Chapter 2

Systems Model

Introduction

- Ways to view the organization of a distributed system
  - logical organization of the collection of software components (Software architecture)
    - tell us how the various software components are to be organized and how they should interact.
  - actual physical realization
    - requires that we instantiate and place software components on real machines.
    - The final instantiation of software architecture is also referred to as system architecture.

Architectural Styles

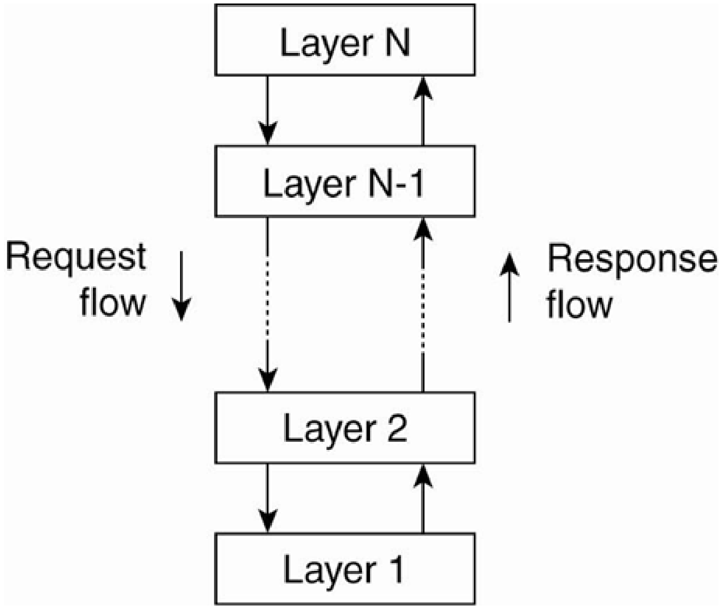
- Logical organization (Software architecture)
- formulated in terms of
  - Components
    - the way that components are connected to each other
    - the data exchanged between components
    - how these elements are jointly configured into a system
  - connectors
    - mechanism that mediates communication, coordination, or cooperation among components
    - It can be formed by the facilities for (remote) procedure calls, message passing, or streaming data.

- Using components and connectors, we can come to various configurations which in turn have been classified in to architectural styles.
  - Layered architecture
    - Pure, mixed and layered organization with upcalls
  - Object-based architecture
  - Data-centered architecture
  - Event-based architecture

Pure Layered architecture

- components are organized in a layered fashion where a component at layer L_i is allowed to call components at the underlying layer L_{i-1}

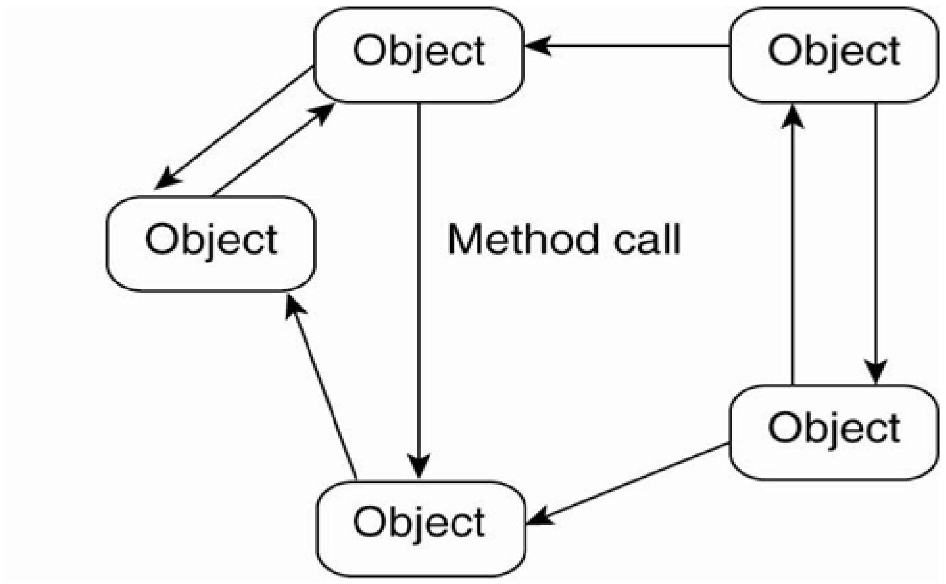
- This model has been widely adopted by the networking community (i.e. network layer).



