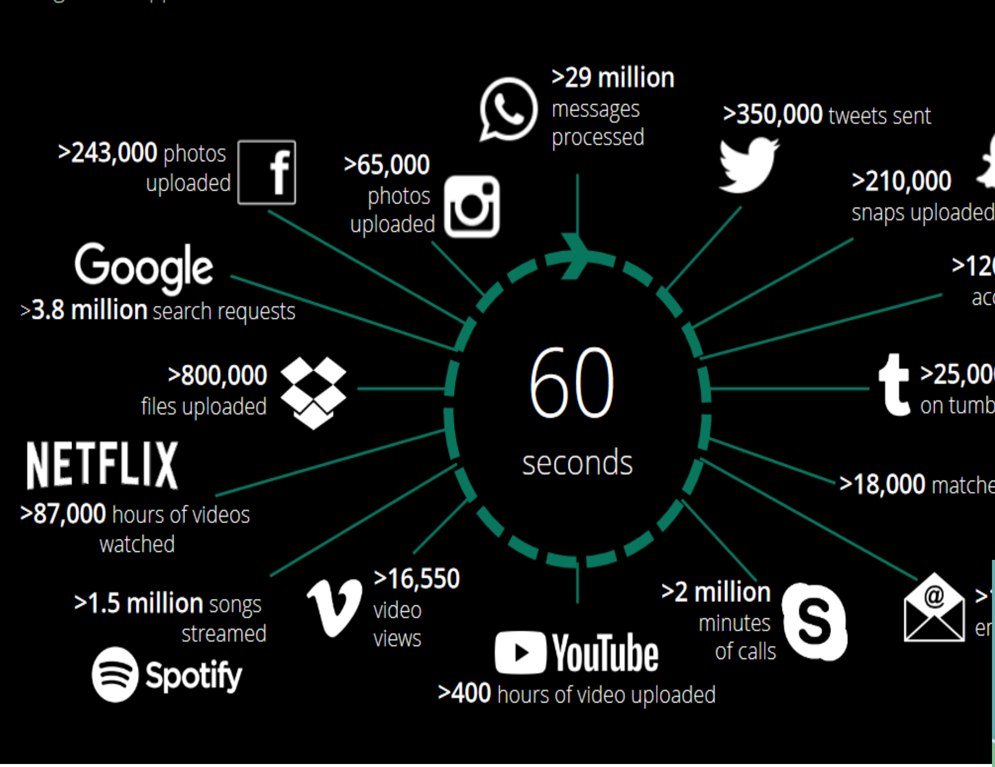
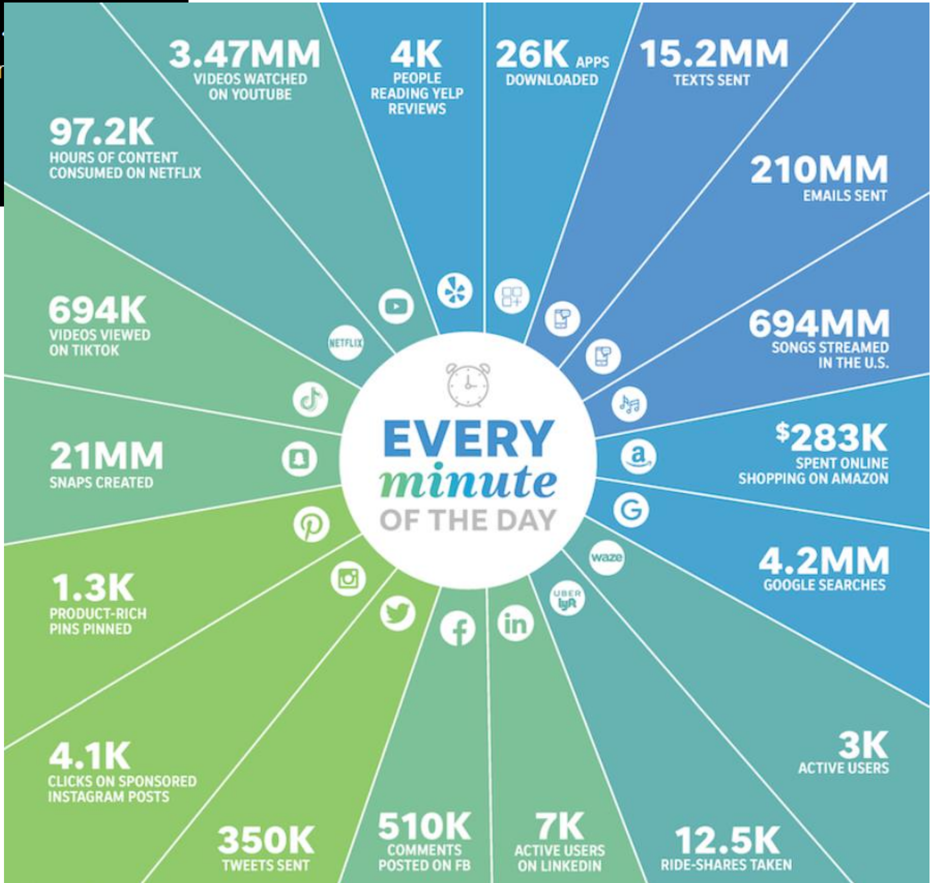
Peer-to-Peer Systems and Networks

Mestrado em Engenharia de Computadores e Telemática

2022/2023

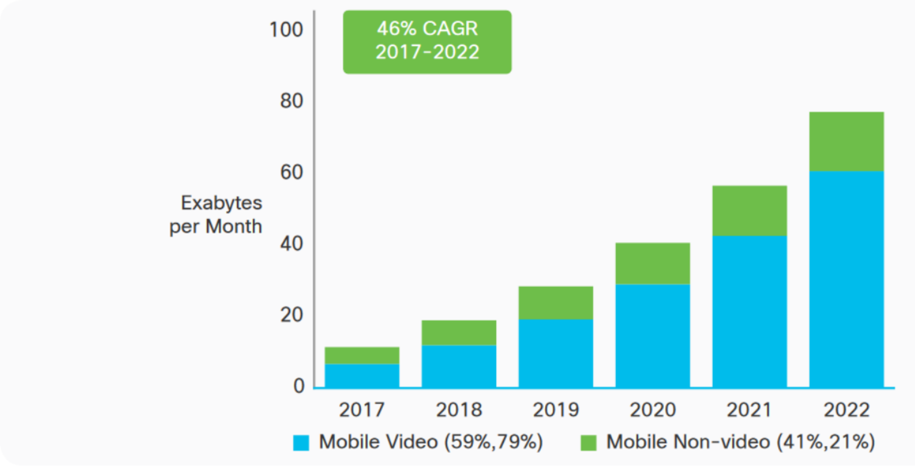

>29 million messages processed
>350,000 tweets sent
>243,000 photos uploaded
>65,000 photos uploaded
>210,000 snaps uploaded
>120 new accounts
Google
>3.8 million search requests
60 seconds
>800,000 files uploaded
>25,000 posts on tumblr
NETFLIX
>87,000 hours of videos watched
>18,000 matches tinder
>1.5 million songs streamed
Spotify
>16,550 video views
>2 million minutes of calls
>
er
YouTube
>400 hours of video uploaded


EVERY minute OF THE DAY
3.47MM VIDEOS WATCHED ON YOUTUBE
4K PEOPLE READING YELP REVIEWS
26K APPS DOWNLOADED
15.2MM TEXTS SENT
97.2K HOURS OF CONTENT CONSUMED ON NETFLIX
210MM EMAILS SENT
694K VIDEOS VIEWED ON TIKTOK
694MM SONGS STREAMED IN THE U.S.
21MM SNAPS CREATED
$283K SPENT ONLINE SHOPPING ON AMAZON
1.3K PRODUCT-RICH PINS PINNED
4.2MM GOOGLE SEARCHES
4.1K CLICKS ON SPONSORED INSTAGRAM POSTS
3K ACTIVE USERS
350K TWEETS SENT
510K COMMENTS POSTED ON FB
7K ACTIVE USERS ON LINKEDIN
12.5K RIDE-SHARES TAKEN
NETFLIX
waze
UBER lyft

Motivation

- IP based networks
- Web based applications have become the norm for corporate internal networks and many business-to-business interactions
- Large acceptance and explosive growth
  - Serious performance problems
  - Degraded user experience

  For a large set of applications, including VIDEO access
- Improving the performance of networked applications
  - Use many sites at different points within the network
    - Stand alone servers
    - Routers



Note: Figures in parentheses refer to 2017 and 2022 traffic share.
Source: Cisco VNI Mobile, 2019

Content distribution networks

- Client attempts to access the main server site for an application
- It is redirected to one of the other sites
- Each site caches information
  - Avoid going to the main server to get the information/application
- Access a closelly located site
  - Avoid congestion on the path to the main server
- Set of sites used to improve the performance of web-based applications collectivelly
  - Content distribution network

What is a CDN?

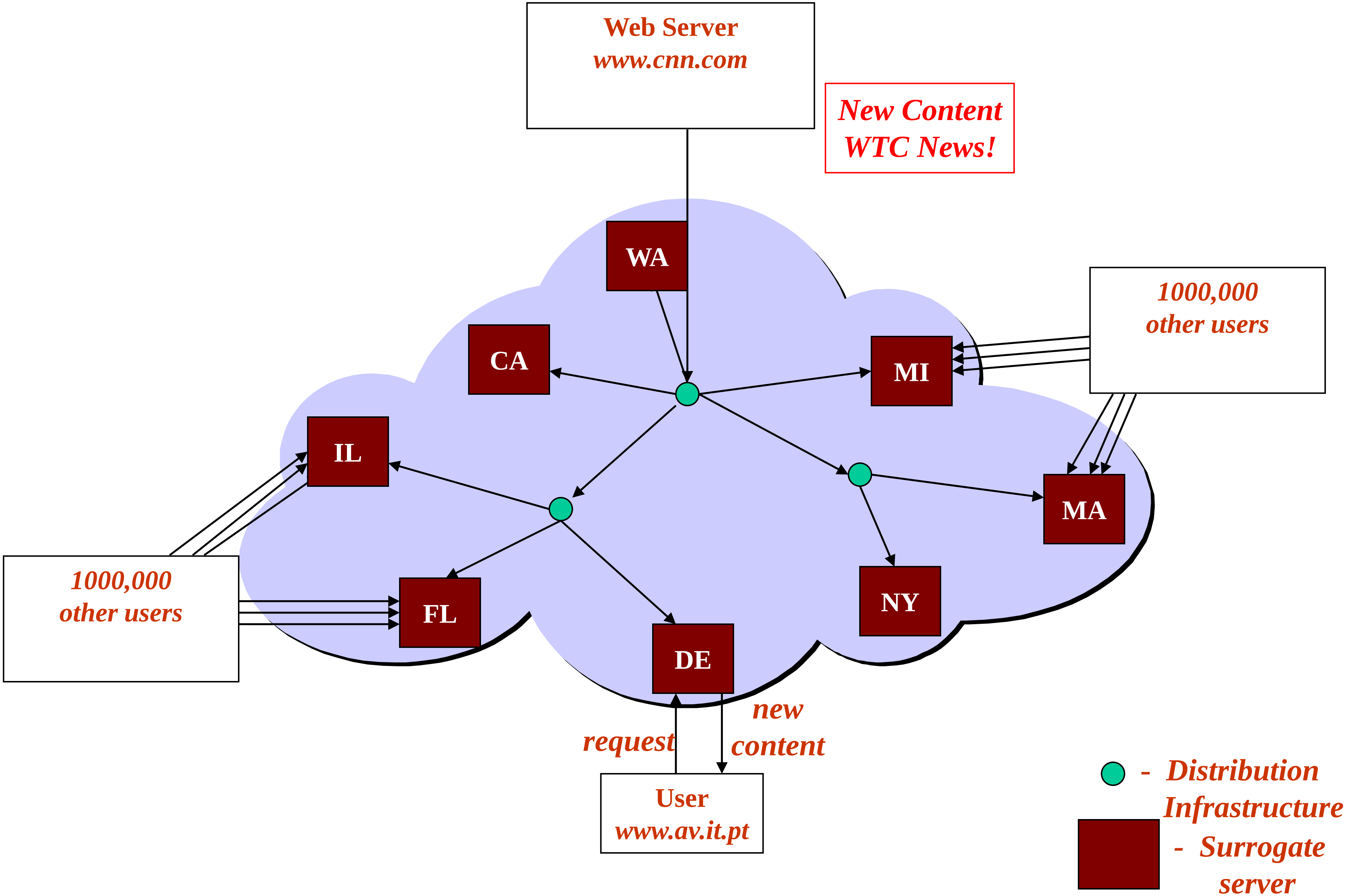
- Content Delivery Network
  - Also sometimes called Content Distribution Network
  - At least half of the world’s bits are delivered by a CDN
    - Probably closer to 80/90%

