UNIT 1: Introduction
UNIT 2: Architecture

Yuba Raj Devkota | NCCS

BCA 6th Sem

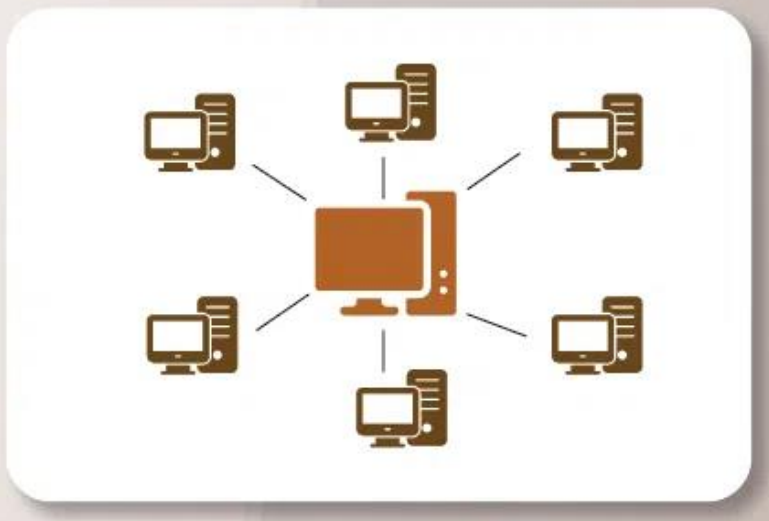
Unit 1. Introduction **4 Hrs.**

1.1 Characteristics
1.2 Design Goals
1.3 Types of Distributed Systems
1.4 Case Study: The World Wide Web

Unit 2. Architecture **4 Hrs.**

2.1 Architectural Styles
2.2 Middleware organization
2.3 System Architecture
2.4 Example Architectures

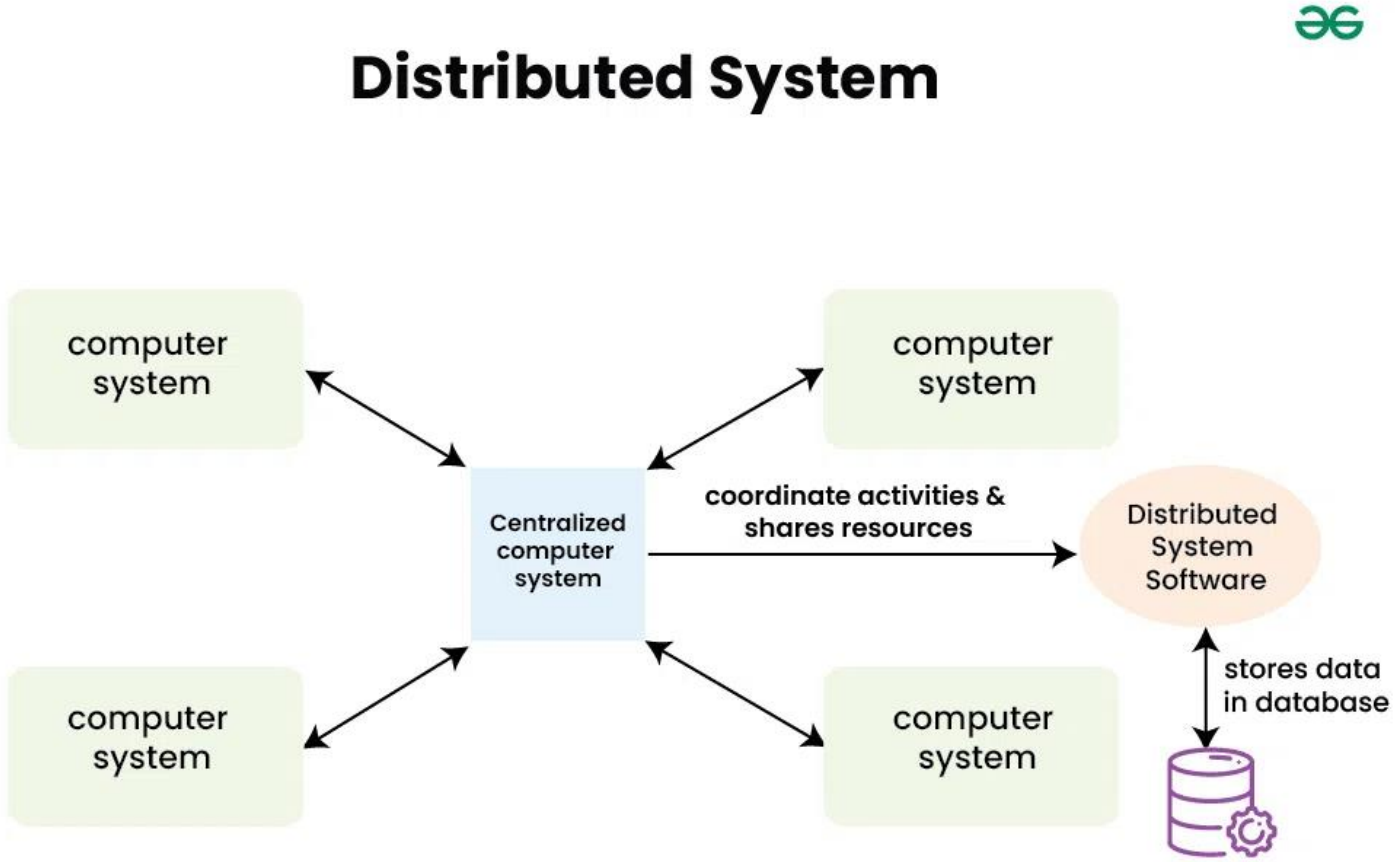
What is a Distributed System?



- A distributed System is a collection of autonomous computer systems that are physically separated but are connected by a centralized computer network that is equipped with distributed system software.
- The autonomous computers will communicate among each system by sharing resources and files and performing the tasks assigned to them.

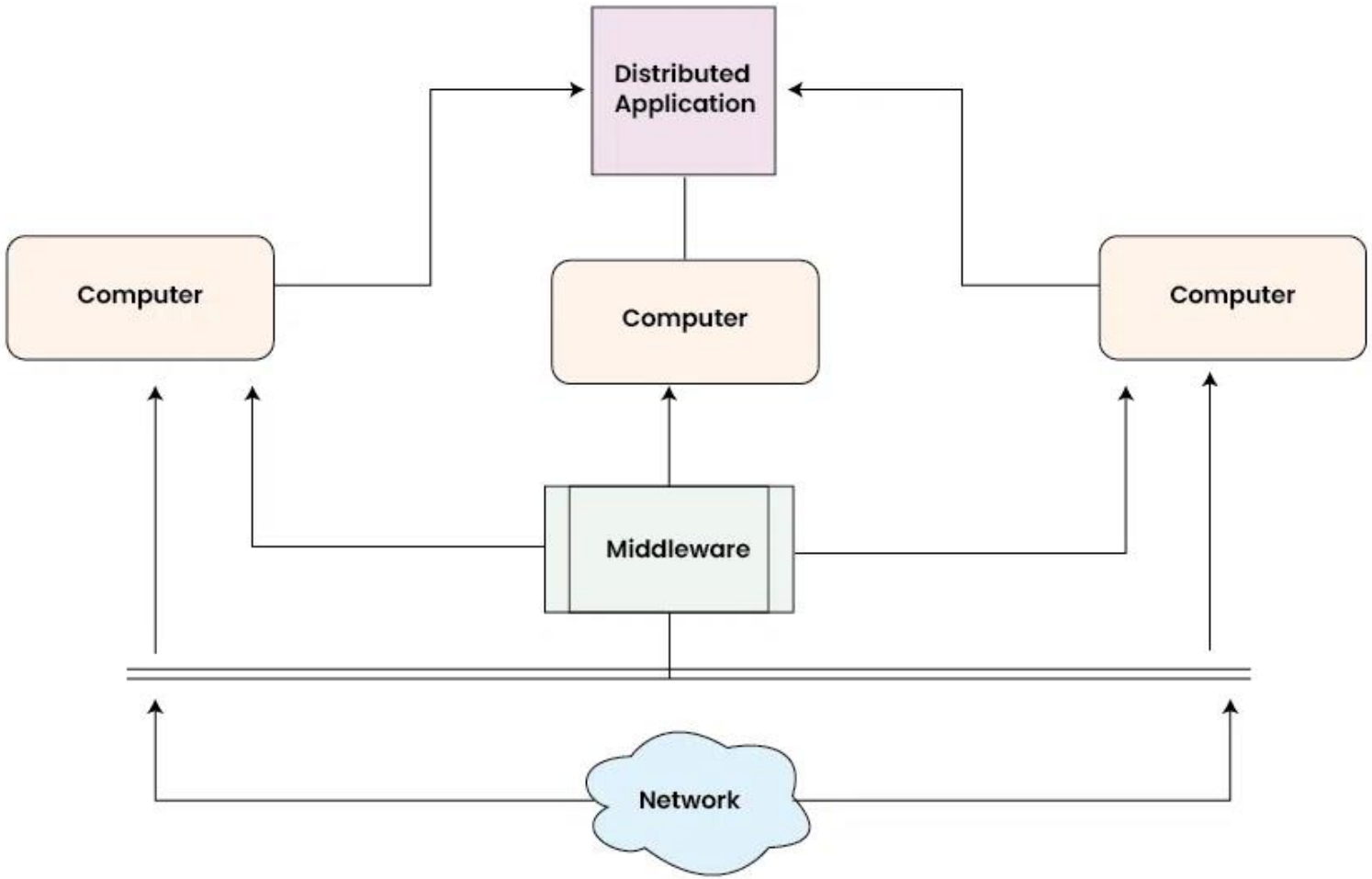
Example of a Distributed System

Any Social Media can have its Centralized Computer Network as its Headquarters and computer systems that can be accessed by any user and using their services will be the Autonomous Systems in the Distributed System Architecture.



- **Distributed System Software:** This Software enables computers to coordinate their activities and to share the resources such as Hardware, Software, Data, etc.
- **Database:** It is used to store the processed data that are processed by each Node/System of the Distributed systems that are connected to the Centralized network.

