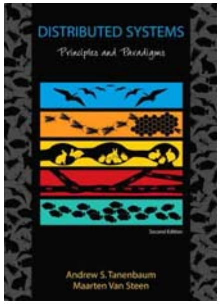
Architectures

527950-1
Fall 2019
9/26/2019
Kyoung Shin Park
Applied Computer Engineering
Dankook University

Chapter 2. Architectures



From Andrew S Tanenbaum, Maarten Van Steen
Distributed Systems: Principles and Paradigms
Edition 2, © Prentice Hall 2007

Overview

- System Architectures
  - **Software Architectural Styles**
    - **Layered architectures**
    - **Object-based architectures**
    - **Data-centered architectures**
    - **Event-based architectures**
    - **Shared data-space architectures**
  - **System Architectures**
    - **Centralized architectures**
    - **Decentralized architectures**
    - **Hybrid architectures**

Architectures

- There are different ways on how to view the organization of a distributed system
- Make a distinction between the logical organization of the collection of software components and the actual physical realization
- **Software architecture** (Architectural Models in CDK)
  - How the various software components are to be organized
  - How they should interact
  - Aim at achieving (at a reasonable level) distribution transparency
    - Internal details of the distribution are hidden from the users
- **System architecture** (Physical Models in CDK)
  - Instantiate and place software components on real machines

Software Architectural Styles

- Architectural Styles are formulated in terms of
  - Components (communication entities in CDK)
    - A modular unit with well-defined required and provided interfaces that is replaceable within its environment, provided we respect its interfaces
  - The way that components are connected to each other(communication paradigms in CDK)
    - A mechanism that mediates communication, coordination, or cooperation among components
    - Formed by the facilities for RPCs, message passing, or streaming data
  - The data exchanged between components
  - Configuration (architectural patterns in CDK)
    - How these elements are jointly configured into a system

Software Architectural Styles

- Various Configurations :
  - **Layered architectures**
  - **Object-based architectures**
  - **Data-centered architectures**
  - **Event-based architectures**

  - Researchers have abandoned the idea that a single distributed system can be used to cover 90% of all possible cases.

Layered Architectures

- Components are organized in a layered fashion
- A component at layer Li is allowed to call components at the underlying layer Li-1, but not the other way around
- This model has been adopted by the networking community

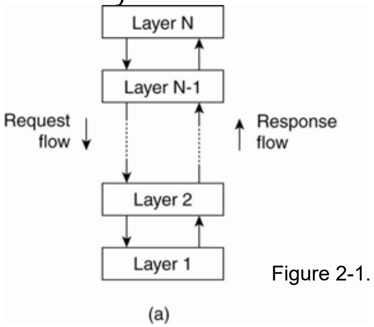


Figure 2-1. The (a) layered architectural style

Object-based Architectures

- Each object corresponds to a component
- These components are connected through a remote procedure call mechanism
- This architecture matches the client-server system architecture

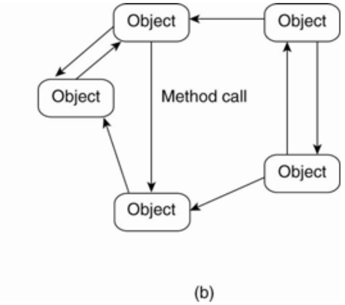


Figure 2-1. (b) The object-based architectural style.

Data-Centric Architectures

- Processes communicate through a common passive or active repository
- Shared distributed file system
- Web-based distributed systems
  - Data-centric
  - Process communicate through the use of shared web-based data services

Event-based Architectures

- Processes communicate through the propagation of events, which optionally carry data
- Publish/subscribe systems
  - Processes publish events after which the middleware ensures that only those processes that subscribed to those events will receive them
- Processes are loosely coupled. They need not explicitly refer to each other
  - Decoupled in space or referentially decoupled

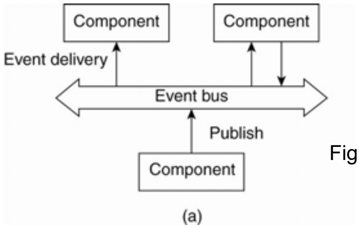


Figure 2-2. (a) The event-based architectural style

Shared Data-Space Architectures

- Event-based architectures combined with data-centered architectures
- Processes are also decoupled in time
  - They need not both be active when communication takes place
- Use a SQL-like interface to the shared repository

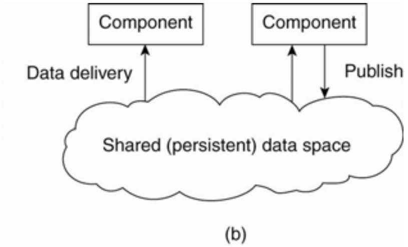


Figure 2-2. (b) The shared data-space architectural style.

