Architectures

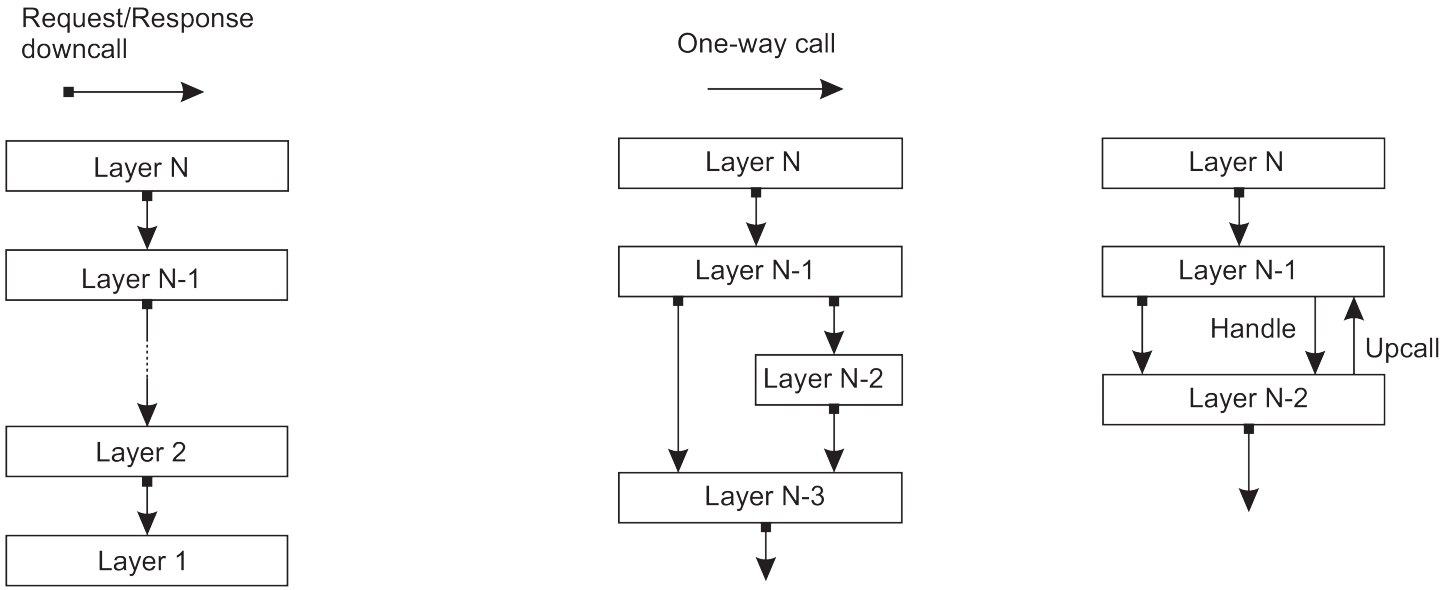
Dr. Padraig Corcoran

- A **software architecture** concerns the logical organisation of the software components.
- Important software architectures in distributed systems are:
  - Layered architecture.
  - Object-based architecture.
  - Service-Oriented Architecture (SOA).

- A **system architecture** concerns where these software components are placed across the various machines.
- Important system architectures in distributed systems are:
  - Centralised client-server architecture.
  - Decentralised peer-to-peer architecture.
  - Hybrid architecture.
  - Edge computing

Layered architectures

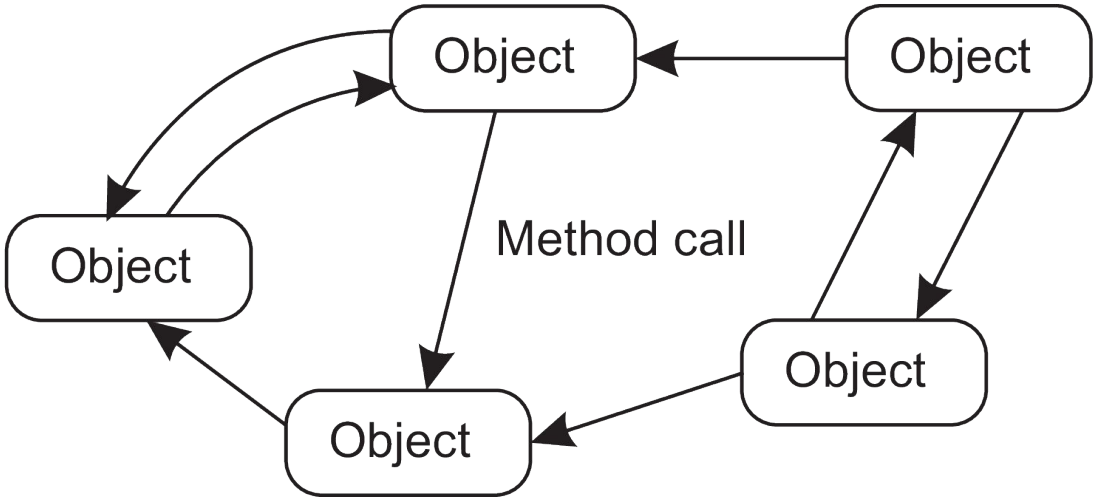
- Components are organized in a layered fashion.
- A component at layer j can make a downcall to a component at a lower-level layer i (with i<j).
- Only in exceptional cases will an upcall be made.



(a) Pure layered organization. (b) Mixed layered organisation. (c) Layered organization with upcalls

Object-based architecture

- Objects provide a way of *encapsulating* data and operations that can be performed on that data.
- Each object corresponds to a software component and these are connected through method calls.



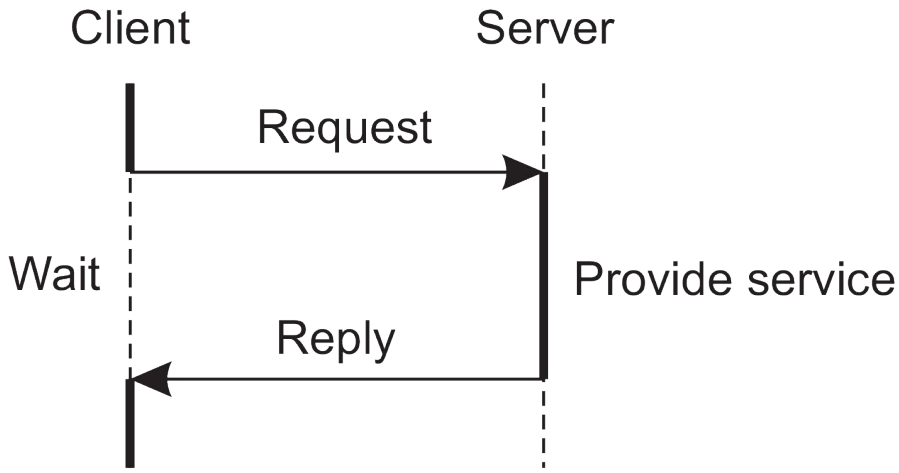
An object-based architectural style.

Service-Oriented Architecture (SOA)

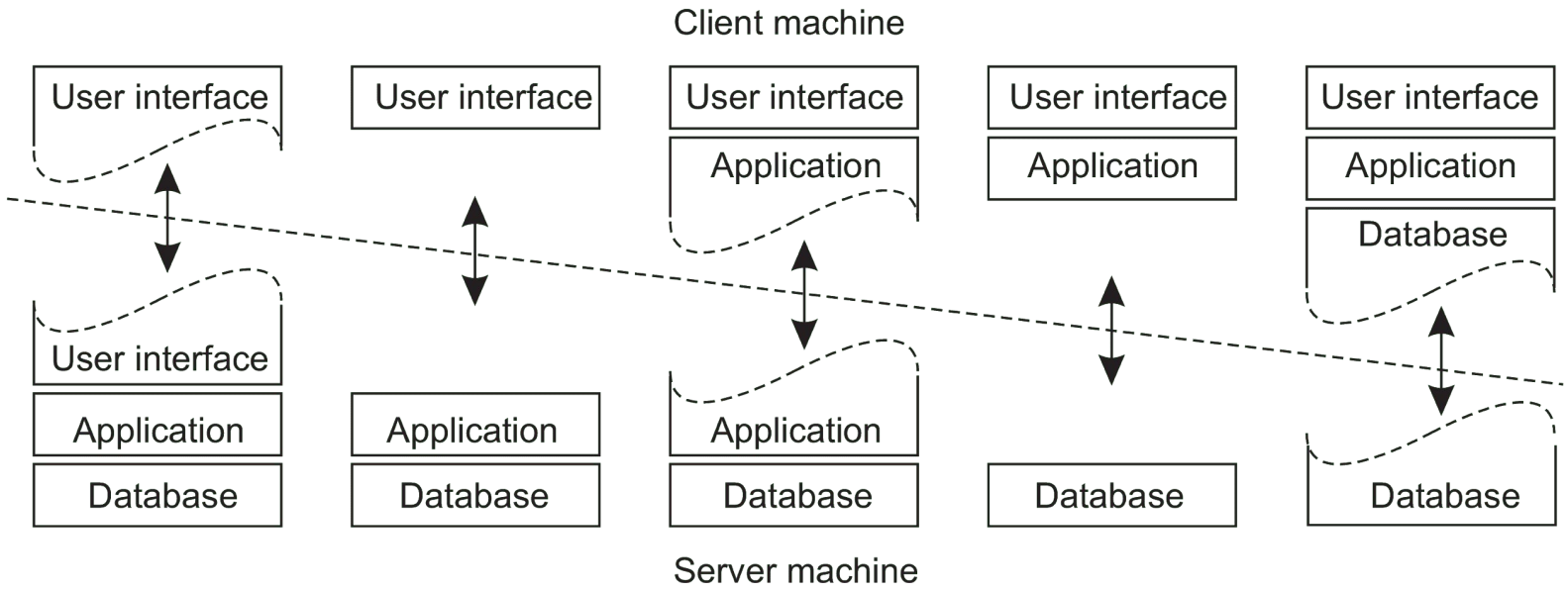
- A distributed system is constructed as a composition of different services (e.g. payment and database services).
- Developing a distributed system involves service composition or mashup.
- Each service must offer a well-defined (programming) interface.
- Can be implemented using Web services (e.g. SOAP and RESTful web services).

