CSIE52400/CSIEM0140 Distributed Systems

Lecture 03 Architectures & Models

吳 秀 陽
Shiow-yang Wu

Department of Computer Science and Information Engineering
National Dong Hwa University

Architectural Styles

- An architectural style is formulated by:
  - (replaceable) components with well-defined interfaces
  - the way that components are connected to each other
  - the data exchanged between components
  - how these components and connectors are jointly configured into a system.
- Connector: A mechanism that mediates communication, coordination, or cooperation among components. Example: facilities for (remote) procedure call, messaging, or streaming.

System Models

- Use models to capture and discuss the properties and design issues of distributed systems.
- Types of models
  - **Physical models** – Describe a system by its hardware composition of computers and networks.
  - **Architectural models** – Describe a system by its computational and communication tasks performed by its computational elements.
  - **Fundamental models** – Describe a system from an abstract perspective to examine individual aspects.

Physical Models

- An abstract representation of the hardware elements of a distributed system.
- Baseline physical model: a set of computer nodes interconnected by a network and coordinated by passing messages.
- Three generations of distributed systems: (next slide)
  - Early
  - Internet-scale
  - Contemporary(當代的)

Generations of Distributed Systems

Distributed systems:	*Early*	*Internet-scale*	*Contemporary*
Scale	Small	Large	Ultra-large
Heterogeneity	Limited (typically relatively homogenous configurations)	Significant in terms of platforms, languages and middleware	Added dimensions introduced including radically different styles of architecture
Openness	Not a priority	Significant priority with range of standards introduced	Major research challenge with existing standards not yet able to embrace complex systems
Quality of service	In its infancy	Significant priority with range of services introduced	Major research challenge with existing services not yet able to embrace complex systems

Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5

Architectural Models

- System architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.
- Describe architectural models from three perspectives
  - Architectural elements
  - Architectural patterns
  - Middlewares

Architectural Elements

- Communicating entities – Basic elements
- Communication paradigm – How do elements comm.
- Roles and responsibilities of elements.
- Placement – How are they mapped on to the physical infrastructure?

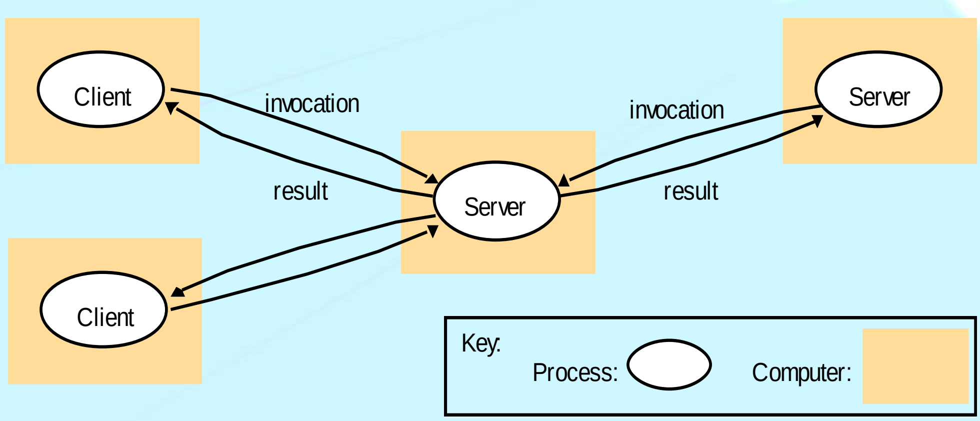
Communicating Entities and Communication Paradigms

Communicating entities (what is communicating)		*Communication paradigms (how they communicate)*		
System-oriented entities	*Problem-oriented entities*	*Interprocess communication*	*Remote invocation*	*Indirect communication*
Nodes	Objects	Message passing	Request-reply	Group communication
Processes	Components	Sockets	RPC	Publish-subscribe
	Web services	Multicast	RMI	Message queues
	Agents			Tuple spaces
				DSM

Roles and Responsibilities

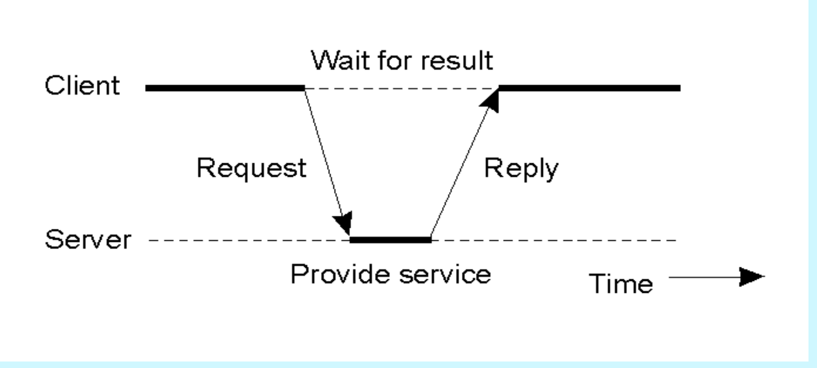
- Entities can take on different roles in a distributed system.
- These roles are fundamental in establishing the overall architecture.
- Two examples of architectural styles with distinctive roles:
  - Client-server
  - Peer-to-peer (P2P)

Client-Server Architecture



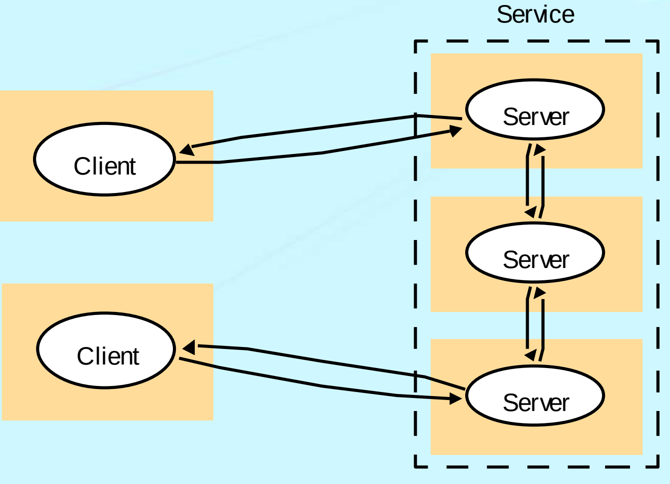
Client-Server Interaction

- General interaction (request-reply behavior) between a client and a server.