- Primary Goals
  - Create replicas of content throughout the Internet
  - Ensure that replicas are always available
  - Direct clients to replicas that will give good performance

Key Components of a CDN

- Distributed servers
  - Usually located inside of other ISPs
- High-speed network connecting them
- Clients
  - Can be located anywhere in the world
  - They want fast Web performance
- Binding clients and distributed servers
  - Something that binds clients to “nearby” replica servers

CDN Architecture



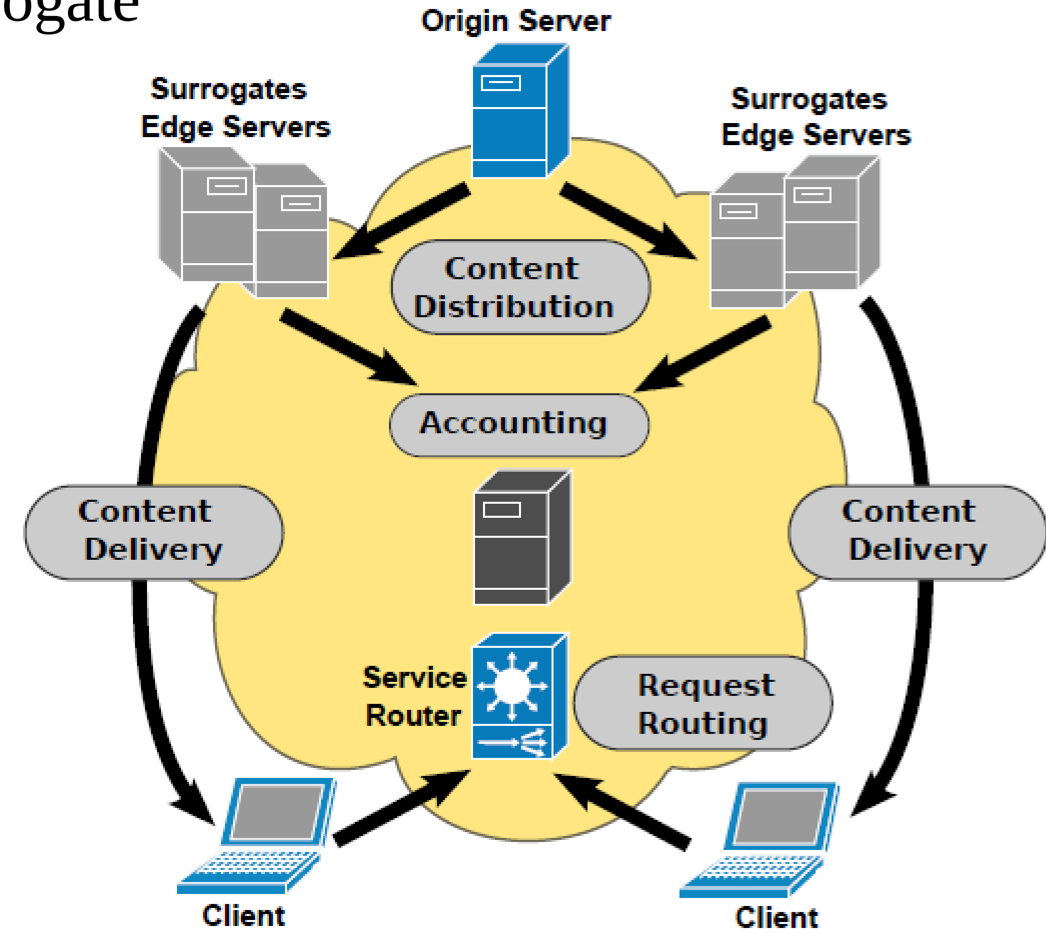
CDN Components

- *Content Delivery Infrastructure:* Delivering content to clients from surrogates

- *Request Routing Infrastructure:* Steering or directing content request from a client to a suitable surrogate

- *Distribution Infrastructure:* Moving or replicating content from content source (origin server, content provider) to surrogates

- *Accounting Infrastructure:* Logging and reporting of distribution and delivery activities



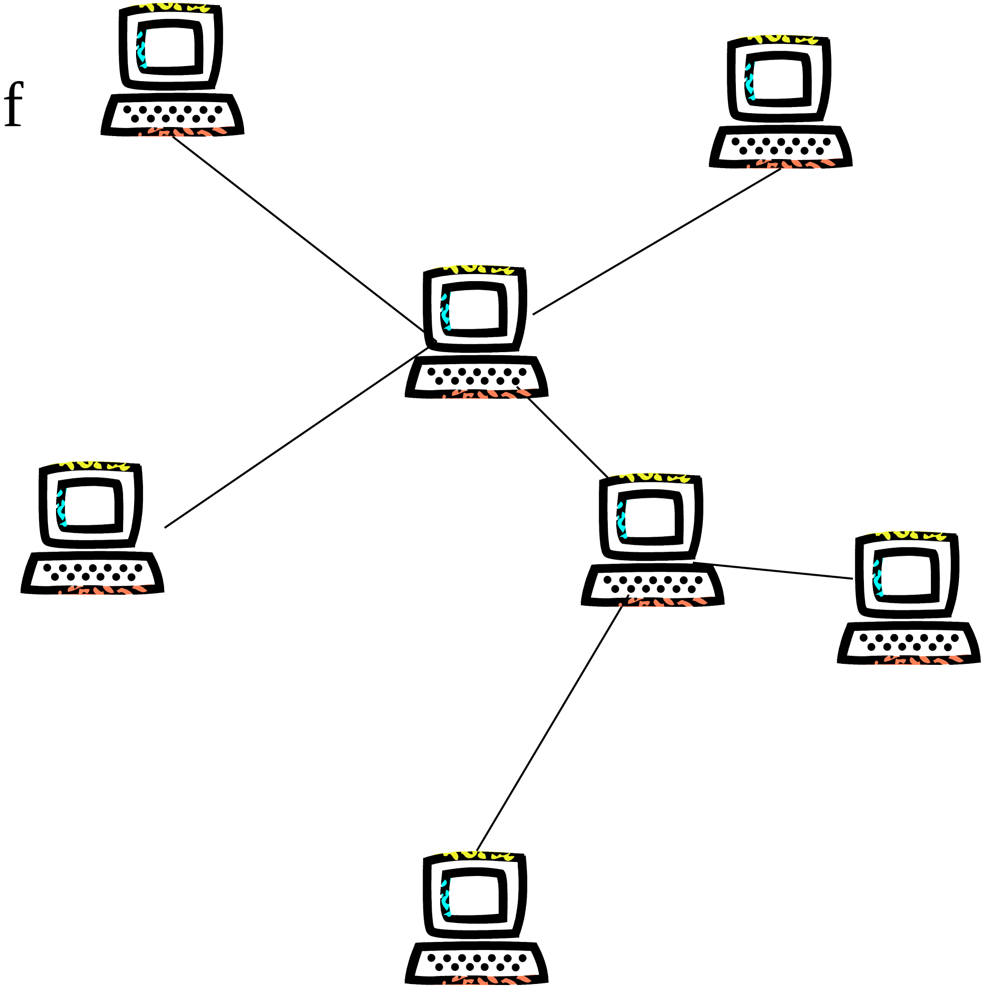
Peer-to-peer networks

Peer to peer networks

- Exploits diverse connectivity between participants in a network
- Exploits the cumulative bandwidth of network participants
- Typically used for connecting nodes via large ad-hoc connections
  - Sharing content files containing audio, video, data
  - Even real-time data, such as telephony traffic, is also passed using P2P technology (Skype)
- Pure peer-to-peer network
  - There is no notion of clients or servers
  - Equal peer nodes that simultaneously work as both "clients" and "servers" to the other nodes on the network

The P2P Model

- A peer’s resources is similar to the resources of the other participants
- P2P – peers communicating directly with other peers and sharing resources
- P2P services
  - Distributed Computing
  - File Sharing
  - Collaboration



Advantages

- Clients provide resources, including bandwidth, storage space, and computing power
- As nodes arrive and the demand on the system increases, the total capacity of the system also increases
- Distributed nature also increases robustness in case of failures by replicating data over multiple peers
  - Enable peers to find the data without relying on a centralized index server

P2P applications

- File sharing
  - Using application layer protocols
    - DirectConnect (centralized), Gnutella (flooding), BitTorrent (hybrid), IPFS
- VoIP
  - Using application layer protocols
    - SIP
- Streaming media
- Instant messaging
- Software publication and distribution
- Media publication and distribution
  - radio, video

Challenges

- Peer discovery and group management
- Data location, searching and placement
  - Search and routing
- Reliable and efficient file delivery
- Security/privacy/anonymity/trust

P2P types

- **Pure P2P** refers to an environment where all the participating nodes are peers
  - No central system controls, coordinates, or facilitates the exchanges among peers
- **Hybrid P2P** refers to an environment where there are servers which enable peers to interact with each other
  - The degree of central system involvement varies with the application
  - Different peers may have different functions (simple nodes, routers, rendezvous)

Used depending on the application

Simple Peers

- Single end user, allowing that user to provide services from his device and consuming services provided by other peers on the network
  - Will usually be located behind a firewall, separated from the network at large; peers outside the firewall will probably not be capable of directly communicating with the simple peer located inside the firewall.
  - Because of their limited network accessibility, simple peers have the least amount of responsibility in any P2P network.

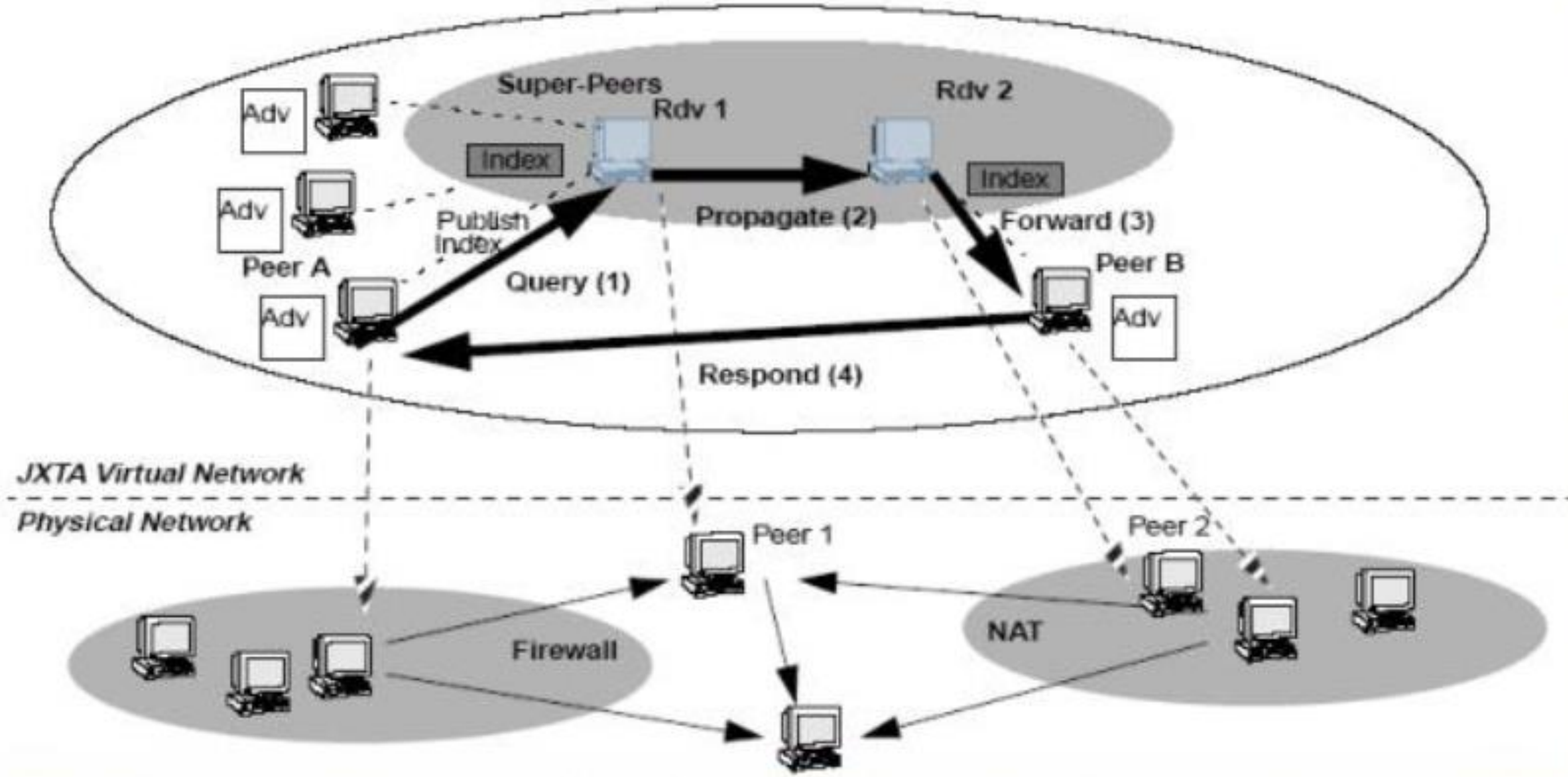
- They are not responsible for handling communication on behalf of other peers.