Working of Distributed System

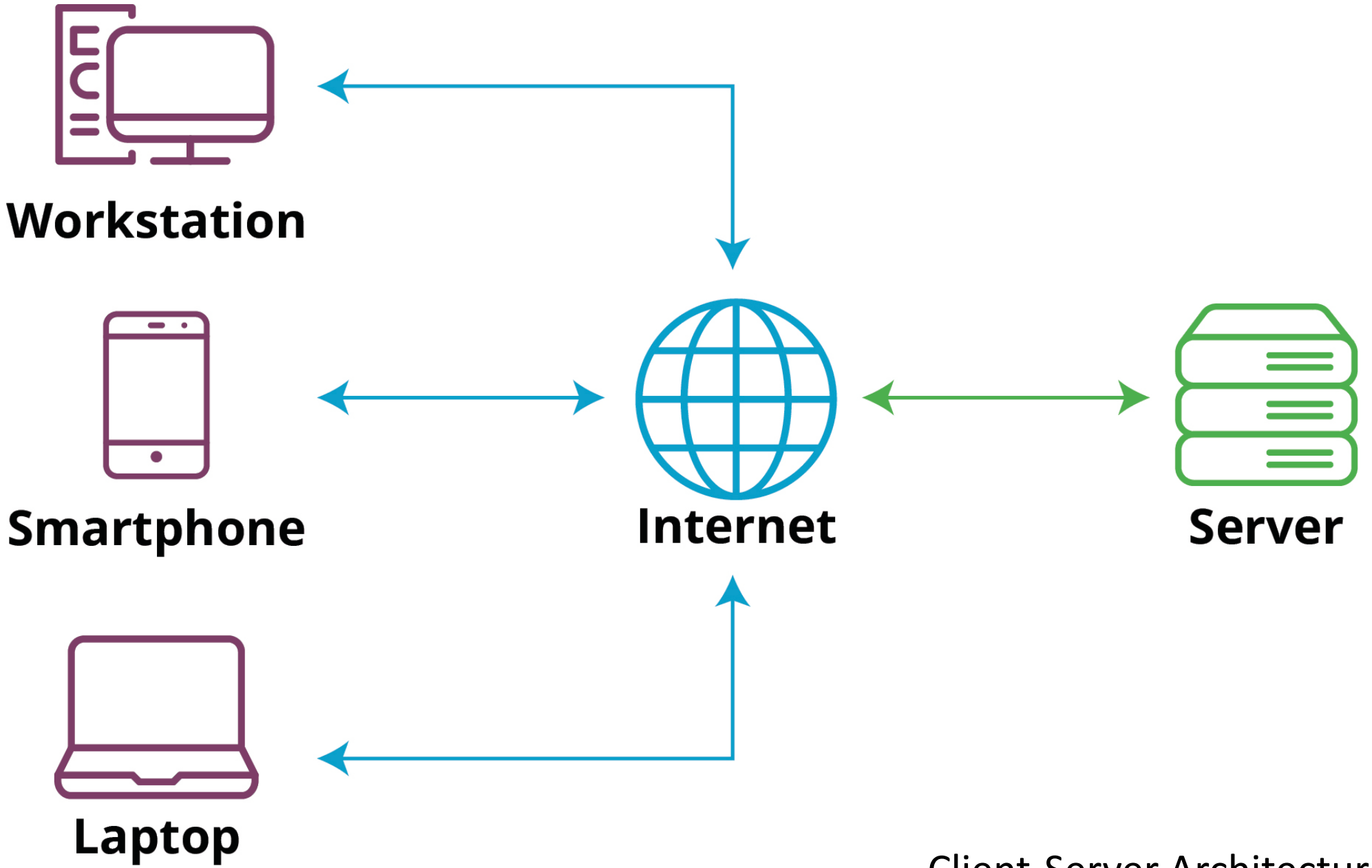


- As we can see that each Autonomous System has a common Application that can have its own data that is shared by the Centralized Database System.
- To Transfer the Data to Autonomous Systems, Centralized System should be having a Middleware Service and should be connected to a Network.
- Middleware Services enable some services which are not present in the local systems or centralized system default by acting as an interface between the Centralized System and the local systems. By using components of Middleware Services systems communicate and manage data.
- The Data which is been transferred through the database will be divided into segments or modules and shared with Autonomous systems for processing.
- The Data will be processed and then will be transferred to the Centralized system through the network and will be stored in the database.

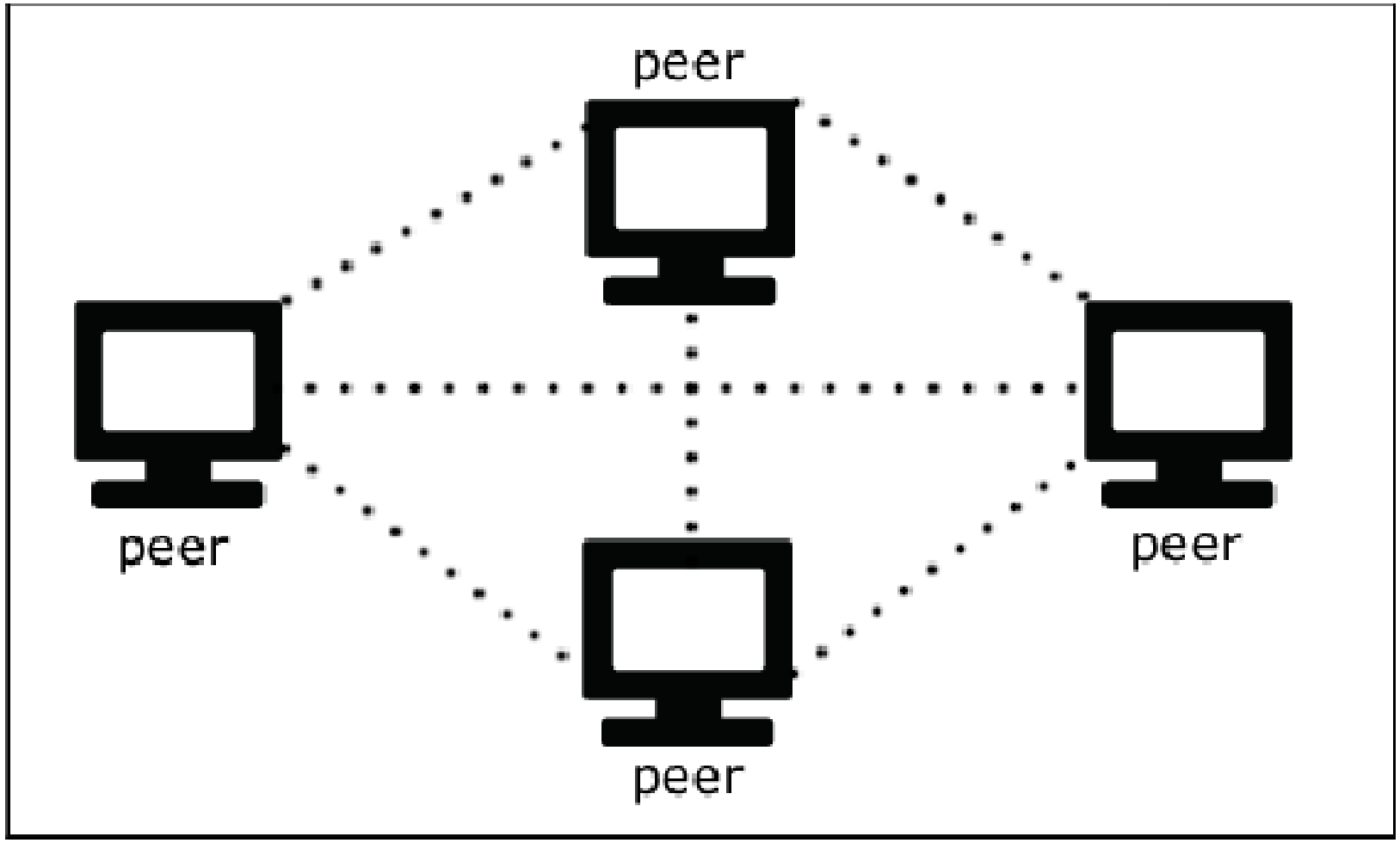
Types of Distributed System

1. **Client-server systems:** The most traditional and simple type of distributed system, involves a multitude of networked computers that interact with a central server for data storage, processing, or other common goal.
2. **Peer-to-peer networks:** They distribute workloads among hundreds or thousands of computers all running the same software.
3. **Cell phone networks:** It is an advanced distributed system, sharing workloads among handsets, switching systems, and internet-based devices.

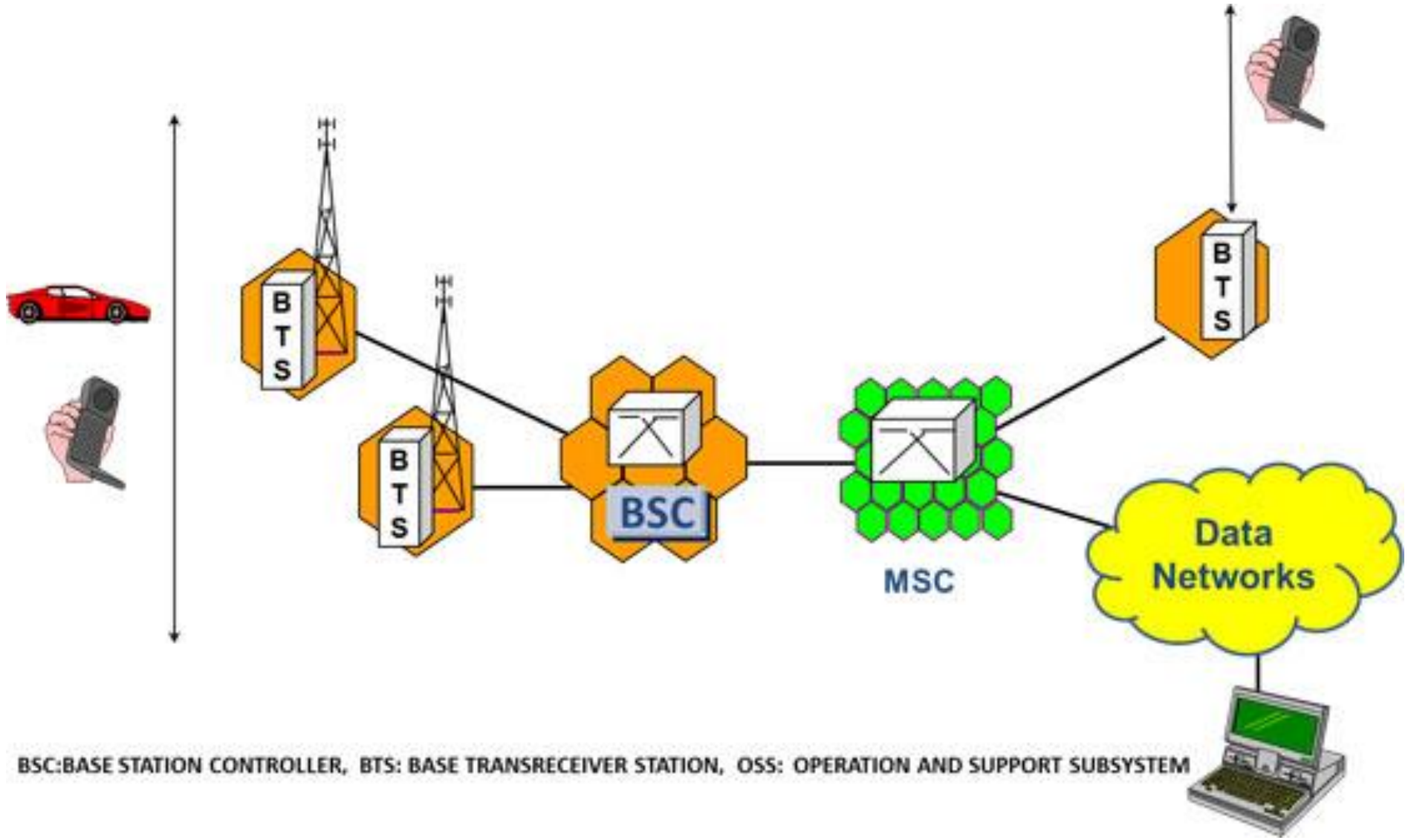
The most common forms of distributed systems today operate over the internet, handing off workloads to dozens of cloud-based virtual server instances that are created as needed, and then terminated when the task is complete.



Client-Server Architecture



Peer-To-Peer architecture



Cellular Network Architecture

Characteristics of Distributed System

- **Resource Sharing:** It is the ability to use any Hardware, Software, or Data anywhere in the System.
- **Openness:** It is concerned with Extensions and improvements in the system (i.e., How openly the software is developed and shared with others)
- **Concurrency:** It is naturally present in Distributed Systems, that deal with the same activity or functionality that can be performed by separate users who are in remote locations. Every local system has its independent Operating Systems and Resources.
- **Scalability:** It increases the scale of the system as a number of processors communicate with more users by accommodating to improve the responsiveness of the system.
- **Fault tolerance:** It cares about the reliability of the system if there is a failure in Hardware or Software, the system continues to operate properly without degrading the performance the system.
- **Transparency:** It hides the complexity of the Distributed Systems to the Users and Application programs as there should be privacy in every system.
- **Heterogeneity:** Networks, computer hardware, operating systems, programming languages, and developer implementations can all vary and differ among dispersed system components.

Advantages of Distributed System

- Applications in Distributed Systems are Inherently Distributed Applications.
- Information in Distributed Systems is shared among geographically distributed users.
- Resource Sharing (Autonomous systems can share resources from remote locations).
- It has a better price performance ratio and flexibility.
- It has shorter response time and higher throughput.
- It has higher reliability and availability against component failure.
- It has extensibility so that systems can be extended in more remote locations and also incremental growth.

Disadvantages of Distributed System

- Relevant Software for Distributed systems does not exist currently.
- Security possess a problem due to easy access to data as the resources are shared to multiple systems.
- Networking Saturation may cause a hurdle in data transfer i.e., if there is a lag in the network then the user will face a problem accessing data.
- In comparison to a single user system, the database associated with distributed systems is much more complex and challenging to manage.
- If every node in a distributed system tries to send data at once, the network may become overloaded.

Use cases of Distributed System

- **Finance and Commerce:** Amazon, eBay, Online Banking, E-Commerce websites.
- **Information Society:** Search Engines, Wikipedia, Social Networking, Cloud Computing.
- **Cloud Technologies:** AWS, Salesforce, Microsoft Azure, SAP.
- **Entertainment:** Online Gaming, Music, youtube.
- **Healthcare:** Online patient records, Health Informatics.
- **Education:** E-learning.
- **Transport and logistics:** GPS, Google Maps.
- **Environment Management:** Sensor technologies.

