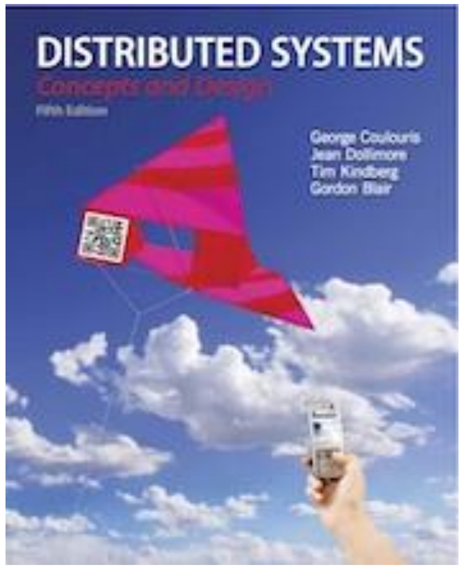
Peer-to-Peer Systems

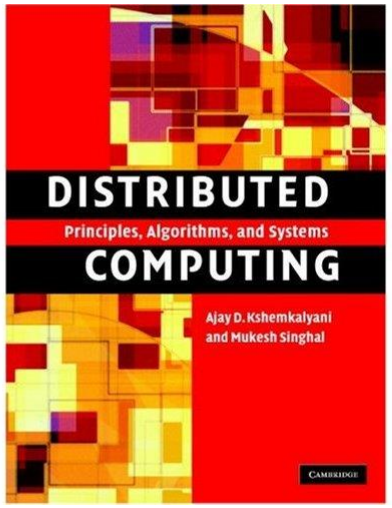


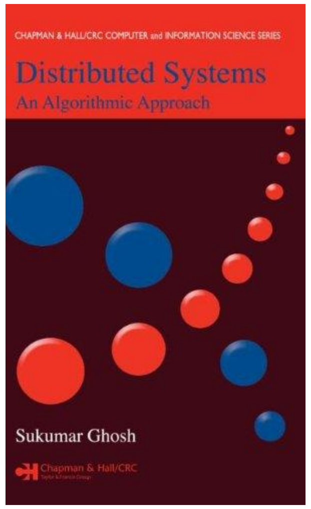
From **Coulouris, Dollimore, Kindberg and Blair**

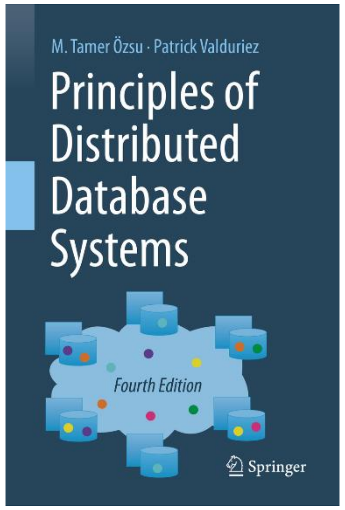
Distributed Systems:

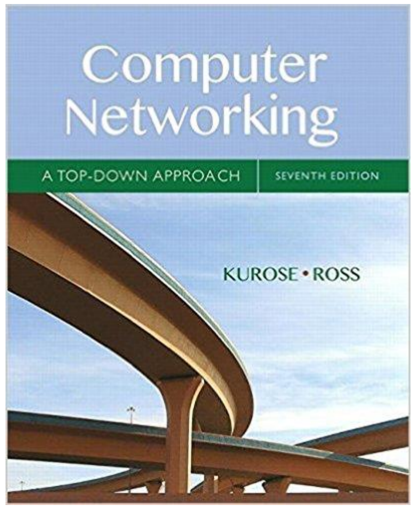
Concepts and Design

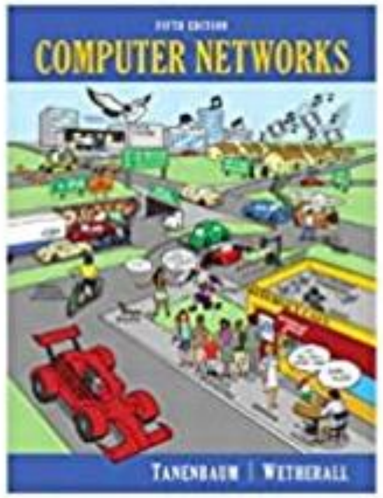
Edition 5, © Addison-Wesley 2012











Agenda

1. P2P networks
2. Unstructured overlays
   - Napster
   - BitTorrent
3. Structured overlays (DHT)
   - Chord
   - IPFS (https://ipfs.io/)

4. Bitcoin

File storage and distribution

1. P2P networks

Application-level organization of the overlay network to flexibly share resources between users

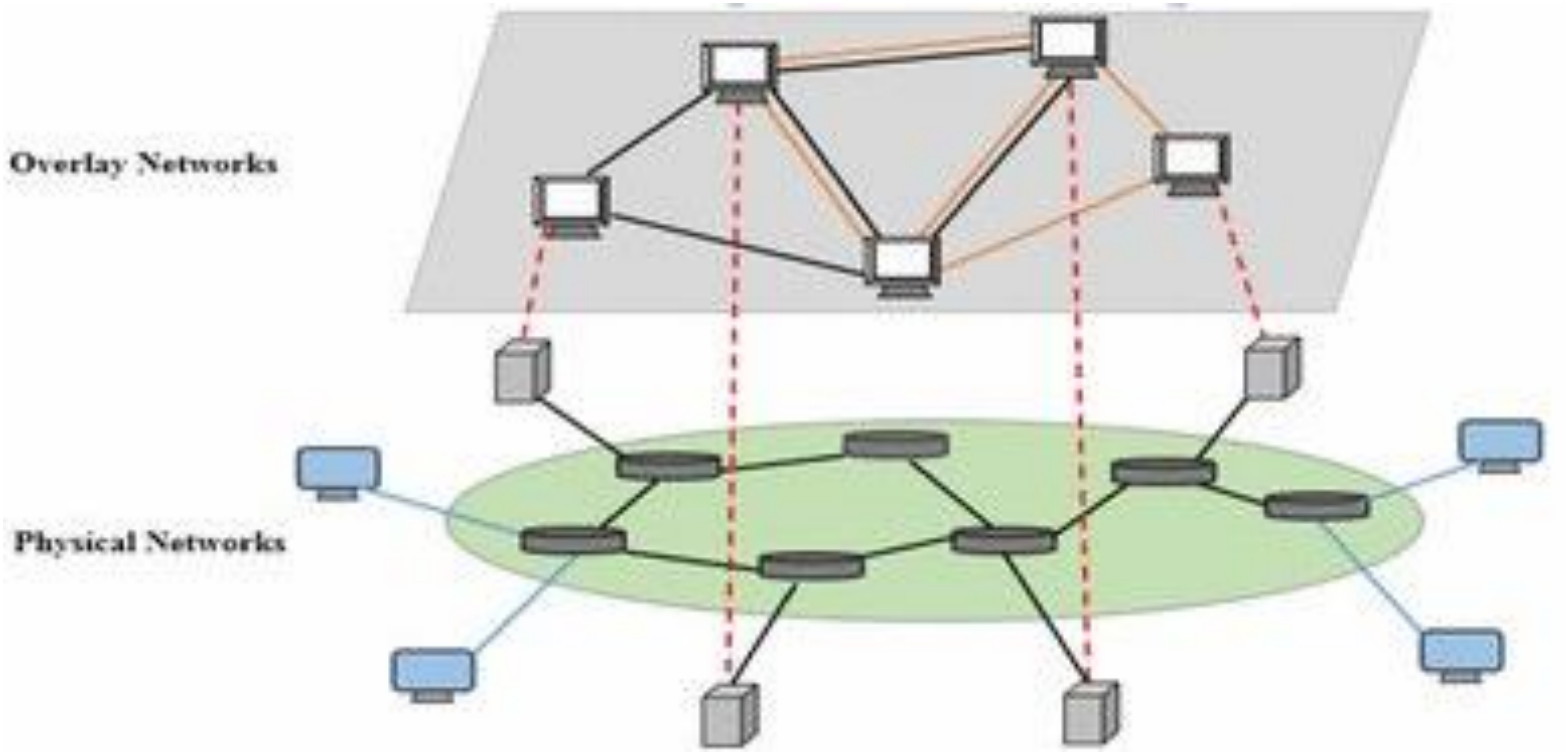
All nodes are equal; communication directly between peers (no central authority)

Large combined storage, CPU power, and other resources without scalability costs

Allow location of arbitrary objects; no DNS servers required

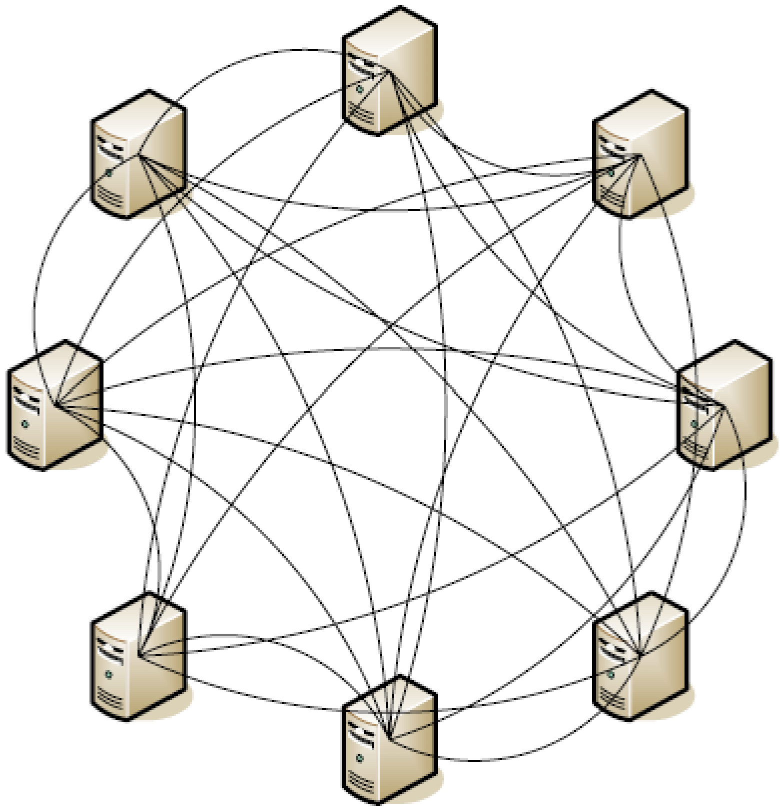
Dynamic insertion and deletion of nodes, as well as of resources, at low cost

