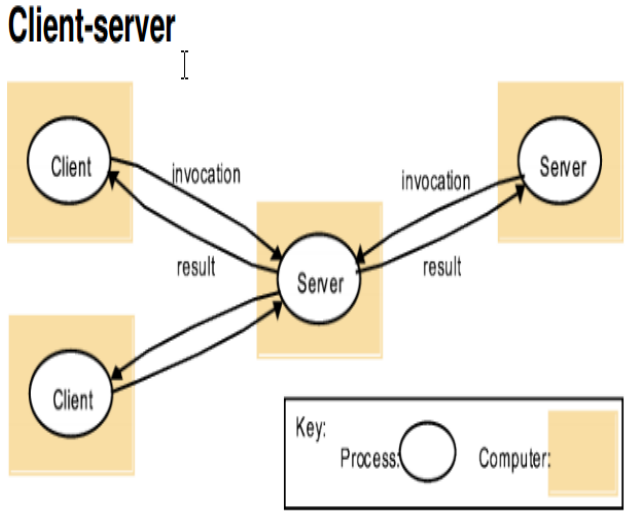
Discussion on open questions on Distributed Systems models

Operating Systems Support Questions

Q1. Briefly explain the difference between a client-server architecture and a peer-to-peer architecture.

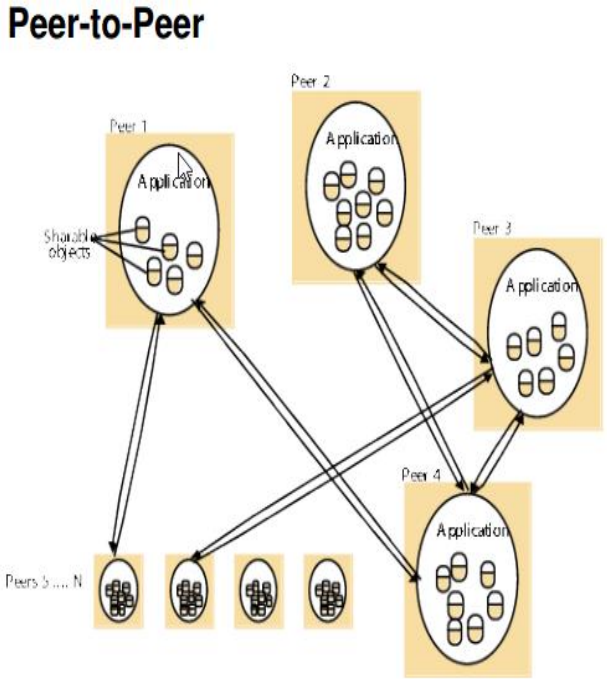
Client- Server

- A client requests some processing or information from a server that it needs.
- It waits in a blocking fashion for the reply containing the result,
- It then can proceed with it's execution
- There can be many variants of client server model



Peer-to-Peer

- Peer model suits ad-hoc groupings of participants
- No central point of failure (reliable)
- No central point of control (difficult to deny service for adversaries)
- Some peers will typically contribute more than others (I.e. seed or super-peer)
- Examples- Napster, BitTorrent



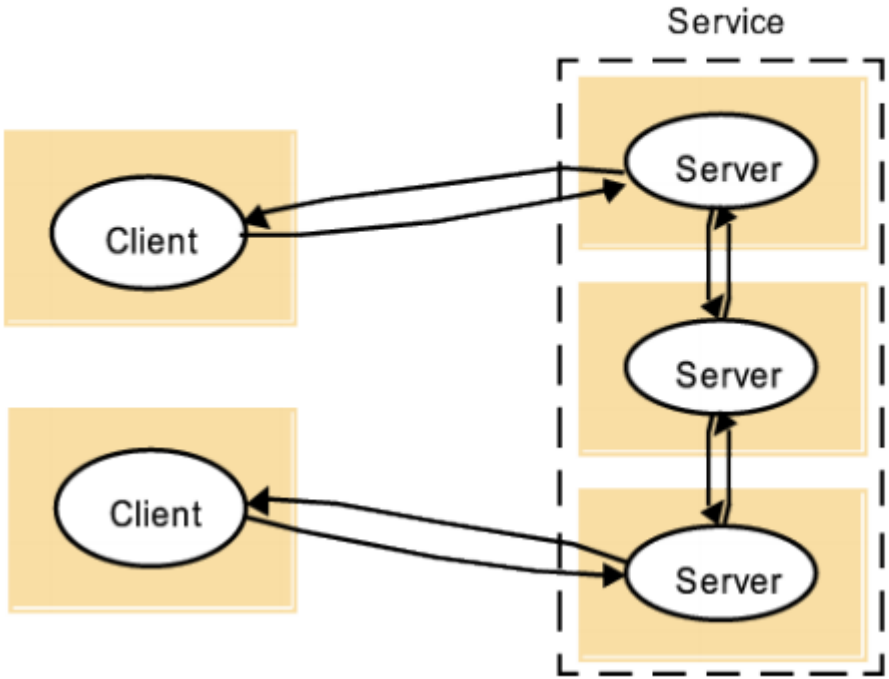
Q2. Briefly explain each of the following distributed system architecture variations, giving also a reason or a benefit of its use:

- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and Mobile Agents
- Network computers
- Thin clients
- Tiered Architecture

- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and Mobile Agents
- Network computers
- Thin clients
- Tiered Architecture

A service provided by multiple servers

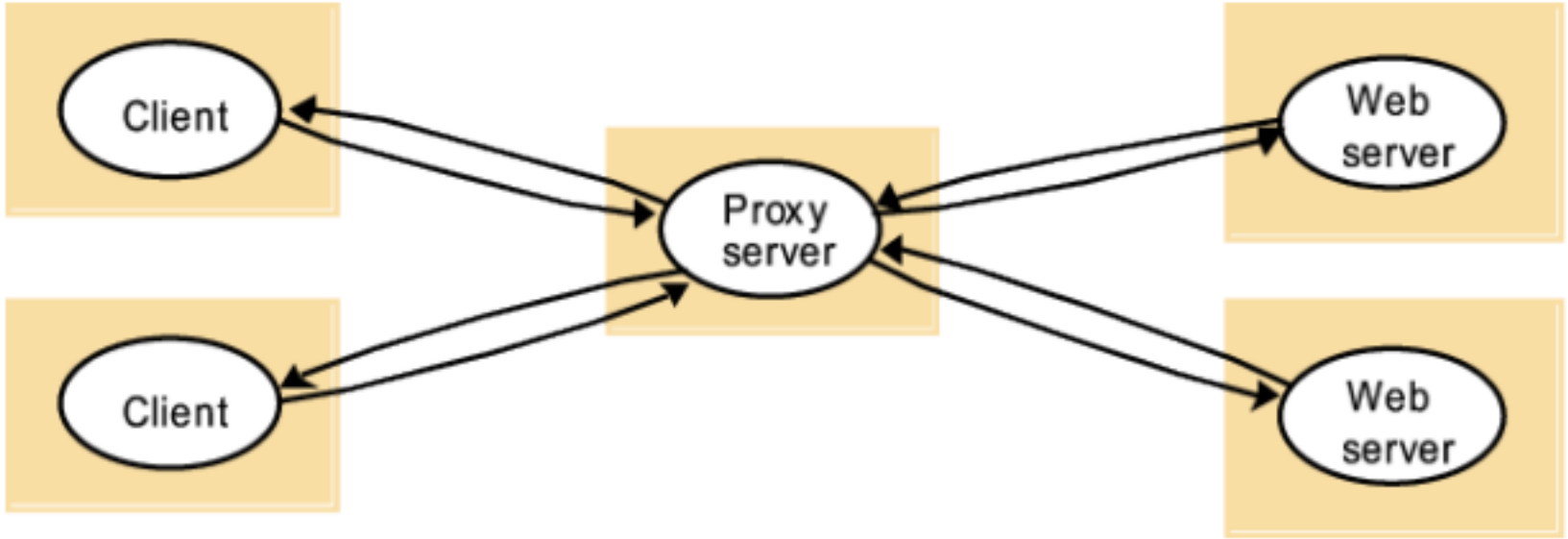
Service is provided by several server processes interacting with each other. Objects may be partitioned (e.g web servers) or replicated across servers (e.g. Sun Network Information Service (NIS)).



- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and Mobile Agents
- Network computers
- Thin clients
- Tiered Architecture

Proxy servers and caches

- Cache is a store of recently used objects that is closer to client
- New objects are added to the cache replacing existing objects
- When an object is requested, the caching service is checked to see if an up-to-date copy is available (fetched in not available)



- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and Mobile Agents
- Network computers
- Thin clients
- Tiered Architecture

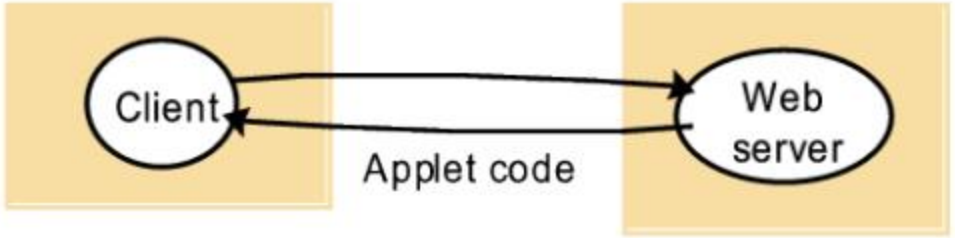
Mobile Code and Agents

Mobile Code is down loaded to the client and is executed on the client (e.g. applet).

Mobile agents are running programs that includes both code and data that travels from one computer to another.

E.g. Web Applets:

a) client request results in the downloading of applet code



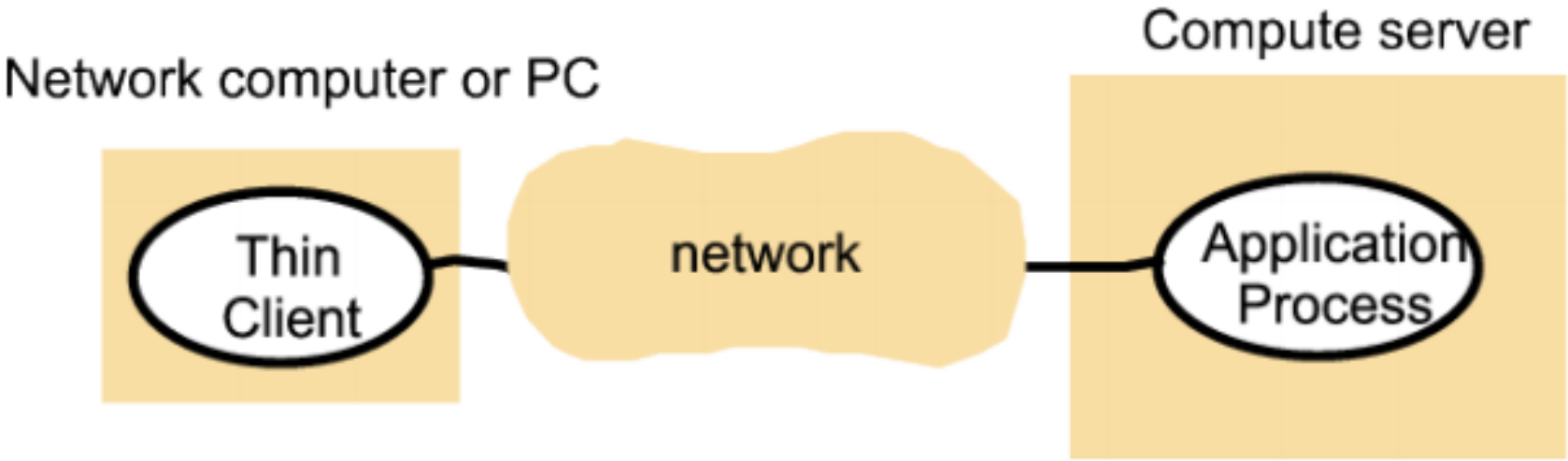
b) client interacts with the applet

- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and Mobile Agents
- Network computers
- Thin clients
- Tiered Architecture

Network Computers and Thin clients

- **Network Computers**: download their operating system and application software from a remote file system. Applications are run locally.
- **Thin Clients**: application software is not downloaded but runs on the computer server - e.g. UNIX.

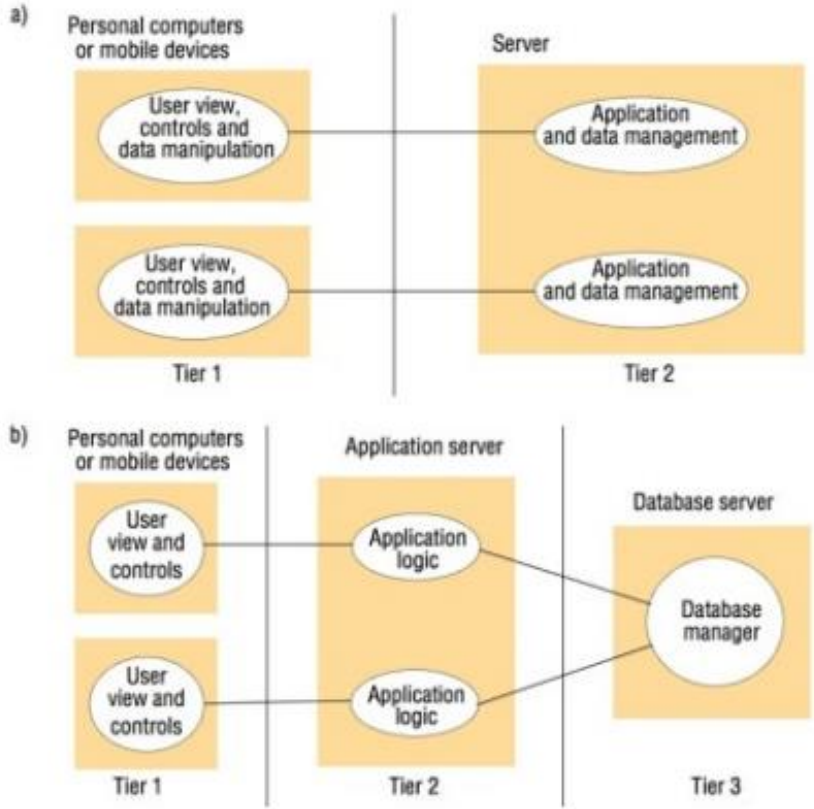
This paradigm is usually not suitable for highly interactive graphical activities.



- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and Mobile Agents
- Network computers
- Thin clients
- Tiered Architecture

Tiered architecture

Tiered architectures are complementary to layering. Layering deals with vertical organization of services.



Operating Systems Support Questions

1. Discuss the difference between a network operating system and a distributed operating system.

1. Discuss the difference between a network operating system and a distributed operating system.

- A *networked operating system* provides support for networking operations. The users are generally expected to make intelligent use of the network commands and operations that are provided. Each host **remains autonomous** in the sense that it can continue to operate when disconnected from the networking environment.

- A *distributed operating system* tries to abstract the network from the user and thereby remove the need for the user to specify how the networking commands and operations should be undertaken. This is sometimes referred to as providing a ***single system image***. Each host may not have everything that would be required to operate on its own, when disconnected from the network.

1. Discuss the difference between a network operating system and a distributed operating system.

Network operating system

- Users retain autonomy in managing their own processing resources
- It does not manage processes across the nodes
- Provides support for networking operations

1. Discuss the difference between a network operation system and a distributed operating system.

Distributed operating system

• Users are never concerned with where their programs run, or the location of any resources

• Has control over all the nodes in the system, and it transparently locates new processes at whatever node suits its scheduling policies

• Each host may not have everything that would be required to operate on its own

• Single system image

2. What are the core OS components?

2. What are the core OS components?

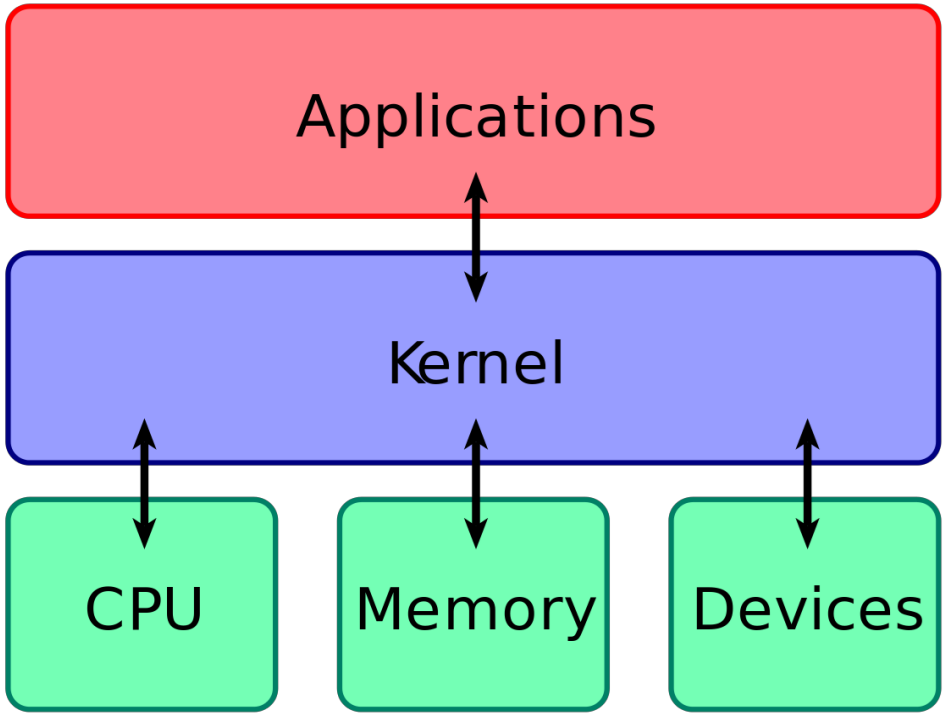
- **Process manager** -- Handles the creation of processes, which is a unit of resource management, encapsulating the basic resources of memory (address space) and processor time (threads).
- **Thread manager** -- Handles the creation, synchronization and scheduling of one or more threads for each process. Threads can be scheduled to receive processor time.
- **Communication manager** -- Handles interprocess communication, i.e. between threads from different processes. In some cases this can be across different hosts

2. What are the core OS components?

- **Memory manager** -- Handles the allocation and access to physical and virtual memory. Provides translation from virtual to physical memory and handles paging of memory.
- **Supervisor** -- Handles privileged operations, i.e. those that directly affect shared resources on the host, e.g. to and from an I/O device. The supervisor is responsible for ensuring that host continues to provide proper service to each client.

3. What is the kernel? and also discuss about popular kernel implementation methods

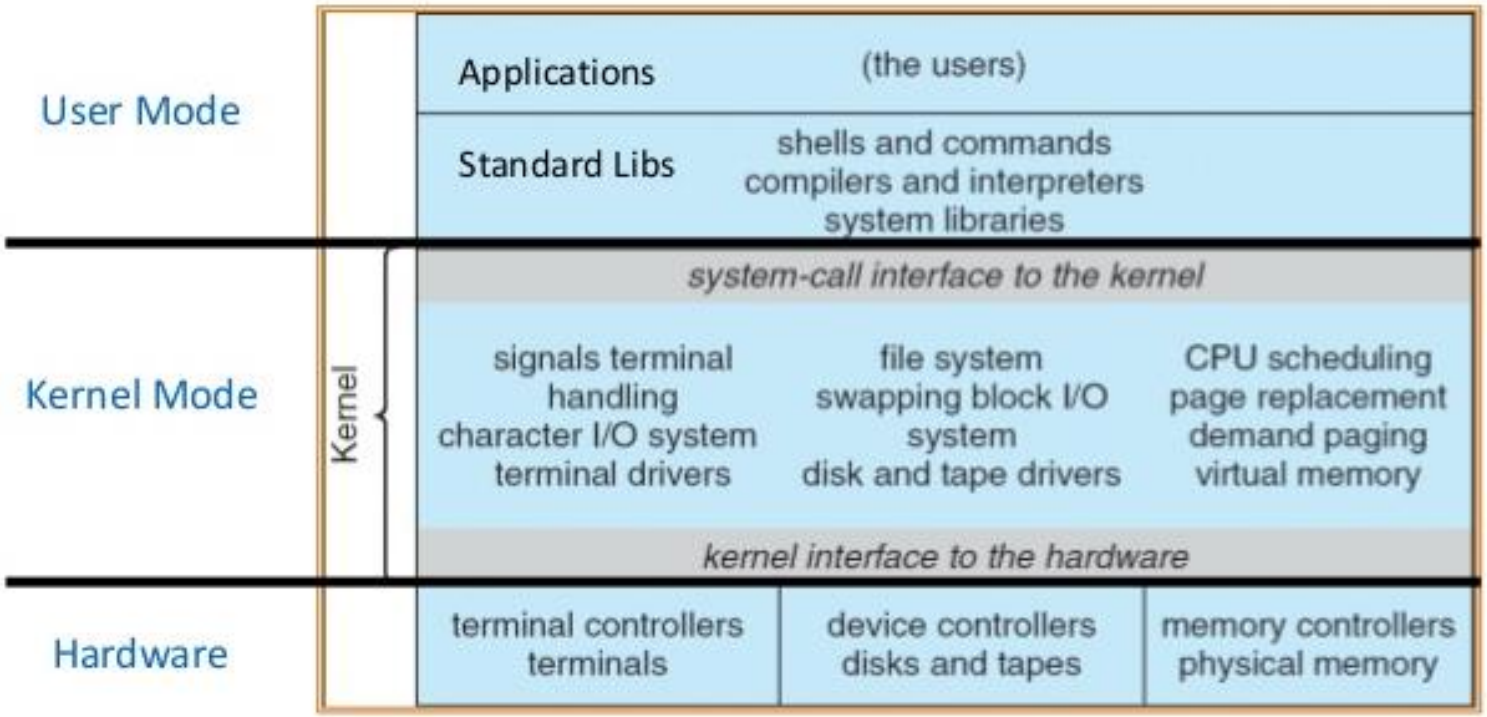
3. What is the kernel? and also discuss about popular kernel implementation methods



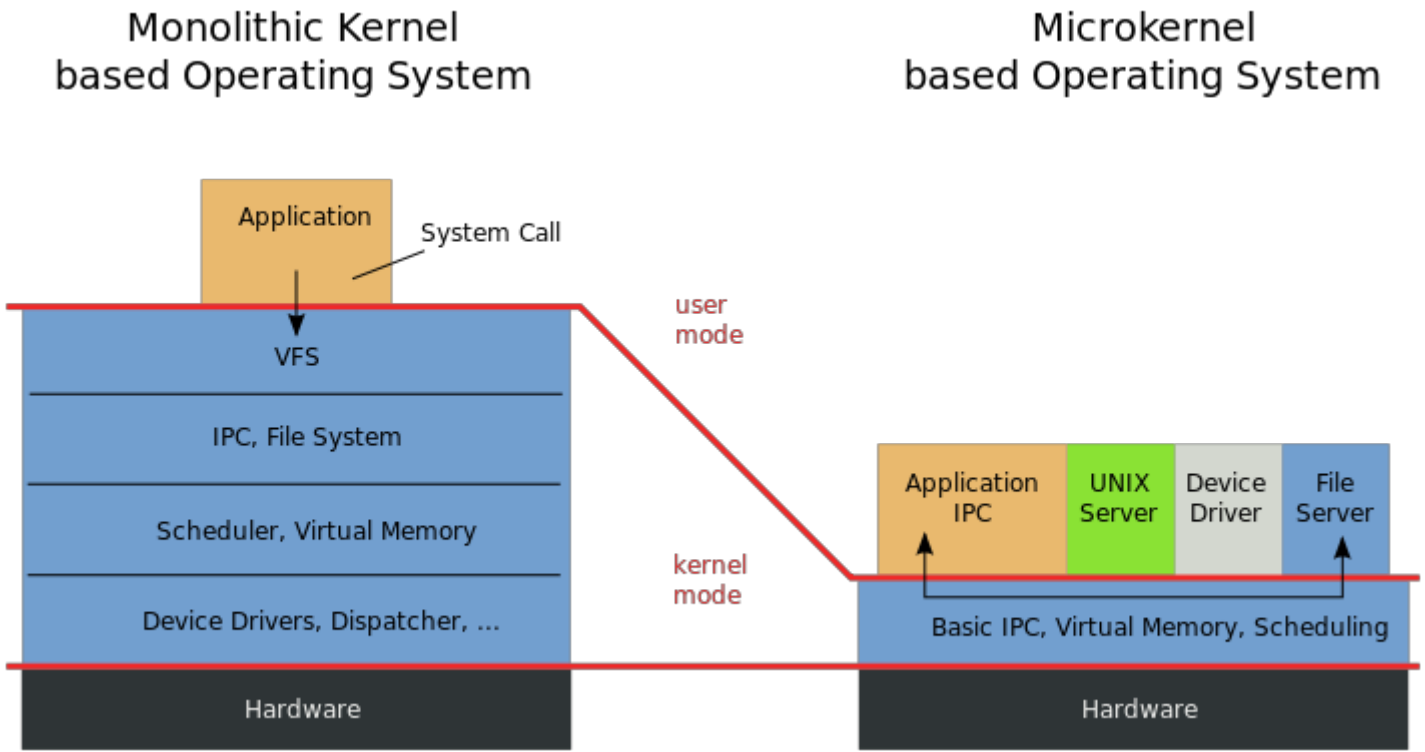
3. What is the kernel ? and also discuss about popular kernel implementation methods.

- Part of the Operating System
- Has full access to the host’s resources
- Kernel begins execution after the host is powered up and continues to execute while the host is operational
- The kernel has access to all resources and shares access to all other processes that executing on the host

A Real-World Example - UNIX



Comparison of Kernel Implementations



Disadvantages of Monolithic OS

- It is massive
  - codebase
- It is undifferentiated:
  - non-modular (traditionally), although modern ones are much more layered.
- It is intractable:
  - Altering any individual software component to adapt for new requirements is difficult.

4. What is Supervisor Mode vs User Mode?

4. What is Supervisor Mode vs User Mode?

- Operating modes supported by the hardware at the machine instruction level.
- **Supervisor / Kernel mode** -- instructions that execute while the processor is in supervisor (or privileged) mode are capable of accessing and controlling every resource on the host,
- **User mode** -- instructions that execute while the processor is in user (or unprivileged) mode are restricted, by the processor, to only those accesses defined or granted by the kernel.

