Dependable Distributed Systems Master of Science in Engineering in Computer Science

AA 2022/2023

LECTURE 25: OVERLAY NETWORKS

Peer to Peer (P2P)

P2P systems comprise **self-organized** equal and autonomous entities (**peers**) aiming to **share distributed resources** in a given network

Peers are **both client and servers, each node carries out the same tasks**

Local knowledge: nodes only know a small set of other nodes

Decentralization: nodes must self-organize in a decentralized way

Robustness: several nodes may fail with little or no impact

High scalability: **high aggregate capacity, load distribution**

Low-cost: storage and bandwidth are contributed by users

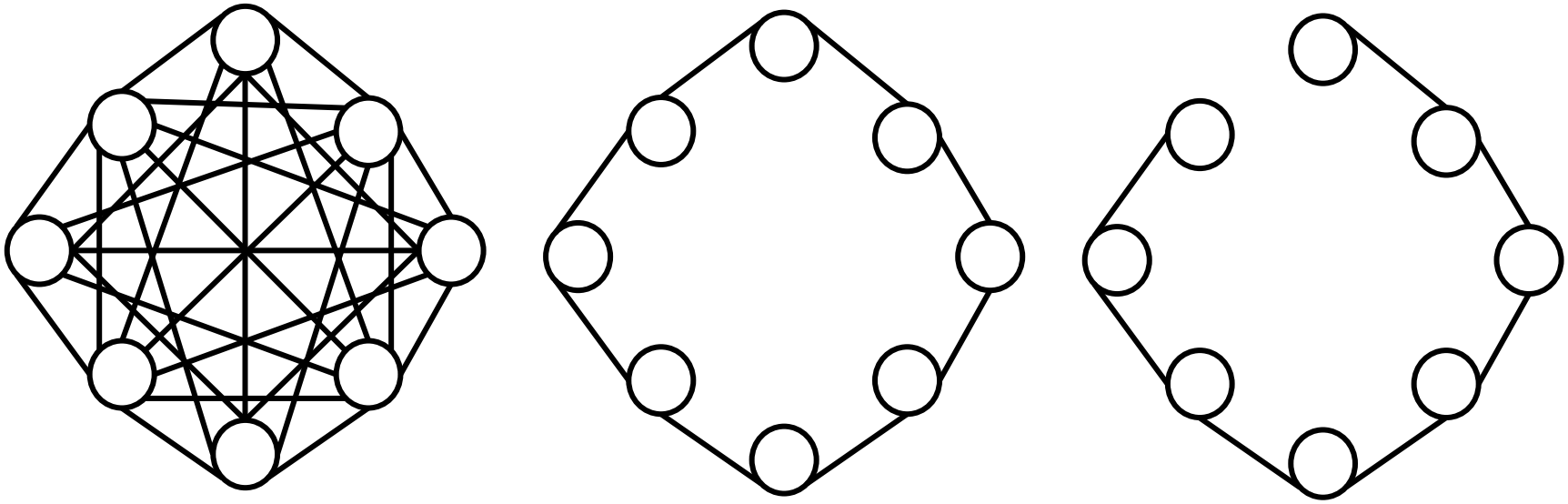
Peer to Peer (P2P)

P2P systems attempt to provide a long (not exhaustive) **list of features**:

- Selection of nearby peers
- redundant storage
- efficient search/location of data items
- data permanence or guarantees
- hierarchical naming
- trust and authentication
- anonymity
- …

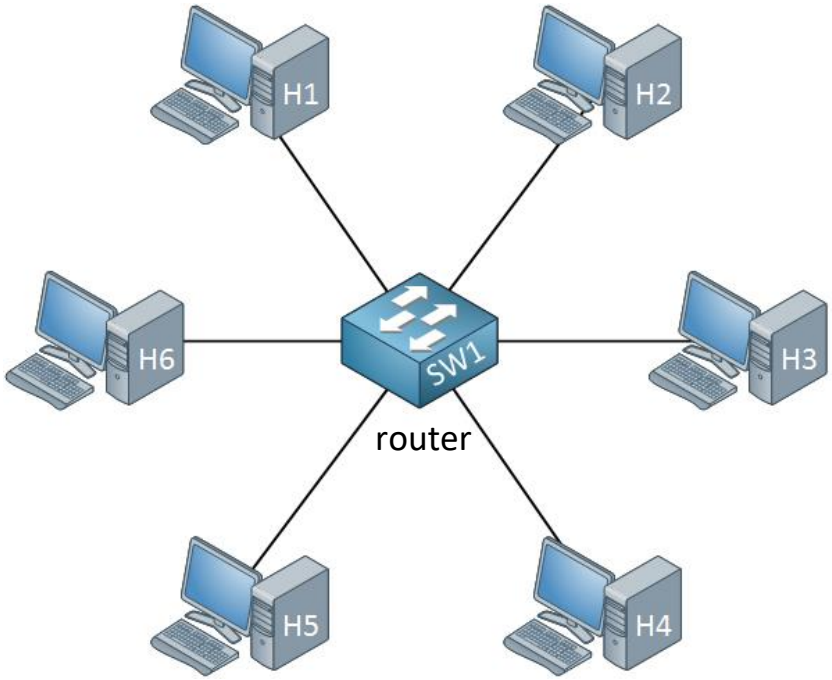
P2P networks potentially offer an **efficient routing architecture** that is **self-organizing, massively scalable, and robust** in the wide-area, **combining fault tolerance, load balancing, and explicit notion of locality**