Centralized client-server architecture

- Processes divided into two (possibly overlapping) groups.
- A client is a process that requests a service from a server by sending it a request and subsequently waiting for reply.
- This client-server interaction, also known as **request-reply behaviour,** is shown.

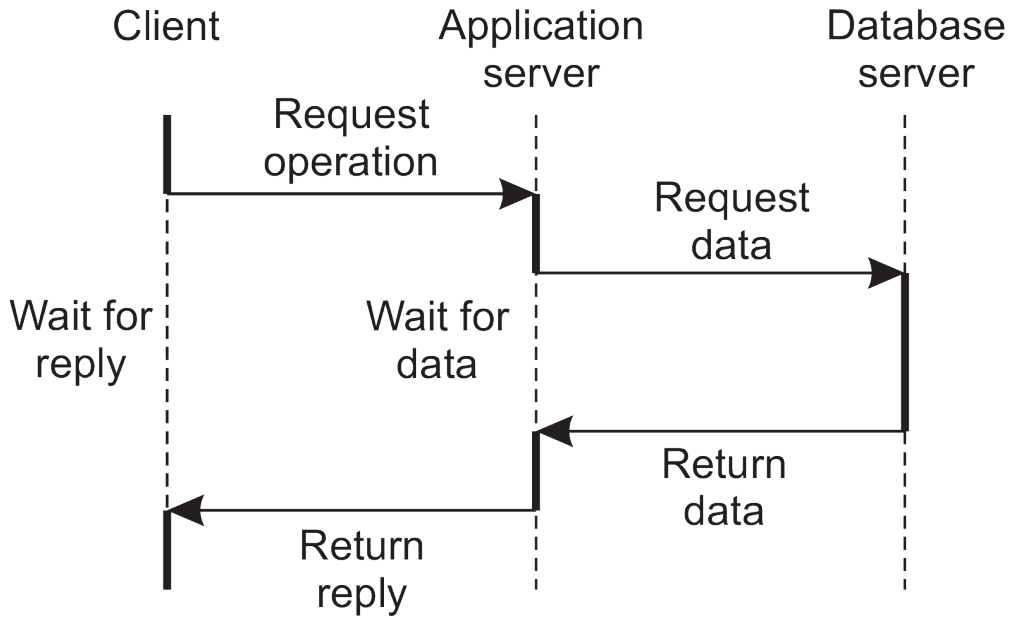


General interaction between a client and a server.



Thin and fat clients are possible.

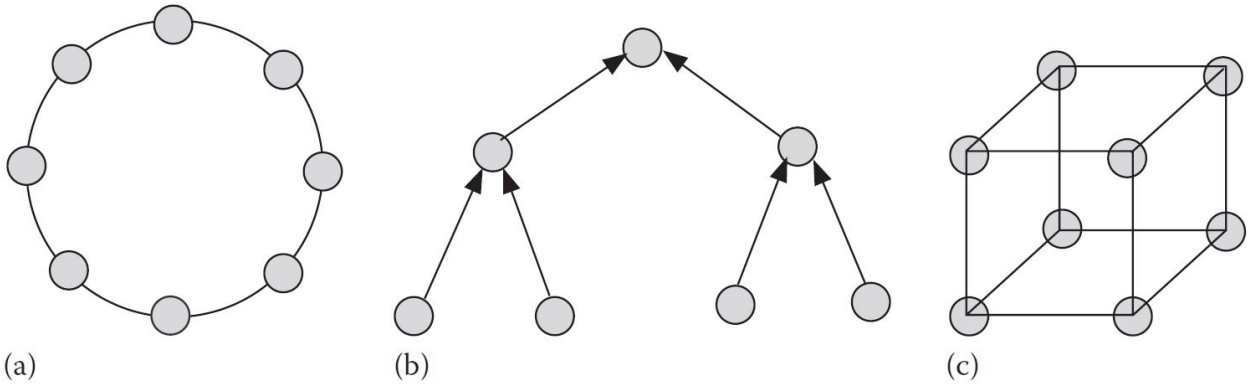
- A server may sometimes need to act as a client.



An example of a server acting as client.

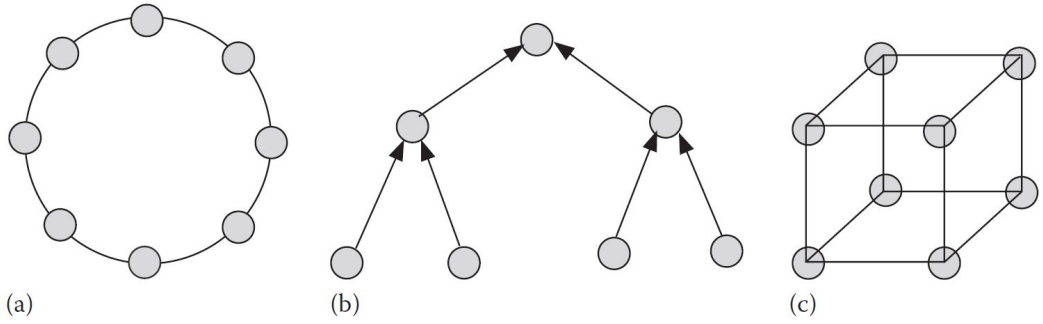
Network topology

- A graph describing the layout of the distributed system.
- Nodes can correspond to physical devices or processes.
- Edges correspond to communication channels.

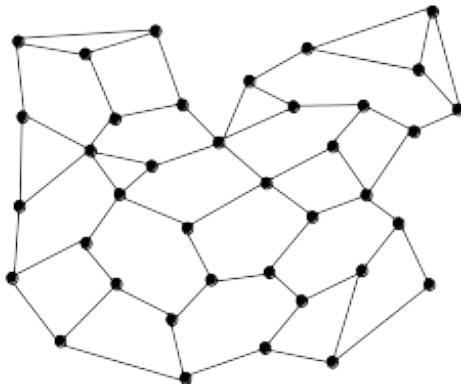


Examples of network topologies: (a) ring, (b) directed tree and (c) 3D cube.

- The topology is **structured** if it “has a pattern”.
- The topology is **unstructured** if the edges are random.
- In unstructured networks many tasks, such as routing, are more complicated.

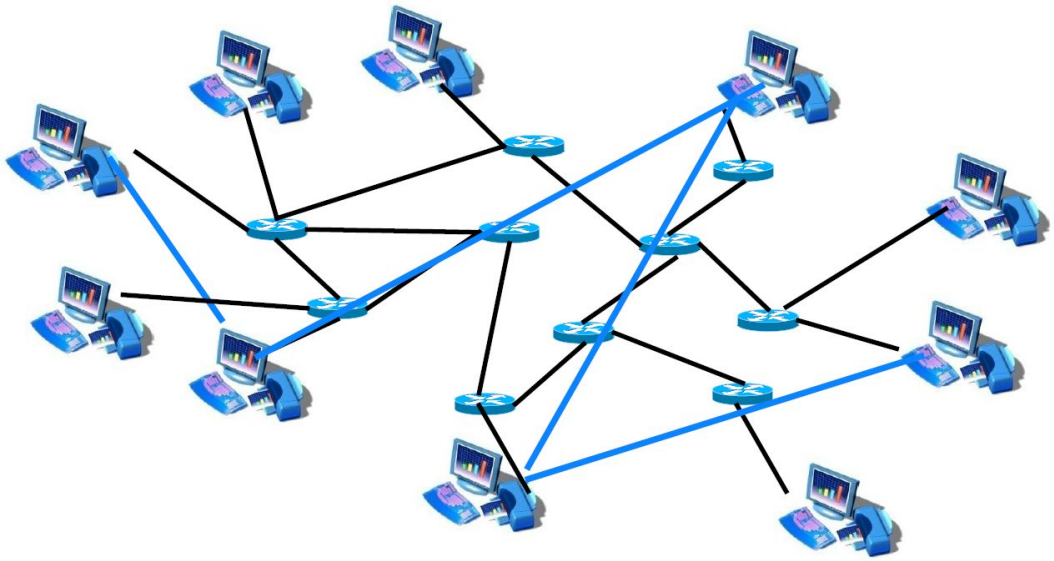


Structured topology



Unstructured topology

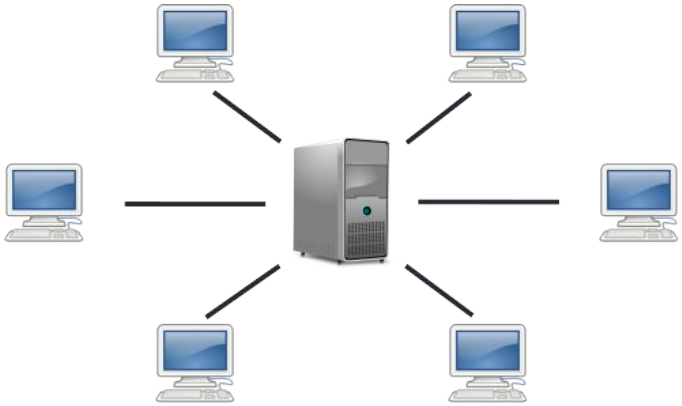
- If the nodes correspond to processes, the graph is commonly referred to as an **overlay network**.
- Nodes in the overlay network can be thought of as being connected by virtual or logical links, each of which corresponds to a path in the underlying network.



