Lecture 2 - Distributed System Architectures

Definitions

- **Software Architectures** – describe the organization and interaction of software components; focuses on logical organization of software (component interaction, etc.)

- **System Architectures** - describe the placement of software components on physical machines

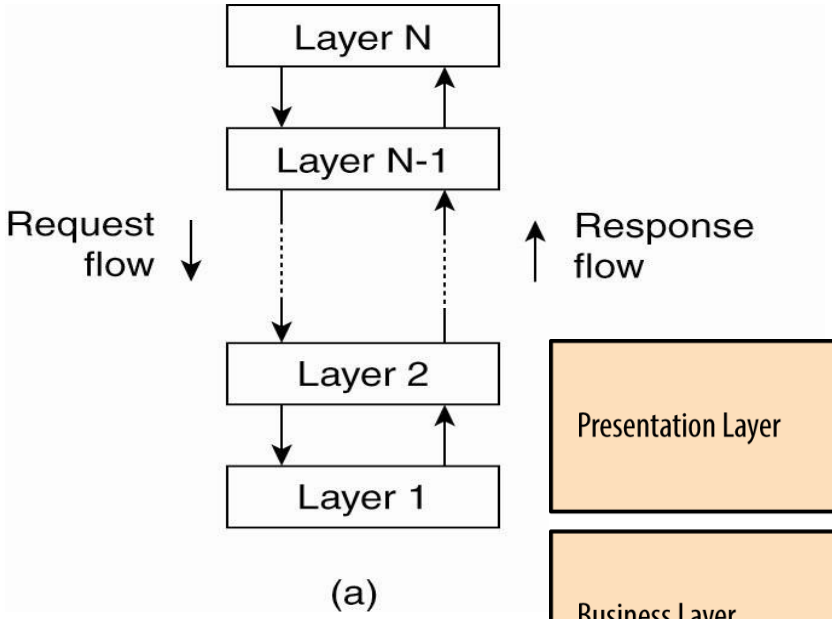
Architectural Styles

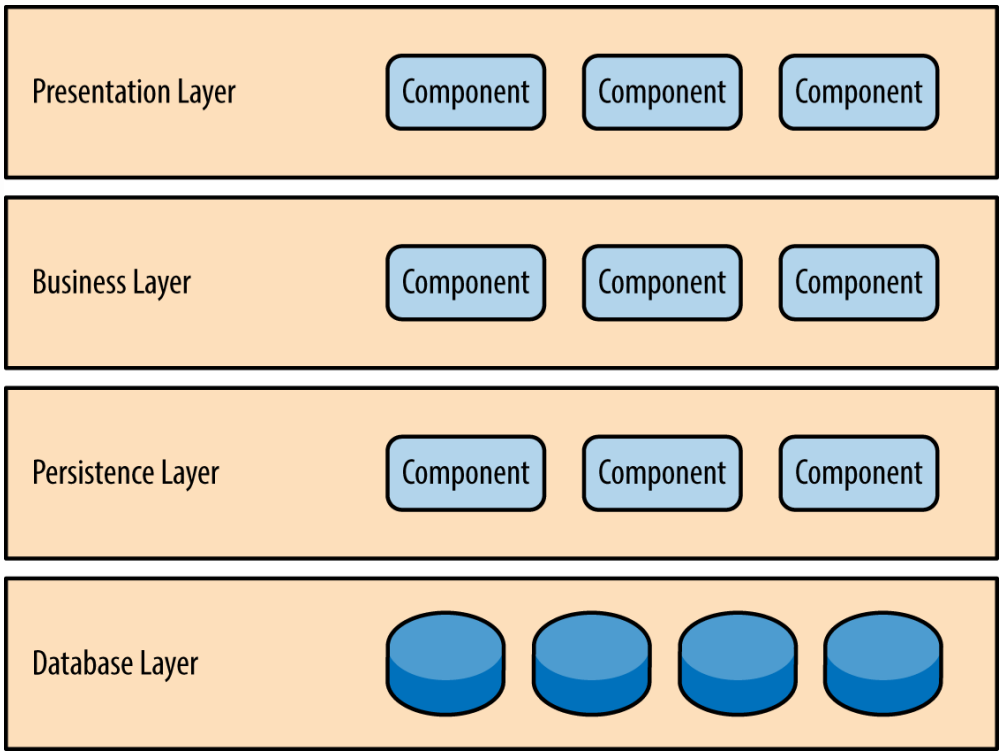
- An **architectural style** describes a particular way to configure a collection of components and connectors.
  - **Component** - a module with well-defined interfaces; reusable, replaceable
  - **Connector** – communication link between modules
- Architectures suitable for distributed systems:
  - Layered architectures*
  - Component-based architectures*
  - Data-centered architectures
  - Event-based architectures

Layered Architecture

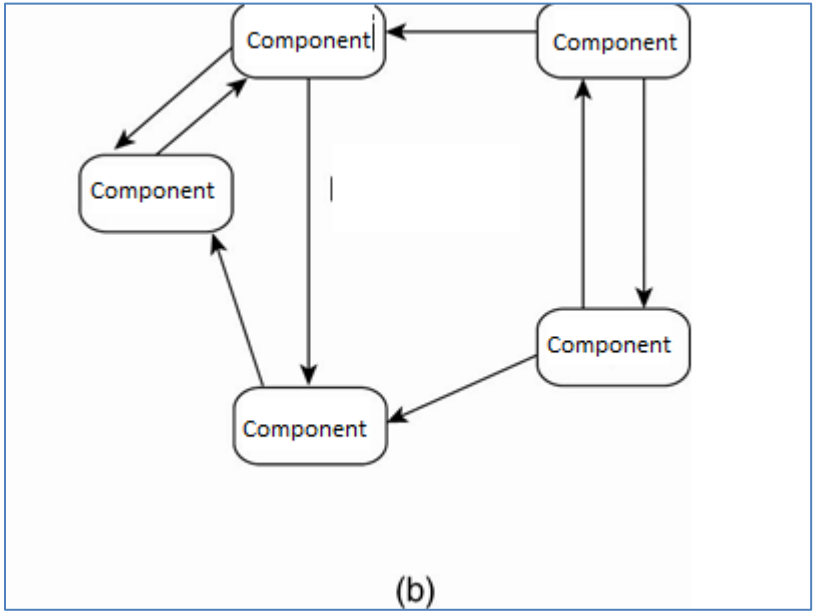
- Each layer/tier is allocated a specific responsibility of the system
- Each layer can only interact with the neighboring layers (important for integrity and security)
- Each layer may contain layers of its own (e.g. Presentation layer could contain two layers: client layer and client presentation layer)

