MANCHESTER 1824

The University of Manchester

Architectures of Distributed Systems

Decentralised System Architectures: Peer-to-Peer Systems

- In decentralised architectures, there is a greater concern about distributing client and server functionality more evenly across machines to achieve better workload balance.
- As a consequence, a client (or a server) may be *physically* split up into a number of *logical parts*, with each part operating on "its own share of the complete data set". This is a **horizontal distribution** of functionality.
- A class of modern system architectures that support this horizontal distribution is known as peer-to-peer systems.

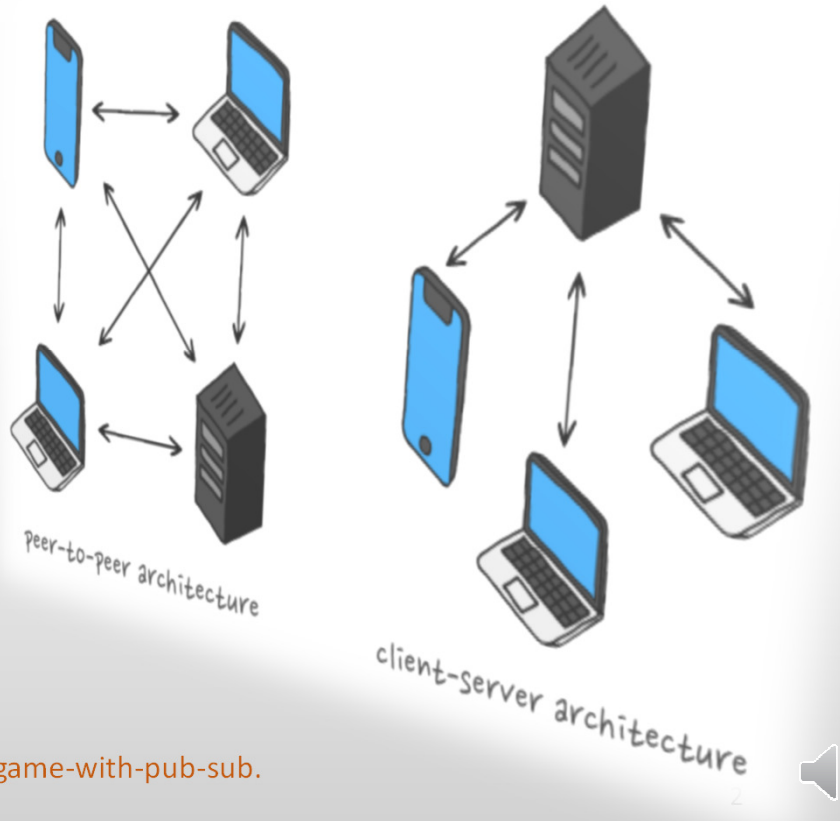


Figure source: https://ably.com/blog/peer-to-peer-game-with-pub-sub.

- From a high-level perspective, the processes that constitute a peer-to-peer system are all equal.
- Much of the interaction between processes is symmetric: each process will act as a client and a server.