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 07

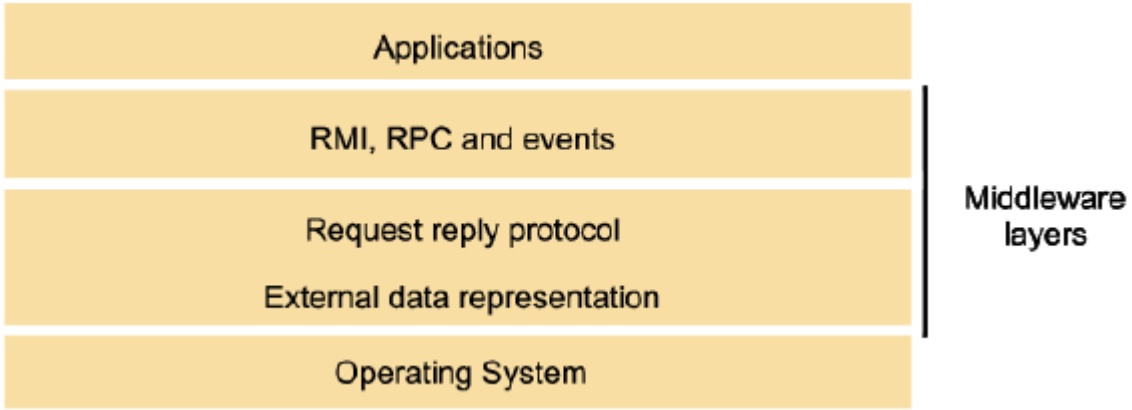
Today’s Agenda

1. Questions/Discussion on Remote Invocation, RPC and RMI
2. Code Demonstration : RMI Client Server

What is Remote Invocation?

Remote Invocation

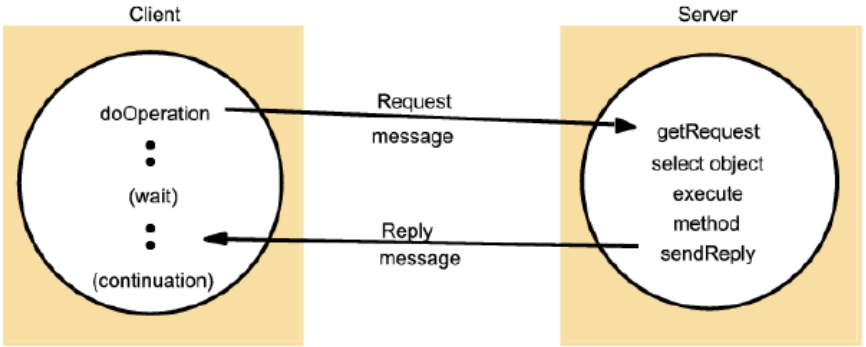
- A set of information exchange protocols at the Middleware layer



1. Can you explain three different types of protocols typically used to address the design issues of Remote Invocation?

1. Can you explain three different types of protocols typically used to address the design issues of Remote Invocation?

- **Request** : Client sends message
- **Request-Reply** : Client sends message, Server sends reply
- **Request-Reply-Acknowledgement** : Same as above, but once the client receives the message, the client will acknowledge it.



Possible Issues in Request-Reply

- What are the issues that may arise in a request-reply process?
  - Request timeouts
    - Retry request message after timeout
    - Do nothing
  - Reply timeouts
    - Retransmit results after timeout
    - Do nothing
  - Receiving duplicate requests/replies
    - Discard duplicate messages
    - Perform the same action again if the logic is idempotent

2. Explain different invocation semantics

2. Explain different invocation semantics

- Different issue handling strategies lead to different invocation semantics

- Invocation semantics
  - Define what the client can assume about the execution of the remote procedure
  - Offer different reliability guarantees in terms of the number of times that the remote procedure is executed

2. Explain different invocation semantics

- **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. **Suitable for idempotent** operations.
- **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. **Suitable for non-idempotent operations.**

Figure 5.5
Invocation semantics

Fault tolerance measures			*Invocation semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

3. Please explain Remote procedure call (RPC) and describe its key components.

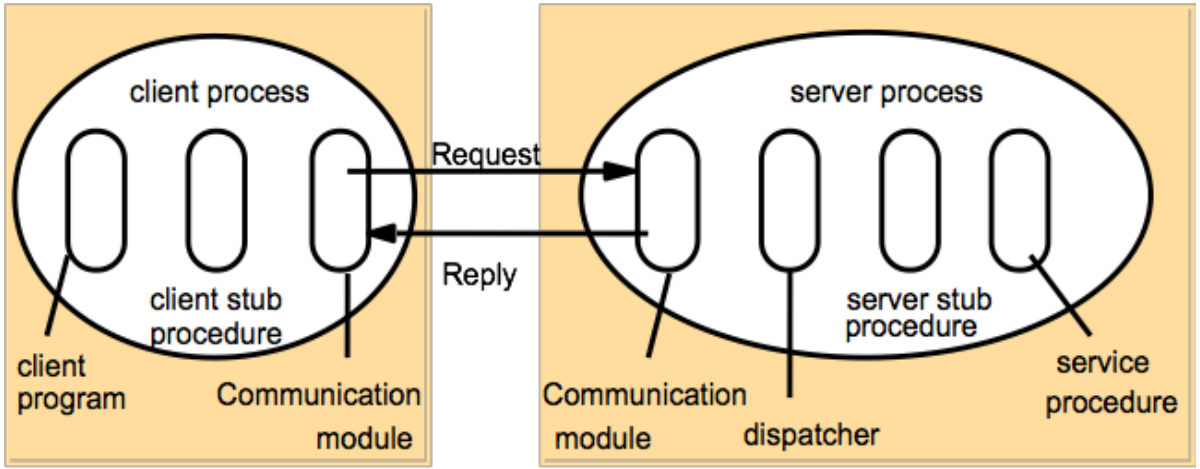
3. Remote Procedure Call

- Please explain Remote procedure call (RPC) and describe its key components.
    - RPCs enable clients to execute procedures in server (remote) processes based on a defined service interface.
    - Used in procedural languages such as Fortran, C, and GO.

3. Remote Procedure Call (RPC) and key components

Key components of RPC:

- Communication Module
- Client Stub Procedure
- Dispatcher
- Server stub procedure

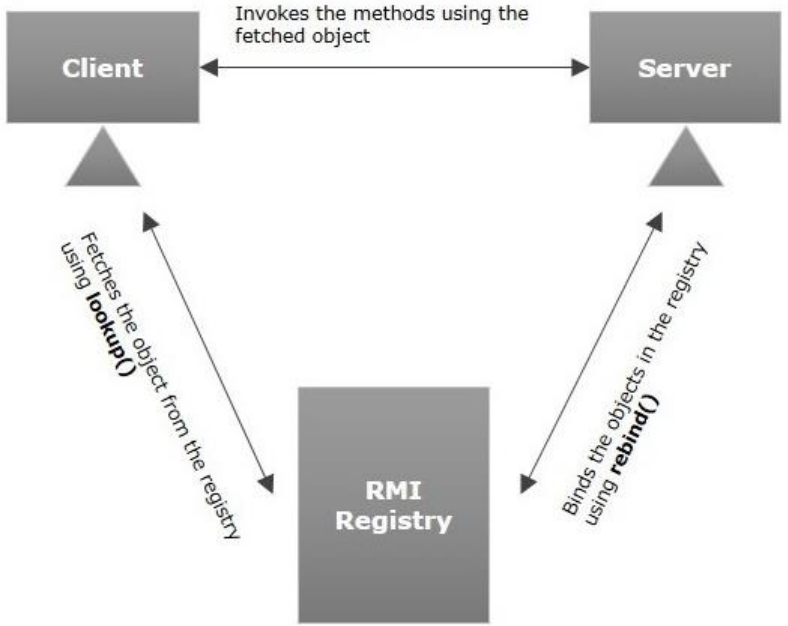


3. Remote procedure call (RPC) and key components3.

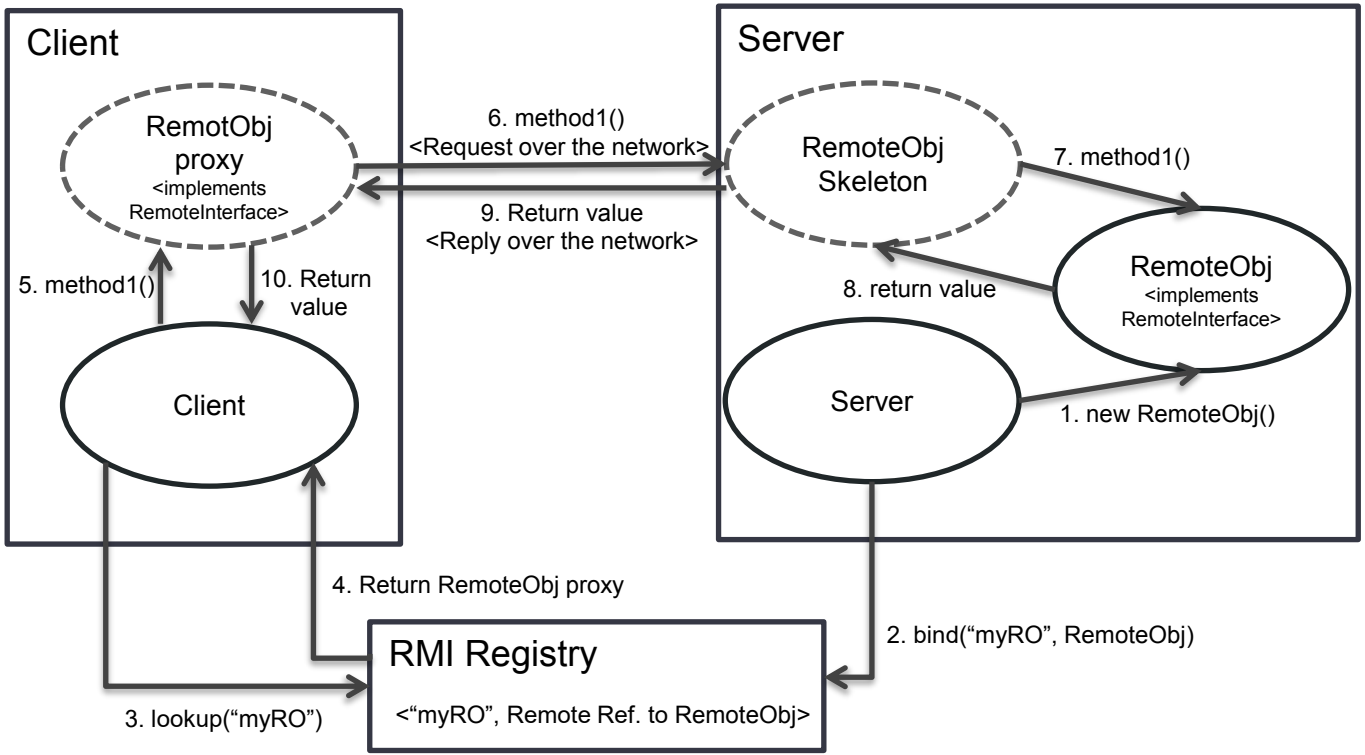
- **Communication Module**
  Implements the desired design choices in terms of retransmission of requests, dealing with duplicates and retransmission of results
- **Client Stub Procedure**
  Behaves like a local procedure to the client. Marshals the procedure identifiers and arguments which is handed to the communication module
  Unmarshalls the results in the reply
- **Dispatcher**
  Selects the server stub based on the procedure identifier and forwards the request to the server stub
- **Server Stub Procedure**
  Unmarshalls the arguments in the request message and forwards it to the Service Procedure. Marshalls the arguments in the result message and returns it to the client

4. Explain how one goes about using Java RMI to build a distributed system. Discuss what needs to be compiled and important aspects of the system that must be used. Actual code is not required.

Java RMI


Client
Invokes the methods using the fetched object
Server
Fetches the object from the registry using **lookup()**
RMI Registry
Binds the objects in the registry using **rebind()**

Java RMI Overview



4. Explain how one goes about using Java RMI to build a distributed system. Discuss what needs to be compiled and important aspects of the system that must be used. Actual code is not required.

- *Defining the interface for remote objects* - Interface is defined using the interface definition mechanism supported by the particular RMI software.
- *Compiling the interface* - Compiling the interface generates proxy, dispatcher and skeleton classes.
- *Writing the server program* - The remote object classes are implemented and compiled with the classes for the dispatchers and skeletons. The server is also responsible for creating and initializing the objects and registering with the binder.
- *Writing client programs* - Client programs implement invoking code and contain proxies for all remote classes. Uses a binder to lookup for remote objects.

Demo Time!

RMI Demo

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 08

Today’s Agenda

- Assignment 2 Q & A
- Questions on Security (partial)
- Demo on File Transfer
- Demo on Git

Assignment 2 Q & A

1. What are the different security threats and methods of attacks in a distributed system?

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

2. List and briefly explain some worst case assumptions when designing a secure system.

2. List and briefly explain some worst case assumptions when designing a secure system.

- **Networks are insecure.**
  - Messages can be looked at, copied, modified and retransmitted,
  - Attackers can obtain information that they should not and can pretend to be a legitimate party.
- **The source code is known to the attacker.**
  - Knowing the source code can help the attacker discover vulnerabilities.
- **Interfaces are exposed**
  - Communication interfaces are necessarily open to allow clients to access them.
  - Attackers can send messages to any interface.
- **The attacker has unlimited computing resources.**
  - Assume that attackers will have access to the largest and most powerful computers projected in the lifetime of a system.

