Parallel & Distributed System

Distributed Systems Architecture - 03

Md. Biplob Hosen

Lecturer, IIT-JU

Email: biplob.hosen@juniv.edu

Contents

- Decentralized Organizations
  - ✓ Peer-to-peer Systems
- Hybrid Architectures
  - ✓ Edge-server Systems
- Example Architectures
  - ✓ The Network File System
  - ✓ The Web

- Reference Books
  - Distributed Systems: Principles and Paradigms, 3rd Edition by Andrew S. Tanenbaum & Maarten van Steen, Publisher: Pearson Prentice Hall **[CH-02]**.

Decentralized Organizations

- Multitiered client-server architectures are a direct consequence of dividing distributed applications into a user interface, processing components, and data-management components.
- In many business environments, distributed processing is equivalent to organizing a client-server application as a multitiered architecture which is referred to **vertical distribution**.
- The characteristic feature of vertical distribution is that it is achieved by placing logically different components on different machines.
- The term is related to the concept of vertical fragmentation as used in distributed relational databases, where it means that tables are split column-wise, and subsequently distributed across multiple machines.
- Again, from a systems-management perspective, having a vertical distribution can help: functions are logically and physically split across multiple machines, where each machine is tailored to a specific group of functions.

Decentralized Organizations: peer-to-peer Systems

- Vertical distribution is only one way of organizing client-server applications.
- In modern architectures, it is often the distribution of the clients and the servers that counts, which we refer to as **horizontal distribution**.
- In this type of 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.
- For example, **peer-to-peer systems** is a class of modern system architectures that support horizontal distribution.
- From a high-level perspective, the processes that constitute a peer-to-peer system are all equal, which 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 **servant**).

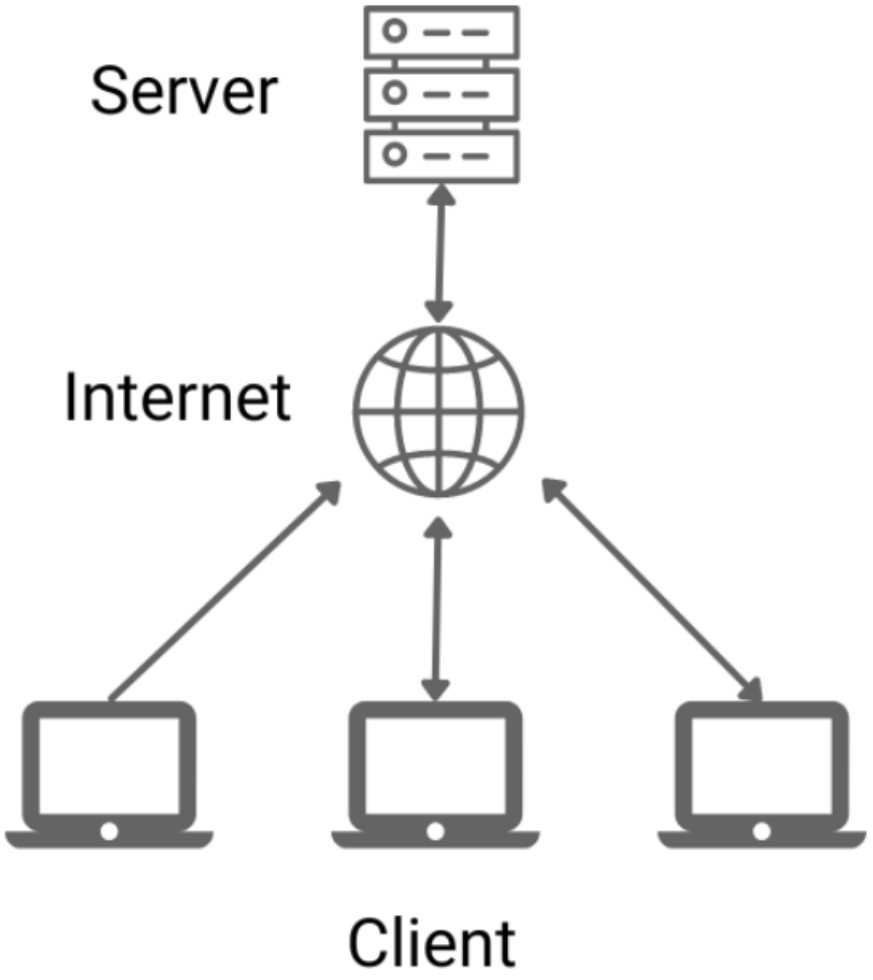
P2P Architecture

Peer-to-peer (P2P) architecture is a distributed system in which each node acts as both a client and a server. This allows the nodes to share the workload or tasks among themselves. In a P2P system:

