IT4371: Distributed Systems

Spring 2016

Architectural Models of Distributed Systems

Dr. Nguyen Binh Minh

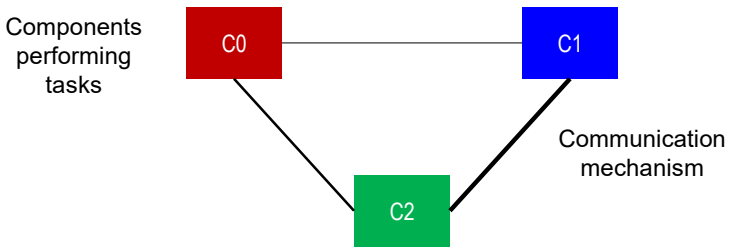
Department of Information Systems
School of Information and Communication Technology
Hanoi University of Science and Technology

Today…

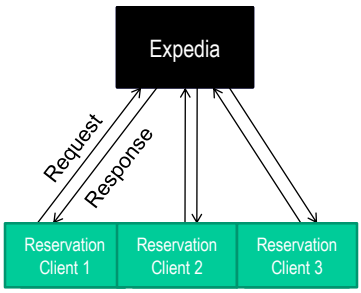
- Last Session:
  - Trends and challenges in Distributed Systems
- Today's session:
  - Architectural Models of Distributed Systems

A Distributed System

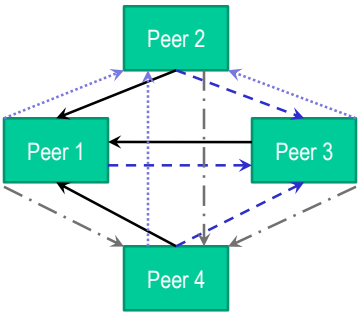
- A distributed system is simply a collection of hardware or software components that communicate to solve a complex problem
- Each component performs a "task"



Bird's Eye View of Some Distributed Systems



Google Search
Airline Booking



Bit-torrent
Skype

- How would one classify these distributed systems?

Classification of Distributed Systems

What are the entities that are communicating in a DS?

a) Communicating entities

How do the entities communicate?

b) Communication paradigms

What roles and responsibilities do they have?

c) Roles and responsibilities

How are they mapped to the physical distributed infrastructure?

d) Placement of entities

Classification of Distributed Systems

What are the entities that are communicating in a DS?

a) Communicating entities

How do the entities communicate?

b) Communication paradigms

What roles and responsibilities do they have?

c) Roles and responsibilities

How are they mapped to the physical distributed infrastructure?

d) Placement of entities

Communicating Entities

What entities are communicating in a DS?

- System-oriented entities
  - Nodes
  - Processes
  - Threads
- Problem-oriented entities
  - Objects (in *object-oriented programming* based approaches)

Classification of Distributed Systems

What are the entities that are communicating in a DS?

a) Communicating entities

How do the entities communicate?

b) Communication paradigms

What roles and responsibilities do they have?

c) Roles and responsibilities

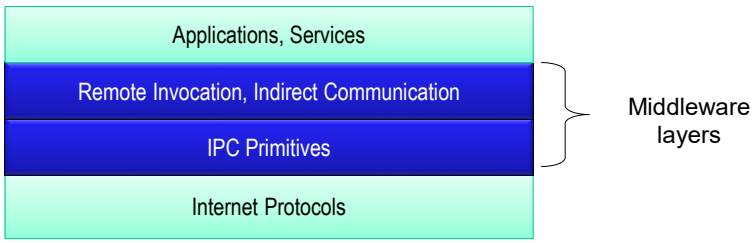
How are they mapped to the physical distributed infrastructure?

d) Placement of entities

Communication Paradigms

Three types of communication paradigms

- Inter-Process Communication (IPC)
- Remote Invocation
- Indirect Communication



Inter-Process Communication (IPC)

Relatively low-level support for communication

- e.g., Direct access to internet protocols (Socket API)

Advantages

- Enables seamless communication between processes on heterogeneous operating systems
  - Well-known and tested API adopted across multiple operating systems