Rendez-vous Peers

- Gathering or meeting place
  - Provides peers with a network location to use to discover other peers and peer resources.
- Peers issue discovery queries to a rendez-vous peer, and the rendez-vous provides information on the peers it is aware of on the network.
- May cache information on peers for future use or by forwarding discovery requests to other rendez-vous peers.
  - Improve responsiveness, reduce network traffic, and provide better service to simple peers.

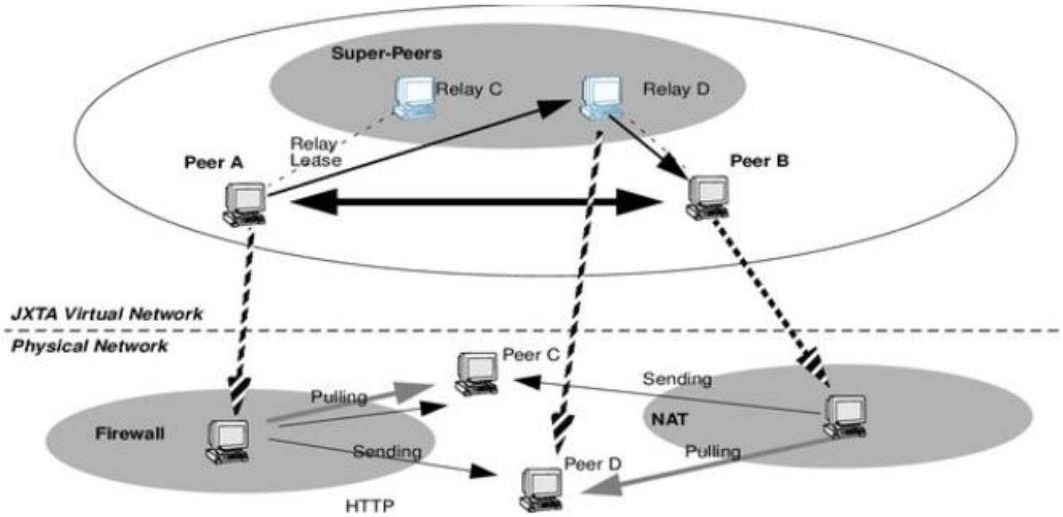
- Usually outside a private internal network's firewall. A rendez-vous could exist behind the firewall, but it would need to be capable of traversing the firewall using either a protocol authorized by the firewall or a router peer outside the firewall.

Rendez-vous Peers



Router (Relay) Peers

- A router peer provides a mechanism for peers to communicate with other peers separated from the network by firewall or Network Address Translation (NAT) equipment.
- Peers outside the firewall to communicate with a peer behind the firewall, and vice versa.
- A relay is not necessarily a rendez-vouz peer
  - Relay is on the data stream
  - Rendez-vous is always on the discovery path (and maybe in the data stream).



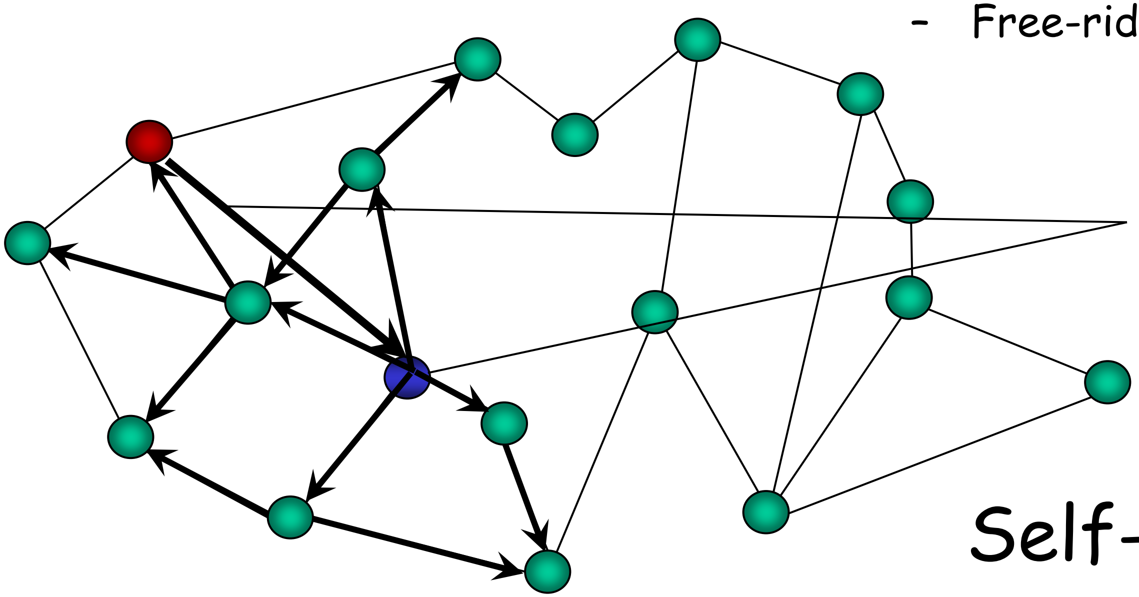
Structured vs Unstructured

- Unstructured P2P networks
  - Formed when the overlay links are established arbitrarily.
  - If a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to find as many peers as possible that share the data
    - The queries may not always be resolved
      - If a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful
    - Flooding causes a high amount of signalling traffic in the network
    - Gnutella and FastTrack/KaZaa, BitTorrent
- Structured P2P networks
  - Globally consistent protocol (logic) to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare
  - The most common type of structured P2P network is the Distributed Hash Table (DHT)
    - A variant of consistent hashing is used to assign ownership of each file to a particular peer
    - Chord, Pastry, Tapestry, CAN, Tulip, Kadmelia, BitTorrent (trackerless), IPFS

Fully Decentralized Information Systems

- P2P file sharing
  - Global scale application
- Example: Gnutella
  - 40.000 nodes, 3 Mio files (August 2000)
  - 3M nodes (Jan 2006)

- Strengths
  - Good response time, scalable
  - No infrastructure, no administration
  - No single point of failure
- Weaknesses
  - High network traffic
  - No structured search
  - Free-riding

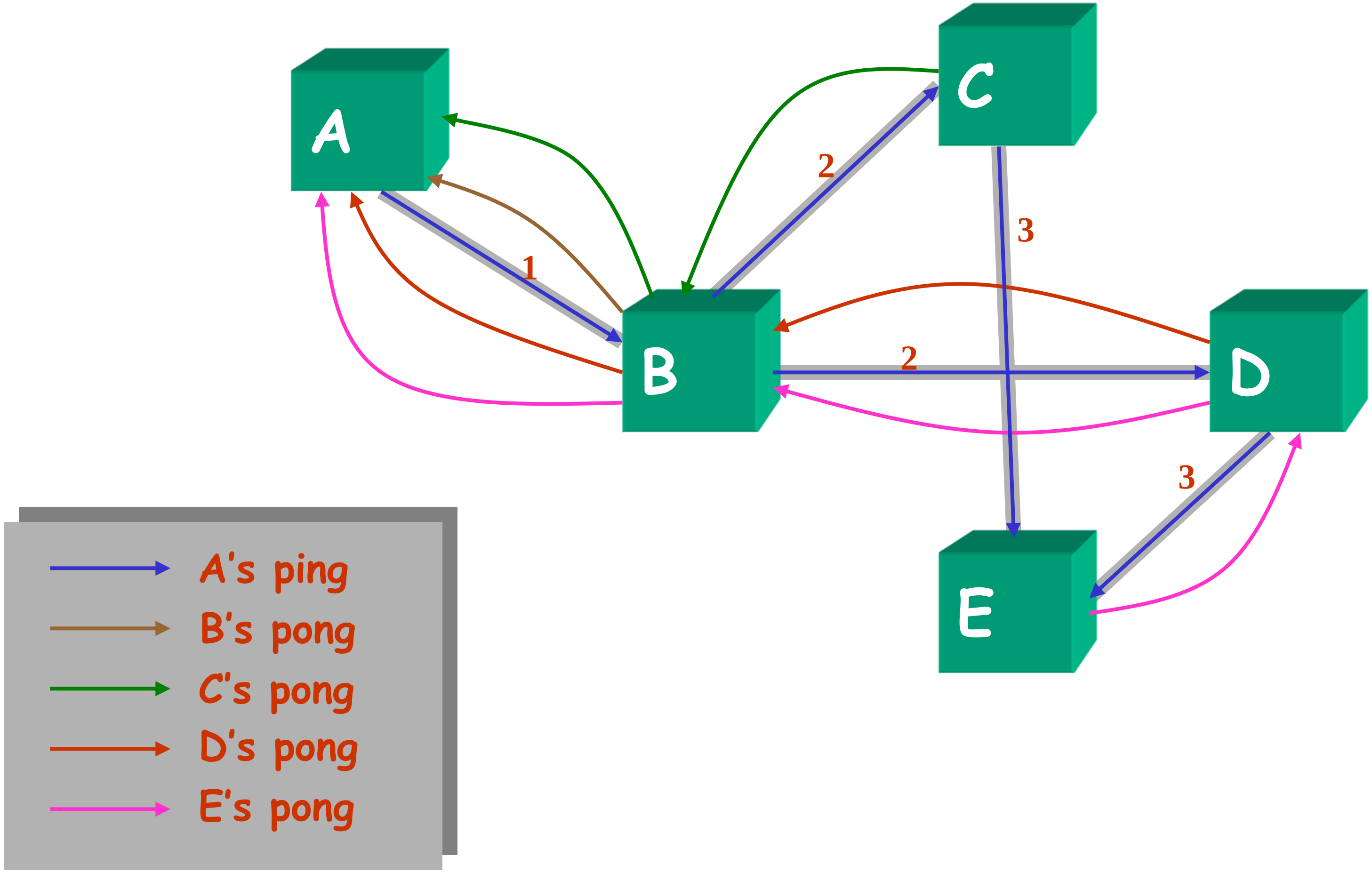


I have "brick_in_the_wall.mp3" ….