Challenges of Distributed Systems

While distributed systems offer many advantages, they also present some challenges that must be addressed. These challenges include:

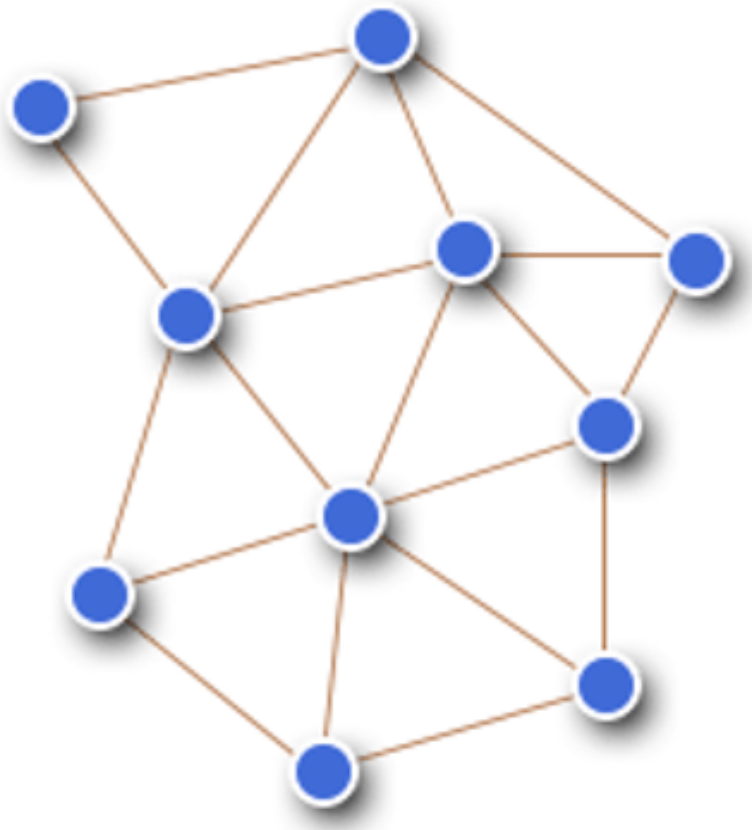
- **Network latency:** The communication network in a distributed system can introduce latency, which can affect the performance of the system.
- **Distributed coordination:** Distributed systems require coordination among the nodes, which can be challenging due to the distributed nature of the system.
- **Security:** Distributed systems are more vulnerable to security threats than centralized systems due to the distributed nature of the system.
- **Data consistency:** Maintaining data consistency across multiple nodes in a distributed system can be challenging.

Architecture Styles in Distributed Systems

Distributed systems are a set of autonomous computers that appears to be a single coherent system to its users from outside. There are actually two different types of systems that exist in prospective of computers:

- **Centralized System :**
  - A centralized system consists of a single machine.
  - All calculations are done by a particular computer.
  - Its performance is low as the workload is not divided.
  - There is also no machine present in backup if the original computer system fails.
- **Distributed Systems :**
  - A distributed system consists of multiple machines.
  - All computation work is divided among the different systems.
  - Its performance is high as the workload is divided among different computers to efficiently use their capacity.
  - There are systems present in backup, so if the main system fails then work will not stop.

Centralized



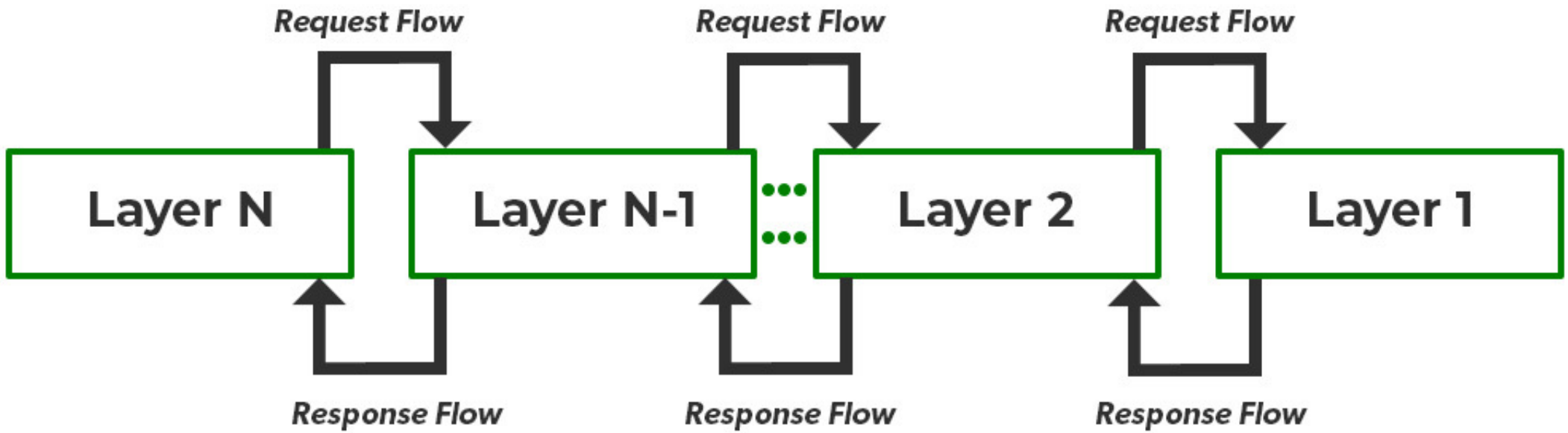
Distributed

Architecture Styles:

To show different arrangement styles among computers Architecture styles are proposed.

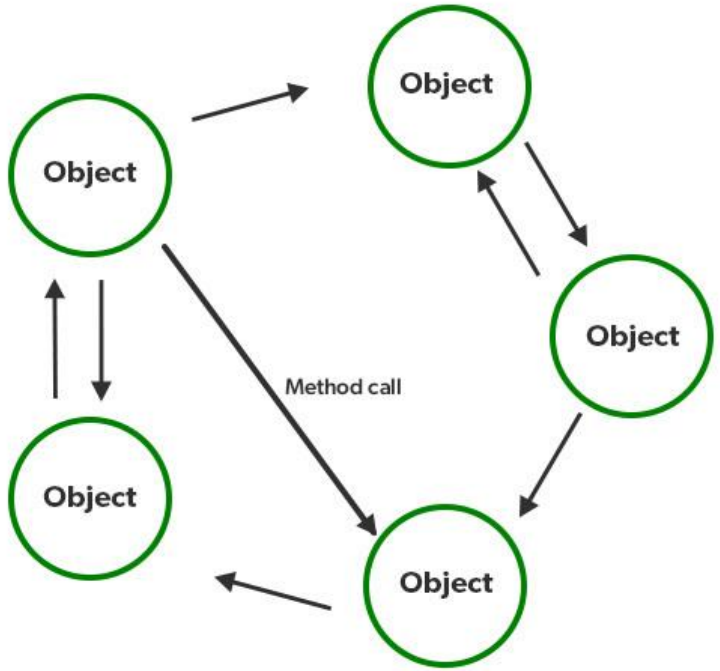
1. Layered Architecture:

- In Layered architecture, different components are organized in layers. Each layer communicates with its adjacent layer by sending requests and getting responses. The layered architecture separates components into units. It is an efficient way of communication. Any layer can not directly communicate with another layer. A layer can only communicate with its neighboring layer and then the next layer transfers information to another layer and so on the process goes on.
- In some cases, layered architecture is in cross-layer coordination. In a cross-layer, any adjacent layer can be skipped until it fulfils the request and provides better performance results. Request flow from top to bottom(downwards) and response flow from bottom to top(upwards). The advantage of layered architecture is that each layer can be modified independently without affecting the whole system. This type of architecture is used in Open System Interconnection (OSI) model.
- To the layers on top, the layers at the bottom offer a service. While the response is transmitted from bottom to top, the request is sent from top to bottom. This method has the advantage that calls always follow a predetermined path and that each layer is simple to replace or modify without affecting the architecture as a whole.


Request Flow
Request Flow
Request Flow
Layer N
Layer N-1
Layer 2
Layer 1
Response Flow
Response Flow
Response Flow

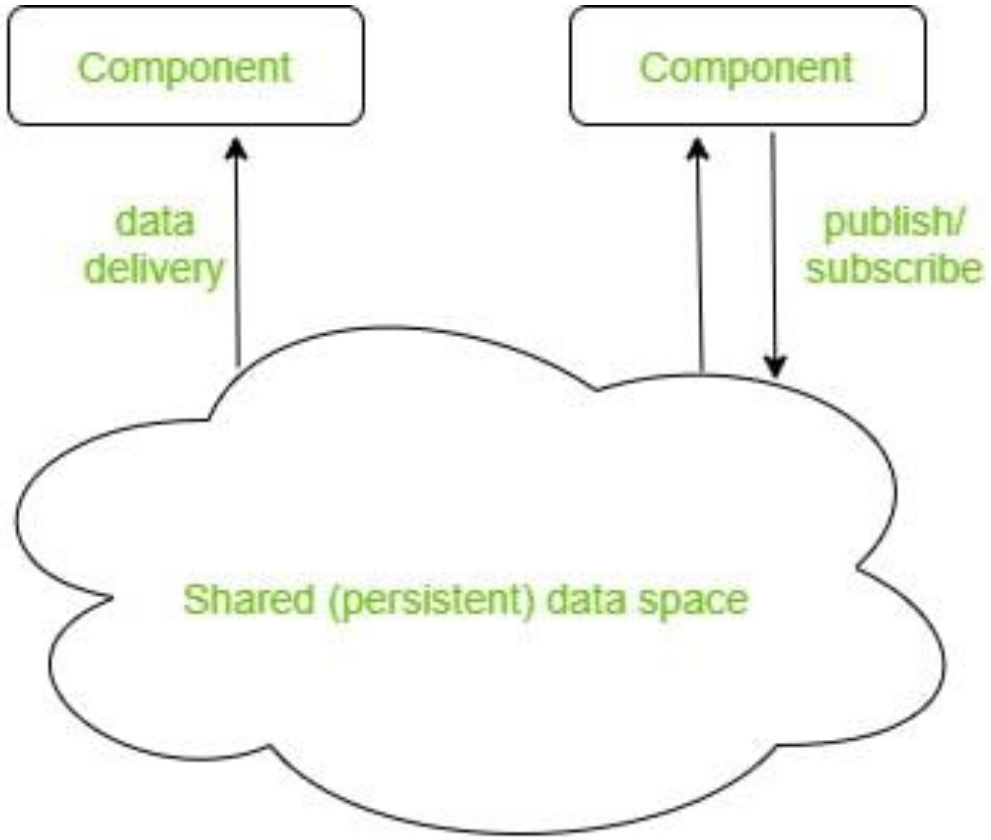
2. Object-Oriented Architecture:

- In this type of architecture, components are treated as objects which convey information to each other. Object-Oriented Architecture contains an arrangement of loosely coupled objects. Objects can interact with each other through method calls. Objects are connected to each other through the Remote Procedure Call (RPC) mechanism or Remote Method Invocation (RMI) mechanism.
- Web Services and REST API are examples of object-oriented architecture. Invocations of methods are how objects communicate with one another. Typically, these are referred to as Remote Procedure Calls (RPC). REST API Calls, Web Services, and Java RMI are a few well-known examples. These characteristics apply to this.

