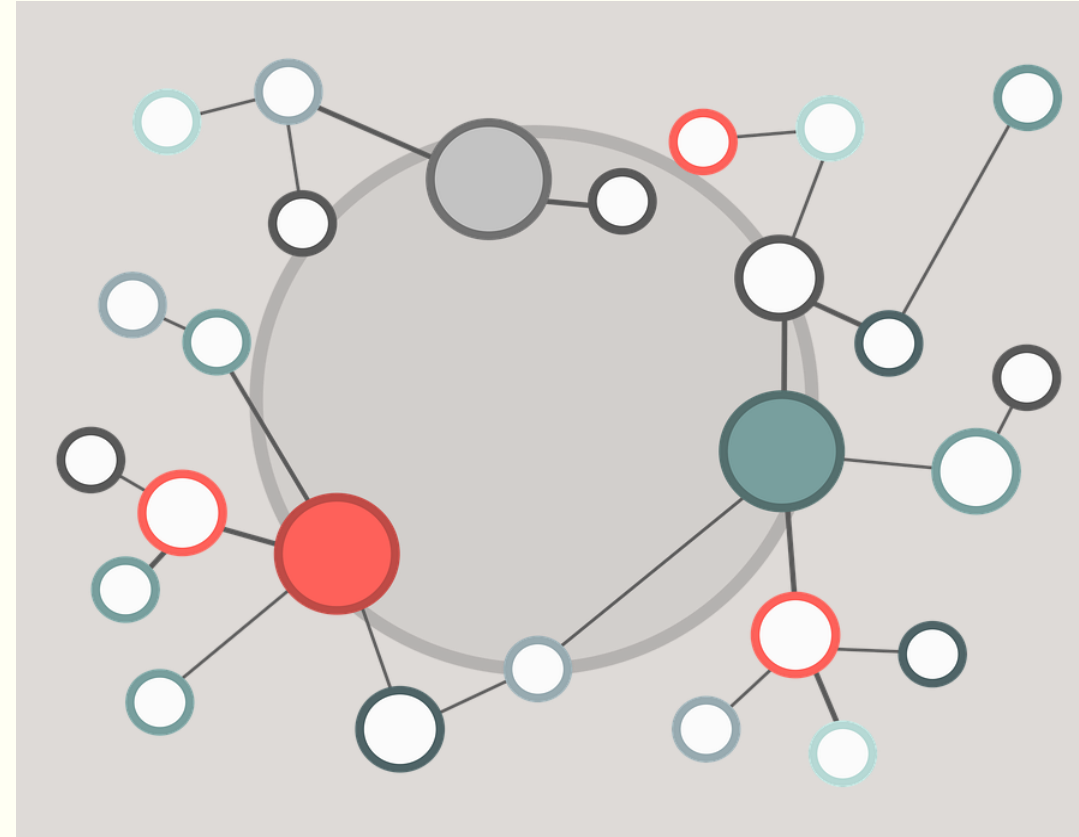


BLOCKCHAIN P2P NETWORK

Blockchain technologies, lecture 5



Course overview

- Block propagation
 - Topology
 - Discovery protocol
 - Broadcasting, messages
-
- Kademlia a peer to peer information system, RLPx
 - Wire protocol



BITCOIN TOPOLOGY

Bitcoin topology

- Nodes in the network form a random graph.
- Newly joined nodes query DNS servers.
- DNS servers return a random set of bootstrap nodes.
- A node learns about other nodes by listening from advertisements of new addresses coming from their neighbors.
- Each node keeps list of opened connections. Node randomly selects an address from a set of known addresses and attempts to establish a connection.
- Default number of connections: 8. Node's number of connections may exceed the default number due to incoming connections.

Bitcoin topology

- Each node tries to connect to peers using TCP (*outbound connections*).
- Default number of outbound connections: 8.
- A node stores IP addresses in two lists: **new** and **tried**.
 - **new** list. Addresses of peers to which the node has not yet tried to connect.
 - **tried** list. Addresses known as reachable
- A nodes accepts *inbound connections* from other peers.

Bitcoin topology

- A node accepts *inbound connections* from other peers.
- A listening node (a node that accepts inbound connections) may issue **ADDR** messages to advertise neighbors that it accepts inbound connections. Neighbors may relay ADDR message to their own neighbors by following a *gossip protocol*.
- A node may request to discover other active peers by sending a **GETADDR** message.
- A node periodically verifies the state of the nodes it is connected to by issuing a **PING** message and waiting for **PONG** responses.

Bitcoin topology

- **Full nodes** download every block and every transaction and verifies all consensus rules.
- **Miners** nodes extending the blockchain by creating new blocks.
 - Miners may work alone or in mining pool with an administrator running a full node.
- **Lightweight nodes** download only block headers and relies on full nodes.
 - Full nodes serve lightweight clients by notifying them when a transaction affects their wallet and transmitting transactions to the network.
- **DNS seeder** server that responds to DNS query by initiating a message that contains a list of IPS. Six DNS seeds periodically crawl the network to obtain active IP addresses
 - DNS seeders are queried by new nodes
 - DNS seeders are queried by a node that restarts and tries to reconnect to new peers.