3. Define encryption and describe the two main types of keys used by encryption algorithms.

3. Define encryption and describe the two main types of keys used by encryption algorithms.

- ***Encryption***
  - process of encoding a message in such a way as to hide its contents.
- ***Shared secret keys*** (symmetric)
  - Sender and recipient share knowledge of the key and it must not be revealed to anyone else.
- ***Public/private key pairs*** (asymmetric)
  - The sender uses a public key to encrypt the message.
  - The recipient uses a corresponding private key to decrypt the message.
  - Only the recipient can decrypt the message, because they have the private key.
  - Typically require 100 to 1000 times as much processing power as secret-key algorithms.

Code Demo

- Sending/Receiving files with Java Sockets

Demo : Git

For more details: EGitTutorial.pdf

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 09

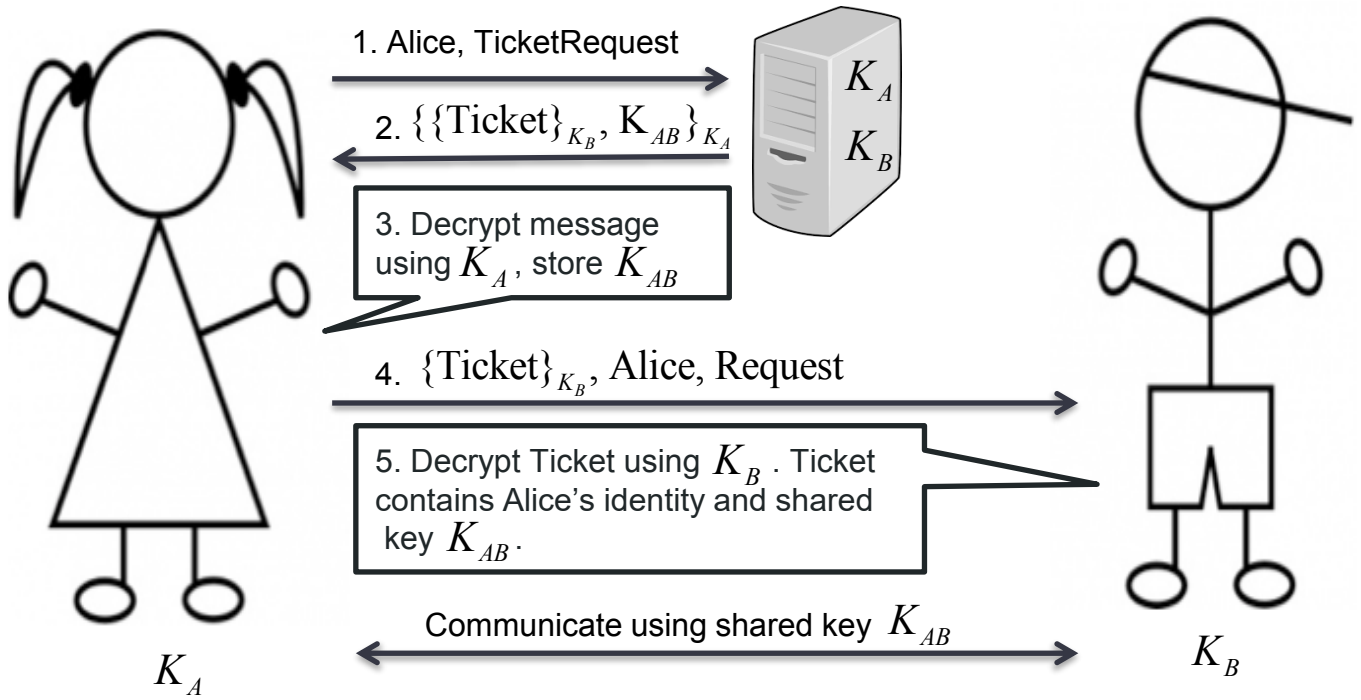
Today’s Agenda

- Assignment 2 Q & A
- Questions on Security (continued)
- Demo - Client Server Encryption

Assignment 2 Q & A

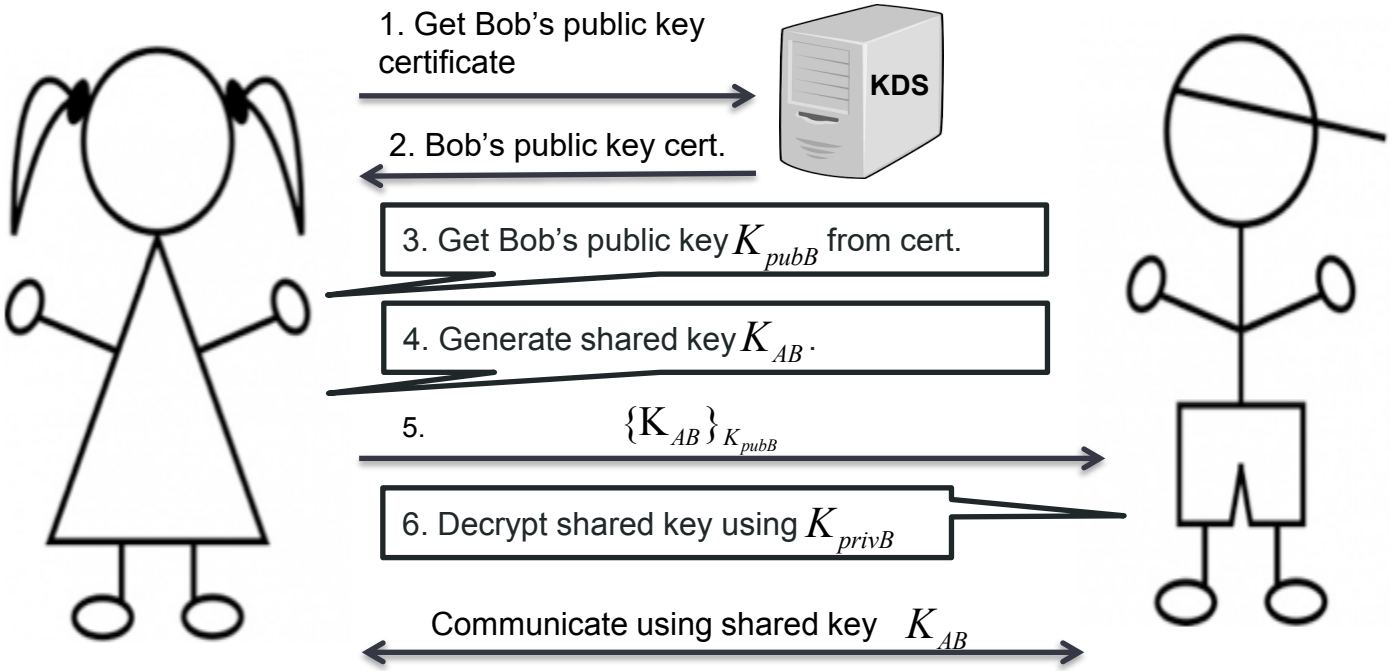
1. How can Alice authenticate and communicate secretly with Bob assuming there is an authentication server that knows Alice’s and Bob’s secret keys?

1. How can Alice authenticate and communicate secretly with Bob assuming there is an authentication server that knows Alice's and Bob's secret keys?



2. Assuming that Bob has a public/private key pair, how can Alice and Bob establish a shared key to communicate secretly using a Key Distribution Service?

2. Assuming that Bob has a public/private key pair, how can Alice and Bob establish a shared key to communicate secretly using a Key Distribution Service?

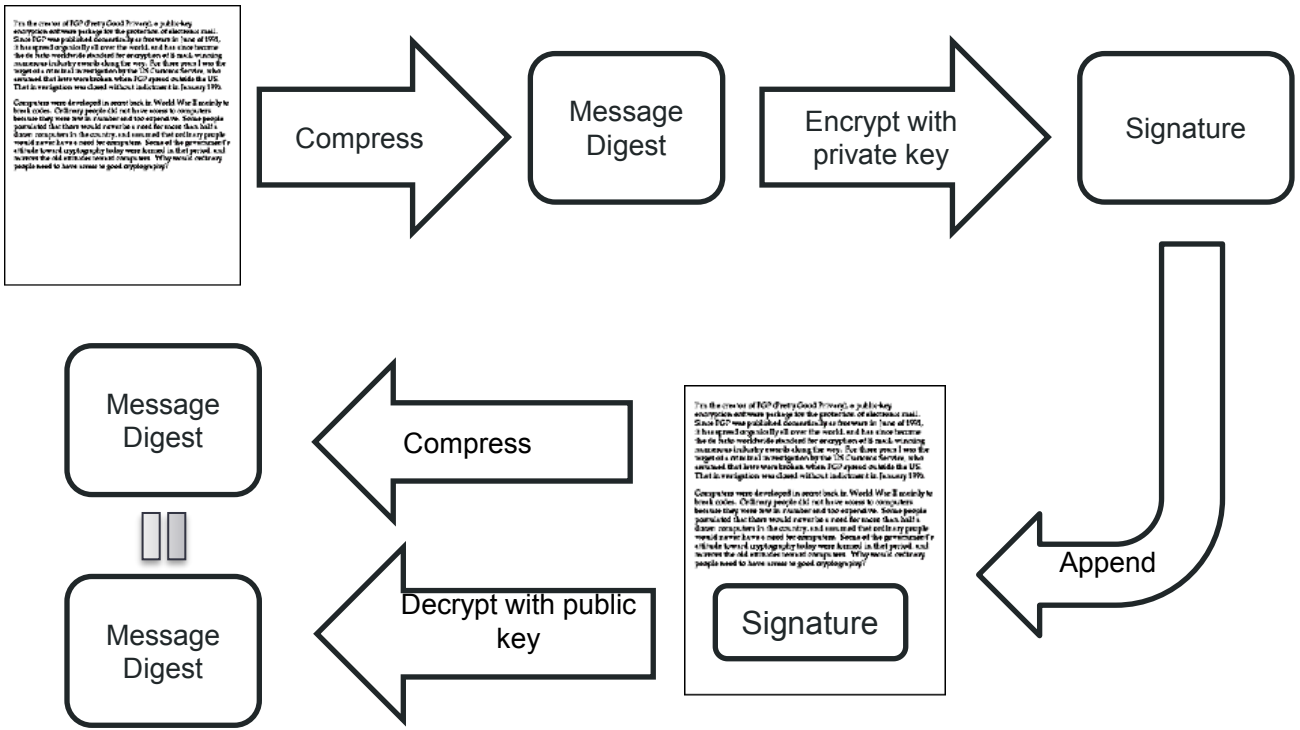


3. Explain how digital signatures work.

3. Explain how digital signatures work.

1. To sign a document, Bob first ‘compresses’ the message into just a few lines. This is called a *digest*.
2. Bob then encrypts the message digest with his private key. The result is the digital signature.
3. Bob appends the digital signature to the document. All of the data that was ‘compressed’ into the digest has been signed.
4. Bob sends the document to Alice.
5. Alice decrypts the signature (using Bob's public key) changing it back into a message digest.
6. Alice ‘compresses’ the document data into a message digest. If the message digest is the same as the message digest created when the signature was decrypted, then Alice knows that the signed data has not been changed and that Bob signed the document (because only Bob has his private key)

3. Explain how digital signatures work.



4. What is a digital certificate and why do we need it?

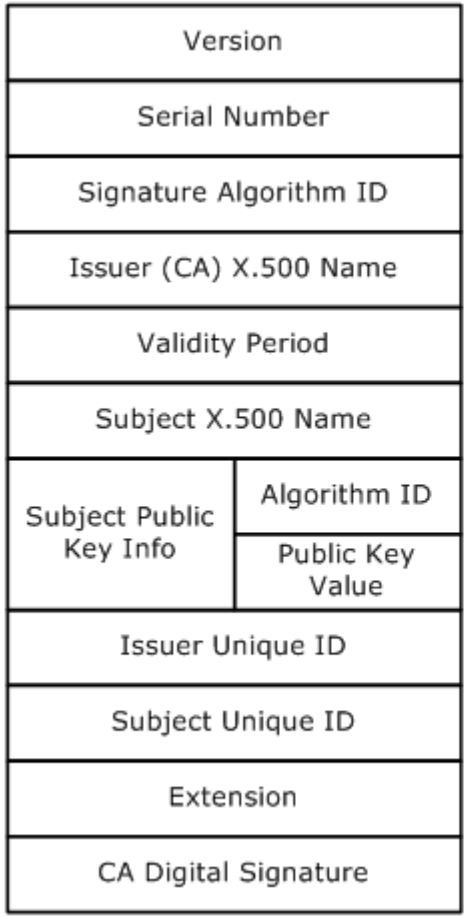
4. What is a digital certificate and why do we need it?

- A digital certificate is a digital form of identification, like a passport.
- A digital certificate provides information about the identity of an entity.
- A digital certificate is issued by a Certification Authority (CA).
    - Examples of trusted CA across the world are Verisign, Entrust, etc.
    - The CA guarantees the validity of the information in the certificate.

- The issue of distributing Public Key is massive, because the Public Key should be distributed in a scalable and truthful way

Public Key Infrastructure (PKI)

- **Public Key Infrastructure (PKI)** consists of protocols, standards and services, that allows users to authenticate each other using digital certificates that are issued by CA. For a digital certificate to be useful, it has to be structured in a standard way so that information within the certificate can be retrieved and understood regardless of who issued the certificate. The ***X.509, PKI X.509*** and ***Public Key Cryptography Standards (PKCS)*** are the building blocks a PKI system that defines the standard formats for certificates and their use.