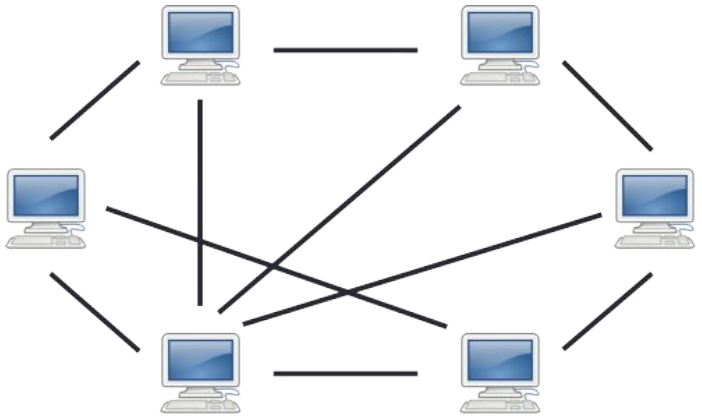
Example of overlay network.

Decentralized peer-to-peer (P2P) architecture

- **Vertical distribution** is characterised by having logically different processes (e.g. client-server model).
- **Horizontal distribution** is characterised by having logically equal processes (e.g. peer-to-peer model).
- The processes in a peer-to-peer system are all equal (will act as both client and server).
- The peer-to-peer architecture is commonly employed in file sharing systems.

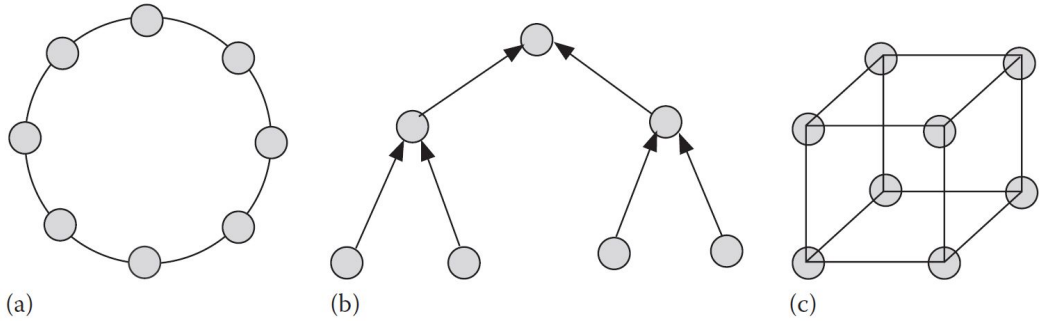


Client-server model

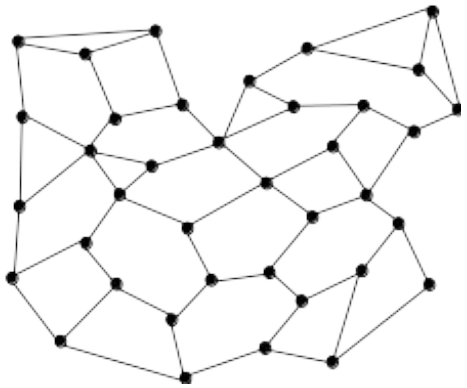


Peer-to-peer model

- In a **structured peer-to-peer architecture** the network topology is structured.
- In an **unstructured peer-to-peer architecture** the network topology is unstructured.



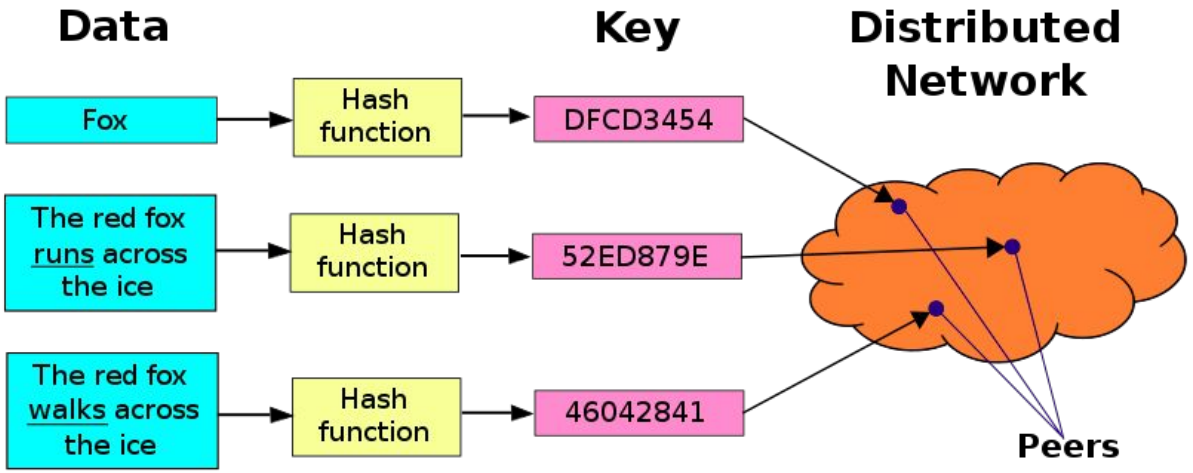
Structured topology



Unstructured topology

Locating data in a P2P system

- A given node wants to locate/access data stored on a different node.
- The node seeking the data only knows the **key** corresponding to the data in question.
- Problem - given a key locate the node containing the corresponding data.



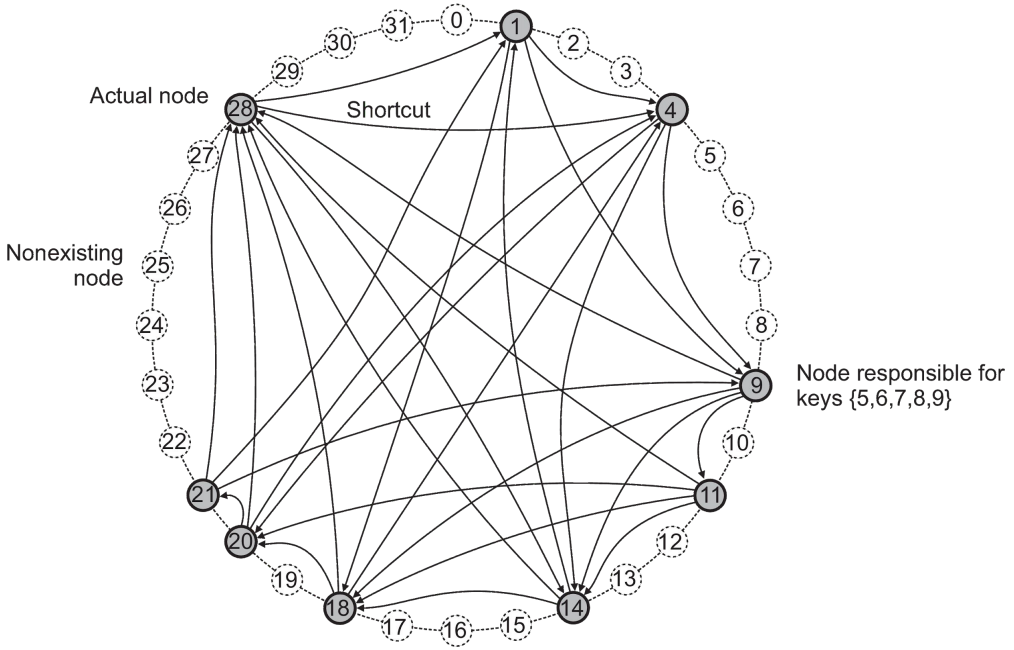
https://en.wikipedia.org/wiki/Distributed_hash_table

- Keys are often determined using a hash function.
- A hash function *H(.)* reduces a variable length input to a fixed length output.
- A hash function will have the following properties:
  - Computed $m = H(M)$ is easy but computing the inverse is impossible.
  - If $M \neq M'$ then $H(M) \neq H(M')$.

