CISC 434: Distributed Systems Architecture - II

- Architectural Styles
- System Architectures
  - Centralized Architectures
  - Decentralized Architectures
  - Hybrid Architectures
- Self-Management in Distributed Systems
Decentralized Architectures - Some Distributions

- **Vertical distribution**
  - Organizing a client-server application as a multi-tiered architecture
  - Place logically different components on different machines

- **Horizontal distribution**
  - Distribution of the clients and servers - more common in modern architecture
  - A client or server may be physically split up into logically equivalent parts operating on its own share of the complete data set - load balancing
Horizontal Distribution – An Example

An example system for Web services

Front end handling incoming requests

Replicated Web servers each containing the same Web pages

Requests handled in round-robin fashion

Disks

Internet
Peer-to-peer Distribution

- Supports horizontal distribution
- No server at all – for simple collaborative applications
- Each process performs all the functions
- Most processes interactions are symmetric: each process is a client and a server at the same time
- Architectures depend on the organization of the processes in an overlay network
Peer-to-peer Distribution – contd.

Overlay Networks

- A logical network in which every process has a local list of other peers that it can communicate with
- A process can communicate with another process through a message, if there is any available communication channel between them
- Two types – structured and unstructured
Peer-to-peer Distribution – contd.

Structured Peer-to-Peer Architectures
- The overlay network is constructed using a deterministic scheme
- The scheme is used for routing messages between processes (e.g. a fixed ring network)

Unstructured Peer-to-Peer Architectures
- Constructs random graph – each node maintains a list of random neighboring nodes
- The neighbor list is also referred to as partial view
- Search algorithms are needed for locating data or other nodes
Superpeers

- Nodes that maintain an index or act as a broker
- A hierarchical organization of nodes
Hybrid Architectures

- Combination of client-server systems and decentralized architectures
  - **Edge Server Systems**
    - Servers are placed at the edge of the network
    - The edge is formed by the boundary between enterprise networks and the ISP
    - An ISP can be considered as an edge server, where the end users connect to the Internet through their ISP
  - **Collaborative Distributed System** – facilitate collaboration between the communicating parties
Example Edge-Server System

Internet as consisting of a collection of edge servers
Example Collaborative Distributed System

**BitTorrent**

- A peer-to-peer file downloading system to ensure collaboration
- An end user downloads chunks of the file from other users until the downloaded chunks can form the complete file
- A file can be downloaded only when the downloading client is providing content to someone else – “tit-for-tat” or “reward-punishment” approach
Example Collaborative Distributed System

**BitTorrent - Downloading a file**

- User accesses a **global directory** (some well known websites) which have references to **.torrent files** (information to download a file - tracker)
- **Tracker** is a server that keeps track of active nodes that have chunks of the requested file
- An active node will be forced to help others by sharing the (chunks) files requested from it
Example Collaborative Distributed System

The working principle of BitTorrent

- If node P notices that Q is downloading more than it is uploading, P can decide to decrease the rate at which it sends data to Q.
Self-Management in Distributed Systems

- A.K.A. autonomic computing, self-healing software, self-star system
- Generally organized as feedback-control loops. Loops contain
  - A monitoring component to measure the behavior of the distributed system
  - An analysis component to see whether anything needs to be adjusted
  - A collection of various instruments for changing the behavior
The Feedback Control Model

An example logical organization of a feedback control system

Uncontrollable parameters (disturbance / noise)

Initial configuration → Corrrections → Core of distributed system → Observed output

Adjustment measures

Reference input → Metric estimation → Measured output

Adjustment triggers
Software Monitors

Objectives

- Check the correctness of the results of each run of the system while the system is operating
- Expected not to interrupt the normal operation of the system, i.e., I/O are not controlled
- Online testing (failure detection) and fault localization
- Issue an advance warning, adapt, fault tolerance

Some General Issues

- Defining (correct) expected behavior
- Monitored system visibility
- Target system observation non-determinism
- Detection latency
- Monitoring overhead
Fault Tolerance Phases

- Error Detection
- Damage Confinement and Assessment
- Error Recovery
- Fault Treatment and Continued Service
Example: Automatic Component Repair Management in Jade

- **Jade – Java Agent Development Framework**
  - Applicable to component-based servers
  - Detects component failure and have them automatically replaced

- **Steps required in a repair procedure (after the detection)**
  - Terminate every binding between a component on a non faulty node, and a component on the node that just failed (damage confinement)
  - Request the node manager to start and add a new node to the domain
  - Configure the new node with exactly the same components as those on the crashed node (error recovery)
  - Re-establish all the bindings that were previously terminated (continued service)
Summary

- **Architectural Styles**
- **System Architectures**
  - Centralized Architectures
  - Decentralized Architectures
  - Hybrid Architectures
- **Self-management in Distributed Systems**