Advanced Topics in Distributed Systems

Dr. Ayman A. Abdel-Hamid

Computer Science Department
Virginia Tech

Distributed Systems Architectures
(Based on Ch2 in Distributed Systems: Principles and Paradigms, 2/E)

Outline

• Architecture Styles
• System Architectures
Introduction

• Logical organization of distributed systems into software components and their interactions
  ➢ Software architecture
• Instantiation of a software architecture
  ➢ System architecture
    ❑ Place software components on real machines

Architectural Styles

• Style: components, components interconnection, data exchange, and configuration
• Styles
  ➢ Layered
  ➢ Object-based
  ➢ Data-centered
  ➢ Event-based
Layered Architecture Style

Object-based Architecture Style
Data-centered Architecture Style

- Processes communicate through a common (active or passive) repository
  - Shared distributed file system
  - Shared web-based data services

Event-based Architecture Style

- Events can carry data
- Publish/subscribe systems
- Processes loosely coupled (no explicit reference)
Shared Data Spaces

- Combination of event-based and data-centered
- Processes decoupled in time
- Need not both be active when communication takes place
- Data accessed using a description rather than explicit reference

Centralized Architectures

- Client-server interaction in a request-reply behavior
- Interaction based on
  - Unreliable Connectionless protocol
  - Reliable connection-oriented
    - Data and control connections
- Application layering
  - User-interface, processing, and data levels
Example: Internet Search Engine in 3 Layers

Multi-tiered Architectures 1/2
Multi-tiered Architectures 2/2

Three-tiered architecture (server acting as a client)

Decentralized Architectures

- Vertical distribution
  - Place logically different components on different machines
  - Each machine tailored to a specific group of functions
- Horizontal distribution
  - Client or server maybe split into logically equivalent parts
  - Each part operating on complete data set
  - P2P systems: all processes are equal!
  - Organize processes into an overlay network
    - Structured
    - Unstructured
Structured P2P Architectures

- Overlay network constructed using a deterministic procedure
- Distributed Hash Tables (DHT)
  - Data items assigned a random key from large identifier space
  - Nodes in system assigned a random number from same identifier space
  - Find a deterministic scheme that maps the key of data item to the identifier of a node based on some distance metric
  - When lookup, network address for node responsible for data item is returned

Mapping data onto nodes: Chord

- Data with key $k$ mapped to node with $id \geq k$
  - Node is $\text{succ}(k)$
- A lookup would return the network address of $\text{succ}(k)$
- Joining (versus leaving)
  - Generate a random $id$
  - Lookup on $id = \text{succ}(id)$
  - Contact $\text{succ}(id)$ and its predecessor and insert into ring
  - Data item whose key associated with node $id$ is transferred from $\text{succ}(id)$
Unstructured P2P Architectures

- Randomized algorithms for constructing an overlay network
- Each node maintains a list of neighbors
- Data randomly placed on nodes
  - Flood the network with a search query
- How to construct the list of neighbors → partial view
  - Nodes regularly exchange entries from partial view
  - Each entry identifies another node and has an associated age indicating how old the reference is
  - When building partial views, discard old nodes

Topology Management in Unstructured P2P overlays
Superpeers in Unstructured P2P overlays

- No deterministic way of routing a lookup request to a specific data item
- Superpeers: special nodes that maintain an index of data items
- Communication through superpeers

Hybrid Architectures: Edge-server Systems

- Hybrid P2P architectures
  - Direct peer-to-peer connections
  - Edge-server systems
    - Content provider
    - ISP
    - Core Internet
    - Edge server
    - Enterprise network
Hybrid Architectures: Collaborative DS

- To get started, a client-server scheme
- Once joined, node uses fully decentralized scheme for collaboration
- Example: BitTorrent file sharing system