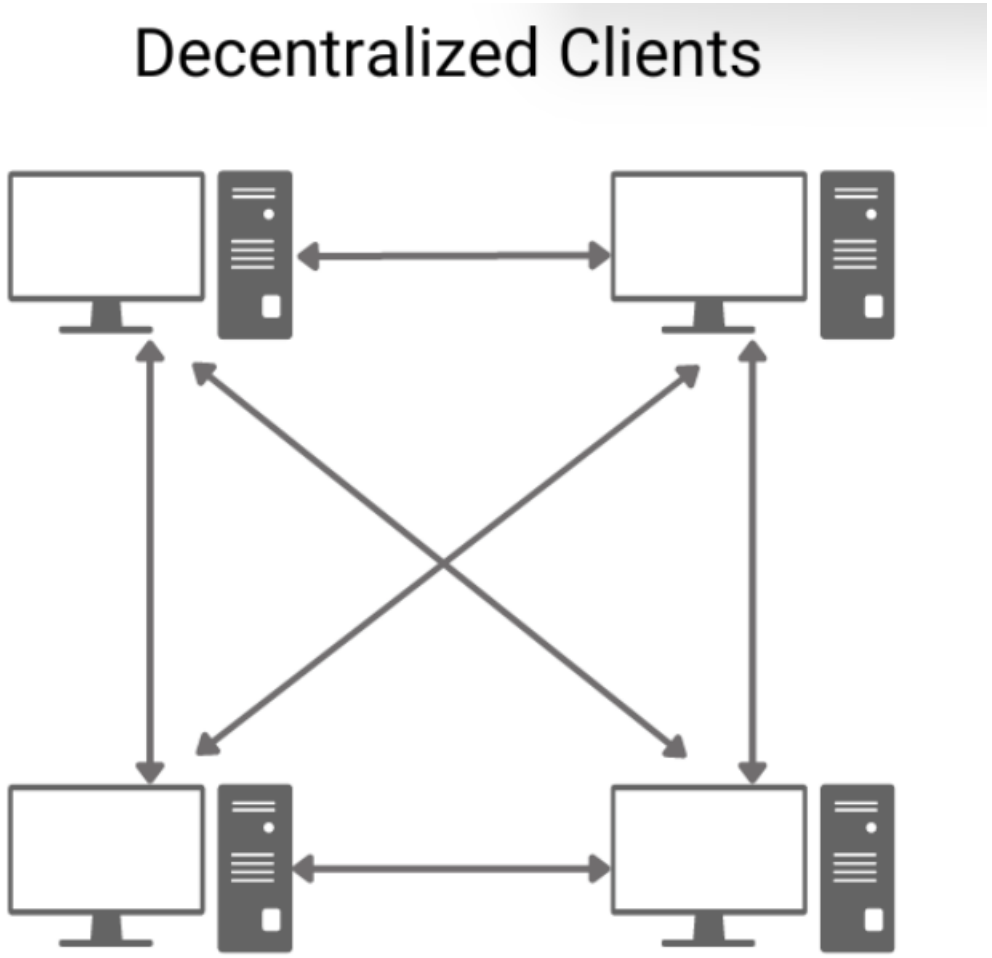
- Each node, or peer, has the same functional capability and holds part of the resources in the system. Some peers may have more resources and may be able to contribute more to the network, while others may not contribute any resources and only consume from the network.
- Peers are visible to each other and can connect with as many other peers as they wish. They are free to join or leave the network at any time and can transmit and receive data from other peers simultaneously.
- P2P systems are often used in applications where decentralized control is desired, such as file sharing or distributed computing.

In a P2P network, tasks are distributed among all the peers rather than being handled by a single server, which makes P2P networks more resilient as there is no single point of failure. In some cases, a central tracking server may be used to help the peers find each other and manage the network, but it does not have control over the network itself. The decentralized nature of P2P networks allows for more flexibility and autonomy compared to traditional centralized systems.

P2P Architecture



Client Server Architecture



Peer to Peer Architecture

P2P Architecture

How does P2P network work?

Peer-to-peer (P2P) networks do not have a dedicated server for user authentication. This means that each computer is responsible for its own protection and may require a separate user account for each machine that a user wants to access. So users save files on their own machines and are responsible for backing them up properly.

Security is a major concern in P2P networks, as access to data and resources is managed on each individual machine. Each computer can allow or deny access to other computers, so users must be careful about what they share. For example, a user might share a folder containing sensitive payroll information, but only allow other users to access the files within that folder. Because there is no central administrator in a P2P network, users must manage access to their own files and resources. Therefore, P2P networks are often used in small deployments or situations where security is not a major concern, such as home networks or small businesses.

P2P Architecture

Applications of peer-to-peer architecture

P2P architecture is most effective when there are many active peers in the network. This allows new peers to easily find other peers to connect to and ensures that there are enough remaining peers to take up the slack if many peers leave the network. When there are only a few peers in the network, there are fewer resources available overall.

In a P2P file-sharing program, the more common a file is, the faster it can be downloaded because many peers are sharing it. To make P2P work more efficiently, the workload is often divided into small pieces that can be reassembled later. This allows many peers to work on the same task simultaneously and reduces the amount of work that each peer has to do.

P2P Architecture

Some use cases of P2P architecture

There are several uses for peer-to-peer (P2P) architecture, including:

- **File sharing:** P2P networks are often used for file sharing because they allow users to share files directly with each other without the need for a central server.
- **Instant messaging**: P2P networks can be used to enable instant messaging between users, allowing them to communicate in real-time.
- **Voice communication**: P2P networks can also be used for voice communication, such as through VoIP (Voice over Internet Protocol) technology.
- **Collaboration**: P2P networks can enable collaboration among users by allowing them to share and access resources and data directly with each other.
- **High performance**: P2P networks can offer high performance and scalability due to their decentralized nature and the ability to share resources among multiple nodes.