Layered Architecture



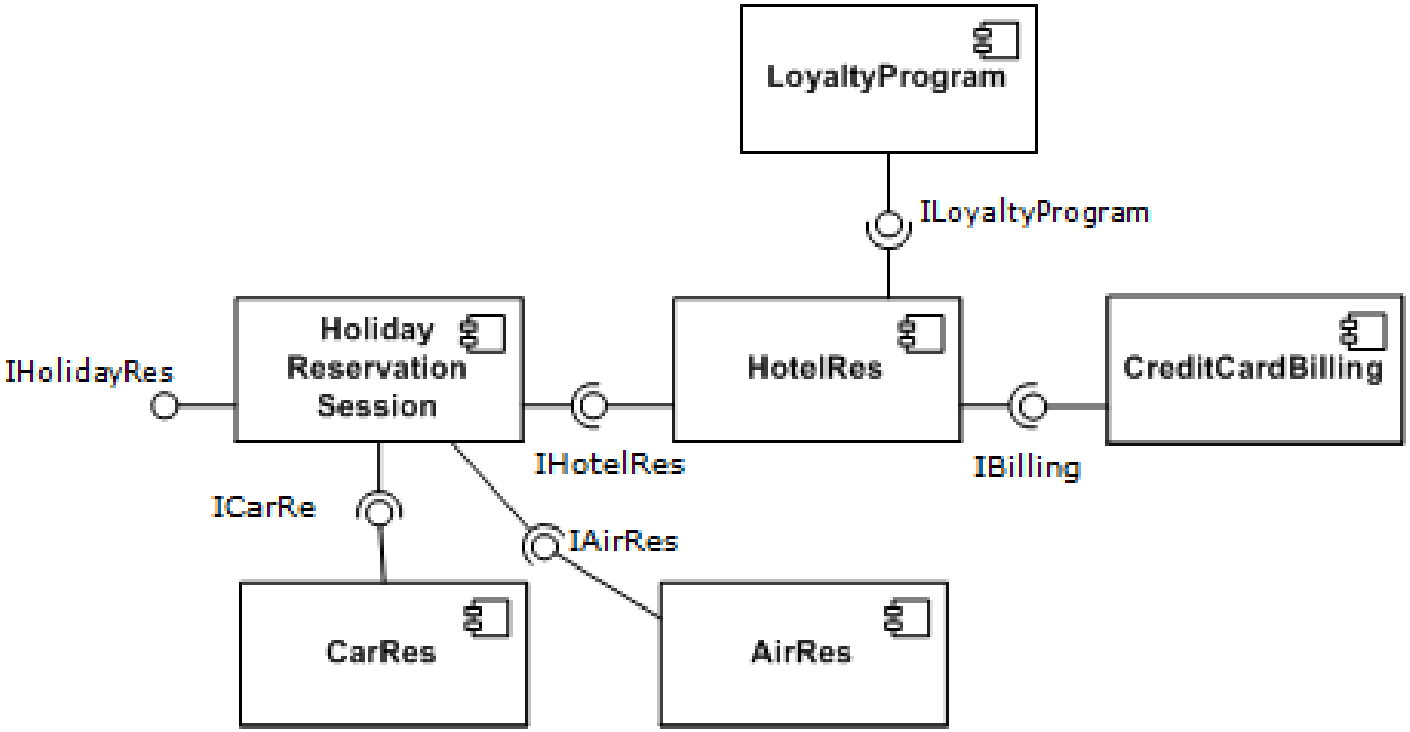


Component based architecture



- Consists of components and connectors
- Component based is less structured

Component based architecture

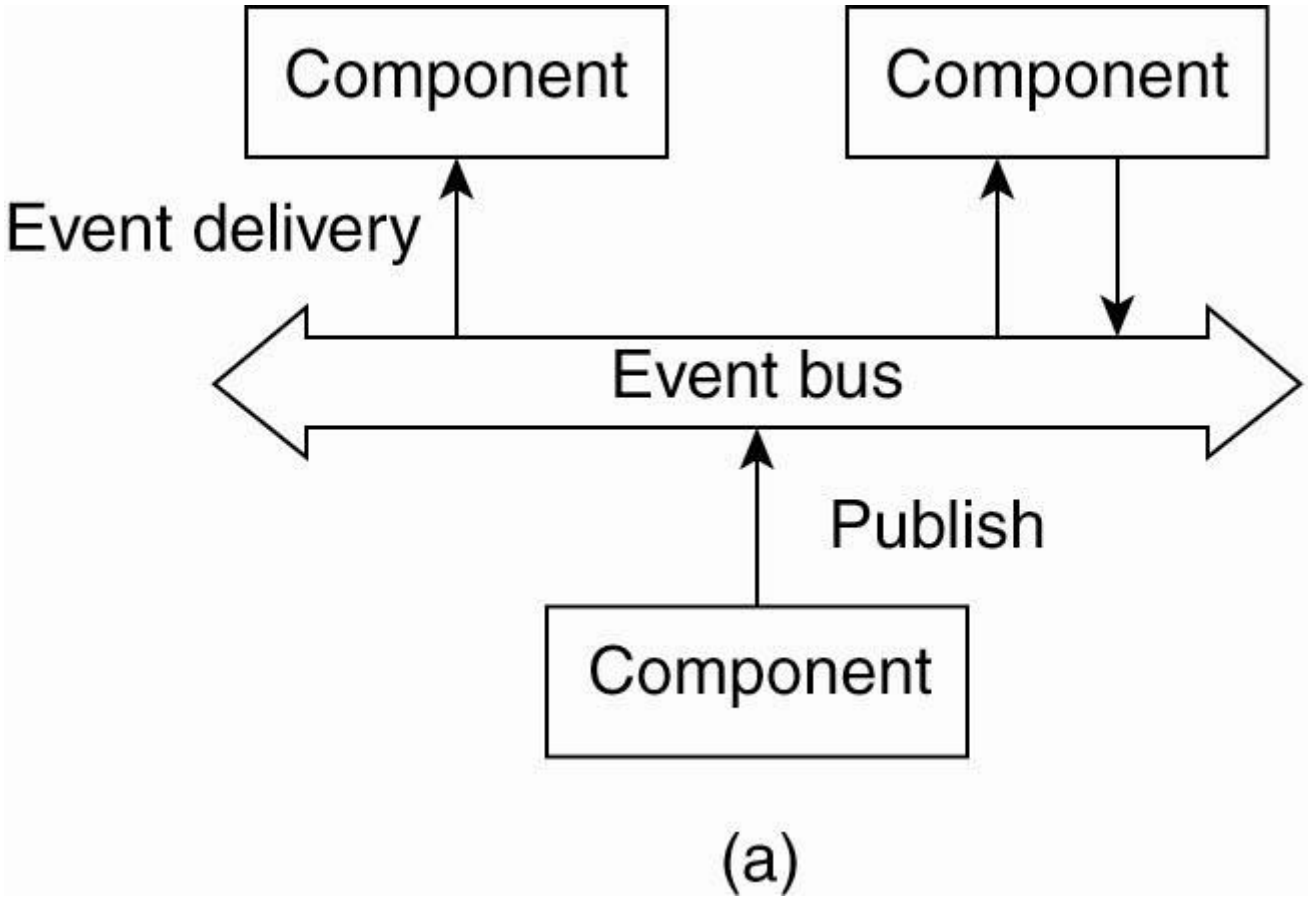


Data-Centered Architectures

- Main purpose: data access and update
- Processes interact by reading and modifying data in some shared repository (active or passive)

  - Traditional data base (passive): responds to requests
  - Blackboard system (active): clients solve problems collaboratively; system updates clients when information changes.