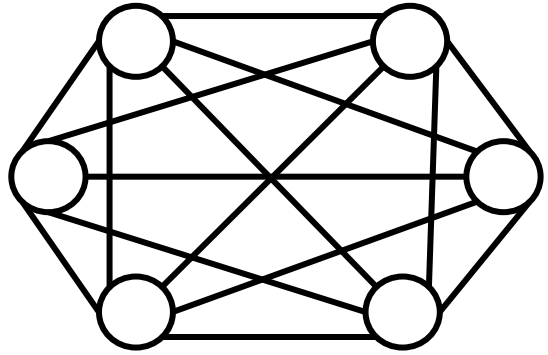
Topology



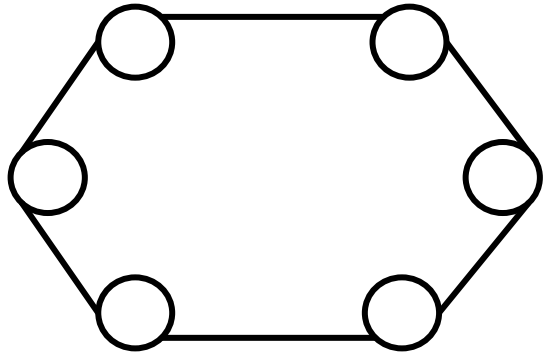
Links: model the capability of a pair of processes in exchanging messages

Topology (behind)





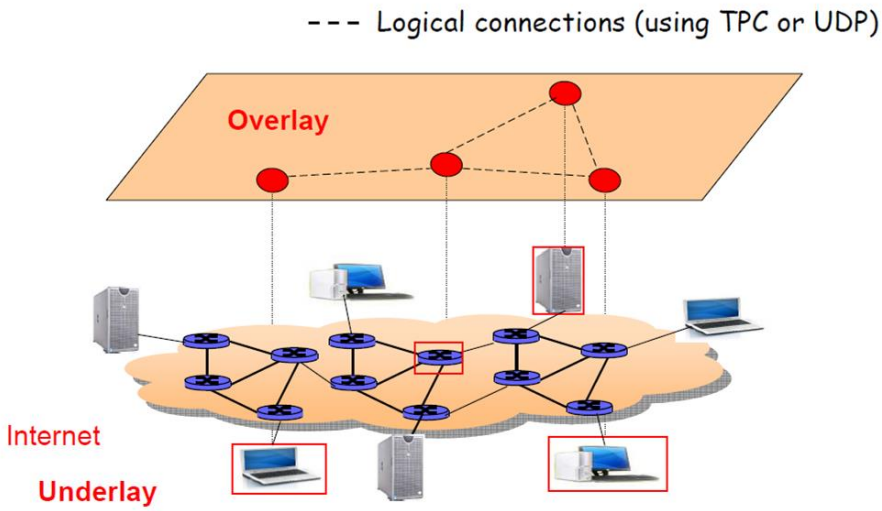
If all machines can exchange messages among them

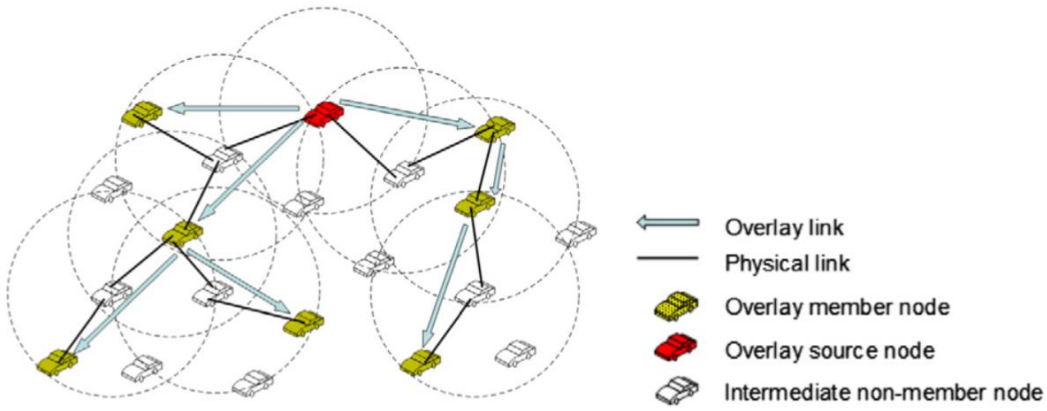


If every machine i can exchange messages only with i%6 ± 1

Overlay Network

A logical structure over a physical network





Overlay Network

An overlay network is a network that is **built on top of an existing network**.

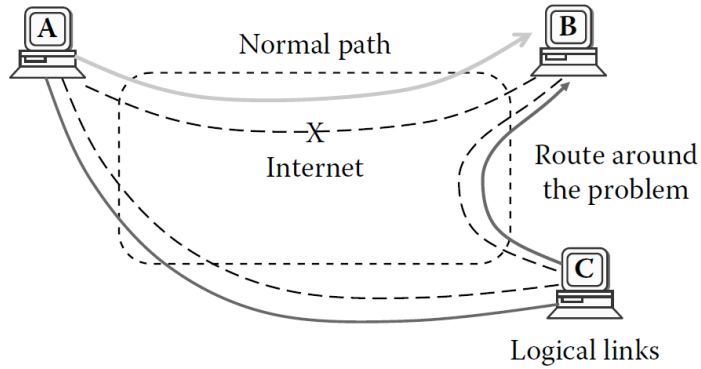
The overlay therefore **relies on the so-called underlay network for** basic networking functions, namely **routing and forwarding**

Today, most **overlay networks are built in the application layer** on top of the TCP/IP networking suite.

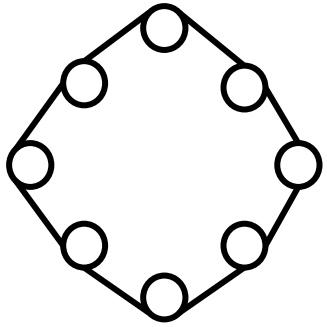
The nodes in an overlay network are connected via logical links that can span many physical links. **A link between two overlay nodes may take several hops in the underlying network**.

Why set up an overlay?