P2P overlay networks

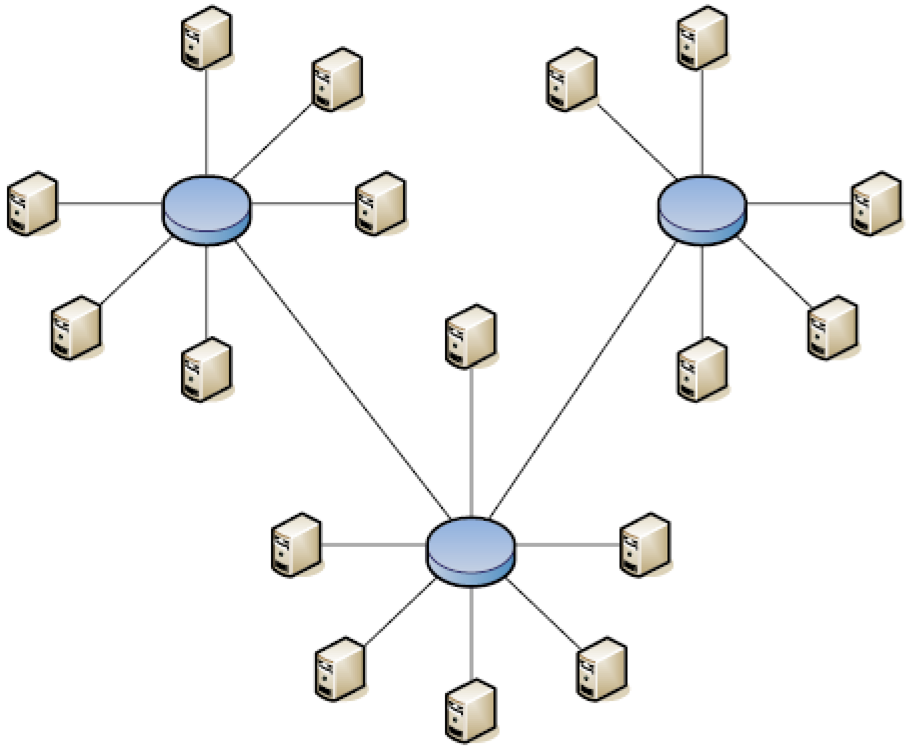


Infrastructure buit on top of a physical network

P2P networks



Pure P2P



Hybrid P2P

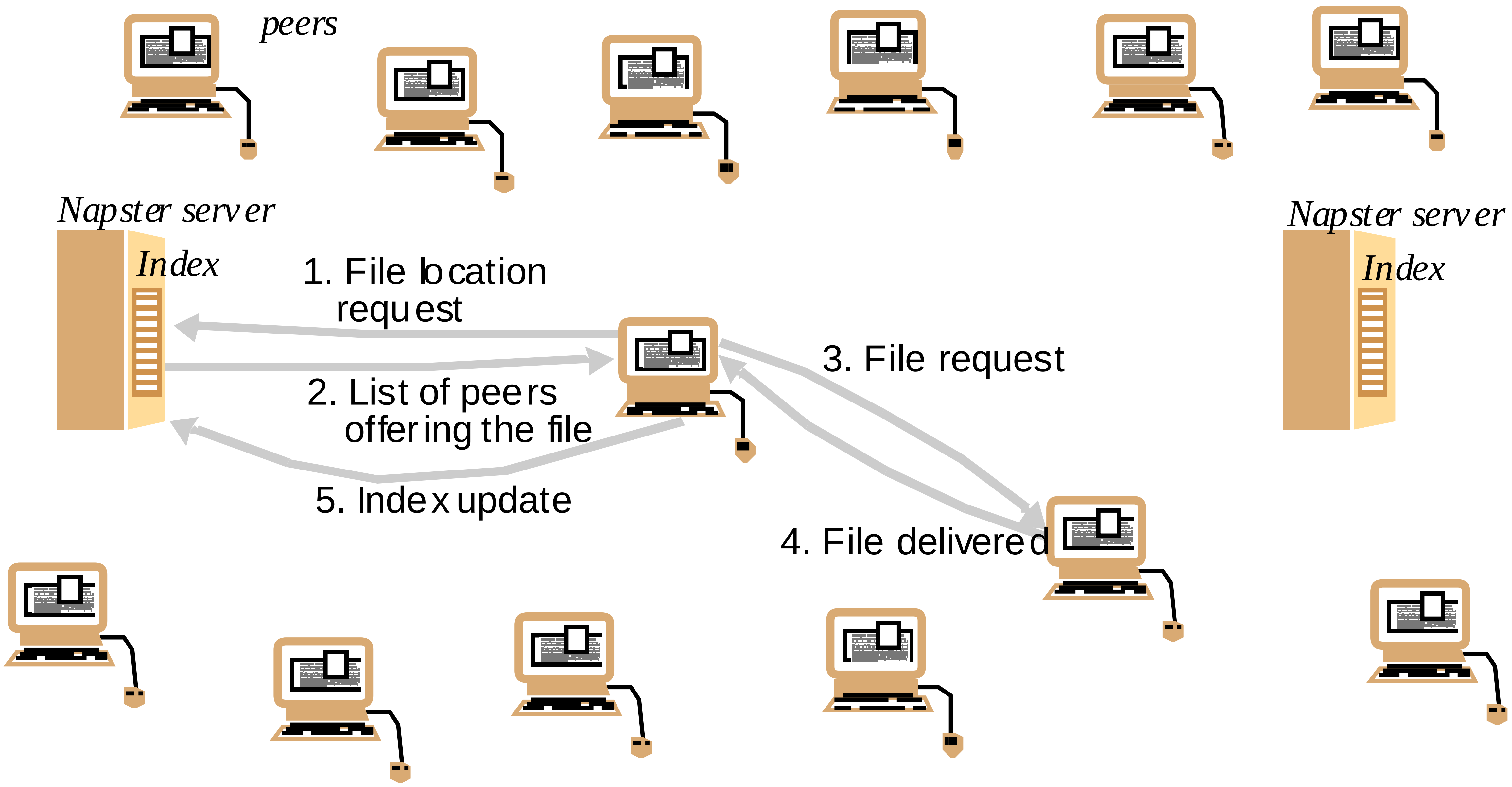
2. Unstructured overlays

The overlay network is created in a no deterministic (ad hoc) manner and the data placement is completely unrelated to the overlay topology

Replicated copies of popular files are shared among peers, without the need to download them from a central server

The peers are often simply home computers, they do not need to be machines in Internet data centers

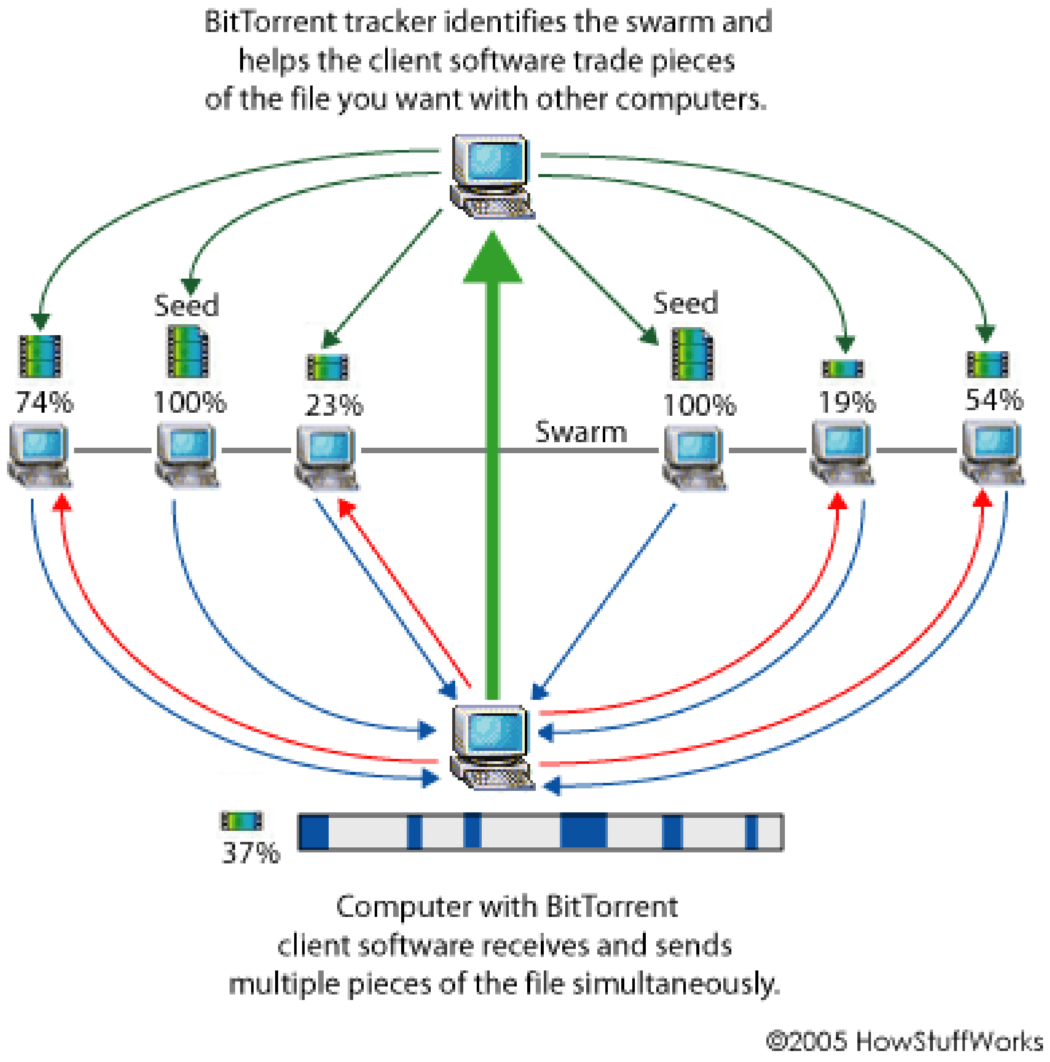
Napster: P2P file sharing (1999-2001)



Centralized, replicated index

BitTorrent

Cohen, B. (2003). Incentives Build Robustness in BitTorrent, May 22, http://bittorrent.org/bittorrentecon.pdf


BitTorrent tracker identifies the swarm and helps the client software trade pieces of the file you want with other computers.
Seed
Seed
74%
100%
23%
100%
19%
54%
Swarm
37%
Computer with BitTorrent client software receives and sends multiple pieces of the file simultaneously.
©2005 HowStuffWorks

BitTorrent™

BitTorrent

There are three main problems that need to be solved to share content:

1. How does a peer find other peers that have the content it wants to download?
2. How is content replicated by peers to provide high-speed downloads for everyone?
3. How does peers encourage each other to upload content to other as well as download content for themselves?

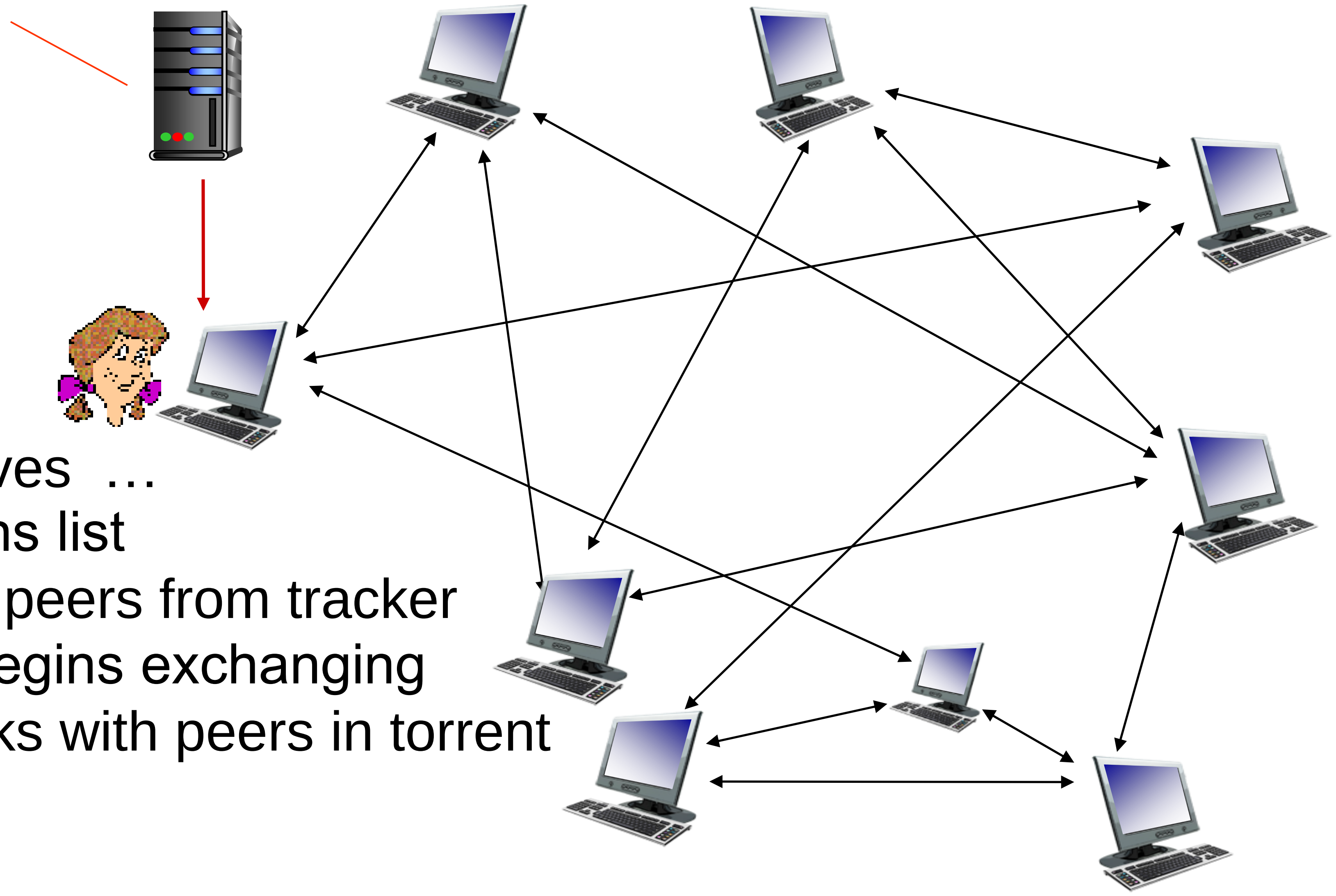
BitTorrent: Architecture

Files divided into 256kB chunks

Peers send/receive file chunks

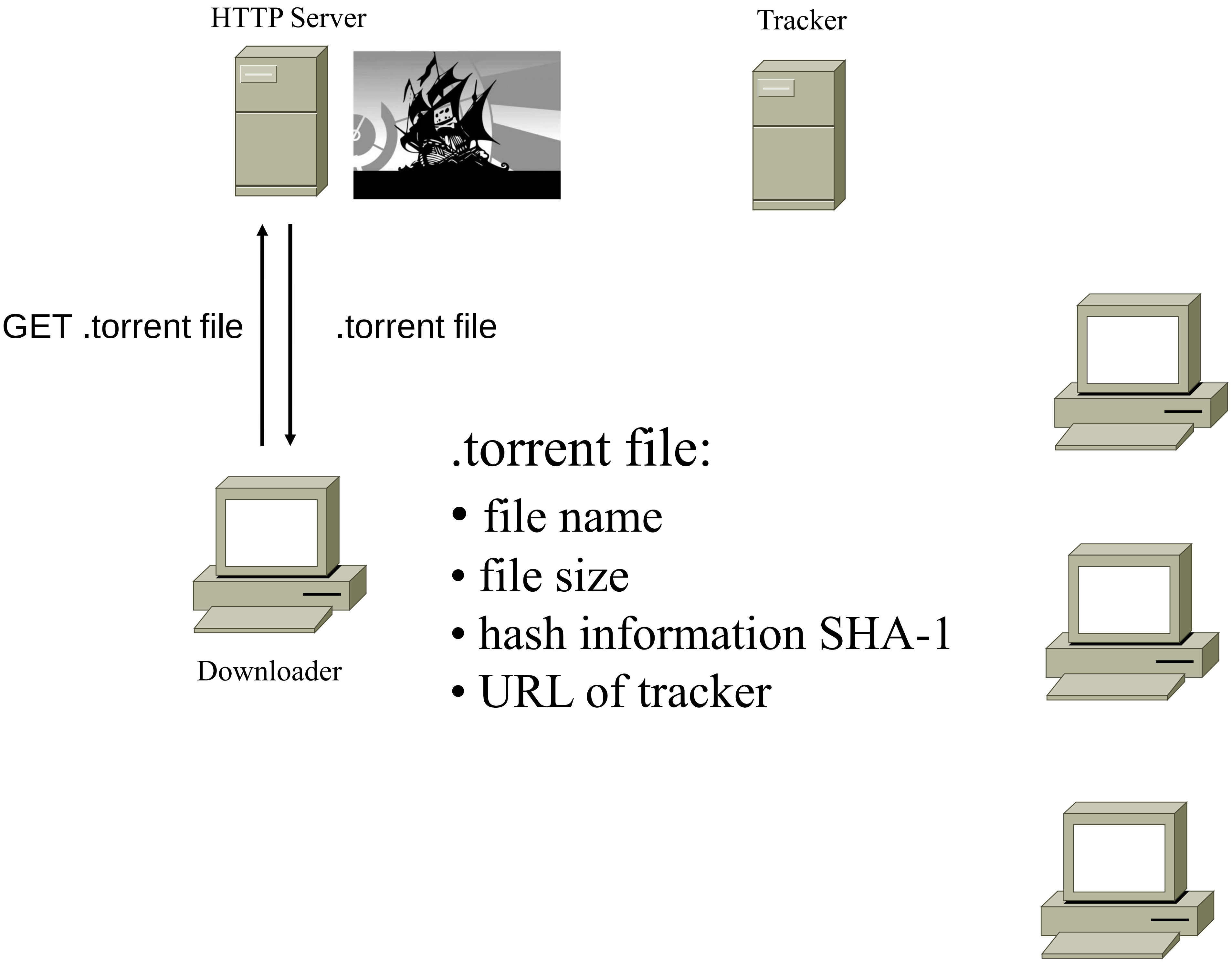
tracker: tracks peers participating in file distribution

swarm: group of peers exchanging chunks of a file



Alice arives …
… obtains list
of some peers from tracker
… and begins exchanging
file chunks with peers in torrent