Event based Architectures



Event based Architectures

- Communication via event propagation, in dist. systems seen often in Publish/ Subscribe; *e.g.,* register interest in market info; get email updates
- Decouples sender & receiver; asynchronous communication

- Event-based architecture supports several communication styles:
  - Publish-subscribe
  - Broadcast
  - Point-to-point

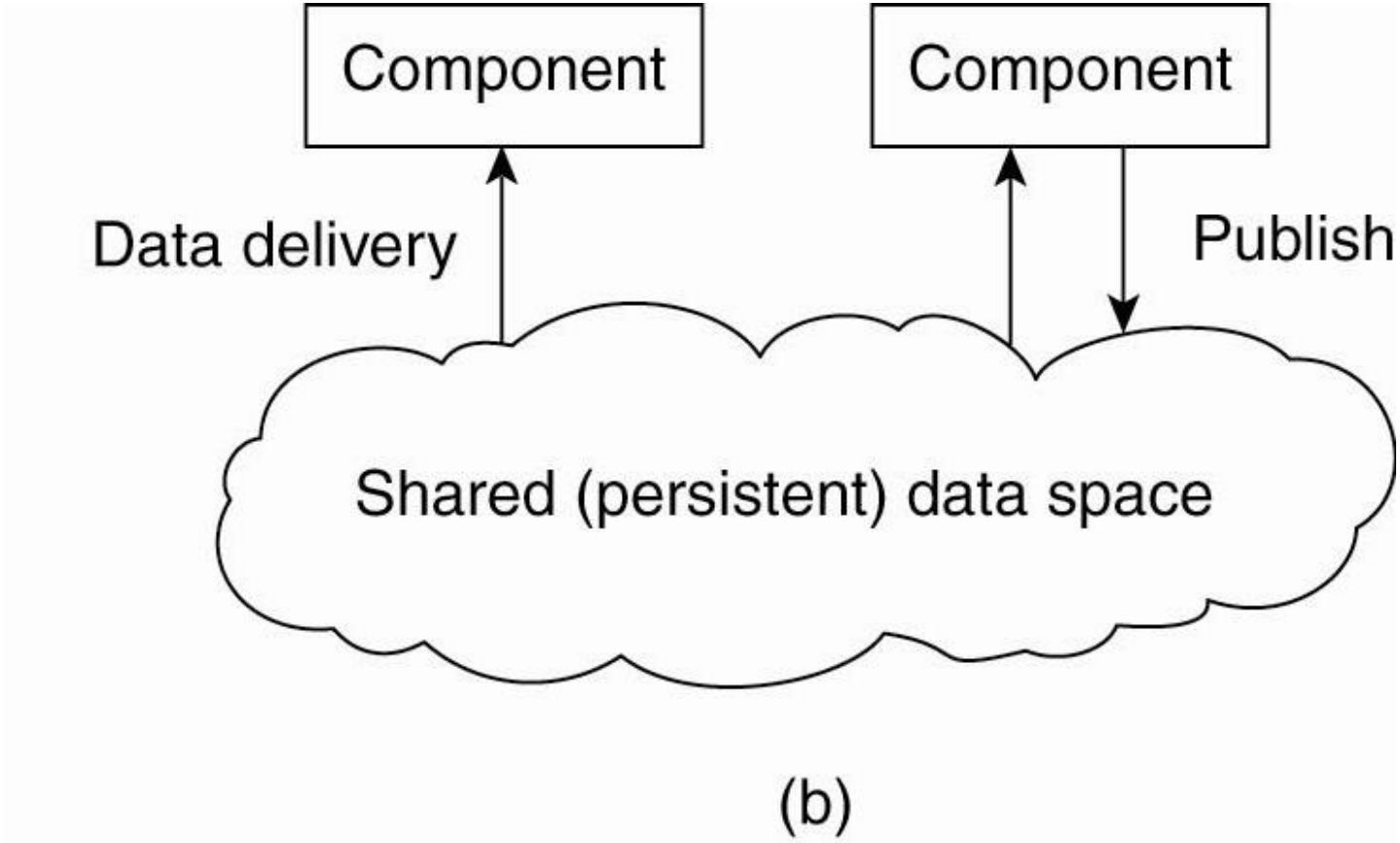
Shared Data-Space Architecture

- Multiple Architectures can be combined in the same system architecture

  E.g. A component in the object based architecture may have a layered architecture

- Shared Data-Space Architecture combines the Data-centric architecture and Event based architecture

Shared Data-Space Architecture



- E.*g.*, shared distributed file systems or Web-based distributed systems
- Processes communicate asynchronously

Which Software Architecture?

- An online forum to share travel information among users
- A remote monitoring system that monitors the health of an elderly person.
- A music file sharing system among a group of users.
- A mobile taxi app

Distribution Transparency

- An important characteristic of software architectures in distributed systems is that they are designed to support distribution transparency.
- Transparency involves trade-offs
- Different distributed applications require different solutions/architectures
  - There is no “silver bullet” – no one-size-fits-all system. (Compare NOW, Seti@home, Condor)

System Architectures

System Architectures for Distributed Systems

- **Centralized**: traditional client-server structure
  - Vertical (or hierarchical) organization of communication and control paths (as in layered software architectures)
  - Logical separation of functions into client (requesting process) and server (responder)
- **Decentralized**: peer-to-peer
  - Horizontal rather than hierarchical comm. and control
  - Communication paths are less structured; symmetric functionality
- **Hybrid:** combine elements of C/S and P2P
  - Edge-server systems
  - Collaborative distributed systems.
- Classification of a system as centralized or decentralized refers to communication and control organization, primarily.

Traditional Client-Server

- Processes are divided into two groups (clients and servers).
- Synchronous communication: request-reply protocol
- In LANs, often implemented with a connectionless protocol (unreliable)
- In WANs, communication is typically connection-oriented TCP/IP (reliable)
  - High likelihood of communication failures