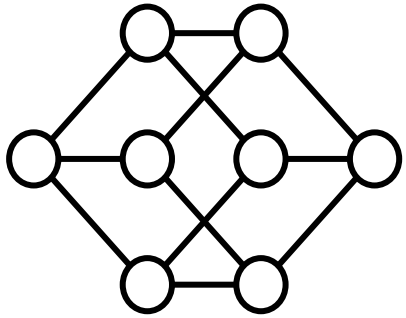
- **Performance** and/or **scalability**
- **Induced**: processes may not know all the peers of the system (**membership**), or **reachability** issues
- **Incremental deployment**: do not require changes to the existing routers, can grow node by node
- **Adaptable**: The overlay algorithm can utilize a number of metrics when making routing and forwarding decisions. Thus the overlay can take application-specific concerns into account
- **Robust**: to node and network failures due to its adaptable nature; with a sufficient number of nodes in the overlay, the network may be able to offer multiple independent (router-disjoint) paths to the same destination. At best, overlay networks are able to route around faults.



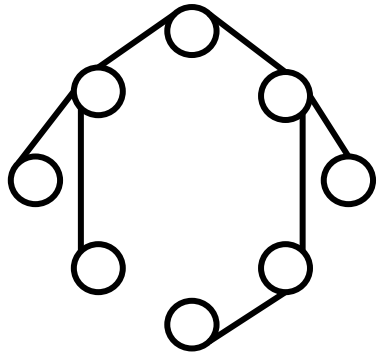
Basic Graph Metrics



Distance(a,b): length of the shortest path between nodes a and b



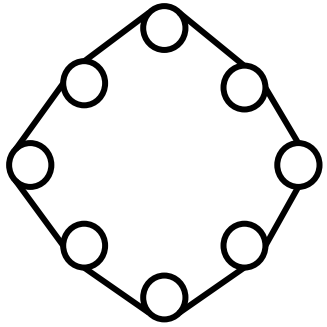
Diameter: max among all distances



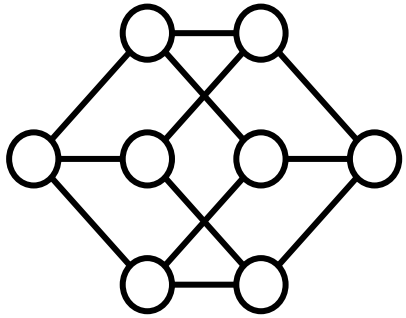
Closeness centrality: the average length of the shortest path between the node and all other nodes in the graph

Betweenness centrality: how important a node is to the shortest paths through the network

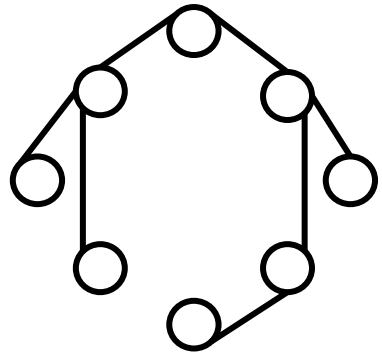
Basic Graph Metrics



Connected: it exists a path between every two nodes

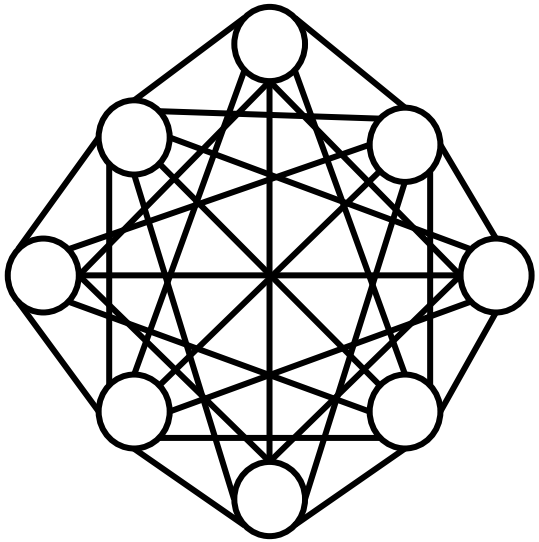


Edge-Connectivity: minimum number of *edges* that has to be removed to disconnect the network

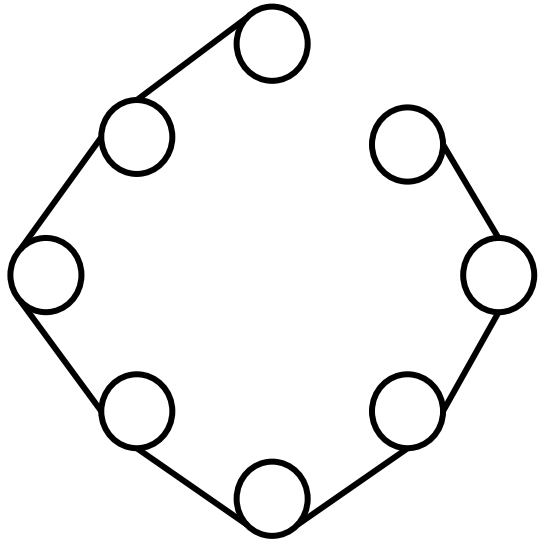


Node-Connectivity: minimum number of *nodes* that has to be removed to disconnect the network

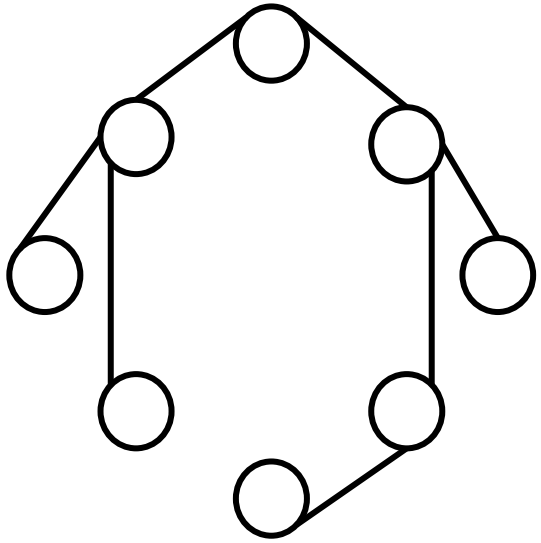
Example (BEB - Performance)