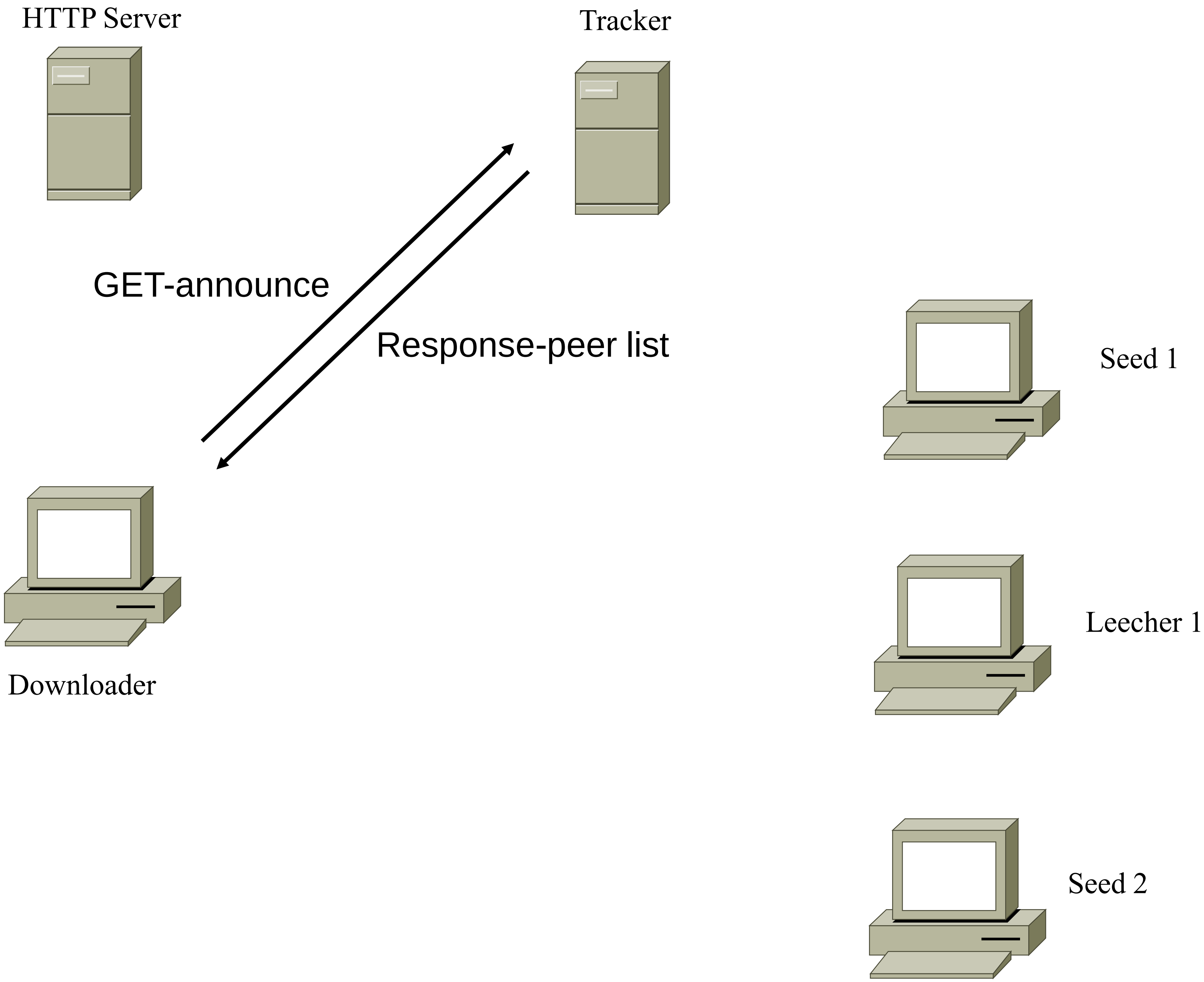
BitTorrent: Get .torrent



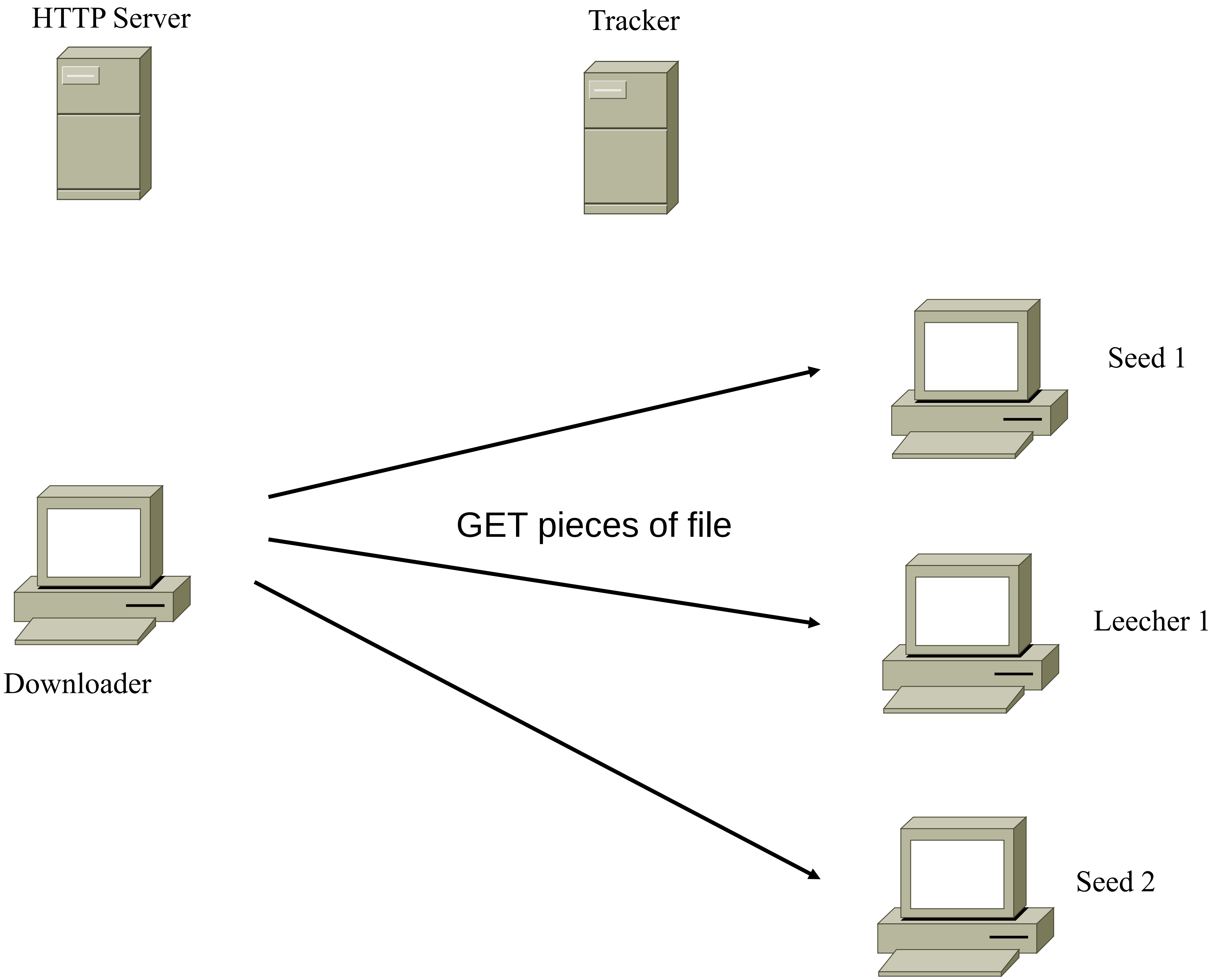
.torrent file:

- file name
- file size
- hash information SHA-1
- URL of tracker

BitTorrent: Get Peer List

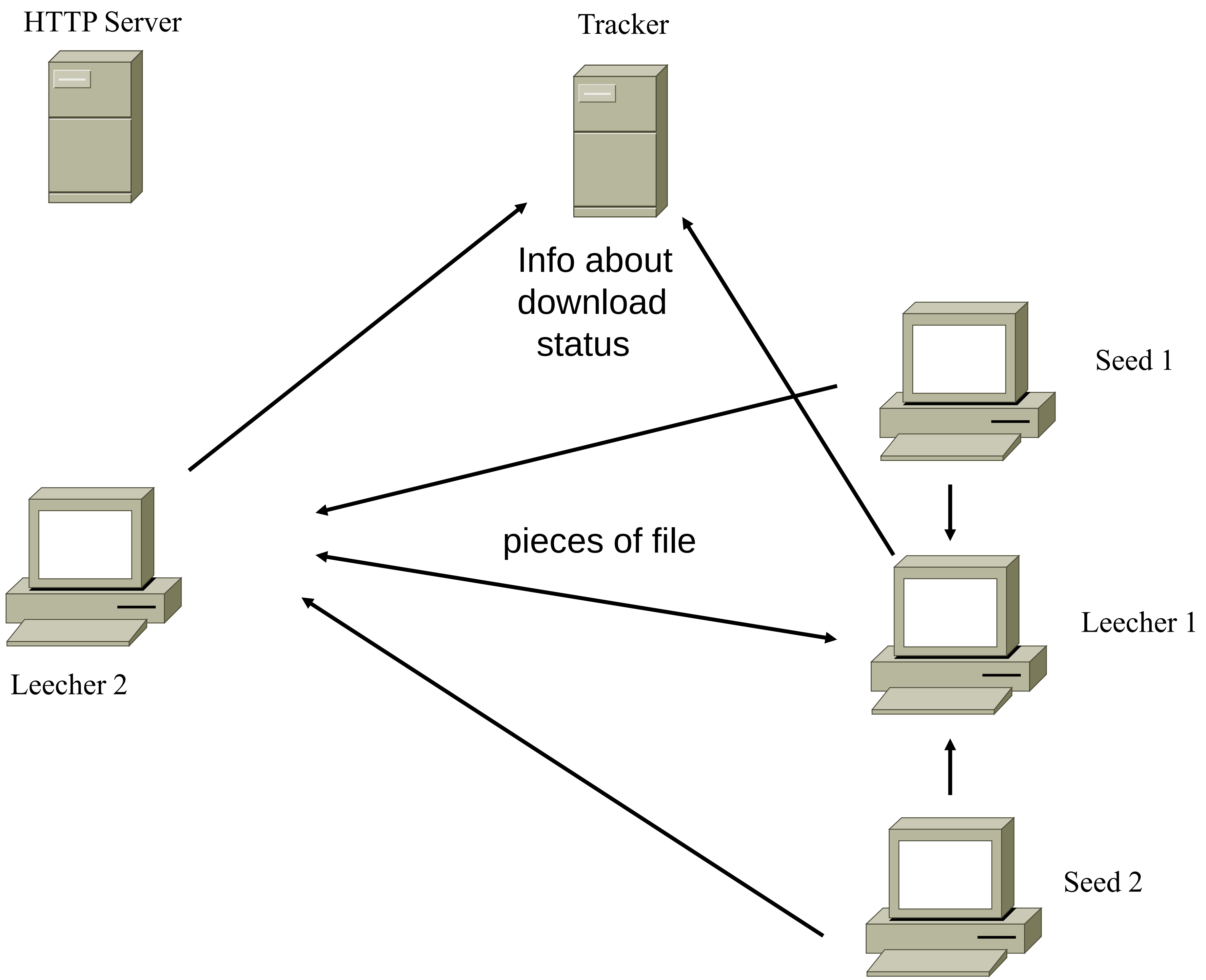


BitTorrent: Query File Pieces



GET pieces of file

BitTorrent: File Pieces



BitTorrent: File Pieces

At any given time, different peers have different subsets of file chunks

Periodically, Alice asks each peer for list of chunks that they have

Alice requests missing chunks from peers following four simple policies:

1. Random first
2. Rarest first
3. Strict policy
4. Endgame mode

BitTorrent: Fairness

While downloading, peer uploads chunks to other peers

A peer may change peers with whom it exchanges chunks to maximize its own download rate

P2P systems depend on all the peers cooperating to store files and allowing other nodes to download from them

BitTorrent: tit-for-tat

Selfish behavior (free-riding) degrades P2P performance.

Need incentives and punishments to control selfish behavior.

A Pareto-optimal solution is one in which the overall good of all participants is maximized.

Tit-for-tat strategy:

- First step, cooperate.
- Subsequent steps, reciprocate the action done by the other in the previous step.