C/S Architectures

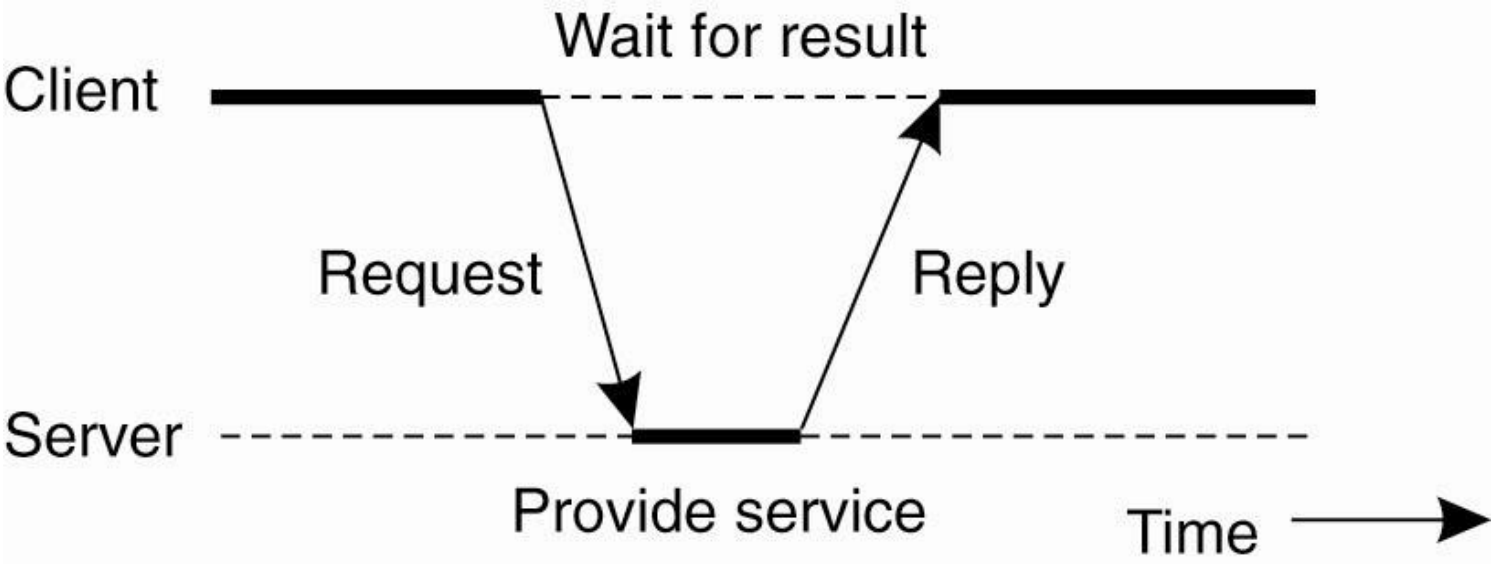


Figure 2-3. General interaction between a client and a server.

Two-tiered C/S Architectures

- Server provides processing and data management; client provides simple graphical display (**thin-client**)
  - Perceived performance loss at client
  - Easier to manage, more reliable, client machines don't need to be so large and powerful
- At the other extreme, all application processing and some data resides at the client (**fat-client** approach)
  - Pro: reduces work load at server; more scalable
  - Con: harder to manage by system admin, less secure

Three-tiered Architectures

- In some applications servers may also need to be clients, leading to a three level architecture
  - Distributed transaction processing
  - Web servers that interact with database servers
- Distribute functionality across three levels of machines instead of two.

Multitiered Architectures (3 Tier Architecture)

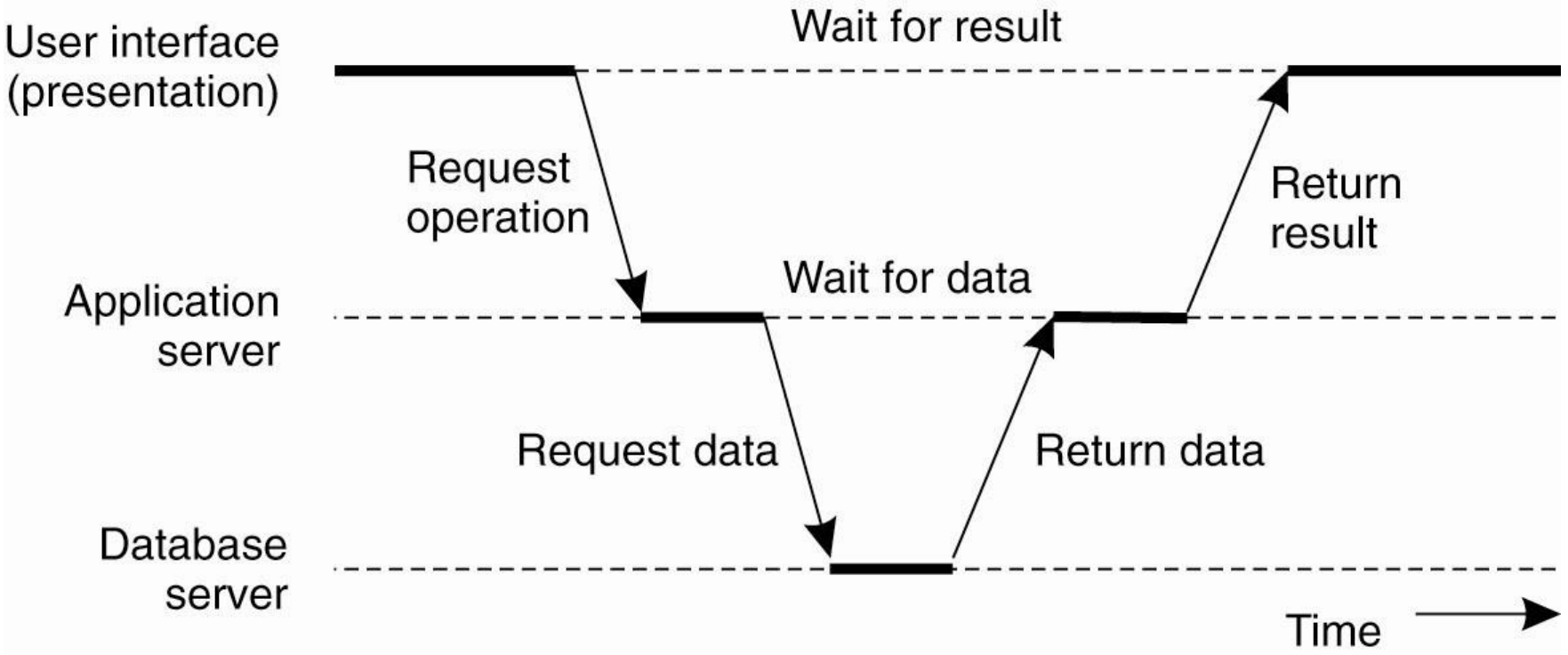


Figure 2-6. An example of a server acting as client.