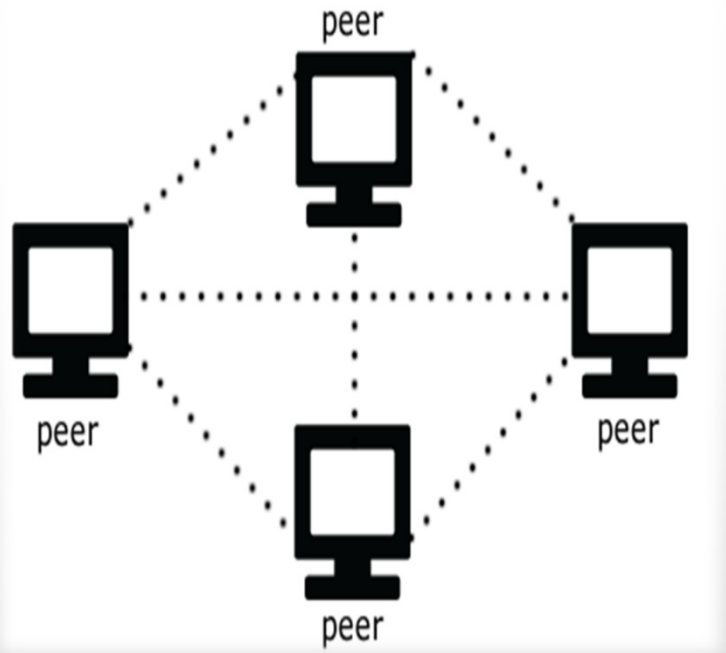
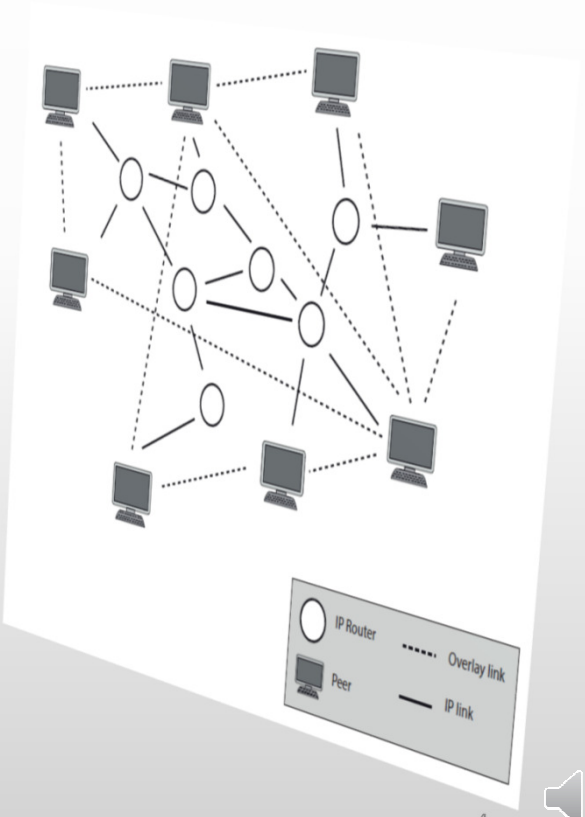


Figure source: https://gamedevelopment.tutsplus.com/tutorials/building-a-peer-to-peer-multiplayer-networked-game--gamedev-10074

- Due to the symmetric behaviour of processes in peer-to-peer architectures, processes are organised in an overlay network;
  - i.e., its nodes are formed by the processes and its links represent the possible communication channels (TCP connections).
  - Thus node may not be able to communicate directly with an arbitrary other node, but is required to send messages through the available communication channels.
- Two types of overlay networks exist, characterising peer-to-peer systems as:
  - Structured
  - Unstructured

The Transmission Control Protocol (TCP) is one of the main Internet Protocols.



Structured Peer-to-Peer Systems

- In structured P-2-P systems, nodes are organized in an overlay that adheres to a specific, deterministic topology: a ring, a binary tree, a grid, etc.
- This topology is used to efficiently look up data that is maintained by the system,
  - i.e., each data item is uniquely associated with a key, typically obtained by a hash function on the data item's value. This key is used as an index, since it identifies a node in the system.

key(data item) = hash(data item's value)

- The topology of a structured peer-to-peer system plays a crucial role: any node can be asked to *look up* a given key, i.e., to efficiently *route a request for data* to the node responsible for storing the data associated with the given key.