Version of X.509 to which the Certificate conforms

A number that uniquely identifies the Certificate

The names of the specific Public Key algorithms that the CA has used to sign the Certificate (Ex.- RSA with SHA-1)

The identity of the CA Server who issued the Certificate

The period of time for which the Certificate is valid with start date and expiration date

The owner's identity with X.500 Directory format (Ex.- cn=auser, ou=SP, o=Alphawest)

The Public Key of the owner of the Certificate and the specific Public Key algorithms associated with the Public Key

Information used to identify the issuer of the Certificate

Information used to identify the Owner of the Certificate

Additional information like Alternate name, CRL Distribution Point (CDP)

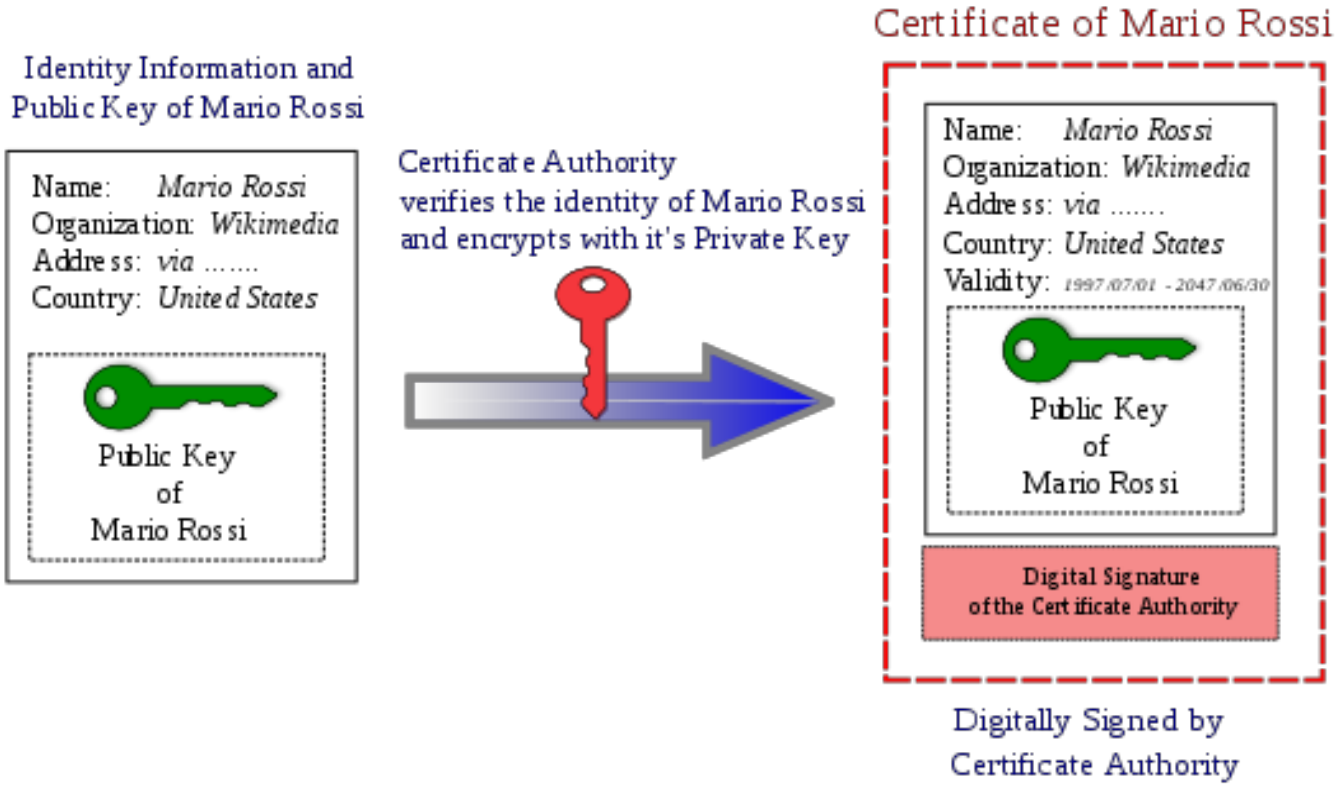
The actual digital signature of the CA

Certificates

Certificate type: Public key
Name: Bob
Public key: kBpub
Certifying authority: Sara
Signature: {Digest(field 2+field 3)}_{kSpriv}

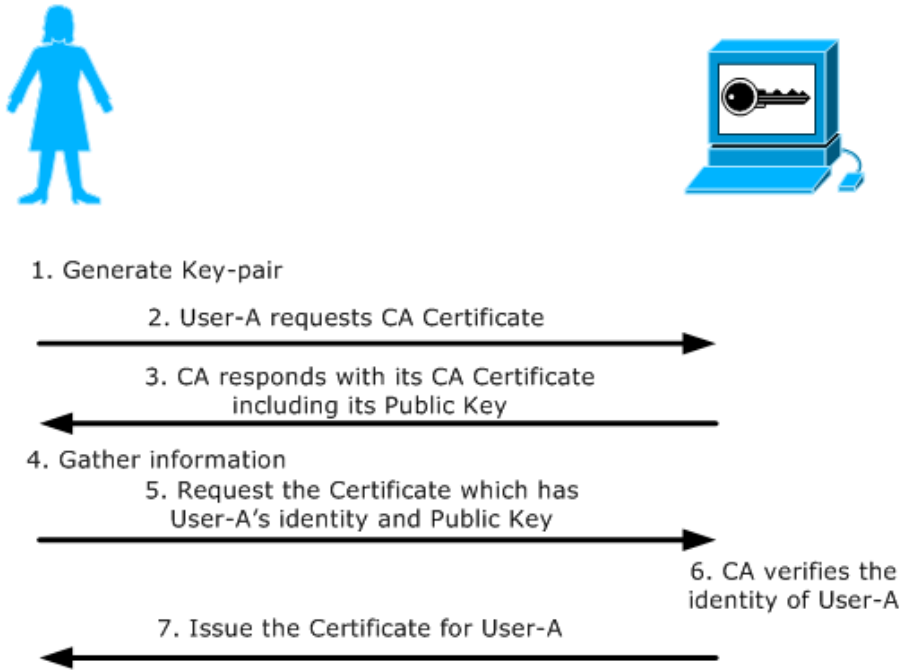
- In your own words, what is this certificate saying?
  - Sara certifies that Bob's public key is kBpub
- Why can't Sara deny that she has attested to this fact?
  - Because if someone can decrypt the signature using kSpub, only someone who had kSpriv could have encrypted it.
- What must be known to anyone who wants to make sure the certificate is authentic?
  - kSpub

Public Key Certificate



5. What is the process to obtain a digital certificate?

5. What is the process to obtain a digital certificate?



1. Generate Key-pair: User-A generates a Public and Private key-pair or is assigned a key-pair by some authority in their organization.

2. Request CA Certificate: User-A first requests the certificate of the CA Server.

3. CA Certificate Issued: The CA responds with its Certificate. This includes its Public Key and its Digital Signature signed using its Private Key.

4. Gather Information: User-A gathers all information required by the CA Server to obtain its certificate. This information could include User-A email address, fingerprints, etc. that the CA needs to be certain that User-A claims to be who she is.

5. Send Certificate Request: User-A sends a certificate request to the CA consisting of her Public Key and additional information. The certificate request is signed by CA's Public Key.

6. CA verifies User-A: The CA gets the certificate request, verifies User-A's identity and generates a certificate for User-A, binding her identity and her Public Key. The signature of CA verifies the authenticity of the Certificate.

7. CA issues the Certificate: The CA issues the certificate to User-A.

Code Demo

- Client Server Encryption with AES (Shared Secret Key)

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 10

Today’s Agenda

- Distributed File System (DFS) questions (Overview)
- Case Study: Drop Box

1. Name and explain transparencies that should be addressed by distributed file systems.

1. Name and explain transparencies that should be addressed by distributed file systems.

Access transparency

• Client programs don’t know if the file is local of remote

Location transparency

• Client programs don’t know where the file is stored

• Files can be relocated without changing their pathname

Mobility transparency

• Neither client programs nor system administration tables in client nodes need to be changed when files are moved

1. Name and explain transparencies that should be addressed by distributed file systems.

Performance transparency

• Maintain acceptable performance while the load on the service varies within a specified range

Scaling transparency

• Service can be expanded without loss of performance

2. What are the advantages and disadvantages of using absolute names as a naming strategy?

2. What are the advantages and disadvantages of using absolute names as a naming strategy?

- **Absolute name**
  - provides a complete address to a file including both the server and path names: <machine name: path name>
- **Advantages**
  - Trivial to find a file once the name is given
  - No additional state must be kept since each name is self contained (No global state)
  - Greater scalability
  - Easy to add and delete new names
- **Disadvantages**
  - No location transparency
  - File is location dependent and cannot be moved
  - Less resilient to failure

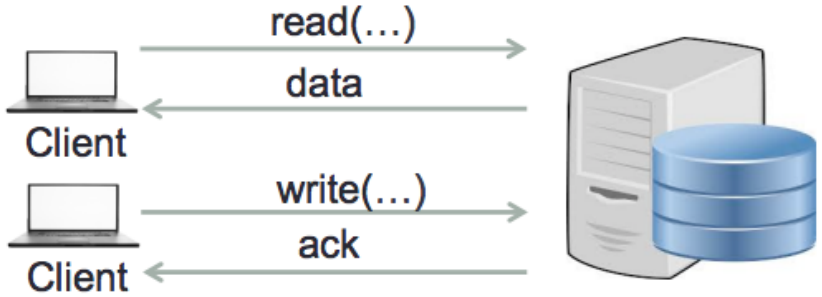
3. What are the advantages and disadvantages of a naming strategy based on mount points?

3. What are the advantages and disadvantages of a naming strategy based on mount points?

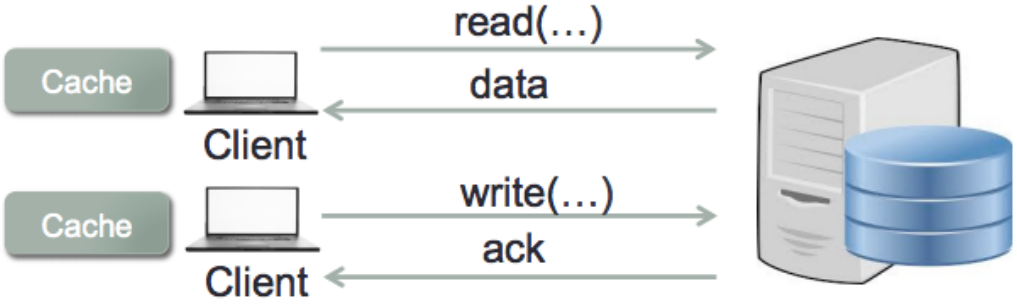
- **Mount Points (used by Sun's Network File System - NFS)**
- The client machine creates a set of "local names" which are used to refer to remote locations: mount points
- At boot time, the local name is bound to the remote name
- The operating system must maintain a table to maintain the mapping of what server and path are mapped to each mount point
- **Advantages:**
- Names do not contain information about the file location
- Remote location can change between reboots
- **Disadvantages:**
- Hard to maintain
    - What happens when machines fail?
    - What happens when files are migrated?
- Can lead to confusion since two different local names may map to the same file on a remote system

4. What are the advantages and disadvantages of the following two types of distributed file systems?

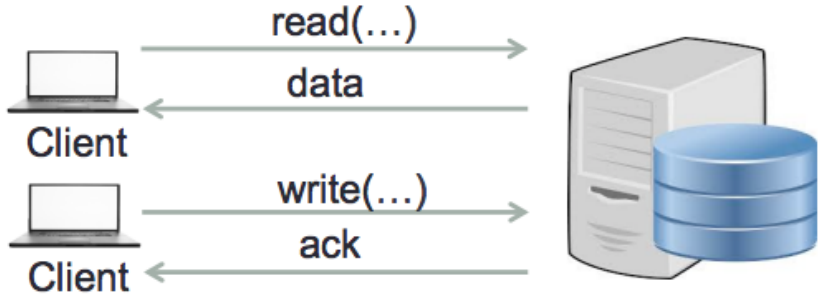
(A)



(B)

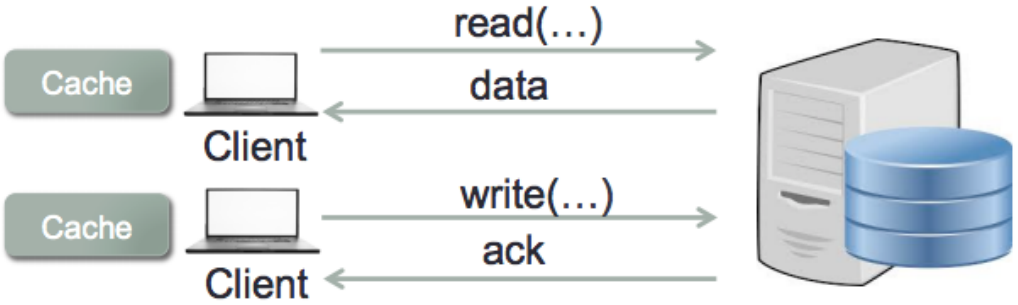


(A)



- RPC to access file system calls
- No client/local caching
- Advantages
- Consistent view of file system
- Disadvantages
- Performance (network access, server becomes potential bottleneck)

(B)

