Distributed Systems

COMP90015 2019 Semester 1
Tutorial 02

Q1. Define and briefly explain each of the following distributed system challenges:

- Concurrency

- Heterogeneity

- Failure Handling

- Openness

- Security

- Scalability

- Transparency

Concurrency

- Multiple clients can access the same resource at the same time, in some cases for updates
- One approach to handling concurrency is making access sequential - slows down the system
- Semaphores supported by the operating system is a well accepted mechanism to handle concurrency

Heterogeneity

- Where?
  - Networks
  - Computer hardware
  - Operating systems
  - Programming Languages
  - Implementation by different developers
- How to deal with it?
  - Standard protocols
  - Adhering to articulated APIs, message formats, and data types
  - Middleware
  - Portable code

Heterogeneity and mobile code

Mobile code is sent from one computer to the anther to run at the destination (e.g. Java applets):

- Code that is compiled to run in one OS does not run in another:
  - Different hardware
  - Different OS versions
- Virtual Machine approach provides a way of making code executable on any hardware – compiler produces code that is interpreted by the virtual machine
- Cross-platform compilation and code portability is another way, that compiles source code to multiple targets.

Failure Handling

- **Detecting**: some types of failures can be detected, e.g. checksums can be used to detect corrupted data in a message, and some kinds of failure are hard to be certain about, e.g. the failure of a remote server
- **Masking**: some failures that have been detected can be hidden or made less severe, e.g. timeout and message retransmission
- **Tolerating**: it is sometimes impractical to try and handle every failure that occurs, sometimes it is better to tolerate them, e.g. failure is reported back to the user, e.g. web server not being available, try again later, or video is rendered with errors
- **Recovery**: failure sometimes leads to corrupted data and software can be designed so that it can recover the original state after failure, e.g. implementing a roll back mechanism.
- **Redundancy**: services can be made to tolerate failures using redundant components, e.g. multiple servers that provide the same service, as so called fail over.

Openness

Openness refers to the ability to extend the system in different ways, by adding hardware or software resources. Following are some approaches to address openness:

- Publishing key interfaces
- Allowing a uniform communication mechanism to communicate over the published interfaces
- Ensuring all implementations adhere to the published standards

Security

- There has three aspects of security:
  - Confidentiality (protection against disclosure to unauthorized individuals)
  - Integrity (protection against alteration and corruption)
  - Availability (protection against interference with the means of access)
- Security Mechanisms:
  - Encryption (e.g. Blowfish, RSA)
  - Authentication (e.g. passwords, public key authentication)
  - Authorization (e.g. access control lists)

Scalability

A system is considered to be scalable if it can handle the growth of the number of users.