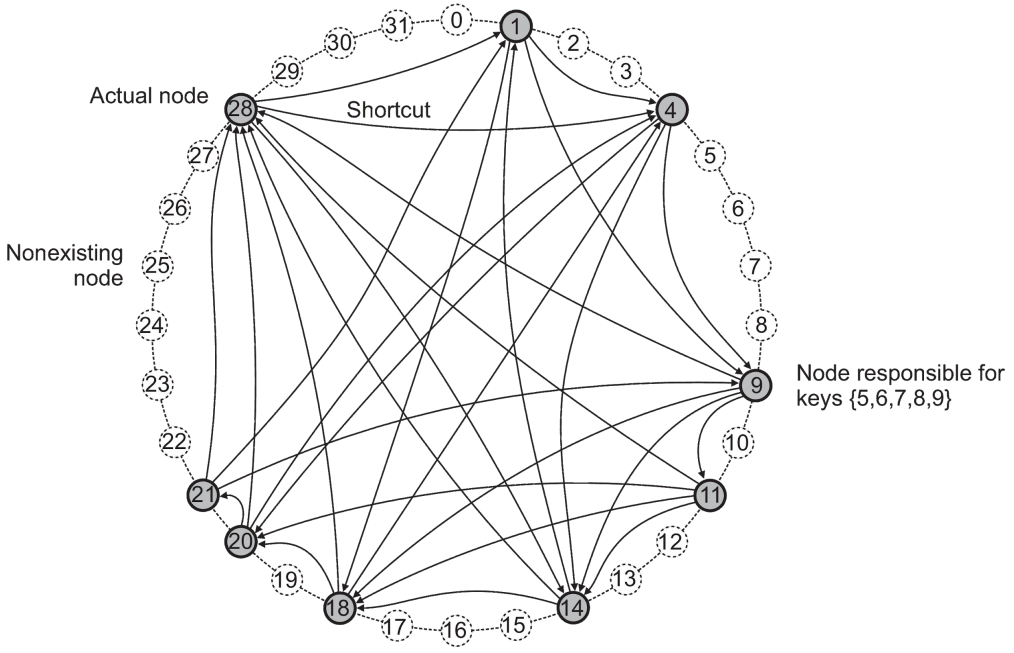
- In structured peer-to-peer systems, one can efficiently determine the location of data (mapping from **data key** to **data location**).
- Algorithms include **Chord**.
- In unstructured peer-to-peer system this is not the case and instead searching must be performed.
- Searching algorithms include **flooding** and **random walk**.

Searching in structured P2P using Chord

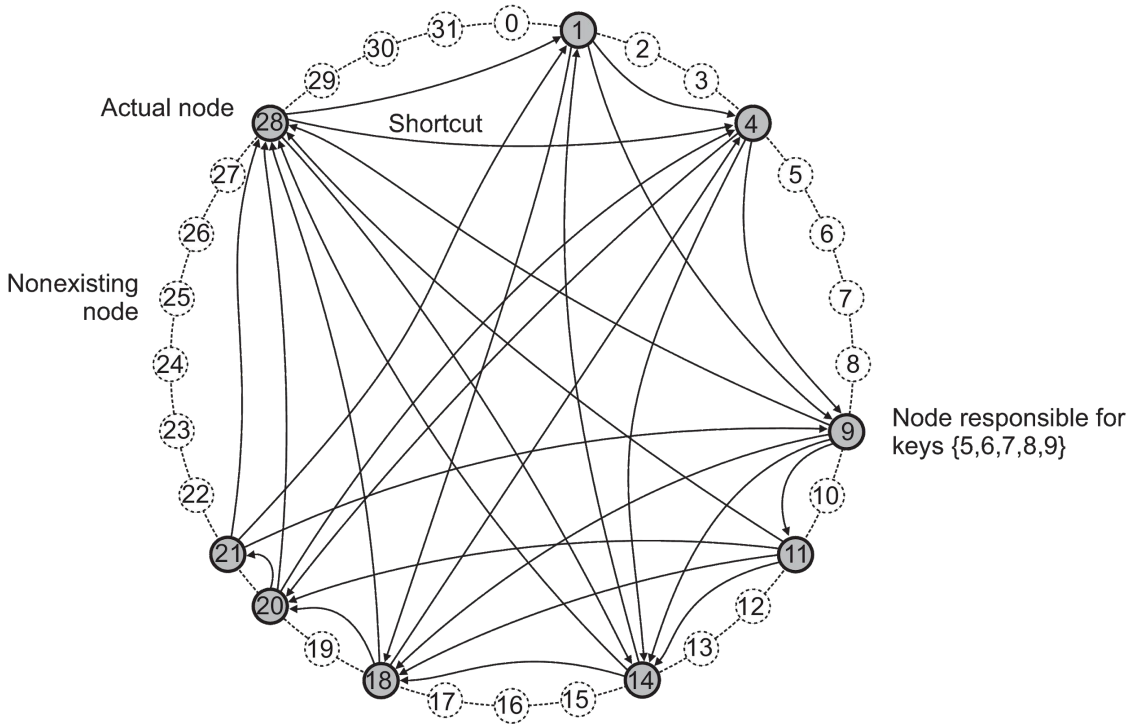
- Nodes organized in a ring topology.
- Each node is assigned an *identifier* number.
- Each data item is assigned a *key* number.



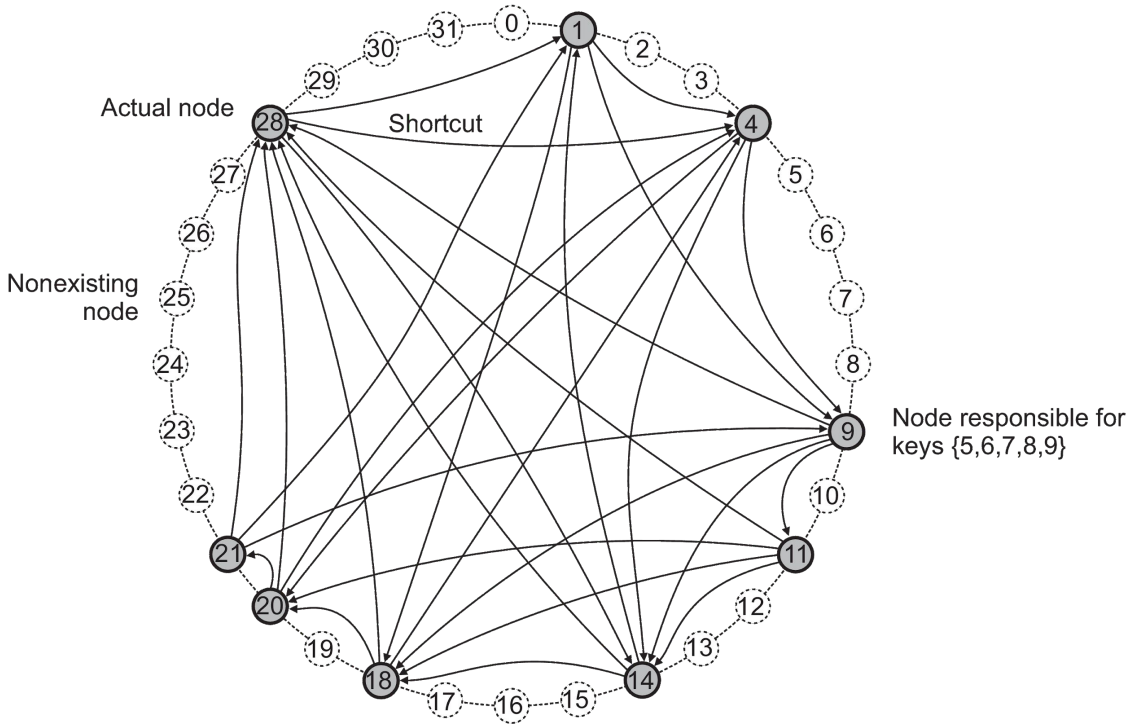
- Data item with key *k* is mapped to node with smallest identifier *p* greater than or equal to *k*.
- This node is referred to as the successor of key *k* and denoted *succ(k)*.
- In this example, succ(5) = 9 and succ(9) = 9.



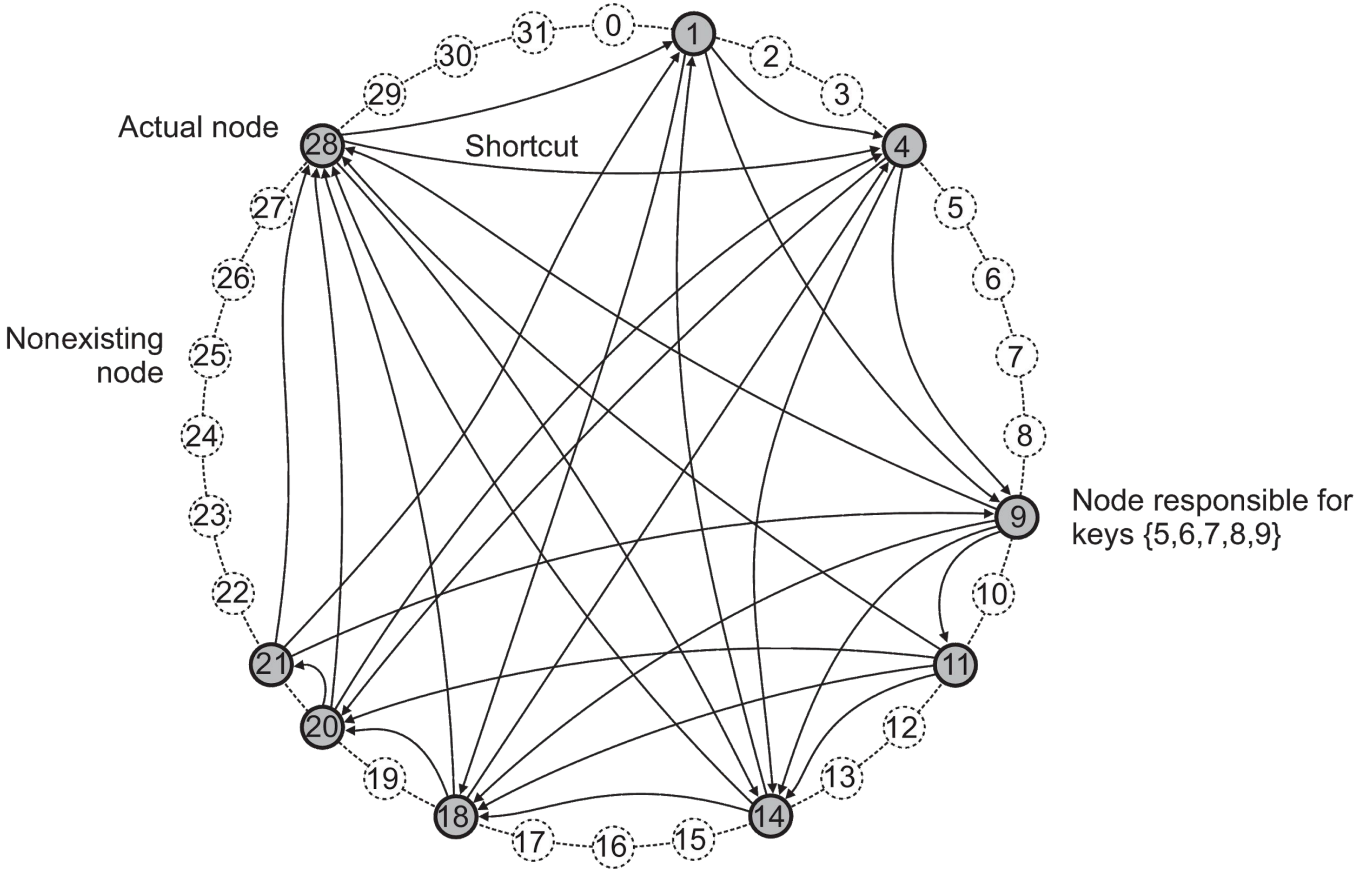
- Each node maintains shortcuts to other nodes.
- This ensures length of the shortest path between any pair of nodes is O(log N) where N is the number of nodes.