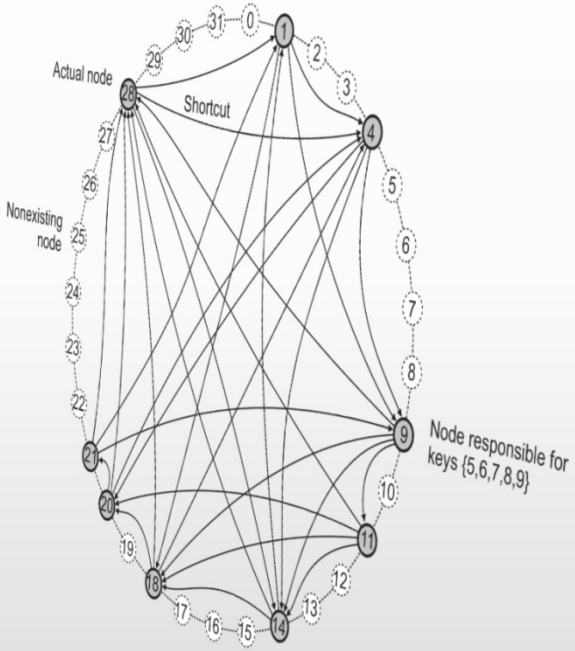


Figure source: https://medium.com/@macworks/chord-52c452808a60

A Simple Example

- A peer-to-peer system with a fixed number of nodes, organised into a hypercube.
- Each data item is associated with one of the 16 nodes of the hypercube - by hashing the value of a data item to a key k ϵ {0, 1, 2, . . . , $2^4 - 1$}.

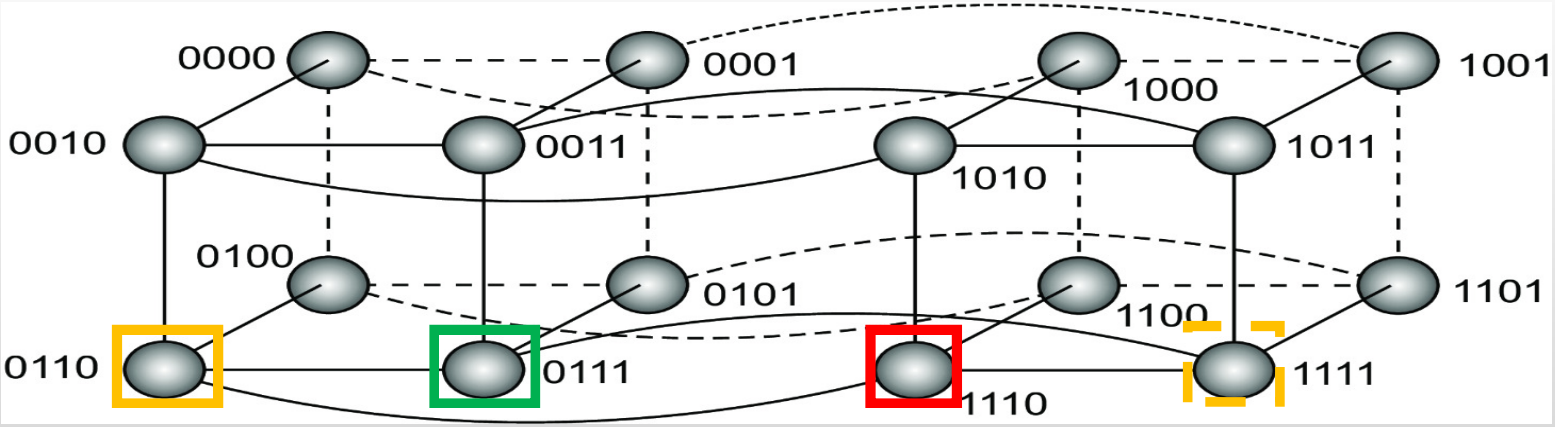


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

Unstructured Peer-to-Peer Systems

- In an unstructured peer-to-peer system each node maintains an ad hoc list of neighbours, such that the resulting overlay resembles a random graph: a graph in which an edge <u, v> between two nodes u and v exists only with a certain probability P[<u, v>].
- When a node joins in, it often contacts a well-known node to obtain a starting list of other peers in the system. This list can then be used to find more peers, and perhaps ignore others, and so on. In practice, a node generally changes its local list almost continuously.
- Unlike structured P-2-P systems, looking up data cannot follow a predetermined route, because lists of neighbours are constructed in an ad hoc fashion. Instead, searching for data is necessary.