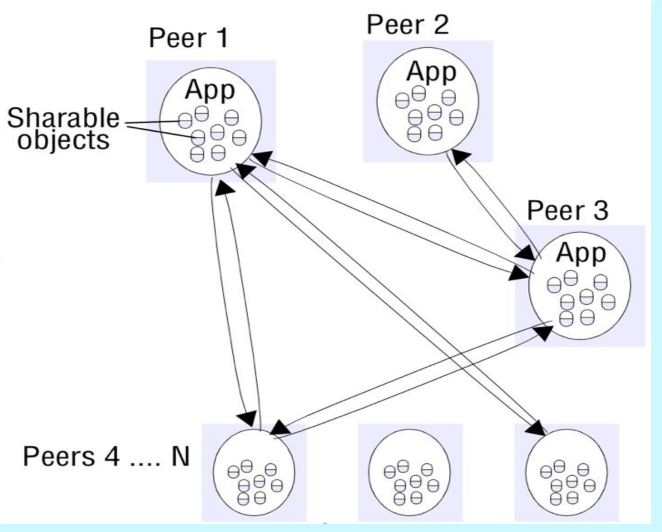
CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 11

Multiple Servers Architecture



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 12

Peer-to-peer Architecture



Peer-to-Peer Systems

- Each component is symmetric in functionality
  - Servent: Combination of server-client
- How does a node find the other?
  - No "well-known" centralized server

Overlay Network

- A logical network consisting of participant components (processes/machines)
  - Built on top of physical network
- Can be thought of as a graph
  - Nodes are processes/machines, links are communication channels (e.g., TCP connections)

Types of P2P Systems

- Unstructured: Built in a random manner
  - Each node can end up with any sets of neighbors, any part of application data
  - E.g.: Gnutella, Kazaa
- Structured: Built in a deterministic manner
  - Each node has well-defined set of neighbors, handles specific part of application data
  - E.g.: CAN, Chord, Pastry

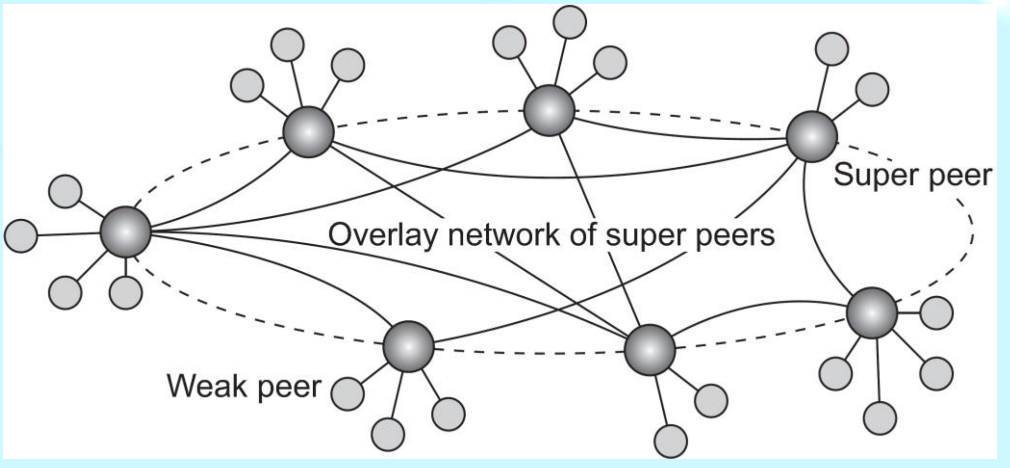
Unstructured P2P Architectures

- Each node has a list of neighbors to which it is connected
  - Communication to other nodes in the network happens through neighbors
  - Neighbors are discovered in a random manner
  - Exchange information with other nodes to maintain neighbor lists
- Application data is randomly spread across the nodes
  - Flooding: To search for a specific item

Structured P2P Architectures

- Nodes and data are organized deterministically
- Distributed Hash Tables (DHT)
  - Each node has a well-defined ID
  - Each data item also has a key
  - A data item resides in the node with nearest key
- Each node has information about neighbors in the ID space
- Searching for a data item:
  - Routing through the DHT overlay network

Super-peer Networks



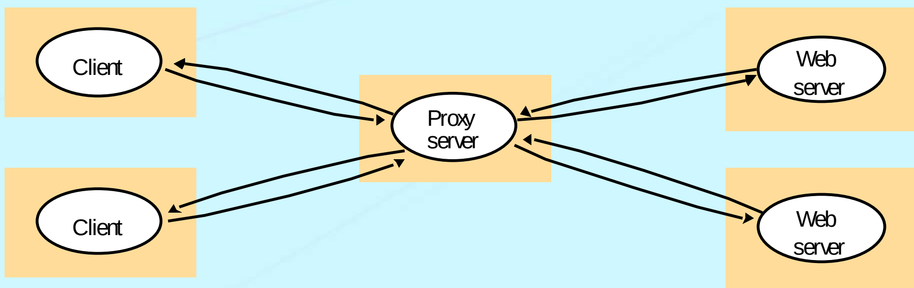
CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 19

Placement

- The mapping of entities on to physical infrastructure.
- Where to place different entities?
- Placement is crucial in determining the properties such as
  - Performance
  - Reliability
  - Security

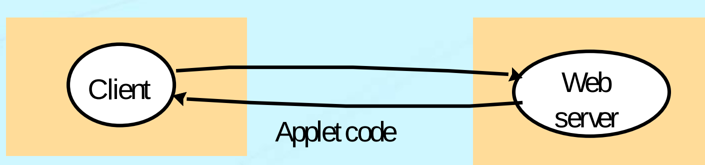
CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 20