System Architectures

- Various instantiations (placement) (roles and responsibilities in CDK) of a software architecture :
  - **Centralized** architectures - **Client-Server architecture**
  - **Decentralized** architectures - **Peer-to-peer architecture**
  - **Hybrid** architectures

Centralized Architectures

- Centralized Architectures (Client-Server Architecture)
- A server is a process implementing a specific service
- A client is a process that requests a service from a server
- By sending it a request and subsequently waiting for the server's reply

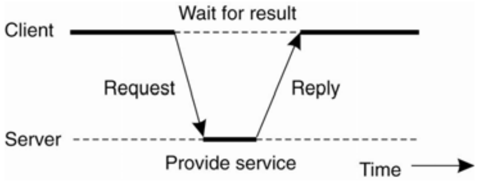


Figure 2-3. General interaction between a client and a server.

Centralized Architectures

- **Communication** between a client and a server can be implemented
  - **A connectionless protocol**
    - When the underlying network is fairly reliable
    - Efficient
    - Making the protocol resistant to occasional transmission failures is not trivial
      - An operation is said to be idempotent when it can be repeated multiple times without harm
  - **A reliable connection-oriented protocol**
    - Relatively low performance
    - Works fine in wide-area systems in which communication is unreliable

Centralized Architectures

- **Application Layering**
  - How to draw a clear distinction between a client and a server? Server may act as a client
  - Many client-server applications are targeted toward supporting user access to database, many people have advocated a distinction between the following three levels.
    - **User-interface (Presentation) Level**
    - **Processing (Business Logic, Application Processing) Level**
    - **Data (Data Access, Data Persistence) Level**

Centralized Architectures

- **Application Layering**
  1. **User-interface (Presentation) Level**
     - Contains all that is necessary to directly interface to user, such as display management
     - A part that handles interaction with a user
  2. **Processing (Business Logic, Application Processing) Level**
     - Typically contains the applications
     - A middle part that generally contains the core functionality of an application
     - In contrast to user-interface and data levels, there are not many aspects common to the processing level

Centralized Architectures

Application Layering

3. Data(Data Access, Data Persistence) Level
   - Manages the actual data that is being acted on
   - A part that operates on a database
   - Data are often persistent, that is, even if no application is running, data will be stored somewhere for next use
   - Responsible for data consistent across different applications
   - The data are organized independent of the applications in such a way that changes in the data organization do not affect applications, and neither do the applications affect the data organization

Centralized Architectures

Application Layering

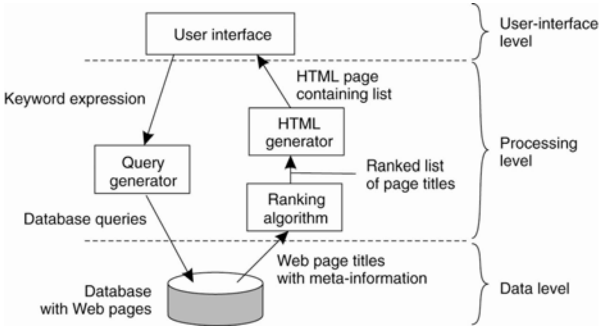


Figure 2-4. The simplified organization of an Internet search engine into three different layers.

Centralized Architectures

Multi-tiered Architectures

- The distinction into three logical levels suggests a number of possibilities for physically distributing a client-server application across several machines
- (Physically) Two-tiered Architecture
  - Only two kinds of machines: client machines and server machines
- (Physically) Three-tiered Architecture
  - A server may sometimes need to act as a client

Centralized Architectures

Two-tiered Architecture

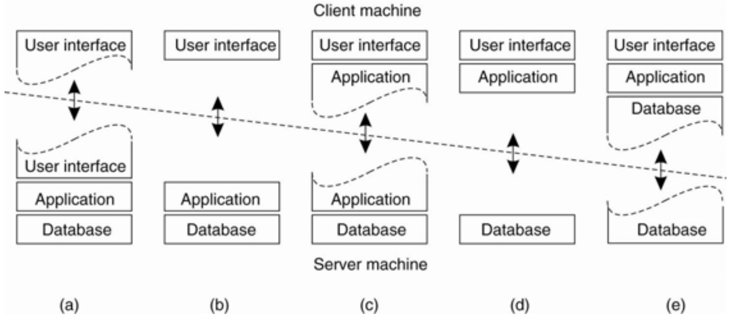


Figure 2-5. Alternative client-server organizations (a)–(e).

Centralized Architectures

- **Two-tiered Architecture**
  - Only the terminal-dependent part of the user interface on the client machine – dumb terminal
  - Divide the application into a graphical front end, which communicates with the rest of the application through an application-specific protocol
  - Move part of the application to the front end. The application uses a form that needs to be filled in entirely before it can be processed.
  - Most of the application is running on the client machine, but operations on database entries go to the server
  - Client's local disk contains part of the data
    - **Fat clients : (d)-(e)**
      - Not optimal from a system's management perspective
    - **Thin Clients : (a)-(c)**
      - Much easier

Centralized Architectures

- **Three-tiered Architecture**
  - Programs that form part of the processing level reside on a separate server
  - Organization of many web sites

"A **web server** acts as an entry point to a site, passing requests to an **application server** where the actual processing takes place. This application server, in turn, interact with a **database server**."