Self-organizing System

Gnutella: no servers

Gnutella: Meeting Peers (Ping/Pong)



→ A's ping

→ B's pong

→ C's pong

→ D's pong

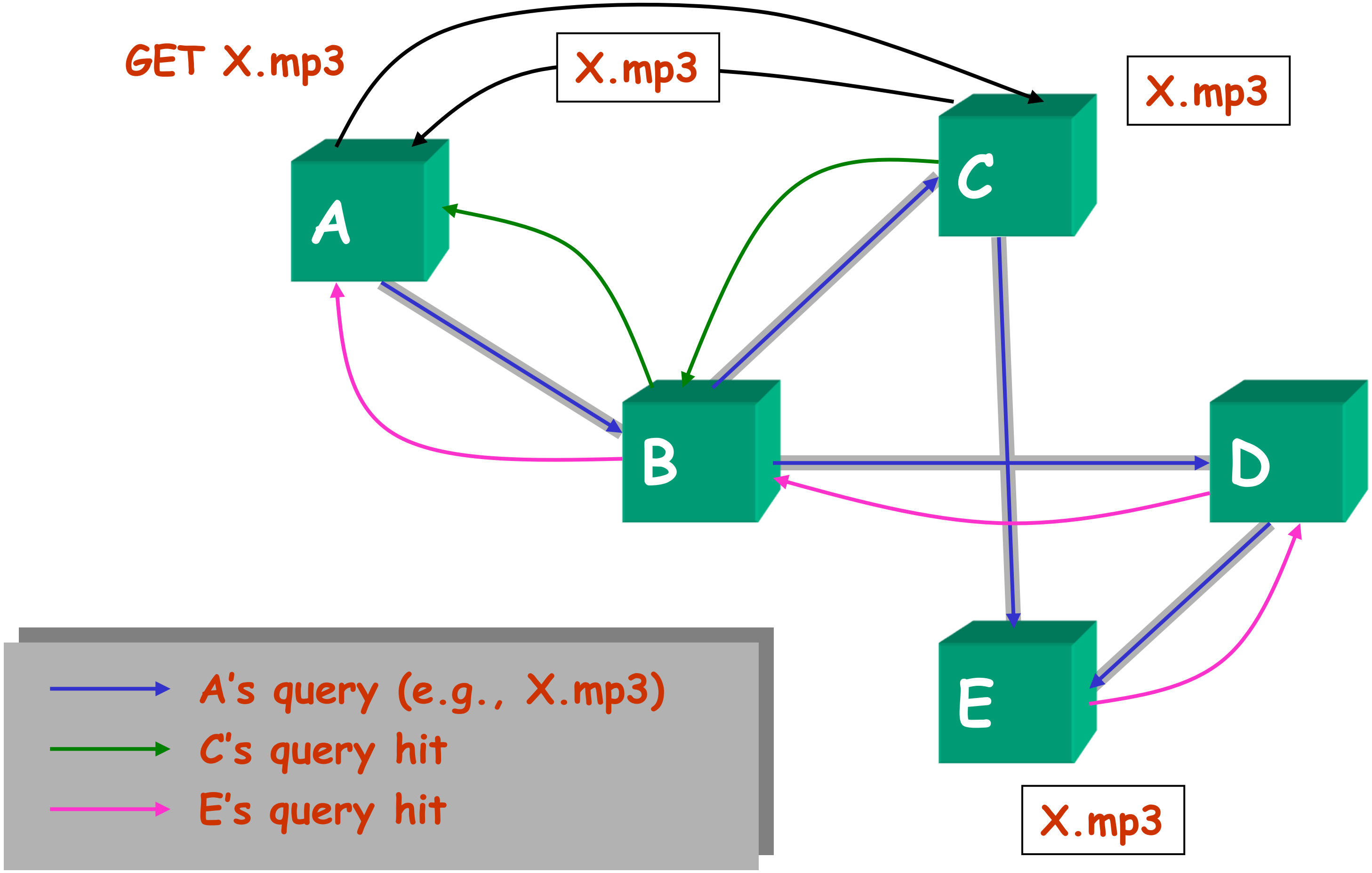
→ E's pong

Gnutella: Protocol Message Types

Type	Description	Contained Information
Ping	Announce availability and probe for other servents	None
Pong	Response to a ping	IP address and port# of responding servent; number and total kb of files shared
Query	Search request	Minimum network bandwidth of responding servent; search criteria
QueryHit	Returned by servents that have the requested file	IP address, port# and network bandwidth of responding servent; number of results and result set
Push	File download requests for servents behind a firewall	Servent identifier; index of requested file; IP address and port to send file to

Gnutella: Searching

(Query/QueryHit/GET)



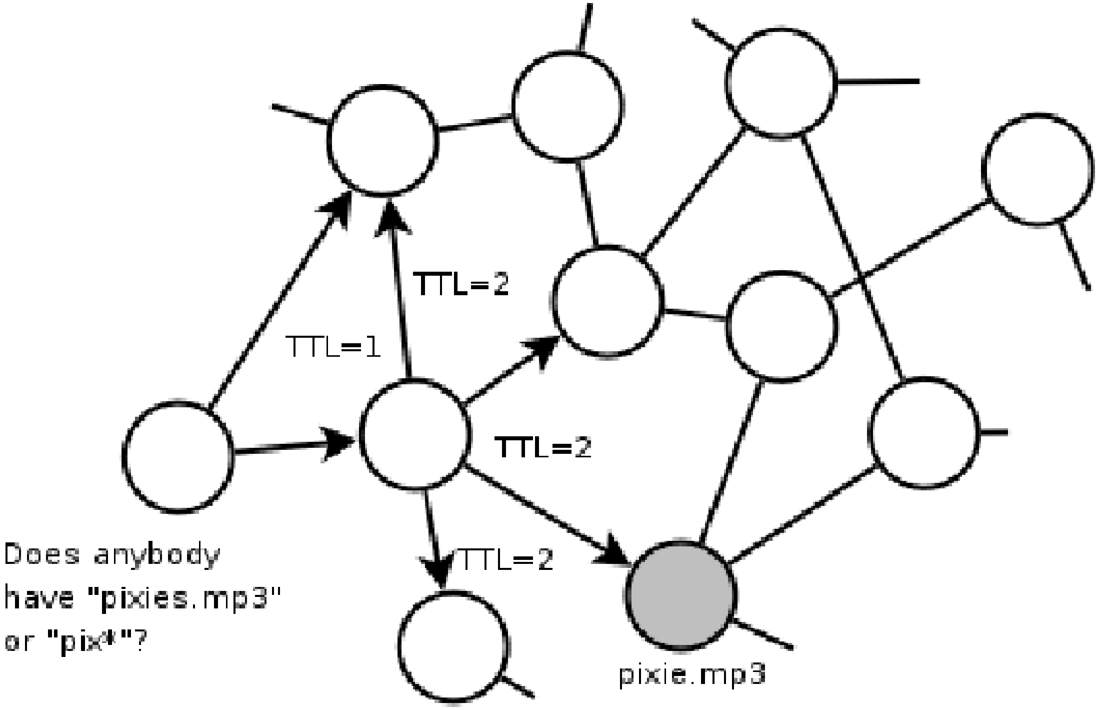
→ A's query (e.g., X.mp3)

→ C's query hit

→ E's query hit

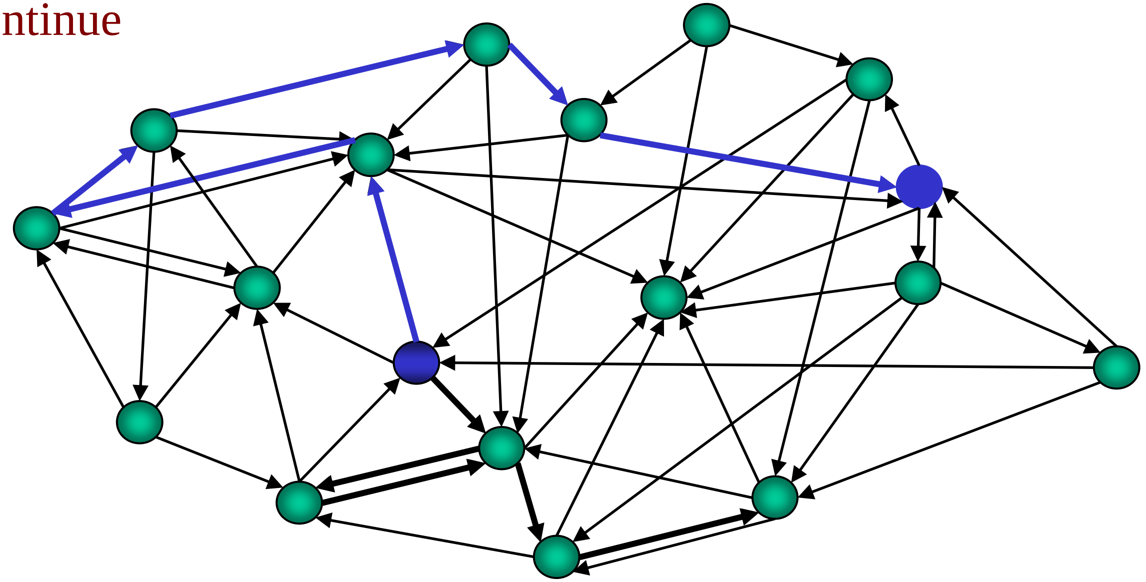
Searching in Gnutella (structureless)

- Queries are flooded to neighbours, have a TTL, and are forwarded only once
- Query may obtain several responses indicating which peers provide the requested file. Among those it selects one, and directly contacts it in order to download the file.
  - Can we search using fewer packets?



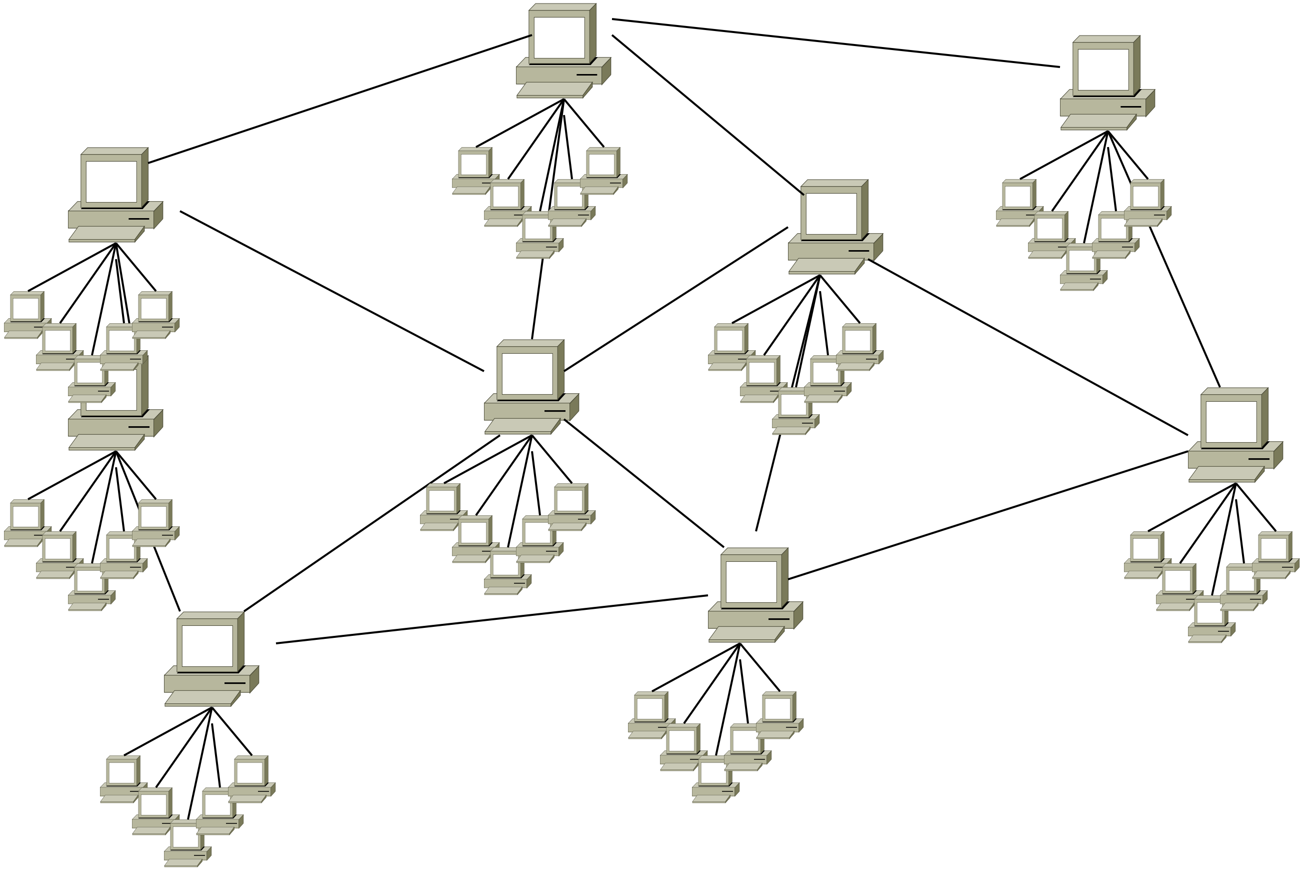
Improvements of Message Flooding

- Expanding Ring
  - start search with small TTL (e.g. TTL = 1)
  - if no success iteratively increase TTL (e.g. TTL = TTL +2)
- k-Random Walkers
  - forward query to one randomly chosen neighbor only, with large TTL
  - start k random walkers
  - random walker periodically checks with requester whether to continue