Multitiered Architectures

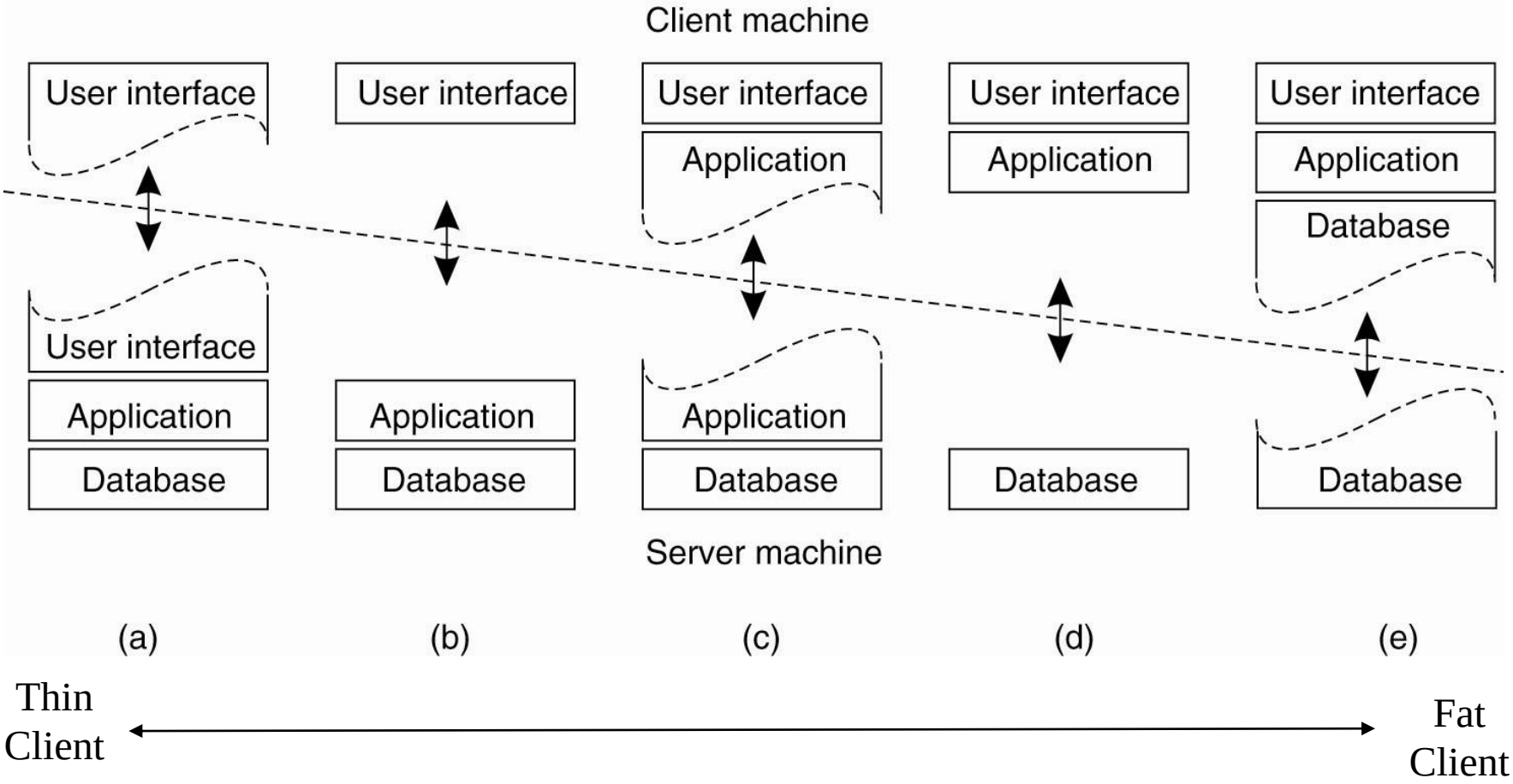


Figure 2-5. Alternative client-server organizations (a)–(e).

Layered (software) Architecture for Client-Server Systems

- **User-interface level**: GUI's (usually) for interacting with end users
- **Processing level**: data processing applications – the core functionality
- **Data level**: interacts with data base or file system
  - Data usually is persistent; exists even if no client is accessing it
  - File or database system

Examples

- Web search engine
  - Interface: type in a keyword string
  - Processing level: processes to generate DB queries, rank replies, format response
  - Data level: database of web pages
- Stock broker's decision support system
  - Interface: likely more complex than simple search
  - Processing: programs to analyze data; rely on statistics, AI perhaps, may require large simulations
  - Data level: DB of financial information
- Cloud based "office suites"
  - Interface: access to various documents, data,
  - Processing: word processing, database queries, spreadsheets,…
  - Data : file systems and/or databases