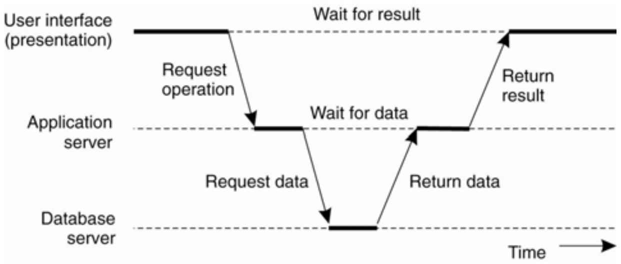


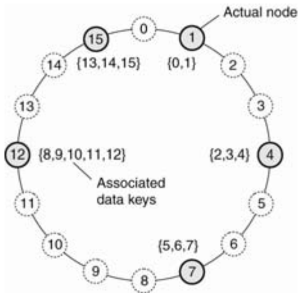
Figure 2-6. An example of a server acting as client.

Decentralized Architectures

- **Decentralized Architectures(Peer-to-peer Architecture)**
- Vertical distribution
  - Placing logically different components on different machines
  - Functions are logically and physically split across multiple machines
- Horizontal distribution
  - A client or server may be physically split up into logically equivalent parts, but each part is operating on its own share of the complete data set, thus balancing the load
  - The processes are all equal.
  - Much of the interaction between processes is symmetric - each process will act as a client and a server at the same time
  - How to organize the processes in an overlay network - the nodes are formed by the processes and the links represent the possible communication channels
    - **Structured peer-to-peer architectures: distributed hash table(DHT)**
    - **Unstructured peer-to-peer architectures**

Structured Decentralized Architectures

- Structured Peer-to-Peer Architectures



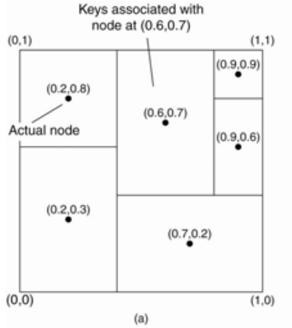


Figure 2-7. The mapping of data items onto nodes in Chord.

Figure 2-8. (a) The mapping of data items onto nodes in CAN.

Unstructured Decentralized Architectures

- Unstructured Peer-to-Peer Architectures

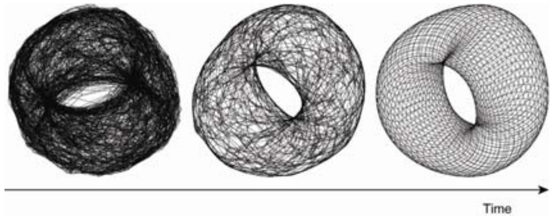


Figure 2-11. Generating a specific overlay network using a two-layered unstructured peer-to-peer system [adapted with permission from Jelasity and Babaoglu (2005)].

Hybrid Architectures

- **Client-server architectures combined with decentralized architectures**
- Edge-Server Systems
  - Deployed on the Internet where servers are placed at the edge of the network
  - The edge server serves content after applying filtering functions
  - A collection of servers can be used to optimize content and application distribution

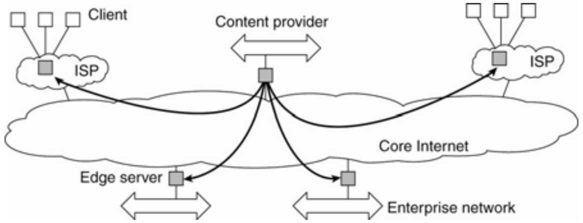


Figure 2-13. Viewing the Internet as consisting of a collection of edge servers.

Hybrid Architectures

- Collaborative Distributed Systems
  - When collaborative distributed systems first get started, often a traditional client-server model is deployed
  - Once a node has joined the system, it can use a fully decentralized scheme for collaboration
  - BitTorrent file-sharing system

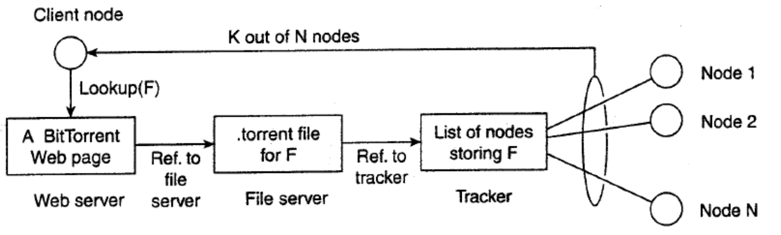


Figure 2-14. The principal working of BitTorrent [adapted with permission from Pouwelse et al. (2004)].