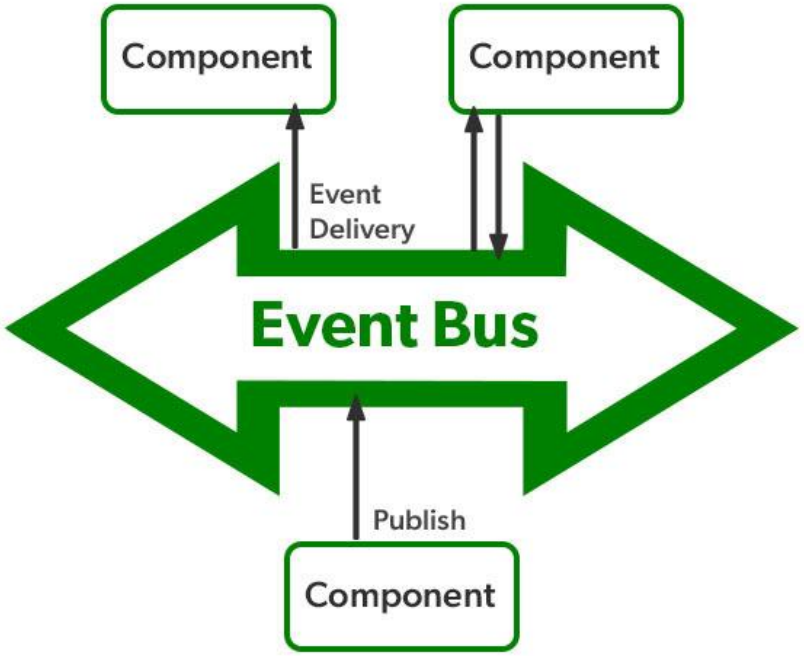


3. Data Centered Architecture:

- Data Centered Architecture is a type of architecture in which a common data space is present at the center. It contains all the required data in one place a shared data space. All the components are connected to this data space and they follow publish/subscribe type of communication. It has a central data repository at the center. Required data is then delivered to the components. Distributed file systems, producer-consumer systems, and web-based data services are a few well-known examples.
- For example Producer-Consumer system. The producer produces data in common data space and consumers request data.

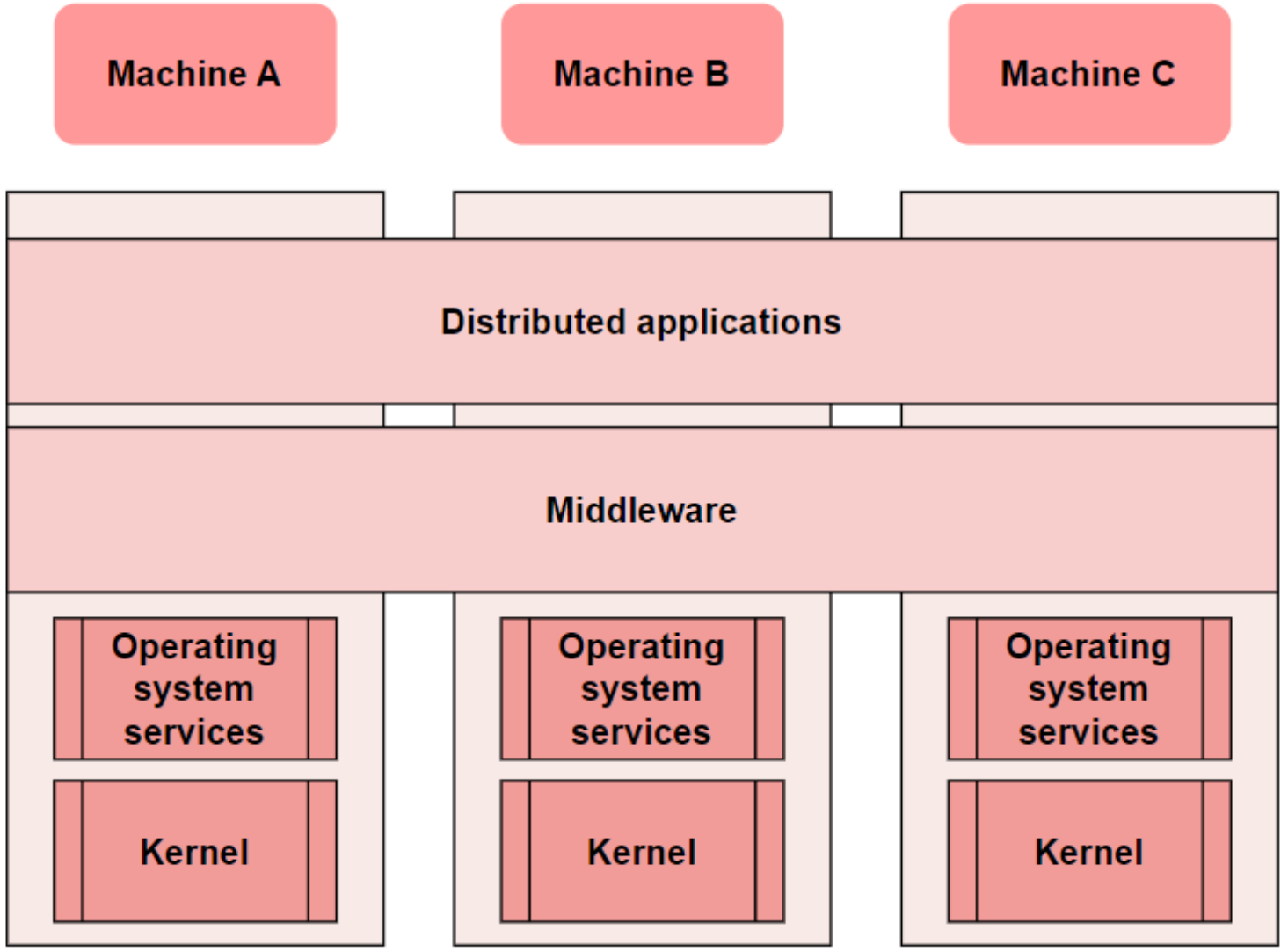


- **4. Event-Based Architecture:**
  - Event-Based Architecture is almost similar to Data centered architecture just the difference is that in this architecture events are present instead of data. Events are present at the center in the Event bus and delivered to the required component whenever needed. In this architecture, the entire communication is done through events.
  - When an event occurs, the system, as well as the receiver, get notified. Data, URLs etc are transmitted through events. The components of this system are loosely coupled that's why it is easy to add, remove and modify them. Heterogeneous components can communicate through the bus. One significant benefit is that these heterogeneous components can communicate with the bus using any protocol.
  - However, a specific bus or an ESB has the ability to handle any kind of incoming request and respond appropriately.



Role of Middleware in Distributed System

- In a distributed system, **middleware** is a software component that serves between two or more applications.
- Middleware usually resides between the operating system and the end user or end-user application. It provides essential features that the operating system doesn't offer. The term usually refers to large software products, such as database managers, transaction monitors, and web servers.



An example of how a middleware would exist in a distributed system application

Why use middleware?

Middleware can perform numerous functions such as:

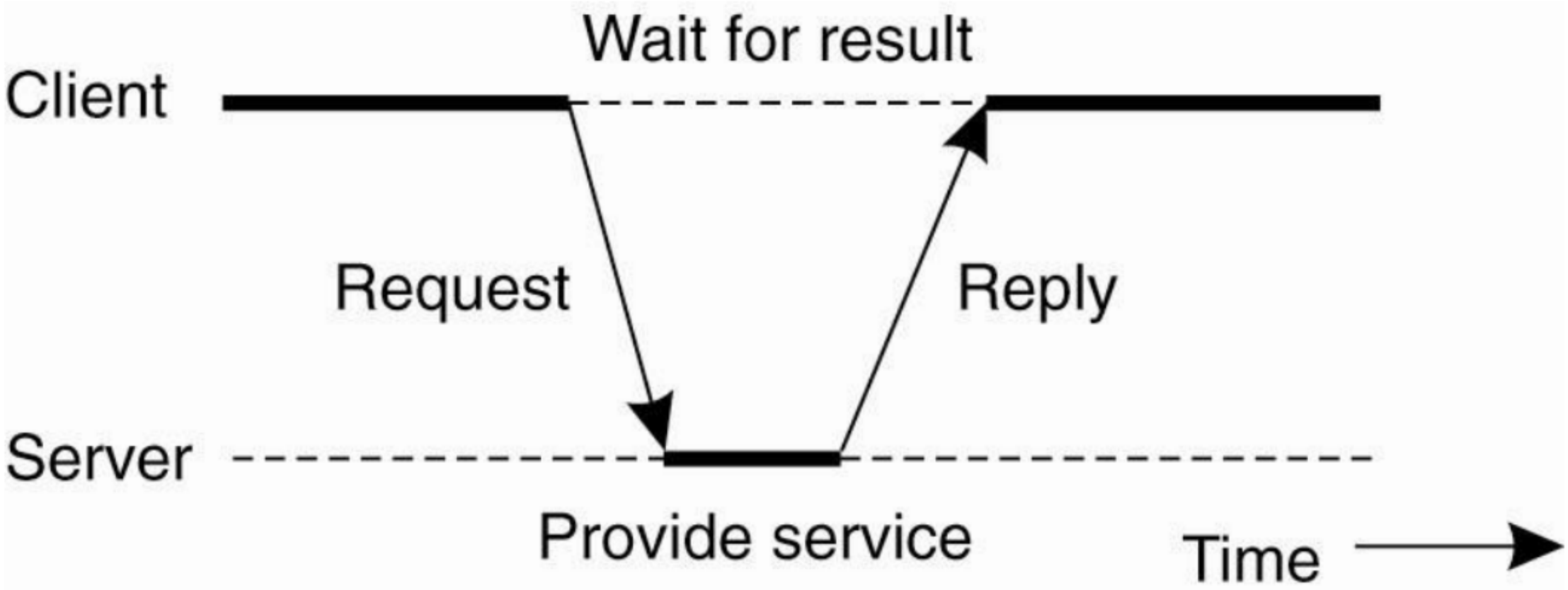
- It manages connections to various backend resources. Middleware components can create connection pools to provide fast and efficient access to popular backend databases.
- Second, middleware software can implement logic based on client requirements. For example, the middleware component can detect that the language header of the client browser making a particular request is set to English. As a result, queries sent to the backend will adjust to returning only English-based results.
- Third, middleware is essential in concurrency, load balancing, and transaction management. It's scaled up and down to distribute incoming client requests across multiple servers, virtual machines, or cloud availability zones.
- Finally, middleware plays an essential role in protecting access to backend resources. It requires a secure connection using technology such as SSL and a username/password combination or digital certificate authentication.
- For example, Message-oriented middleware is designed for the purpose of transporting messages between two or more applications and is best suited for distributed applications that require transaction-oriented messaging. It could be used to monitor network traffic flows or to monitor the health of a distributed system.

System Architectures

- Centralized Architectures
- Decentralized Architectures
- Hybrid Architectures

Centralized Architectures

- Client-server model:
  - Processes are divided into two (possibly overlapping) groups
  - **Server**: a process implementing a specific service
    for example, a file system service or a database service.
  - **Client**: a process sending a request to a server and subsequently waiting for the server's reply


Wait for result
Client
Request
Reply
Server
Provide service
Time

Communication between Clients and Servers

- Connectionless protocol
  - Efficient, but unreliable
    - Good for LANs

In these cases, when a client requests a service, it simply packages a message for the server, identifying the service it wants, along with the necessary input data. The message is then sent to the server. The latter, in turn, will always wait for an incoming request, subsequently process it, and package the results in a reply message that is then sent to the client

- Connection-oriented protocol
  - Inefficient, but reliable
    - Good for WANs

For example, virtually all Internet application protocols are based on reliable TCP/IP connections. In this case, whenever a client requests a service, it first sets up a connection to the server before sending the request. The server generally uses that same connection to send the reply message, after which the connection is torn down. The trouble is that setting up and tearing down a connection is relatively costly, especially when the request and reply messages are small.