Bitcoin topology

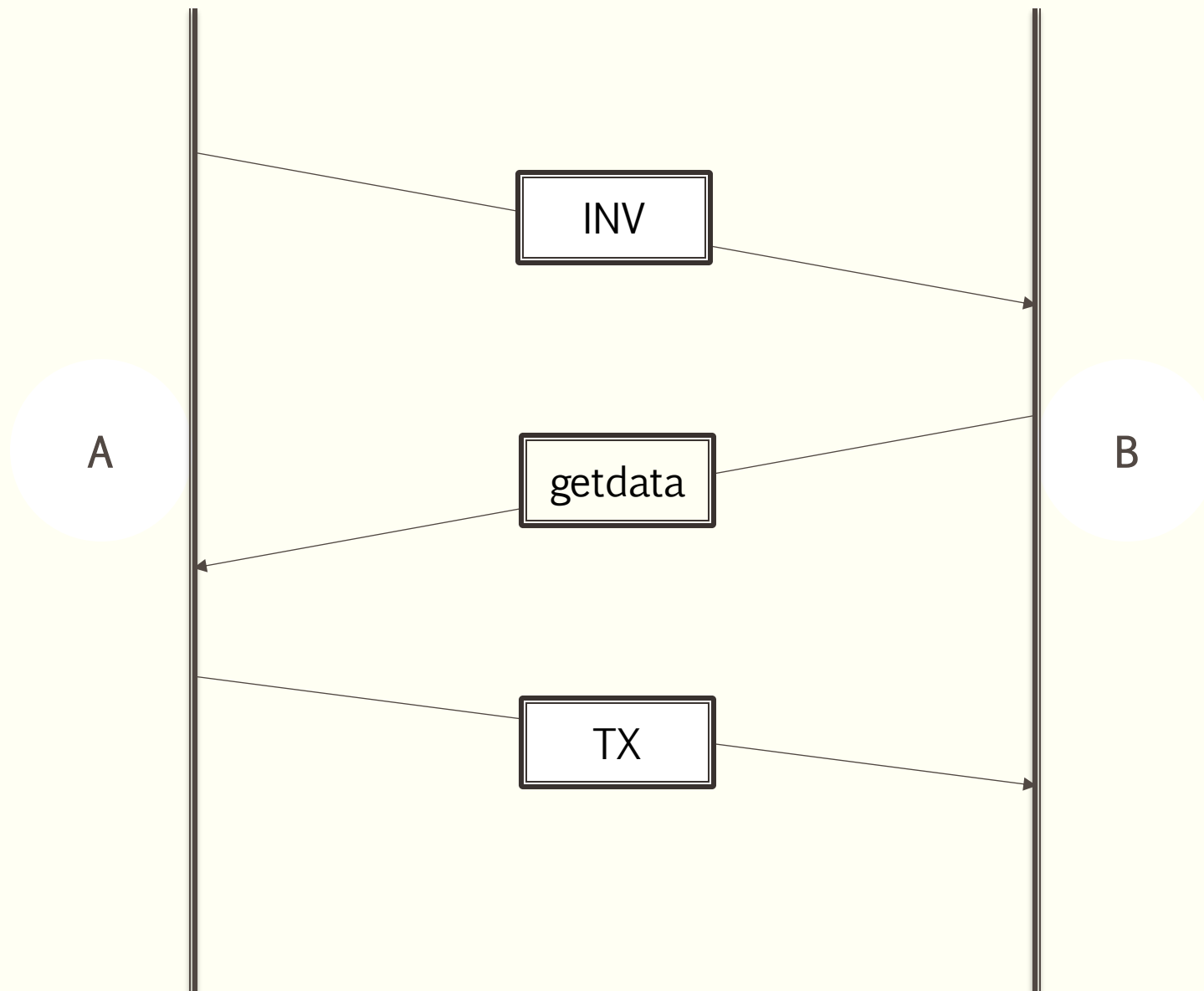
- **DNS seeder** server that responds to DNS query by initiating a message that contains a list of IPS. Six DNS seeds periodically crawl the network to obtain active IP addresses
 - DNS seeders are queried by new nodes
 - DNS seeders are queried by a node that restarts and tries to reconnect to new peers.
- DNS servers are hard-coded as trusted DNS servers maintained by the core developers.
- **SPAM score.** Each node scores peers, higher scores are assigned if a peer act as malicious node. Node stops sending messages to a peer that accumulated 100 points, for a period of 24h.



BITCOIN NETWORK MESSAGES

Updating and synchronization

- There are two types of transactions that update the distributed ledger replicas:
tx messages and *block* messages.
- *tx* and *block* messages are advertised with *inv* messages.
 - *inv* message contains a set of transaction hashes and block headers received by the sender.
 - *getdata* message is issued by the receiver of a *inv* message to the sender for a transaction or a block
- The propagation delay is the sum of transmission time and the local verification time of the block or transaction.