BitTorrent: tit-for-tat

Alice sends chunks to those four peers currently sending her chunks at highest rate

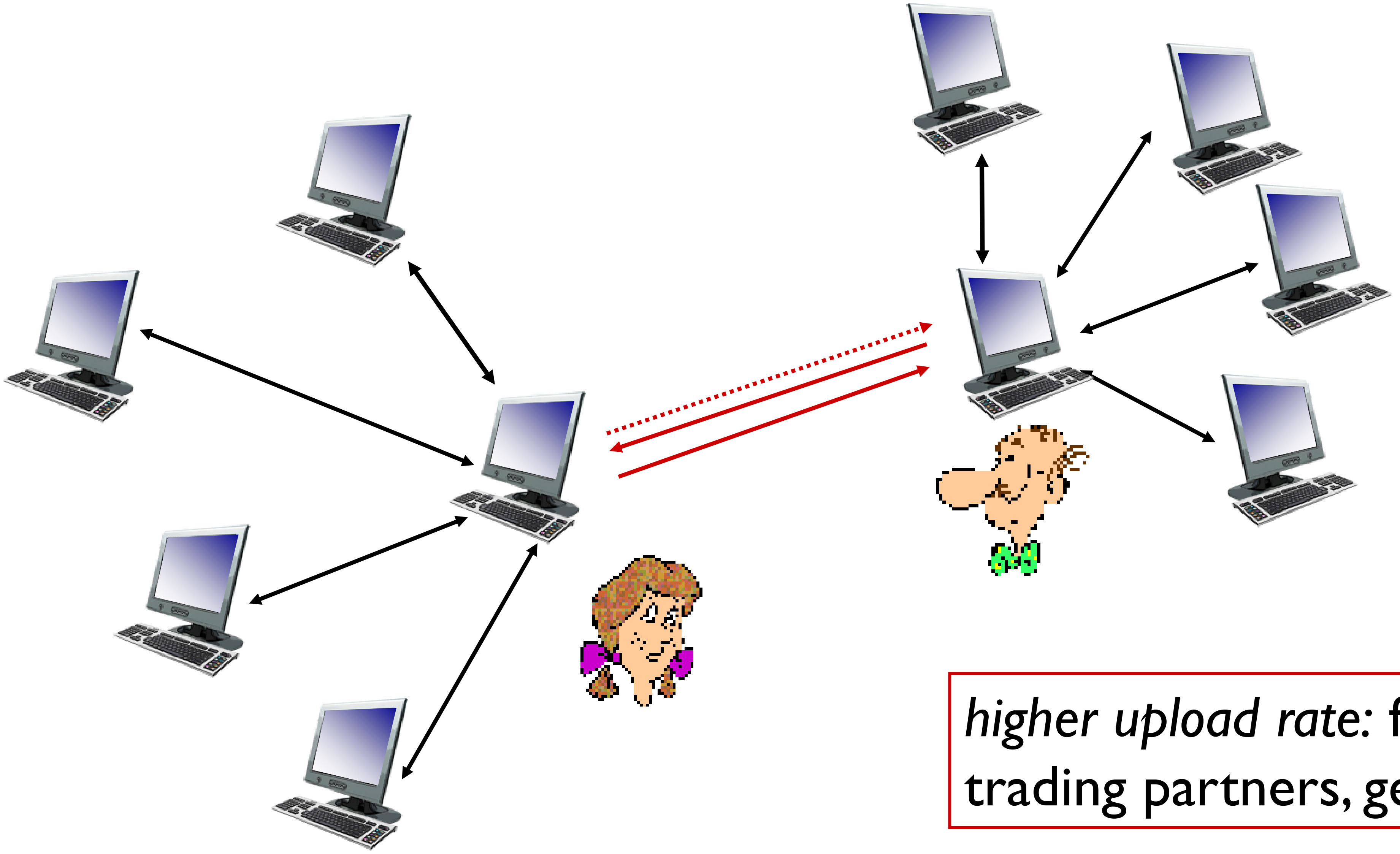
Other peers are choked by Alice (do not receive chunks from her)

Every 10s: re-evaluate top 4

Every 30s: randomly unchoke another peer

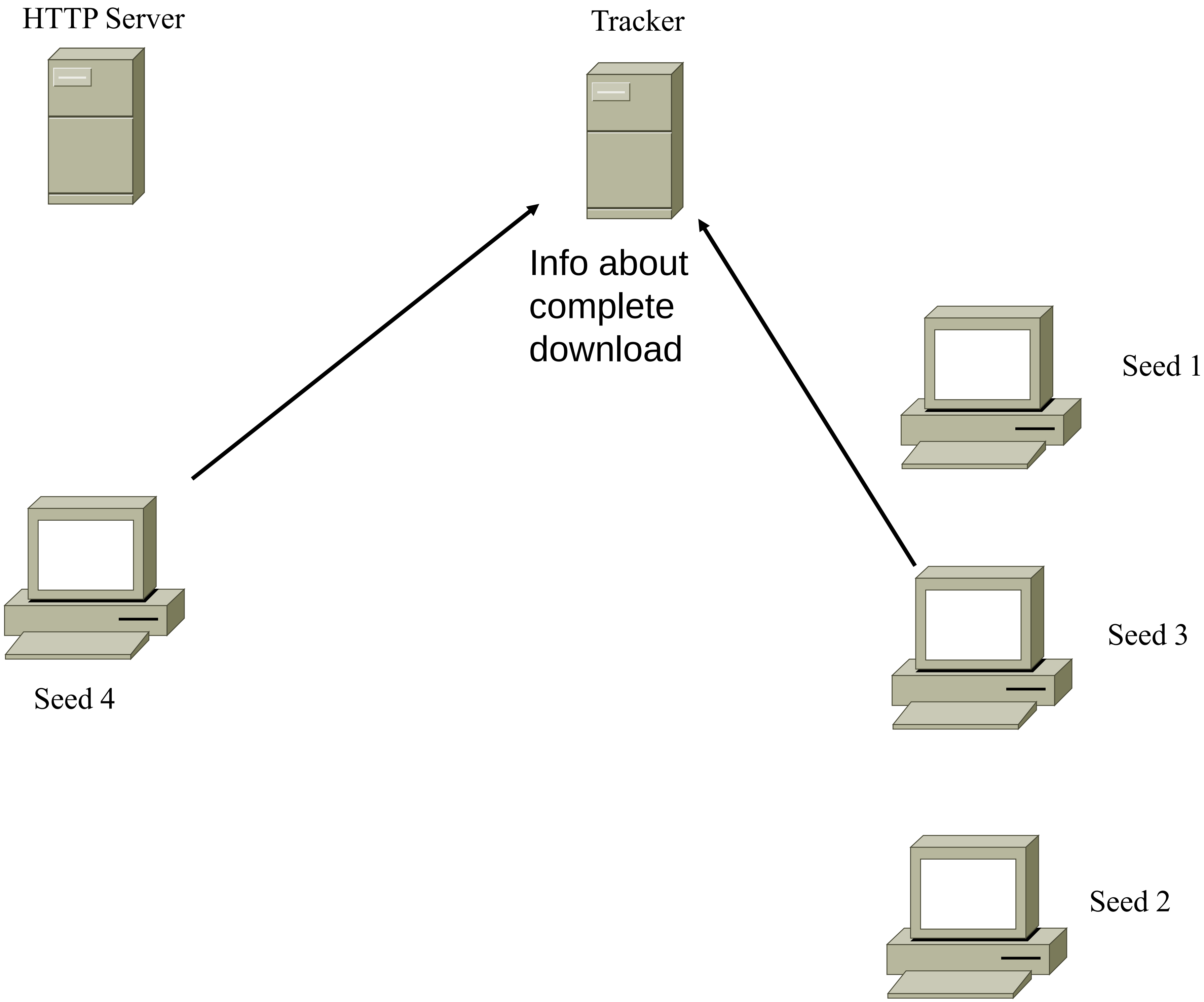
BitTorrent: tit-for-tat

(1) Alice "optimistically unchokes" Bob
(2) Alice becomes one of Bob's top-four providers; Bob reciprocates
(3) Bob becomes one of Alice's top-four providers



higher upload rate: find better trading partners, get file faster !

BitTorrent: Status Information



HTTP Server

Tracker

Info about complete download

Seed 1

Seed 3

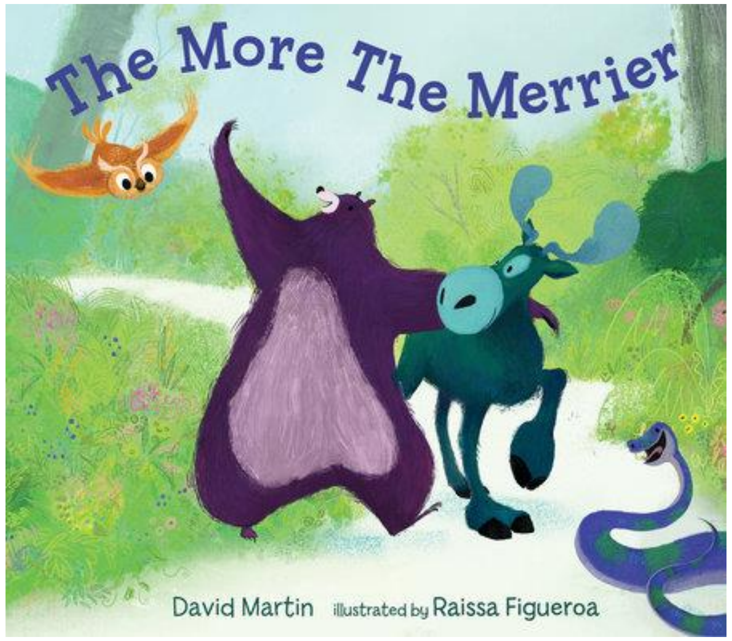
Seed 2

Seed 4

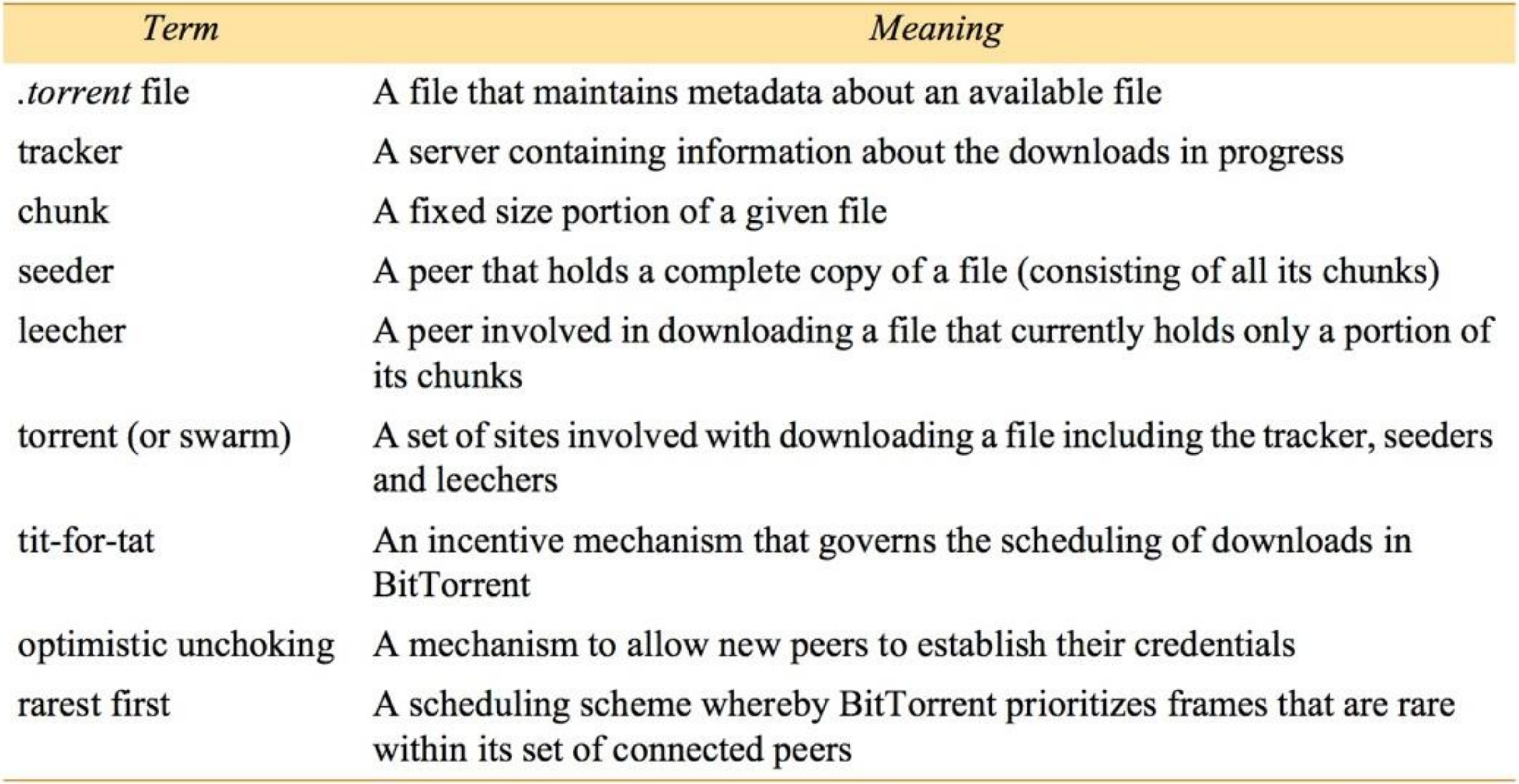
BitTorrent

Peers may come and go (churn)

Once a peer has the entire file, it may (selfishly) leave or (altruistically) remain in swarm

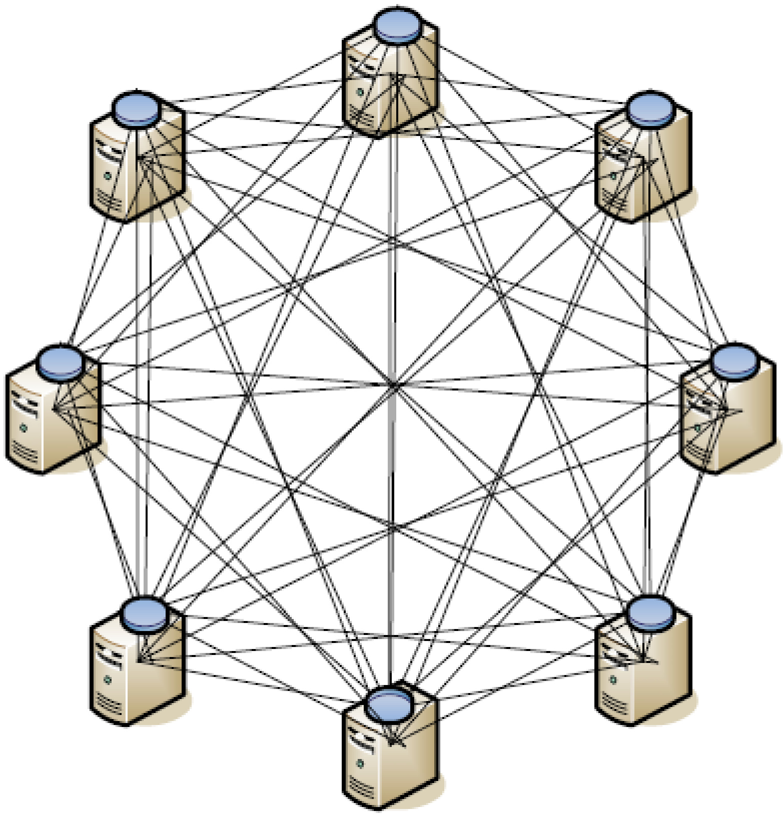


BitTorrent terminology



Term	*Meaning*
.torrent file	A file that maintains metadata about an available file
tracker	A server containing information about the downloads in progress
chunk	A fixed size portion of a given file
seeder	A peer that holds a complete copy of a file (consisting of all its chunks)
leecher	A peer involved in downloading a file that currently holds only a portion of its chunks
torrent (or swarm)	A set of sites involved with downloading a file including the tracker, seeders and leechers
tit-for-tat	An incentive mechanism that governs the scheduling of downloads in BitTorrent
optimistic unchoking	A mechanism to allow new peers to establish their credentials
rarest first	A scheduling scheme whereby BitTorrent prioritizes frames that are rare within its set of connected peers