- To look up a key, a node will try to forward the request “as far as possible” but without passing it beyond the node responsible for that key.



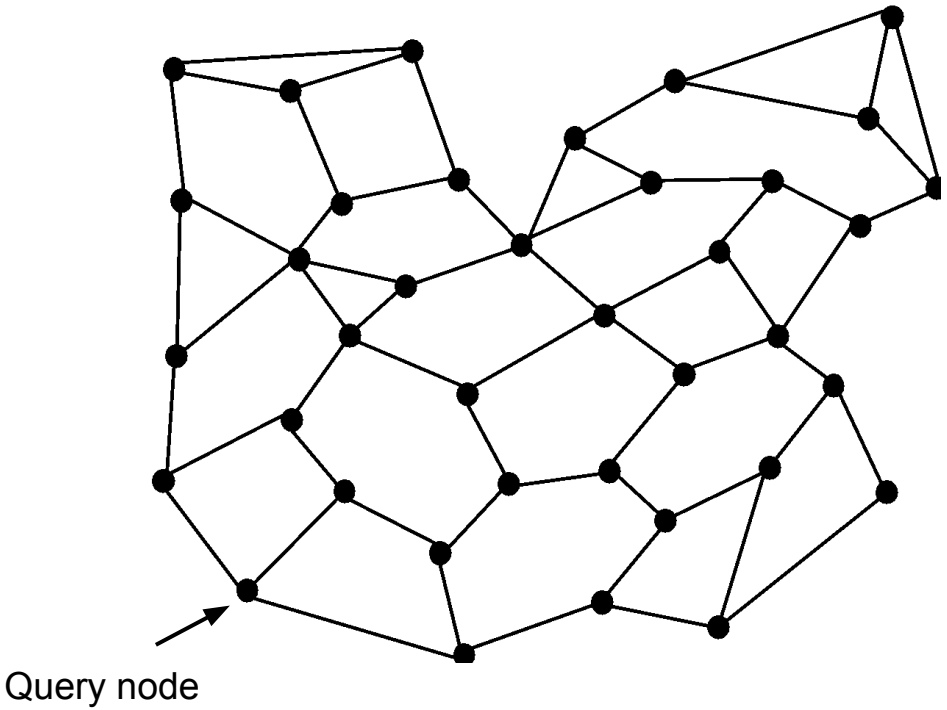
- Example - node 9 is asked to look up the node responsible for key 3 (which is node 4).

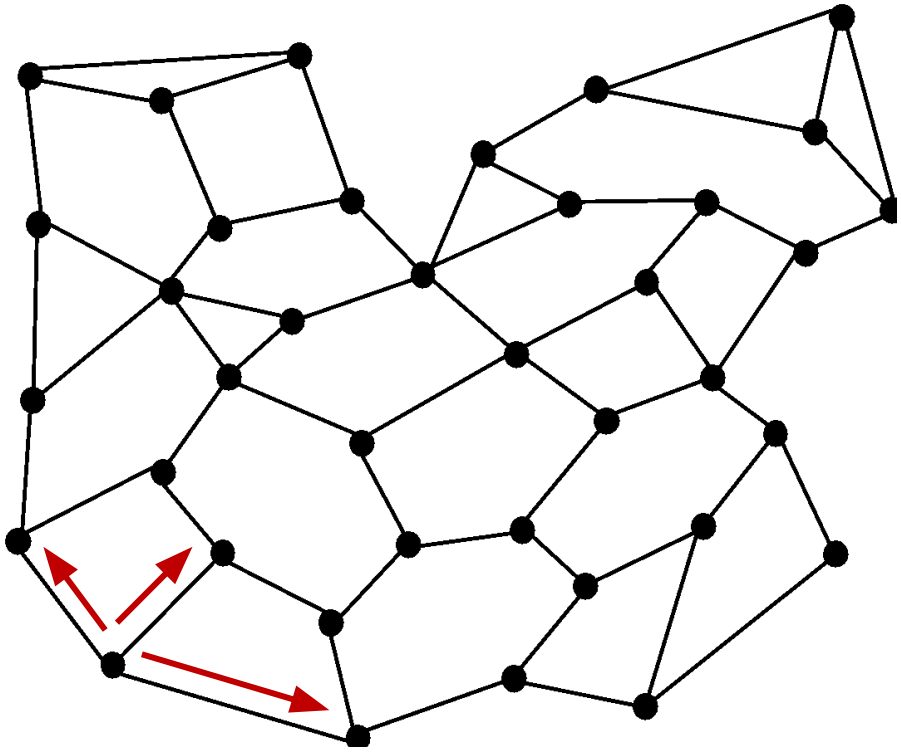


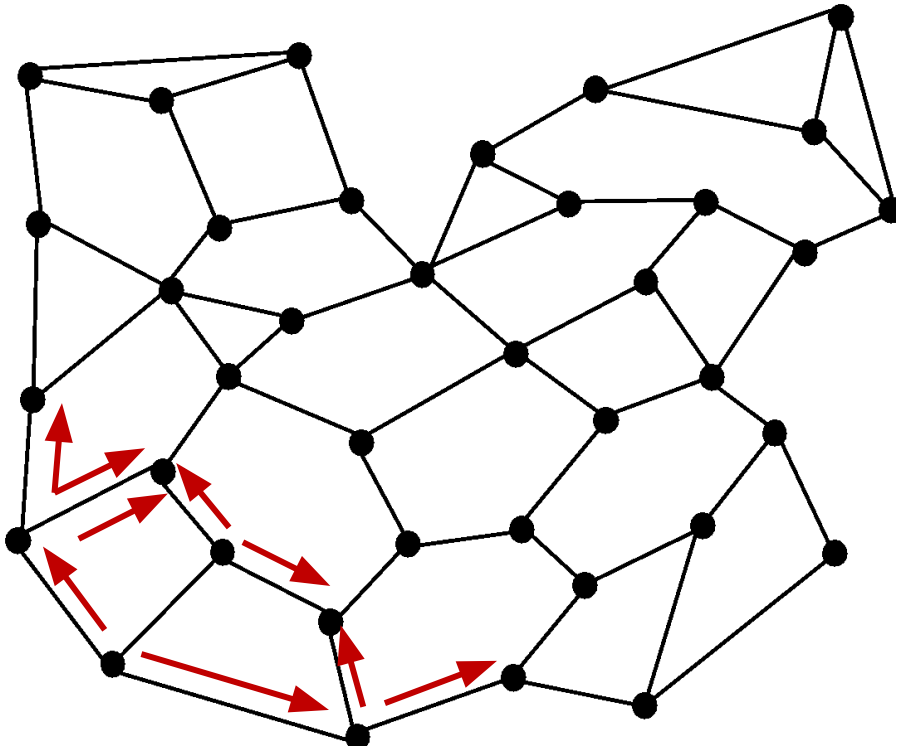
- Node 9 has four shortcuts: to nodes 11, 14, 18, and 28.
- Node 28 is the farthest node and still preceding the one responsible for key 3; it will get the lookup request.
- Node 28 has three shortcuts: to nodes 1, 4, and 14.
- Node 1 is the farthest node and still preceding the one responsible for key 3; it will get the lookup request.
- Node 1 knows that its successor in the ring is node 4, and thus that this is the node responsible for key 3, to which it will subsequently forward the request.

Searching in unstructured P2P using flooding

- In flooding an issuing node passes a request for a data item to **all neighbours**.
- A receiving node forwards the request to **all** its own neighbours.
- Searching terminates when the data is found or maximum number of hops (time-to-live or TTL) is reached.

