Distributed Systems

Delhi Technological University
Architecture
Instructor: Divyashikha Sethia
divyashikha@dce.edu

Architecture

- Distributed System consists of complex pieces of software. Components are dispersed across multiple machines
- Crucial to organize systems
- Software architectures: how software components are to be organized and how they should interact

i. **Traditional centralized architecture**
   - Single server implements software components
   - Remote clients access server using simple communication

ii. **Decentralized architectures**
   - Machines play equal roles

iii. **Hybrid organizations**

Architectural Styles

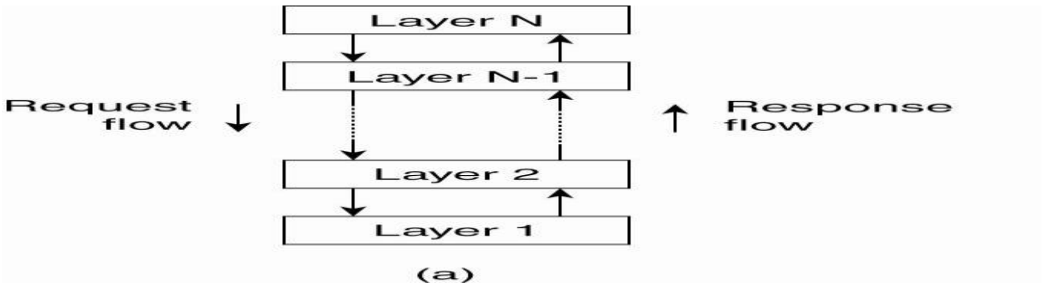
- Software Architecture: Logical organization of distributed systems into software components
- Component: Modular unit with well-defined requirements and provides interfaces that are replaceable within its environment
- Style is formulated in terms of:
  - Components
  - Interconnection of components
  - Inter component exchange of data

Architectural Styles …

Using components and connectors, important styles of architecture for distributed systems

- Layered architectures
- Object-based architectures
- Data-centered architectures
- Event-based architectures

Layered architectural style

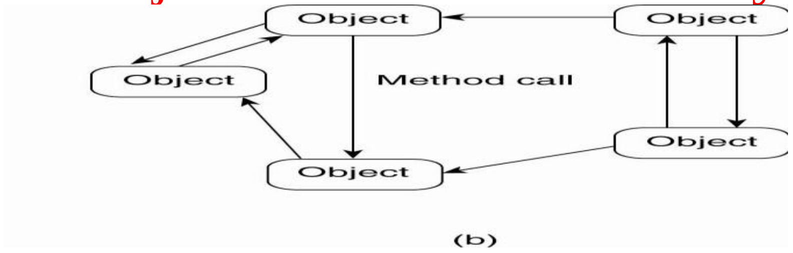


- Components are organized in layered fashion where a component at layer L; is allowed to call components at the underlying layer Li

- Adopted by the networking community

- Requests go down the hierarchy; Results flow upward

The object-based architectural style



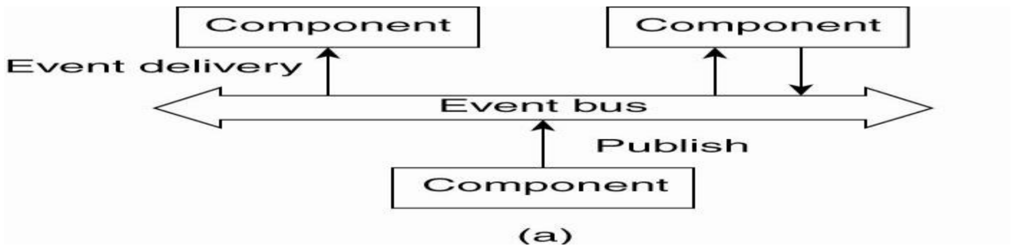
- Each object corresponds to a component
- Components connected through a (remote) procedure call mechanism

Data centered Architecture Style

- Processes communicate through a common (passive or active) repository

- Eg: networked applications communicate with each other through shared distributed file system

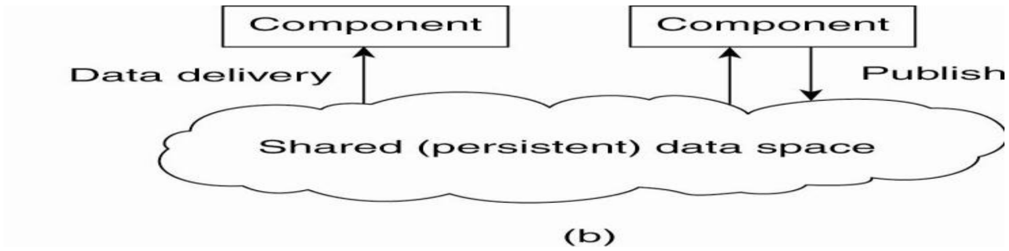
Event-based architectural style



(a)