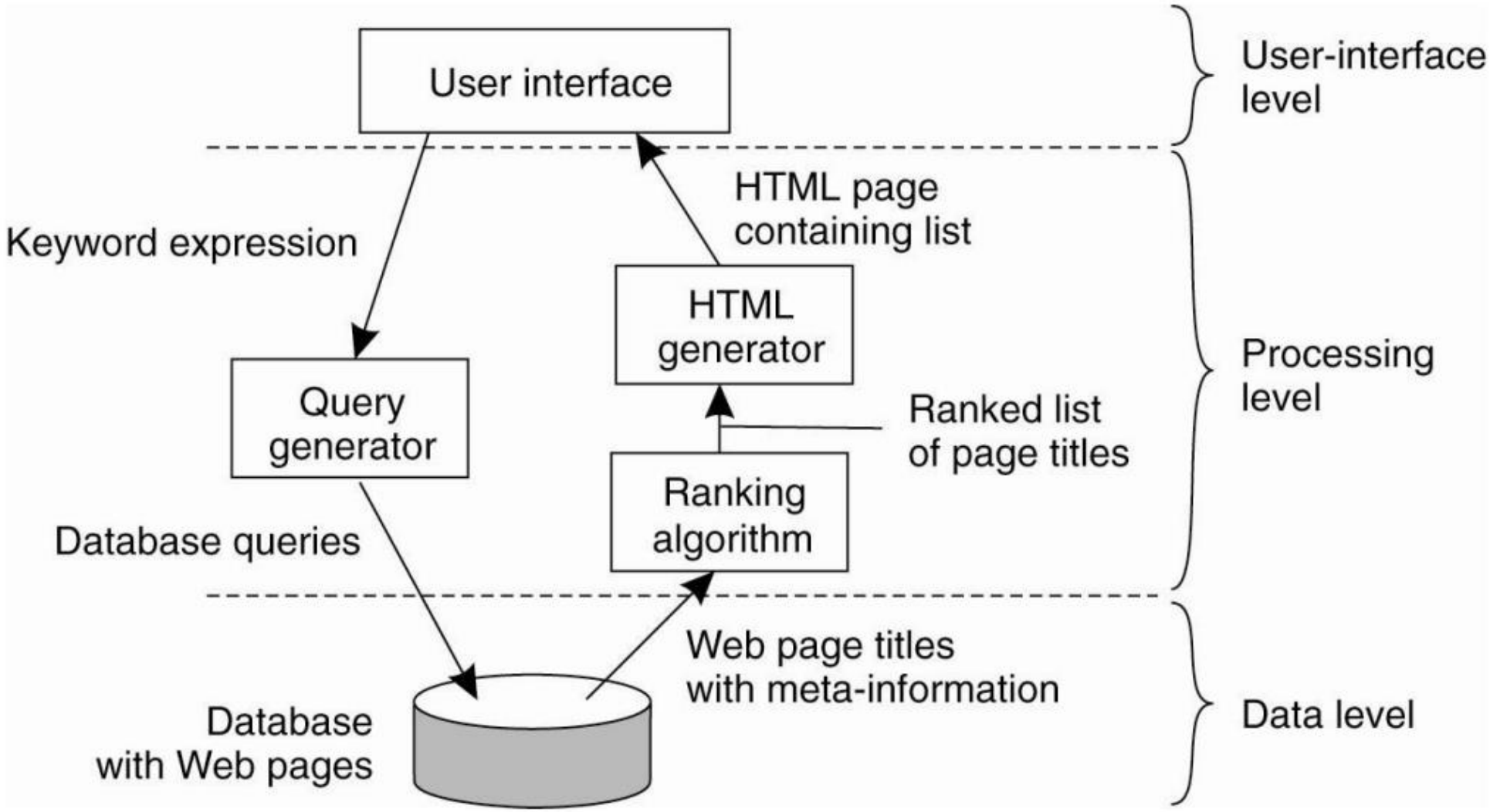
Application Layering

- Traditional three-layered view:
  - User-interface layer
    - Contains units for an application's user interface
  - Processing layer
    - Contains the functions of an application, i.e. without specific data
  - Data layer
    - Contains the data that a client wants to manipulate through the application components

- Observation:
  - This layering is found in many distributed information systems, using traditional database technology and accompanying applications.

Internet Search Engine

- The core : information retrieval part



More Examples

- A Stock Brokerage System
  - User Interface
  - Process Level
    - Analysis of financial data requires sophisticated methods and techniques from statistics and artificial intelligence
  - Data Level
    - Financial database

- Difference between connectionless and connection oriented Protocol. Also focus on their comparative advantages and disadvantages.
- What is the main importance of separating the data level and application layer in Multi-tier Client Server Model. Illustrate.

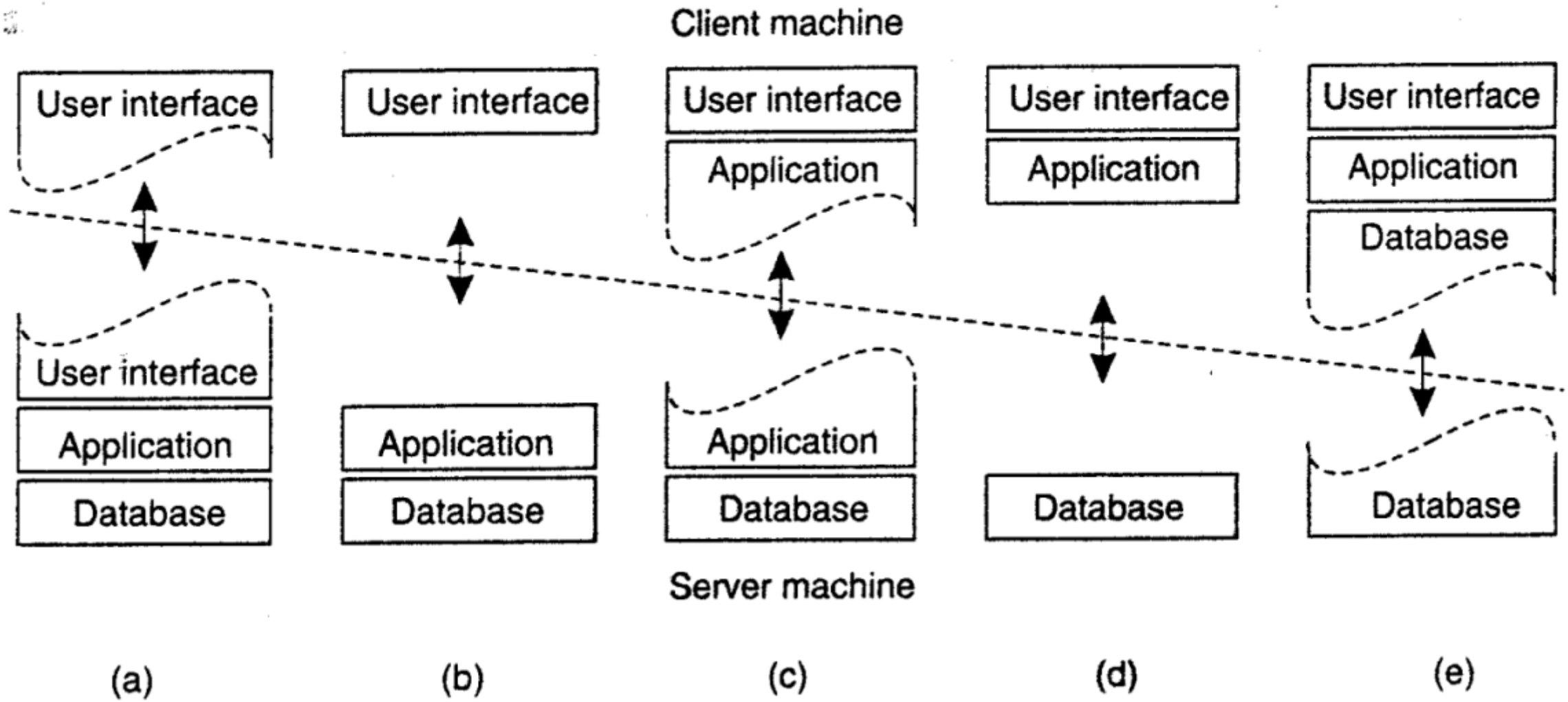
Multi-Tiered Architecture

- The distinction into three logical levels as discussed so far, suggests a number of possibilities for physically distributing a client-server application across several machines.
- The simplest organization is to have only two types of machines:

  1. A client machine containing only the programs implementing (part of) the user-interface level
  2. A server machine containing the rest, that is the programs implementing the processing and data level

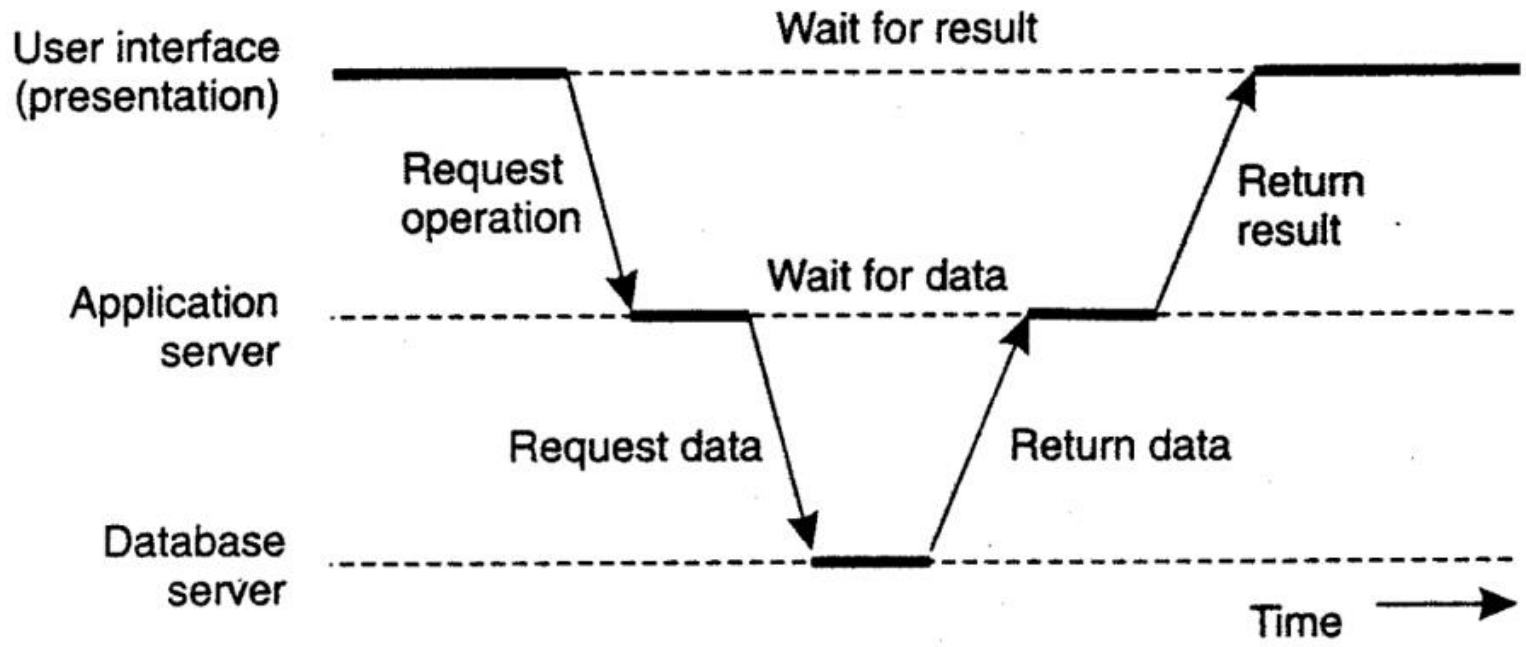
- In this organization everything is handled by the server while the client is essentially no more than a dumb terminal, possibly with a pretty graphical interface.