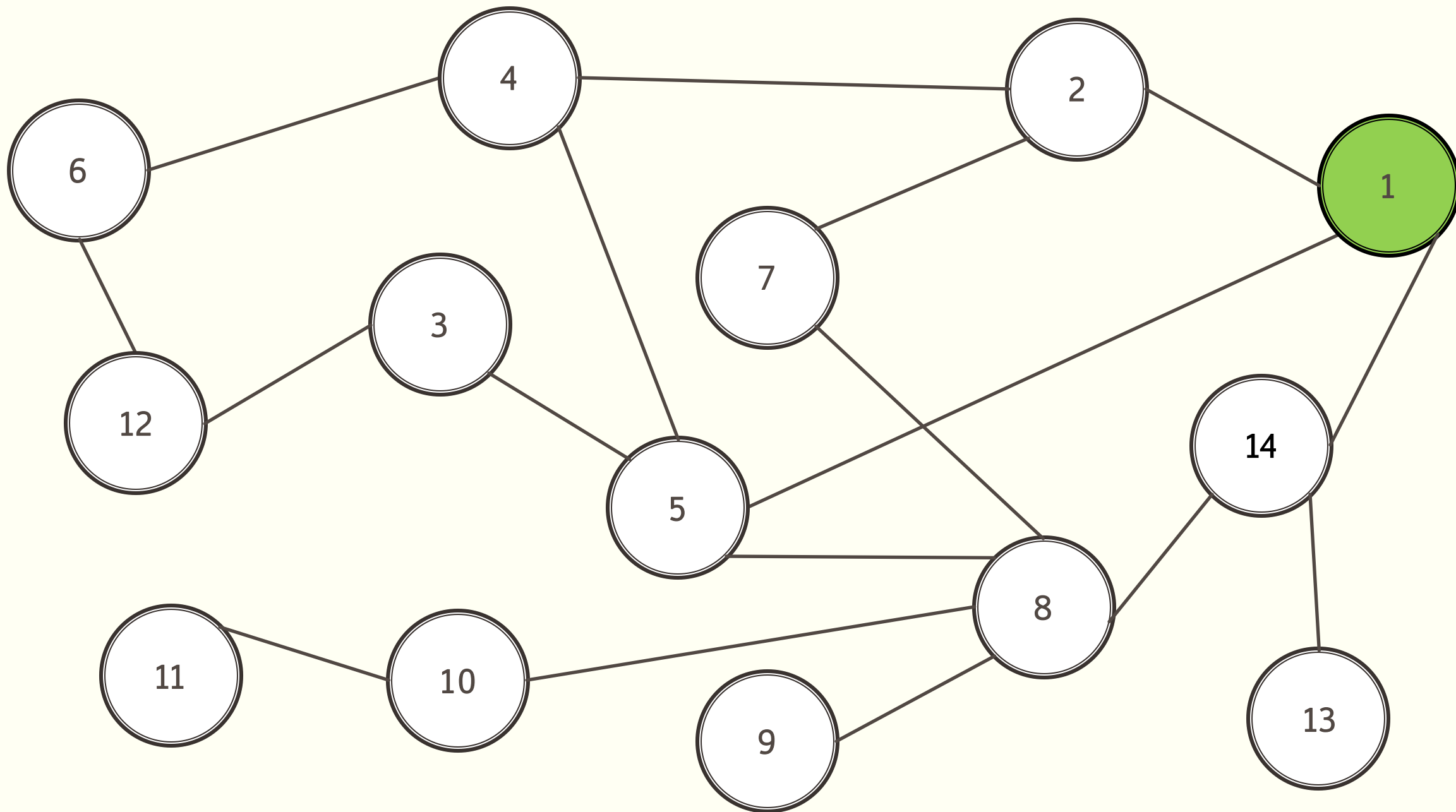


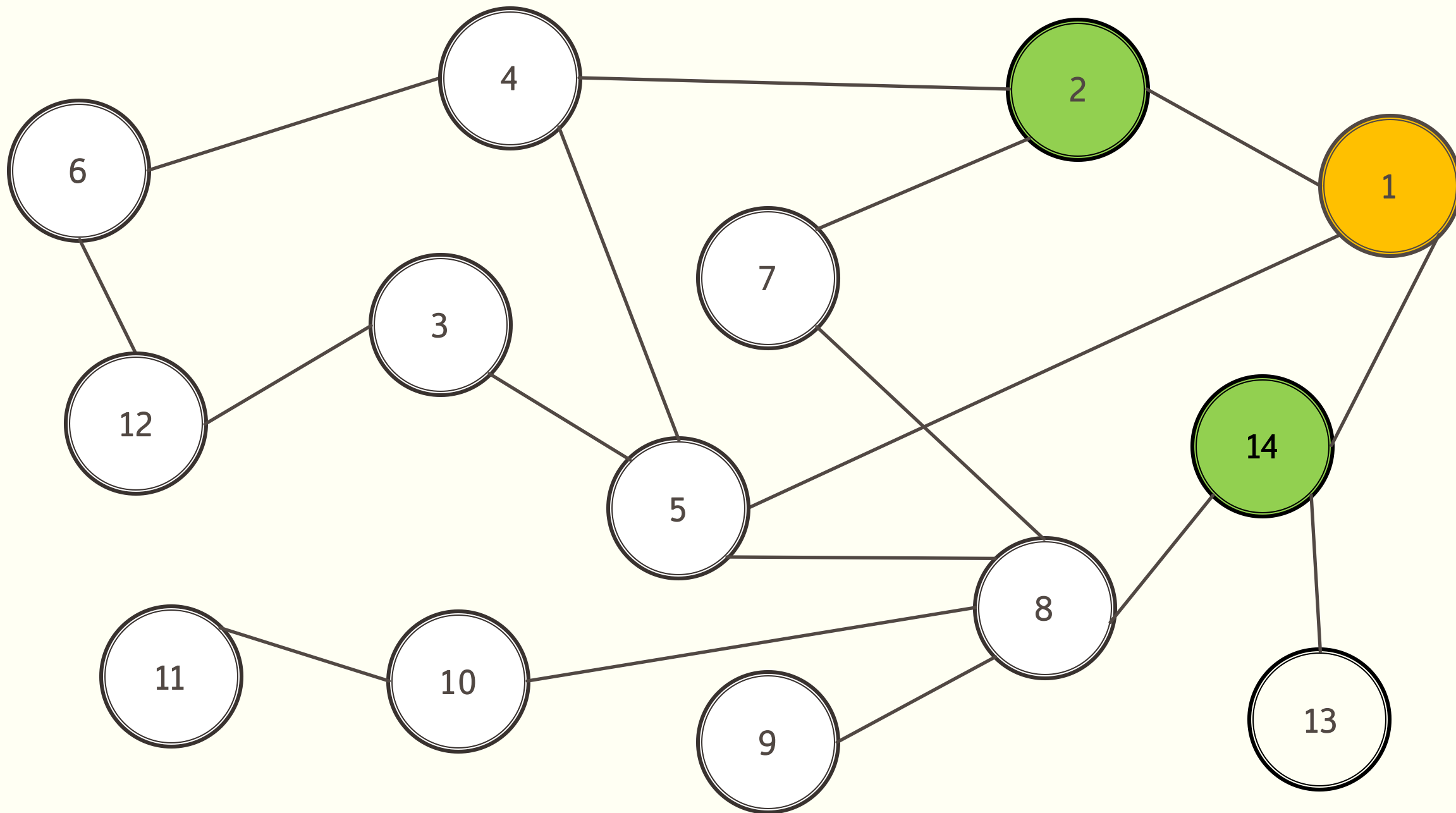


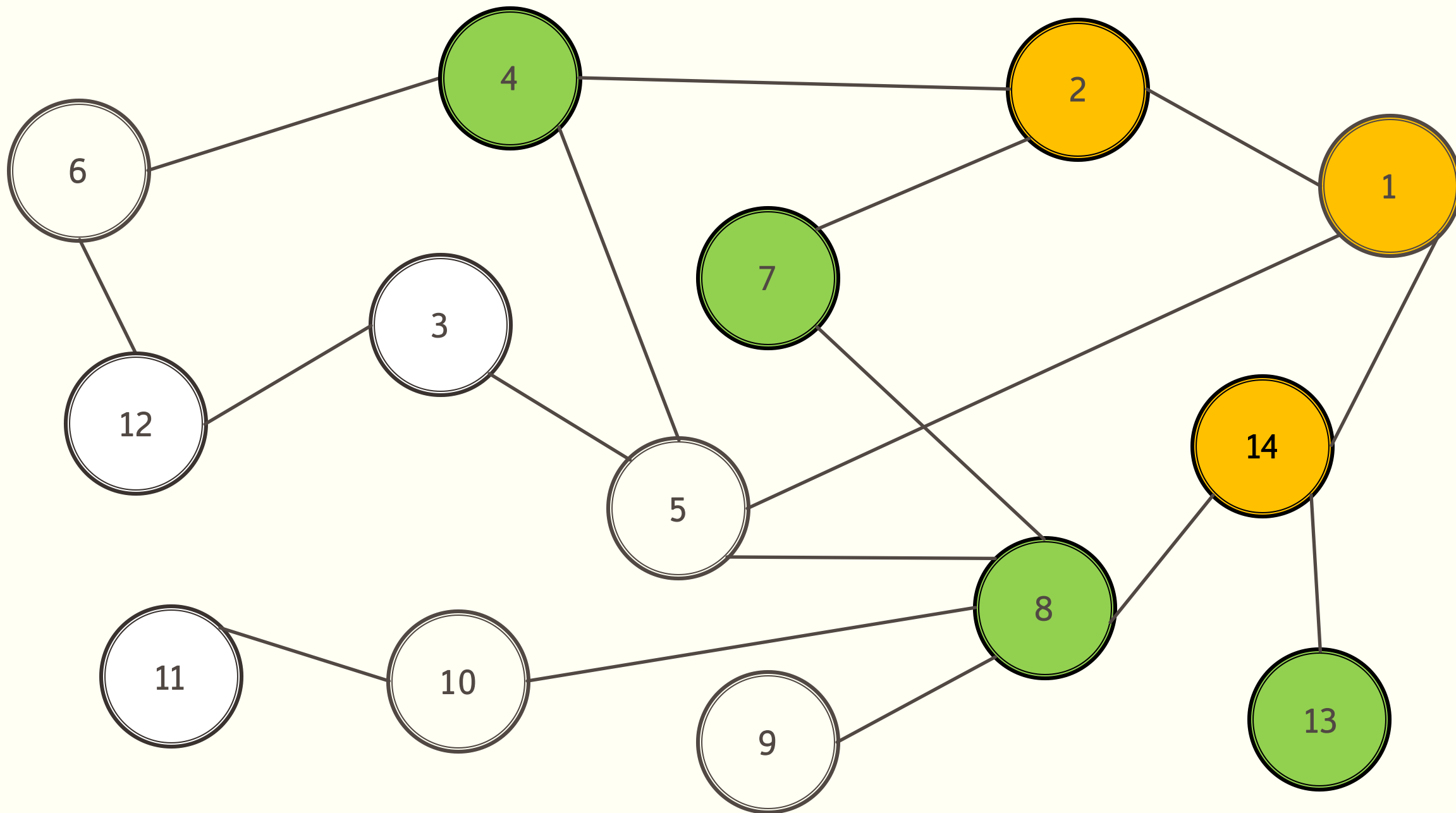
GOSSIP PROTOCOLS

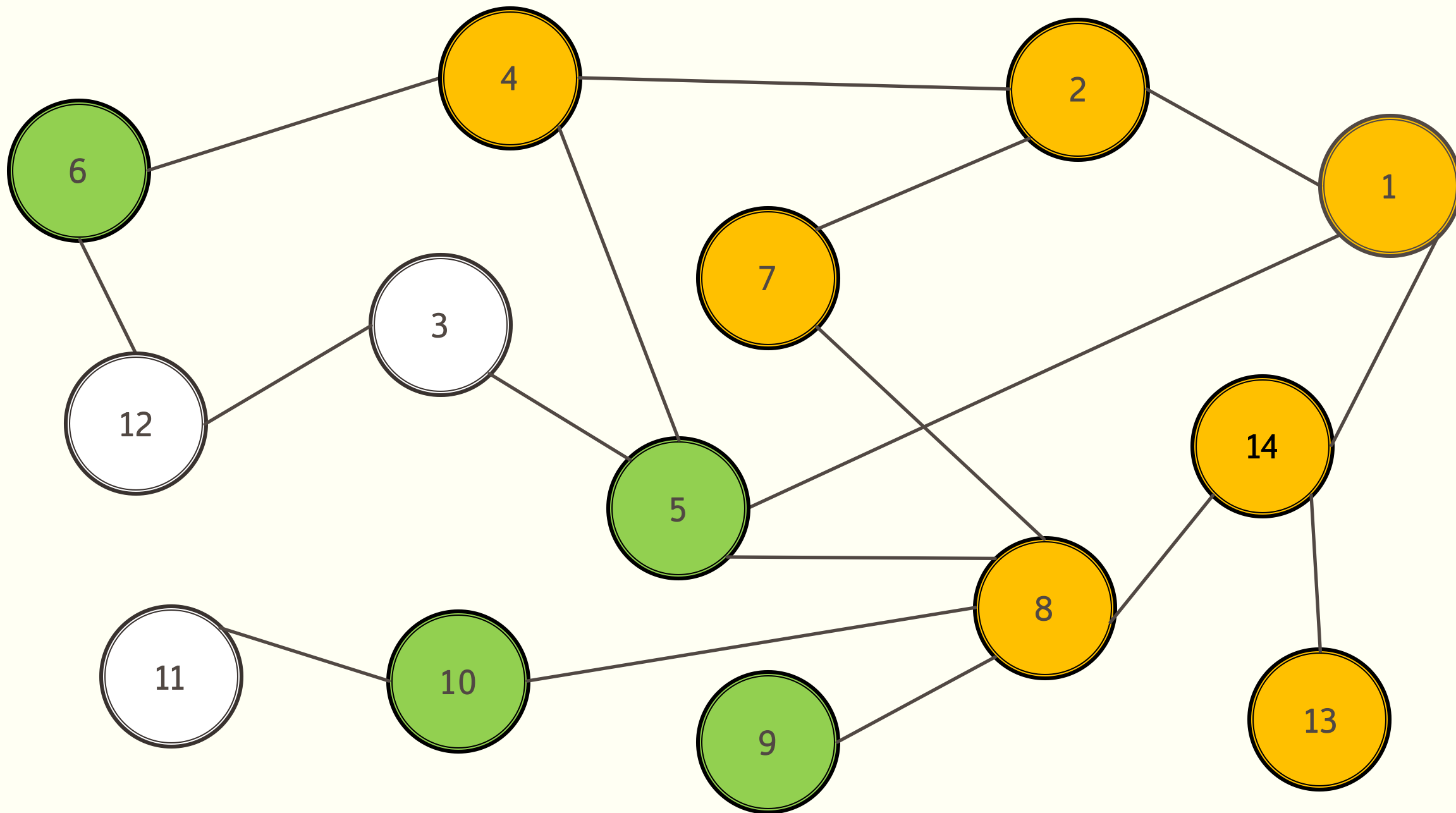