Load: the source has to send n messages
Latency: O(1) hop

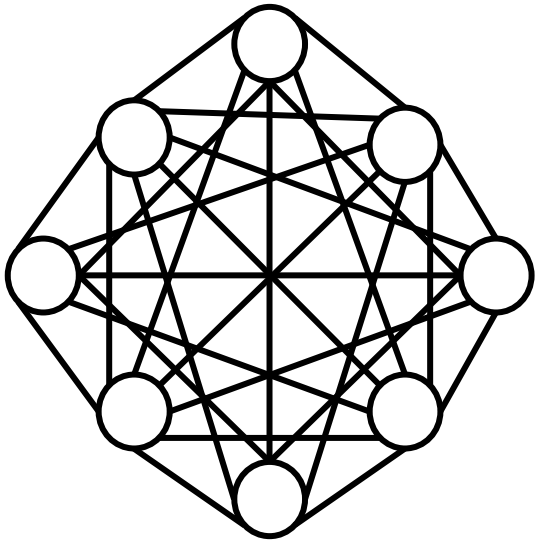


Load: the source has to send 1 message
Latency: O(n) hops

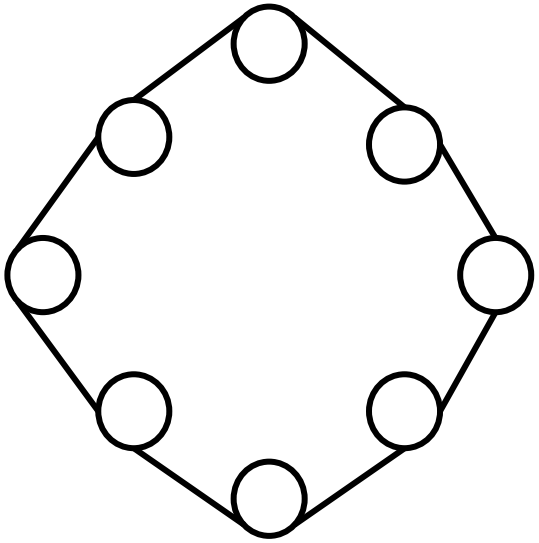


Load: the source has to send 3 messages
Latency: $\simeq$ O(log2(n)) hops

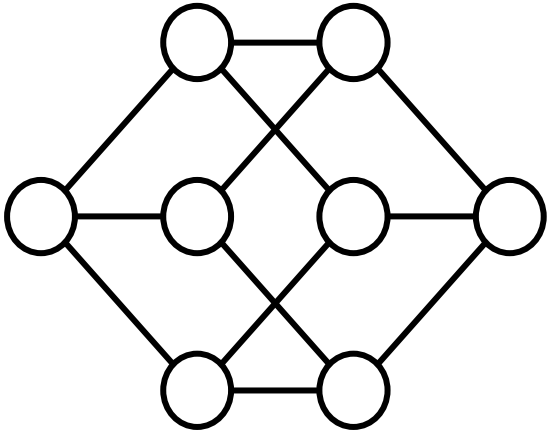
Example (BEB - Fault Tolerance)



Correctness: n-1 crashes



Correctness: 1 crash (worst case)



Correctness: 2 crashes (worst case)

Dynamic Distributed Systems

Informally, it is a distributed systems that **evolves over time**

The are several ways in which a distributed system may evolve,

we consider two types of evolutions:

- **Set of links** (i.e. the available links between the processes change over time)
- **Set of processes** (i.e. the actual processes in the system change over time)

In an actual distributed system **they may both change**

Overlay network protocols allows to preserve correctness of distributed protocols despite the evolution of the system

Note: the evolution of a distributed system can either be intentional or due to faults

Dynamic Distributed Systems Main Classes

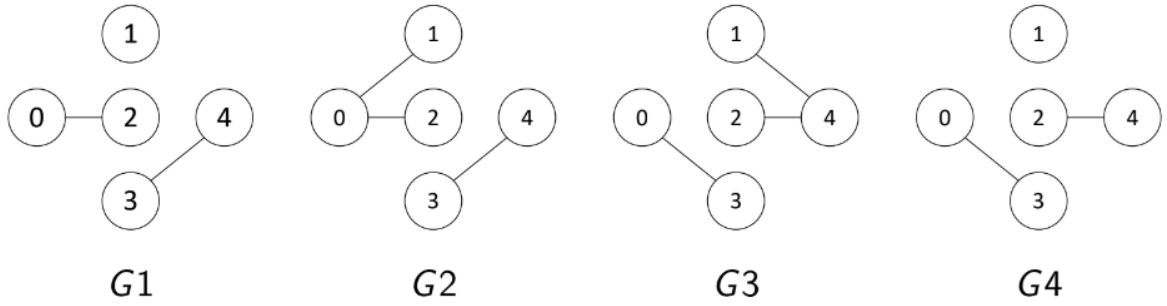
Dynamicity of the processes:

Reconfiguration: a protocol manages join and leaves (request to join, request to leave, reconfiguration of the system)