Disadvantages

- Increased programming effort for application developers
  - Socket programming: Programmer has to explicitly write code for communication (in addition to program logic)
  - Space Coupling (Identity is known in advance): Sender should know receiver's ID (e.g., IP Address, port)
  - Time Coupling: Receiver should be explicitly listening to the communication from the sender

Remote Invocation

An entity runs a procedure that typically executes on an another computer
without the programmer explicitly coding the details for this remote interaction

- A middleware layer will take care of the raw-communication

Examples

- Remote Procedure Call (RPC) – Sun's RPC (ONC RPC)
- Remote Method Invocation (RMI) – Java RMI

Remote Invocation

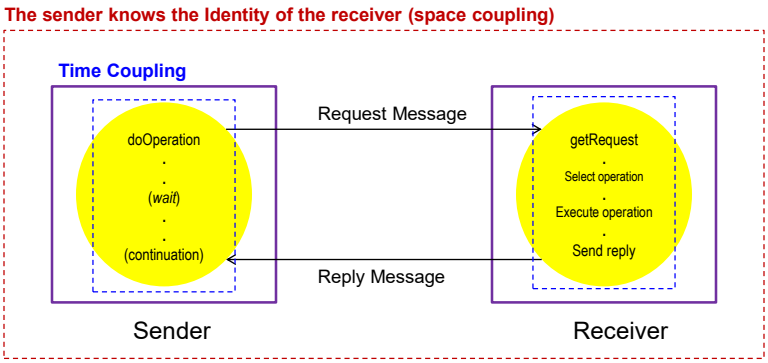
Advantages:

- Programmer does not have to write code for socket communication

Disadvantages:

- Space Coupling: Where the procedure resides should be known in advance
- Time Coupling: On the receiver, a process should be explicitly waiting to accept requests for procedure calls

Space and Time Coupling in RPC and RMI



Indirect Communication Paradigm

Indirect communication uses middleware to:

- Provide one-to-many communication
- Some mechanisms eliminate space and time coupling
  - Sender and receiver do not need to know each other's identities
  - Sender and receiver need not be explicitly listening to communicate

Approach used: Indirection

- Sender → A middle-man → Receiver

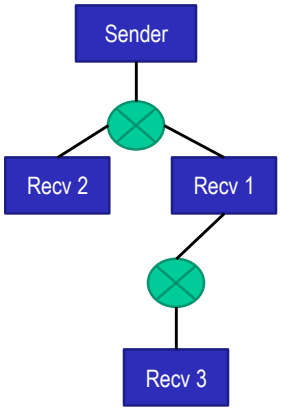
Types of indirect communication

1. Group communication
2. Publish-subscribe
3. Message queues

1. Group Communication

One-to-many communication
- Multicast communication

Abstraction of a group
- Group is represented in the system by a *groupId*
- Recipients join the group
- A sender sends a message to the group which is received by all the recipients



1. Group Communication (cont'd)

Services provided by middleware
- Group membership
- Handling the failure of one or more group members

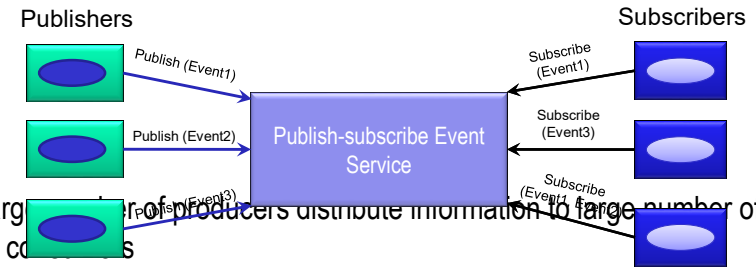
Advantages
- Enables one-to-many communication
- Efficient use of bandwidth
- Identity of the group members need not be available at all nodes

Disadvantages
- Time coupling

2. Publish-Subscribe

An event-based communication mechanism

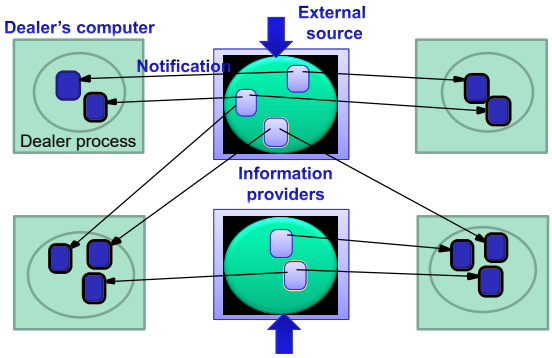
- Publishers publish events to an event service
- Subscribers express interest in particular events