BitTorrent: Decentralized tracker (2005)



The solution is based on **distributed hash tables** (DHTs)

3. Structured overlays

Middleware layers for the **application-independent** management, storage and fast search, of distributed resources on an imposed regular structure (overlay network)

• Chord	(ring)
• CAN	(hypercube)
• Tapestry	(tree)

Structured overlays

The objective is to build P2P indexes (content, location) that are entirely distributed and perform well

- Each node can look up entries in the index quickly
- Each node keeps only a small amount of information about other nodes
- Each node can use the index at the same time, even as other nodes come and go

Structured overlays – DHT

Achieve higher scalability than unstructured P2P networks at the expense of lower autonomy as each peer that joins the network allows its resources to be placed on the network based on a particular control method

Use a hash function to map objects to machines (to place and locate the objects)

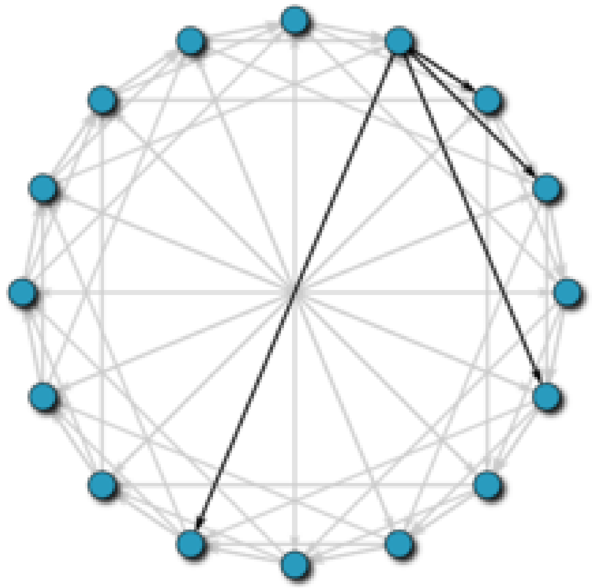
(key, value) pairs

Each key is hashed to generate a logical peer id, which stores the data (value) corresponding to object contents

Chord

Stoica, I., et al. (2001). Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *SIGCOMM'01,* pp.149-160, August 27-31, San Diego, California.

IEEE Transactions on Networking, February 2003.



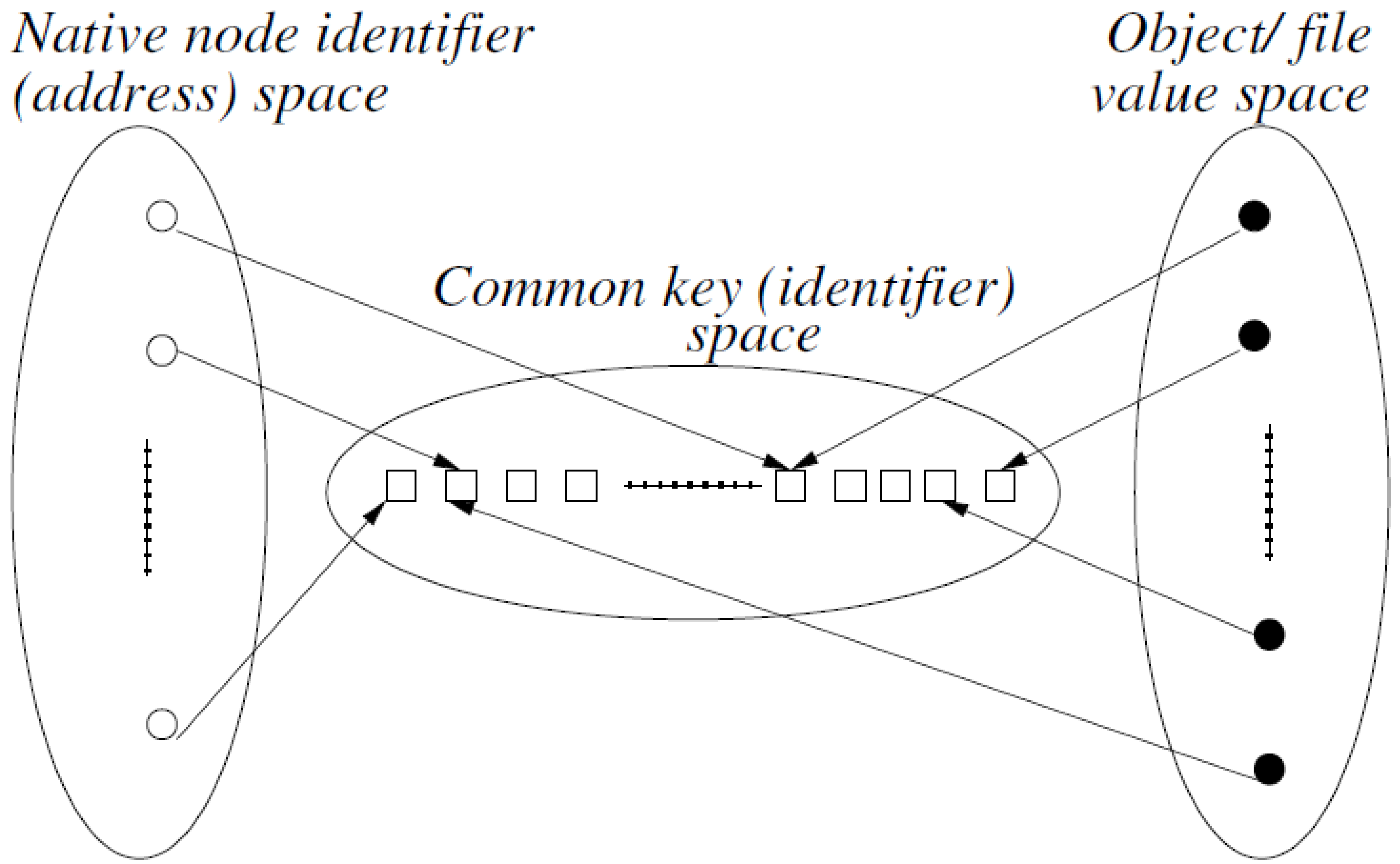
Chord

Chord provides support for just one operation, lookup: given a key, it determines efficiently the node responsible for storing the key´s value

Keys and node identifiers are mapped to an m-bit logical identifier in a common flat key space using a consistent hash function

Object location can be easily implemented on top of Chord by associating a key with each object, and storing the key/object pair at the node to which the key maps

Chord



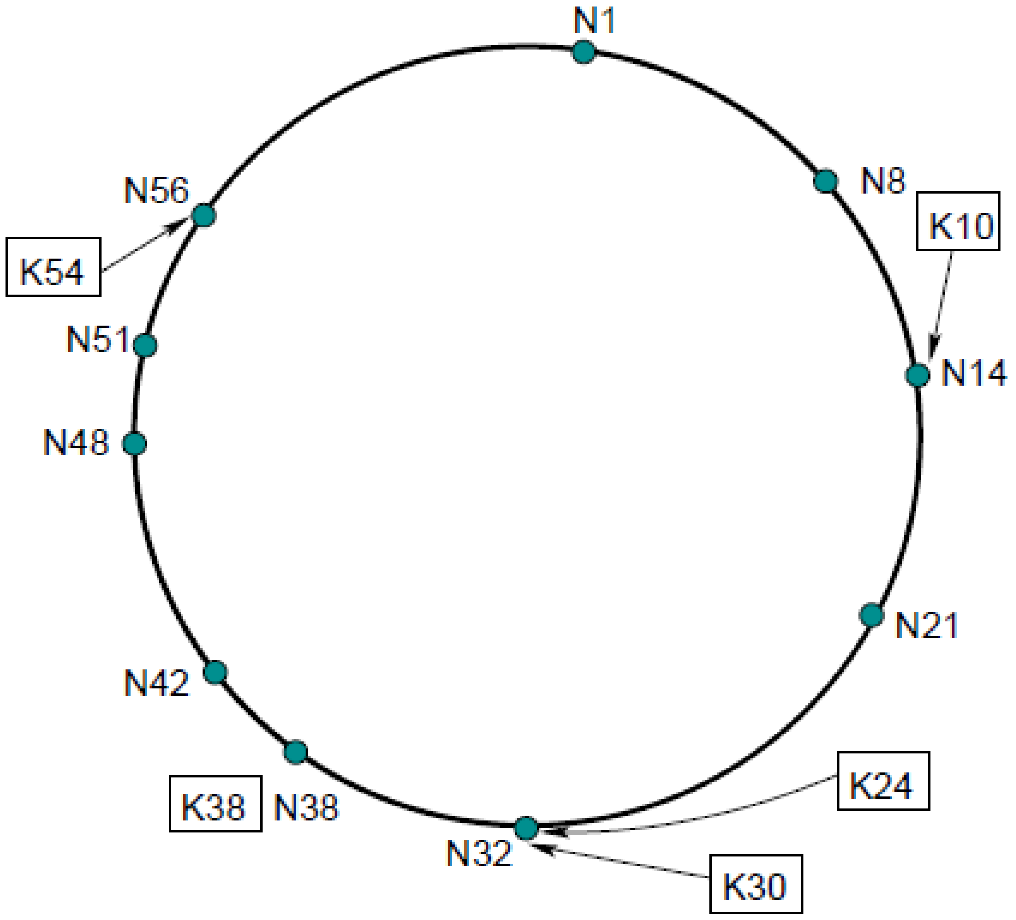
The keys are distributed roughly equally among the nodes

When a node joins or leaves a network having n nodes, only O(1/n) keys need to be moved to a different location

Chord

Common key space has 2^m identifiers, arranged on a logical ring (mod 2^m)

A key *K* gets assigned to the first node such that the node identifier N equals or is greater than the key identifier *K* in the logical space address



m=6

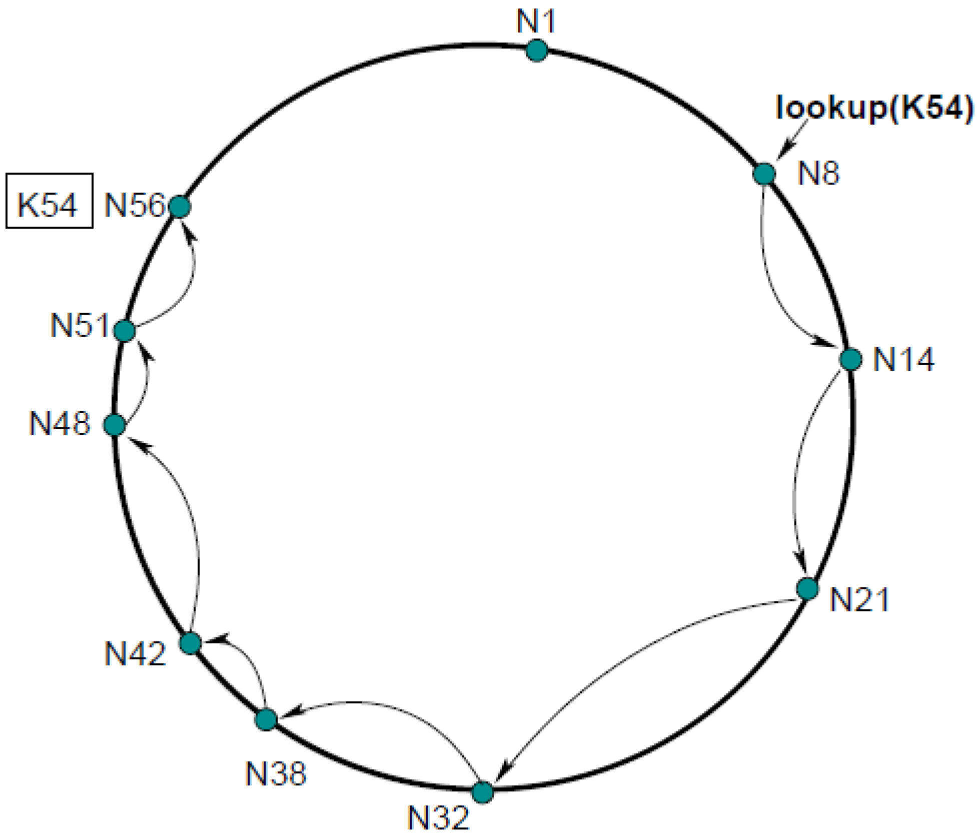
Chord

A query for key *X* is forwarded on the ring until it reaches the first node whose identifier *Y* ≥ X (mod 2^m)

The result, which includes the IP address of the node with key *Y*, is returned to the querying node along the reverse of the path that was followed by the query