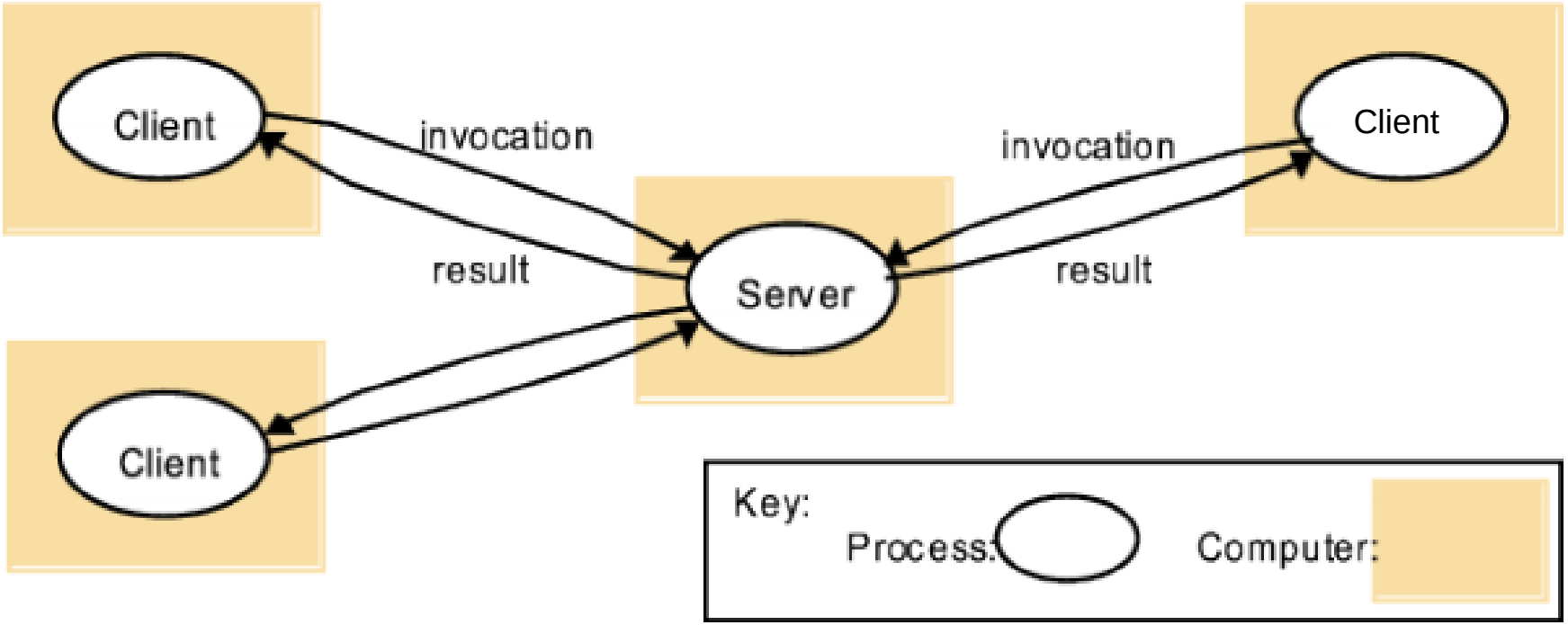
- Scalability Challenges:
  - Cost of physical resources
  - Controlling the performance loss
  - Resources should not run out
  - Avoiding Performance bottlenecks

Transparency

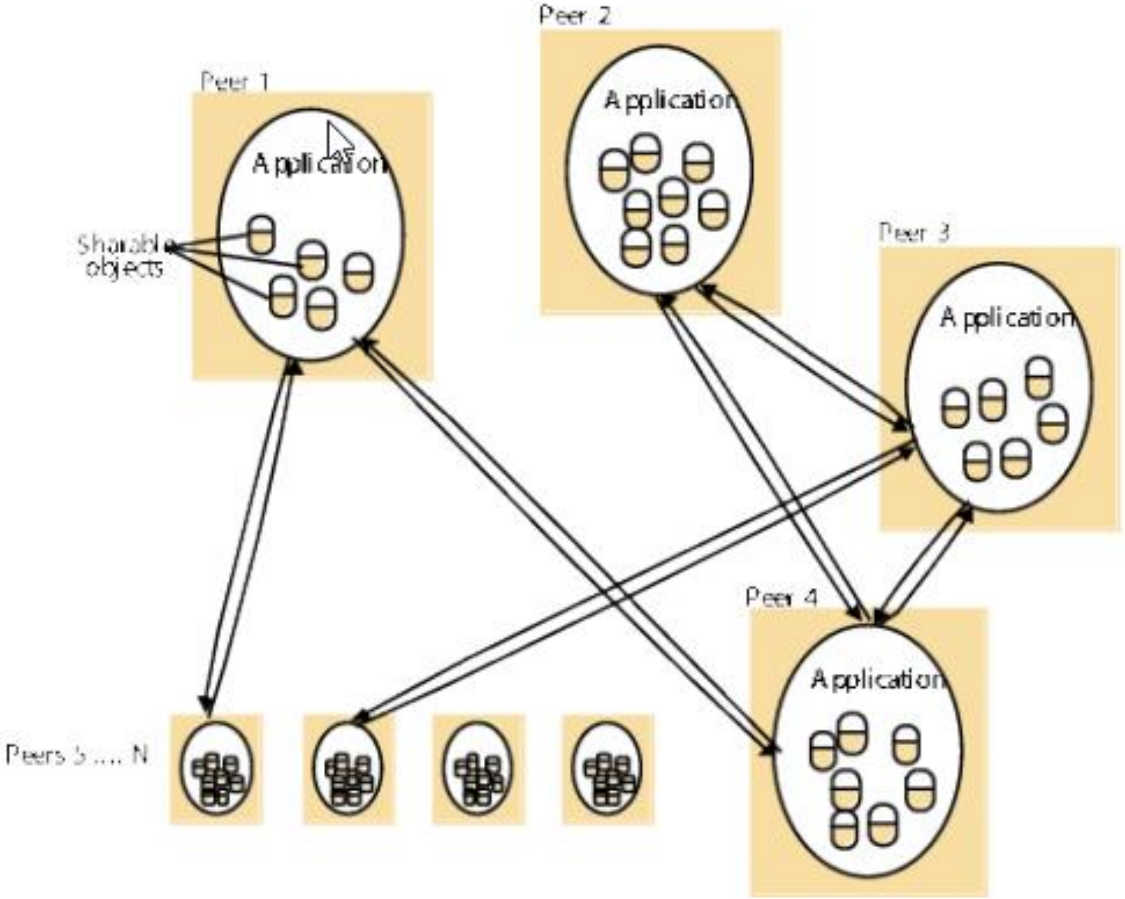
- Access transparency
  - Access to local or remote resources is identical
- Location transparency
  - Access without knowledge of location
- Failure transparency
  - Tasks can be completed despite failures
- And many more