Large number of producers distribute information to large number of consumers

2. Publish-Subscribe (cont'd)

Example: Financial trading



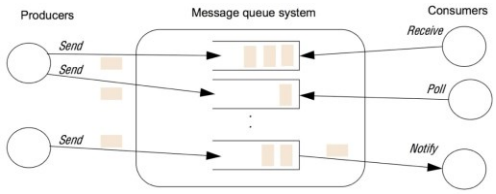
3. Message Queues

A refinement of Publish-Subscribe where

- Producers deposit the messages in a queue
- Messages are delivered to consumers through different methods
- Queue takes care of ensuring message delivery

Advantages

- Enables space decoupling
- Enables time decoupling



Recap: Communication Entities and Paradigms

Communicating entities (what is communicating)	
System-oriented	*Problem-oriented*
• Nodes • Processes • Threads	• Objects

Communication Paradigms (how they communicate)		
IPC	*Remote Invocation*	*Indirect Communication*
• Sockets	• RPC • RMI	• Group communication • Publish-subscribe • Message queues

Classification of Distributed Systems

What are the entities that are communicating in a DS?

a) Communicating entities

How do the entities communicate?

b) Communication paradigms

What roles and responsibilities do they have?

c) Roles and responsibilities

How are they mapped to the physical distributed infrastructure?

d) Placement of entities

Roles and Responsibilities

In DS, communicating entities take on roles to perform tasks

Roles are fundamental in establishing overall architecture

- Question: Does your smart-phone perform the same role as Google Search Server?

We classify DS architectures into two types based on the roles and responsibilities of the entities

- Client-Server
- Peer-to-Peer

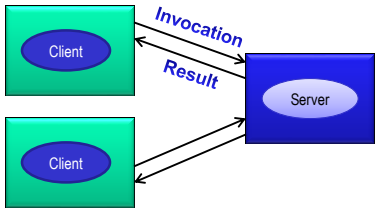
Client-Server Architecture

Approach:

- Server provides a service that is needed by a client
- Client requests to a server (invocation), the server serves (result)

Widely used in many systems

- e.g., DNS, Web-servers



Client-Server Architecture: Pros and Cons

Advantages:

- Simplicity and centralized control
- Computation-heavy processing can be offloaded to a powerful server
  Clients can be "thin"

Disadvantages

- Single-point of failure at server
- Scalability

Peer to Peer (P2P) Architecture

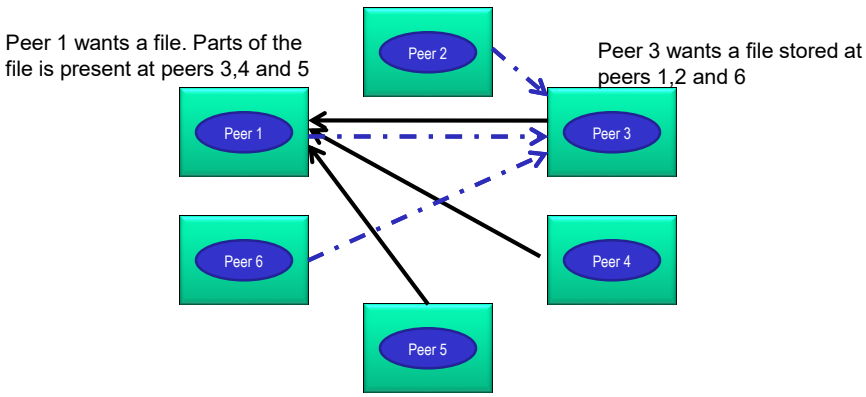
In P2P, roles of all entities are identical