Proxy Servers and Cache Architecture



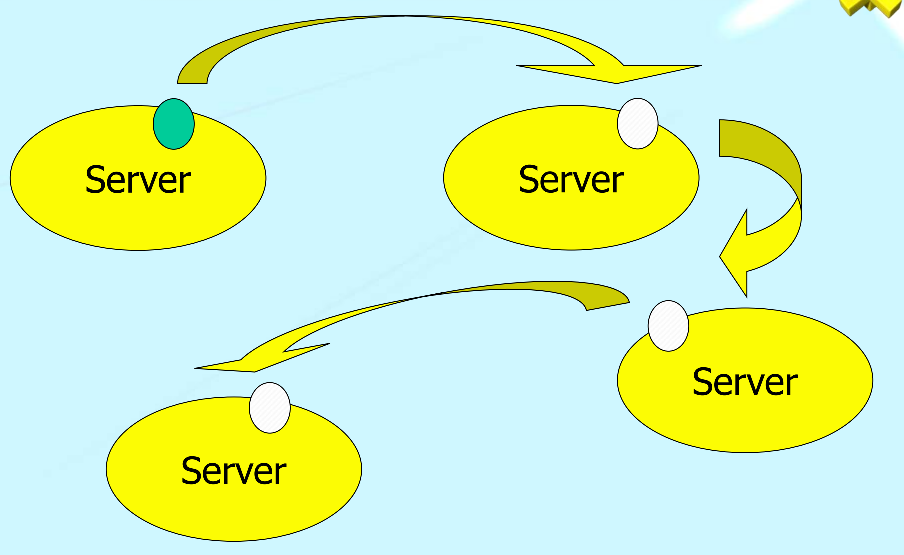
Web Applets

a) client request results in the downloading of applet code



b) client interacts with the applet

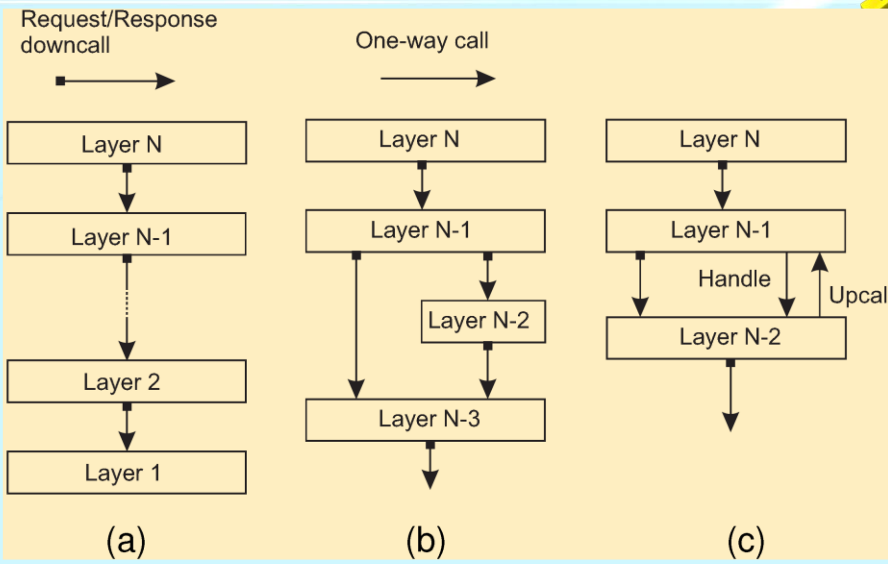
Mobile Code and Mobile Agent



Architectural Patterns

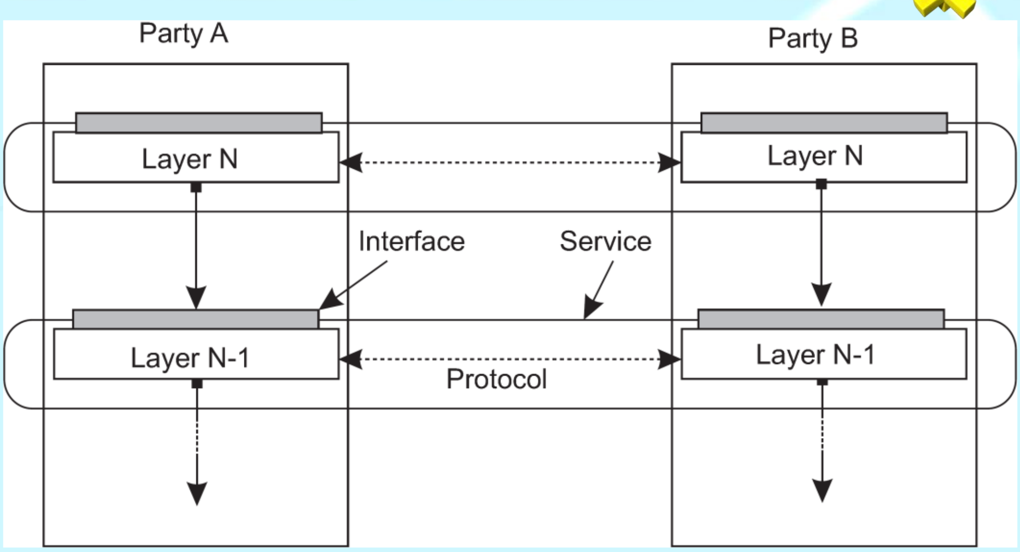
- Composite structures on top of architectural elements
- Some patterns have been shown to work well in given circumstances
- Not necessarily complete solutions but offer partial insights.

Layered Architecture



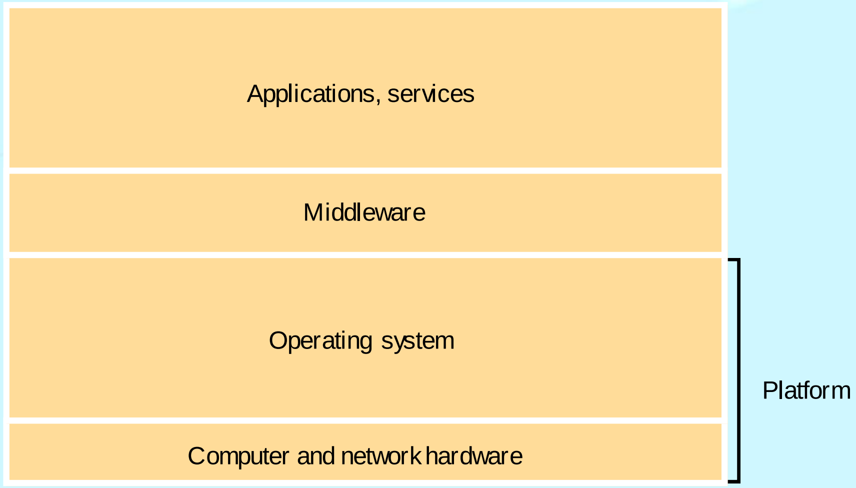
CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 25