- control generally flows from layer to layer: ***requests go down*** the hierarchy whereas the ***results flow upward***

Object-based architecture

- each object corresponds to a component and these components are connected through a (remote) procedure call mechanism
- matches the client-server system architecture.



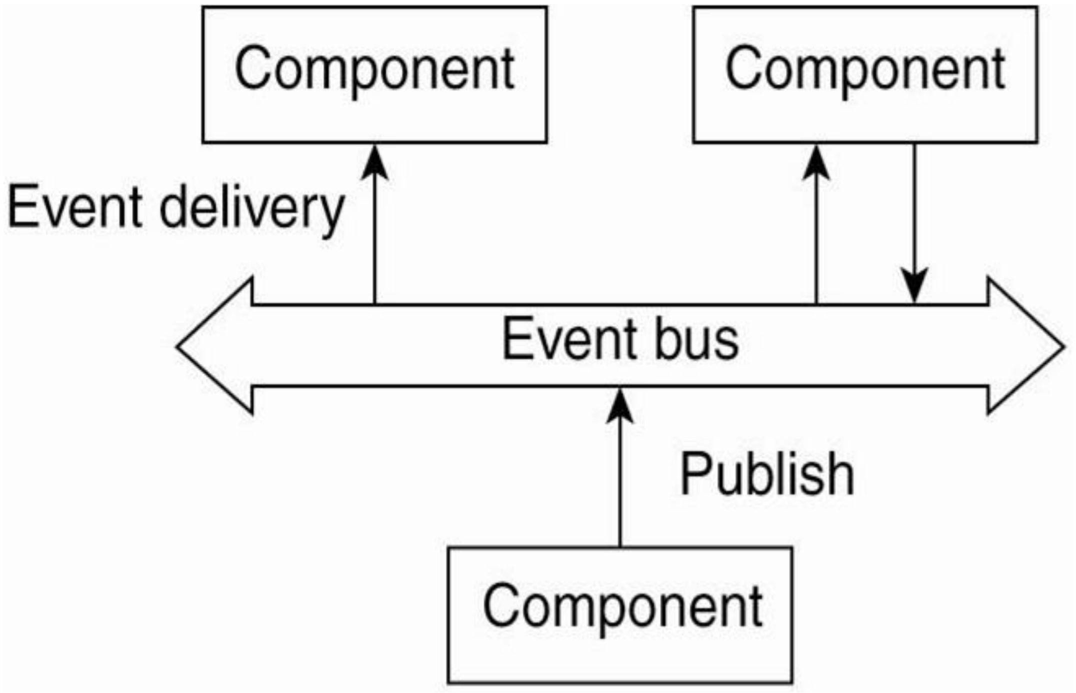
Data-centered architecture

- Evolve the idea that processes communicate through a common repository like shared distributed file system
- For example, web-based distributed systems are largely data-centric