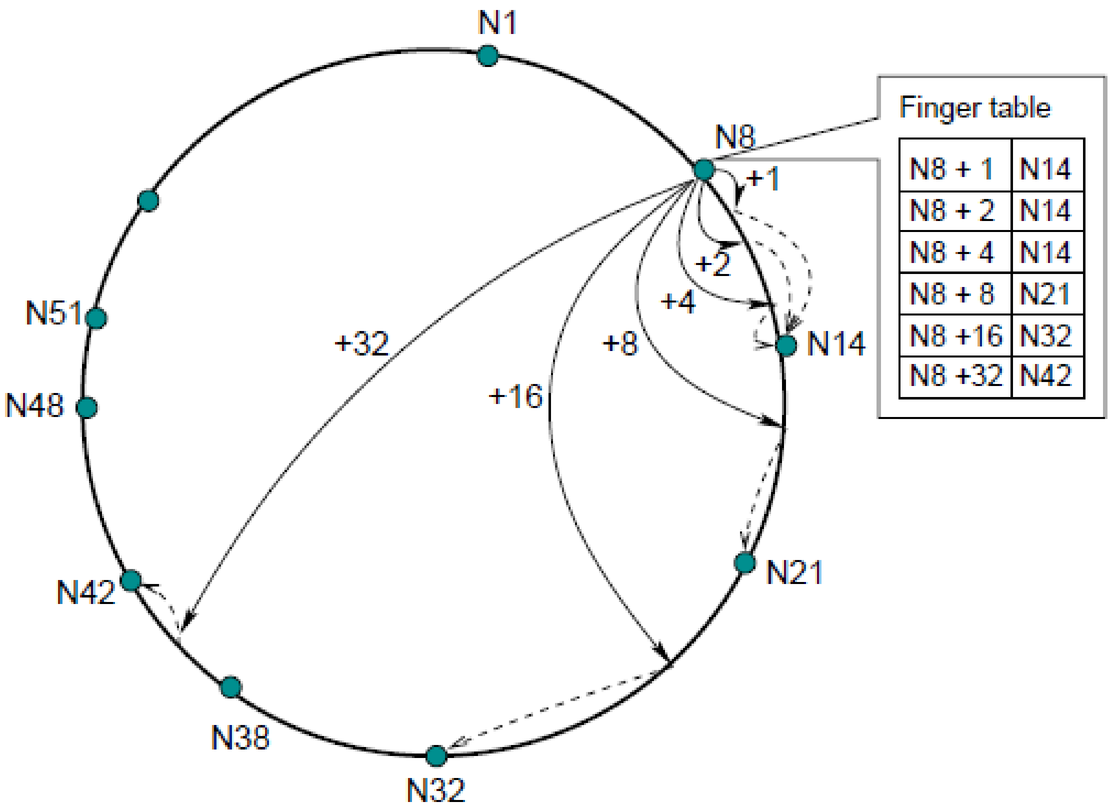
Chord

If each node only tracks its successor on the ring, the lookup mechanism requires O(1) local space, but O(n) hops

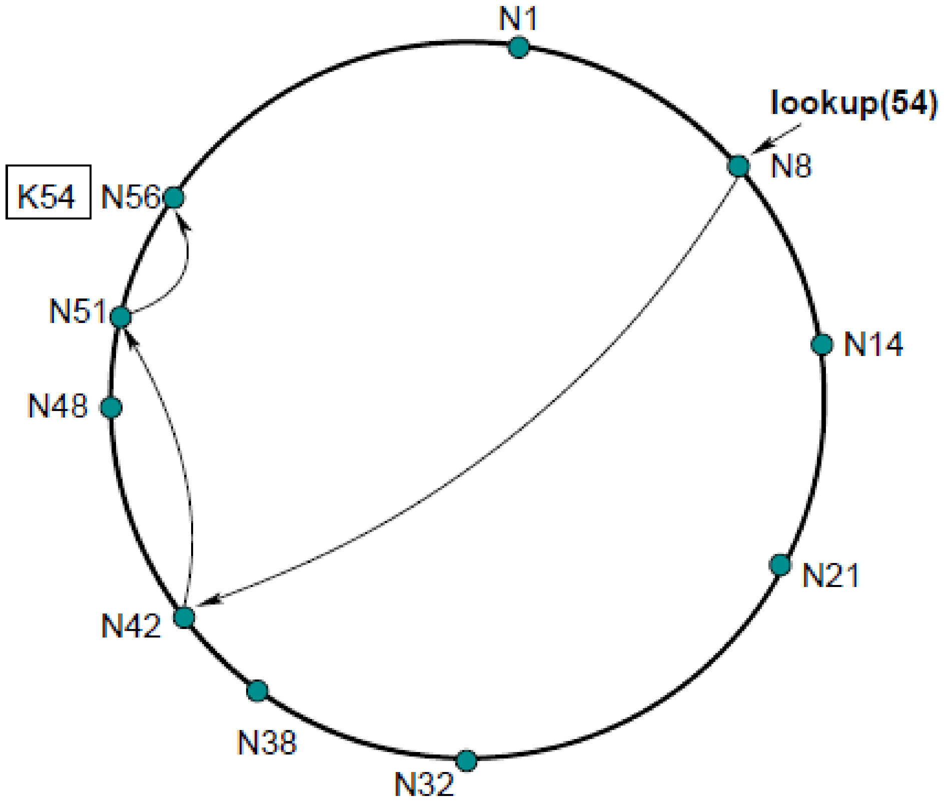


Chord

To avoid a linear search, Chord implements a faster search method by requiring each node to keep a finger table containing up to m entries



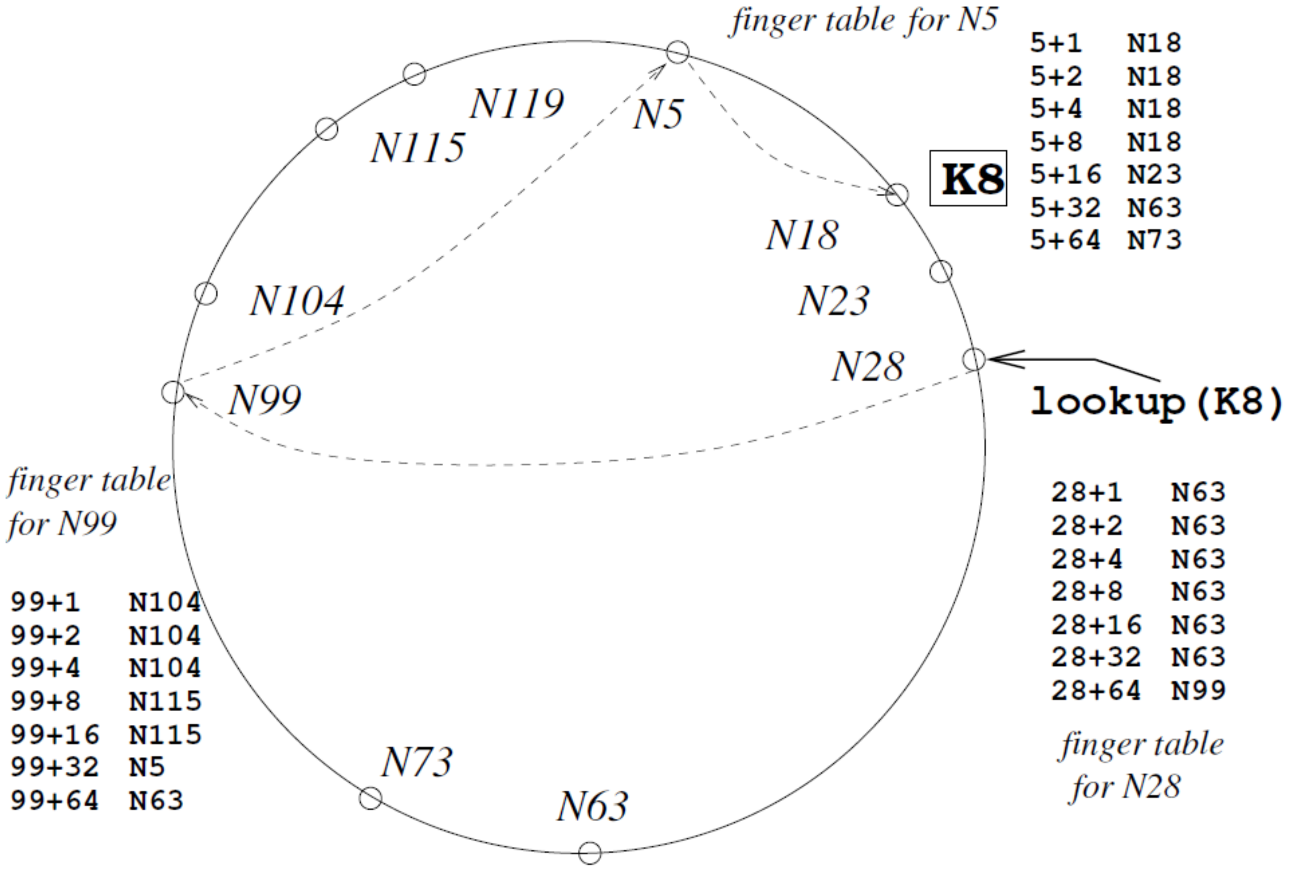
N8 + 1	N14
N8 + 2	N14
N8 + 4	N14
N8 + 8	N21
N8 +16	N32
N8 +32	N42



Shortcuts - chords

Chord

This scalable lookup algorithm uses O(log n) message hops at the cost of O(log n) space in the local routing tables



finger table for N5

5+1	N18
5+2	N18
5+4	N18
5+8	N18
5+16	N23
5+32	N63
5+64	N73

finger table for N99

99+1	N104
99+2	N104
99+4	N104
99+8	N115
99+16	N115
99+32	N5
99+64	N63

28+1	N63
28+2	N63
28+4	N63
28+8	N63
28+16	N63
28+32	N63
28+64	N99

finger table for N28

Applications

A new generation of distributed **applications** have been proposed on top of structured P2P systems, validating them as novel application infrastructures

- Decentralized file systems: CFS (Chord), Mnemosyne (Tapestry), OceanStore (Tapestry), PAST (Pastry)
- Application level multicast: CAN-MC (CAN), Scribe (Pastry), Bayeux (Tapestry)

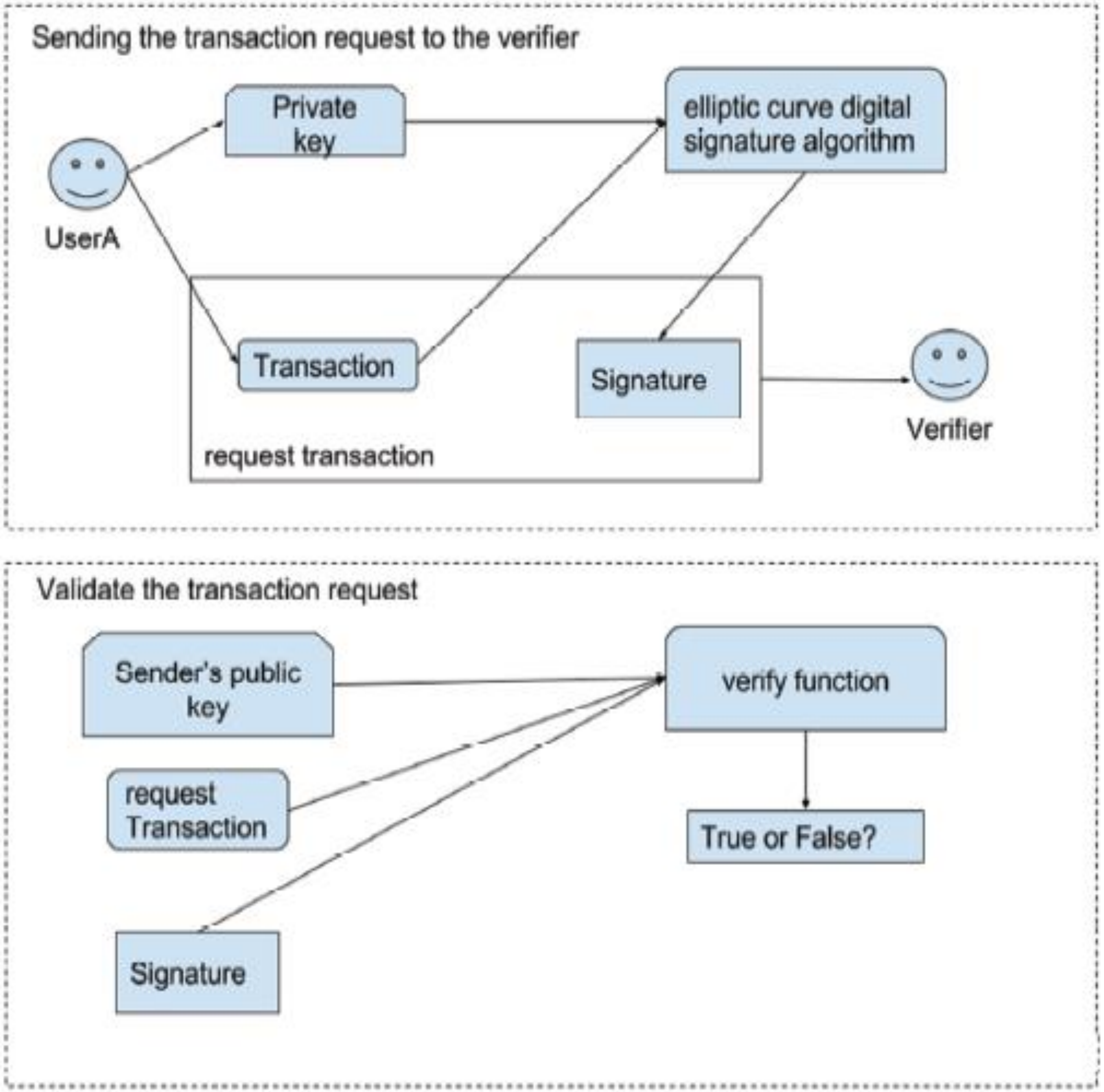
4. Bitcoin

Nakamoto, S. (2000). Bitcoin: A Peer-to-Peer Electronic Cash System, October 31, http://bitcoin.org/bitcoin.pdf