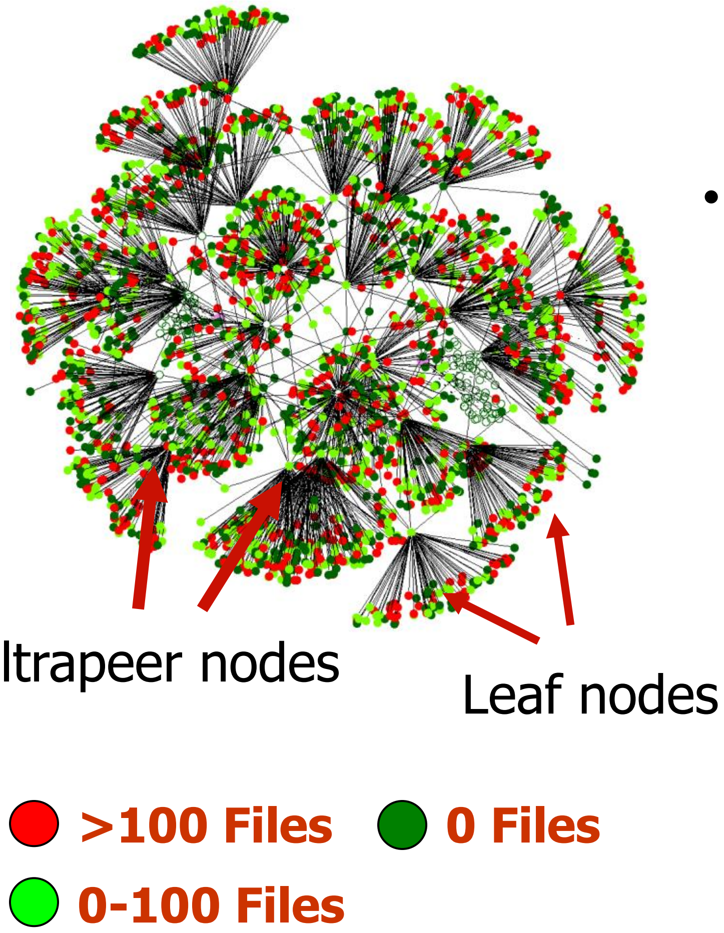
Hybrid Gnutella: “Ultrapeers”

- Ultrapeers can be installed (KaZaA) or self-promoted (Gnutella v.2)



Real Gnutella Network

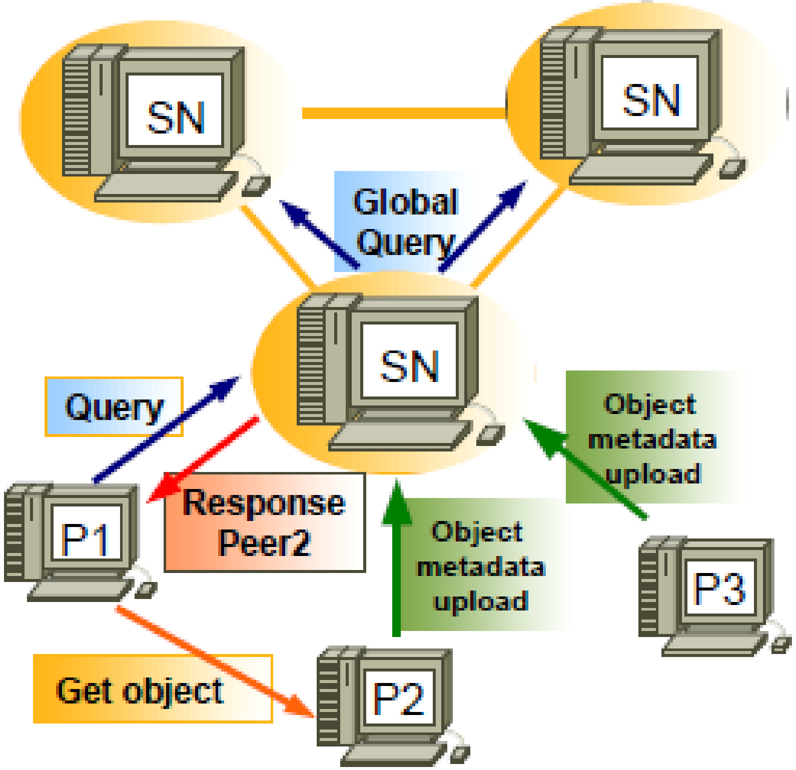
Oct 2003 Crawl of public gnutella (v.2)



- Popular open-source file-sharing network
  - ~450,000 users as of 2003
  - ~2,000,000 users as of 2022
- Ultrapeer-based Topology
  - Queries flooded among ultrapeers
  - Leaf nodes shielded from query traffic
  - Based on multiple crawlers

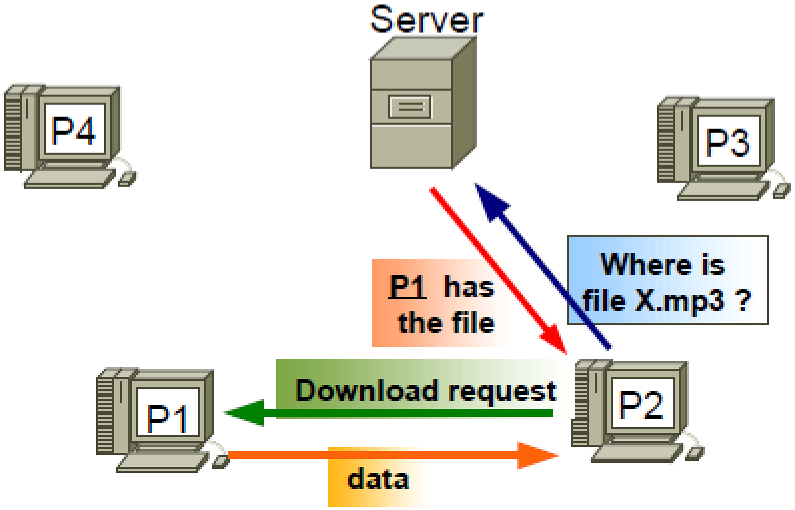
FastTrack/KaZaA

- It is an extension of the Gnutella protocol which adds super-nodes to improve scalability (~gnutella v.2)
  - A peer application hosted by a powerful machine with a fast network connection becomes automatically a super-node, effectively acting as a temporary indexing server for other slower peers
  - Communicate between each other in order to satisfy search requests
- Network architecture: Hybrid Unstructured.
- Algorithm: Flooded Requests Model (FRM)



OpenNAP/Napster

- Files (music) are on the client machine
- Servers provide search (rendez-vous) and initiate direct transfers between clientes
- OpenNAP is an extension to other types and linking servers.
- Network architecture: Hybrid Unstructured.
- Algorithm: Centralized Directory Model (CDM)



BitTorrent

- BitTorrent offloads some of the file tracking work to a central server denoted as tracker
- Uses a principal called tit-for-tat
  - To receive files, you have to give them
  - Solves the problem of leeching
- Enables fast donwloading for large files using minimum internet bandwidth

- .torrent: pointer file that directs the computer to the file it wants to download
- Swarm: group of computers simultaneously downloading or uploading the same file
- Tracker: server that manages the BitTorrent file transfer process

BitTorrent

- BitTorrent client software communicates with a tracker to find other computers running BitTorrent that have the complete file (seeders), or that have a portion of the file (currently donwnloading the file)

- The tracker identifies the swarm: this group of computers

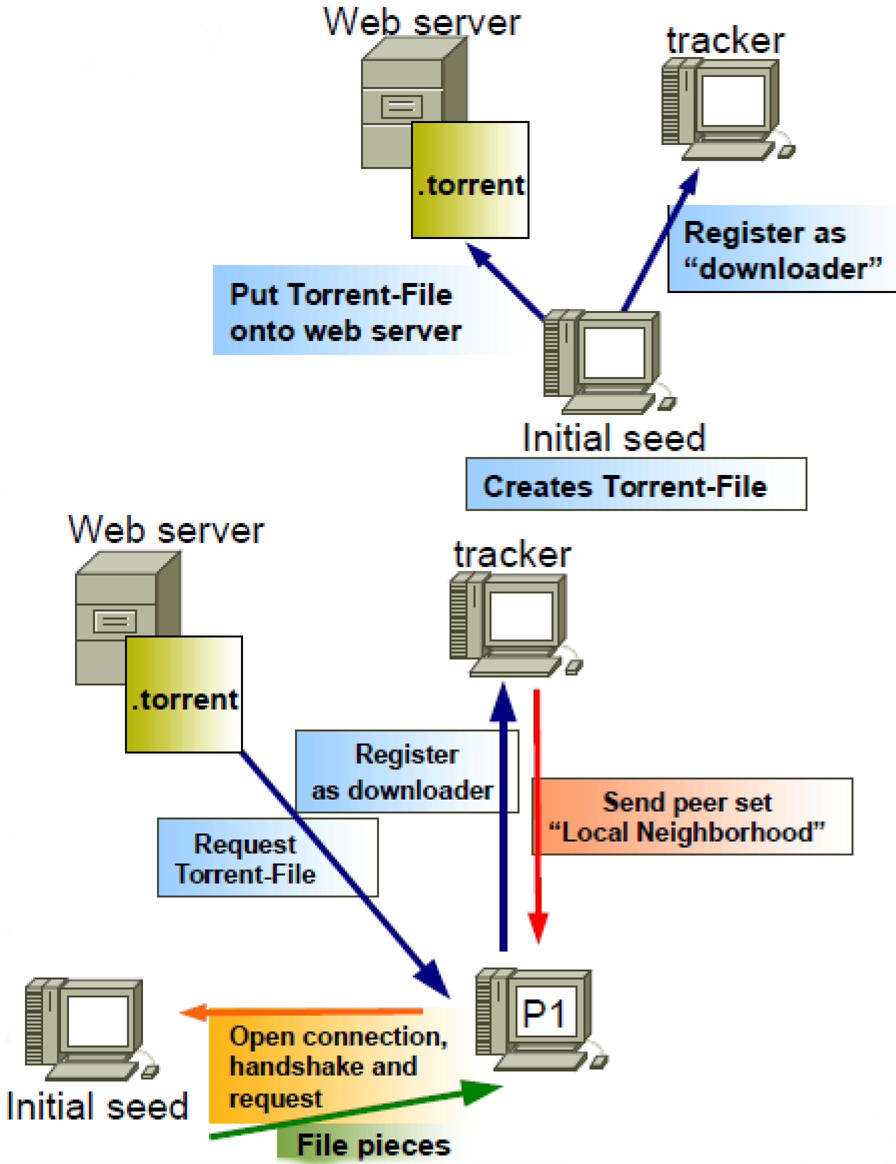
- The tracker helps the client software to trade pieces of the file with other computers in the swarm

- The computer receives multiple pieces of the file simultaneously

- While running the BitTorrent software after the download is complete, others can receive the .torrent file from this computer
  - Ranked higher in the tit-for-tat system

BitTorrent

- Trackers: keep track of the number of seeds/peers; responsible for helping downloaders find each other, using a simple protocol on top of HTTP.
- Downloader sends status info to trackers, which reply with lists of contact information for peers which are downloading the same file.
- Web servers do not have information about content location
  - Only store metadata files describing the objects (length, name, etc.) and associating to each of them the URL of a tracker
- Network architecture: Hybrid unstructured
- Algorithm: Centralized Directory Model (CDM)
- "trackerless" torrents through a system called the "distributed database", through DHT (Distributed Hash Tables)