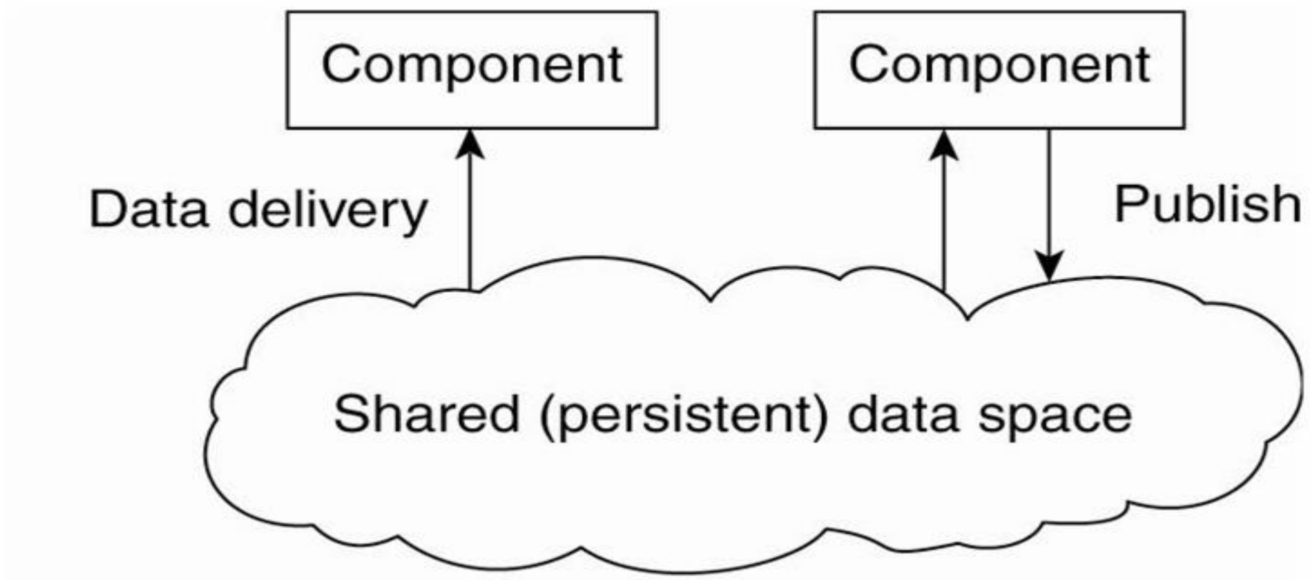
Event-based architecture

- processes essentially communicate through the propagation of events
- event propagation has been associated with what are known as publish/subscribe systems

- The basic idea is that processes publish events after which the middleware ensures that only those processes that subscribed to those events will receive them



- Event-based architectures can be combined with data-centered architectures, this is known as shared data spaces

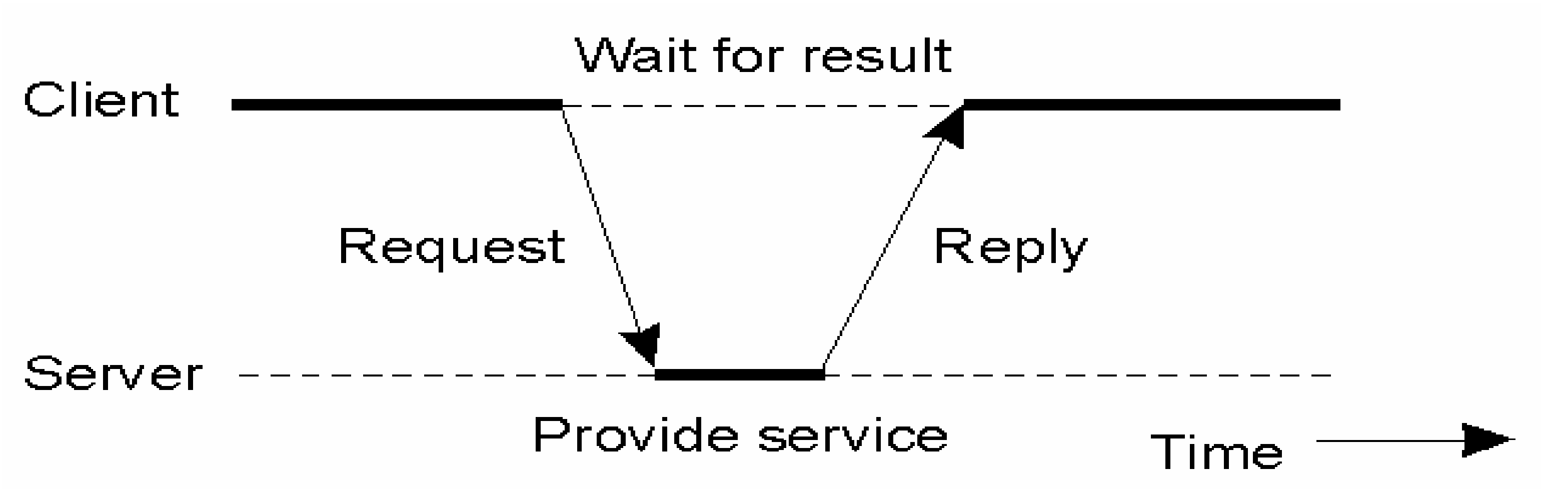


System Architecture

- Deals with
  - how processes are organized in a system;
  - where do we place software components
- Deciding their interaction
  - Centralized architectures
  - Decentralized architectures
  - Hybrid architectures

Centralized Architecture

- is a client-server model
- A server is a process implementing a specific service, for example, a file system service or a database service, web service, mail service etc
- A client is a process that requests a service from a server by sending a request and subsequently waiting for the server's reply
- client-server interaction is also known as request-reply behavior



- Communication between a client and a server can be implemented by:
  - Connectionless Protocol called ***UDP (User Datagram Protocol)***
  - Connection Oriented Protocol called ***TCP (Transmission Protocol)***

Connectionless Protocol

- It is fairly reliable. It does not require acknowledgement. (standard letter through PO)
- Preferable with applications such as streaming audio, video and voice over IP (VoIP).
- When a client requests a service,
  - it identifies the services it want, packages the message along with the necessary input data and send it to the server
- The server
  - Always waits for an incoming request, process it, package the results in a reply message and finally send it to the client.
- It is efficient. As long as messages do not get lost or corrupted, it works fine.