P2P Architecture

Some examples of P2P architecture

- **Napster:** Napster was a popular file-sharing service that was shut down in 2001 due to copyright issues. It used a central tracking server to facilitate file sharing among users.
- **BitTorrent:** BitTorrent is a popular P2P file-sharing protocol that allows users to share large files directly with each other without the need for a central server.
- **Skype:** Skype originally used a proprietary hybrid P2P protocol for voice and video communication, but after being acquired by Microsoft, it now uses a client-server model.
- **Bitcoin:** Bitcoin is a P2P cryptocurrency that operates without a central monetary authority. Transactions are verified and recorded on a decentralized network of computers, rather than being centrally managed.

P2P Architecture

Types of peer-to-peer network

There are three different types of P2P networks.

- Unstructured P2P networks
- Structured P2P networks
- Hybrid P2P networks

P2P Architecture

Unstructured P2P networks

In unstructured peer-to-peer (P2P) networks, the nodes are not arranged in any particular order, meaning that node-to-node communication is random. This makes unstructured P2P networks well-suited for high-activity use cases, such as social platforms where users may regularly join or leave the network.

However, unstructured P2P networks have a disadvantage in that they require a significant amount of CPU and memory to function properly. The hardware must be able to support the maximum number of network transactions, ensuring that all nodes can communicate with each other at any given time. This can be a challenge, especially if the network is large or has a high volume of activity.

P2P Architecture

Structured P2P networks

Structured peer-to-peer (P2P) networks are the opposite of unstructured P2P networks in that the nodes have a way to interact with each other in a more organized manner. This is achieved through a well-organized architecture that allows users to find and use files more efficiently rather than searching randomly. Hash functions are often used for database lookups in structured P2P networks.

While structured P2P networks are generally more effective, they have some centralization due to their organized architecture. This means that they may be more expensive to maintain and set up than unstructured P2P networks. However, structured P2P networks are more stable than unstructured P2P networks.

Hybrid P2P networks

Hybrid peer-to-peer (P2P) networks combine the peer-to-peer architecture with the client-server model. This allows for a central server with P2P capabilities, which can be beneficial for certain types of networks.

Hybrid P2P networks offer several benefits over structured and unstructured P2P networks, including a more strategic approach, increased performance, and other advantages. Overall, hybrid P2P networks can be a good choice for networks that need the benefits of both P2P and client-server architectures.

P2P Architecture

Advantages of peer-to-peer networks

Peer-to-peer (P2P) networks offer several advantages. One of the main benefits of P2P networks is that they can be more cost-effective due to the absence of a central server that needs to be maintained and paid for (apart from monitoring servers). This also means there is no need for a network operating system, further reducing costs.

Another advantage of P2P networks is that they are resistant to single points of failure. Unless the network is very small, it is unlikely that the failure of a single peer will have a significant impact on the overall network. In addition, changes in the number of peers in the network do not affect its stability.

P2P networks can also handle large numbers of simultaneous connections without difficulty due to their decentralized structure. This also makes them more resistant to attacks since no central server could be targeted.

P2P Architecture

To summarise, some of the benefits of peer-to-peer (P2P) networks include:

1. **No single point of failure:** If the server hosting your content goes down, all of your data may be lost. However, in a P2P network, this is not a concern because the data is distributed among multiple peers.
2. **Scalable bandwidth and storage:** P2P networks are decentralized and distributed, which means that as more people interact with your data, more nodes (i.e., computers or devices) are added to the network. This allows for unlimited and automatic scaling of bandwidth and storage.
3. **No third-party intervention:** In a P2P network, data is secure because no third party is involved. You can share data only with the friends you choose to share with.
4. **An open, neutral platform:** In a P2P network, all nodes have equal privileges, ensuring minimum corporate control or intervention. This means that you have control over your data.

P2P Architecture

Disadvantages of Peer to Peer Networks

Peer-to-peer (P2P) networks have some security risks that should be considered. One potential issue is that if a peer becomes infected with a virus and uploads a file containing the virus, the virus can easily spread to other peers in the network. Another concern is that it can be difficult to ensure that all peers have the correct permissions to access the network, particularly if there are a large number of peers.

Some users of P2P networks take advantage of resources shared by other nodes without contributing anything themselves. These users are known as "leachers." While the decentralized nature of P2P networks can make them difficult to shut down, this can also be a drawback if the network is used for illegal or unethical activities.

In addition, the widespread use of mobile devices has led many businesses to adopt different types of architecture. However, using mobile devices can present challenges for P2P networks, as it may be difficult for users to contribute to the network without draining their battery life or using up their mobile data.

Some other drawbacks of the P2P network are:

- P2P networks can make it easy to distribute copyrighted content without control.
- If all the nodes in the P2P network are offline, the data becomes unavailable.