- Processes communicate through events (optional data)
- Processes publish events and other processes subscribe to those events
- Middleware ensures only subscribed processes receive events
- Processes do not refer explicitly to each other

shared data-space architectural style



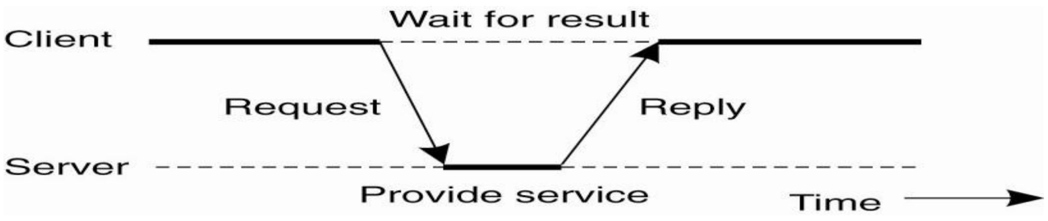
- Event-based architectures combined with data-centered architectures
- Processes need not both be active when communication takes place
- Data can be accessed using a description rather than an explicit reference

System Architecture

➢Organization of Distributed Systems by considering where software components are placed

- Centralized
- Distributed
- Hybrid

Centralized Architectures



- ➢Server: process implementing specific service, eg: file system service
- ➢Client: process that requests service from server by sending request &
- ➢waiting for server's reply
- ➢Connectionless protocol is efficient but not reliable if messages are lost
- ➢Connection oriented reliable protocol like TCP:
  - Sets up connection to server before sending request.
  - Sends requests to server and waits for reply
  - Acknowledgements for reliability
  - Connection establishment costly for small message exchange

Application Layering

•Issue: how to draw a clear distinction between a client and a server

•Server for a distributed database may continuously act as a client because it is forwarding requests to different file servers responsible for implementing the database tables

•Database server itself essentially does no more than process queries

Application Layering (1)

Layered Architectural Style:

➢ **The user-interface level**

- Software necessary to directly interface with user, such as display management
- Implemented by **clients** as programs that can interact with applications

➢ **The processing level**

- Contains applications

➢ **The data level**

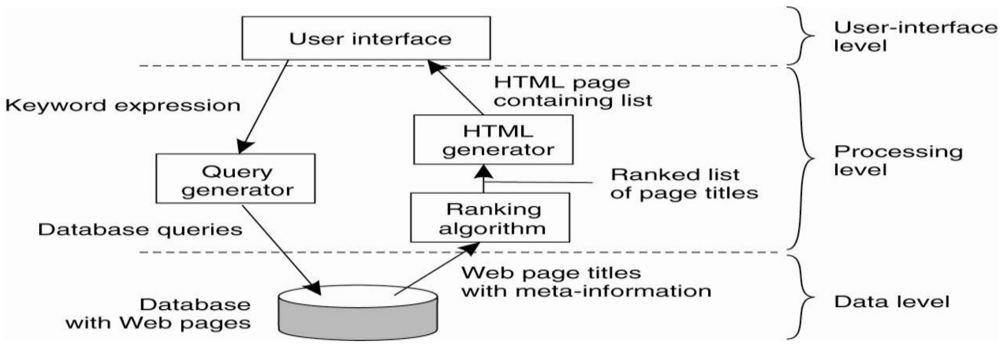
- Manages actual data that is being acted on.

Application Layering

Data Level:

- Data often persistent: stored somewhere even if no application is running for next use.
- Data level consists of a file system or database
- In the client-server model, the data level is typically implemented at the **server side**
- Also responsible for keeping data consistent across different applications

Organization of Internet search engine into three different layers.