Gossip protocols

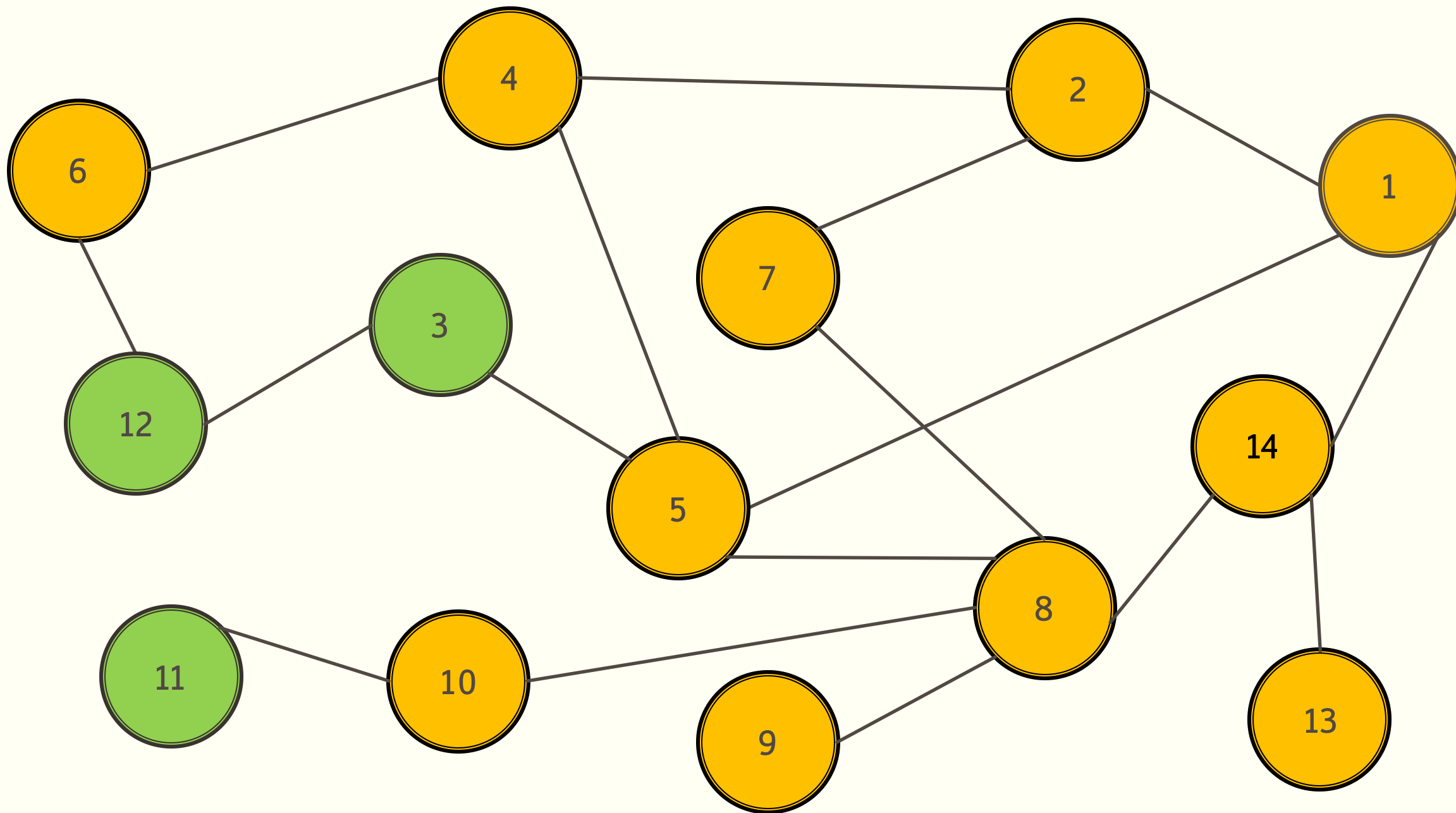
- Each node sends a message to K random targets (multicast).
- K -- infection factor.
- Flooding: If a node gossips to all neighbors.
- Each target randomly select another K targets.
- Process stops when all nodes receive the message or when the message expires.