Hybrid Architectures

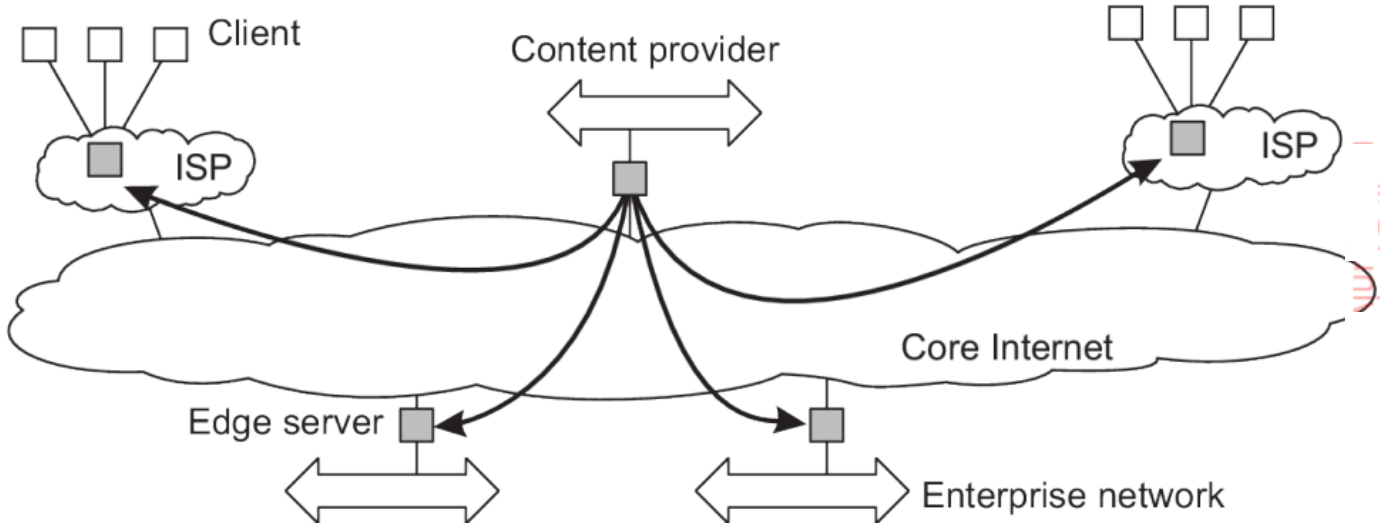
- Many distributed systems combine architectural features, in which client-server solutions are combined with decentralized architectures.
- Examples:
  - Edge-server Systems
  - Collaborative Distributed Systems

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, for example, as provided by an Internet Service Provider (ISP).
- 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.
- 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.

Edge-server Systems

- 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.



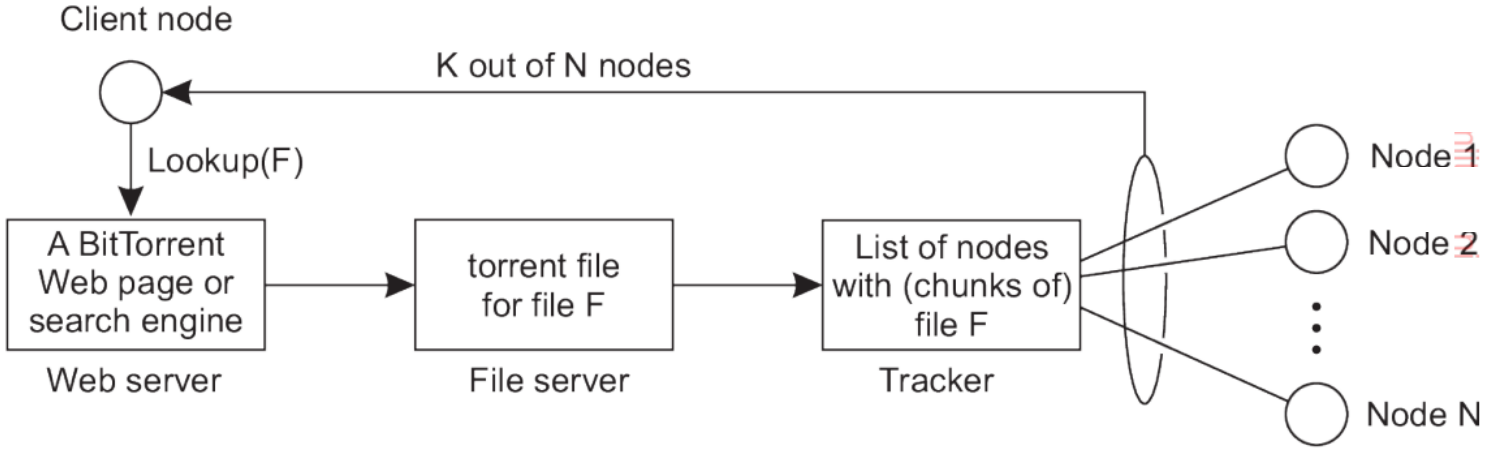
Viewing the Internet as consisting of a collection of edge servers

- This concept of edge-server systems is now often taken a step further: taking cloud computing as implemented in a data center as the core, additional servers at the edge of the network are used to assist in computations and storage, essentially leading to distributed cloud systems.
- In the case of fog computing, even end-user devices form part of the system and are controlled by a cloud-service provider.

Collaborative Distributed Systems

- Hybrid structures are notably deployed in 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.
- To make matters concrete, let us consider the widely popular BitTorrent file-sharing system.
- BitTorrent is a peer-to-peer file downloading system which principal working is shown in the following figure.
- 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.
- An important design goal was to ensure collaboration.
- In most file-sharing systems, a significant fraction of participants merely download files but otherwise contribute close to nothing, a phenomenon referred to as free riding.
- To prevent this situation, in BitTorrent a file can be downloaded only when the downloading client is providing content to someone else.