Q2. Briefly explain the difference between a client-server architecture and a peer-to-peer architecture.

Client-server



Peer-to-Peer


Peer 1
Application
Shareable objects
Peer 2
Application
Peer 3
Application
Peer 4
Application
Peers 5 .... N

Q3. Briefly explain each of the following distributed system architecture variations, giving also a reason or a benefit for its use:

- Services provided by multiple servers

- Proxy servers and caches

- Mobile code and Mobile Agents

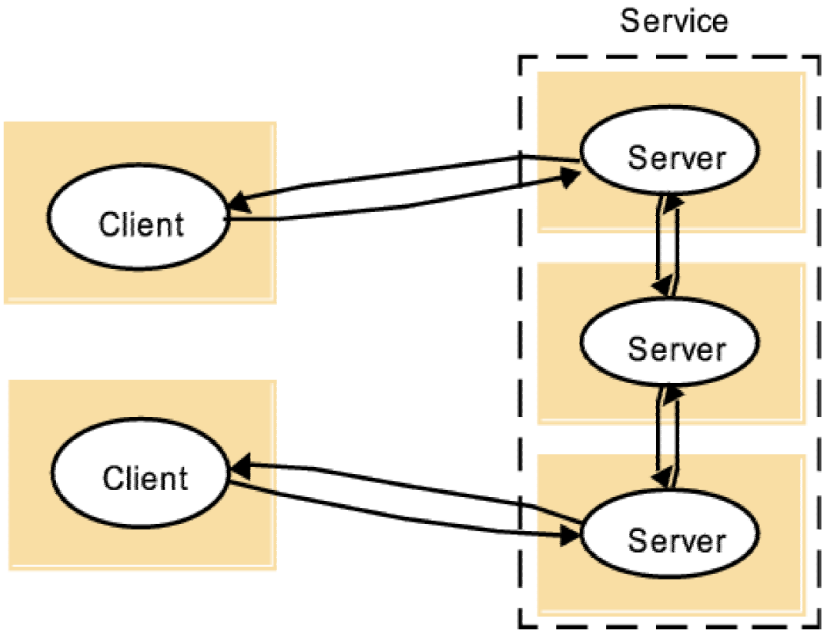
- Network computers

- Thin clients

- Tiered Architecture

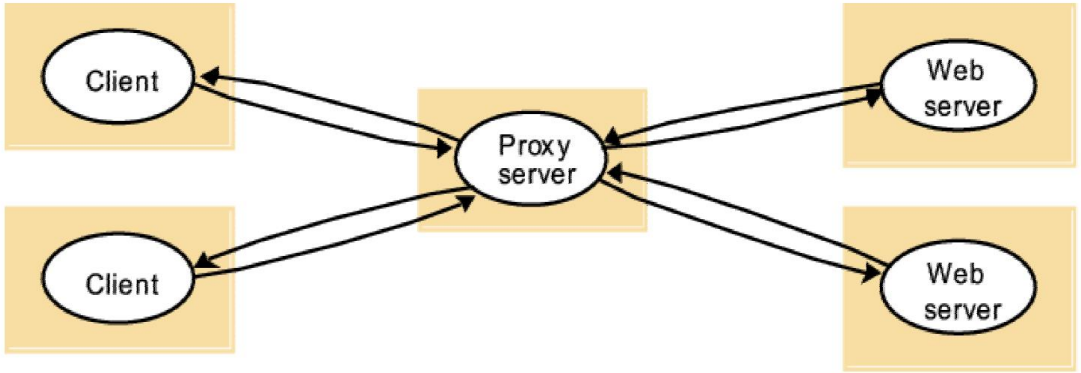
Services provided by multiple servers

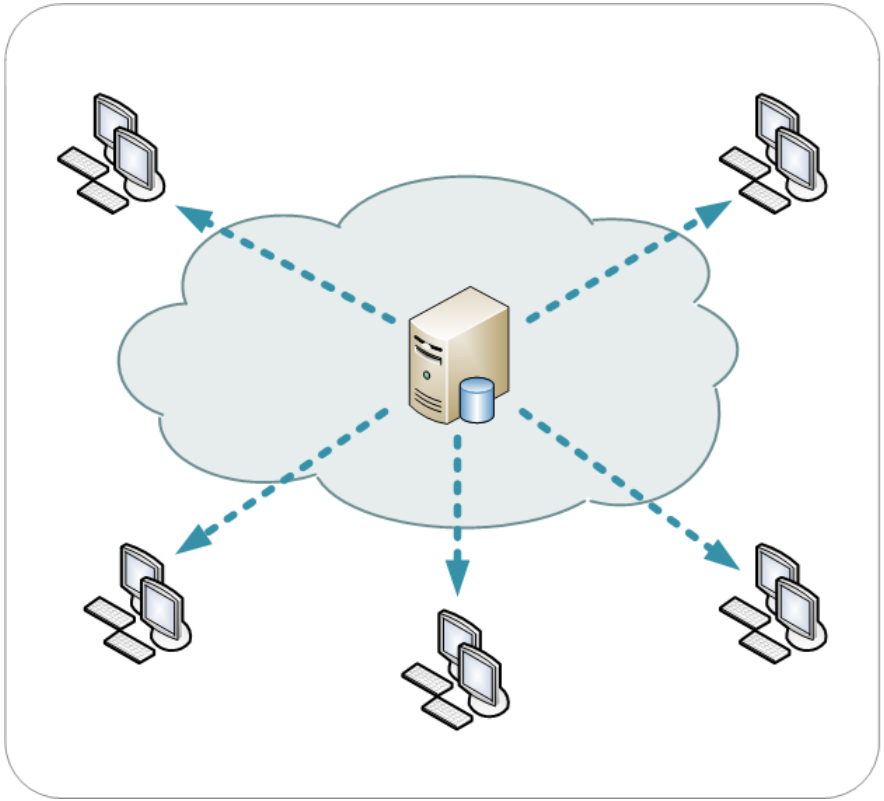
- Services may be implemented as several server processes in separate host computers interacting as necessary to provide a service to client processes
- Servers may
  - Partition the set of objects on which the service is based and distribute those objects between themselves
  - Maintain replicated copies of them on several hosts
- Improve performance and reliability