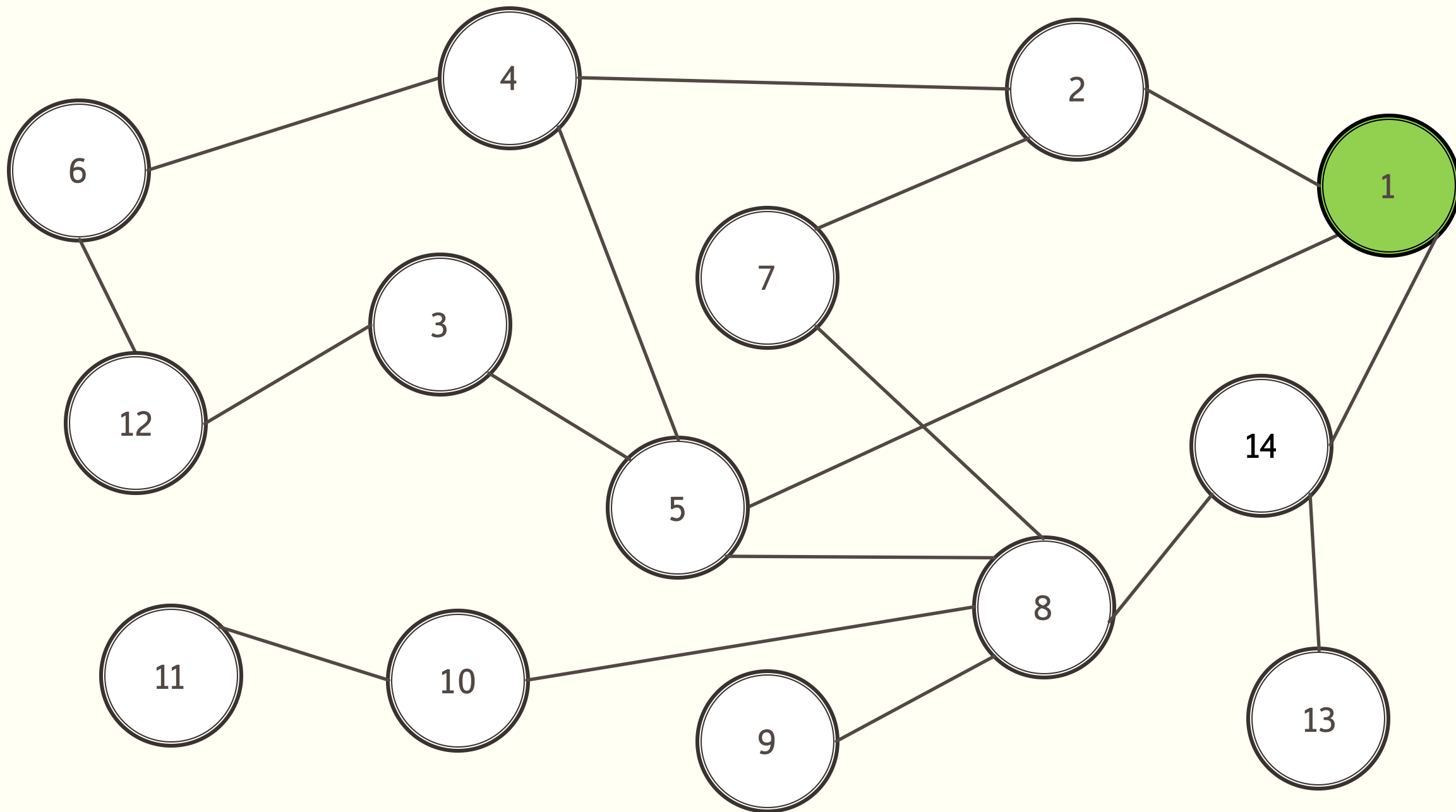


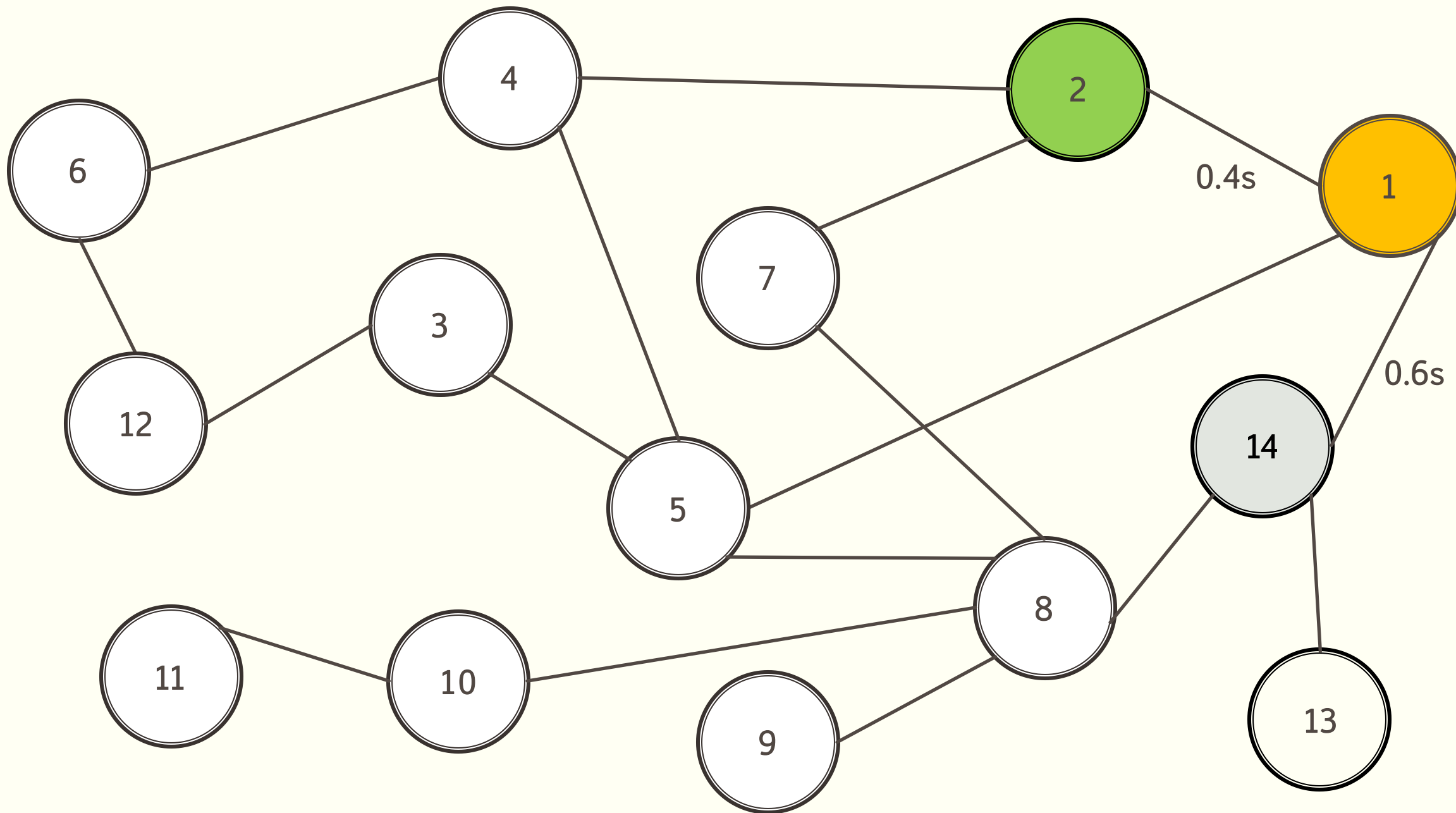
Gossip protocols

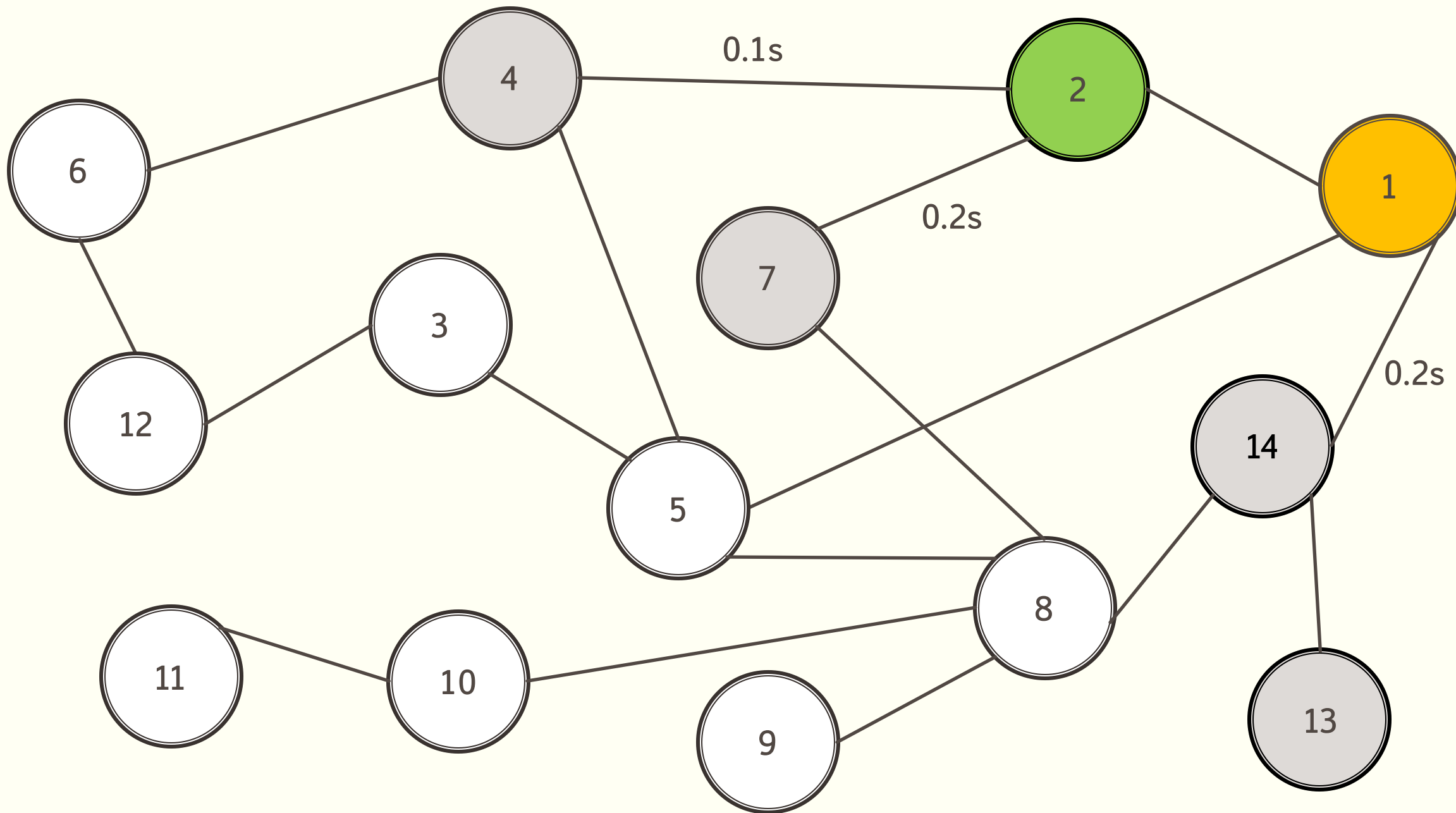
- Reliability: broadcast to entire network.
 - A node send a small number of messages.
 - Fault-tolerant
 - Small number of rounds to reach the entire network.
-
- A super node could keep track of every message.
 - A super node may deanonymize Bitcoin transactions.