Collaborative Distributed Systems



The principal working of BitTorrent

- To download a file, a user needs to access a global directory, which is generally just one of a few well-known Web sites.
- Such a directory contains references to what are called torrent files.
- A torrent file contains the information that is needed to download a specific file.
- In particular, it contains a link to what is known as a tracker, which is a server 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 the file as well.
- Obviously, there will be many different trackers, although there will generally be only a single tracker per file (or collection of files).

Collaborative Distributed Systems

- 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, for example by providing chunks of the file it is downloading that others do not yet have.
- 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.
- This scheme works well provided P has something to download from Q. For this reason, nodes are often supplied with references to many other nodes putting them in a better position to trade data.
- Clearly, BitTorrent combines centralized with decentralized solutions. As it turns out, the bottleneck of the system is, not surprisingly, formed by the trackers.

Collaborative Distributed Systems

- In an alternative implementation of BitTorrent, a node also joins a separate structured peer-to-peer system (i.e., a DHT) to assist in tracking file downloads.
- In effect, a central tracker's load is now distributed across the participating nodes, with each node acting as a tracker for a relatively small set of torrent files.
- The original function of the tracker coordinating the collaborative downloading of a file is retained.
- However, we note that in many BitTorrent systems used today, the tracking functionality has actually been minimized to a one-time provisioning of peers currently involved in downloading the file.
- From that moment on, the newly participating peer will communicate only with those peers and no longer with the initial tracker.
- The initial tracker for the requested file is looked up in the DHT through a so-called magnet link.

The Network File System

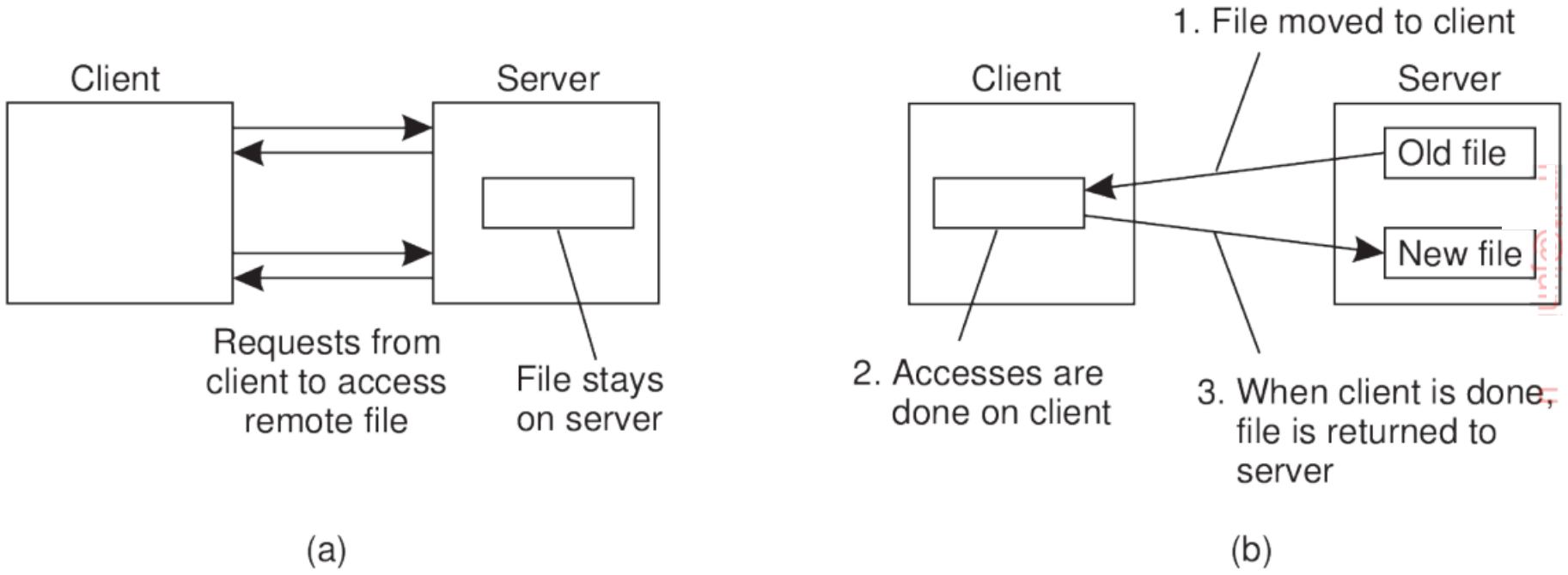
- Many distributed files systems are organized along the lines of client-server architectures, with Sun Microsystem's Network File System (NFS) being one of the most widely-deployed ones for Unix systems.
- The basic idea behind NFS is that each file server provides a standardized view of its local file system.
- In other words, it should not matter how that local file system is implemented; each NFS server supports the same model.
- This approach has been adopted for other distributed files systems as well.
- NFS comes with a communication protocol that allows clients to access the files stored on a server, thus allowing a heterogeneous collection of processes, possibly running on different operating systems and machines, to share a common file system.

The Network File System

- The model underlying NFS and similar systems is that of a remote file service.
- In this model, clients are offered transparent access to a file system that is managed by a remote server.
- However, clients are normally unaware of the actual location of files.
- Instead, they are offered an interface to a file system that is similar to the interface offered by a conventional local file system.
- In particular, the client is offered only an interface containing various file operations, but the server is responsible for implementing those operations.
- This model is therefore also referred to as the remote access model.