Alternative Client Server Organization

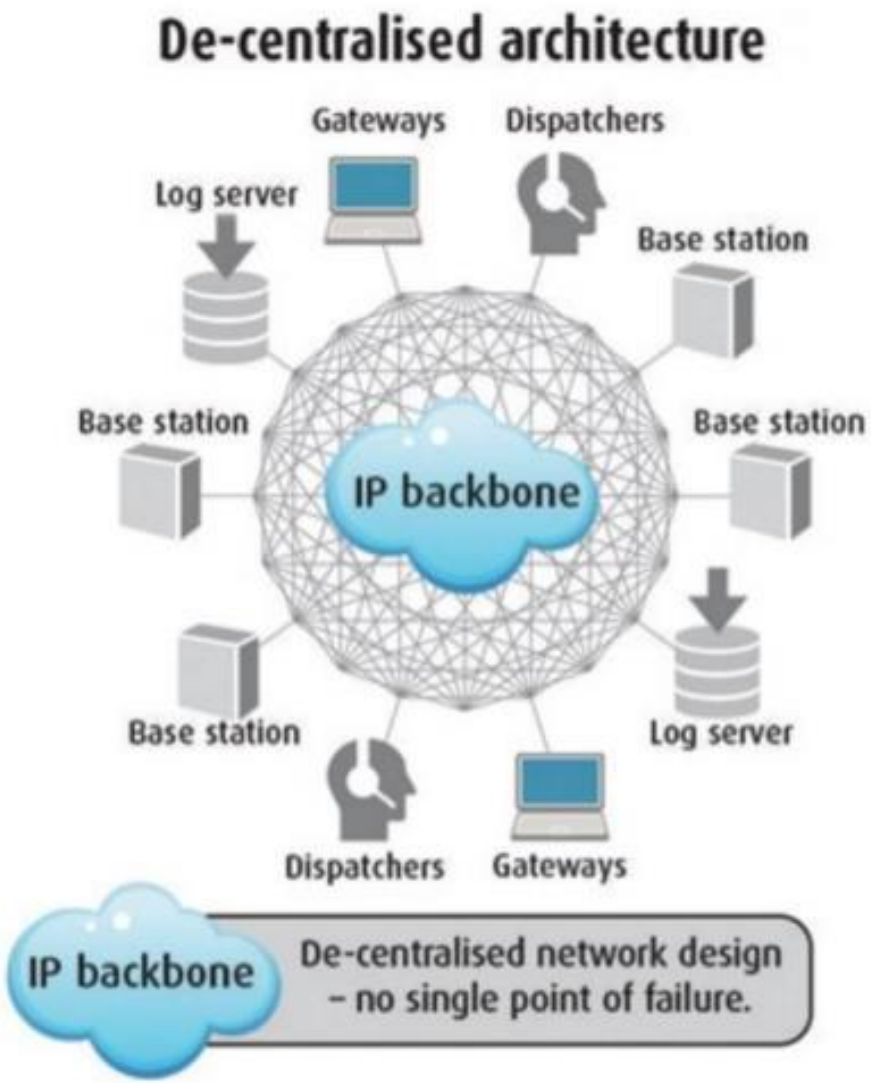


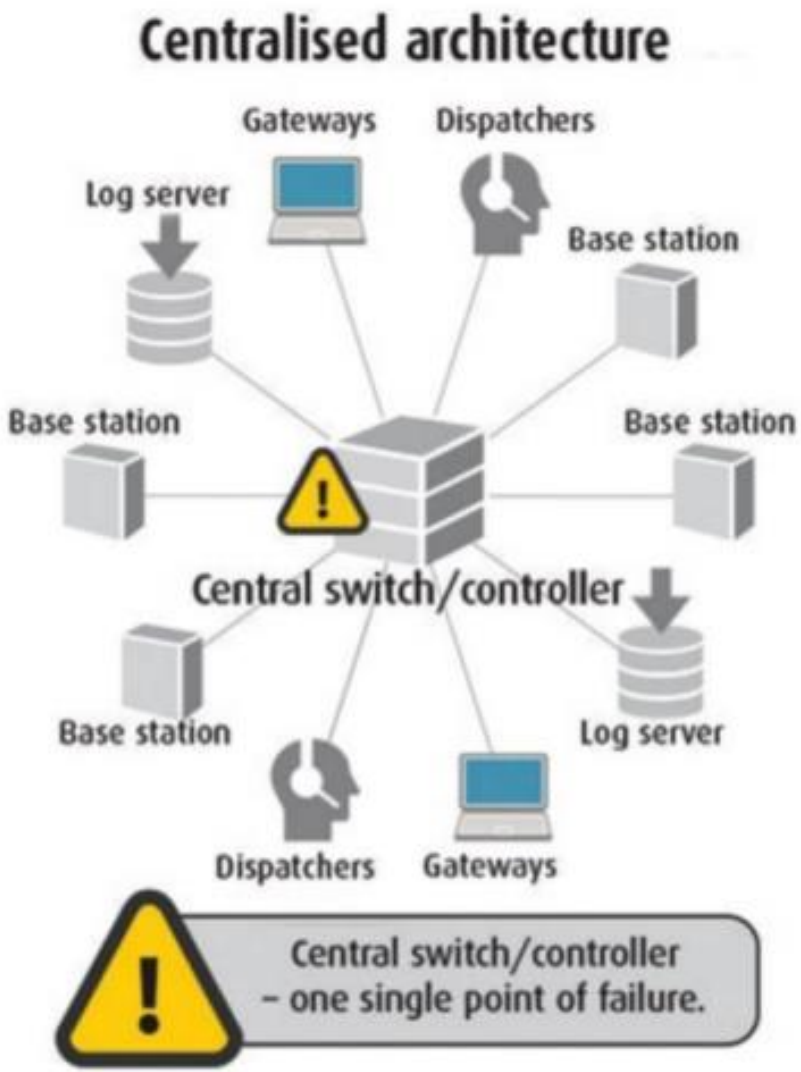
An example of a server acting as client

- 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.
- Eg. Transaction Processing : a separate process, called the transaction processing monitor, coordinates all transactions across possibly different data servers



Decentralized Architectures





- Multitiered client-server architectures → Vertical Distribution
- Vertical distribution is only one way of organizing client-server applications.
- Modern architectures →Horizontal Distribution → AKA peer-to-peer systems

P2P System

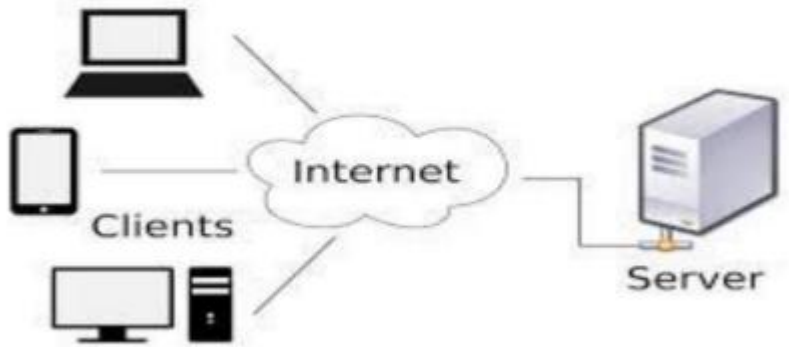
- Peer – to – Peer Computing model is based on how we (human) communicate in real world.
- If we need something then we communicate directly to other corresponding peers (may be friends) who may in turn refer us to their corresponding peers for working towards completion of the request.
- Thus there is a direct access between the peers without any third party intervention.

Examples

- Easy file sharing
- Efficient instant messaging
- Smooth voice communication
- Secure search and communication network
- High performance computing

Evolution of P2P from Client-Server Model

- Many of the internet applications are using Client-Server model Example: WWW, email etc.
- In such model there will be a centralized server shared by many clients.
- The clients query the request to server and get services.
- It will be facilitated if the server is available and capable of serving all the requests from distinct clients at a particular moment.

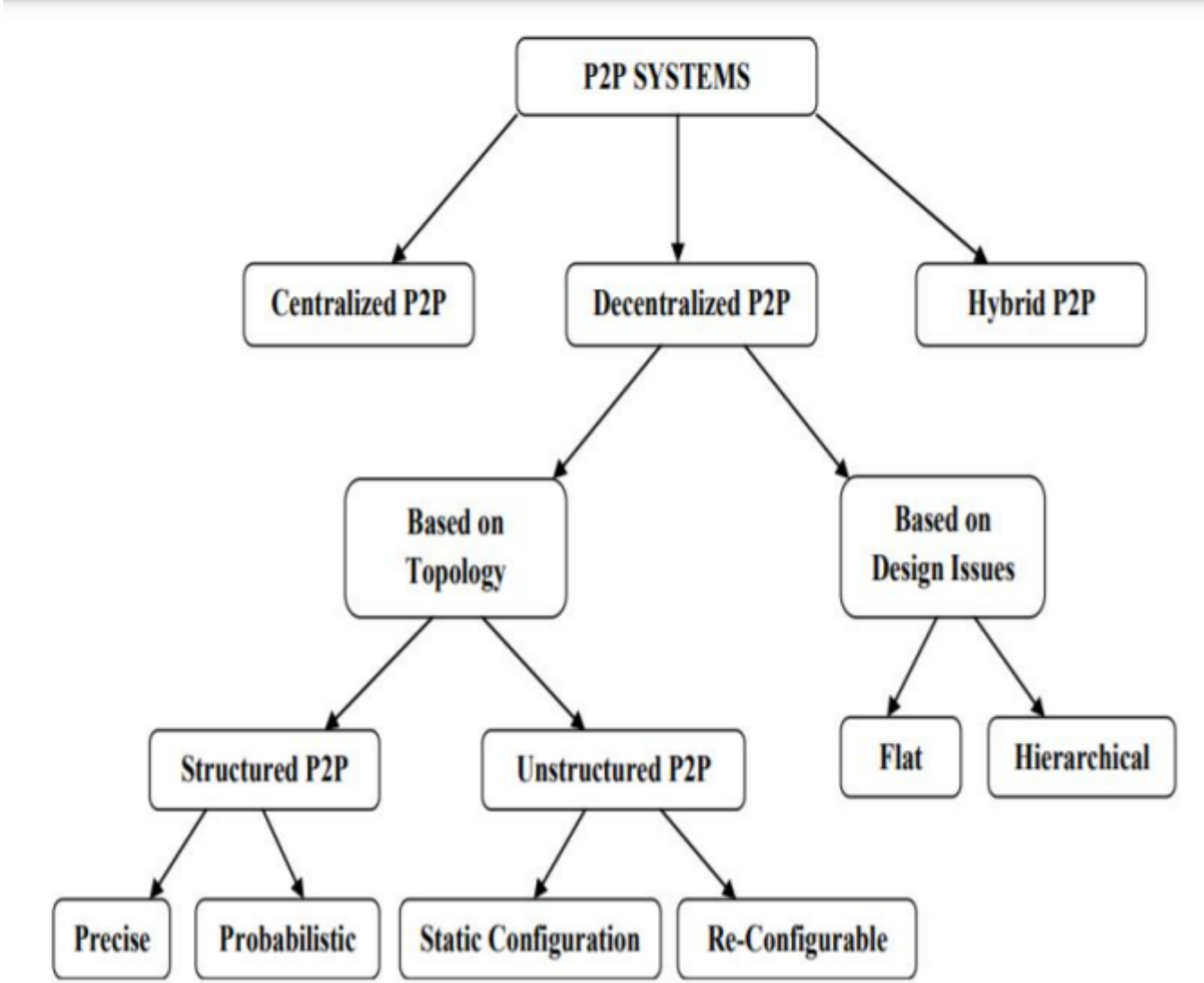


- There may be a chance of getting performance issues because of the unavailability of resources (e.g.: memory, bandwidth, processing speed) at the server system when too many requests arise.

peer-to-peer systems

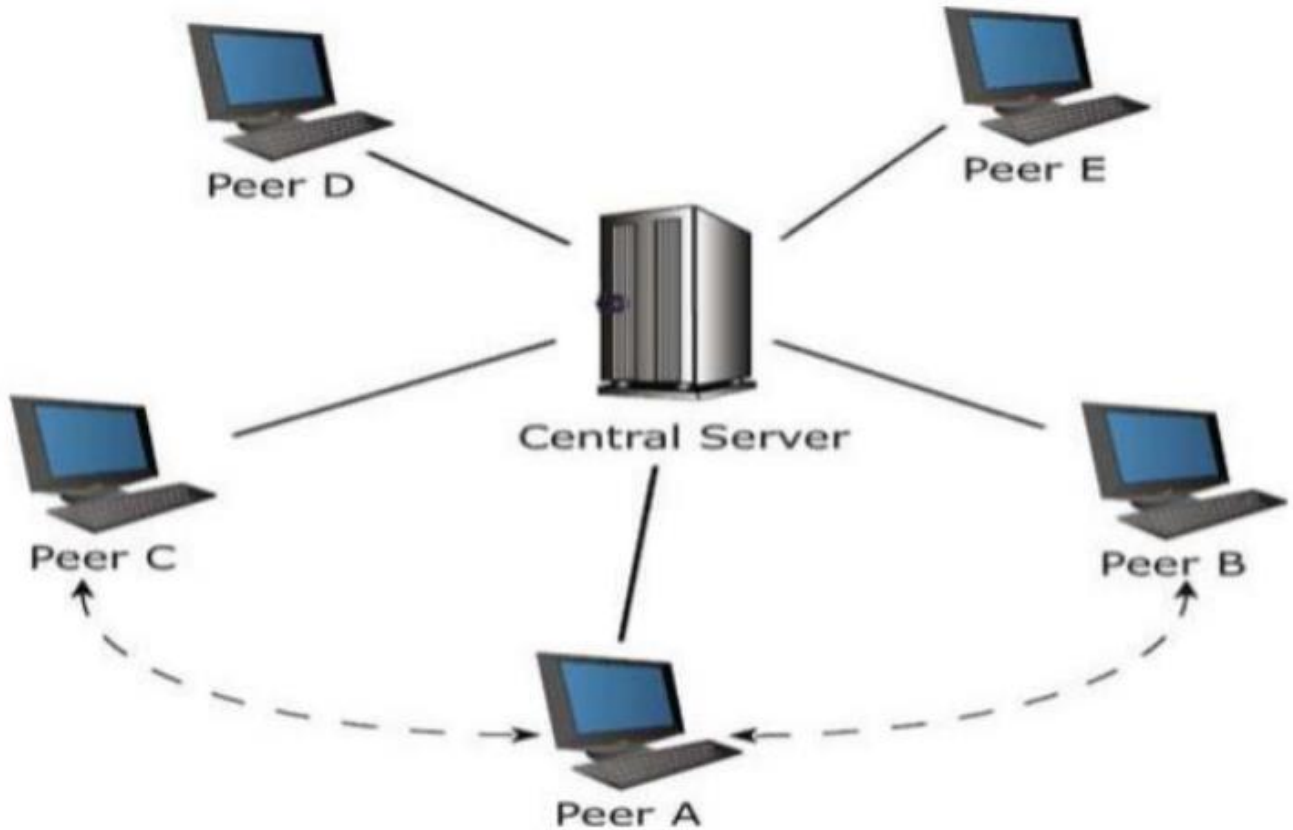
- From a high-level perspective, the processes that constitute a peer-to-peer system are all equal.
- This means that the functions that need to be carried out are represented by every process that constitutes the distributed system.
- As a consequence, much of the interaction between processes is symmetric: each process will act as a client and a server at the same time (which is also referred to as acting as a servent).
- Types:
  - Structured Peer-to-Peer Architectures
  - Unstructured Peer-to-Peer Architectures