₿

Unicode U+20BF

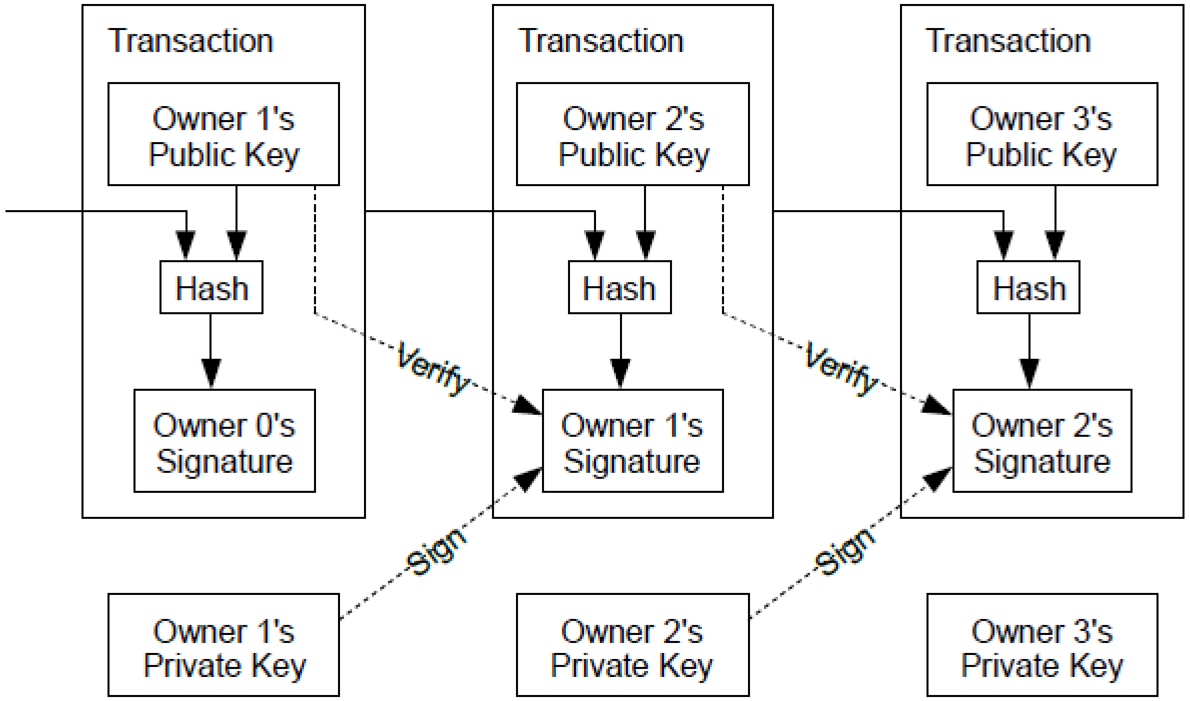
Bitcoin



A Survey about Consensus Algorithms Used in Blockchain
https://doi.org/10.3745/JIPS.01.0024

Bitcoin

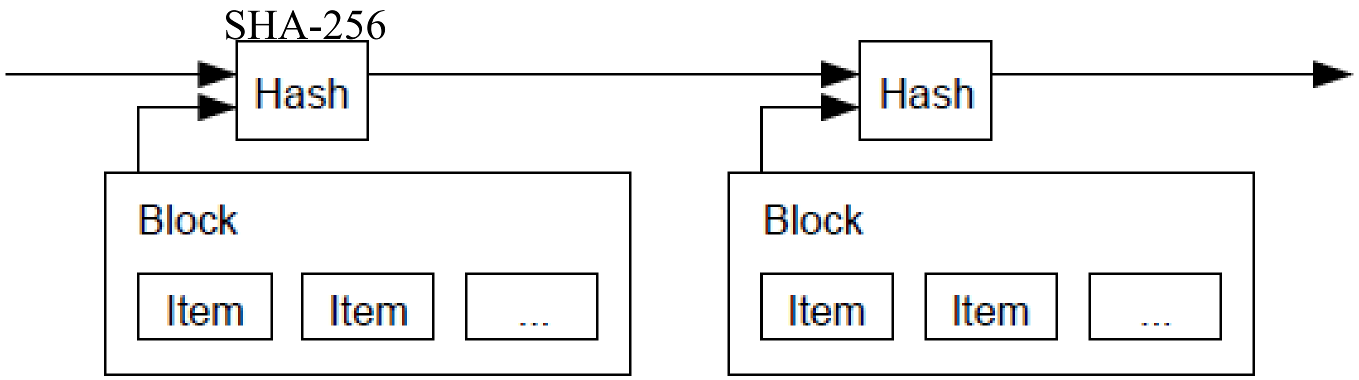
An electronic coin is defined as a chain of digital signatures



Double spending?

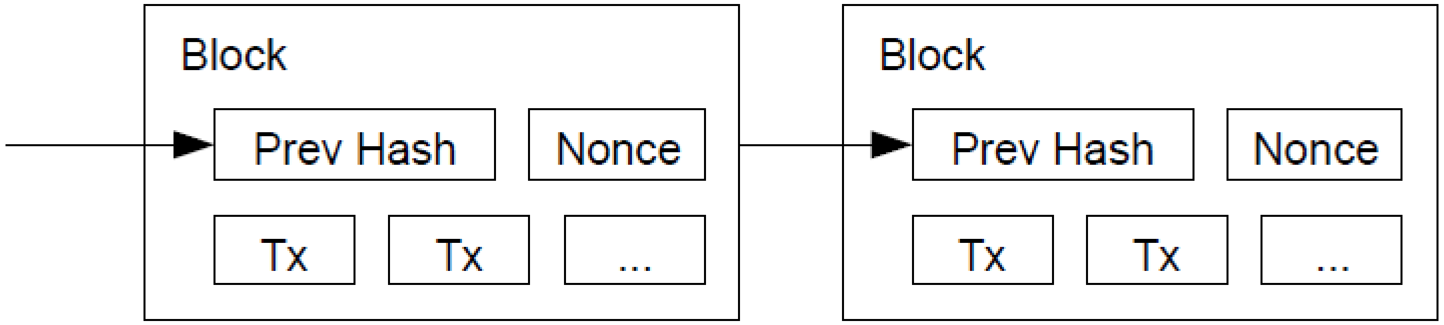
Bitcoin

The payee must know that the previous owners did not sign any earlier transactions

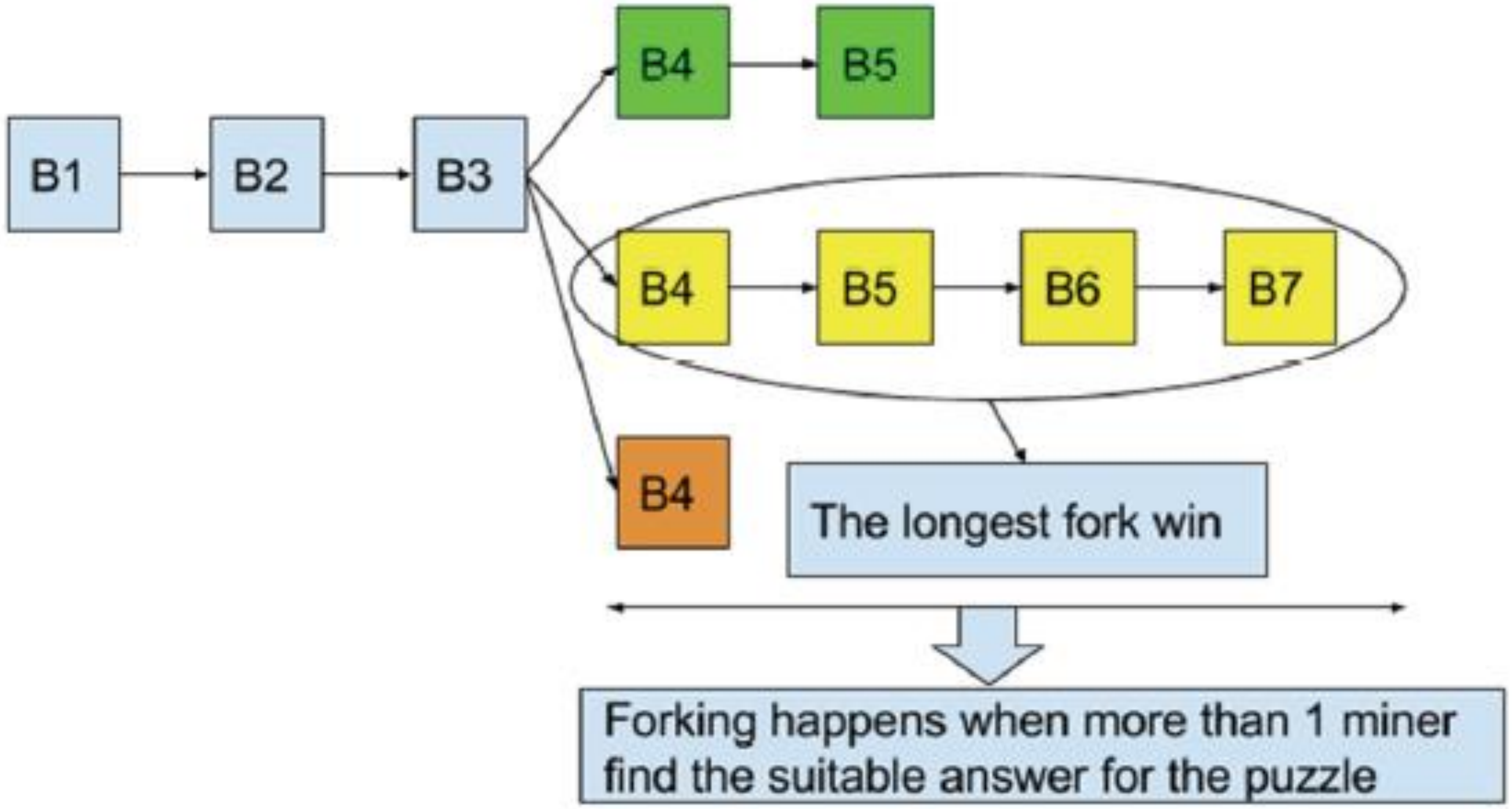


Bitcoin

Use of a proof-of-work system to implement a distributed “timestamp” server



Bitcoin


B1
B2
B3
B4
B5
B4
B5
B6
B7
B4
The longest fork win
Forking happens when more than 1 miner find the suitable answer for the puzzle

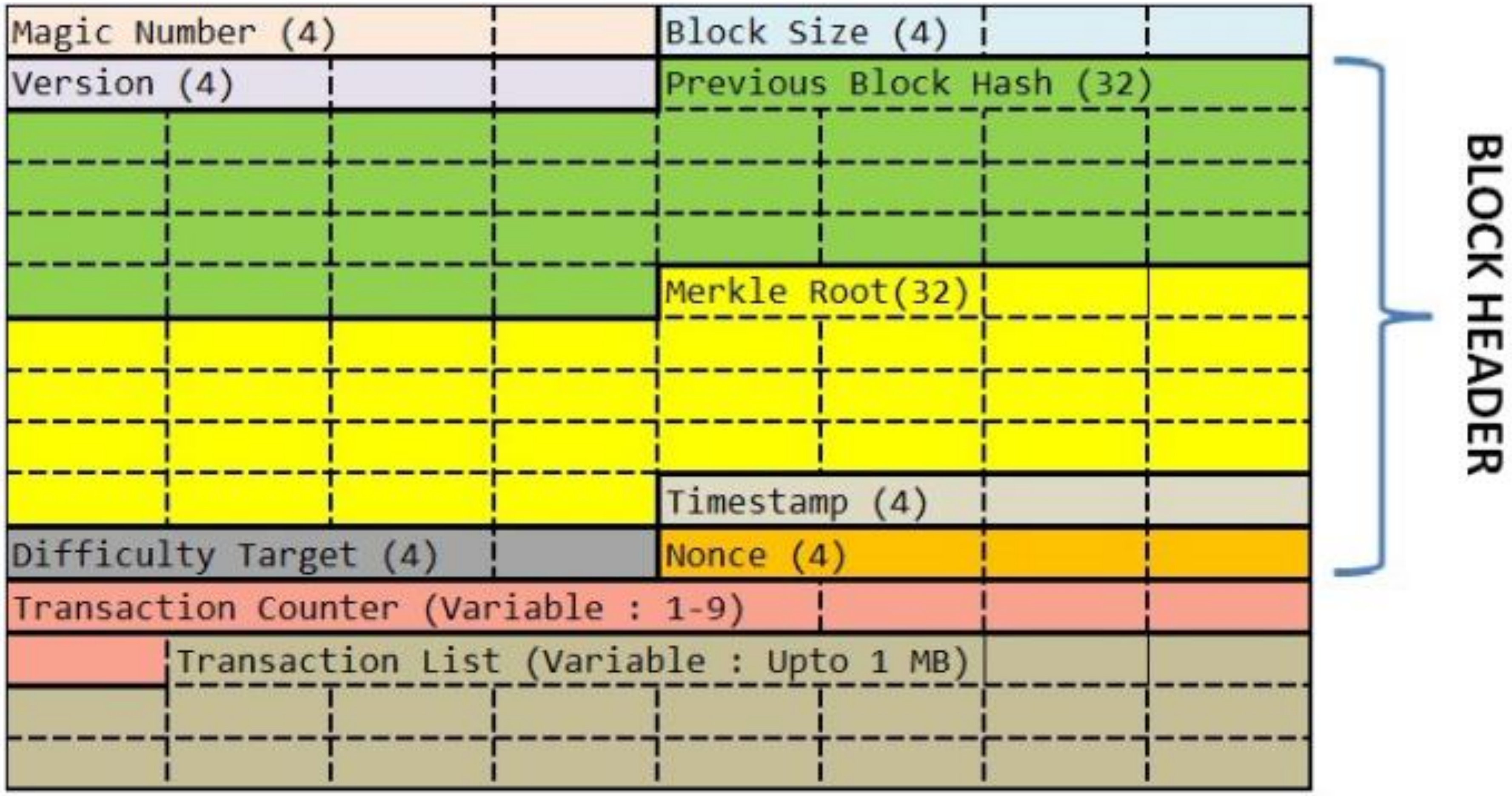
A Survey about Consensus Algorithms Used in Blockchain
https://doi.org/10.3745/JIPS.01.0024

Bitcoin

The steps to run the network are as follows:

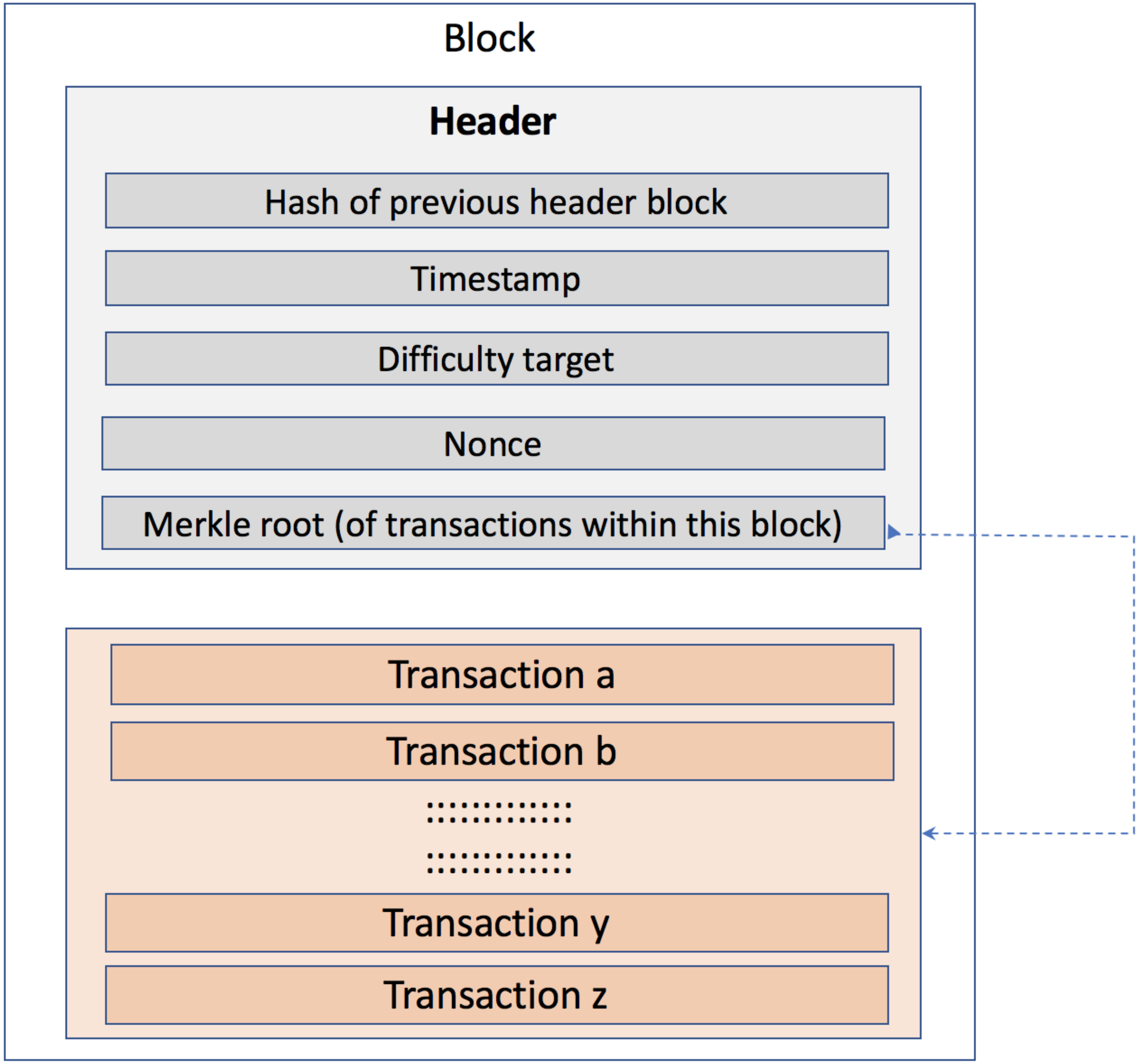
1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. Each node works on finding a difficult proof-of-work for its block.
4. When a node finds a proof-of-work, it broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent.
6. Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash.

Bitcoin



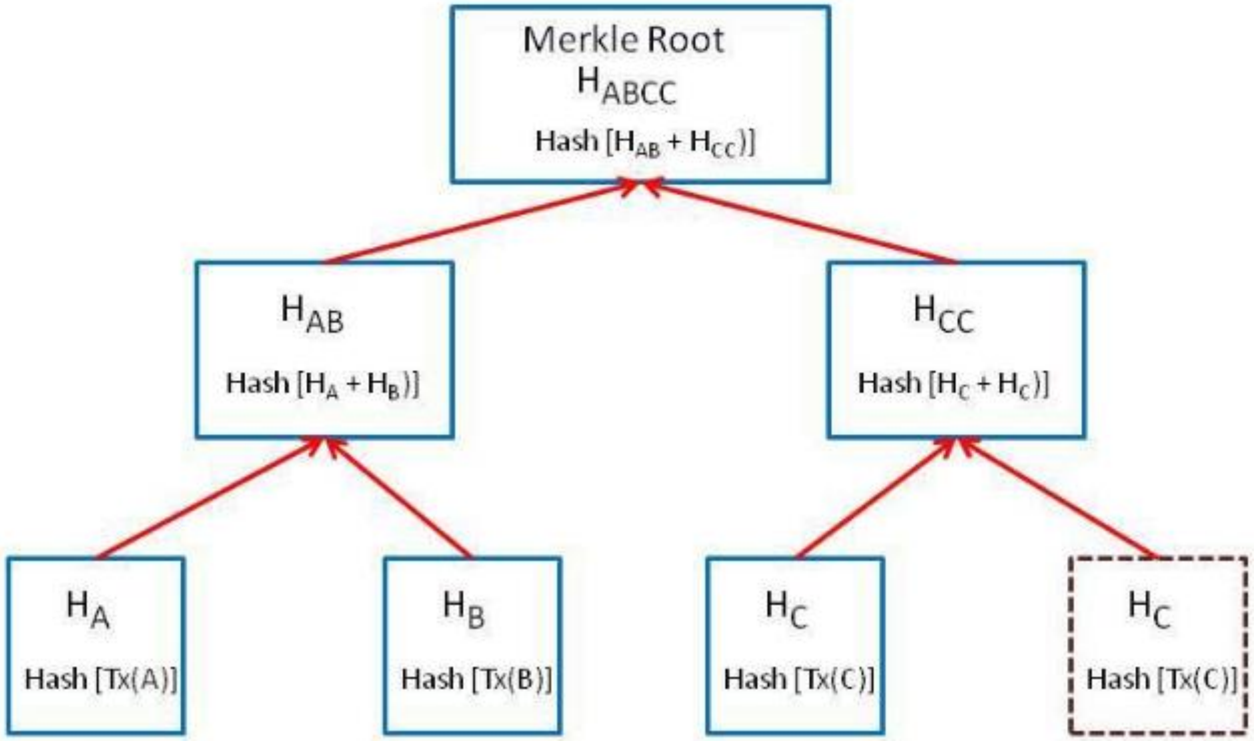
https://medium.com/all-things-ledger/bitcoins-implementation-of-blockchain-2be713f662c2

Bitcoin

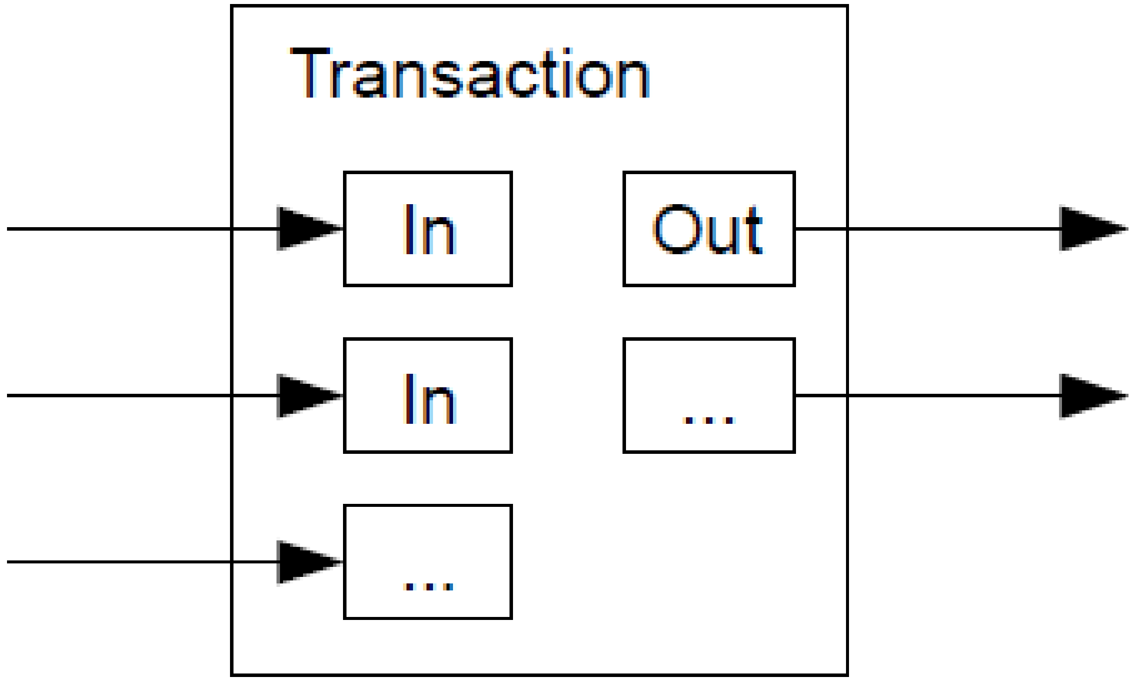


https://luxsci.com/blog/understanding-blockchains-and-bitcoin-technology.html

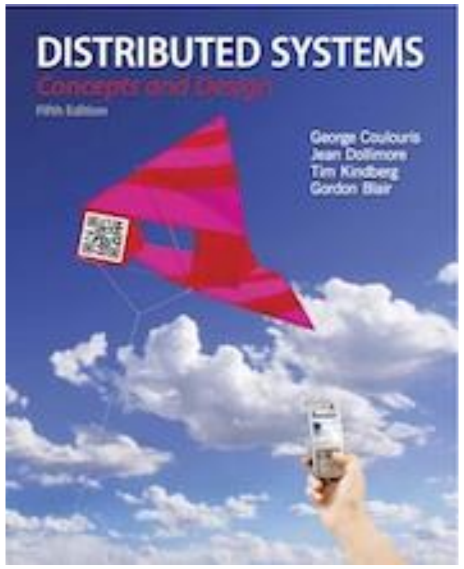
Bitcoin


Merkle Root
H_{ABCC}
Hash [H_{AB} + H_{CC})]
H_{AB}
Hash [H_A + H_B)]
H_{CC}
Hash [H_C + H_C)]
H_A
Hash [Tx(A)]
H_B
Hash [Tx(B)]
H_C
Hash [Tx(C)]
H_C
Hash [Tx(C)]

Bitcoin



Peer-to-Peer Systems



From **Coulouris, Dollimore, Kindberg and Blair**

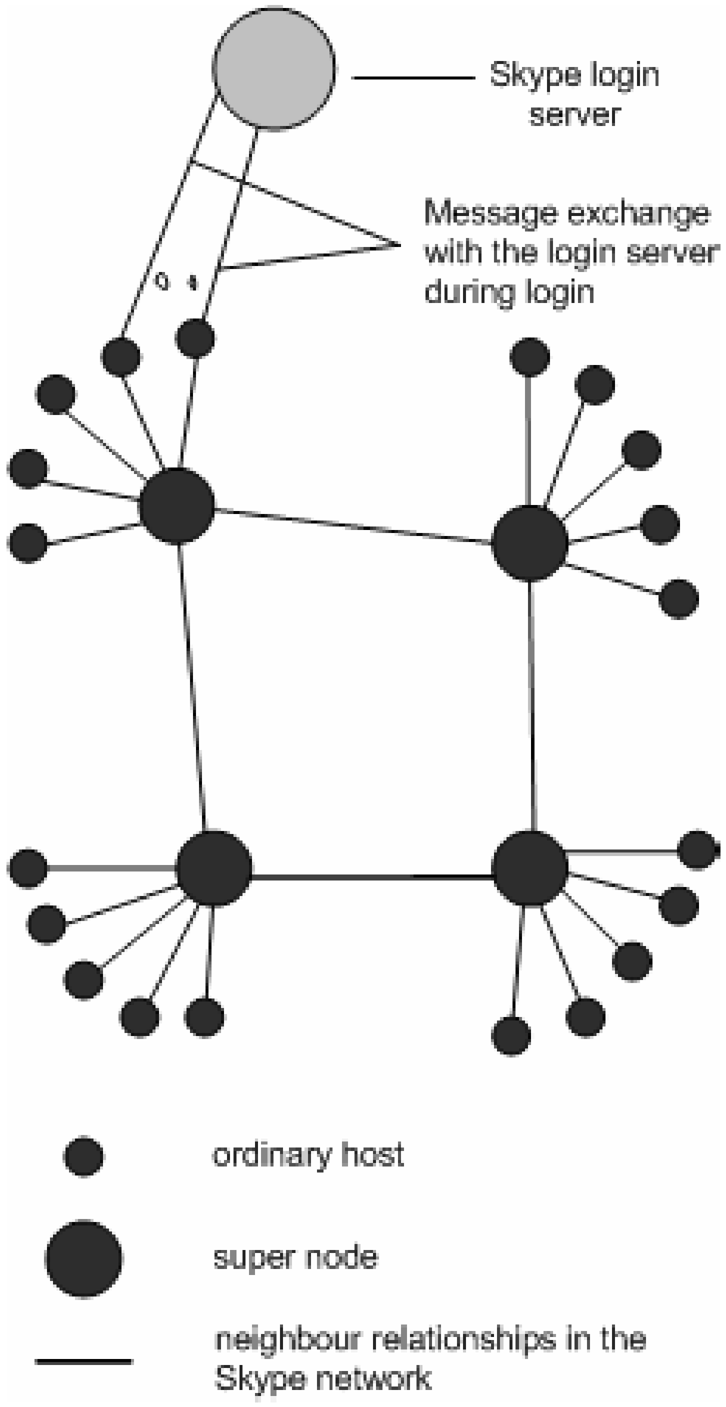
Distributed Systems:

Concepts and Design

Edition 5, © Addison-Wesley 2012

Skype

Baset, S.A. and Schulzrinne, H.G. (2006). An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol, *INFOCOM'06*, pp. 1-11, April 23-29, Barcelona, Spain.


Skype login server
Message exchange with the login server during login
ordinary host
super node
neighbour relationships in the Skype network

Skype

The Skype network is an overlay network and thus each client (ordinary host) needs to build and refresh a table of reachable nodes

This table (host cache) contains IP address and port number of super nodes

A client must connect to a super node and must register itself with the login server for a successful login

Skype

A client making a phone call contacts super nodes from its host cache, asking them to help find the user

Super nodes return a list of nodes to contact

The client contacts those nodes (if unsuccessful, the client asks for more nodes)