Application Layering

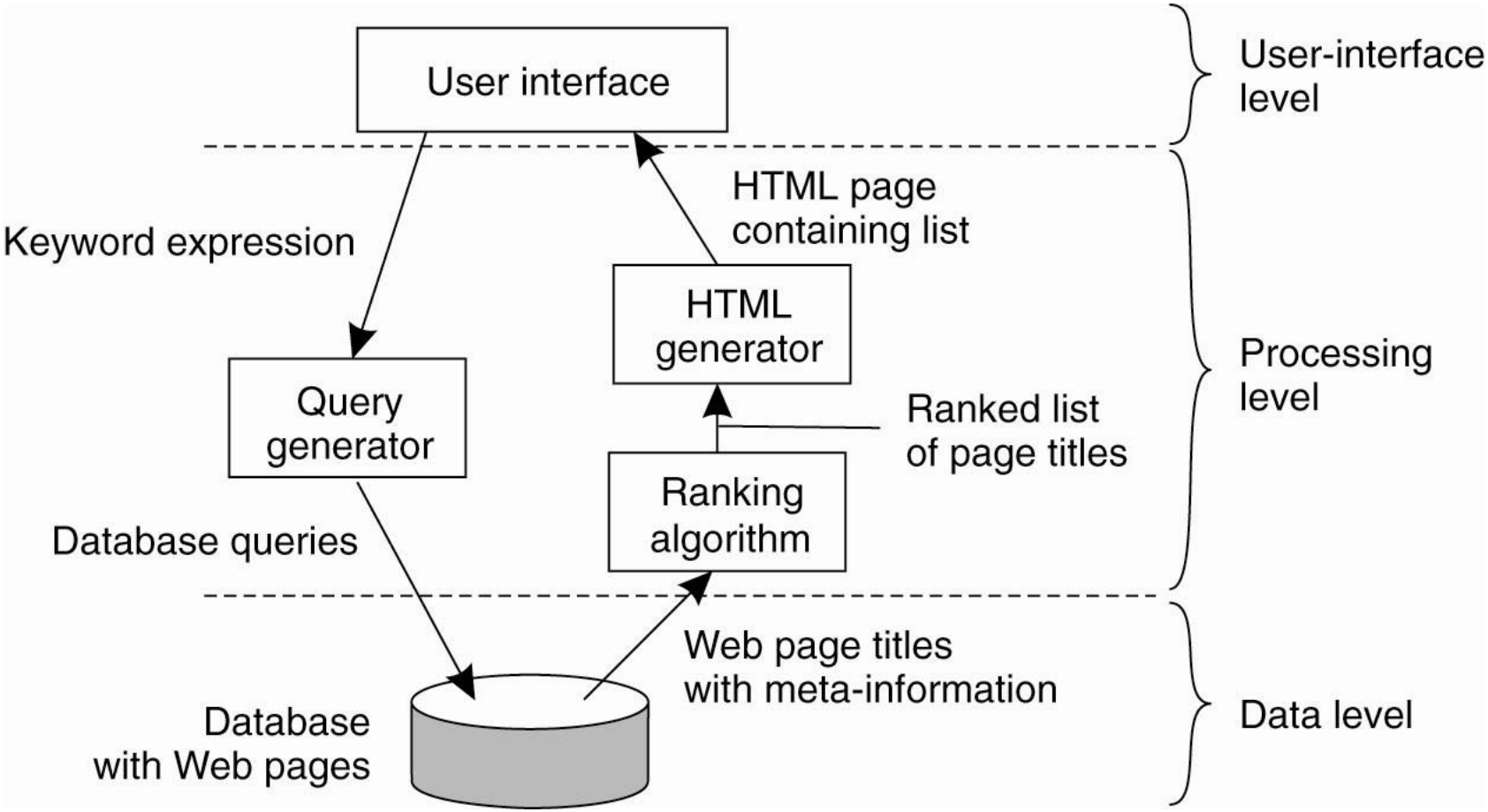
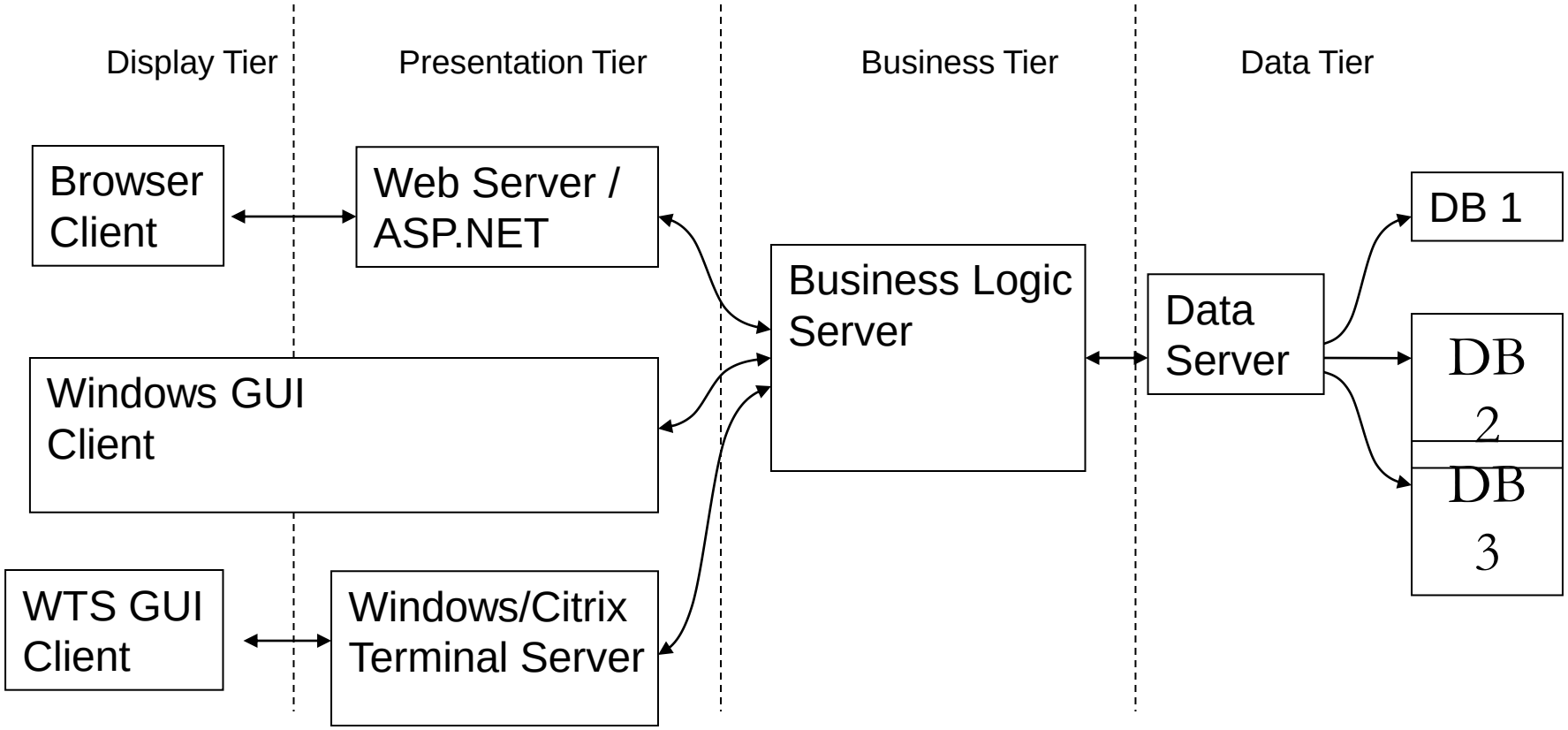


Figure 2-4. The simplified organization of an Internet search engine into three different layers.