The Network File System

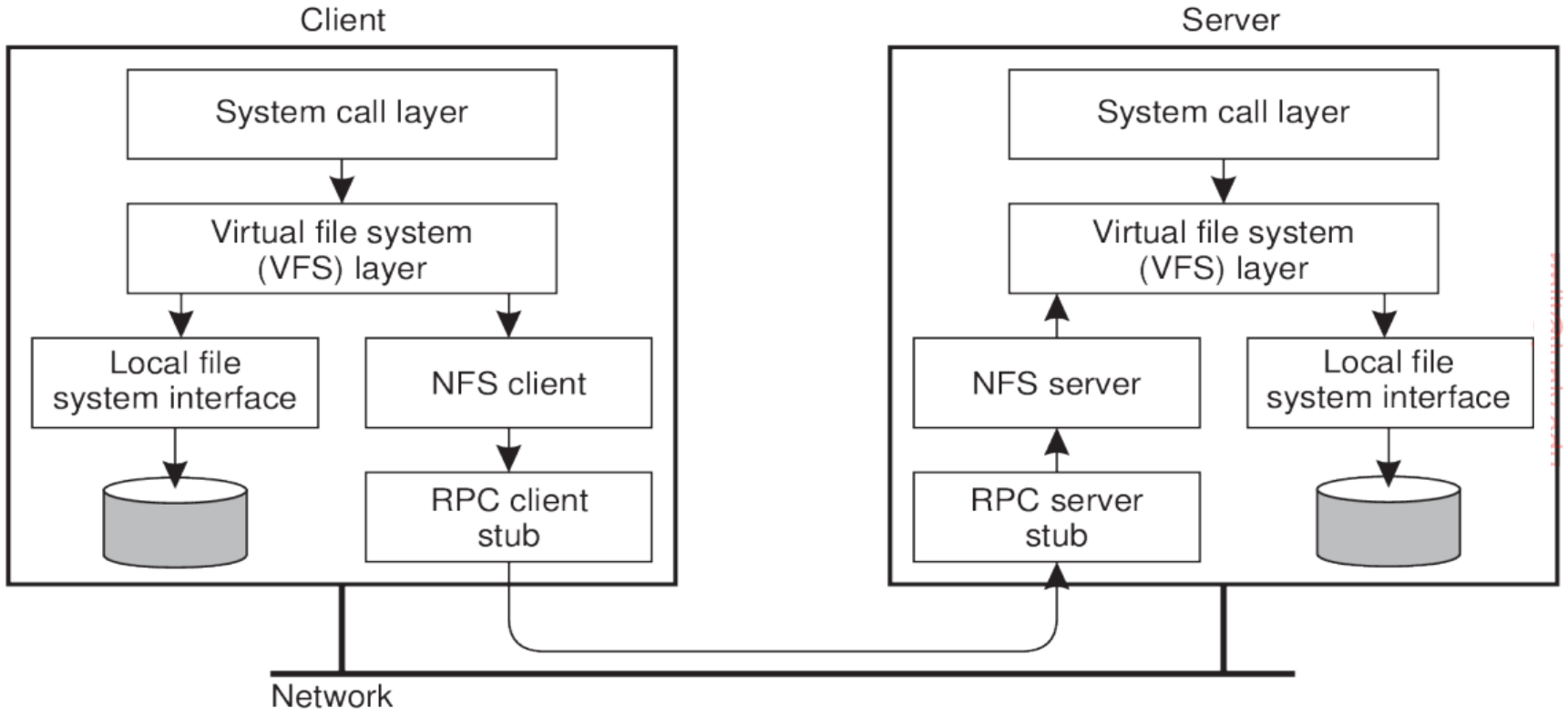
- In contrast, in the upload/download model a client accesses a file locally after having downloaded it from the server.
- When the client is finished with the file, it is uploaded back to the server again so that it can be used by another client.



(a) The remote access model. (b) The upload/download model

The Network File System

- The Internet's FTP service can be used this way when a client downloads a complete file, modifies it, and then puts it back.



The basic NFS architecture for Unix systems.

- NFS has been implemented for a large number of different operating systems, although the Unix versions are predominant.
- For virtually all modern Unix systems, NFS is generally implemented following the layered architecture shown in the figure.

The Network File System

- A client accesses the file system using the system calls provided by its local operating system.
- However, the local Unix file system interface is replaced by an interface to the Virtual File System (VFS), which by now is a de facto standard for interfacing to different (distributed) file systems.
- Virtually all modern operating systems provide VFS, and not doing so more or less forces developers to largely re-implement huge parts of an operating system when adopting a new file-system structure.
- With NFS, operations on the VFS interface are either passed to a local file system, or passed to a separate component known as the NFS client, which takes care of handling access to files stored at a remote server.
- In NFS, all client-server communication is done through so-called remote procedure calls (RPCs).
- An RPC is essentially a standardized way to let a client on machine A make an ordinary call to a procedure that is implemented on another machine B.
- The NFS client implements the NFS file system operations as remote procedure calls to the server.