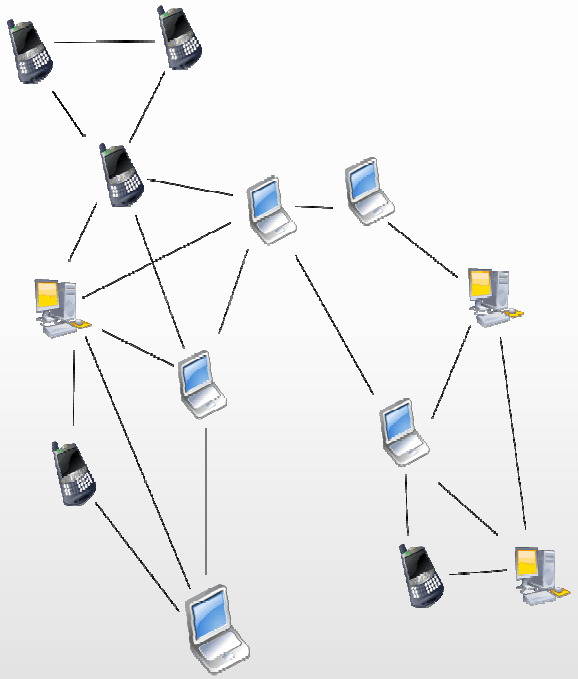


Figure source: https://networkencyclopedia.com/peer-to-peer-network-p2p/

Examples of Searching Methods

Flooding

- Assume an issuing node, *u*, passes a request for a data item to all its neighbours.
- Each of its neighbours, *v*:
  - Ignores the request when it has seen it before, otherwise, searches locally for the requested data item.
  - If it has the required data, it can either respond directly to the issuing node u, or send it back to the original forwarder, who will then return it to its original forwarder, and so on.
  - If it does not have the requested data, it forwards the request to all of its own neighbours.
- Obviously, flooding is expensive, for which reason a request often has an associated *time-to-live* or TTL value, giving the maximum number of hops a request is allowed to be forwarded.

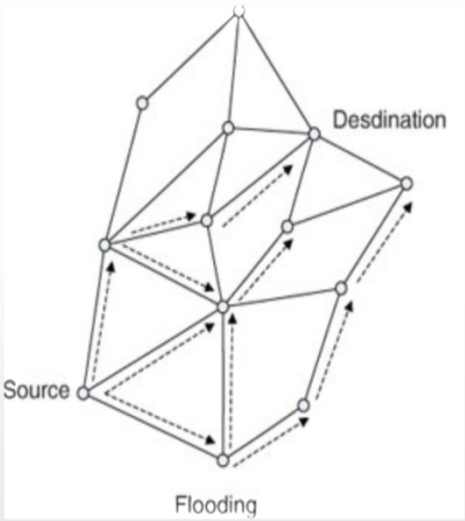
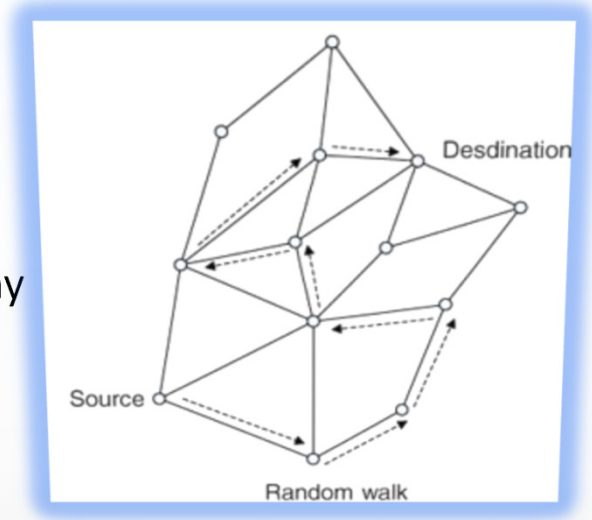


Figure source: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_1215

Random Walks

- An issuing node, u, tries to find a data item by asking a randomly chosen neighbour, v.
- If v does not have the data, it forwards the request to one of its randomly chosen neighbours, and so on.