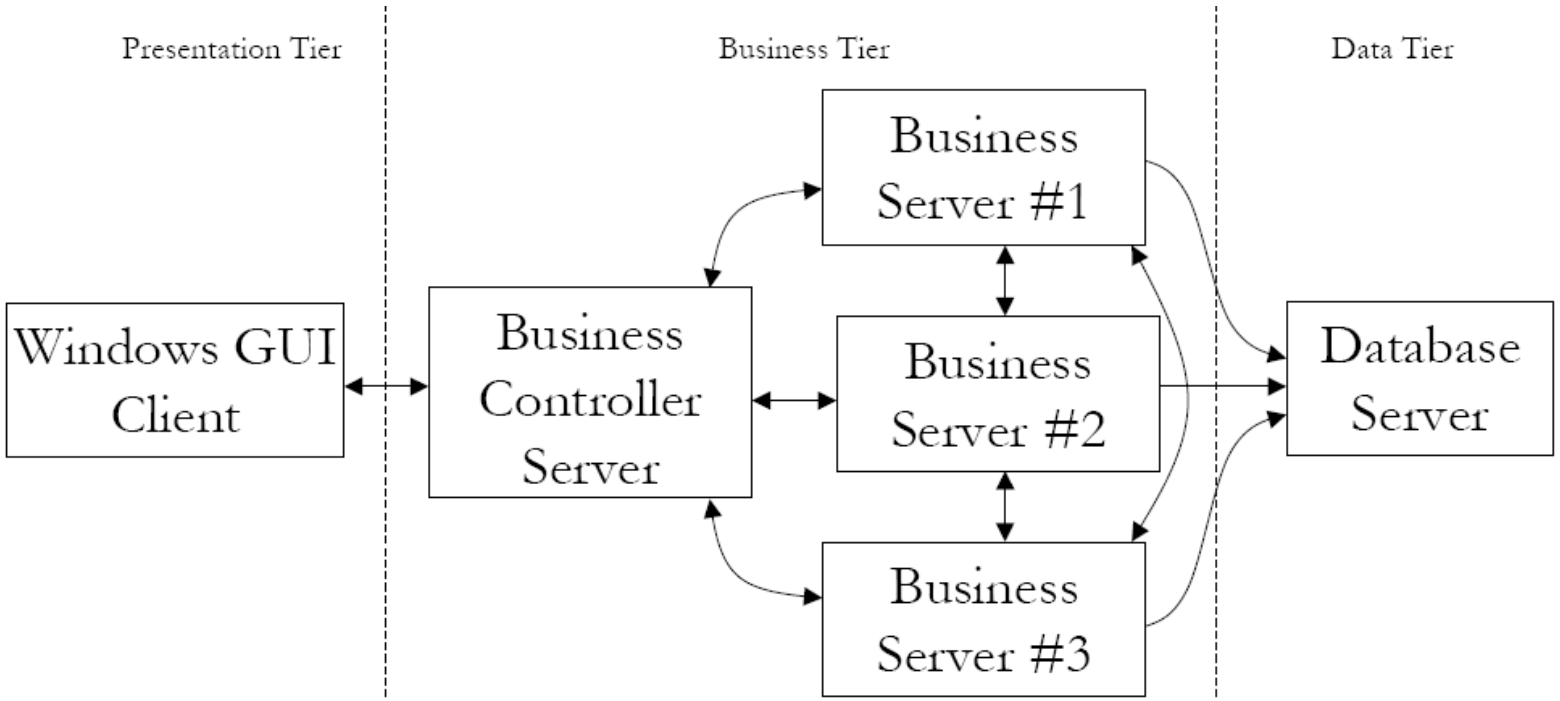
Flexible/Scalable Architecture



Distributing the Business Tier

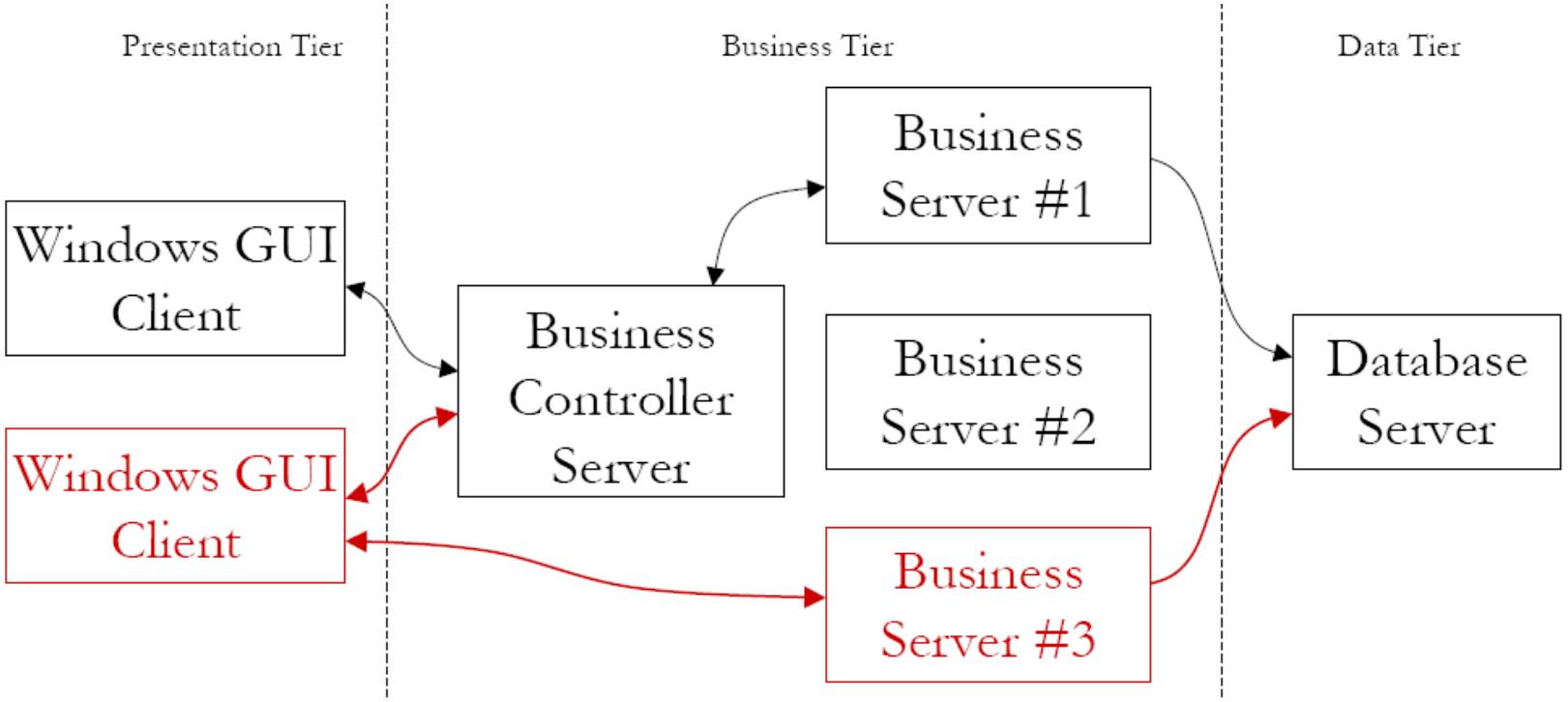
- Business tier could be distributed for various reasons:
  - Parallel computing - split one job among many servers
  - Load balancing - have a controller component redirect presentation clients to a least-utilized business tier server
  - Fault tolerance - robustness to system faults or data faults

Parallel Computing Architecture



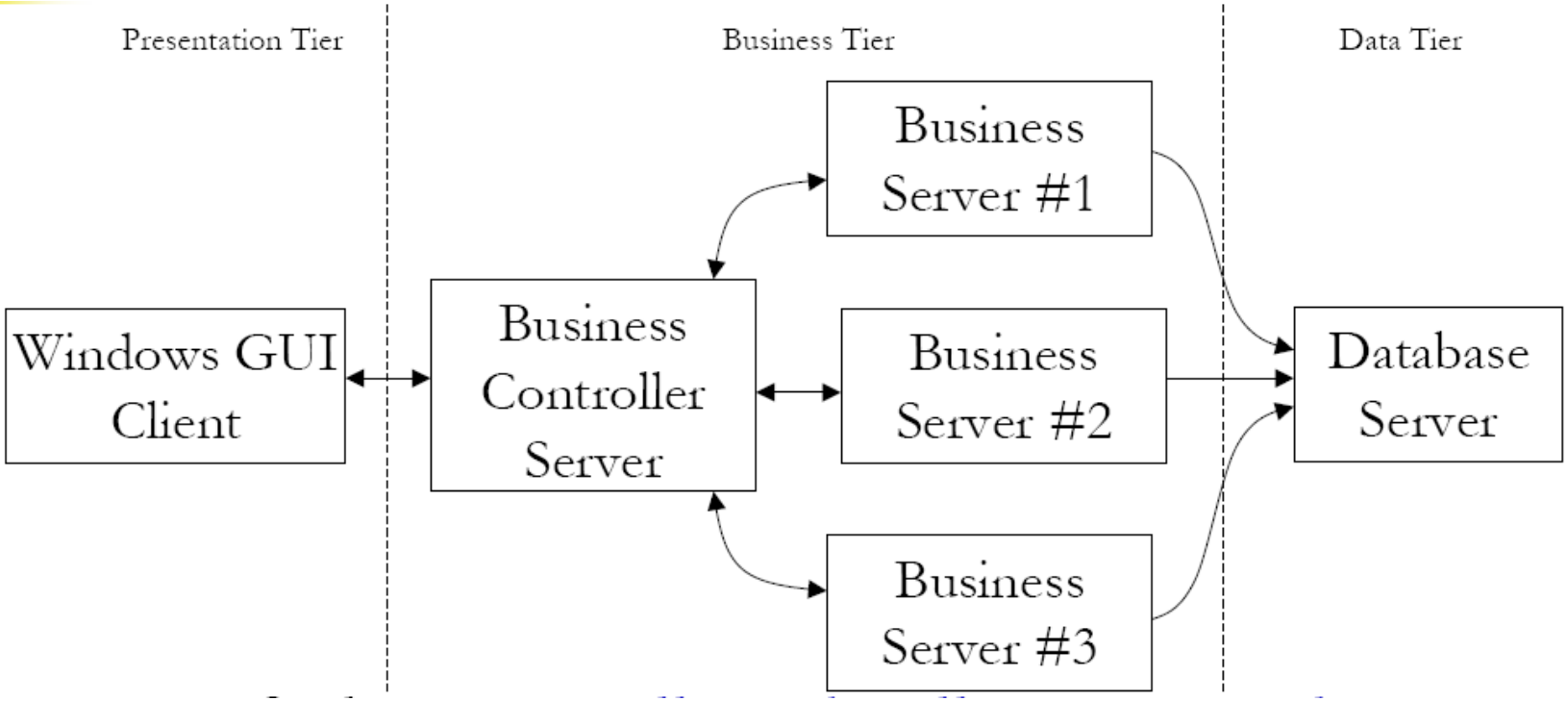
- Controller passes job to server farm, returns response
  - Business servers collaborate via data sharing