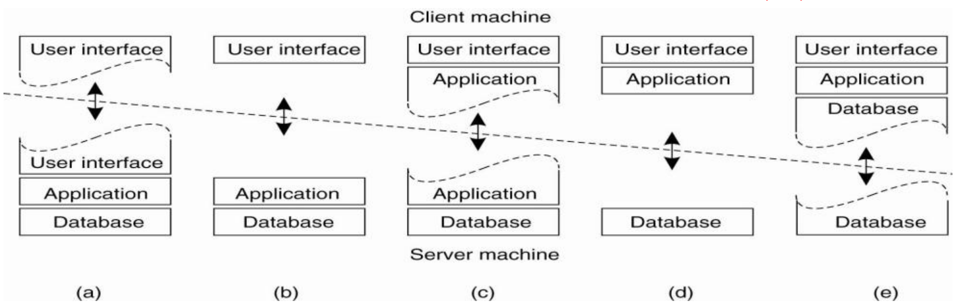


Multitiered Architectures (1)

Number of possibilities for physically distributing a client-server application across several machines
The simplest organization is to have only two types of machines
(Two tiered architecture)

- A client machine containing only the programs implementing (part of) the user-interface level
- A server machine containing the rest of the programs implementing the processing and data level

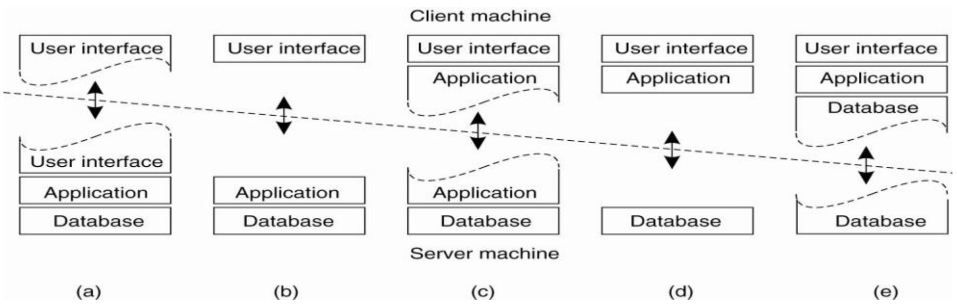
Multitiered Architectures (2)



(a) user interface: client machine and give applications remote control over the presentation of their data

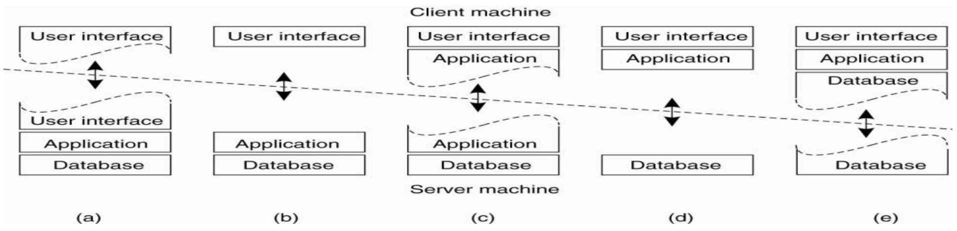
(b) user-interface: client side; graphical front end, which communicates with the rest of the application (residing at the server) through a protocol

Multitiered Architectures (2)



(c) move part of the application to the front end, application makes use of a form that needs to be filled in entirely before it can be processed, check the correctness and consistency of the form

Multitiered Architectures (2)



(d) –(e) Client connected to a distributed file system or database.

Most of the application is running on client machine

All operations on files or database entries go to server.

Eg: Banking applications run on an end-user's machine where user prepares transactions Once finished, application contacts database on bank's server and uploads the transactions for further processing.

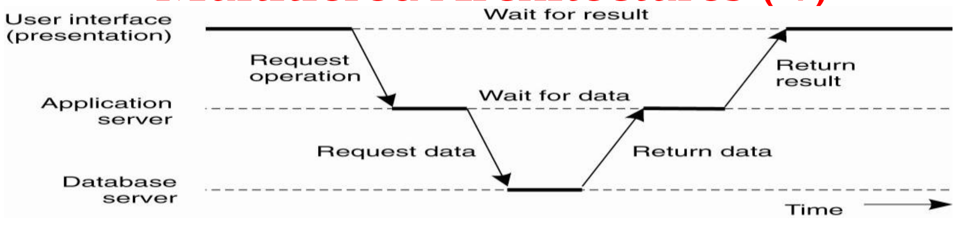
d) - client's local disk contains part of the data.

Multitiered Architectures (3)

•d - e: Fat clients not being preferred since have more functionality on client side which is prone to errors and dependent on client’s platform

• a - c: Thin clients are much easier , less sophisticated UI, client performance better

Multitiered Architectures (4)



- ➢ Distributed Server-side: single server being replaced by multiple servers
- ➢ Three tiered architecture :
  - programs for processing level reside on separate server / partly distributed across the client and server machines. Eg:

  i. Transaction processing

  ii. Web architecture: Web server acts as entry point to a site, passing requests to application server where actual processing takes place. Application server, in turn, interacts with a database server.