Service, Interface, Protocol



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 26

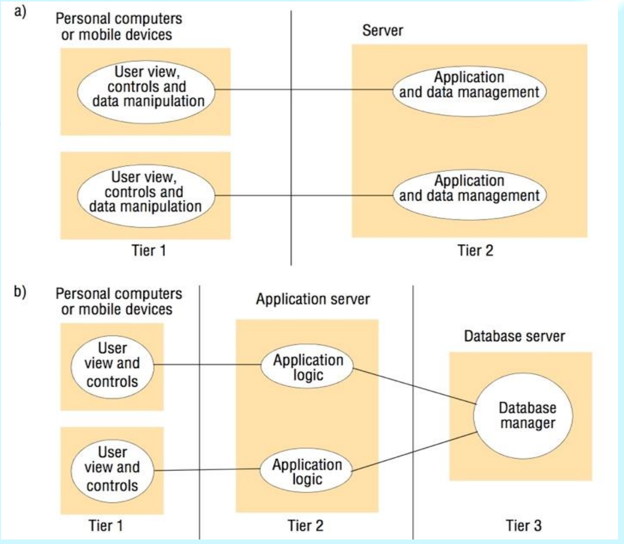
Software and hardware layers in distributed systems



Layering

- Can also think about layering in terms of dependencies:
  - A layer A is above layer B if changes to the interfaces provided by layer A do not require changes to the code of layer B.
- Why layer?
  - Flexible — You can add functionality without changing underlying layers.
  - Reuse — Many applications can use Java jars, for example.
  - Reduce complexity — Too hard to hold everything in your head at once.

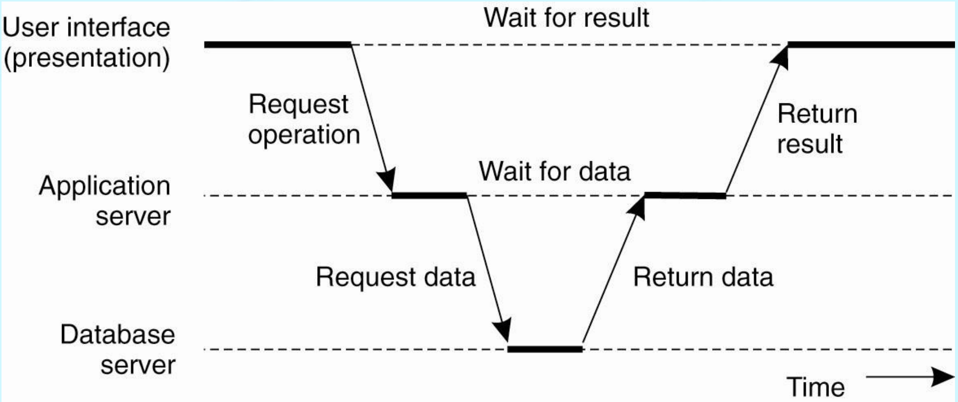
Two-tier and three-tier architectures



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 29

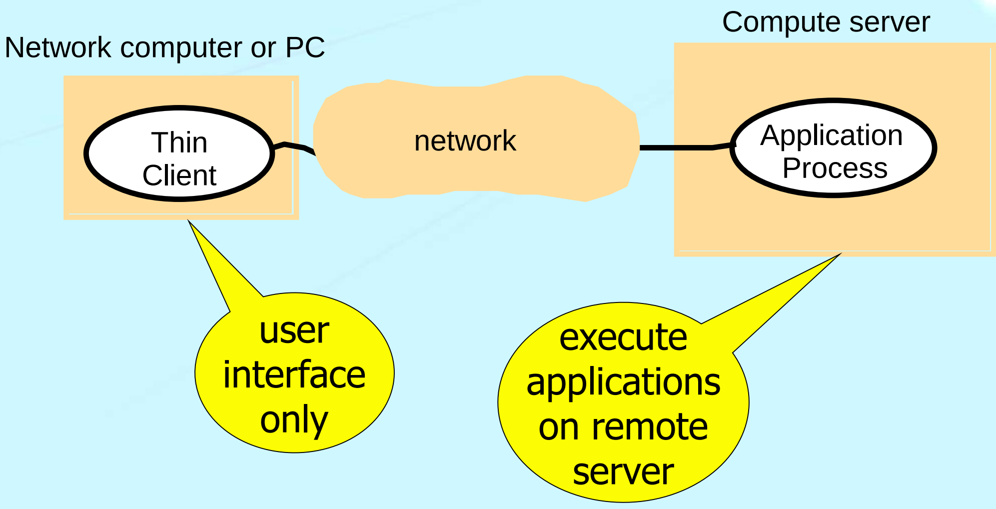
Three-tiered Architecture

- Note that the application server acts as client when requesting to the database server.



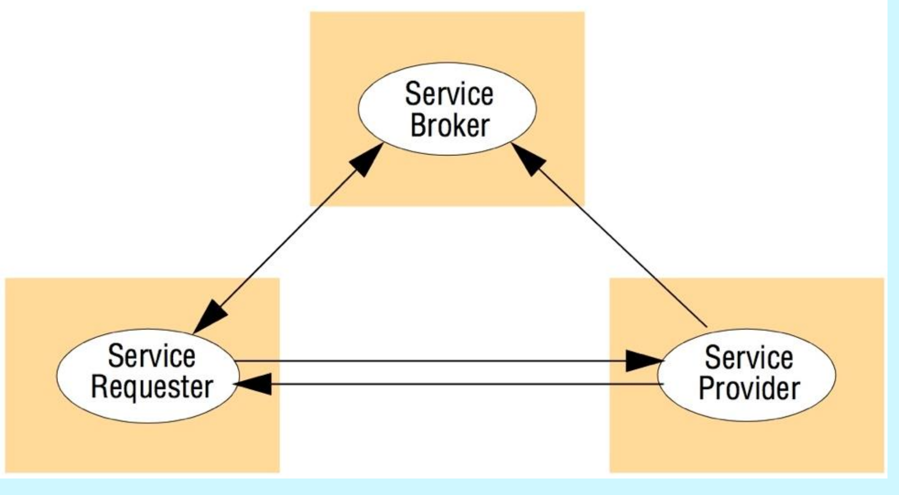
CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 30

Thin Client Architecture



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 31

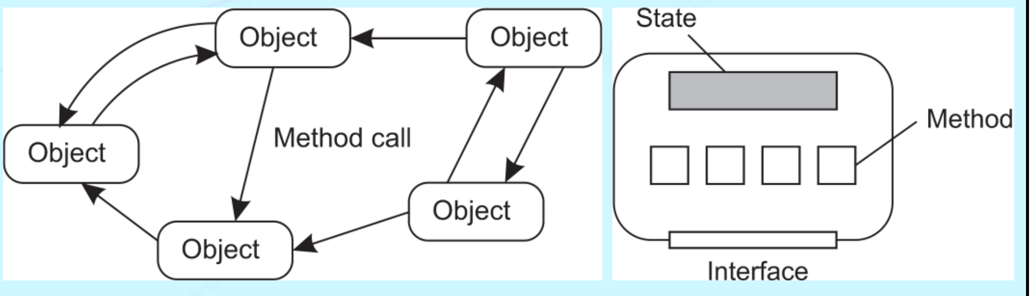
Web Service Architecture



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 32

Object-based Architecture

- The object-based architectural style.



