Lecture II: Architectural Models

CMPT 401 Summer 2007

Dr. Alexandra Fedorova
Introductions

- Architectural model is an *abstract view* of a distributed system
- Models are constructed to simplify reasoning about the system
- A model of a DS is expressed in terms of
  - Components
  - Placement of components
  - Interactions among components
Component of a Distributed System

• Component of a distributed system is a *process*
• A process is *running program*
• Examples:
  – Server process – a program executing server code
  – Client process – a program executing client code
• Processes interact by sending each other messages
Outline

• System Architecture Models
  – Client-Server
  – Peer-to-Peer
  – Variations
• Interaction Models
• Failure Models
• Security Models
Client invoke individual servers

Key:
- Process:
- Computer:
Client-Server Architecture II

• Clients send requests to servers (i.e., invocation)
• Servers send responses to clients (i.e., result)
• Servers may be clients of other servers
  – A web server is often a client of a file server
  – An Internet service is a client of a DNS server – a server that translates DNS names to IP addresses
• Potential problem: a single server is a scalability bottleneck and a single point of failure
Peer-to-Peer Architecture I

A distributed application based on peer processes
Peer-to-Peer Architecture II

- All processes play similar roles – i.e., they interact as peers
- No central component – potentially better scalability and resiliency to failures
- Use the power of modern desktops to implement a large-scale distributed system
- Examples: Napster, Kazaa, Skype, Bittorrent
Architectural Variations

- Services provided by multiple servers
- Proxy servers and caches
- Mobile code
- Mobile agents
- Network computers
- Thin clients
- Mobile devices
Services by multiple servers

- Multiple servers provide services to clients
- Servers may partition the service objects or replicate them
- WWW: partitioned objects
- Sun NIS: replica of a password file maintained at each server
- Computing clusters
Proxy Servers and Caches

- A cache is a store of recently used data objects that is closer than the main store
- A newly accessed object is added to the cache
- When that object is accessed again, it is fetched from the cache, if there is an up-to-date copy in the cache
- Proxy servers intercept communication with the real server to provide faster service (e.g., deliver cached data), better security (e.g., a proxy configured as a firewall)
Mobile Code

Web applets

a) client request results in the downloading of applet code

b) client interacts with the applet

• Code that is downloaded from a remote machine (e.g., a server) and is run in a local machine (e.g., a client)
• Example: Java applet
• Reason: provide better interactive experience
Mobile Agents

• A running program (both code and data) that travels from one computer to another

• Example: a worm
  – Used to attack computer systems
  – Used for system administration
  – The original work at Xerox PARC: to make use of idle computers for a resource-intensive computation
Network Computers

• Does not rely (or relies minimally) on locally installed software

• Downloads operating system and applications from a remote computer

• Applications are run locally, but files are managed on a remote server

• Users can migrate from one network computer to another
Thin Clients

• Similar to a network computer
• Instead of downloading code to the user computer, it runs it on a compute server
• Software layer provides a window-based interface to the client (X Windows)
Mobile Devices

- Cellular phones
- PDAs
- Laptops
- Wearable devices
- Mobile sensors
Architecture Models: Summary

• Classified according to roles of components:
  – Client-server
  – Peer-to-peer

• Variations according to modes of interactions
  – Services provided by multiple servers
  – Proxy servers and caches
  – Mobile code
  – Mobile agents
  – Network computers
  – Thin clients
  – Mobile devices
Outline

• System Architecture Models
• Interaction Models
• Failure Models
• Security Models
Interaction Models

- Represent *communication* and *coordination* among the processes
- Must account for:
  - Performance of communication channels (communication delays determine how well the system works)
  - Differing notions of time across system components
- We will look at the following interaction models
  - Synchronous
  - Asynchronous
Communication Delays

• Message transmission delay is comprised of:
  – Network latency: the time for a bit of information to travel from source network interface to destination network interface
  – Delay in accessing network: i.e., how long it takes for the network to become available
  – Operating system delays: the time taken by operating system services at both ends of communication channel
Clocks and Timing Events

• Each computer has its own clock
• Reading of a local clock will differ from the real clock, because a clock drifts
• Clock drift rates differ from one another
Synchronous Interaction Model

• In a synchronous distributed system there are known bounds on:
  – Time to execute a step of a process
  – Message transmission time
  – Clock drift rate (i.e., the difference between local clock and the real clock)

• To guarantee bounds, one would need to:
  – Know resource requirements of each process
  – Guarantee those resources to the process (including network capacity)
  – Guarantee bounds on clock drift
  – Eliminate the possibility of certain failures

• Synchronous distributed systems are rare, because it is difficult to guarantee such bounds

• Synchronous system models are relatively easy to reason about
Asynchronous Interaction Model

• No bounds on delays determining the length of interaction
  – No bounds on process execution time
  – No bounds on message transmission delays
  – No bounds on clock drift rates
• The Internet is an asynchronous system
• Despite this uncertainty, many distributed systems are useful
Outline

- System Architecture Models
- Interaction Models
- Failure Models
- Security Models
Failure Models

• Types of failures
  – Omission failures
  – Byzantine failures
  – Timing failures
• Masking failures
Omission Failures

• An omission failure occurs when a process stops sending/receiving messages.