Decentralized Architecture

➢**Multitier client-server architectures:**

- Divide applications into user-interface, processing components, & data level
- Different tiers correspond directly with the logical organization of applications.
- Vertical distribution:
  - functions are logically and physically split across multiple machines, where each machine is tailored to a specific group of functions.

Decentralized Architecture:
-Horizontal Distribution: distribution of the clients and the servers physically split up into logically equivalent parts
-Each part is operating on its own share of complete data set, thus balancing the load Eg: peer-to-peer systems

Peer to Peer Systems

- ➢ No centralized control or hierarchical organization
- ➢ Each node runs software with equivalent functionality
- ➢ Processes are peers all equal
- ➢ Interaction between processes is symmetric : each process will act as a client and a server at the same time
- ➢ Peer-to-peer systems often implement an abstract overlay network, built at Application Layer, on top of the native or physical network topology.
- ➢ Overlays are used for indexing and peer discovery and make the P2P system independent from the physical network topology.
- ➢ Content is typically exchanged directly over the underlying Internet Protocol (IP) network

Peer to Peer Systems

➢How to organize the processes in an overlay network, that is, a network in which the nodes are formed by the processes and the links represent the possible communication channels

i. **Structured**
   - peers are organized following specific criteria and algorithms, which lead to overlays with specific topologies and properties
   - use distributed hash table-based (DHT) indexing, such as in the Chord system

ii. **Unstructured**
   - Do not provide any algorithm for organization or optimization of network connections

Structured Peer to Peer Systems

➢ overlay network is constructed using a deterministic procedure.

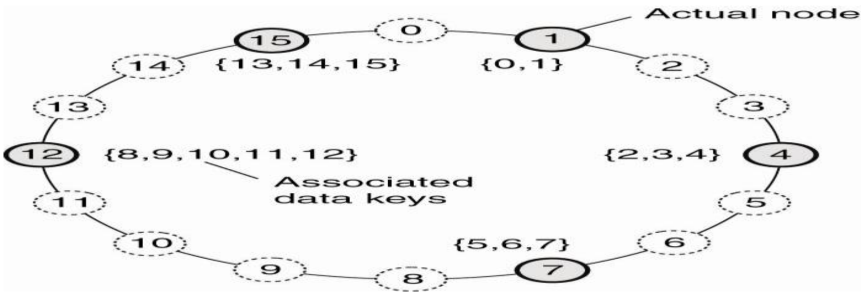
➢ Organize processes through **distributed hash table (DHT).**
- Data items assigned random key from large identifier space,
- (128-bit or 160-bit identifier) Data -> Key
- Nodes are also assigned a random number from the same identifier space.
- Implement an efficient and deterministic scheme that uniquely **maps the key of a data item to the identifier of a node** based on some distance metric
- Key -> NodeId
- When **looking up a data item, returns the network address of the node** responsible for that data item.
- Directly route request for a data item to responsible node

Chord Structured Peer to Peer

➢ Fundamental problem: efficient location of node that stores desired data
➢ Chord provides main operation: given a key, it maps the key onto a node.
➢ Data location can be easily implemented on top of Chord by associating a key with each data item, and storing the key/data pair at the node to which the key maps.
➢ Chord adapts efficiently as nodes join and leave the system, and can answer queries even if the system is continuously changing.
➢ Uses DHT to assign keys to chord nodes which balances load such each node gets the same number of keys and requires less movement of keys when node join or leave the system

Paper: http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf

Chord: Structured Peer-to-Peer Architectures (1)



- ➢Nodes are logically organized in a ring such that a data item with key k is mapped to the node with the smallest identifier id >= k. Node is called successor of key k and denoted as succ(k)
- ➢Arbitrary node that requires a data lookup
  - Calls function LOOKUP(k) returns network address of succ(k) i.e the node mapped to key k and has data associated with the key k.
  - Contact node succ(k) to get the data copy

Chord

- ➢ Looking up a key does not follow the logical organization of nodes in the ring
- ➢ each node will maintain shortcuts to other nodes in such a way that lookups can generally be done in O(log (N) number of steps, where N is the number of nodes participating in the overlay.