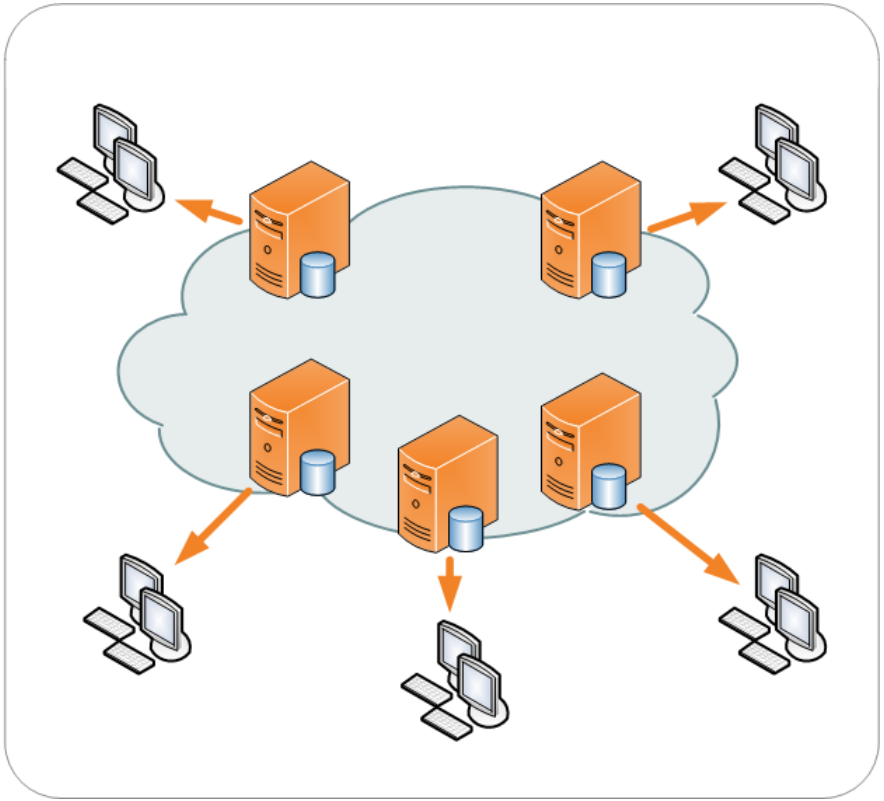
Proxy servers and caches

- Cache
  - A store of recently used data objects that is closer to the client
  - They may be co-located with each client or they may be located in a *proxy server* that can be shared by several clients
- Increase the availability and performance of the service by reducing the load on the wide area network and web servers
- Proxy servers can take on other roles --- better reliability
- Improved security
- Access restriction
- Privacy protection