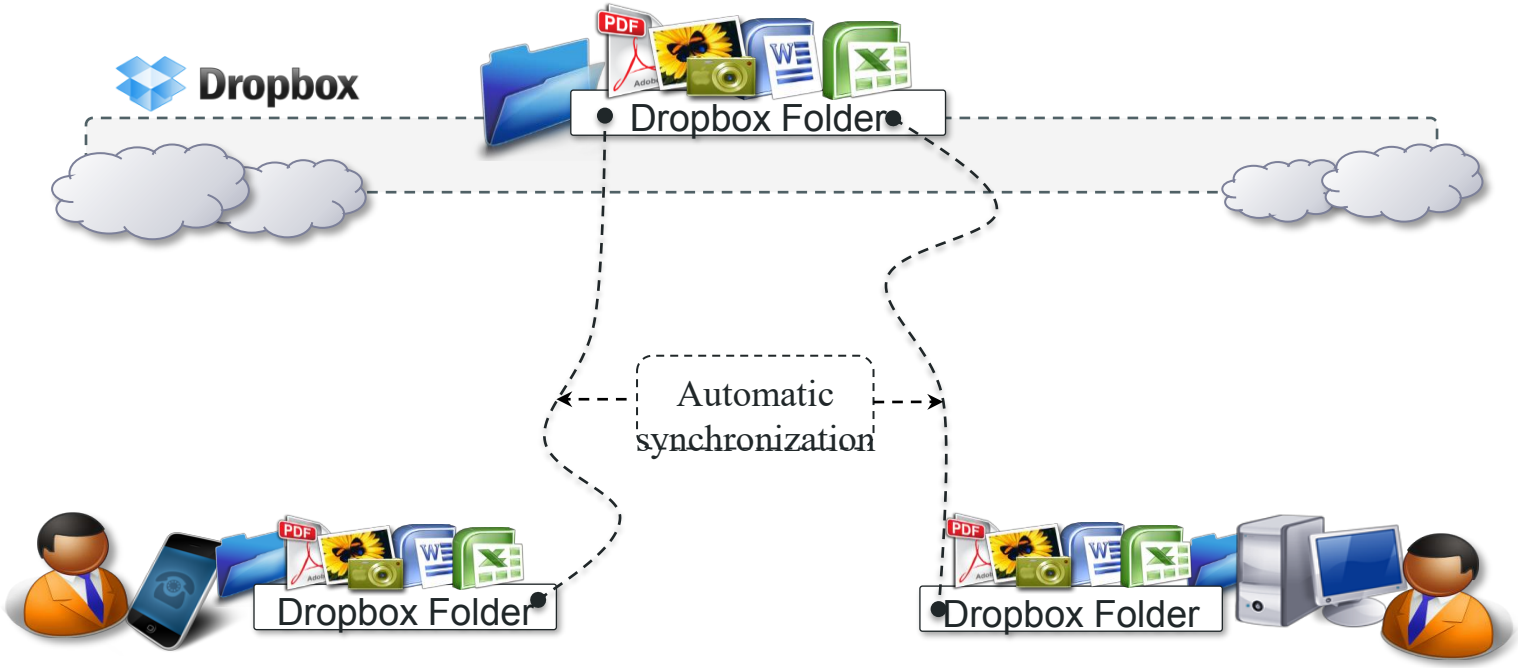


- Cache files on clients and perform local file system operations
- Advantages
- Local operations = better performance
- Disadvantages
- What happens when the client fails?
- Where should the client store the cached files?
- Difficult to keep local copy consistent with remote copy

Case Study: Dropbox

https://youtu.be/PE4gwstWhmc

Dropbox


Dropbox
Dropbox Folder
Automatic synchronization
Dropbox Folder
Dropbox Folder

Dropbox

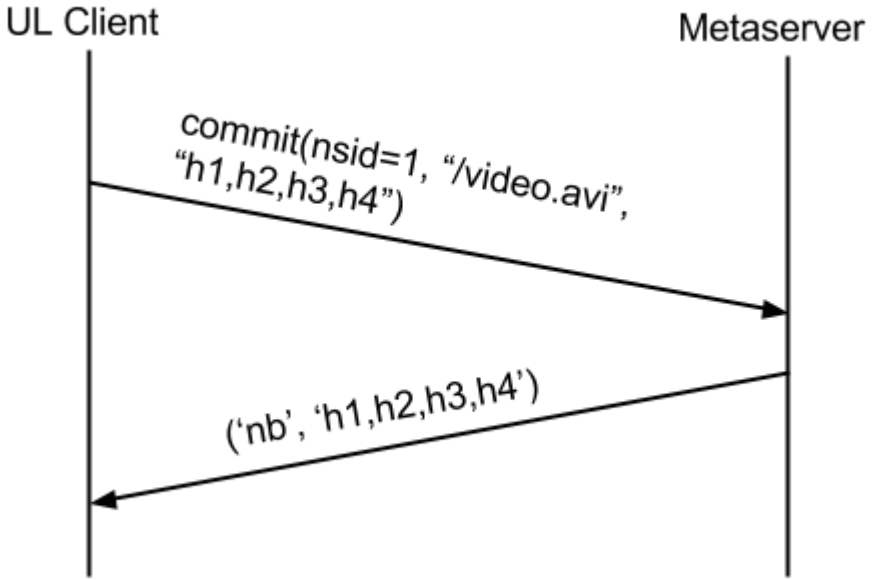
- Client runs on desktop
- Copies changes to local folder
  - Uploaded automatically
  - Downloads new versions automatically
- Huge scale – 100+ million users, 1 billion files/day
- Design
  - Small client, few resources
  - Possibility of low-capacity network to user
  - Scalable back-end
  - (99% of code in Python)

Dropbox

- Everyone’s computer has complete copy of Dropbox
  - Run daemon on computer to track “Sync” folder
- Traffic only when changes occur
  - Results in file upload : file download
  - Huge number of uploads compared to traditional service
  - Dropbox service’s read/write ratio is ***1:1***
- Uses compression to reduce traffic

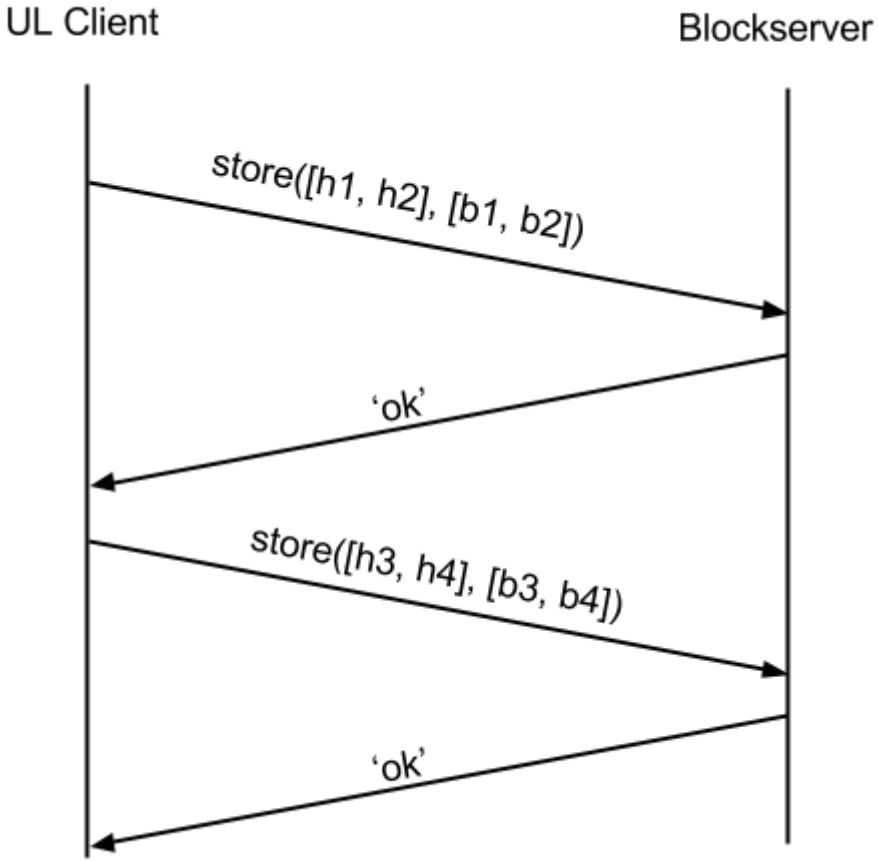
Dropbox - Upload

- Client attempts to “commit” new file
  - Breaks file into blocks, computes hashes
  - Contacts Metaserver
- Metaserver checks if hashes known
- If not, Metaserver returns that it “needs blocks” (nb)



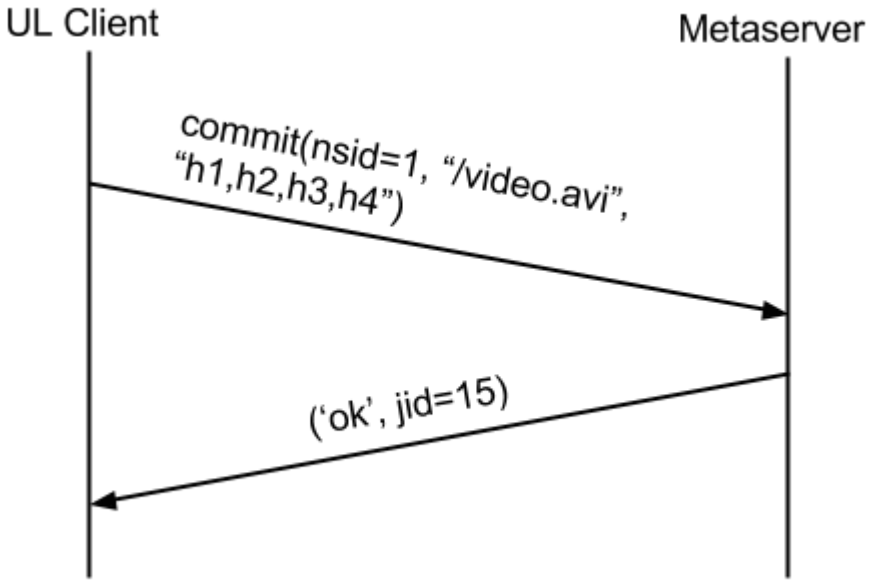
Dropbox - Upload

- Client talks to Blockserver to add needed blocks
- Limit bytes/request (typically 8 MB), so may be multiple requests



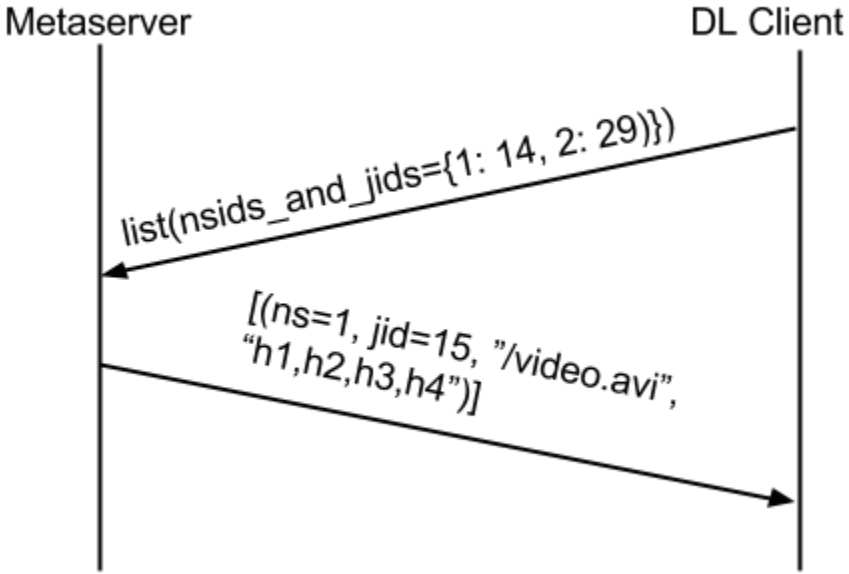
Dropbox - Upload

- Client commits again
  - Contacts Metaserver with same request
- This time, ok



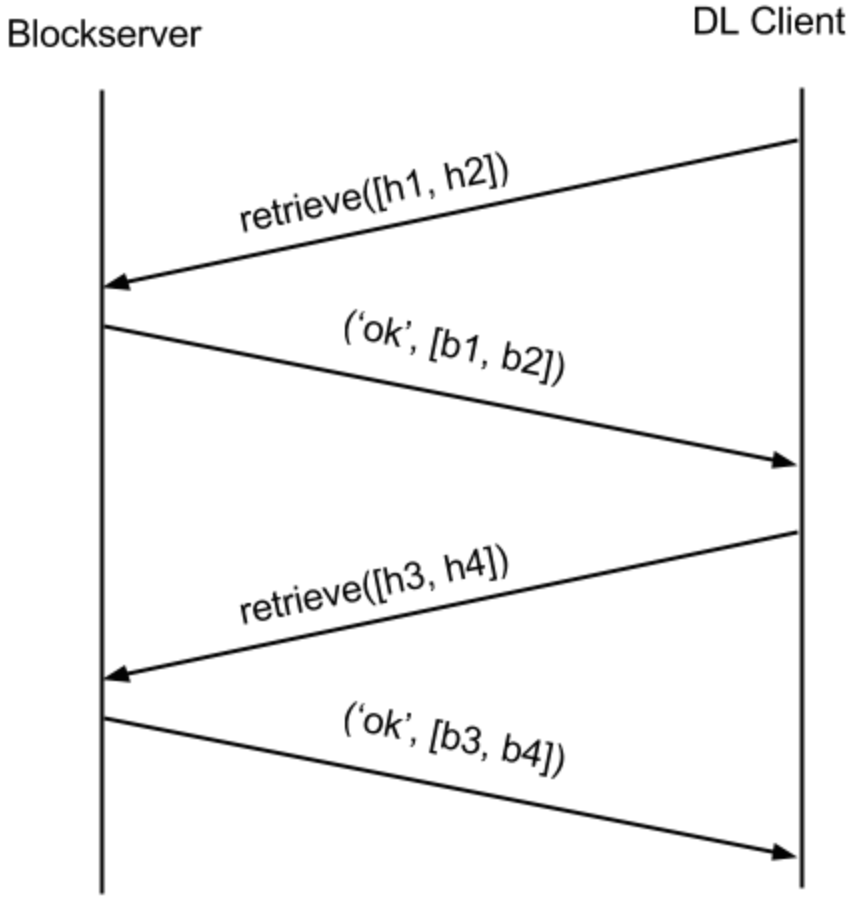
Dropbox - Download

- Client periodically polls Metaserver
  - Lists files it “knows about”
- Metaserver returns information on new files