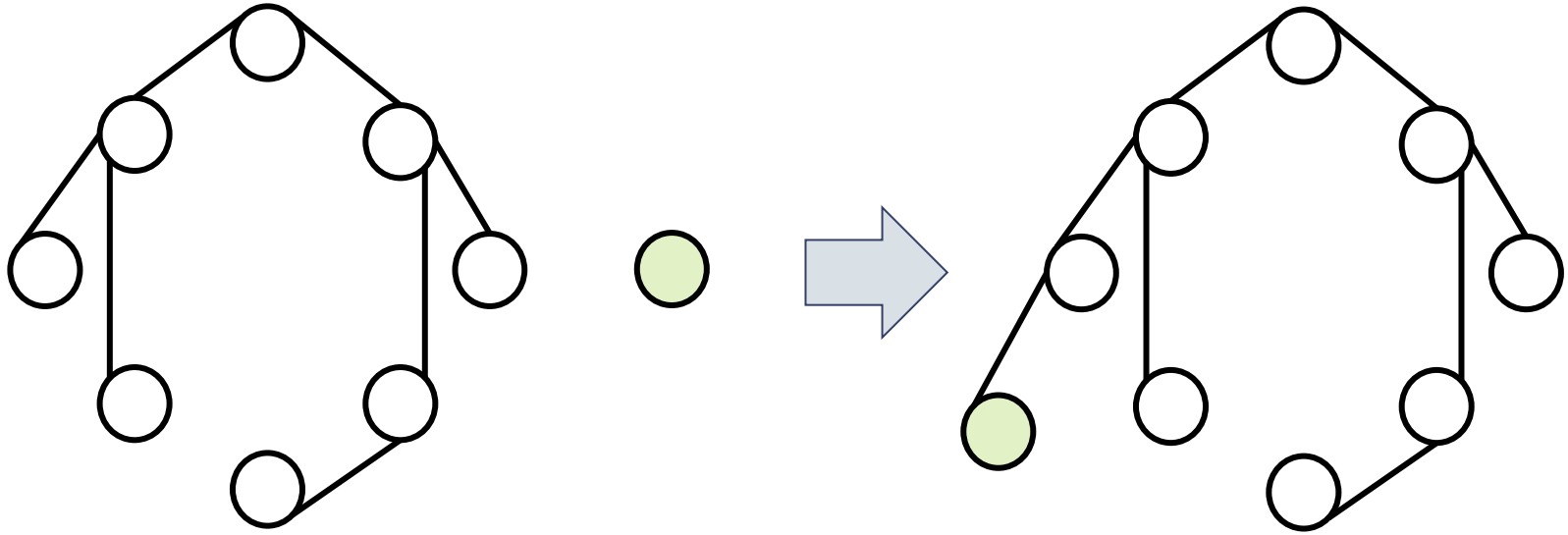
Churn: joins and leaves are unmanaged, the churn is characterized by a patten (e.g. churn rate)

Dynamicity of the links:

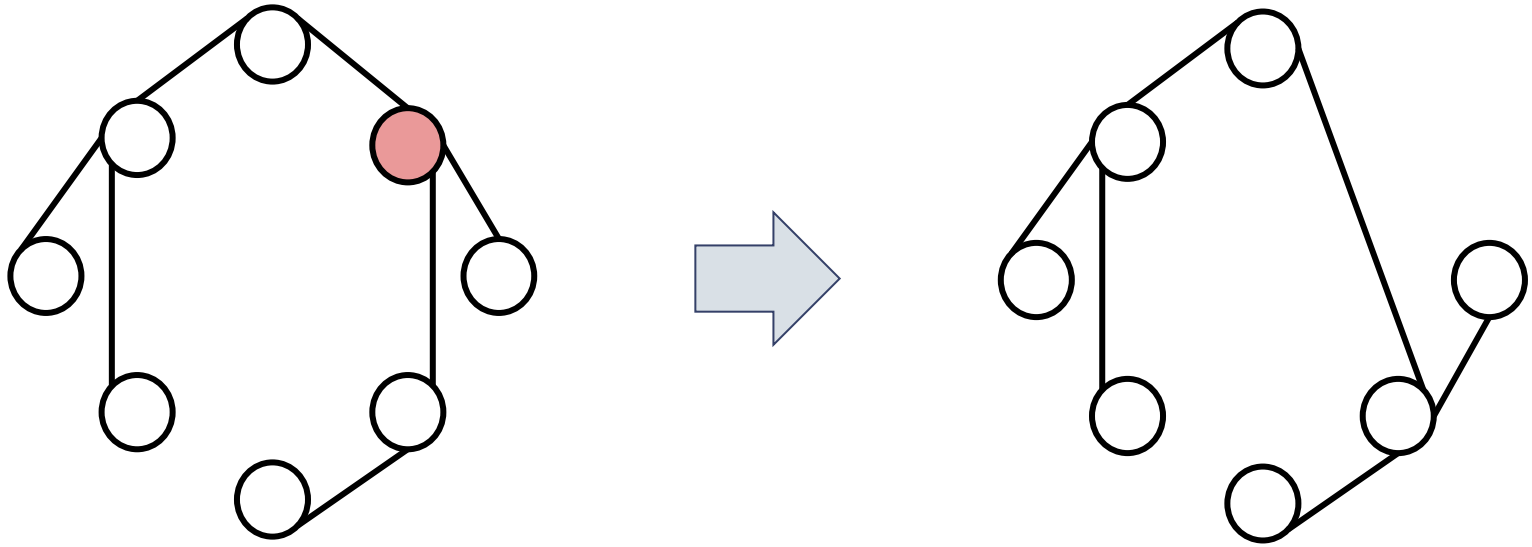
Time-varying graphs, general network features, …