RESTful Architecture

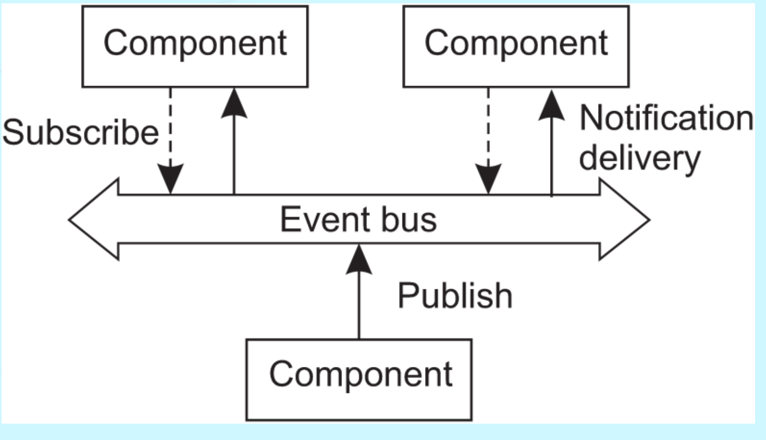
- View a distributed system as a collection of resources, managed by components, and may be added, removed, retrieved, and modified by applications.
  - Resources identified through a single naming scheme (URI) and usually represented by JSON or XML
  - All services offer the same interface (uniform interface)
  - Messages sent to or from a service are fully self-described
  - After executing an operation, that component forgets everything about the caller (stateless)

RESTful Interface

Method	Operation performed on server	Quality
GET	Read a resource.	Safe
PUT	Insert a new resource or update if the resource already exists.	Idempotent
POST	Insert a new resource. Also can be used to update an existing resource.	N/A
DELETE	Delete a resource .	Idempotent
OPTIONS	List the allowed operations on a resource.	Safe
HEAD	Return only the response headers and no response body.	Safe

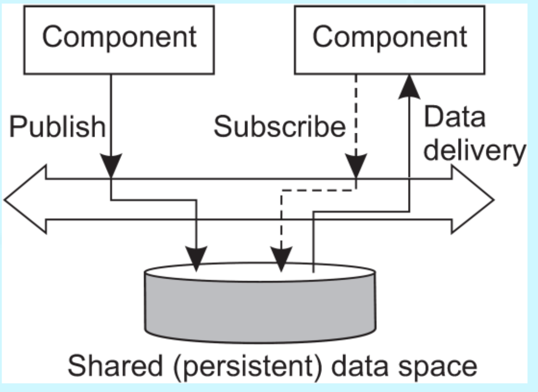
Event-based Architecture

- The event-based architectural style



Shared Data-space Architecture

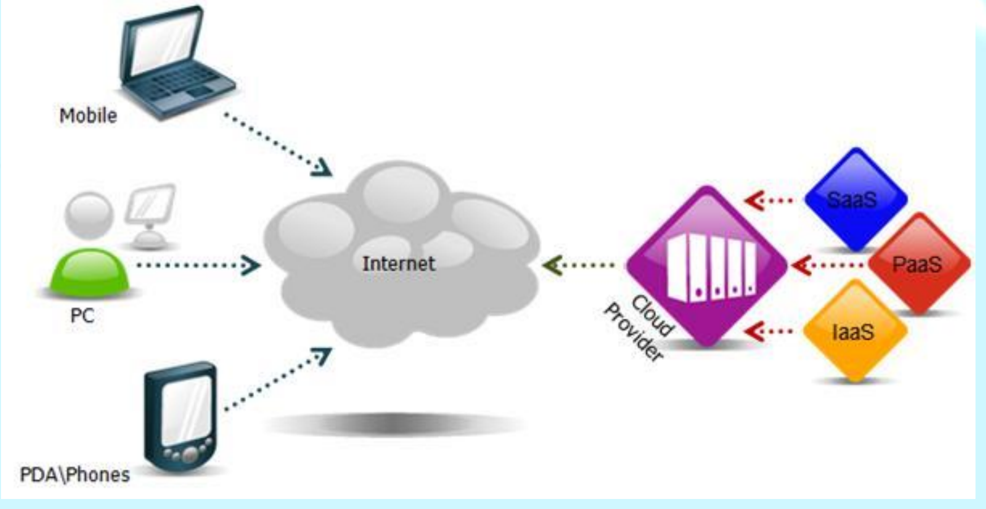
- The shared data-space architectural style.
  - Processes communicate through a shared repository.
  - WebDAV, Linda, tuple-space.



Temporal & Referential Coupling

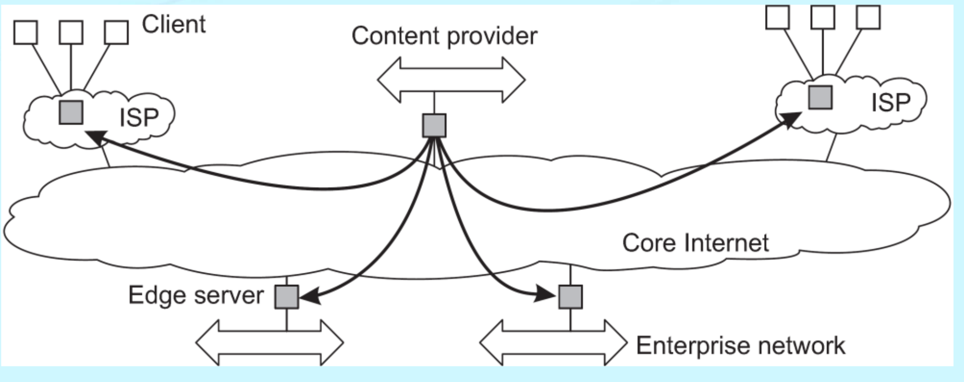
	Temporally Coupled	Temporally Decoupled
Referentially Coupled	Direct	Mailbox
Referentially Decoupled	Event-based	Shared data space

Cloud Service Architecture



Edge-server Architecture

- Systems deployed on the Internet where servers are placed at the edge of the network.