Taxonomy of P2P System

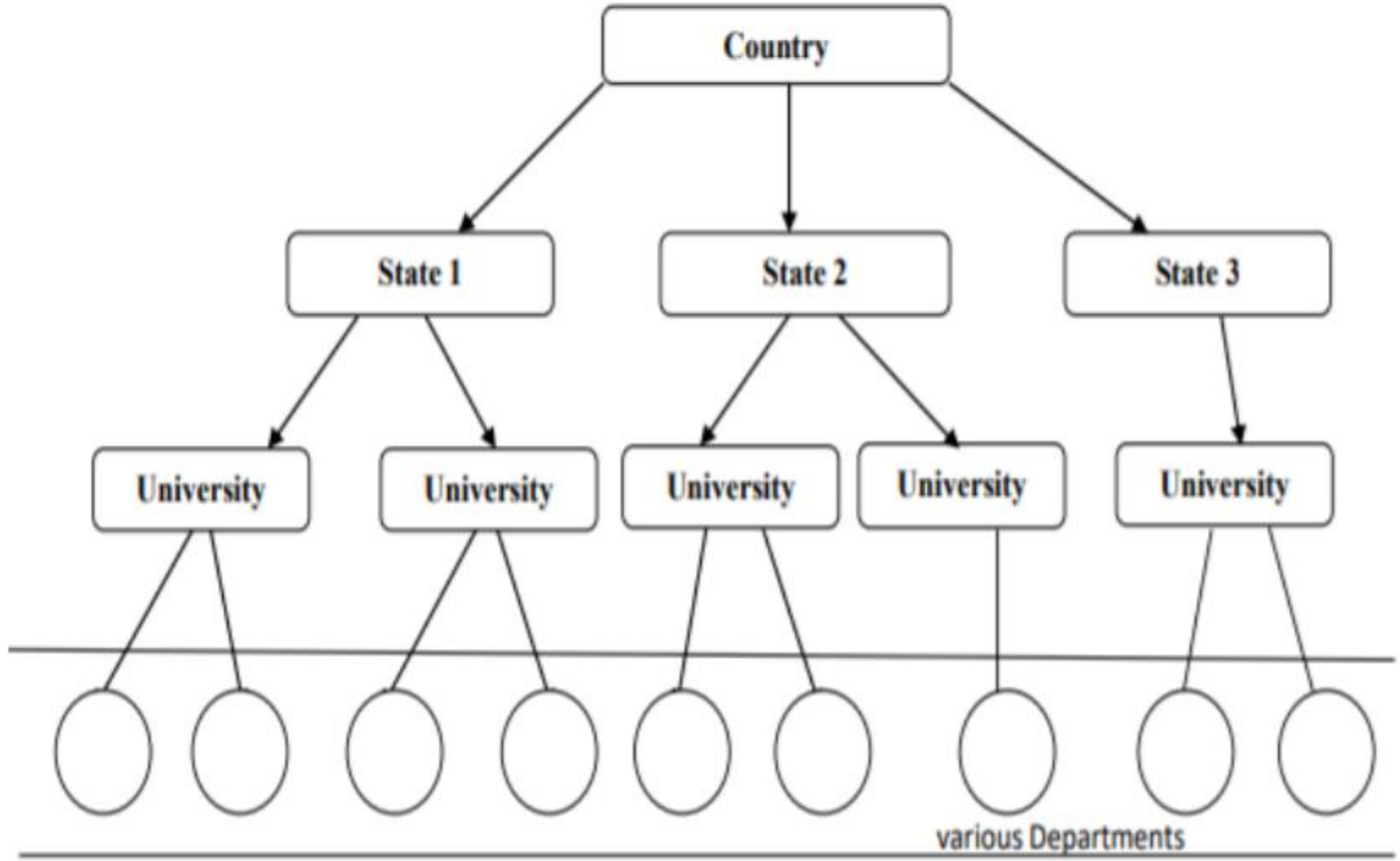


Centralized P2P System

Sharing of MP3 music files



Decentralized P2P System

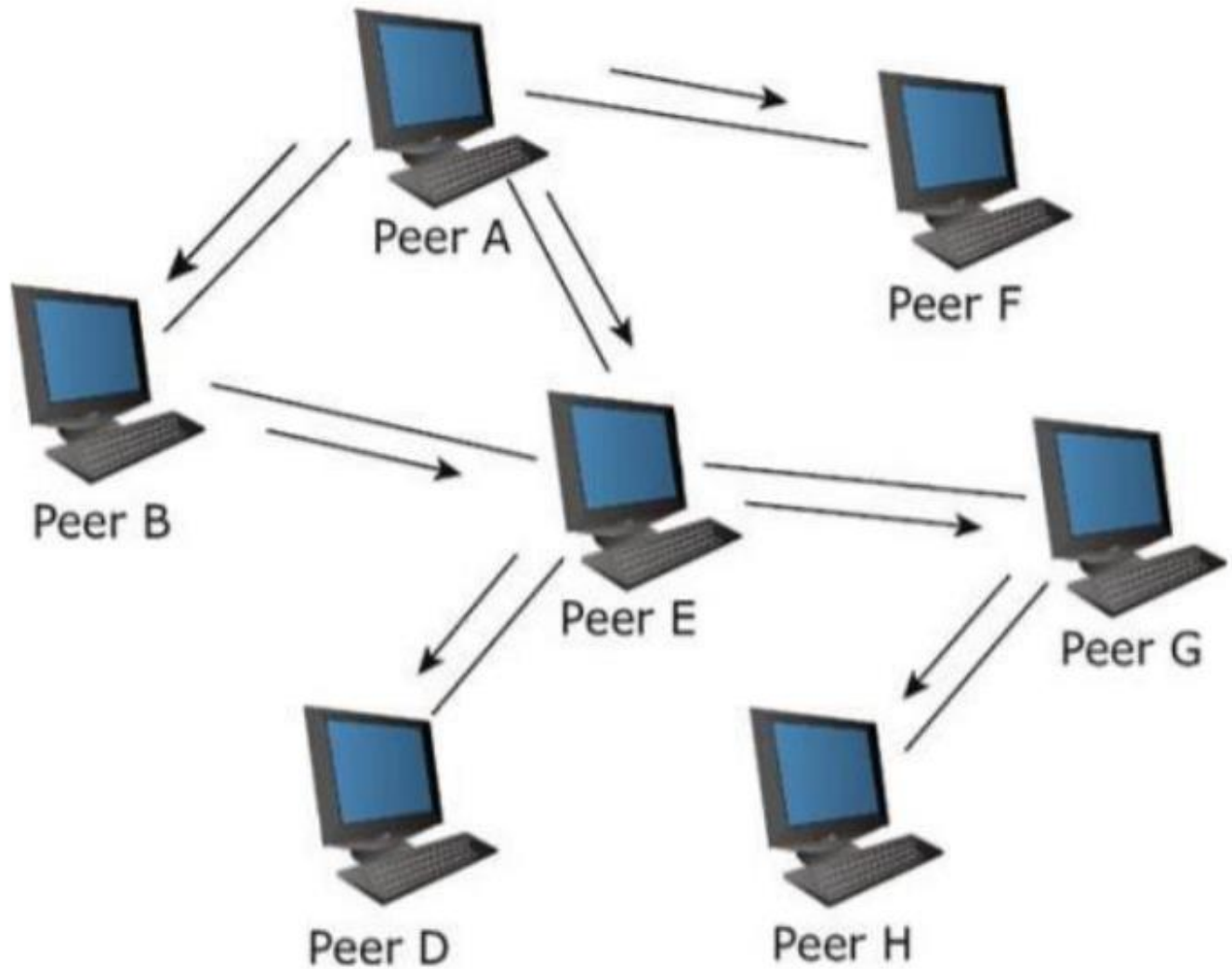


Unstructured P2P Systems

- There is no specific structure in these systems, hence the name "unstructured networks".
- Due to this reason, the scalability of the unstructured p2p systems is very high.
- These systems rely on randomized algorithms for constructing an overlay network.
- As in structured p2p systems, there is no specific path for a certain node. It's generally random, where every unstructured system tried to maintain a random path. Due to this reason, the search of a certain file or node is never guaranteed in unstructured systems.

- The basic principle is that each node is required to randomly select another node, and contact it.
    - Let each peer maintain a partial view of the network, consisting of n other nodes
    - Each node P periodically selects a node Q from its partial view
    - P and Q exchange information and exchange members from their respective partial views

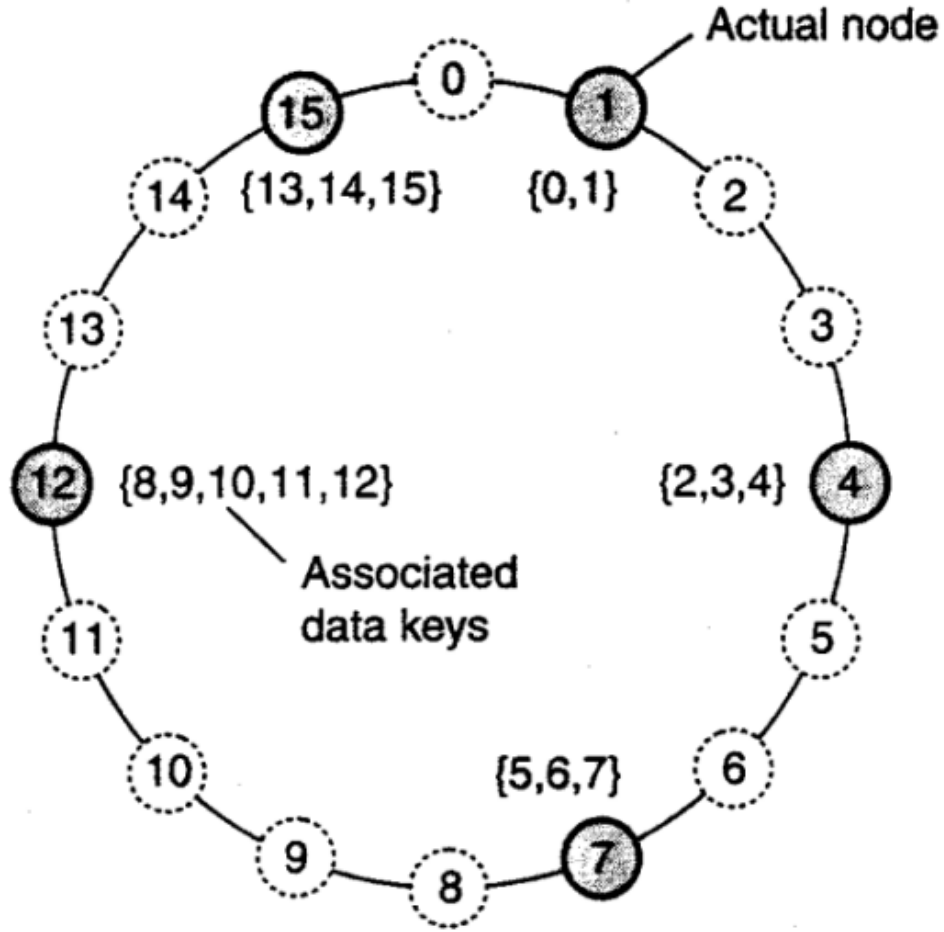
Unstructured P2P System



Structured Peer-to-Peer Architectures

- structured → the system already has a predefined structure that other nodes will follow.
- Every structured network inherently suffers from poor scalability, due to the need for structure maintenance.
- In general, the nodes in a structured overlay network are formed in a logical ring, with nodes being connected to the this ring.
- In this ring, certain nodes are responsible for certain services.