InterPlanetary File System (IPFS)

https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6yRzNde1FQv7uL6X1o4k7zrJa3LX/ipfs.draft3.pdf

https://ria.ua.pt/bitstream/10773/31279/1/Documento_Ricardo_Chaves.pdf

IPFS

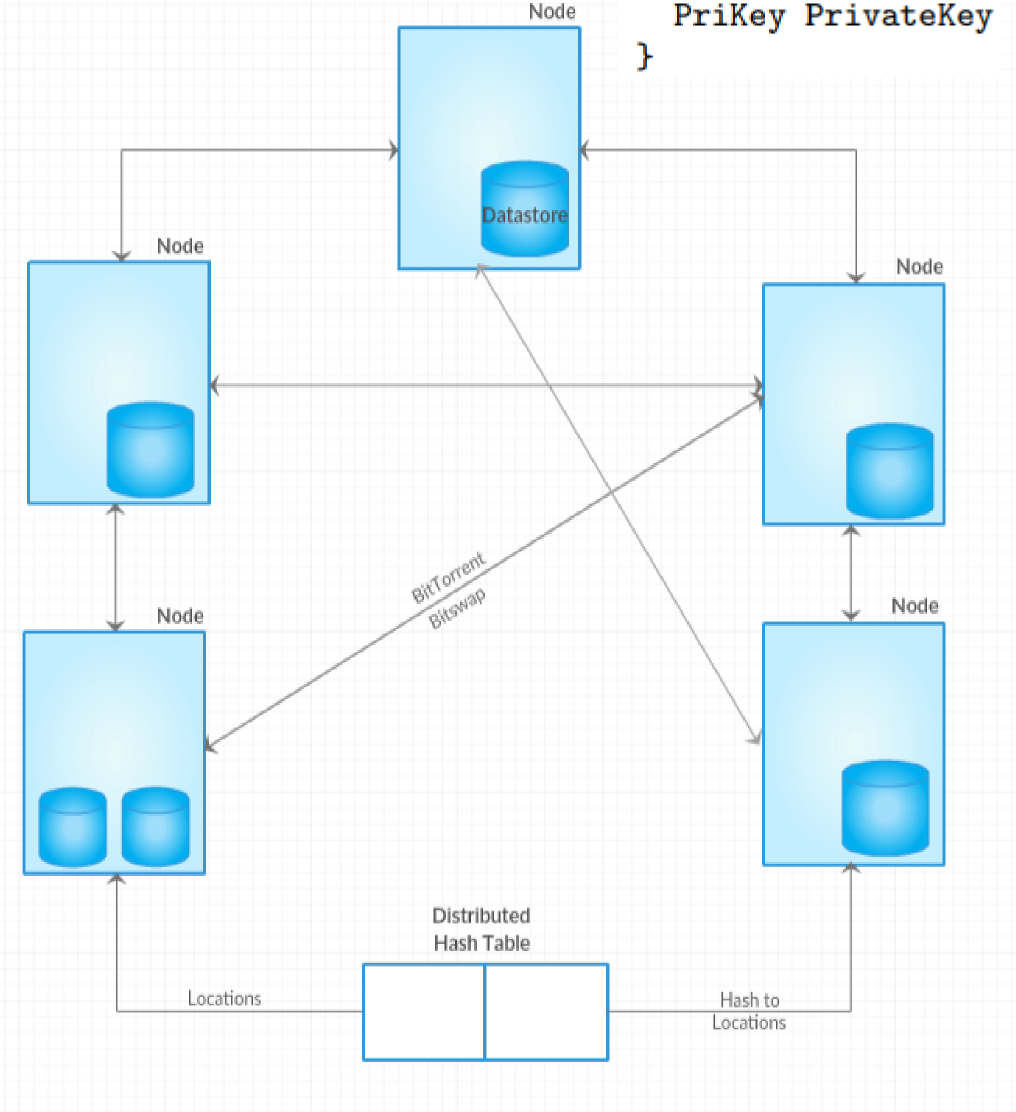
- Global distributed file system: IPFS is about “distribution” decentralization
- Content-based identification with secure hash of contents
- Resolving locations using Distributed Hash Table (DHT)
- Block exchanges using popular Bittorrent peer-to-peer file distribution protocol
- Incentivized block exchange using *Bitswap* protocol

- Merkle DAG (Directed Acyclic Graph) version-based organization of files, similar to Git version control system
- Self-certification servers for the storage nodes for security

IPFS

```
type Node struct {
  NodeId NodeID
  PubKey PublicKey
  PriKey PrivateKey
}
```

- Files in distributed storage
- Distributed hash table, uses the hash of the file as a key to return the location of the file.
- Once the location is determined, the transfer takes place peer-to-peer as a decentralized transfer.



IPFS Architecture

IPFS: Exchange the blocks

Peer nodes holding the data blocks are incentivized by a protocol called Bitswap.

Peer nodes have a *want_list* and *have_list*

Any imbalance is noted in the form of a BitSwap credit and debt

Bitswap protocol manages the block exchanges involving the nodes accordingly

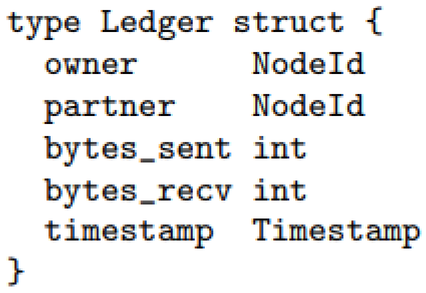
The nodes in the network thus have to provide value in the form of blocks.

If you send a block, you get a IPFS token that can be used when you need a block.

The Bitswap protocol has provisions for handling exceptions such as free loading node, node wanting nothing, node having nothing.

Bitswap calculation

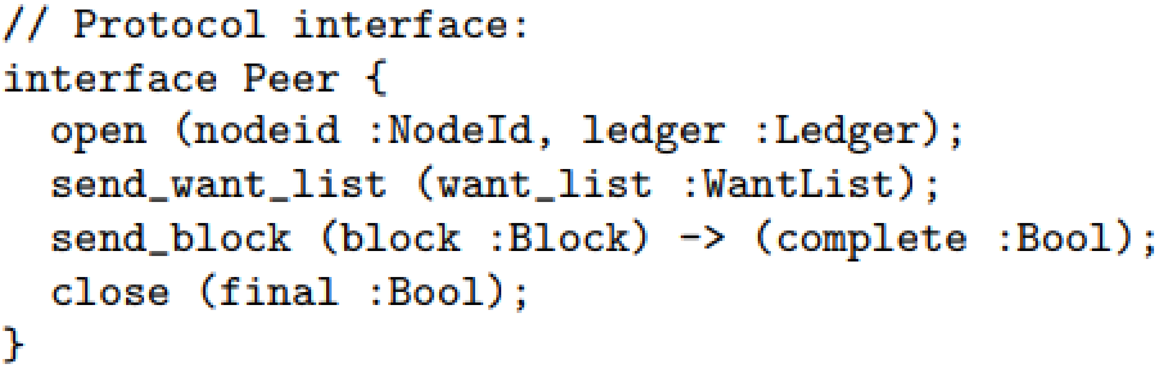
- Debt ratio r $r = \frac{\texttt{bytes\_sent}}{\texttt{bytes\_recv} + 1}$
- Probability of sending to a debtor $P\left(send \mid r \right) = 1 - \frac{1}{1 + exp(6 - 3r)}$
  - Function drops off quickly as the nodes’ debt ratio surpasses twice the established credit
- BitSwap nodes keep ledgers accounting the transfers with other nodes
  - When activating a connection, BitSwap nodes exchange their ledger information. If it does not match exactly, the ledger is reinitialized from scratch, losing the accrued credit or debt.



```
type Ledger struct {
  owner      NodeId
  partner    NodeId
  bytes_sent int
  bytes_recv int
  timestamp  Timestamp
}
```

Bitswap calculation

- Sketch of the lifetime of a peer connection:
  - 1. Open: peers send ledgers until they agree.
  - 2. Sending: peers exchange want_lists and blocks.
  - 3. Close: peers deactivate a connection.
  - 4. Ignored: (special) a peer is ignored (for the duration of a timeout) if a node’s strategy avoids sending



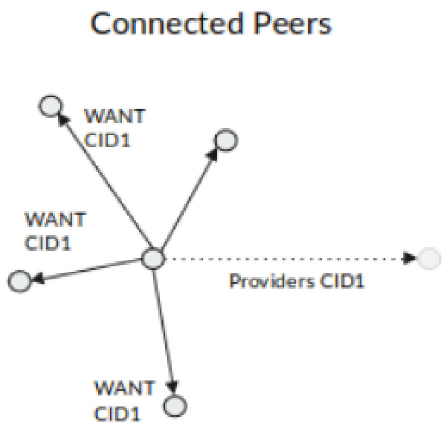
```
// Protocol interface:
interface Peer {
  open (nodeid :NodeId, ledger :Ledger);
  send_want_list (want_list :WantList);
  send_block (block :Block) -> (complete :Bool);
  close (final :Bool);
}
```

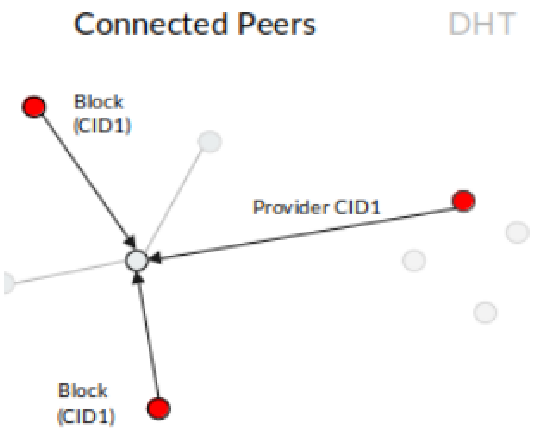
Content ID

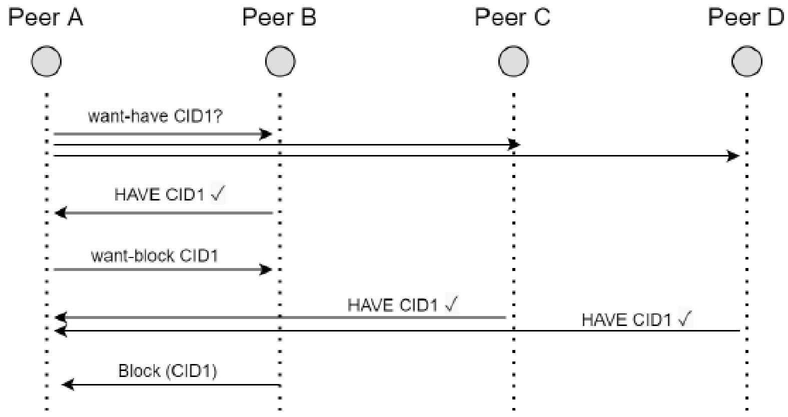
- In IPFS, every file or directory has a CID
  - Unique SHA256 hash used to identify the file.
- Whenever the content changes, the CID also changes.
- To keep track of files after being altered, IPFS uses the InterPlanetary Name System (IPNS) where the name is the hash of a public key, stored in the DHT, pointing to the CID of the latest version