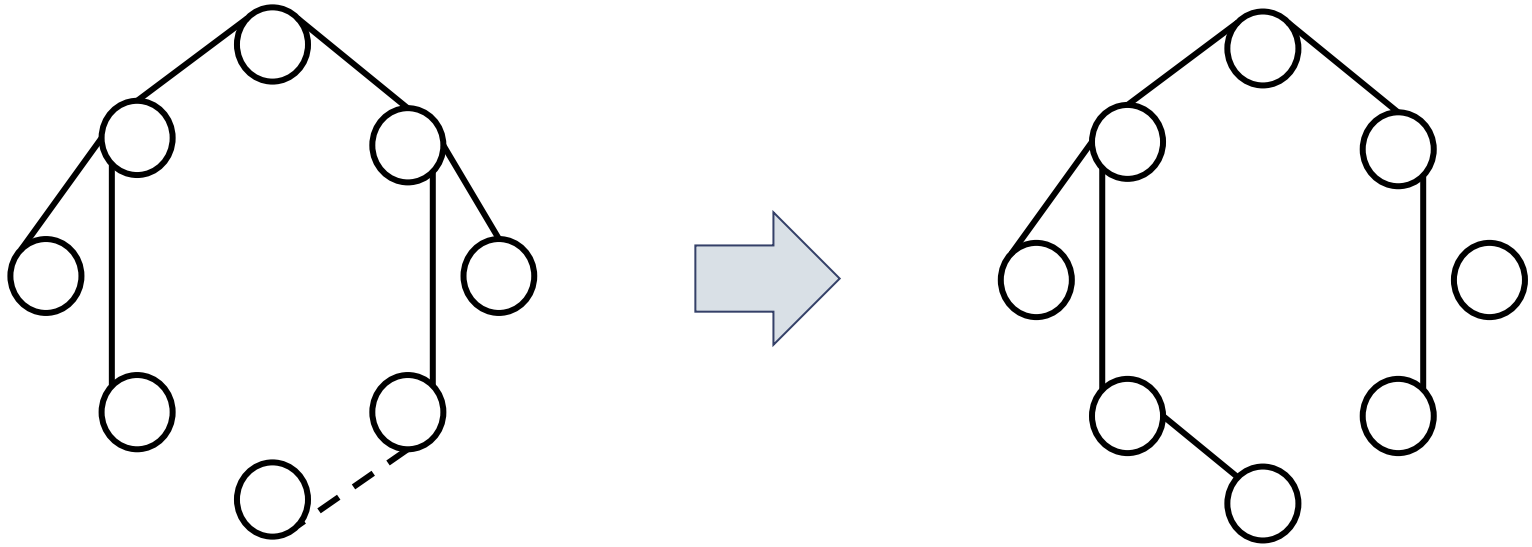
What an overlay network protocol does (reconfiguration)



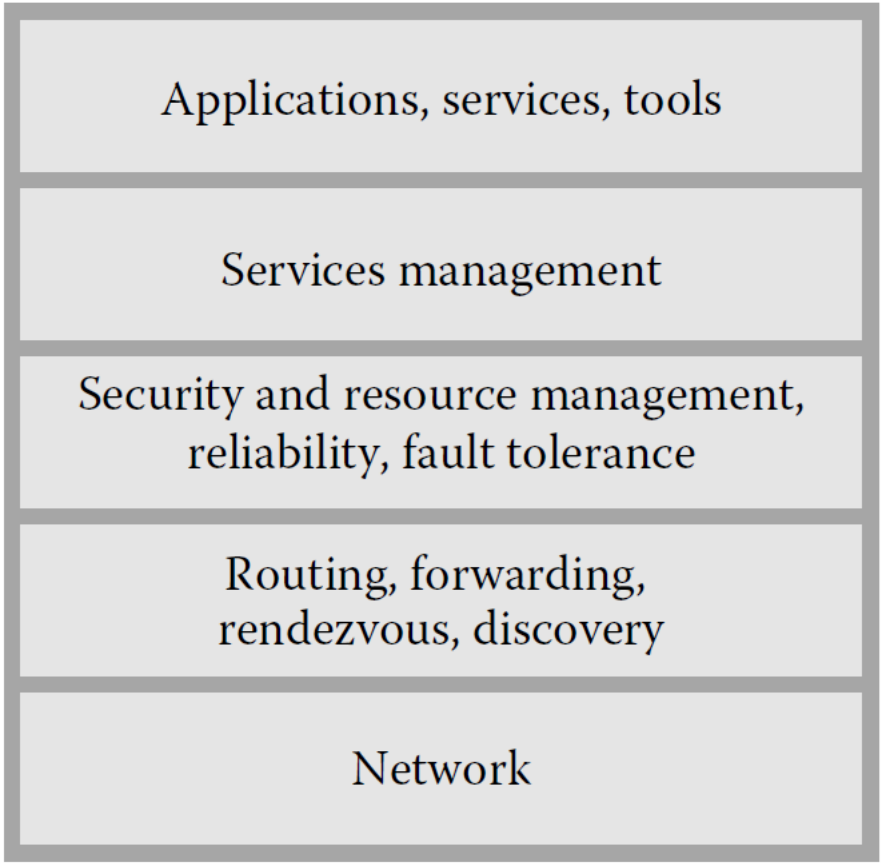
What an overlay network protocol does (reconfiguration)



What an overlay network protocol does (reconfiguration)

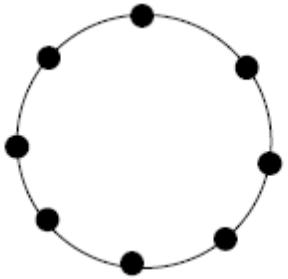


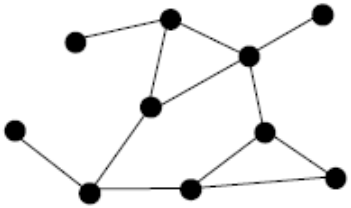
What an overlay network protocol does

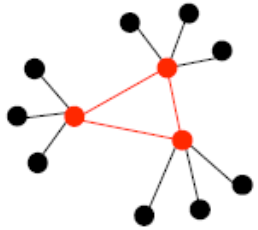


Classes of Overlay Networks

- **Structured**
  - nodes are arranged in a **restricted structure** (ring, tree, etc.)
- **Unstructured**
  - the topology results from **some loose rules**, without any prior knowledge of the topology
- **Hybrid and/or multi-level**:
  - supernodes form a small overlay







Classes of Overlay Networks: Features

- **Structured**
  - efficient data location
  - long join and leave procedures
  - less robust in high-churn environments
- **Unstructured**
  - fast join procedure
  - Usually very tolerant to churn
  - good for data dissemination, bad for location
  - support more complex queries
- **Hybrid and/or multi-level**:
  - Custom features
  - Usually large state and high load on supernodes

Overlay Networks: Applications

- **Data/file sharing**: Bittorrent, Bitcoin
- **CDN**: Content caching to reduce delay and cost
- **Routing and forwarding**: Reduce routing delays and cost, resiliency, flexibility
- **Security**: To enhance end-user security and offer privacy protection. For example, virtual private networks (VPNs), onion routing, anonymous content storage, censorship resistant overlays.
- Other: publish/subscribe, etc.