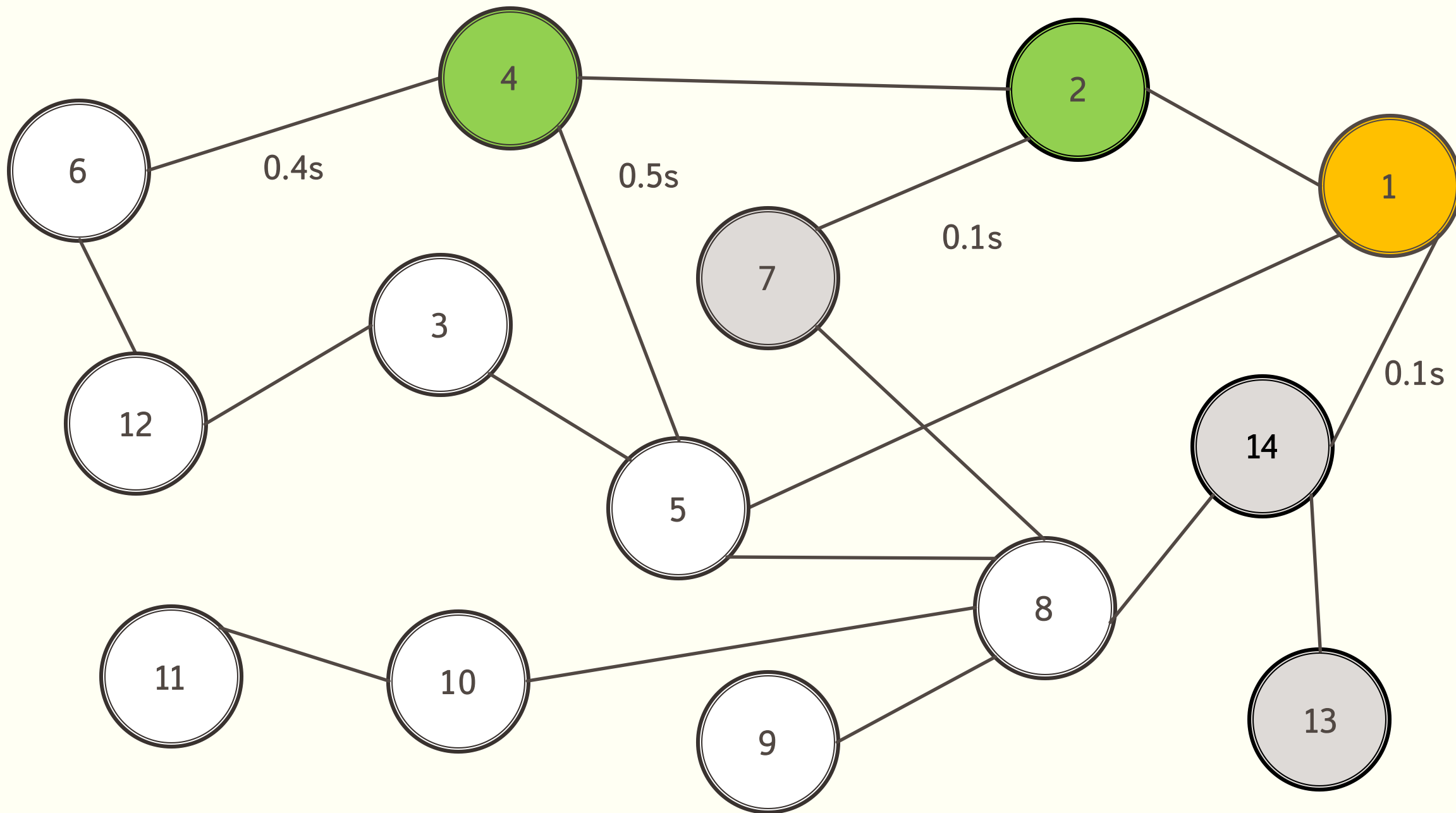
Gossip protocols -- diffusion

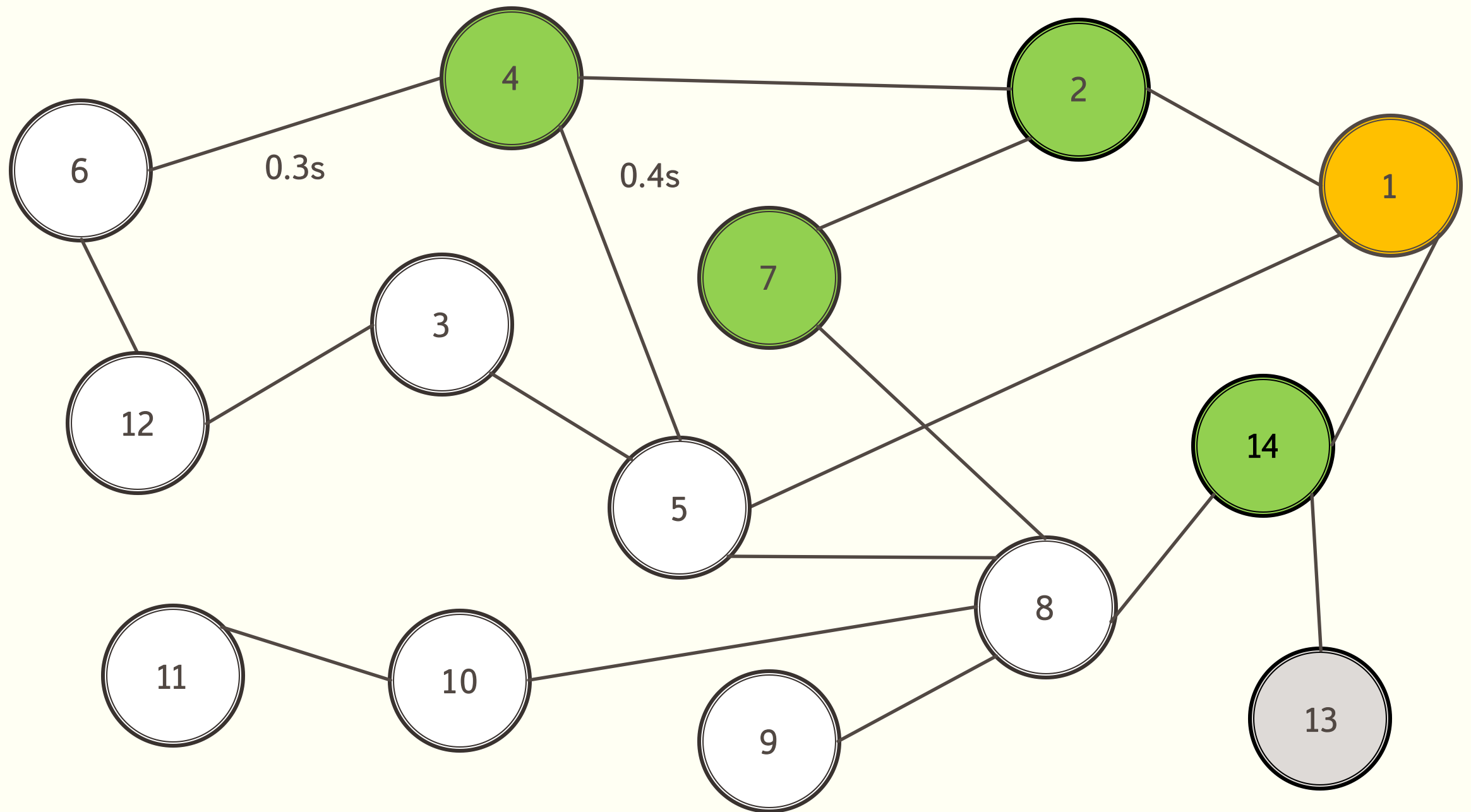
- Each node sends a message to K random targets (multicast).
- Each peer waits a random delay (exponential) before sending the message.













KADEMLIA-RLPX ETHEREUM

Kademlia-RLPx

- Peer-to-peer $\langle \text{key}, \text{value} \rangle$ storage. **DHT** - distributed hash table.
- Lookup system.
 - Routing algorithm based on routing tables
 - Locates servers *near* a destination key.
- Minimizes number of messages nodes must send to each other (configuration messages).
- Uses parallel asynchronous queries.
- Yields a topology of low diameter.
- RLPx, Ethereum node discovery protocol is based on Kademlia.

Kademlia-RLPx

- Each node has an **id**, in Ethereum **ENODEID**, secp256 hash.
- Each node store a node record, in Ethereum **ENR**.
- Distance between nodes is given by XOR metric.
 - $\text{distance}(n_1, n_2) = \text{keccak256}(n_1) \text{ XOR } \text{keccak256}(n_2)$
- XOR symmetric distance: nodes receives queries from nodes contained in their routing tables.
- Routing information are learned from received queries.
- Every message includes sender's node ID, permitting the recipient to record sender's ID if necessary.