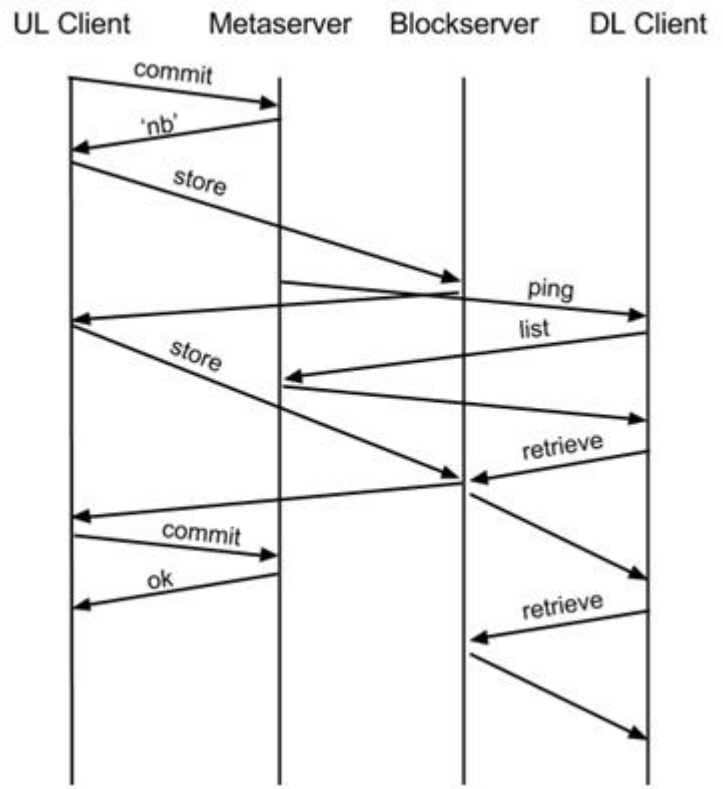
Dropbox - Download

- Client checks if blocks exist
  - For new file, this fails
- Retrieve blocks
- Limit bytes/request (typically 8 MB), so may be multiple requests
- When done, reconstruct and add to local file system
  - Using local filesystem system calls (e.g., open(), write()...)



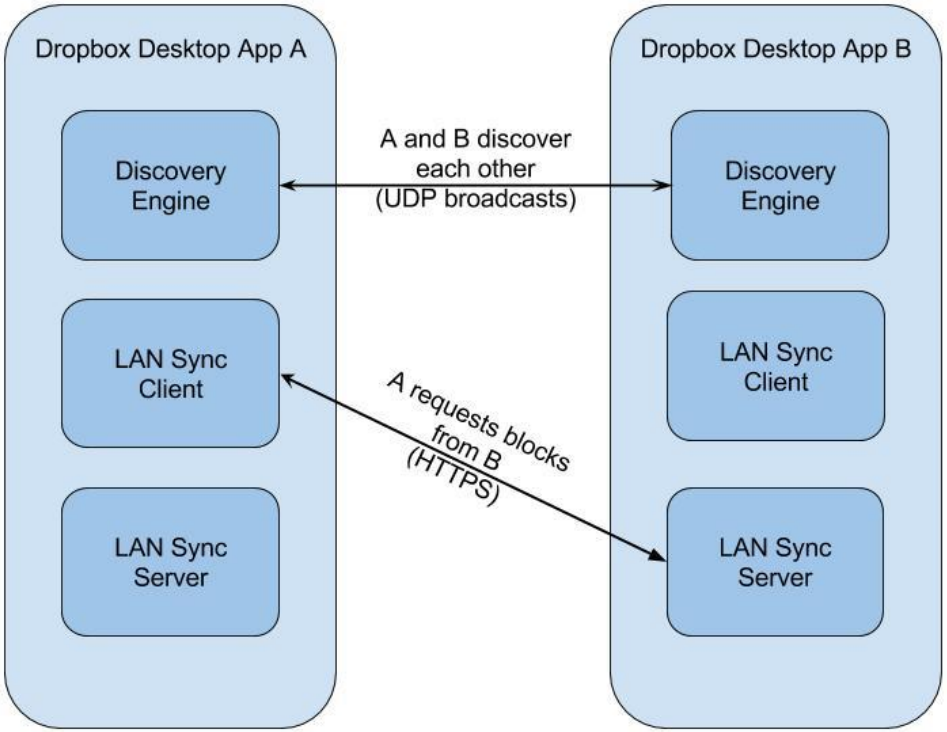
Dropbox – Streaming Sync

- Normally, cannot download to another until upload complete
  - For large files, takes time “sync”
- Instead, enable client to start download when some blocks arrive, before commit
  - Streaming Sync



Dropbox – LAN Sync

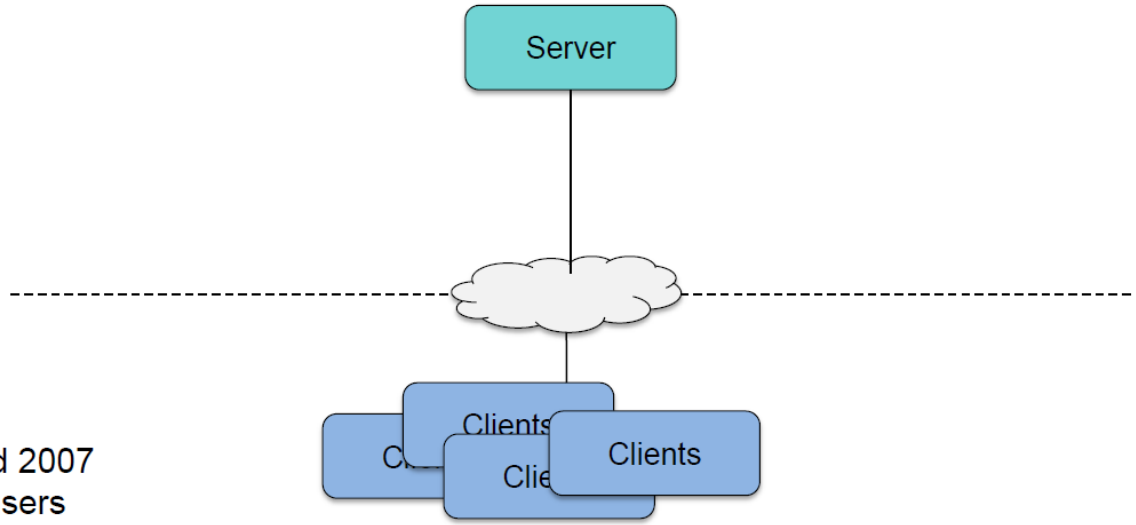
- LAN Sync – download from other clients
- Periodically broadcast on LAN (via UDP)
- Response to get TCP connection to other clients
- Pull blocks over HTTP



DropBox: Architecture Evolution

Dropbox Architecture – v1

– One server: web server, app server, mySQL database, sync server

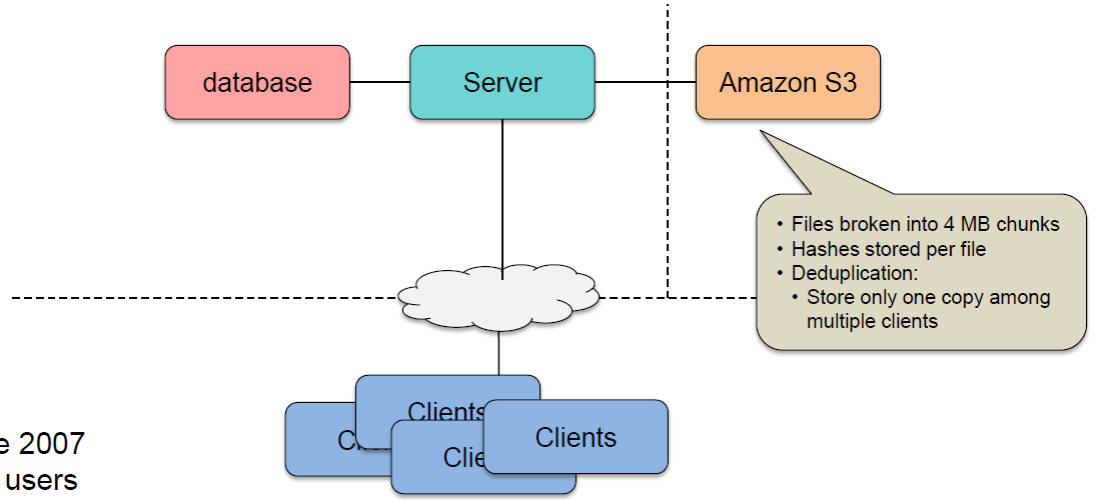


mid 2007
0 users

See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v2

- Server ran out of disk space:
  moved data to Amazon S3 service (key-value store)
- Servers became overloaded: moved mySQL DB to another machine
- Clients polled server for changes periodically

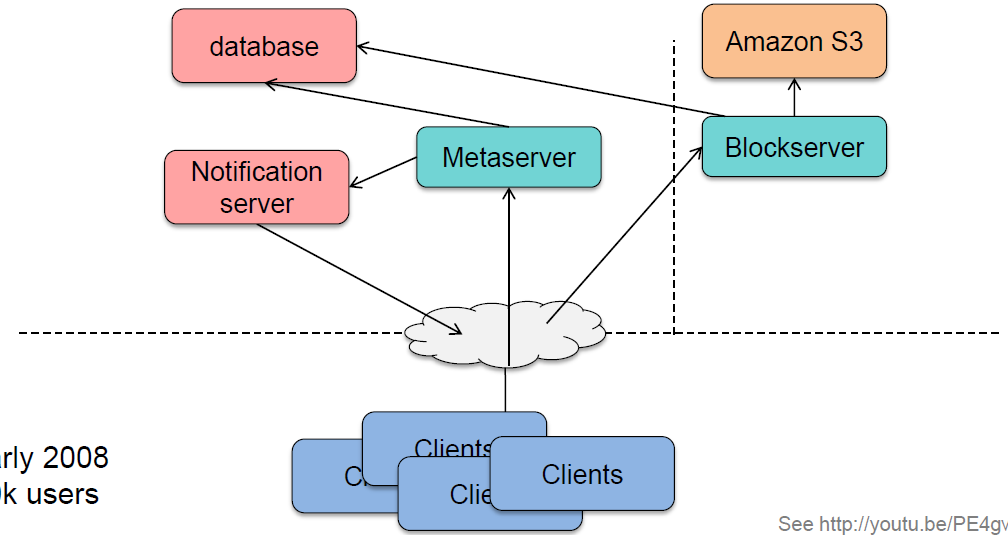


late 2007
~0 users

See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v3

- Move from polling to notifications: add notification server
- Split web server into two:
  - Amazon-hosted server hosts file content and accepts uploads (stored as blocks)
  - Locally-hosted server manages metadata

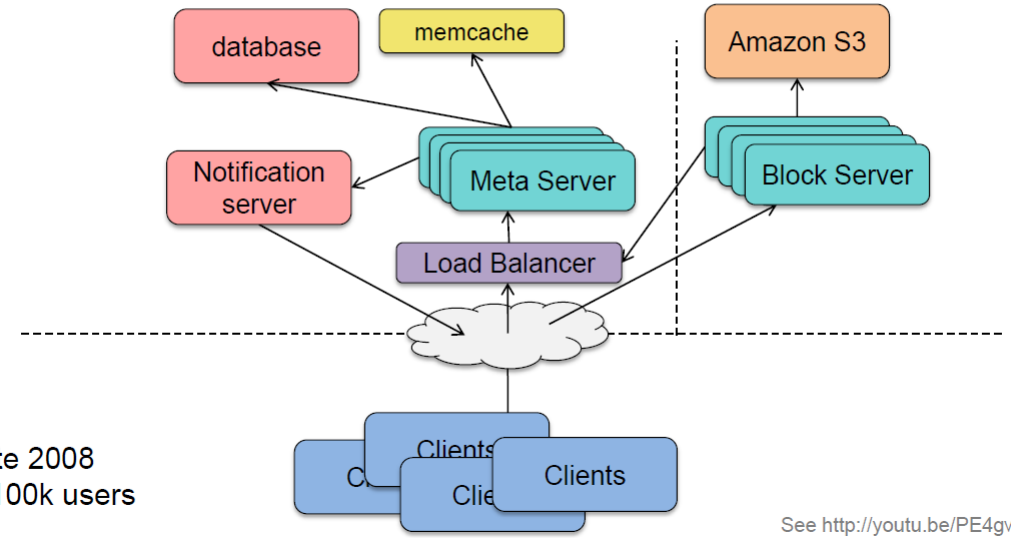


early 2008
50k users

See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v4

- Add more metaservers and blockservers
- Blockservers do not access DB directly; they send RPCs to metaservers
- Add a memory cache (memcache) in front of the database to avoid scaling