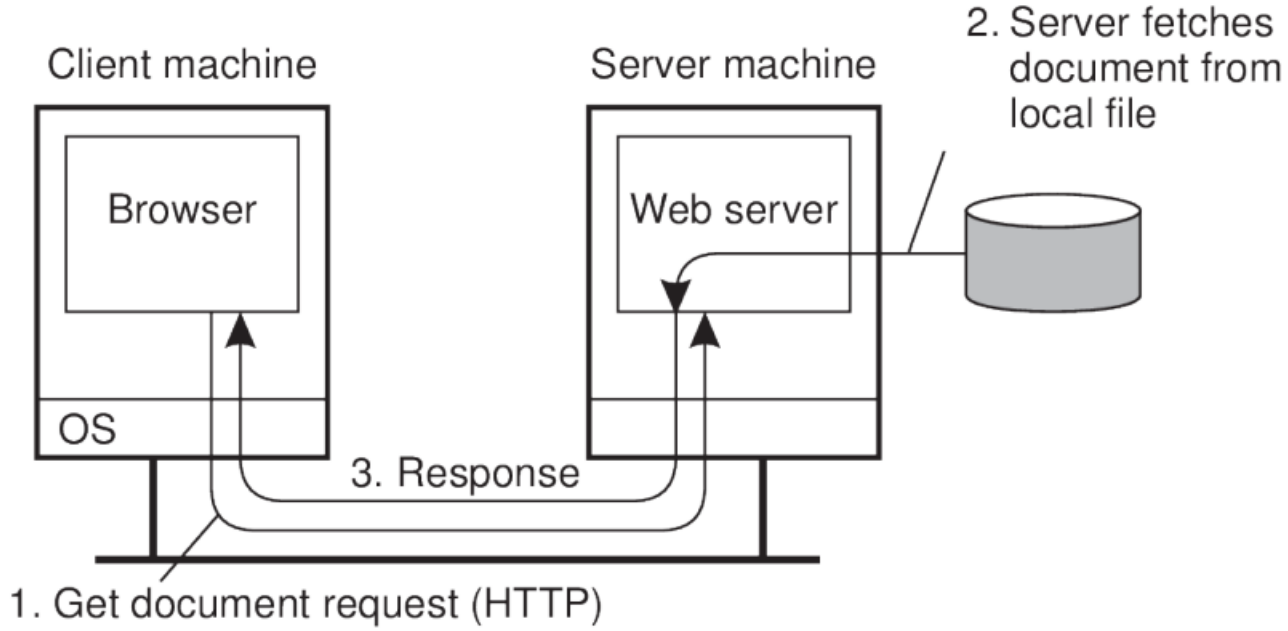
The Network File System

- Note that the operations offered by the VFS interface can be different from those offered by the NFS client.
- The whole idea of the VFS is to hide the differences between various file systems.
- On the server side, we see a similar organization.
- The NFS server is responsible for handling incoming client requests.
- The RPC component at the server converts incoming requests to regular VFS file operations that are subsequently passed to the VFS layer.
- Again, the VFS is responsible for implementing a local file system in which the actual files are stored.
- An important advantage of this scheme is that NFS is largely independent of local file systems.
- In principle, it really does not matter whether the operating system at the client or server implements a Unix file system, a Windows file system, or even an old MS-DOS file system.
- The only important issue is that these file systems are compliant with the file system model offered by NFS.

Simple Web-based Systems

- Many Web-based systems are still organized as relatively simple client-server architectures.
- The core of a Web site is formed by a process that has access to a local file system storing documents.
- The simplest way to refer to a document is by means of a reference called a uniform resource locator (URL).
- It specifies where a document is located by embedding the DNS name of its associated server along with a file name by which the server can look up the document in its local file system.
- Furthermore, a URL specifies the application-level protocol for transferring the document across the network.
- A client interacts with Web servers through a browser, which is responsible for properly displaying a document.
- Also, a browser accepts input from a user mostly by letting the user select a reference to another document, which it then subsequently fetches and displays.
- The communication between a browser and Web server is standardized: they both adhere to the HyperText Transfer Protocol (HTTP).

Simple Web-based Systems



The overall organization of a traditional Web site

- Let us zoom in a bit into what a document actually is.
- Perhaps the simplest form is a standard text file. In that case, the server and browser have barely anything to do: the server copies the file from the local file system and transfers it to the browser.
- The latter, in turn, merely displays the content of the file without further a do.

Simple Web-based Systems

- More interesting are Web documents that have been marked up, which is usually done in the HyperText Markup Language, or simply HTML.
- In that case, the document includes various instructions expressing how its content should be displayed.
- For example, instructing text to be emphasized is done by the following markup: <emph>Emphasize this text</emph>
- There are many more of such markup instructions. The point is that the browser understands these instructions and will act accordingly when displaying the text.
- Documents can contain much more than just markup instructions. In particular, they can have complete programs embedded of which Javascript is the one most often deployed.