- Drawbacks
  - When no reply message comes in, the client can possibly resend the request. Because the client can not detect whether the *original request message* was lost, or that *transmission of the reply failed*
  - If the reply was lost, then resending a request may result in performing the operation twice.(The operation my be harmful or harmless)
    - **Ex.** Transferring some amount of money from your account twice
  - When an operation can be repeated multiple times without harm, it is said to be ***idempotent***

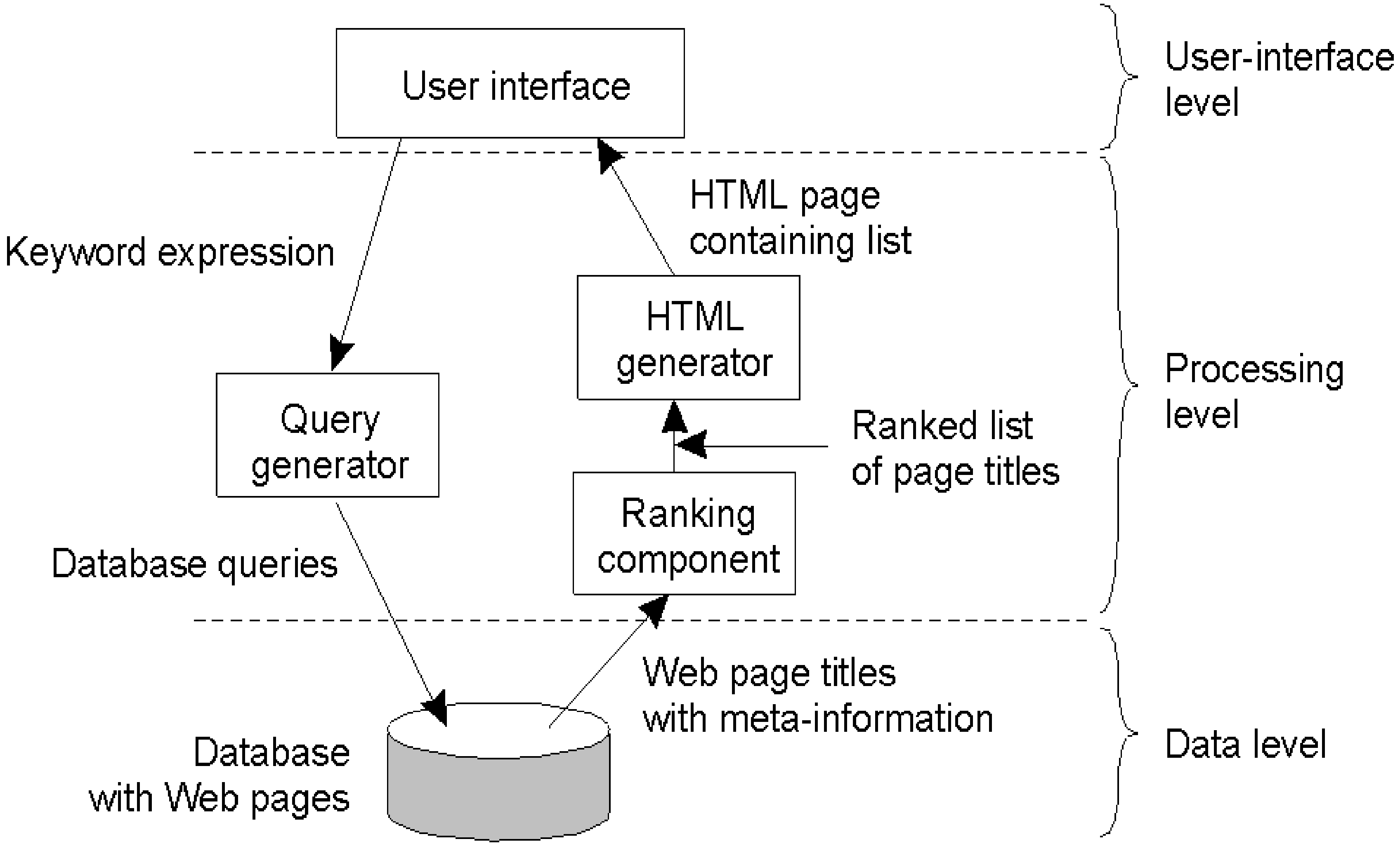
Connection Oriented Protocol

- Many client-server systems use a reliable connection oriented protocol (eg. EMS)
- Connection must be established before communication and terminated at the end.