Load Balancing Architecture



- Controller passes job to one server, returns response
- Alternative: Controller tells client which server to use

Fault Tolerant Architecture



- Data faults: Controller asks all servers to do the same job
  - Then compares results to detect faults
- System faults: Controller uses any server that is up

Whether to tier or not?

- Advantages

- Can reuse code (high coherence, low coupling)
- Scalable
- Integrity
- Fault tolerance

- Disadvantages

- Increases communication overhead
- Errors/Losses in message transmission
- Can pose security risks (e.g. packet sniffing)
- Increased Complexity

Peer-to-Peer

- Nodes act as both client and server; interaction is symmetric (e.g. Pastry, Chord)
- Each node acts as a server for part of the total system data
- **Overlay networks** connect nodes in the P2P system
  - Nodes in the overlay use their own addressing system for storing and retrieving data in the system
  - Nodes can route requests to locations that may not be known by the requester.

P2P v Client/Server

- P2P computing allows end users to communicate without a dedicated server.
- Communication is still usually synchronous (blocking)
- There is less likelihood of performance bottlenecks since communication is more distributed.
  - Data distribution leads to workload distribution.
- Resource discovery is more difficult than in centralized client-server computing & look-up/retrieval is slower
- P2P can be more fault tolerant, more resistant to denial of service attacks because network content is distributed.
  - Individual hosts may be unreliable, but overall, the system should maintain a consistent level of service

P2P

- Structured – E.g. Chord
- Unstructured – E.g. BitTorrent

Hybrid Architectures

- Combine client-server and P2P architectures
  - Edge-server systems; e.g. ISPs, which act as servers to their clients, but cooperate with other edge servers to host shared content
  - Collaborative distributed systems; e.g., BitTorrent, which supports parallel downloading and uploading of chunks of a file. First, interact with C/S system, then operate in decentralized manner.

https://www.youtube.com/watch?v=6PWUCFmOQwQ

Superpeers

- Maintain indexes to some or all nodes in the system
- Supports resource discovery
- Act as servers to regular peer nodes, peers to other superpeers
- Improve scalability by controlling floods
- Can also monitor state of network
- Example: Napster

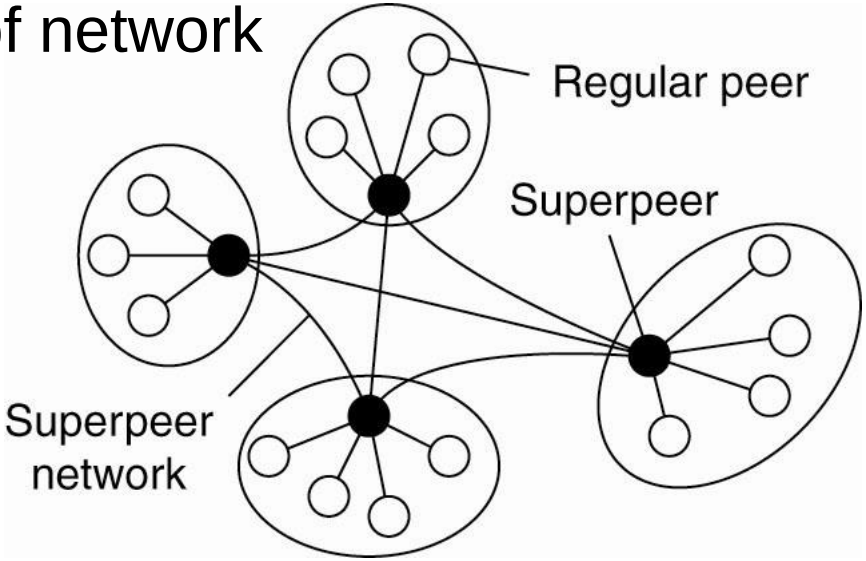


Figure 2-12.

Distributed Hash Tables

- A fully decentralized routing mechanism

https://www.youtube.com/watch?v=-UU_ugiPZ9k

Possibilities with P2P?

- Currently mostly used for file sharing
- Blockchain uses P2P
- Online Social networks?
- Taxi applications?
- Etc etc

Centralized v Decentralized Architectures

- Traditional client-server architectures exhibit **vertical distribution**. Each tier serves a different purpose in the system.
  - *Logically* different components reside on different nodes
- **Horizontal distribution** (P2P): each node has roughly the same processing capabilities and stores/manages part of the total system data.
  - Better load balancing, more resistant to denial-of-service attacks, harder to manage than C/S
  - Communication & control is not hierarchical; all about equal

Architecture versus Middleware

- Where does middleware fit into an architecture?
- Middleware: the software layer between user applications and distributed platforms.
- Purpose: to provide distribution transparency
  - Applications can access programs running on remote nodes without understanding the remote environment

Architecture versus Middleware

- Middleware may also have an architecture
  - e.g., CORBA has an object based style.
- Use of a specific architectural style can make it easier to develop applications, but it may also lead to a less flexible system.
- Possible solution: develop middleware that can be customized as needed for different applications with different architectures.

Summary

- Software Architecture Vs System Architecture
- Different Software Architectural styles – Layered, Component based, event driven, data centered…
- Can have combinations of these styles
- Different System Architectures – Client Server, Peer to Peer, Hybrid