Simple Web-based Systems

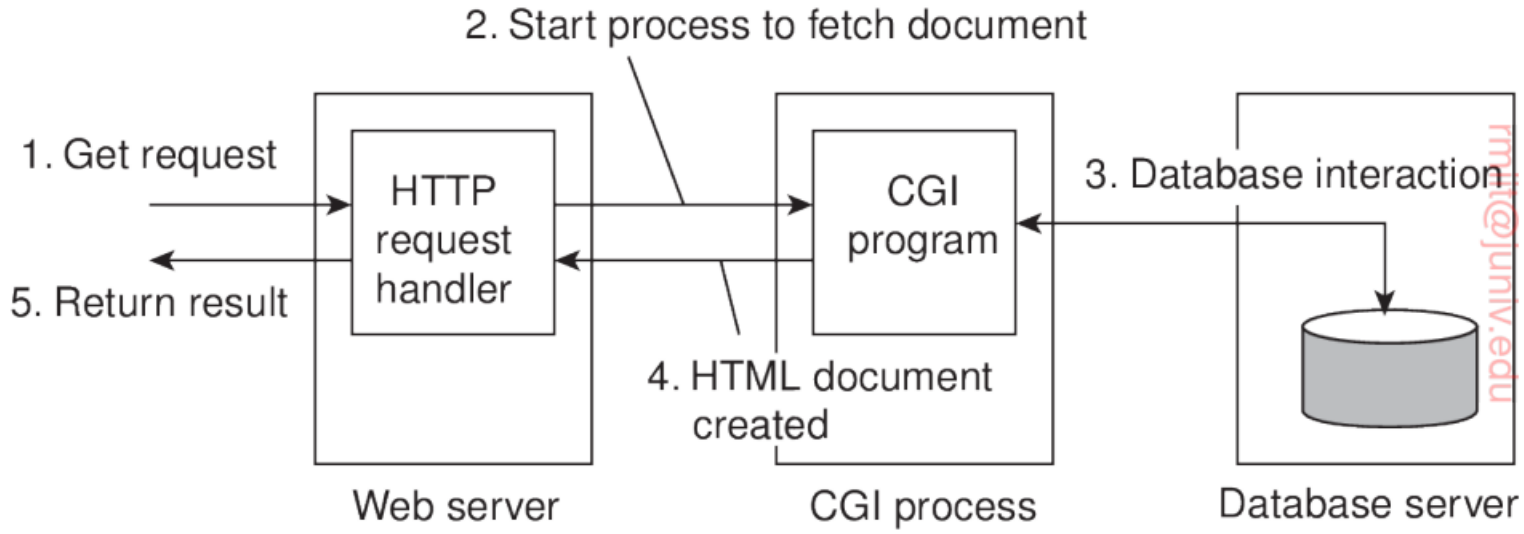
- In this case, the browser is warned that there is some code to execute as in: <script type=”text/javascript”>....</script> and as long as the browser has an appropriate embedded interpreter for the specified language, everything between “<script>” and “</script>” will be executed as any other program.
- The main benefit of including scripts is that it allows for much better interaction with the end user, including sending information back to the server.

Multitiered Architectures

- The Web started out as the relatively simple two-tiered client-server system shown in the previous figure.
- By now, this simple architecture has been extended to support much more sophisticated means of documents.
- In fact, one could justifiably argue that the term “document” is no longer appropriate.
- For one most things that we get to see in our browser has been generated on the spot as the result of sending a request to a Web server.
- Content is stored in a database at the server’s side, along with client-side scripts and such, to be composed on-the-fly into a document which is then subsequently sent to the client’s browser.
- Documents have thus become completely dynamic.
- One of the first enhancements to the basic architecture was support for simple user interaction by means of the Common Gateway Interface or simply CGI.

Multitiered Architectures

- CGI defines a standard way by which a Web server can execute a program taking user data as input.
- Usually, user data come from an HTML form; it specifies the program that is to be executed at the server side, along with parameter values that are filled in by the user.
- Once the form has been completed, the program's name and collected parameter values are sent to the server, as shown in the following figure:



The principle of using server-side CGI programs.

Multitiered Architectures

- When the server sees the request, it starts the program named in the request and passes it the parameter values.
- At that point, the program simply does its work and generally returns the results in the form of a document that is sent back to the user's browser to be displayed.
- CGI programs can be as sophisticated as a developer wants.
- For example, as shown in the figure, many programs operate on a database local to the Web server.
- After processing the data, the program generates an HTML document and returns that document to the server.
- The server will then pass the document to the client.
- An interesting observation is that to the server, it appears as if it is asking the CGI program to fetch a document.
- In other words, the server does nothing but delegate the fetching of a document to an external program.

Multitiered Architectures

- The main task of a server used to be handling client requests by simply fetching documents.
- With CGI programs, fetching a document could be delegated in such a way that the server would remain unaware of whether a document had been generated on the fly, or actually read from the local file system.
- Note that we have just described a two-tiered organization of server-side software.
- However, servers nowadays do much more than just fetching documents.
- One of the most important enhancements is that servers can also process a document before passing it to the client.
- In particular, a document may contain a server-side script, which is executed by the server when the document has been fetched locally.

Multitiered Architectures

- The result of executing a script is sent along with the rest of the document to the client.
- The script itself is not sent. In other words, using a server-side script changes a document by essentially replacing the script with the results of its execution.
- To make matters concrete, take a look at a very simple example of dynamically generating a document.
- Assume a file is stored at the server with the following content:

   <?php echo $_SERVER['REMOTE_ADDR']; ?>
- The server will examine the file and subsequently process the PHP code (between "<?php" and "?>") replacing the code with the address of the requesting client.
- Much more sophisticated settings are possible, such as accessing a local database and subsequently fetching content from that database to be combined with other dynamically generated content.

Thank You ☺