- The client send request and the server replay to the requests using the same connection.
- Acknowledgement is required and lost messages must be retransmitted
- FTP and HTTP are examples of applications that use TCP
- Drawback: establishing and terminating connection is costly

Application Layer

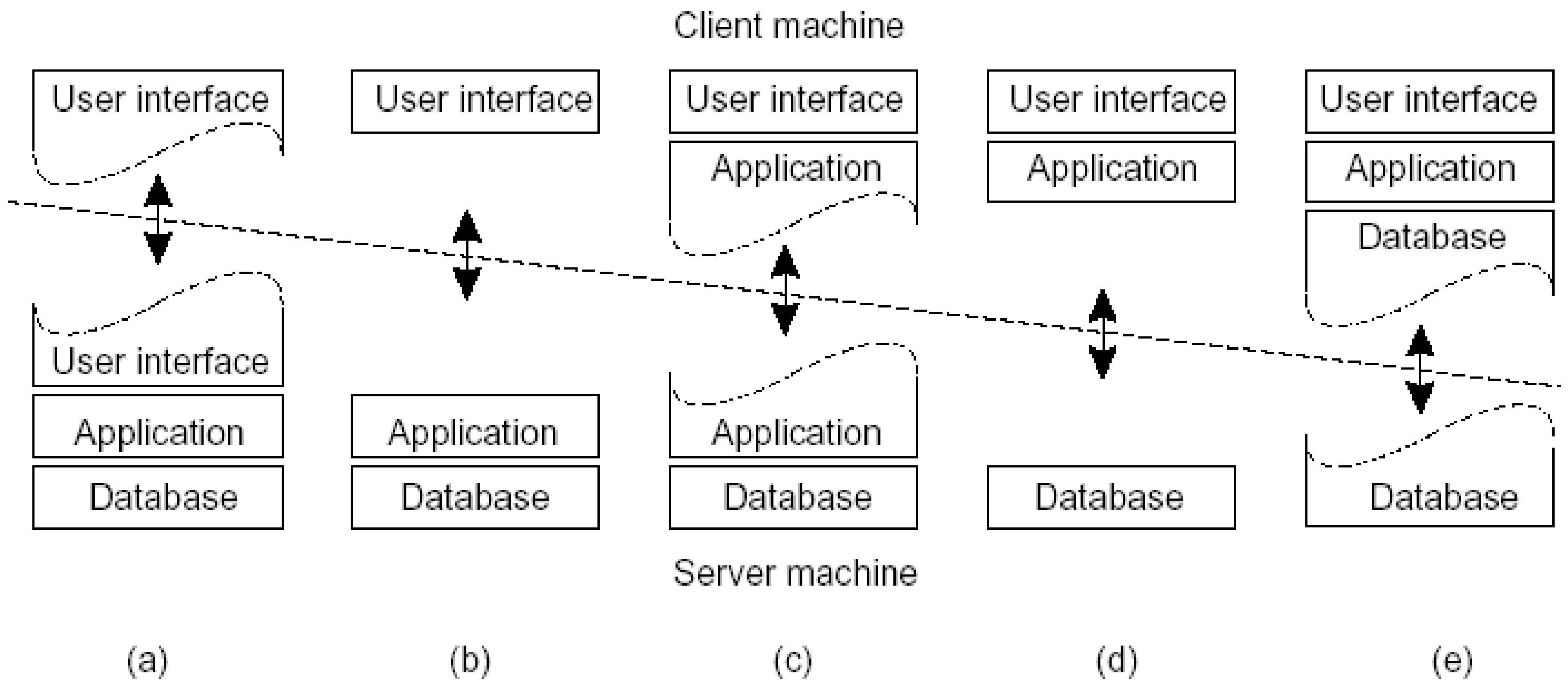
- In client-server model there is no clear distinction between a client and a server. For example, a server for a distributed database may act as a client when it forwards requests to different file servers responsible for implementing the database tables.
- client-server applications support user to access databases. The applications consists of three levels:
  - The user-interface level:
  - The processing level
  - The data level