DHT

Distributed hash tables are a class of decentralized distributed algorithms that **offer a lookup service**

DHTs **store (key, value) pairs**, and they support the **lookup** of the value associated with a given key

The **keys and values are distributed** in the system, and the DHT system must ensure that the nodes have sufficient information of the global state to be able to forward and process lookup requests properly

Data object (or value) location information is placed deterministically, at the peers with identifiers corresponding to the data object's unique key.

Linear Hashing and LH*: support table expansion

DHT

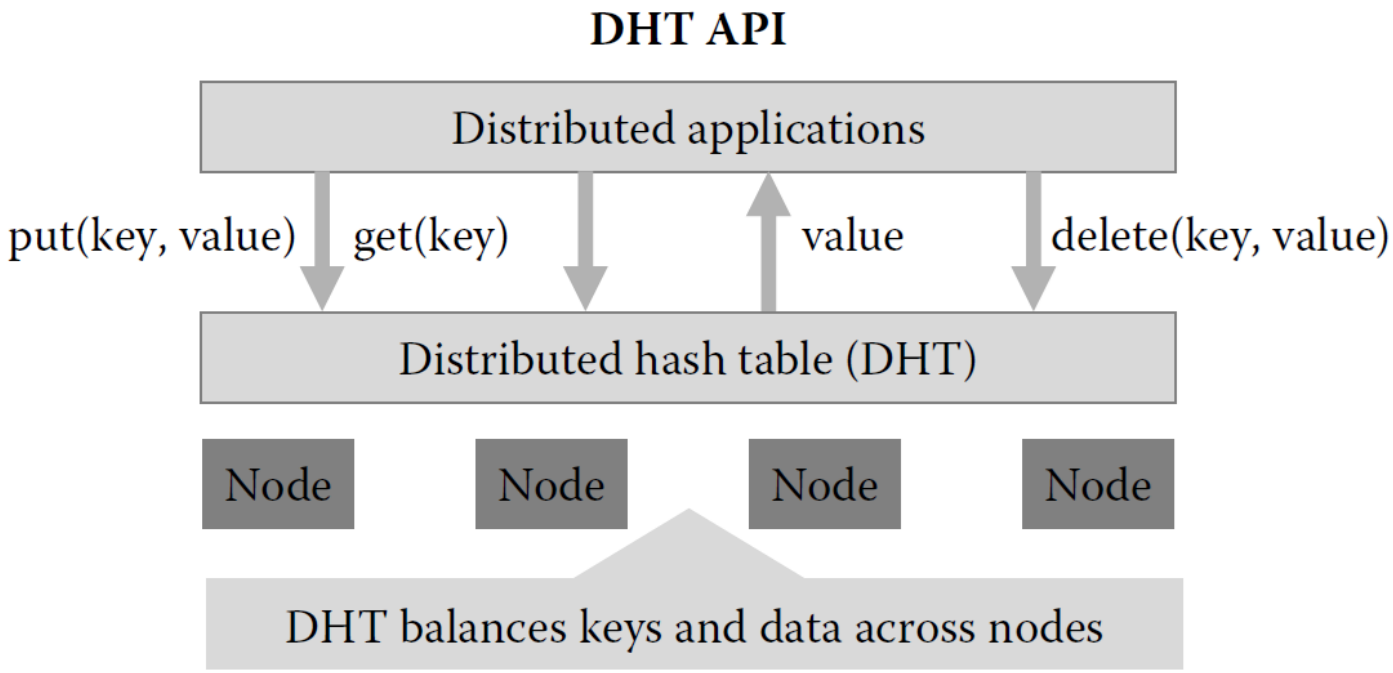
The key-based structured algorithms have a **desirable property: namely, that they can find data locations within a bounded number of overlay hops**

The **DHT algorithm is responsible for distributing the keys and values** in such a way that efficient lookup of the value corresponding to a key becomes possible.

Since peer nodes may come and go, this requires that **the algorithm be able to cope with changes in the distributed system**. In addition, the locality of data plays an important part in all overlays, since they are executed on top of an existing network, typically the Internet

The overlay should **take the network locations of the peers into account** when deciding where data is stored, and where messages are sent, in order to **minimize networking overhead**

DHT


DHT API
Distributed applications
put(key, value)
get(key)
value
delete(key, value)
Distributed hash table (DHT)
Node
Node
Node
Node
DHT balances keys and data across nodes

Retrieve data in a P2P system

- Central server (e.g. Napster)
- Flooding Search (e.g. Gnutella)
- Distributed Indexing (e.g. Chord)

Locating Data

= Routing

Structured Networks:

- low hop count for large network sizes
- each object must be tracked by a different node
- the overlay must be structured according to a given topology in order to achieve a low hop count
- routing tables must be updated every time a node joins or leaves the overlay

Locating Data

Unstructured Networks:

- Flooding or Gossip: send a message to all nodes
  - simple, no topology constraint
  - high network overhead (huge traffic generated by each search request)
  - flooding stopped by TTL
  - only applicable to small number of nodes
  - it may fail
- Random walk, Expanding-ring time-to-live, etc.
- Gossip characterized mainly by maximum number of hop and the fan-out

Performance in P2P Networks

In general, what **impact most the performance** metrics of P2P applications are:

The **topology** of the overlay network

The **routing** protocol

Bootstrapping

Bootstrapping basically describes the process of i**ntegrating a new node into a P2P system**

- Dedicated bootstrap servers
- Local host cache
- Random Address Probing

Blockchain Networks

Blockchains in general and cryptocurrencies such as Bitcoin **typically run on top of a P2P network**

In permissionless protocols, such as Bitcoin and Ethereum, **any machine can join the network** and become a node of the P2P network.