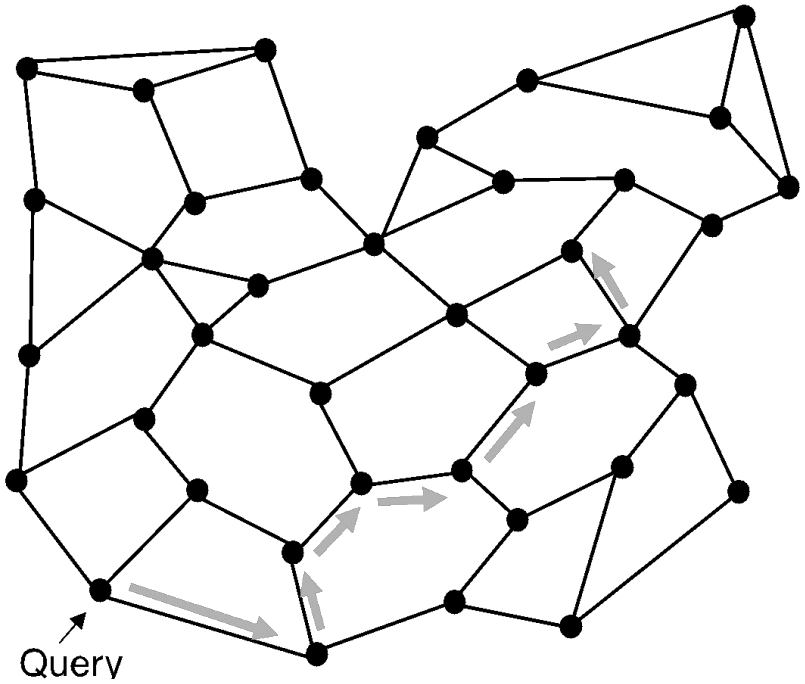

Query node





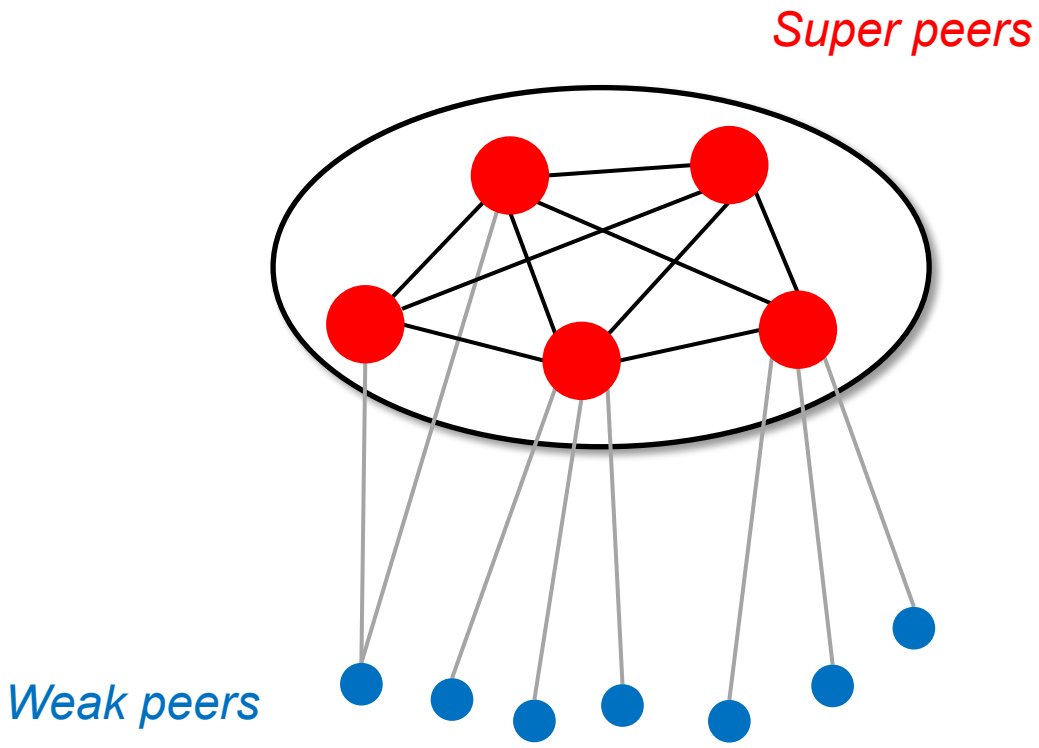
Searching in unstructured P2P using random walk

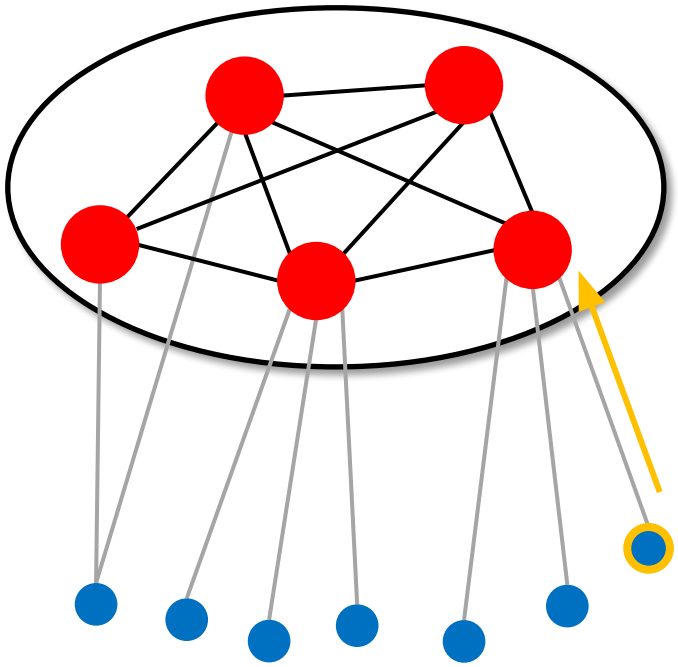
- In random walk an issuing node passes a request for a data item to a **randomly chosen neighbour**.
- A receiving node forwards the request to a **randomly chosen neighbour**.
- Searching terminates when the data is found or maximum number of hops (time-to-live or TTL) is reached.
- To decrease waiting time, an issuer can start multiple random walks simultaneously.

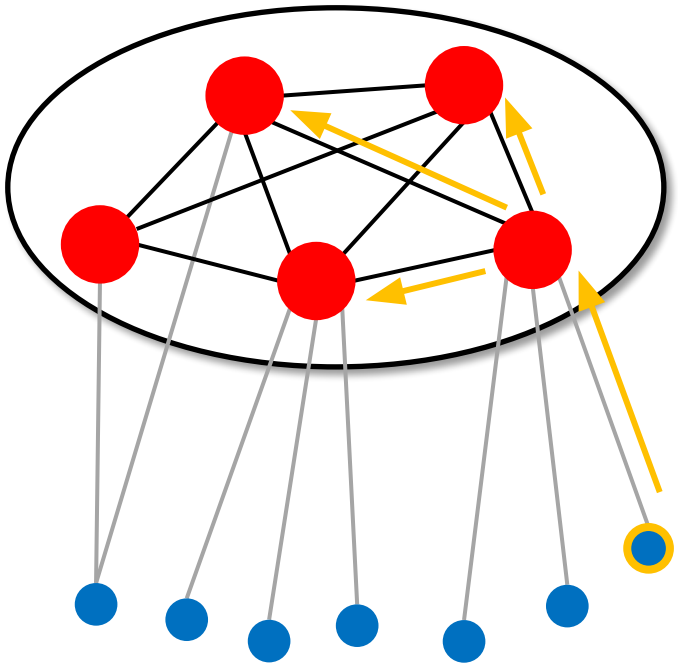

Query

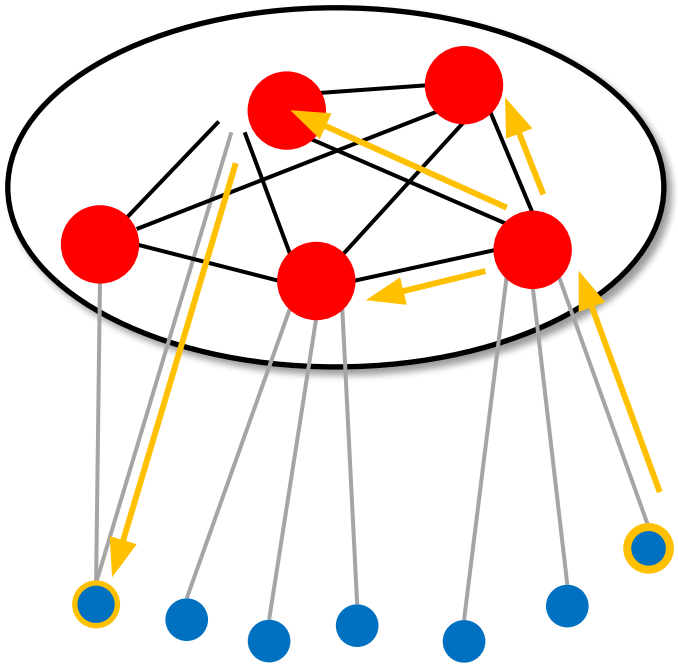
Hierarchically organized P2P architecture

- In unstructured peer-to-peer systems, locating data can be slow in large networks (not scalable).
- One solution is to use **super peers** that maintain an index of data items on a subset of nodes.
- Every regular peer, called a **weak peer**, is connected to a super peer.
- All communication from and to a weak peer proceeds through that peer's associated super peer.
- Reduces the number of hops required.