- Generally, a random walk imposes less network traffic than Flooding, but it may take longer before a node is reached that has the requested data. To decrease the waiting time, an issuer can simply start *n* random walks simultaneously.
- A random walk also needs to be stopped. To this end, a TTL can be used, or alternatively, when a node receives a lookup request, it can check with the issuer whether forwarding the request to another randomly selected neighbour is still needed.



Notably in unstructured P-2-P systems, **locating relevant data items can become problematic as the network grows**, causing a scalability problem.

Figure source: https://link.springer.com/referenceworkentry/10.1007%2F978-0-387-39940-9_1215

Making Data Search more Scalable in Unstructured P-2-P Systems

- To improve scalability of data search, P-2-P systems can make use of special nodes that maintain an index of data items, abandoning their symmetric nature by creating “special” collaborations among nodes.
  - For example, in a collaborative content delivery network (CDN), nodes may offer storage for hosting copies of Web documents, allowing Web clients to access pages nearby, and thus to access them quickly.

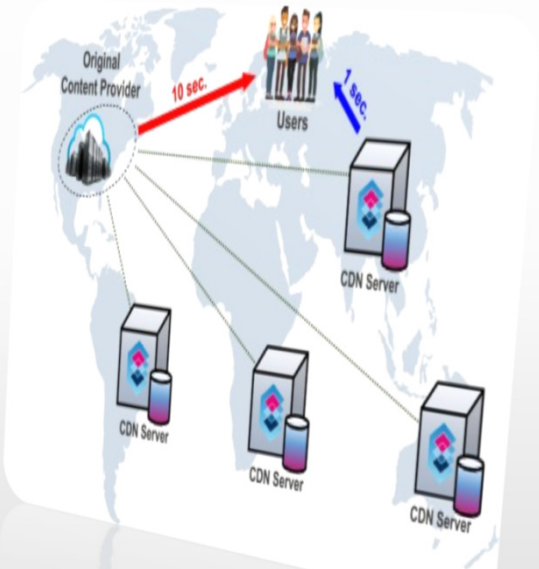


Figure source: https://www.sciencedirect.com/topics/computer-science/content-delivery-network

Hybrid Architectures

- Encompass classes of distributed systems in which client-server solutions are combined with decentralized architectures. Examples:
  - Edge-server systems
  - Collaborative distributed systems

One of the main motivations for these hybrid architectures, is the scalability problems of unstructured peer-to-peer systems, and the difficulties in workload balancing in traditional client-server architectures.

Collaborative Distributed Systems

- Combine traditional client-server structures (when nodes are joining the system) and fully decentralised structures (once a node has joined the system). An example of such a system is BitTorrent, a peer-to-peer file downloading system.
- In BitTorrent, an end user, looking for a file, downloads chunks of the file from other users, until the downloaded chunks can be assembled together, yielding the complete file.
  - To download a file, a user needs to access a global directory containing references to torrent files.
  - A torrent file contains the information that is needed to download a specific file, such as a link to a file tracker, a server that keeps an accurate account of active nodes that have (chunks of) the requested file.
  - Once the nodes have been identified from where chunks can be downloaded, the downloading node effectively becomes active.

There will be many different **trackers**, although there will generally be only a single tracker per file (or collection of files). Once the nodes have been identified from where chunks can be downloaded, the downloading node effectively becomes active.