Bitswap examples

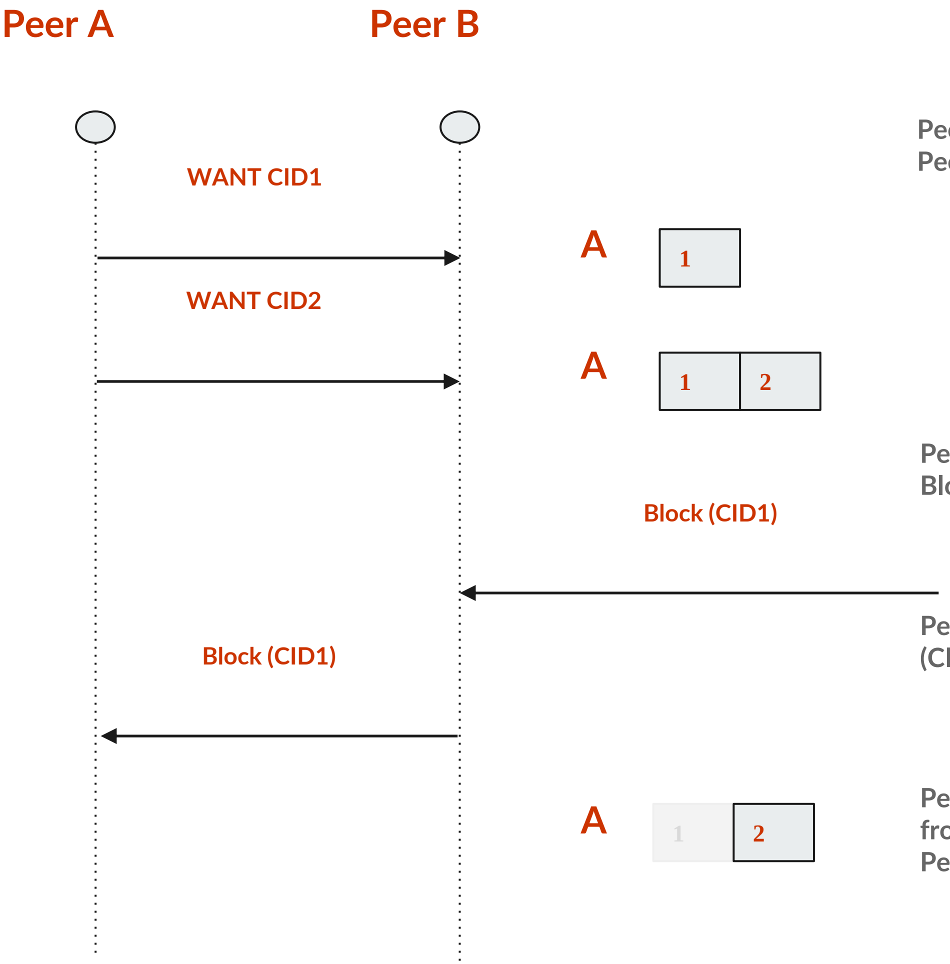
- When a peer wants a block, it broadcasts a Want to all connected peers
- If there is no response, it asks the DHT who has root CID. Peers who respond are added to the session and subsequent requests are sent only to peers in the session







Bitswap examples



Peer B remembers
Peer A’s wantlist

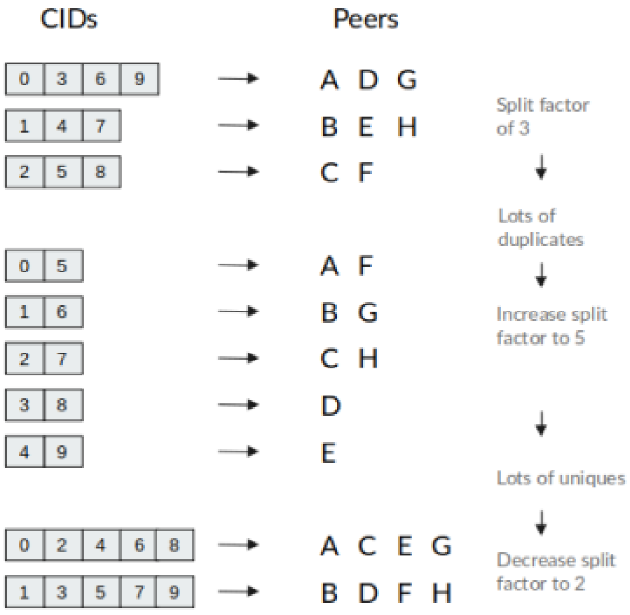
Peer B receives
Block (CID1)

Peer B sends Block
(CID1) to Peer A

Peer B removes CID1
from wantlist for
Peer A

IPFS: Split Factor

- When the local node receives a block, it broadcasts a cancel message for that block CID to all connected peers.
- However, the cancel may not be processed by the recipient peer before it has sent out the block → duplicates
- The local node keeps track of the ratio of duplicates / received blocks and adjusts the split factor.
  - If the ratio goes above 4 (a large number of duplicates), the split factor is increased - the same CID will be sent to less peers.
  - If the ratio goes below 2 (few duplicates) the split factor is decreased - the same CID will be sent to more peers.



IPFS: Cluster

- The cluster facilitates the replication of content across multiple nodes
- All cluster peers need to share the same cluster secret in order to be a part of the same cluster
  - Each cluster peer has its own unique ID.
- When new data is added and pinned to one of the peers of the cluster, all the other peers of that cluster receive the data.
- The peer responsible for initiating the cluster is the one who creates the cluster secret and is selected as the cluster leader.
- Every peer of the cluster is able to modify, add or remove data from the cluster.
- When a peer is added or removed from the cluster, the cluster continues working normally.
- In case the node leader goes down, a new leader is elected based on a RAFT consensus algorithm.
- Facilitates the management of groups that shall receive the same content, for example, software updates, multicast content, location-based content

RAFT consensus algorithm

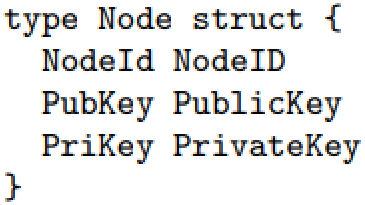
- A leader election is started by a candidate server.
- A server becomes a candidate if it receives no communication by the leader over a period called the election timeout (usually 3*100ms), so it assumes there is no acting leader anymore.
- It starts the election by increasing the term counter, voting for itself as new leader, and sending a message to all other servers requesting their vote.
- A server will vote only **once per term**, on a **first-come-first-served** basis.
- If a candidate receives a message from another server with a term number larger than the candidate's current term, then the candidate's election is defeated and the candidate changes into a follower and recognizes the leader as legitimate.
- If a candidate receives a majority of votes, then it becomes the new leader.
- If neither happens, e.g., because of a split vote, then a new term starts, and a new election begins.

IPFS: Locating nodes

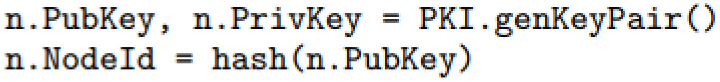
Nodes are identified by cryptographic hashes of public key

They hold the objects that form the files to be exchanged

Objects are identified by a secure hash, and an object may contain sub objects each with its own hash that is used in the creation of the root hash of the object.



```
type Node struct {
  NodeId NodeID
  PubKey PublicKey
  PriKey PrivateKey
}
```



```
n.PubKey, n.PrivKey = PKI.genKeyPair()
n.NodeId = hash(n.PubKey)
```

IPFS: Locating objects

IPFS identifies the resources by a hash.

Instead of identifying the resource by its location as in HTTP, IPFS identifies it by its content or by the secure hash of its content.

How to resolve the location?

Send around a request for anyone with a resource with the hash identifier

Routing part of the IPFS protocol maintains a DHT to locate the nodes as well as for file objects.

A simple DHT holds the hash as the key and location as the value.

Key can directly hash into the location.

DHT resolves to the closest location to the key value.

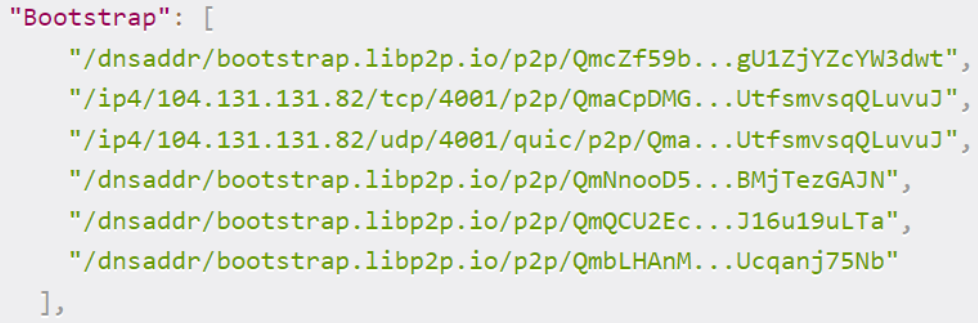
IPFS: Objects

- Object pinning: Nodes who wish to ensure the survival of particular objects can do so by pinning the objects.
  - Objects are kept in the node’s local storage.
- Object publishing: DHT, with content-hash addressing, allows publishing objects in a distributed way
  - Anyone can publish an object by simply adding its key to the DHT, adding themselves as a peer, and giving other users the object’s path.
  - New versions hash differently, and thus are new objects. Tracking versions is the job of additional versioning objects

How to connect an IPFS node to the p2p network?

- The config file ($IPFS_PATH/config) of every IPFS node has a list of bootstrap addresses

IPFS comes with a default list of trusted peers, but it can be modified to suit other needs. One popular use for a custom bootstrap list is to create a personal IPFS network.



```
"Bootstrap": [
    "/dnsaddr/bootstrap.libp2p.io/p2p/QmcZf59b...gU1ZjYZcYW3dwt",
    "/ip4/104.131.131.82/tcp/4001/p2p/QmaCpDMG...UtfsmvsqQLuvuJ",
    "/ip4/104.131.131.82/udp/4001/quic/p2p/Qma...UtfsmvsqQLuvuJ",
    "/dnsaddr/bootstrap.libp2p.io/p2p/QmNnooD5...BMjTezGAJN",
    "/dnsaddr/bootstrap.libp2p.io/p2p/QmQCU2Ec...J16u19uLTa",
    "/dnsaddr/bootstrap.libp2p.io/p2p/QmbLHAnM...Ucqanj75Nb"
  ],
```

- List of peers with which the IPFS daemon learns about other peers on the network
  - Running the IPFS daemon command
  - First establish a p2p connection with Protocol Labs (company behind IPFS) bootstrap nodes
  - Through these bootstrap nodes, it will further find hundreds of other peers
  - Peers will talk through TCP, UDP on port: **4001**

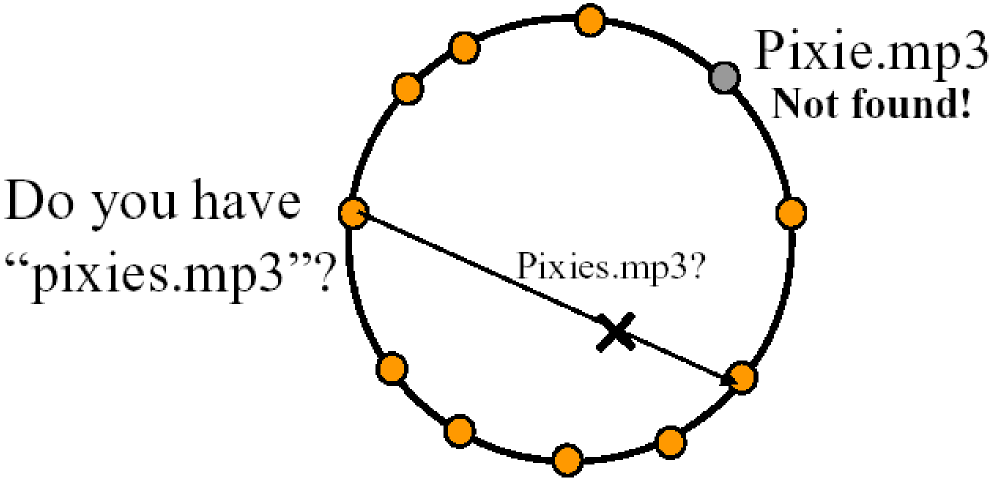
Distributed Hash Tables (DHTs)