- User interface level
  - contains all that is necessary to directly interface with the user.
  - Clients typically implement the user-interface level.
  - consists of the programs that allow end users to interact with applications.
- The Processing level contains applications.
- The data level manages the actual data that is being acted on.
- For example, in an Internet search engine


User interface
User-interface level
Keyword expression
HTML page containing list
HTML generator
Query generator
Processing level
Ranked list of page titles
Ranking component
Database queries
Web page titles with meta-information
Database with Web pages
Data level

Multitier Architectures

- The distinction into three logical levels suggests possibilities for physically distributing a client-server application across several machines.
- The *simplest organization* is to have only *two machines*
  - Client Machine: implement only user interface level(dump terminal)
  - Server Machine: implement process & data level
- The *other approach* is to distribute the programs in the application layers across *different machines*



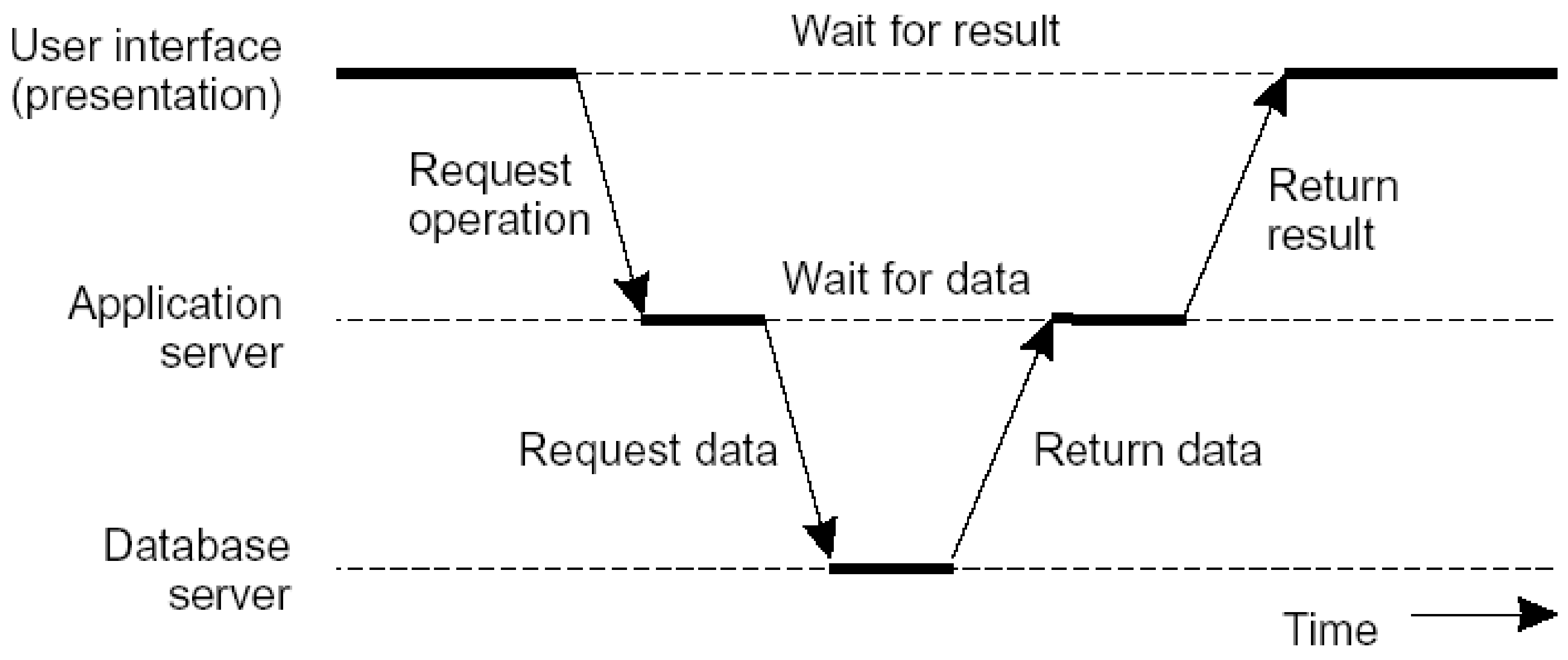
Physically Two-tiered architecture: alternative client-server organizations

In many client-server environments, the organizations shown in Fig. (d) and Fig. (e) are particularly popular

Essentially, most of the application is running on the client machine, but all operations on files or database entries go to the server.