A node bootstraps its operation with a discovery protocol to establish connections with other nodes in the system

Transactions and blocks are typically propagated in the network using a flooding or gossip protocol

Many cryptocurrency networks are characterized by frequent broadcast operations

Knowledge of the network topology can give parties an advantage in the dispersal of information (blocks, transactions), which **can lead to security risks**

Bitcoin Network

Bitcoin and Ethereum rely on flat **random graph** topologies

The Bitcoin P2P network topology is formed by *each peer connecting to 8 nodes (outbound connections) and accepting up to 125 incoming connections*

Outbound destinations are randomly selected among known identities

Before being able to send and receive protocol messages a node has to find other nodes to connect to join the network

With the first messages exchanged between new peers, **they inform each other of a random subset of locally known addresses** with a timestamp of at most 3h ago

Bitcoin Network Bootstrap

In Bitcoin,

- a node **first tries to connect to nodes it knows from participating previously**.
- **If no connections can be established** this way, or if the node connects for the very first time, then it **queries a list of well-known DNS seeds**. The DNS seeds are maintained by Bitcoin community members
- As a last resource, it will try to connect to **hardcoded seed nodes**

Bitcoin Network

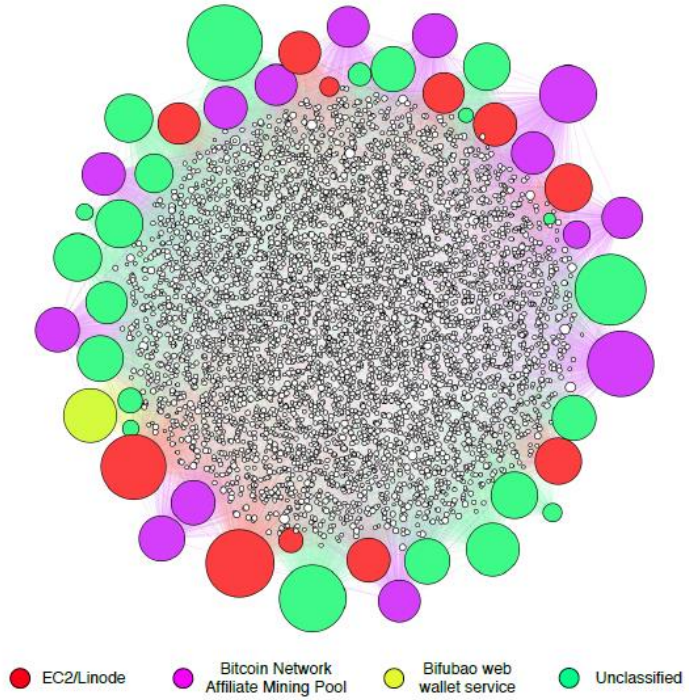
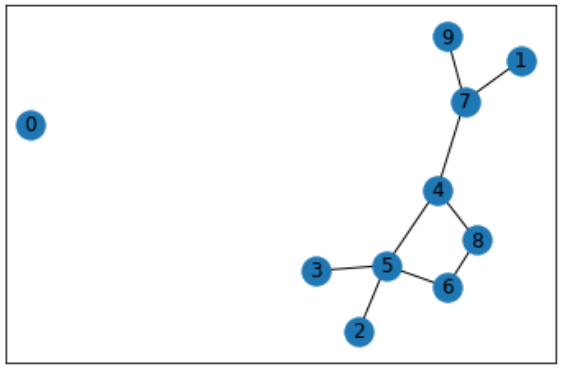


Figure 7: A snapshot of the (reachable) Bitcoin network discovered by AddressProbe on Nov. 5. The highest degree nodes (with degrees ranging 90–708) are colored.

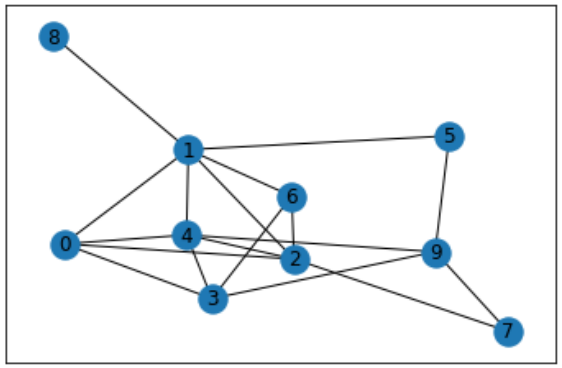
From Miller, Andrew, et al. "Discovering bitcoin's public topology and influential nodes." *et al* (2015).

Erdős-Rényi Random Graphs Model

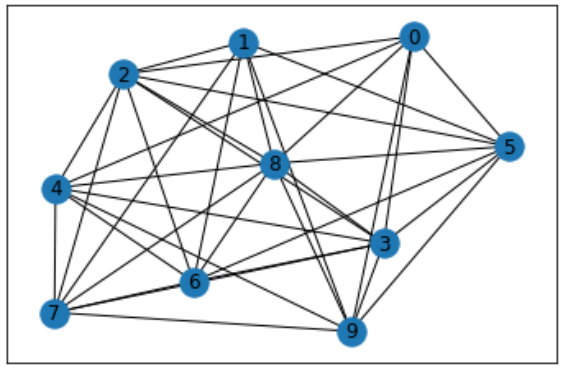
In the G(n,p) model, a graph is constructed by connecting labeled nodes randomly. Each edge is included in the graph with probability p, independently from every other edge.



G(10, 0.2)



G(10, 0.5)



G(10, 0.8)