Super peers
Weak peers

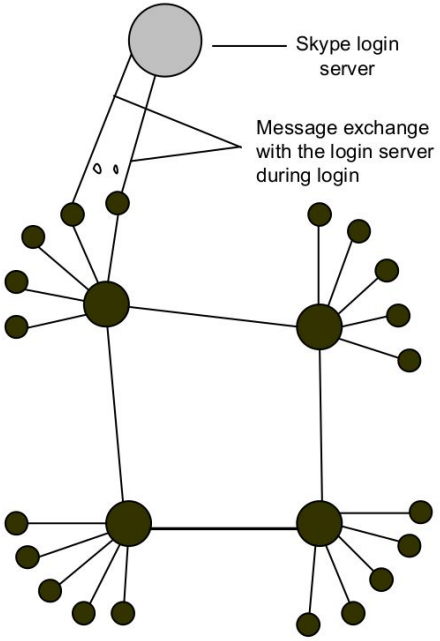


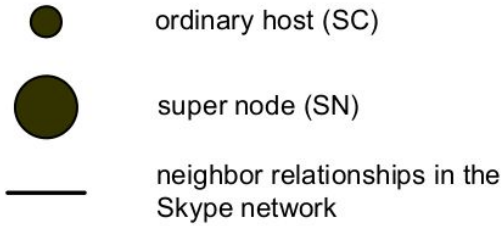




Skype

- Skype is an example of a hierarchically organized peer-to-peer architecture.
- There is an additional Skype login server.





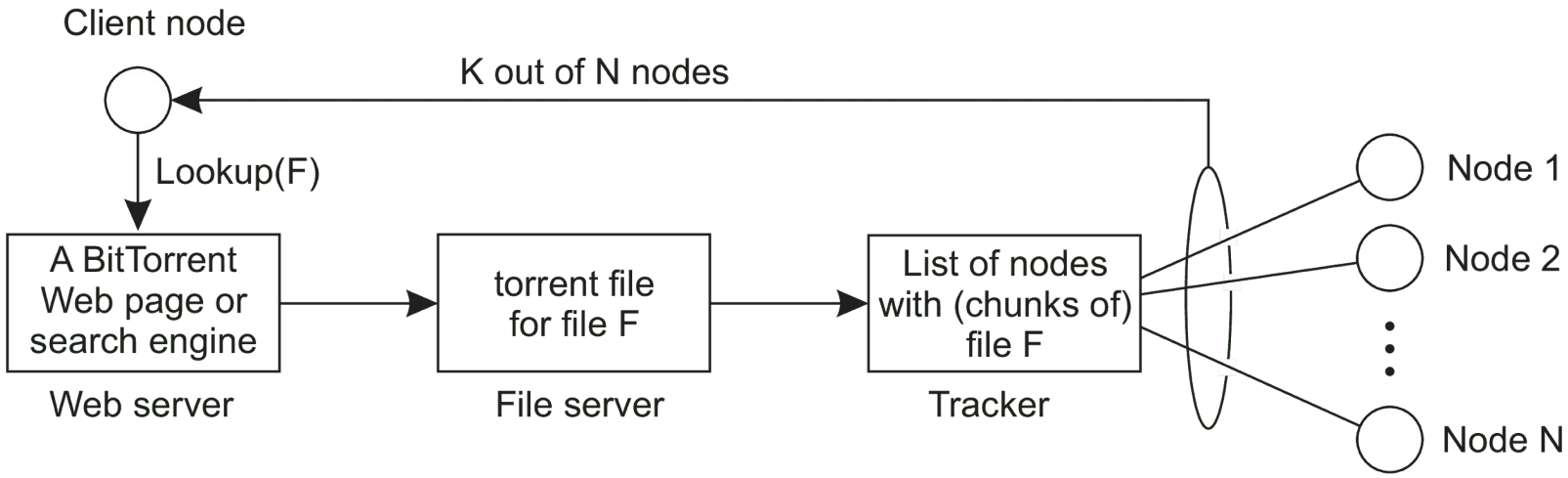
Baset, S.A. and Schulzrinne, H.G. (2006). An analysis of the Skype peer-to-peer Internet telephony protocol. In Proceedings of INFOCOM'06, pp. 1–11.

Structured Vs Unstructured P2P

	Advantage	Disadvantage
Structured	Guaranteed to locate objects, with time and complexity bounds; relatively low message overhead	Need to build and maintain complex overlay structures
Unstructured	Self-organizing and resilient to node failure	No guarantee on locating objects; prone to excessive message overhead which can affect scalability

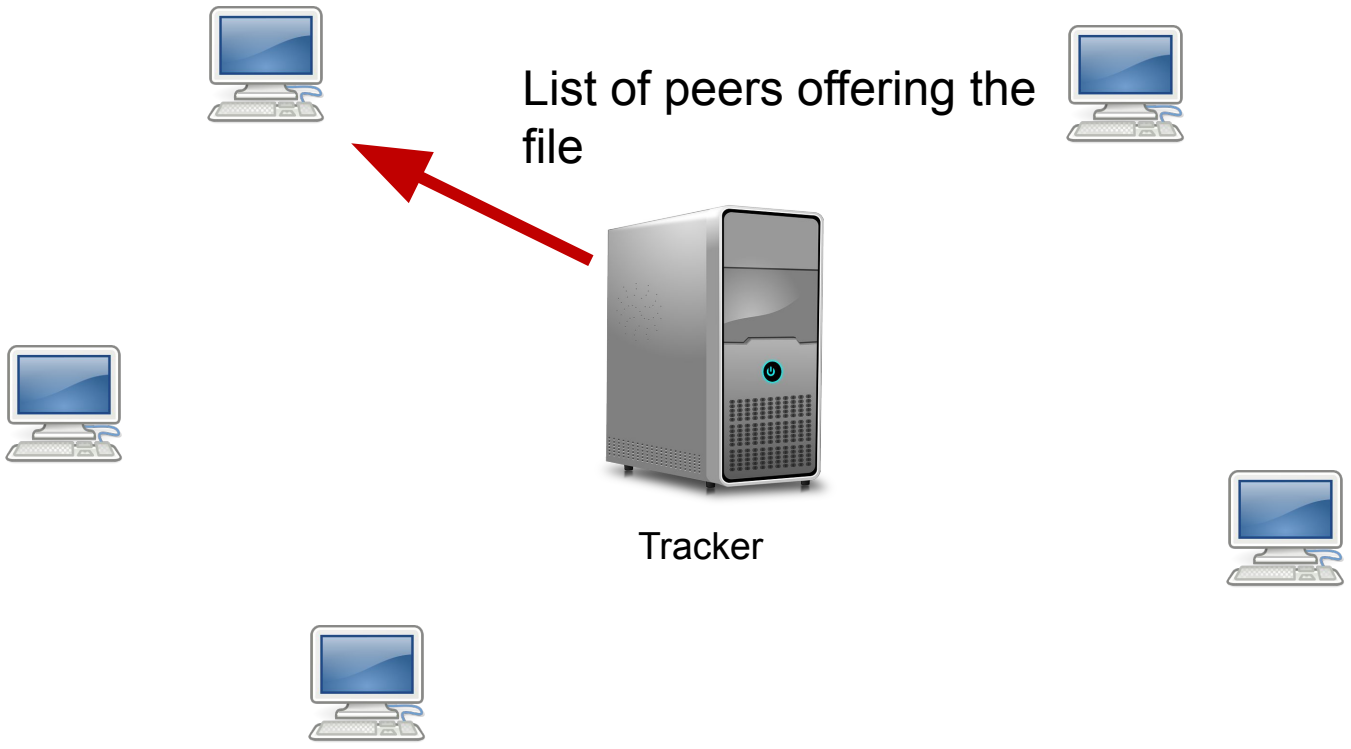
Hybrid architecture - BitTorrent

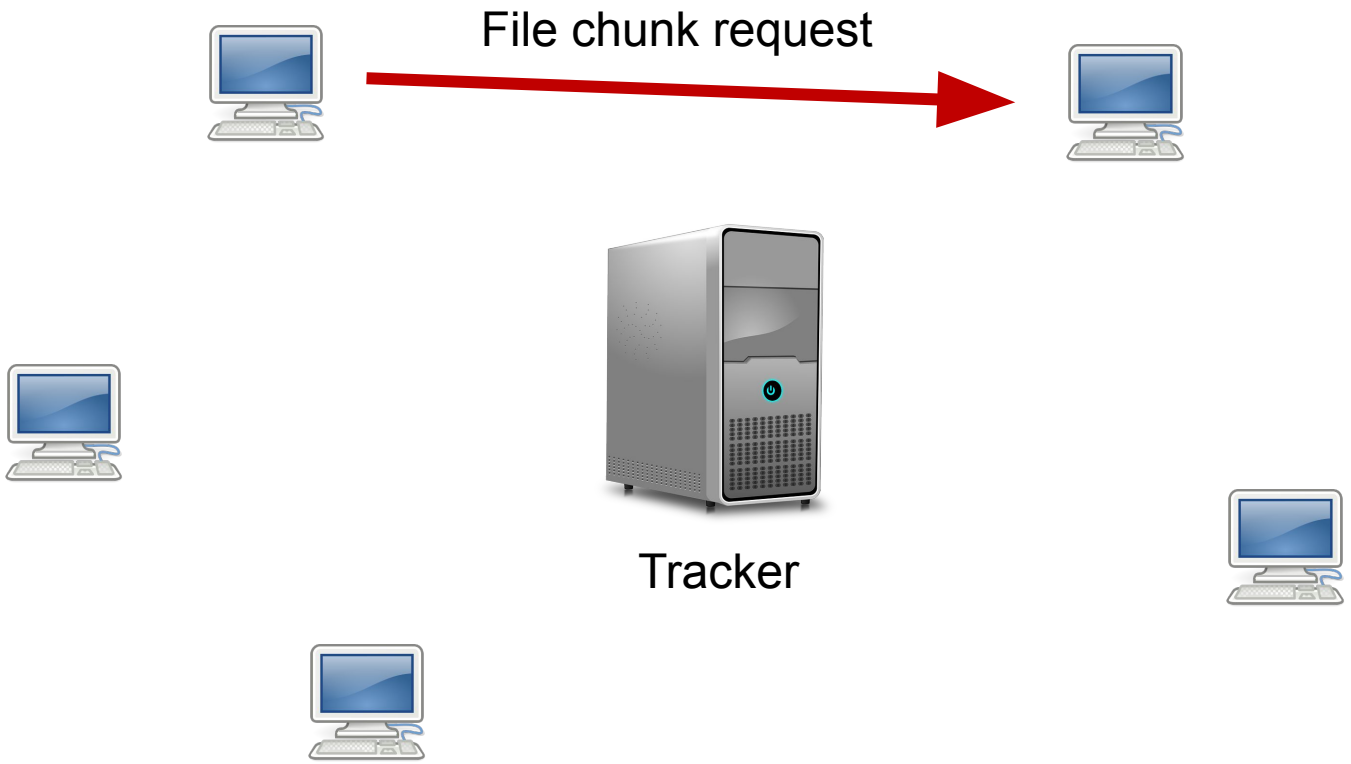
- User downloads chunks of a file from other users and assemble together yielding the complete file.
- To get started a traditional client-server scheme is deployed.
- A tracker is a server that keeps an accurate account of active nodes that have (chunks of) the requested file.