For example, banking applications

•Server may sometimes need to act as a client



Three tiered architecture: an example of a server acting as a client

•In this architecture, programs that form part of the processing level reside on a separate server, but may additionally be partly distributed across the client and server machines.

Decentralized Architecture

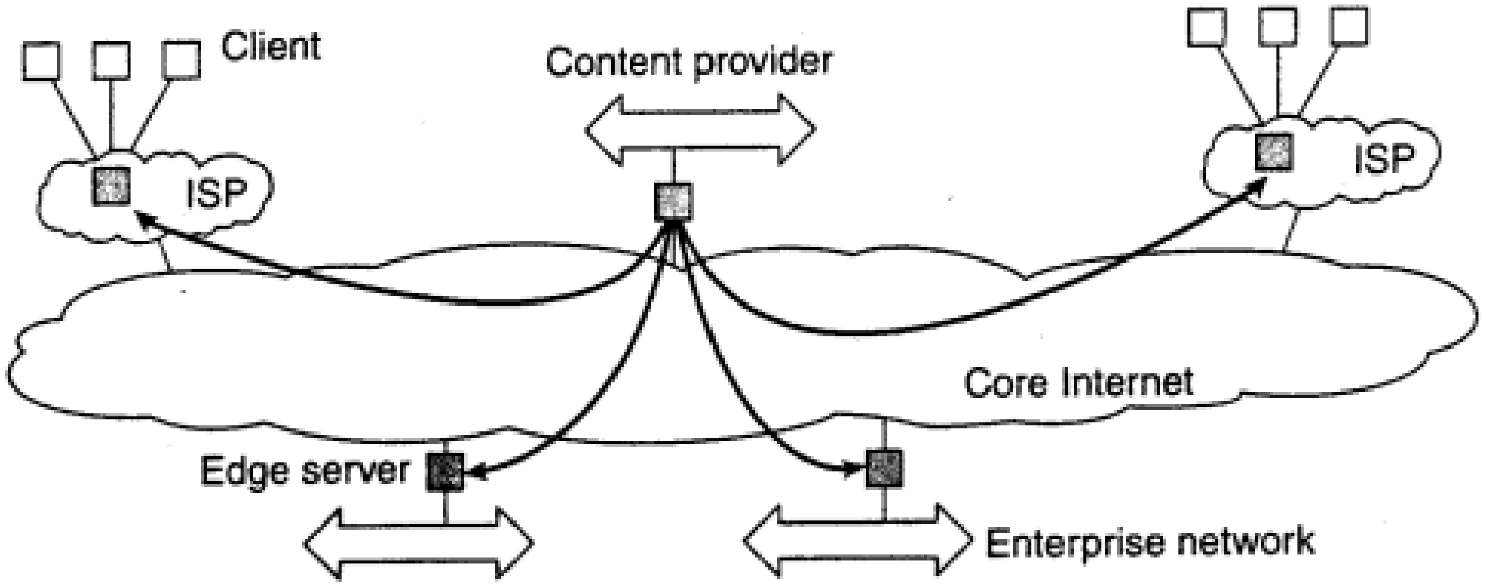
- Multi-tiered client-server architectures are also known as vertical distribution. It is achieved by placing logically different components on different machines.
- In modern architecture, the distribution is horizontal i.e. a client or server may be physically split up into logically equivalent parts, but each part is operating on its own share of the complete data set, thus balancing the load. Example: peer-to-peer system
- Much of the interaction between processes is symmetric: each process can act both as a client and a server at the same time

- Peer-to-peer architectures evolve an overlay network organization.
- In a structured peer-to-peer architecture, the overlay network is constructed using a deterministic procedure.
  - the most-used procedure is to organize the processes through a distributed hash table (DHT).
  - data items are assigned a random key from a large identifier space
  - nodes in the system are also assigned a random number from the same identifier space.
  - A request for a data item is accomplished by routing it to the responsible node.