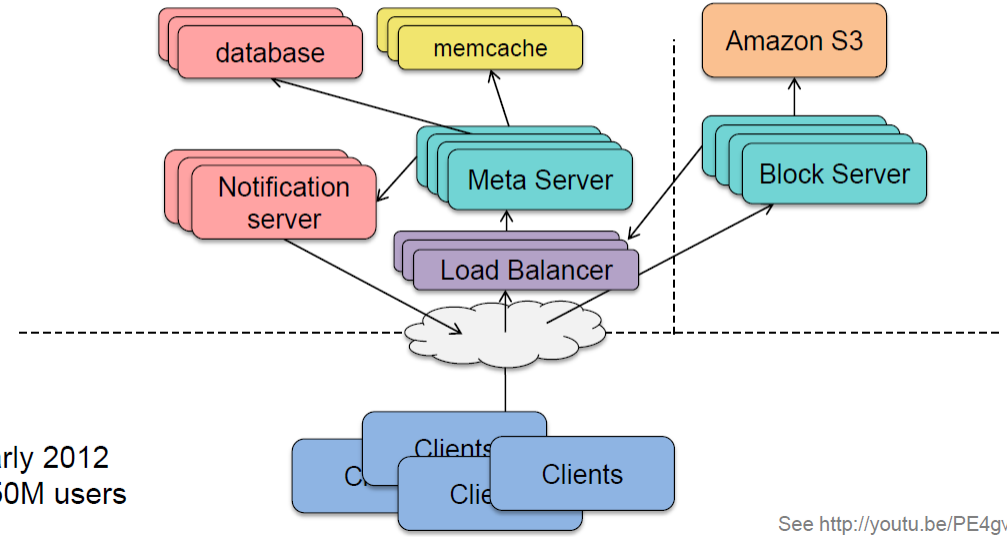


late 2008
~100k users

See http://youtu.be/PE4gwstWhmc

Dropbox Architecture – v5

- 10s of millions of clients – Clients have connect before getting notifications
- Add 2-level hierarchy to notification servers: ~1 million connections/server



early 2012
>50M users

See http://youtu.be/PE4gwstWhmc

Distributed Systems

COMP90015 2021 Semester 1
Tutorial 11

Things to cover today

- Name Services questions
- SSL Demo

Name Service Questions

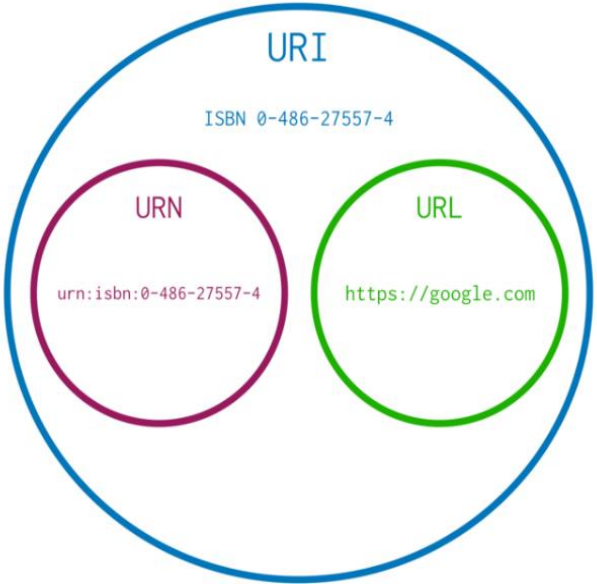
1. Define and discuss the concepts of Uniform Resource Identifiers

1. Define and discuss the concepts of Uniform Resource Identifiers

- *Uniform Resource Identifiers* (URIs) are concerned with identifying resources on the Web, and other Internet resources such as electronic mailboxes.
- URIs are intended to allow a generic way of specifying the identifier so as to make it easy for common software to process the identifier. This allows new types of identifiers to be readily introduced and for existing identifiers to be used by a wide variety of different software and services.
- URLs provide ways to locate the resource being named. They clearly suffer if the resource has since changed its name (e.g. broken links in the Web).
    - With specifying type of protocol (how) and remote name (where)

1. Define and discuss the concepts of Uniform Resource Identifiers

- URI (uniform resource identifier) identifies a resource (text document, image file, etc)
- URL (uniform resource locator) is a subset of the URIs that include a network location
- URN (uniform resource name) is a subset of URIs that include a name within a given space, but no location



2. What are Name Services and why do we need them?

2. What are Name Services and why do we need them?

- In a Distributed System, a Naming Service is a specific service whose aim is to provide a consistent and uniform naming of resources, thus allowing other programs or services to localize them and obtain the required metadata for interacting with them.
- The major operation of a name service is to resolve a name (along with - consistent naming scheme), i.e to lookup the attributes that are bound to the name.

2. What are Name Services and why do we need them?

- Name management is separated from other services largely because of the openness of distributed systems, which brings the following motivations:
  - Unification: Resources managed by different services use the same naming scheme, as in the case of URIs.
  - Integration: To share resources that were created in different administrative domains requires naming those resources. Without a common naming service, the administrative domains may use entirely different name formats.
- Key benefits
  - Resource localization (resources can be files, webpages, services, databases, etc..)
  - Uniform naming
  - Device independent address (e.g., you can move domain name/web site from one server to another server seamlessly).
  - Naming Services are not only useful to locate resources but also to gather additional information about them such as attributes (its domain, resource type, etc ..).

3. What is navigation and what are the approaches to navigation?

3. What is navigation and what are the approaches to navigation?

- When the name service is distributed then a single server may not be able to resolve the name. The resolve request may need to propagate from one server to another, referred to as *navigation*.
- Classification of different navigation approaches.
  - *iterative navigation* -- The client makes the request at different servers one at a time. The order of servers visited is usually in terms of domain hierarchy. Always starting at the root server would put excessive load on the root.
  - *multicast navigation* -- The client multicasts the request to the group (or a subset) of name servers. Only the server that holds the named request returns a result.
  - *non-recursive server-controlled navigation* -- The client sends the request to a server and the server continues on behalf of the client, as above.
  - *recursive server-controlled navigation* -- The client sends the request to a server and the server sends the request to another server (if needed) recursively.

4. What is Domain Name System and in what aspects does it improve on a file-based implementation?

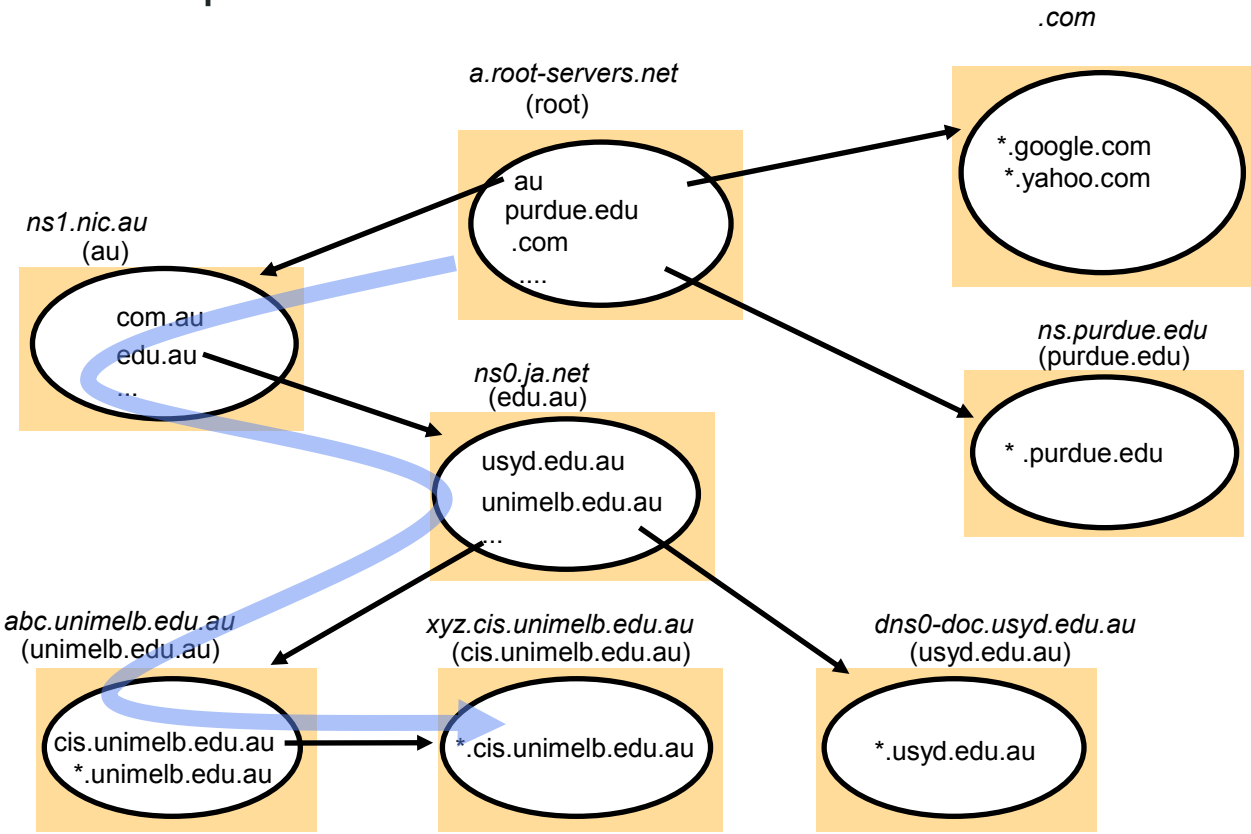
4. What is Domain Name System and in what aspects does it improve on a file-based implementation?

- The Domain Name System (DNS) is a name service design whose main naming database is used across the Internet.
- Before DNS, all host names and addresses were held in a single central master file and downloaded by FTP to all computers that required them.
- The problems with the original name service included:
  - It did not scale to large numbers of computers.
  - Local organizations wished to administer their own naming systems.
  - A general name service was needed -- not one that serves only for looking up computer addresses.
- DNS is designed for use in multiple implementations, each of which may have its own name space, though in practice the Internet DNS name space is the one in widespread use.

4. What is Domain Name System and in what aspects does it improve on a file-based implementation?

Note: Name server names are in italics, and the corresponding domains are in parentheses.
Arrows denote name server entries

authoritative path to lookup:
xyz.cis.unimelb.edu.au



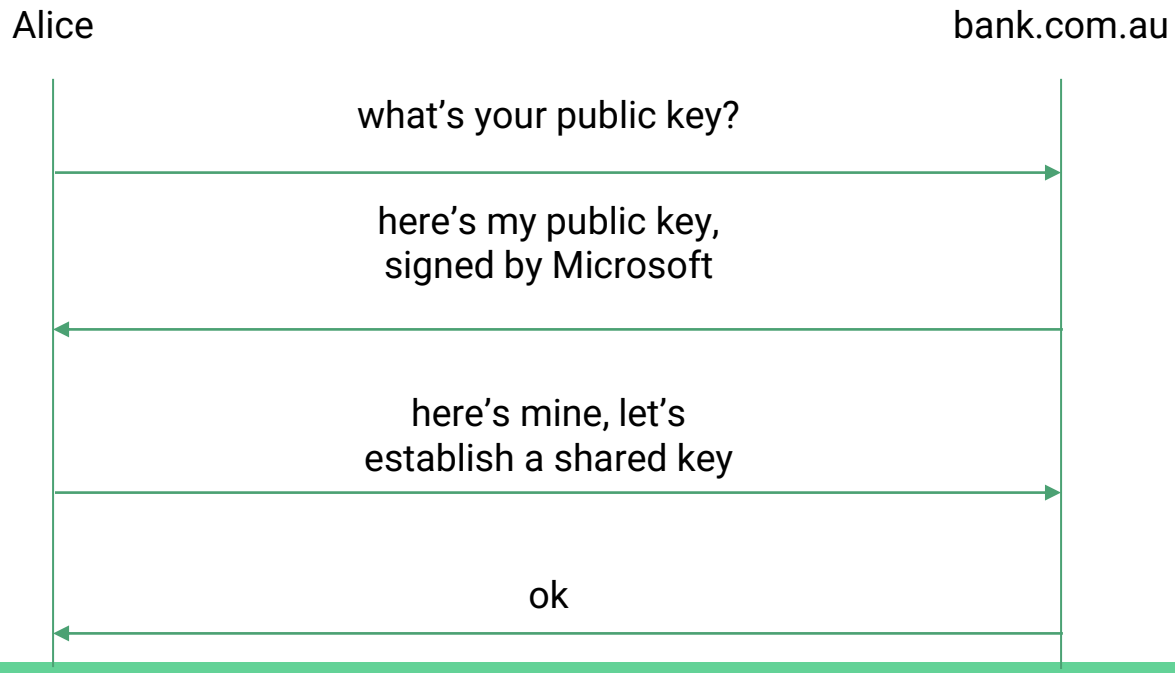
- SSL Demo

Transport Layer Security (TLS)

TLS is a modern protocol that handles this for you. Most communication these days (e.g. HTTPS) uses TLS.

Certificates: here is my public key, and here is a signature from a trusted authority that says it really does belong to me.

Transport Layer Security (TLS)



Transport Layer Security (TLS)

Code demonstration:

Q & A

Best wishes! This is your last tutorial.