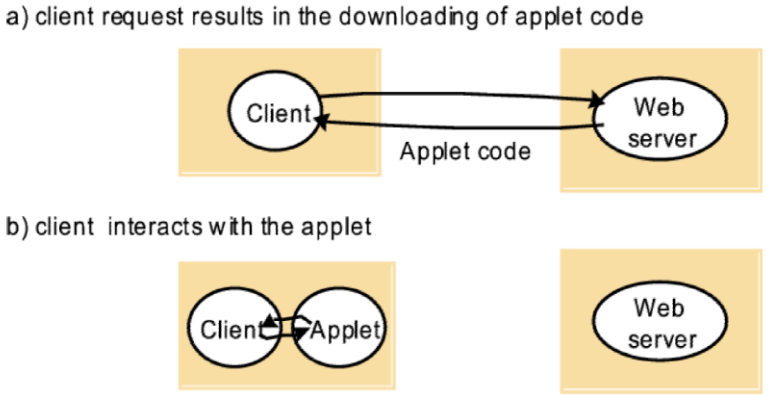
Single server distribution



CDN scheme of distribution

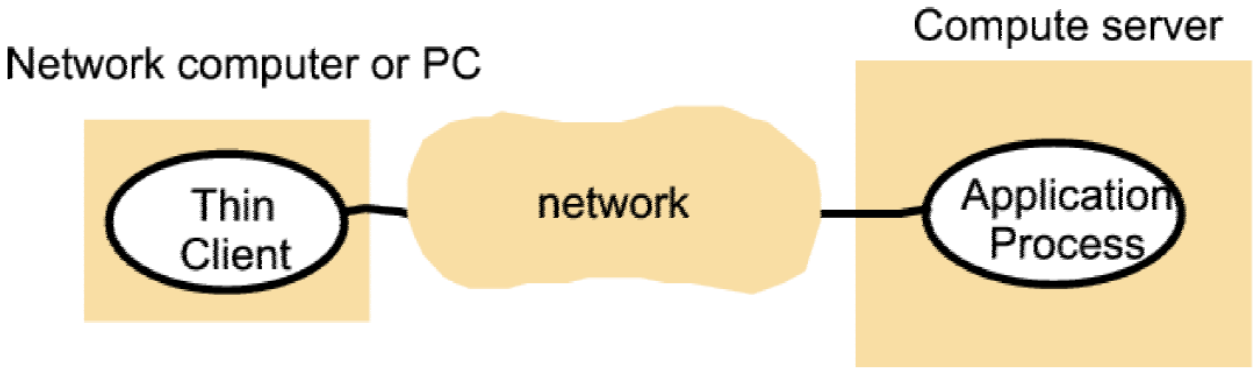
Mobile code and Mobile Agents

- Mobile code
  - Mobile Code is down loaded to the client and is executed on the client (e.g. applet).
  - Good interactive response
  - Security threat
- Mobile agents
  - Mobile agents are running programs that includes both code and data that travels from one computer to another.
  - They process data at the data source, rather than fetching it remotely
    - Less communication overhead by replacing remote invocations with local ones
  - Security threat



Network computers and thin clients

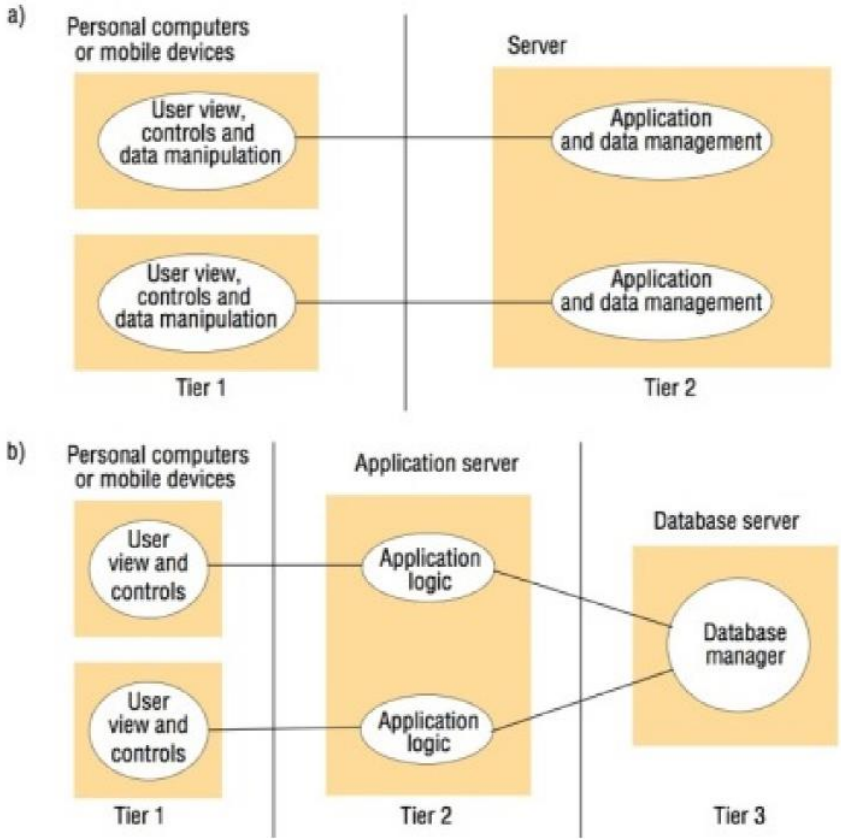
- Network Computers
  - Download their operating system and application software from a remote file system. Applications are run locally.
- Thin clients
  - Move complexity away from the end-user device
  - Local user interface, remote services or applications
  - Few assumptions or demands on the client device



Tiered Architecture

- Tiered architectures are complementary to layering, which deals with horizontal organization of services.

- Layering deals with vertical organization of services



Questions?