Kademlia-RLPx

- **XOR distance:** $d(x,x) = 0$, $d(x,y)=d(y,x)$, $d(x,y)+d(y,z) \geq d(x,z)$.
- **Unidirectional distance:** for any given point x and a distance D , there is exactly only one point y such that $d(x,y) = D$
- All lookups for the same key follow the same path, regardless of the starting node.
- For each $1 \leq i < 160$, every node keeps a list of tuples: (**k-bucket list**)
 - $\langle \text{IP address, UPD port, NODE ID} \rangle$ for nodes of distance between 2^i and 2^{i+1} from itself.
 - tuples in a k-bucket list are sorted by time last seen-least-recently seen node at the head
 - The number of elements in a bucket can grow up to size k .

Kademlia-RLPx

- Live nodes are never removed from the k-bucket list.
- When a node receives a request or a replay from a node it updates the k-bucket list.
 - If the sender already exists in the recipient list, the recipient moves it at the **tail** of the list.
 - If the node is not already in the appropriate k-bucket and the bucket has fewer than k entries, then the recipient inserts the new sender at the **tail** of the list.
 - If the node is not already in the appropriate k-bucket and the bucket is full, then the recipient pings the s the k-bucket's least-recently seen node (**head**).
 - If the least-recently seen node doesn't respond, it is evicted from the k-bucket and the new sender inserted at the tail.
 - if the least-recently seen node responds, it is moved to the **tail** of the list, and the new sender's contact is discarded.

Kademlia-RLPx

- k-buckets maximize the probability that the nodes they contain will remain online, the longer a node has been up, the more likely it is to remain up another hour.
- Node failures are inversely related to uptime.
- DoS attacks resistance.
- RPCs:
 - **PING** check if a node is online.
 - **STORE <key, value>** request a node to store a <key, value> pair.

Kademlia-RLPx

- **FIND-NODE** finds 160-bit ID. It returns a k triples of type:
 - $\langle \text{IP address, UDP port, Node Id} \rangle$ for the k-nearest nodes the recipient knows about closest to requested ID.
 - k triples are gathered from one or more buckets.
- **FIND-VALUE**
 - if the recipient previously received a **STORE $\langle \text{key, value} \rangle$** request it returns the value it stored.
 - If the recipient didn't receive a **STORE $\langle \text{key, value} \rangle$** request it returns a list of k triples:
 - $\langle \text{IP address, UDP port, Node Id} \rangle$ for the k-nearest nodes the recipient knows about closest to requested key.

Kademlia-RLPx

- Recursive lookup.
- Locates the closest node to a node ID.
 - The initiator picks α closest nodes to the target it knows of.
 - The initiator sends concurrent Find-Node packets to the selected nodes.
 - Recursive-step: The initiator picks α closest nodes to the target it knows of from previous queries.
- If a FIND-NODE round fails to return a node closer than the closest already seen, the initiator resends FIND-NODE to all k-closest node it has not queried.

Kademlia-RLPx

- Extension of RLP serialization format.
- Does not use STORE and FIND_VALUE.
- Keys are randomly defined. Geographic information cannot be inferred from distance.
- Nodes generate a ECDSA key-pair, with 512-public key as ENODEID.
- The distance between two nodes N1, N2 is the length of the common prefix of N1 and N2.



WIRE PROTOCOL

Wire protocol-Ethereum

- Application layer protocol for propagating transactions and block.
- Facilitates exchange of Ethereum blockchain information between peers over **TCP**
- To establish a connection **STATUS** messages are sent. Recipient is informed about:
 - version: the current protocol version.
 - networkid
 - td: total difficulty of the best chain
 - blockhash: hash of the block with the highest TD known
 - genesis: hash of the genesis block
 - forkid: fork identifier RLP([FORK_HASH, FORK_NEXT])

Wire protocol-Ethereum

- Examples of validation rules for connections EIP-2124
 - Nodes must run on the same network
 - If two chains share the same genesis, but not forks (ETH / ETC), they should reject each other.
 - If the remote FORK_HASH is a subset of the local past forks and the remote FORK_NEXT matches with the locally following fork block number, connect.
 - Remote node is currently syncing. It might eventually diverge from, but at this current point in time we don't have enough information
- Following STATUS message reception, three operations can be performed:
 - Chain synchronization: New clients request blocks and sync to existing state.
 - Block propagation: newly mined blocks are relayed to all nodes.
 - Transaction exchange: pending transactions are relayed to miners.

Wire protocol-Ethereum

- **Chain synchronization:** New clients request blocks and sync to existing state
- STATUS message includes total difficulty TD and the hash of the best known block.
- NODE with worst TD sends **GetBlockHeaders**
 - The response contain a number of block headers, beginning a a certain start_block, with most limit number of blocks.
- After receiving BlockHeaders nodes verifies PoW and sends a **GetBlockBodies** message.
- Received blocks are executed in the EVM.
- Fast sync: synchronize transactions results (tree states and receipts)

Wire protocol-Ethereum

- **Block propagation:** newly mined blocks are relayed to all nodes
- **NewBlock** announces the receiver of the existence of a new block.
 - Store that sender knows block hash.
 - Receiver verifies header of the block.
 - PoW
 - Gas limit
 - Difficulty'
 - Block time etc.
 - Receiver send the block to a small fraction of connected peers which it didn't notify earlier.
 - Block is executed and state-root must match the post state-root.
 - Receiver sends **NewBlockHashes** message about the new block to all peers which it didn't notify earlier.

▪

Wire protocol-Ethereum

- The reception of a block announcement may also trigger chain synchronization.
- **Reduce block reputation for a block if block:**
 - Announces hashes that later refuses to honor with Block messages.
 - Sends hashes or blocks already known to peers.
 - Sends transaction back to a peer that already know of it.
- A node remembers transactions hashes it has relayed to connected peers.
- A node remembers block hashes it has relayed to connected peers.

Wire protocol-Ethereum

- **Transaction exchange**
- Nodes store pending transactions in transaction pool
- **NewPooledTransactionHashes** and **GetPooledTransactions** for transaction pool synchronization when a new connection is established.
- New transactions are propagated using **Transactions** and **NewPooledTransactionHashes**.
- **Transaction** messages are sent to a small fraction of connected peers.
- **NewPooledTransactionHashes** is sent to all peers unnotified before.
- All peers receiving a notification of the transaction hash can request the complete transaction object if it is unknown to them **GetPooledTransactions**.

Bibliography

- Number of Bitcoin nodes: <https://bitnodes.io/>
- Information Propagation in the Bitcoin Network Christian Decker,* Roger Wattenhofer https://tik-old.ee.ethz.ch/file/49318d3f56c1d525aabf7fda78b23fc0/p2p2013_041.pdf
- Bitocin full nodes: <https://bitcoin.org/en/full-node#what-is-a-full-node>
- https://en.bitcoin.it/wiki/Protocol_documentation
- Kademlia: A Peer-to-peer Information System Based on the XOR Metric <https://www.scs.stanford.edu/~dm/home/papers/kpos.pdf>
- Ethereum RLPx <https://github.com/ethereum/devp2p/blob/master/discv4.md>
- Ethereum Wire Protocol <https://github.com/ethereum/devp2p/blob/master/caps/eth.md>