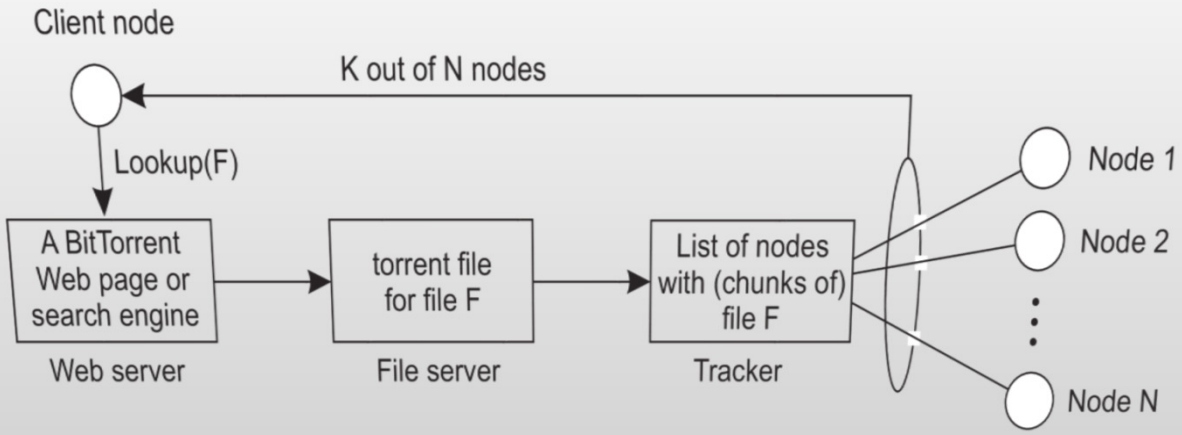


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

Edge-Server Systems

- Are characterised by the following main properties:
  - Are deployed on the Internet
  - Their servers are placed “at the edge” of the network (i.e., the boundary between enterprise networks and the actual Internet).
- The edge server’s main purpose is to serve content, possibly after applying filtering and transcoding functions.
  - 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 high-demand content, e.g., Web pages.

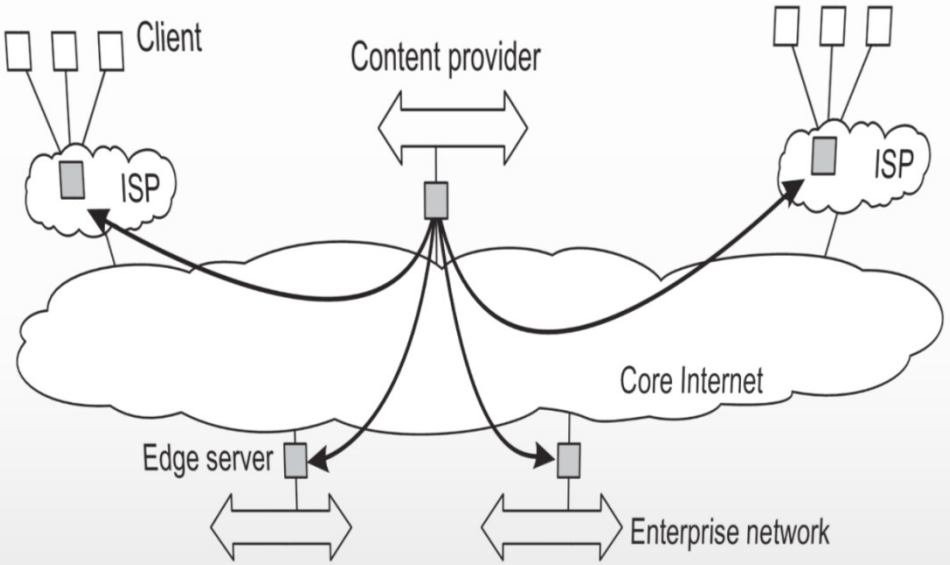


Figure from book 'Distributed Systems Third edition M. van Steen and A. S. Tanenbaum'

Edge-server systems have recently been used to assist data centers in cloud computations and storage, leading to distributed cloud systems. In the case of fog computing, even end-user devices form part of the system and are (partly) controlled by a cloud-service provider.