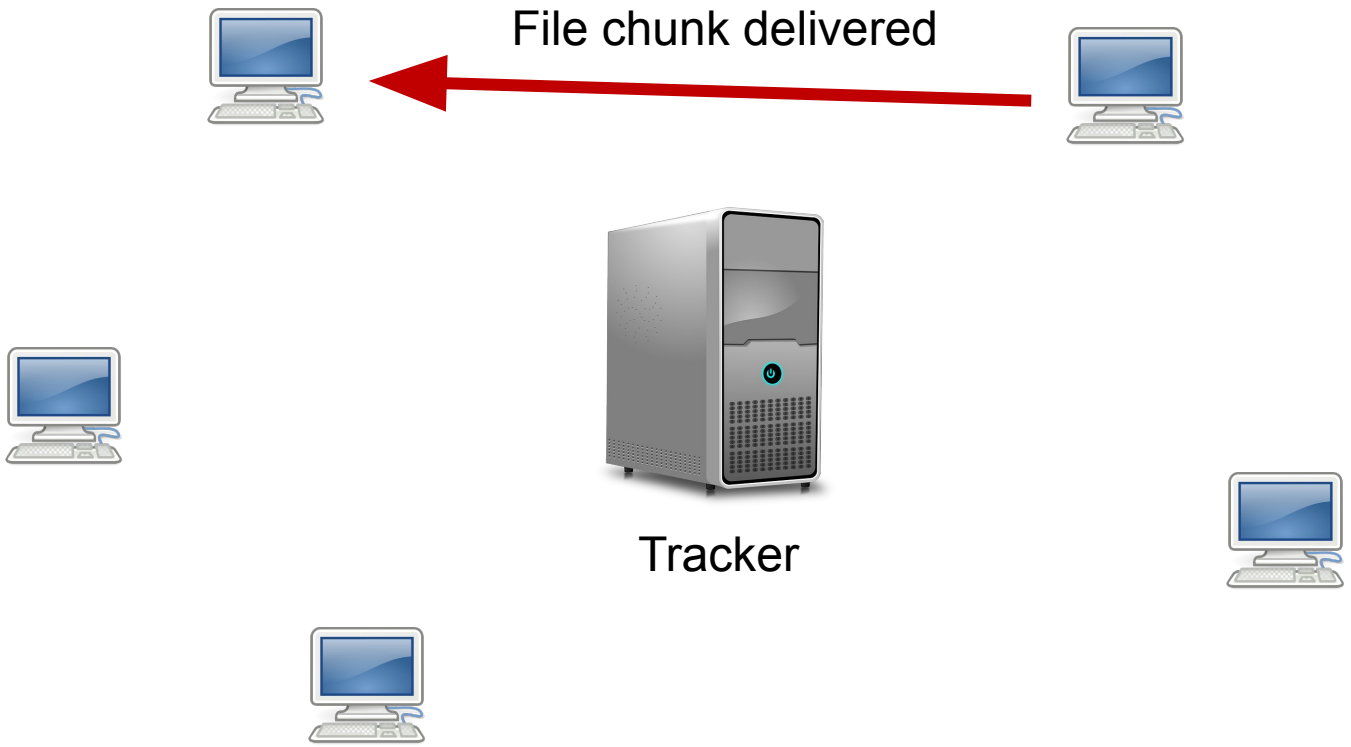


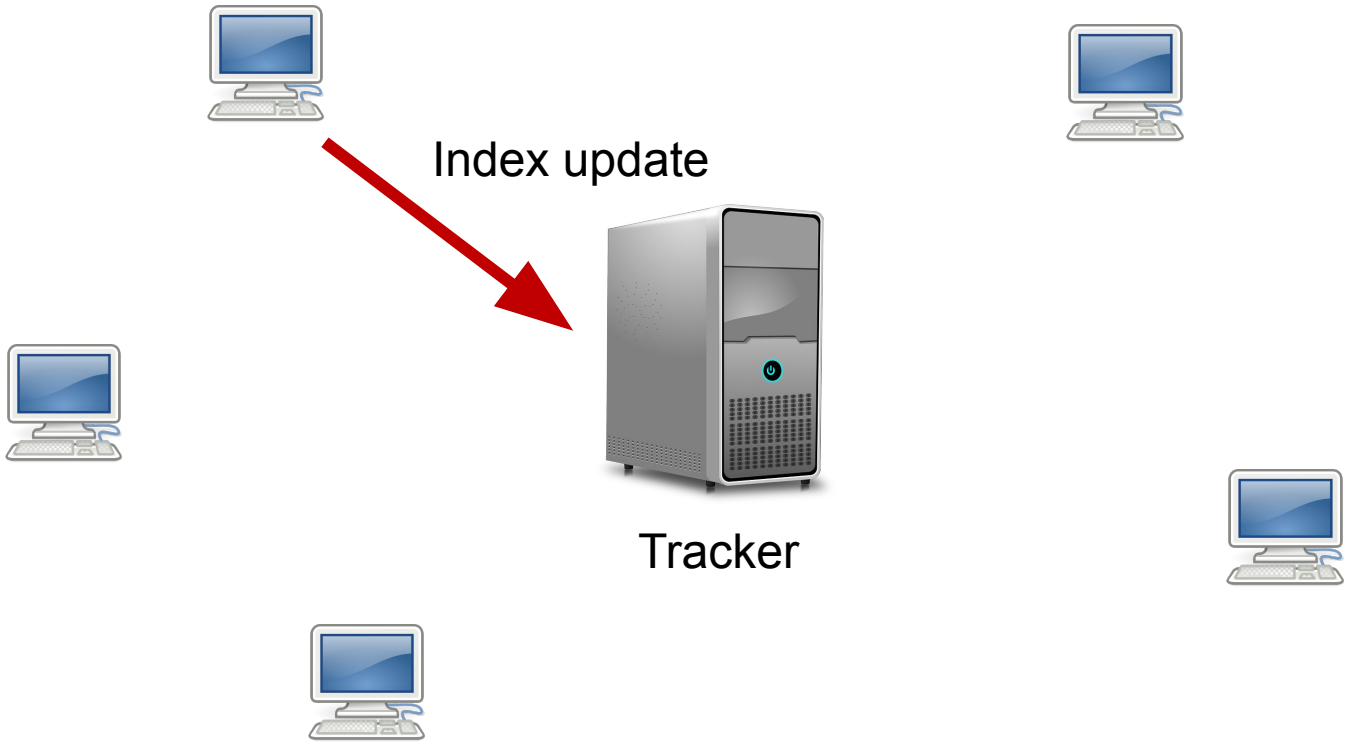
The principal working of BitTorrent; torrent file contains a link to a tracker.

Tracker


List of peers offering the file
Tracker

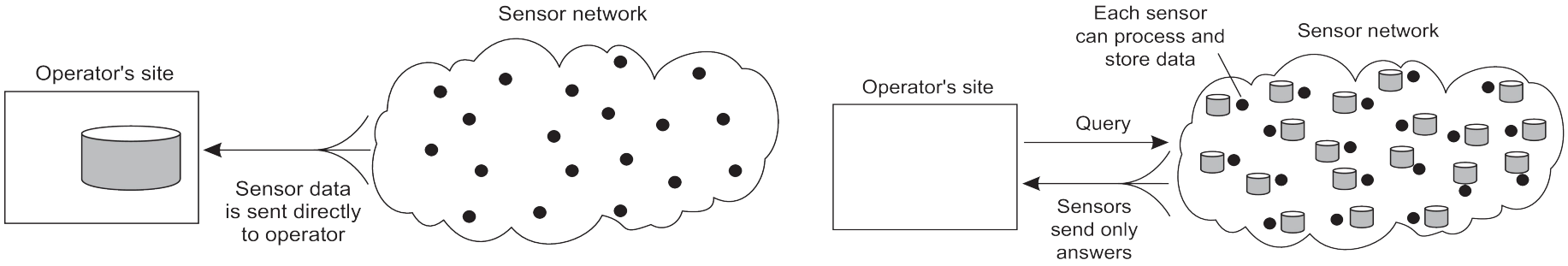

File chunk request
Tracker


File chunk delivered
Tracker


Index update
Tracker

Edge computing

- Computation is performed by devices at the “edge” of the network.
- Reduces the amount of centralized computation.



Organising a sensor network database, while storing and processing data (a) only at the operator’s site or (b) only at the sensors.