• Types of omission failures
  – Process omission failures: the process has crashed
  – Communication omission failures: message has not been delivered
Process Omission Failures

• A process crash is called a **fail-stop failure**
• In a synchronous system a fail-stop failure is determined via timeouts
• In an asynchronous system it is **impossible** to detect reliably that the process has crashed
• If the process is not responding it could have crashed or it could be just running slowly
Structure of a Communication Channel

Processes and channels

process $p$

$send \ m$

Communication channel

Outgoing message buffer

Incoming message buffer

process $q$

receive

©Pearson Education 2001
Communication Omission Failure

- Messages can be lost at the sender, at the receiver and in the network:
  - Receive omission: message is lost on the receiving side
  - Send omission: message is lost while sending
  - Channel omission: message is lost between sender and receiver

- A common cause for message loss:
  - Message buffer overflow due to system being busy
  - Systems drop messages deliberately when their buffers fill up
  - Clever algorithms to decide when to drop messages
Failure Models

• **Types of failures**
  – Omission failures
  – Byzantine failures
  – Timing failures

• **Masking failures**
Byzantine Failures

- Arbitrary failures
  - A process arbitrarily skips processing steps
  - A process takes unintended processing steps
  - Corrupted message contents

- Arbitrary failures can be caused by:
  - Malicious behaviour (attack)
  - Software bugs

- A byzantine failure cannot be reliably detected
Timing Failures

- Apply to synchronous systems
- Relevant for multimedia applications
- Clock failure: a process’s local clock exceeds the bounds on the drift rate from real time
- Performance failure:
  - Process exceeds the bounds on the interval between two steps
  - A message transmission takes longer than the stated bound
Impossibility of Agreement

- **Theorem:** In an asynchronous distributed system it is impossible to reach an agreement in the presence of failures.
Impossibility of Agreement: Proof

1. Suppose agreement is achieved via a sequence of messages
2. Suppose agreement could be reached in spite of message loss
3. Then it must be possible to eliminate the last message in the sequence and still reach the agreement
4. Now you have one fewer messages in a sequence
5. If you keep applying the above argument, you will end up with the sequence of zero messages
6. This is a contradiction: there must be at least one message in the sequence
Masking Failures

• Conversion from one type of failure to another
  – When a corrupted message is detected, the process acts as if the message has been lost
  – Byzantine $\rightarrow$ Omission

• Handling omission failures via retransmission

• Handling fail-stop failures via
  – Replication
  – Restarting the process, restoring its memory state
Outline

- System Architecture Models
- Interaction Models
- Failure Models
- Security Models
Security Models

- **Adversary**: a process that sends messages that would not be sent by a legitimate process
  - The goal is to violate integrity or secrecy of data or to disrupt normal functioning of the system

- **Types of security threats**:
  - Threats to processes
  - Threats to communication channels
  - Denial of service (DoS) attacks
Threats to Processes

• Forged identity
  – The client adversary masquerades as a legitimate user and obtains secret information from the server
  – The server adversary masquerades as a legitimate server and sends a wrong response to the client

• Taking over the system (i.e., a hacked system)
  – An adversary exploits system vulnerability
  – Sends a packet that causes the server to execute the program belonging to the adversary (viruses or buffer overflow attacks)
  – The adversary causes the byzantine failure
Threats to Communication Channels

• Interception of messages:
  – Watch messages sent over the network, read their contents: violation of privacy and secrecy (i.e., someone reads my e-mail)

• Injection of messages:
  – Save a copy of a legitimate message and later “replay” it on the network
  – E.g., send a message asking to charge the credit card multiple times
Denial of Service Attacks

- Flood the system with pointless messages to prevent normal operation of the system
- Causes the system to run very slowly
- Many systems are not designed to handle performance spikes: CNN server became unresponsive on 9/11
Dealing with Security Failures

- **Encryption**
  - Scramble the message so as to hide its content, i.e., encrypt
  - Message can be decrypted using a *key*. A key is usually a large number that is difficult to guess.

- **Authentication**
  - Encrypt a part of the message; in the encrypted part provide enough information to guarantee authenticity
  - Enabled by use of shared secrets and encryption

- **Secure channels**
  - Built on top of regular channels using encryption
  - Communicating processes reliably know each others’ identity
  - Transmitted cannot be tampered with
  - Each message includes a physical or logical timestamp to prevent reordering or replay
  - Examples: Virtual Private Network (VPN), Secure Sockets Layer (SSL)
Summary I

- **System architecture models**
  - Client/server
  - Peer-to-peer
  - Variations: mobile devices, network computer, thin client etc.

- **Interaction models**
  - Synchronous system: known bounds on clock drifts and message delays
  - Asynchronous system: no such bounds
Summary II

• Failure Model
  – Omission failures
    • Process omission: fail-stop – cannot be reliably detected in an asynchronous system
    • Communication omission: Send/receive omission, channel omission
  – Byzantine failures
    • Bugs
    • Message corruption
    • Hardest to deal with
  – Timing failures
    • Apply to synchronous systems

• In an asynchronous system agreement cannot be reached in presence of failures
Summary III

• Threat model
  – Adversary
  – Threats to processes (spoofing identity, hacking the system)
  – Threats to communication channels (intercepting messages, replaying messages)
  – Denial of service attacks: prevent proper functioning of the system by sending useless messages

• Security threads addressed via encryption, authentication and secure channels