Chord : Joining of a Node

- Joining node generates a random identifier *id.*
- node can simply do a lookup on *id, which will return the network* address of *succ(id)*
- joining node can simply contact *succ(id) and its predecessor and insert itself in the ring (requires* each node also stores information on its predecessor)
- Each data item whose key is now associated with node *id, is transferred* from *succ(id)*

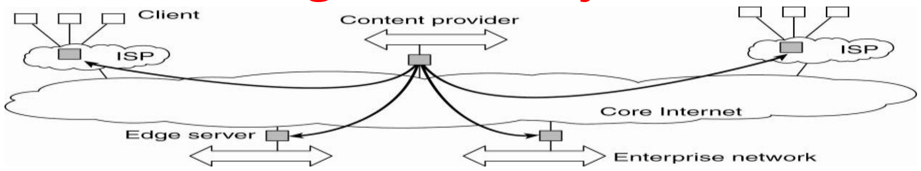
Unstructured Peer to Peer

- Randomized algorithms for constructing an overlay network
- Each node maintains a list of neighbors which is constructed in a random way data items are assumed to be randomly placed on nodes
- Locate a specific data item, the only thing it can effectively do is flood the network with a search query

Hybrid Architectures

•Client-server solutions are combined with decentralized architectures.

Edge-Server Systems



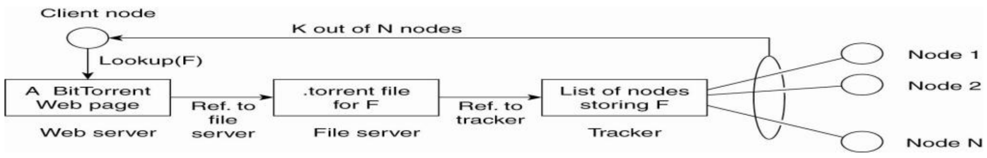
Viewing the Internet as consisting of a collection of edge servers.

- Servers are placed "at the edge" of network. Edge is boundary between enterprise networks and actual Internet, for example, as provided by an Internet Service Provider (ISP).
- Edge server's main purpose: serve content
- Collection of edge servers used to optimize content & application distribution
- One edge server acts as origin server from which all content originates and uses other edge servers for replicating Web pages etc.

Collaborative Distribution (BitTorrent)

- End user looking for a file: downloads chunks of file from other users until the downloaded chunks can be assembled together yielding complete file.
- Unlike most of file-sharing systems where a significant fraction of participants merely download files but otherwise contribute
- Close to nothing, this requires that a file can be downloaded only when downloading client is providing content to someone else. ("tit-for-tat")

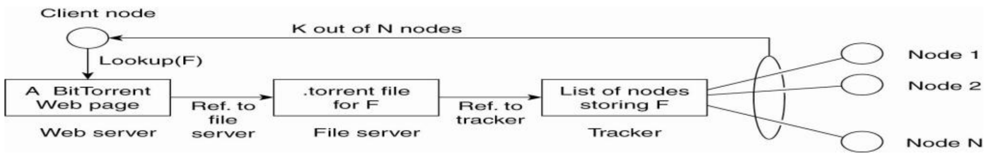
Collaborative Distributed Systems (1)



peer-to-peer file downloading system

- To download user needs to access a global directory(one of a few well-known Web sites.)
- Directory contains references to what are called *.torrent files.*
- *A .torrent file contains the information that is needed to* download a specific file: Refers to a tracker, that is keeping an accurate account of *active nodes that have* (chunks) of the requested file.
- An active node is one that is currently downloading another file.
- There will be many different trackers

Collaborative Distributed Systems (1)



peer-to-peer file downloading system

- Generally only a single tracker per file (or collection of files).
- Once downloading node identifies the nodes from where chunks can be downloaded, the downloading node effectively becomes active.
- Now it will be forced to help others, eg: by providing chunks of the file it is downloading that others do not yet have.
- 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*
- BitTorrent combines centralized with decentralized solutions

Summary

- Distributed systems can be organized in many different ways.
  - System Architecture: components that constitute a distributed system are placed across
  - Software Architecture: more concerned about the logical organization of the software: how do components interact
- Styles include layering, object orientation, event orientation, & data-space orientation
- Centralized Client server architecture
- Decentralized peer to peer architecture:
  processes are organized into an overlay network: a logical network in which every process has a local list of other peers that it can communicate with
  - Structure: deterministic schemes can be deployed for routing messages between processes
  - unstructured networks, the list of peers is more or less random, implying that search algorithms need to be deployed for locating data or other processes

Resources

•Distributed Systems - Tanenbaum
•http://www.jatit.org/research/introduction_grid_computing.htm
•http://www.thepicky.com/tech/difference-cloud-computing-vs-grid-computing/
•http://pdos.csail.mit.edu/chord/papers/paper-ton.pdf
http://www.informatics.sussex.ac.uk/courses/dist-sys/node11.html
http://pdos.csail.mit.edu/papers/chord:sigcomm01/chord_sigcomm.pdf