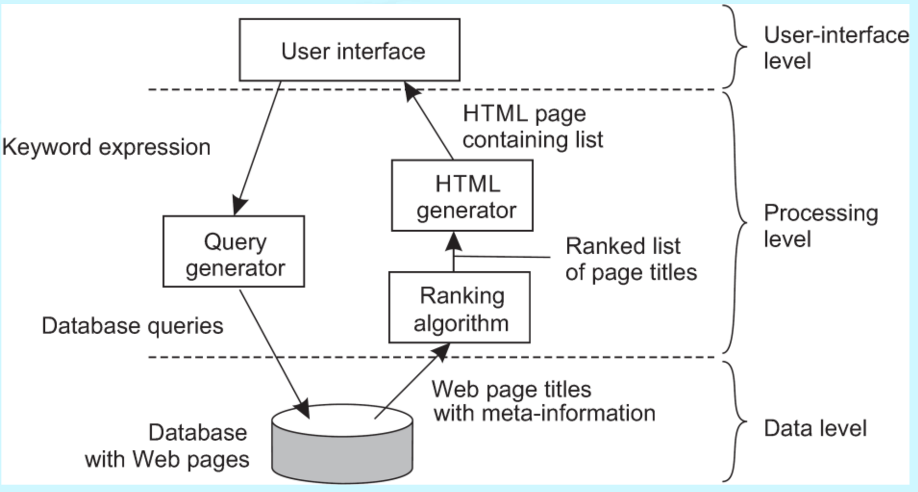


Application Layering (1)

- Client-server applications are usually constructed with a distinction between three levels:
  - User-interface level
  - Processing level
  - Data level
- Clients implement the user-interface level.
- Servers implement the rest.

Application Layering (2)

- The simplified organization of an Internet search engine

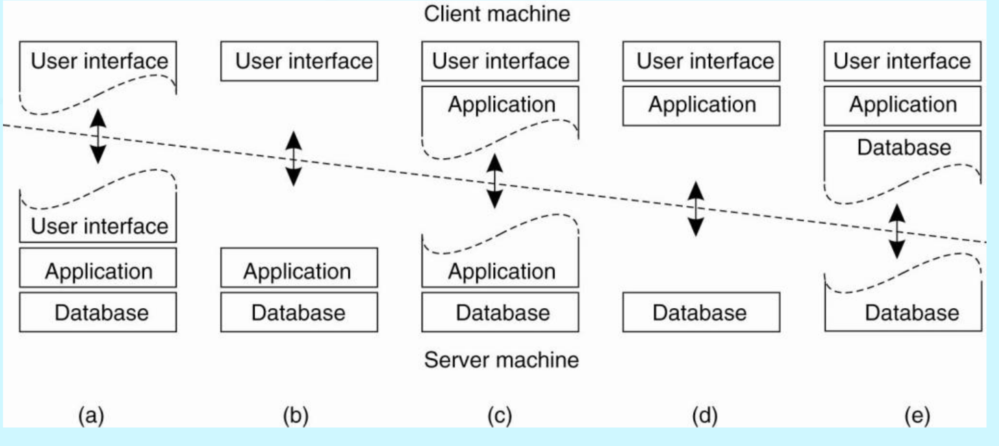


Logical Architecture vs. Physical Architecture

- Physical architecture may or may not match the logical architecture.
- The simplest organization is to have only two types of machines:
  - A client machine containing only the programs implementing (part of) the user-interface level
  - A server machine containing the rest,
    - the programs implementing the processing and data level
- Or could have other partitioning methods.

Alternative Architectures

- Alternative client-server organizations (a) – (e).



Examples of Alternative Architectures

- (a): server-side has some control over UI.
- (c): form checking.
- (d): banking application just uploads transaction.
- (e): Local caching

- Also known as multitiered architectures.
- What's good about moving things out to desktop machines?
- What's bad?

Middleware Solutions

- Middleware
  - Provide higher-level abstraction
  - Hide the heterogeneity
  - Promote interoperability and portability
- Middleware solutions are often based on the architectural models.

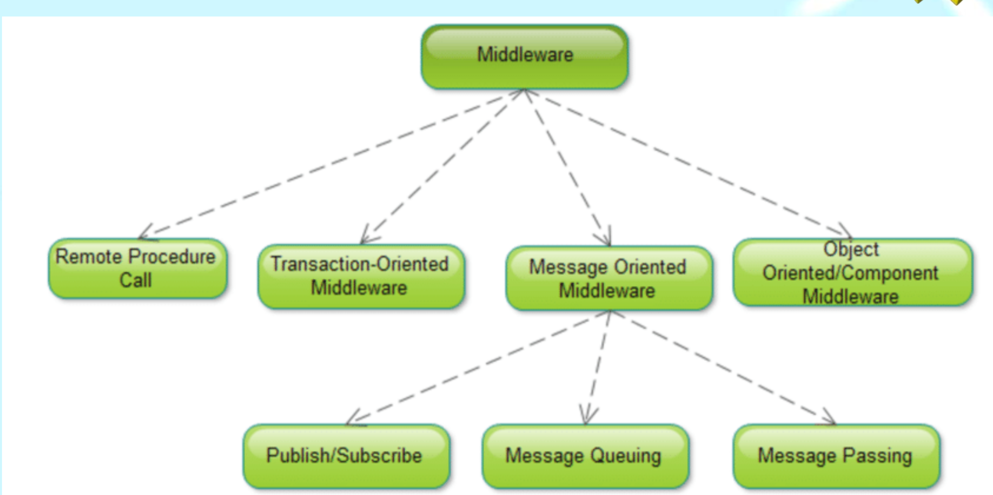
Categories of Middleware

- Distributed objectrs
- Distributed components
- Publish-subscribe systems
- Message queues
- Web services
- Peer-to-peer