- Unstructured peer-to-peer systems, rely on randomized algorithms for constructing an overlay network.
  - each node maintains a list of neighbors
  - data items are assumed to be randomly placed on nodes
  - when a node needs to locate a specific data item, the only thing it can effectively do is flood the network with a search query.

Hybrid Architecture

- Here client-server solutions are combined with decentralized architectures.
- In this architecture the distributed system is formed by edge-server systems
  - These systems are deployed on the Internet where servers are placed "at the edge" of the network. This edge is formed by the boundary between enterprise networks and the actual Internet
  - When end users at home connect to the Internet through their ISP, the ISP can be considered as residing at the edge of the Internet

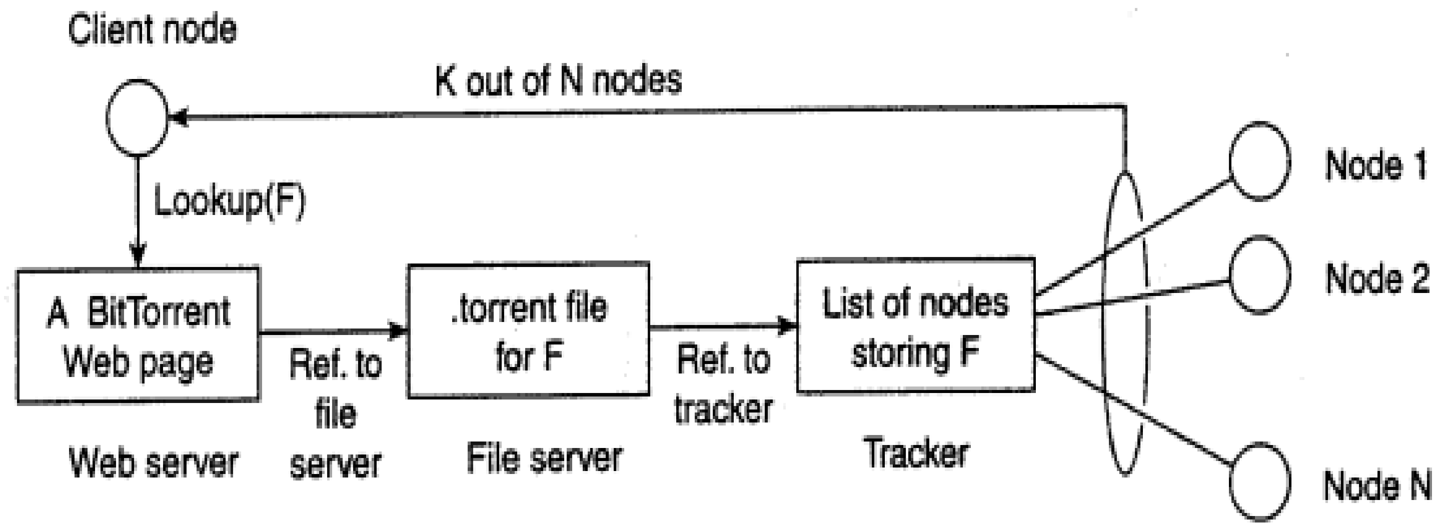


- End users or clients in general, connect to the Internet by means of an edge server.
- For a specific organization, one edge server acts as an origin server from which all content originates. That server can use other edge servers for replicating Web pages.

- Another instance of hybrid structure is that of collaborative distributed systems
- The main issue in many of these systems is to first get started, for which often a traditional client-server scheme is deployed.
- Once a node has joined the system, it can use a fully decentralized scheme for collaboration.

Ex.

- BitTorrent file-sharing system (a p2p file downloading system)
- Globule content distribution system



- To download a file
  - a user needs to access a global directory(web site)
  - Such a directory contains references to .torrent files
    - A .torrent file contains the information to be downloaded
  - it refers to a tracker: is a server that keeps an accurate account of active nodes that have (chunks) of the requested file.
    - there will generally be only a single tracker per file or collection of files

  - An active node is one that is currently downloading another file.
  - Once the nodes have been identified from where chunks can be downloaded, the downloading node effectively becomes active. At that point, it will be forced to help others.
  - This enforcement comes from a very simple rule: if node P notices that node Q is downloading more than it is uploading, P can decide to decrease the rate at which it sends data to Q.
- Super-peer network is also a hybrid network.