Announcements

- Email me your survey: See the Announcements page
- Today
  - Conceptual overview of distributed systems
  - System models
- Reading
  - Today: Chapter 2 of Coulouris
  - Next topic: client-side processing (HTML, CSS)
- Take a break around 10:15am
- Ack: Some figures are from Coulouris
Distributed system models

• Three ways to view distributed models

  • Physical models based on types of computers and interconnecting networks

  • Architectural models in terms of computational and communication tasks performed

  • Fundamental models based on fundamental properties
    • Interaction models
    • Failure models
    • Security models
Physical models
### Physical models

- Generations of distributed systems

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

- Emergence of mobile computing requiring added capabilities such as service discovery and support for spontaneous interoperation
- Emergence of ubiquitous computing with computers are embedded in everyday objects
- Emergence of cloud computing with pools of nodes providing a given service
- Ultra-Large-Scale (ULS) distributed systems: system of systems (cf. the Internet as a network of networks)
Architectural models
Architectural models

Start with

- Architectural elements in distributed systems
- Architectural patterns used in developing distributed systems
- Middleware platforms available to support programming styles useful for distributed architecture
Architectural elements

- Communicating entities
- Communication paradigms
- Roles and responsibilities
- Placement
Communicating entities

- Processes
- Objects
- Components
- Web services
Communication paradigms

• Interprocess communication
  • Low-level support for communication between processes, e.g., message-passing primitives

• Remote invocation
  • Request-reply protocols
  • Remote procedure calls
  • Remote method invocation

• Indirect communication
  • Group communication
  • Publish-subscriber system
  • Distributed shared memory
Roles and responsibilities

- Client-server architecture
- Peer-to-peer architecture
Client-server architecture

- Most common in distributed systems

- **Client**: process wishing to access data, use resources, or perform operations on a different computer

- **Server**: process managing data and all other shared resources among servers and clients, allowing client access to resources, and perform computation

- **Interaction**: request/reply message pairs
Client-server architecture (cont.)

- Servers may in turn be clients of other servers
- Servers, examples:
  - File servers
  - DNS servers
  - Web servers
  - Database servers
  - Search engines
- Clients, examples:
  - Web browsers
  - Web servers (wrt database servers)
  - Search engines (web crawlers)
- Scalability limited by the capacity of the server and bandwidth of network connections
Clients invoke individual servers

invocation = request
result = response
Peer-to-peer architecture

- Processes play similar roles interacting cooperatively as peers
- Exploit resources in many participating computers to fulfill a given task in an application
- Applications are composed of many peer processes running on separate computers connected by application-dependent communication requirements
- Large number of data objects are shared while individual computer holding only a small part of them
- Objects are replicated in multiple computers for load distribution and resilience
- More complex than the client-server architecture
Peer-to-peer architecture
Placement

- Mapping of objects or services on to the underlying physical distributed infrastructure
- Mapping of services to multiple servers
- Caching
- Mobile code
- Mobile agents
A service provided by multiple servers

- For better performance and reliability
- Multiple-server architecture: cluster used for highly scalable web services, e.g., search engines, online stores
  - Service processing can be partitioned or replicated among them
Architectural patterns

- Layering
- Tiered architecture
- Thin clients
Layering

- Software and hardware service layers in distributed systems

Diagram:

1. Applications, services
2. Middleware
3. Operating system
4. Computer and network hardware

Platform
Tiered architecture

- Two-tier and three-tier architecture

![Diagram of tiered architecture](image)
Thin clients

- A software layer supporting a window-based user interface on a local computer while executing an application program on a remote server
- Possible performance problem if the traffic is high over the network, e.g., images
Middleware

- Achieves transparency of heterogeneity at platform level for interoperability and portability
- Provides a higher-level programming abstraction
- Abstractions such as
  - Remote method invocation (RMI)
  - Communication between processes
  - Notification of events
  - Partitioning, placement, and retrieval of shared data
  - Replication of shared data
- Examples:
  - CORBA
  - Java RMI
  - Web services
Fundamental models
Fundamental models

- Different system models share some fundamental properties: processes communicating by messages!
- Design requirements: performance, reliability, security

- Interaction model
- Failure model
- Security model
Interaction model

- Performance of communication channels
- Computer clocks and timing events
- Two variants of the interaction model
  - Synchronous
  - Asynchronous
Failure model

- Correctness is affected by a fault in any of the participating computers or in the network connecting them
  - Process crashes
  - Communication channel not transporting a message from sender’s outgoing message buffer to the receiver’s incoming message buffer
  - Process sets a wrong value in its data items or returns a wrong value in response to an invocation
  - Process exceeds the bounds on the interval between 2 steps
  - A message’s transmission takes longer than the stated bound
- The model defines and classifies the faults (hardware and software)
- This provides a basis for the analysis of their potential effects and for the design of systems that are able to tolerate the faults of each type while continuing to run correctly
Security model

- The modular nature of distributed systems and their openness exposes them to attack by both external and internal agents
- The model defines and classifies the forms that such attacks may take
- It provides a basis for the analysis of threats to a system and for the design of systems that are able to resist them

- Protecting objects
- Securing processes and their interactions
- The enemy
- Defeating security threats
Protecting objects

- Objects and principals

Diagram:
- Client
- Principal (user)
- Network
- Server
- Principal (server)
- Object
- Access rights

Invocation
Result
The enemy

The enemy

Copy of \( m \)

Communication channel

Process \( p \)

m

Process \( q \)

m'

The enemy
Defeating security threats

- Cryptography and shared secrets
- Authentication
- Secure channels using encryption and authentication
Secure channels

Principal A

Process p

Secure channel

Process q

Principal B
Our main concern next . . .

- Multi-tier, client-server architecture

- Programming for the web
  - Client-side programming
  - Server-side programming
Next topic

- Client-side programming: HTML, CSS