Major categories:	*Subcategory*	*Example systems*
Distributed objects (Chapters 5, 8)	Standard	RM-ODP
	Platform	CORBA
	Platform	Java RMI
Distributed components (Chapter 8)	Lightweight components	Fractal
	Lightweight components	OpenCOM
	Application servers	SUN EJB
	Application servers	CORBA Component Model
	Application servers	JBoss
Publish-subscribe systems (Chapter 6)	-	CORBA Event Service
	-	Scribe
	-	JMS
Message queues (Chapter 6)	-	Websphere MQ
	-	JMS
Web services (Chapter 9)	Web services	Apache Axis
	Grid services	The Globus Toolkit
Peer-to-peer (Chapter 10)	Routing overlays	Pastry
	Routing overlays	Tapestry
	Application-specific	Squirrel
	Application-specific	OceanStore
	Application-specific	Ivy
	Application-specific	Gnutella

CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 47

Categories of Middleware



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 48

Fundamental Models

- Abstract models to discuss individual aspects of a distributed system
- Focus on three aspects:
  - Interaction model: Addresses communication and coordination between processes
  - Failure model: Defines and classifies faults and methods of recovery or tolerance
  - Security model: Defines security threats and mechanisms for resisting them

Interaction Model

- Distributed systems are composed of interacting processes.
- Behaviors of processes are captured by distributed algorithms describing the computing steps and message transmission of processes.
- The rate of each process and the timing of message transmission cannot in general be predicted.
- Each process can only access its own state.
- No direct access to the global state of the system.
- No global time.

Communication Channels

- Channels can be modeled in various ways
  - Streams
  - Message passing networks
- Performance characteristics
  - Latency – The delay between the start of message transmission and the beginning of reception.
  - Bandwidth – The total amount of info that can be transmitted over a given time.
  - Jitter – The variation in message delivering time.

Clocks and Timing Events

- Each computer has its own clock.
- Different clocks have different drift rates (the rate a clock deviates from a perfect clock).
- Clock synchronization is to synchronize the clocks of a set of computers.
- In most cases, relative ordering of events is more important than absolute timing.
- It is possible to construct logical clocks for process synchronization.

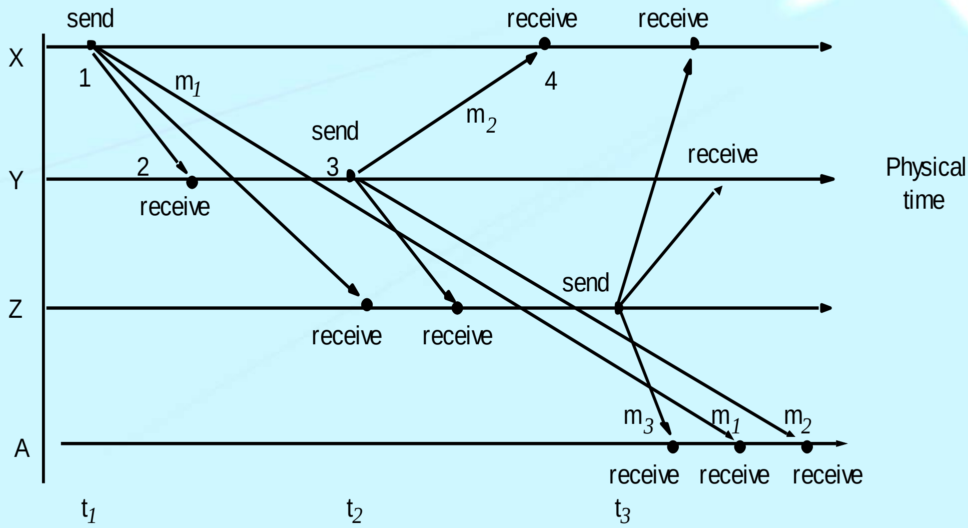
Two Variants of Interaction

- Synchronous distributed systems – Systems in which the following bounds are defined:
  - Each execution step has known lower & upper bounds.
  - Each message transmission is received within known bound.
  - Each process has a local clock with known bound on drift rate.
- In a synchronous system, it is possible to use timeouts in distributed system design.

Two Variants of Interaction

- Asynchronous distributed systems – Systems with no bounds on:
  - Process execution speeds
  - Message transmission delays
  - Clock drift rates
- Actual distributed systems are very often asynchronous.
- Internet is exactly an asynchronous system.

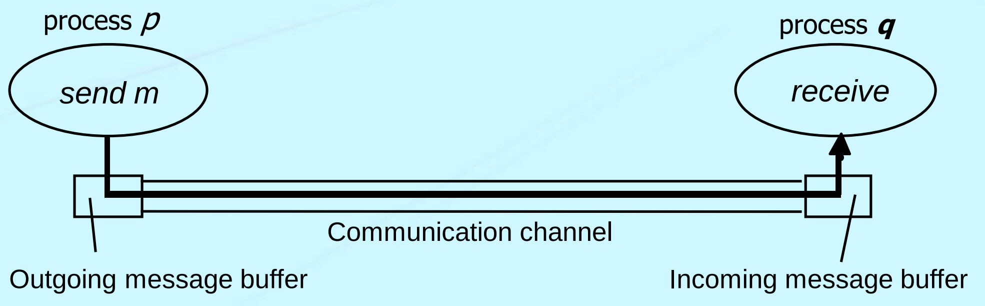
Event Ordering



Failure Models

- A failure model defines the ways in which failure may occur
- Omission Failures - process or channel fails to perform the right actions (more on next slide)
  - process omission failure
  - channel omission failure
- Arbitrary (Byzantine) Failures
- Timing Failures: fail to meet the time bound
- We want to mask failures, i.e. to construct reliable services from components that may exhibit failures.

Processes and Channels



Omission and Arbitrary Failures

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

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.