- All nodes are peers
- Peers are equally privileged participants in the application

e.g.: Napster, Bit-torrent, Skype

Peer to Peer Architecture

Example: Downloading files from bit-torrent



Architectural Patterns

Primitive architectural elements can be combined to form various patterns

- Tiered Architecture
- Layering

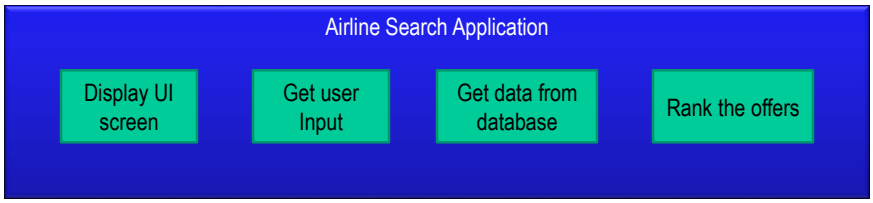
Tiered architecture and layering are complementary

- Layering = vertical organization of services
- Tiered Architecture = horizontal splitting of services

Tiered Architecture

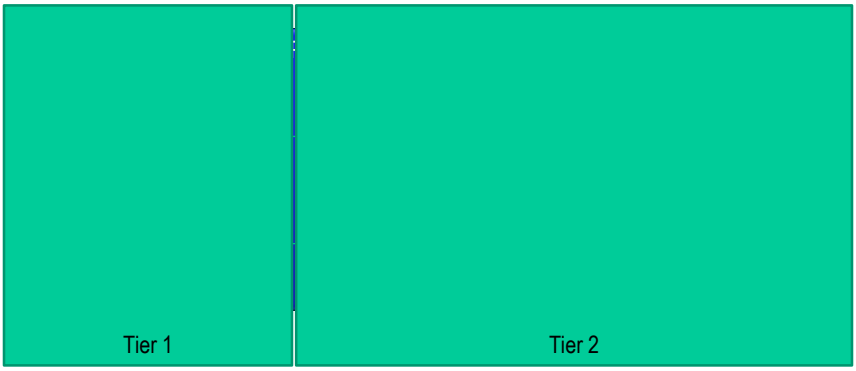
A technique to:

1. Organize the functionality of a service, and
2. Place the functionality into appropriate servers



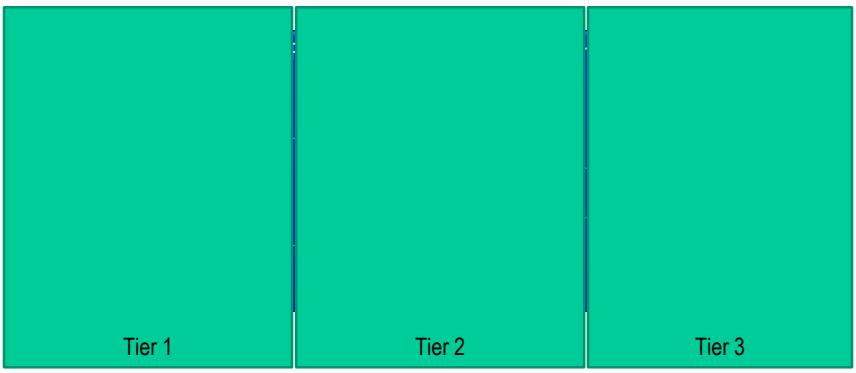
A Two-Tiered Architecture

How do you design an airline search application:

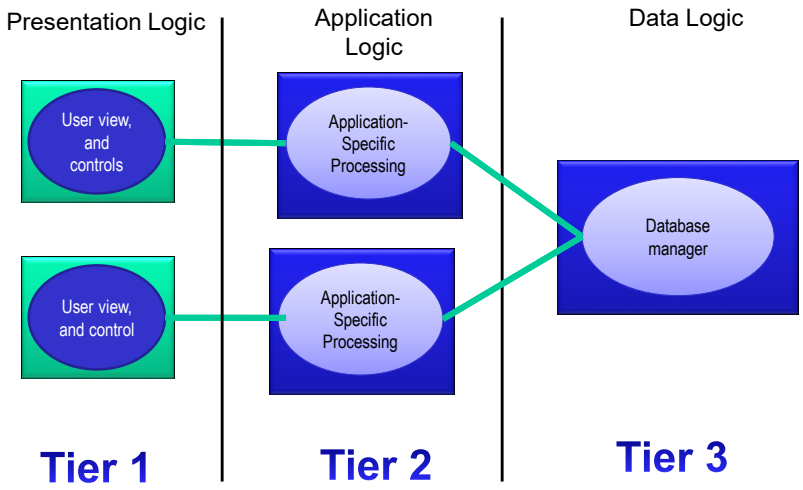


A Three-Tiered Architecture

How do you design an airline search application:



A Three-Tiered Architecture



Three-Tiered Architecture: Pros and Cons

Advantages:

- Enhanced maintainability of the software (one-to-one mapping from logical elements to physical servers)
- Each tier has a well-defined role

Disadvantages

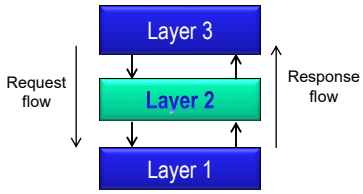
- Added complexity due to managing multiple servers
- Added network traffic
- Added latency

Layering

A complex system is partitioned into layers
- Upper layer utilizes the services of the lower layer
- A vertical organization of services

Layering simplifies design of complex distributed systems by hiding the complexity of below layers

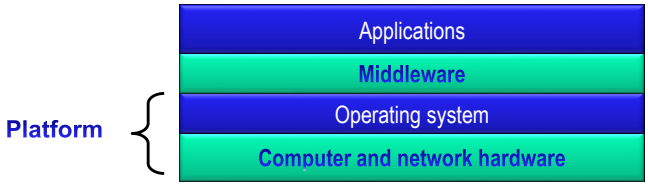
Control flows from layer to layer



Layering – Platform and middleware

Distributed Systems can be organized into three layers

1. Platform
   - Low-level hardware and software layers
   - Provides common services for higher layers
2. Middleware
   - Mask heterogeneity and provide convenient programming models to application programmers
   - Typically, it simplifies application programming by abstracting communication mechanisms
3. Applications



Classification of Distributed Systems

What are the entities that are communicating in a DS?

a) Communicating entities

How do the entities communicate?

b) Communication paradigms

What roles and responsibilities do they have?

c) Roles and responsibilities

How are they mapped to the physical distributed infrastructure?

d) Placement of entities

Placement

Observation:

- A large number of heterogonous hardware (machines, networks)
- Smart mapping of entities (processes, objects) to hardware helps performance, security and fault-tolerance

"Placement" maps entities to underlying physical distributed infrastructure.

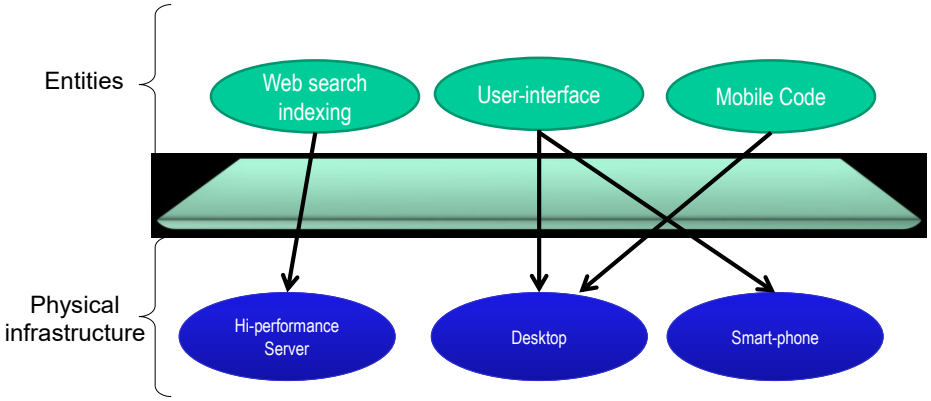
Placement should be decided after a careful study of application characteristics

Example strategies:

Mapping services to multiple servers

Moving the mobile code to the client

Placement



Recap

We have covered primitive architectural elements

- Communicating entities
- Communication paradigms of entities
  - IPC, RMI, RPC, Indirect Communication
- Roles and responsibilities that entities assume, and resulting architectures
  - Client-Server, Peer-to-Peer, Hybrid
- Placement of entities

Next Class

- Identify different types of networks

Describe networking principles such as layering, encapsulation and packet-switching

Examine how packets are routed and how congestion is avoided

Analyze scalability, reliability and fault-tolerance of Internet