- A common approach that can be used to tackle the coordination between nodes, is to use distributed hash tables (DHTs).
- A traditional hash function converts a unique key into a hash value, that will represent an object in the network.
- The hash function value is used to insert an object in the hash table and to retrieve it.

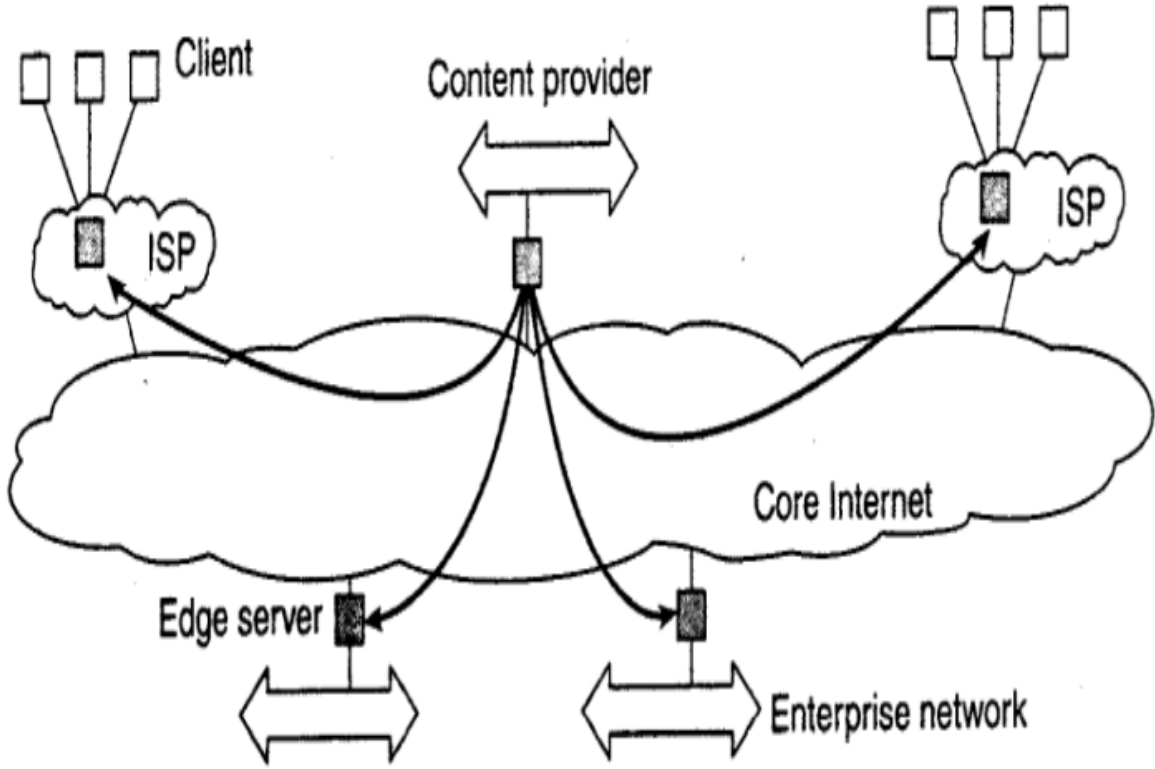


Hybrid Architectures

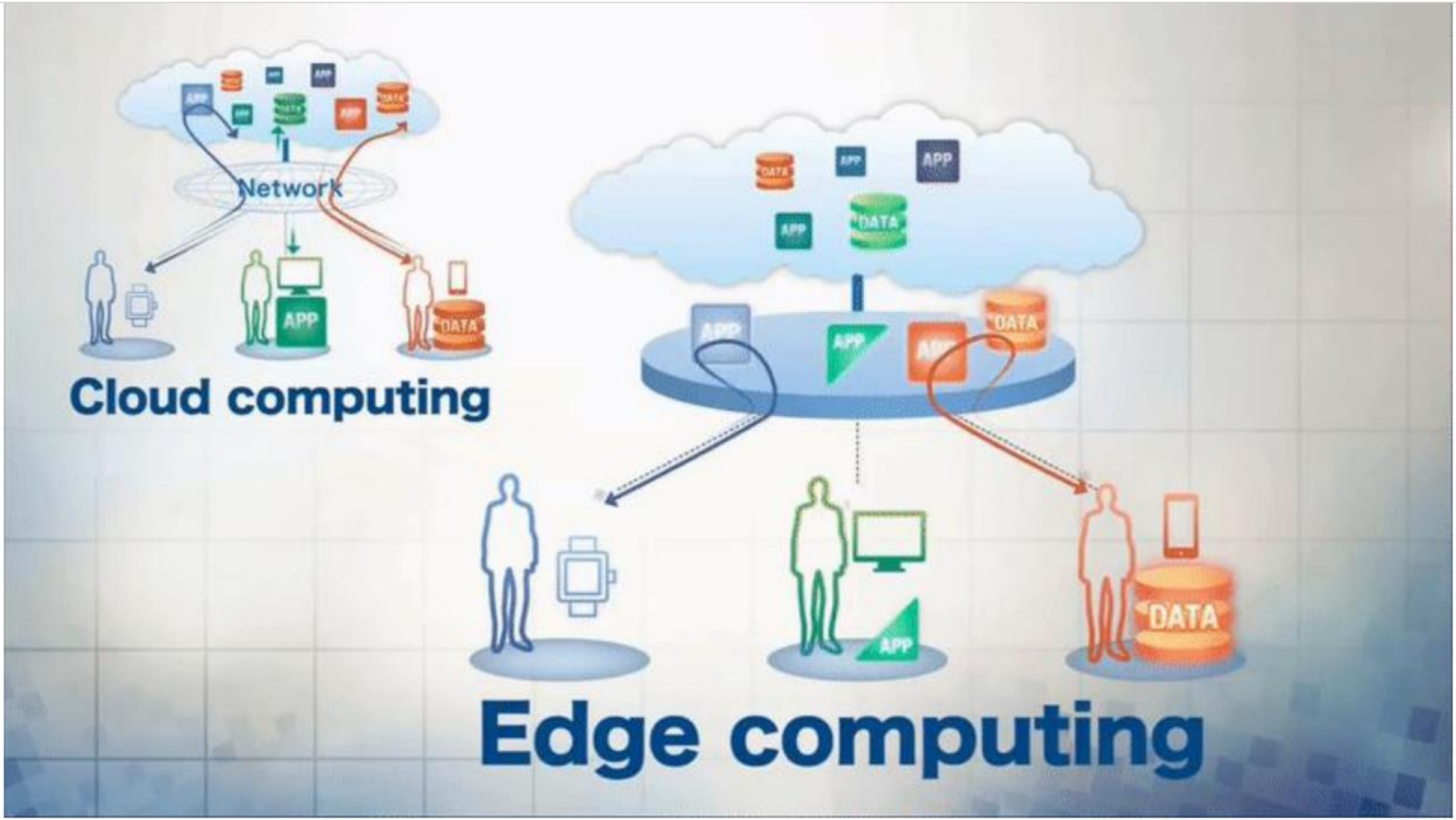
- In this section we take a look at some specific classes of distributed systems in which client-server solutions are combined with decentralized architectures
- Some of these architectures are as follows:
  - Edge-Server Systems
  - Collaborative Distributed Systems

Edge-Server Systems

- An important class of distributed systems that is organized according to a hybrid architecture 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 (ISP).
- EG. Likewise, where end users at home connect to the Internet through their ISP, the ISP can be considered as residing at the edge of the Internet.



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


Network
APP
DATA
Cloud computing
Edge computing

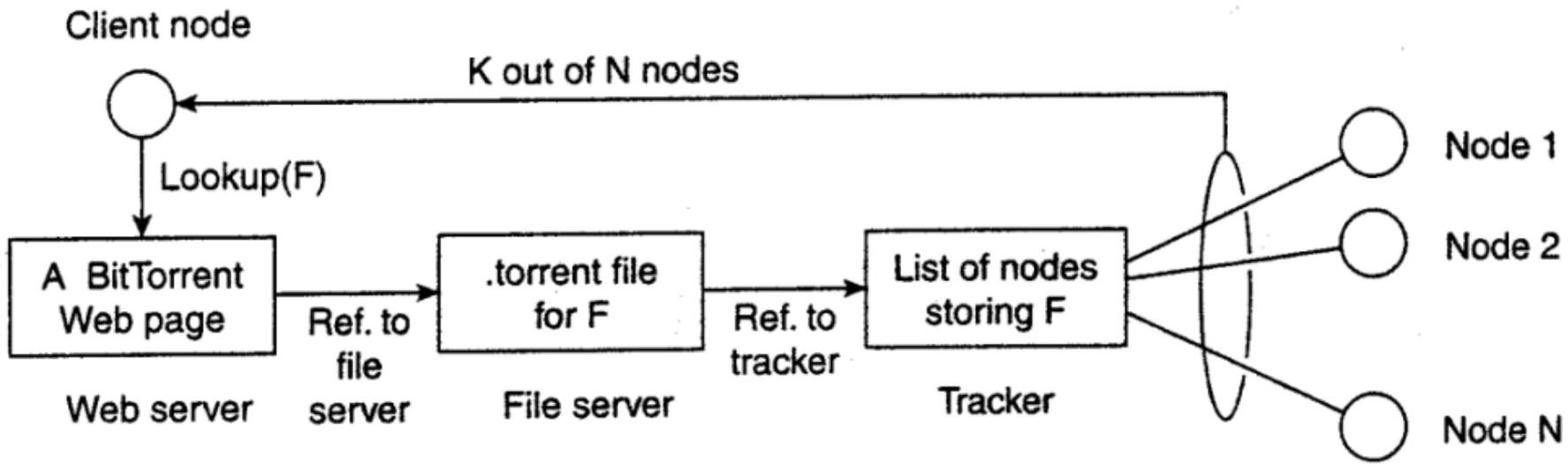
- End users, or clients in general, connect to the Internet by means of an edge server.
- The edge server's main purpose is to serve content, possibly after applying filtering and transcoding functions.
- More interesting is the fact that a collection of edge servers can be used to optimize content and application distribution.
- The basic model is that 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 and such.

Collaborative Distributed Systems

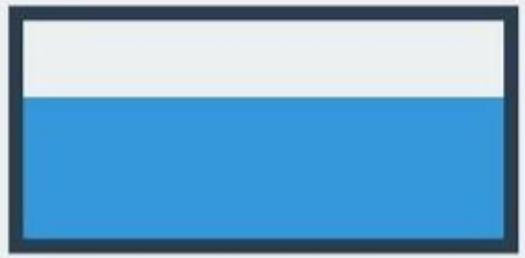
- Hybrid structures are especially deployed in collaborative distributed systems.
- The main issue in many of these systems to first get started, for which often a traditional client-server scheme is deployed.
- Once a node has joined the system, it can use a fully decentralized scheme for collaboration

BitTorrent file-sharing system

- BitTorrent is a peer-to-peer file downloading system.
- The basic idea is that when an end user is looking for a file, he downloads chunks of the file from other users until the downloaded chunks can be assembled together yielding the complete file.



DIRECT DOWNLOAD SERVER



USERS

BITTORRENT PEER-TO-PEER

USERS

1. Explain challenges distributed systems.
2. What are different architecture styles of distributed system? Explain each of them in detail.
3. What are different types of transparencies required in distributed systems? Explain.
4. What is the role of middleware in distributed systems? Explain.
5. Define distributed systems. Explain different types of distributed system in detail.
6. What are the characteristics of distributed system?
7. Explain the foundations of communication in distributed systems. Discuss different communication models and protocols used in distributed systems.
8. Discuss how distributed systems are more scalable than centralized systems.
9. What is the role of middleware in distributed systems? Explain.
10. Explain client-server and peer-to-peer distributed system in details.