Masking Failures

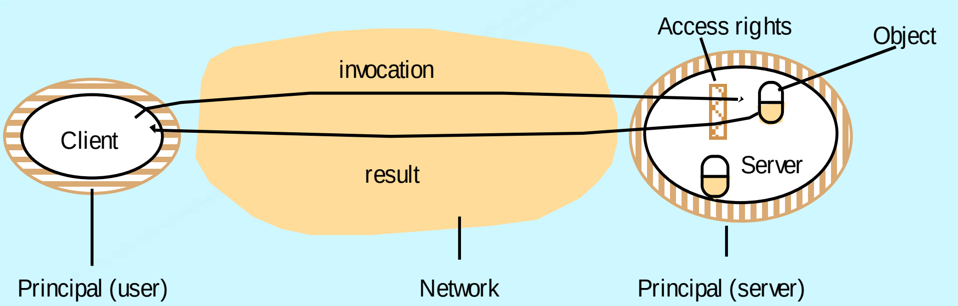
- Failures are unavoidable.
- We can only mask failures
  - By hiding it altogether
  - By converting it into a more acceptable type of failure
- Examples of techniques for masking failures
  - Message checksums
  - Retransmission
  - Replication
  - … (more in later chapters)

Security Models

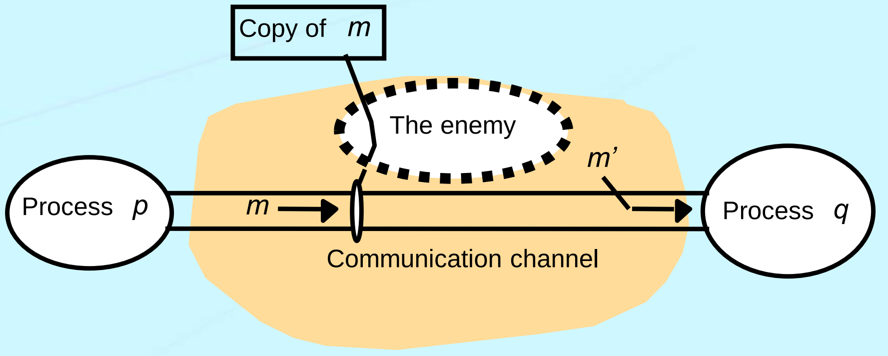
- Threats:
  - threats to processes
  - threats to communication channels
  - denial of service
- Protection:
  - cryptography and shared secrets
  - authentication

Objects and Principals

- Access rights: who can invoke the operations
- Principal: the authority on which an invocation or result is issued.



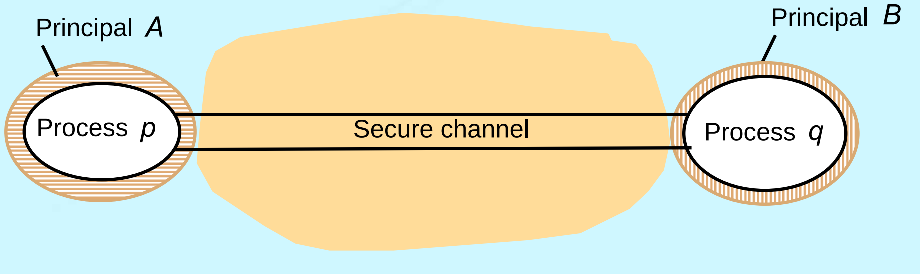
The Enemy



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 63

Secure Channels

- Processes know each others
- Ensure the privacy and integrity of the data transmission
- No message replayed or reordered



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 64

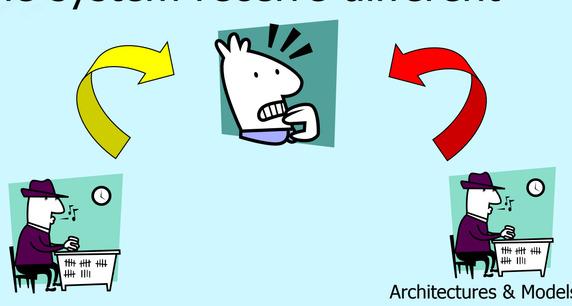
Security Threats

- Security threats can come from any place.
- Two interesting examples
  - Denial of service (DOS) – Enemy interferes with the activities of authorized users by making excessive and pointless accesses in a network.
  - Mobile code
    - Mobile code raises new and interesting security problems.
    - Can easily play a Trojan horse role.
    - Can be carried in many ways: emails, Web pages, applets, Active X, …

CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 65

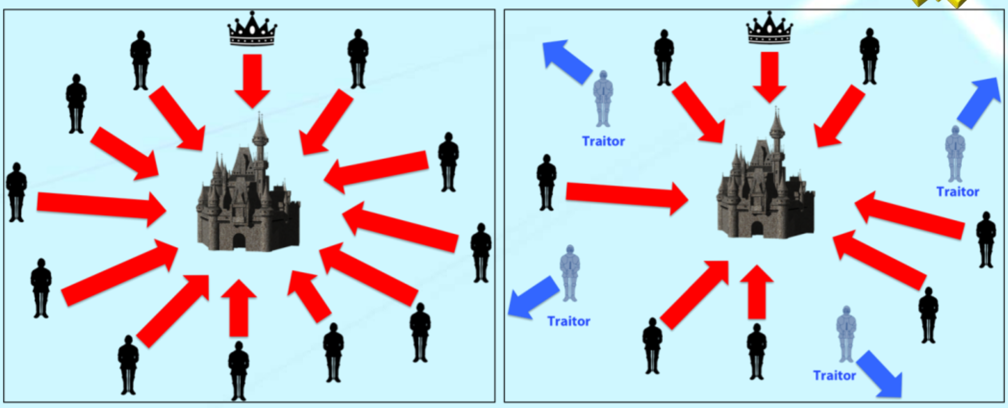
Homework 1: Byzantine Generals Problem

- This is a classic problem in distributed system design.
- In a distributed system, failed components can send conflicting information.
- Different parts of the system receive different information.



CSIE52400/CSIEM0140 Distributed Systems Architectures & Models 66

Attack or Retreat ?



Coordinated Attack Leading to Victory **Uncoordinated Attack Leading to Defeat**

How to reach the same agreement among loyal generals?

The Classic Problem

- Each division of the Byzantine army are directed by its own general.
- Some of the generals may be traitors.
- Generals communicate with each other by reliable messengers.
- Requirements:
  - All loyal generals decide upon the same plan of action.
  - A small number of traitors cannot cause the loyal generals to adopt a bad plan.

Variations and Impossibility

- How many traitors does it take to make the agreement among loyal generals impossible ?
- What if the messengers were not reliable ?
- There are several variant problems. Can you think out a different one by yourself ?
- Do not try to look for answer from the net. It will loose all the fun of this assignment.

Reference

- Lamport, L., Shostak, R., Pease, M. "The Byzantine Generals Problem". ACM TOPLAS. Vol 4. Num. 3, July, 1982.
- There are several variant problems based on the classic problem.
- Due date: Apr 7, 2020