https://www.cs.cmu.edu/~dga/15-744/S07/lectures/16-dht.pdf

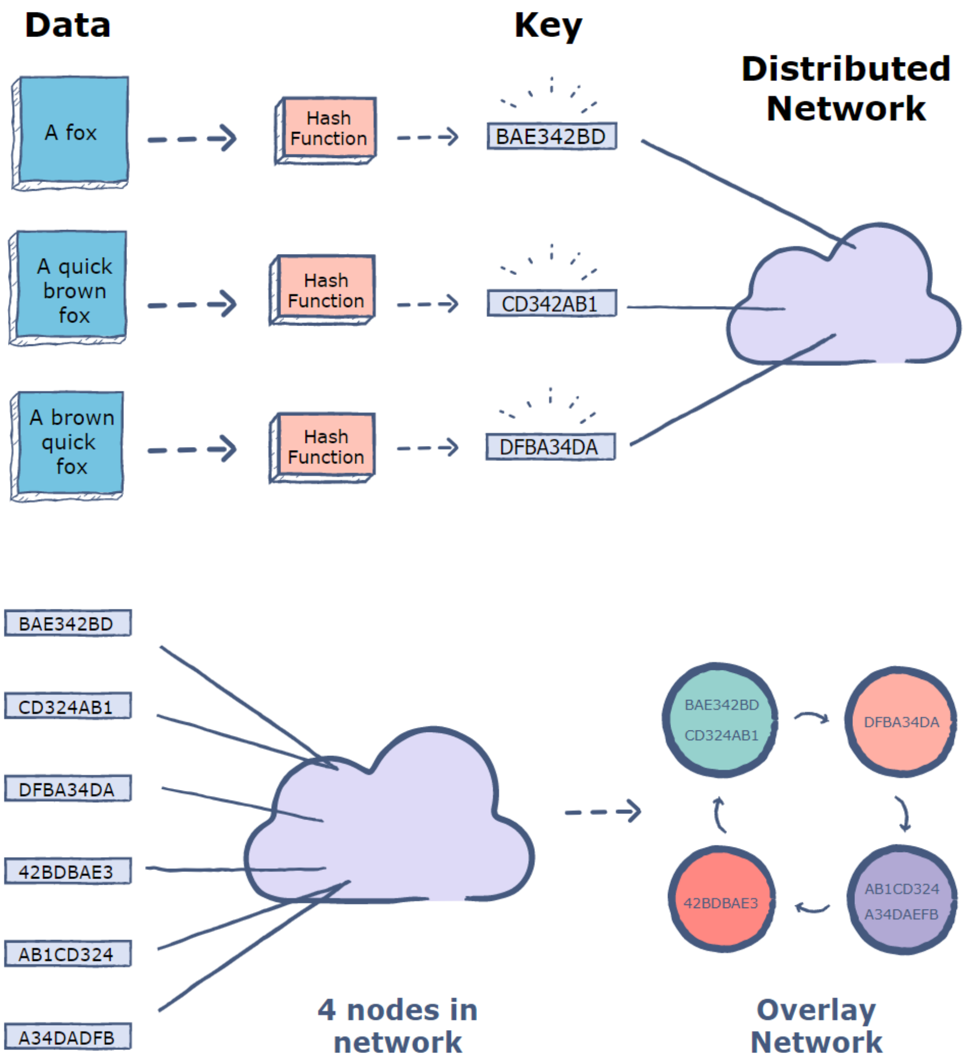
https://pdos.csail.mit.edu/papers/ton:chord/paper-ton.pdf

Searching in DHTs (structured)

- **Need to know the exact filename**
  - **Keys (filenames) map to node-ids**
    - Change in file name → search at different nodes
    - No wildcard matching: cannot ask for file “pix*”



DHT



Data

Key

Distributed Network

A fox

Hash Function

BAE342BD

A quick brown fox

Hash Function

CD342AB1

A brown quick fox

Hash Function

DFBA34DA

BAE342BD

CD324AB1

DFBA34DA

42BDBAE3

AB1CD324

A34DADFB

4 nodes in network

BAE342BD CD324AB1

DFBA34DA

42BDBAE3

AB1CD324 A34DAEFB

Overlay Network

CHORD: DHT Algorithm

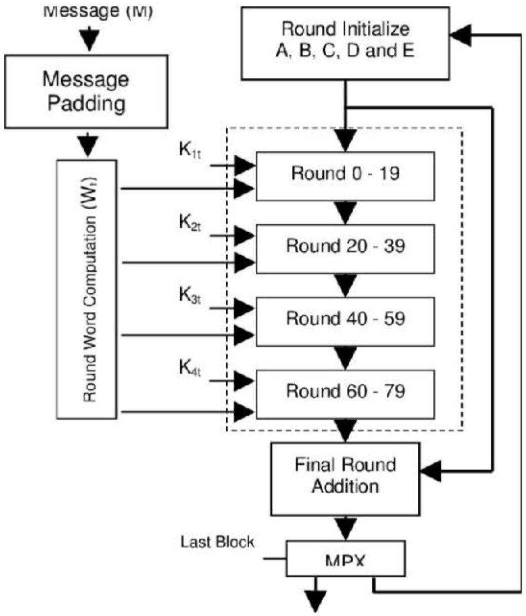
- All files/data items in the network will have an identifier, which will be hashed to give a key for that particular resource
- If a node needs a file/data, it will hash its name

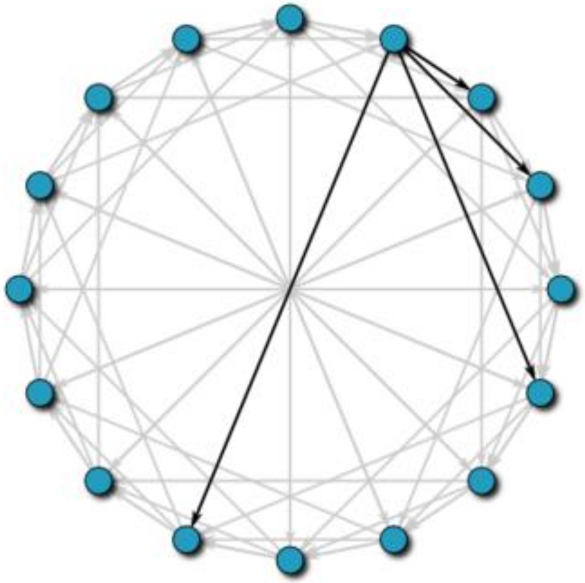
and then send a request using this key.

- All *n* nodes also use the function to hash their IP

addresses, and conceptually, the nodes will form a ring in ascending order of their hashed IP

SHA-1





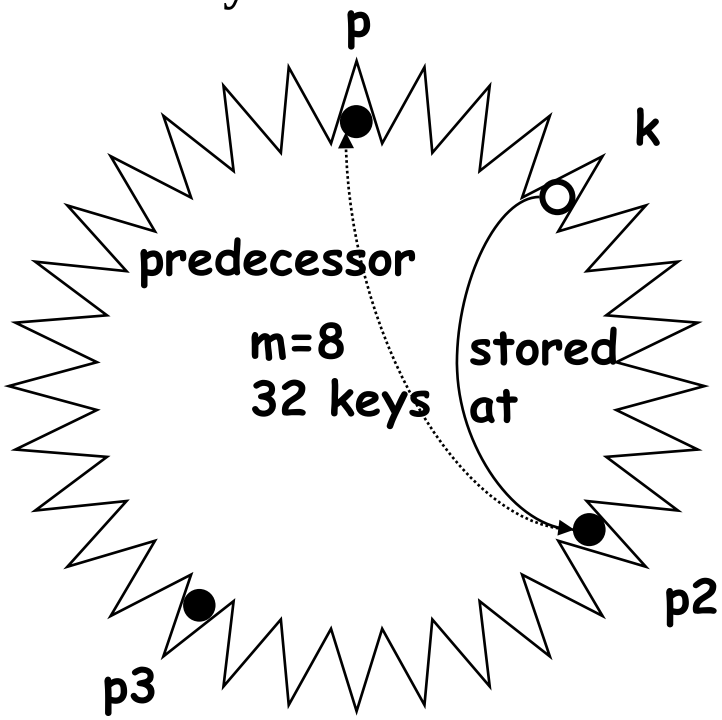
CHORD: DHT Algorithm

- The successor node of a key *k* is the first node whose ID equals to *k* or follows *k* in the identifier circle, denoted by successor(*k*)
- Every key is assigned to its successor node, so looking up a key *k* is to query successor(*k*).

- When a node wants to share a file or some data
  - Hashes the identifier to generate a **key *k***, and sends its **IP** and the **file identifier** to **successor(*k*)**
  - These are then stored at this node
  - All resources are indexed in a large DHT across all participating nodes
  - If there are two or more nodes that hold a given file or resource, the **keys will be stored at the same node** in the DHT, giving the requesting node a choice of requesting the file in one or the other node, or both

DHT: Store Information

- Hashing of search keys AND peer addresses on binary keys of length m
  - e.g. m=8, key("jingle-bells.mp3")=17, key(196.178.0.1)=3
- Calculates hash of data to get *k*
- Routing is used to find the node that stores key *k* (*node*$_k$)
- Data keys are stored at next larger node key

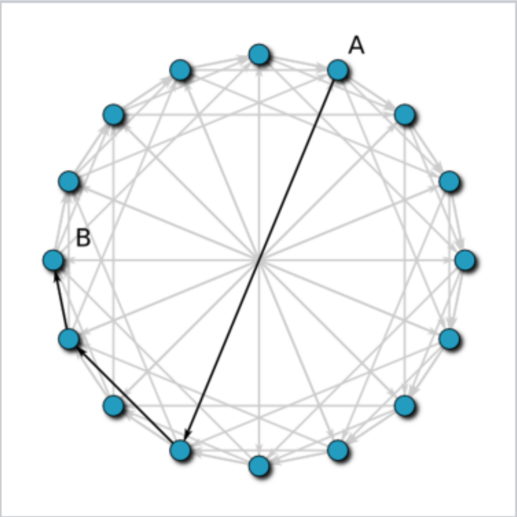


Search possibilities

1. every peer knows every other
   O(n) routing table size
2. peers know successor
   O(n) search cost

DHT: Search Information

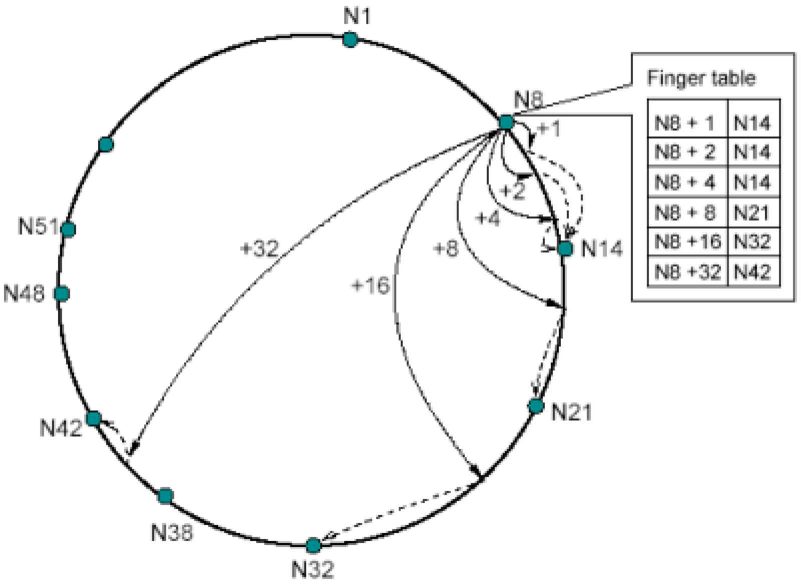
- When a node wants a content
  - Hashes data identifier and sends a request to successor(*k*)
  - Reply with the IP of the node that holds the actual data
  - How does a node request information from successor(*k*), when it doesn't know its IP, but only the key?
    - Every node holds what is known as a finger table
      - Contains a list of keys and their successor IP's
      - Each node holds the IP of an exponential sequence of nodes that follow it, i.e. entry *i* of node *k*'s finger table holds the IP of node *k* + 2^*i*



The routing path between nodes A and B. Each hop cuts the remaining distance in half (or better).

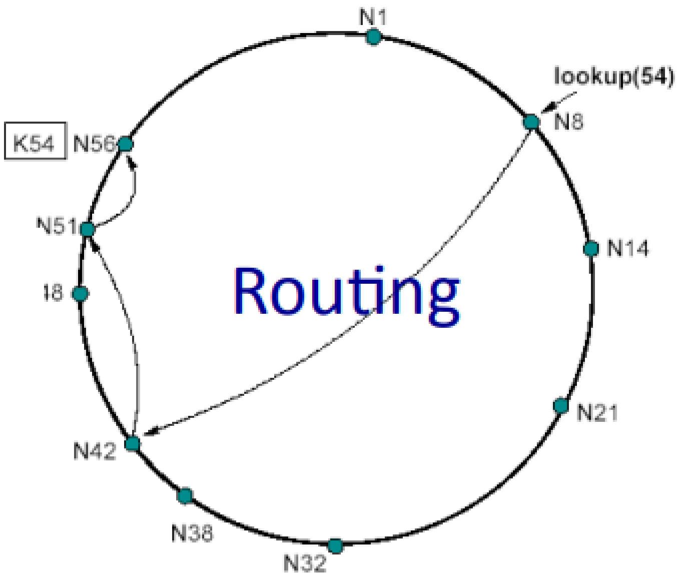
Search information: Finger Table

Routing Tables

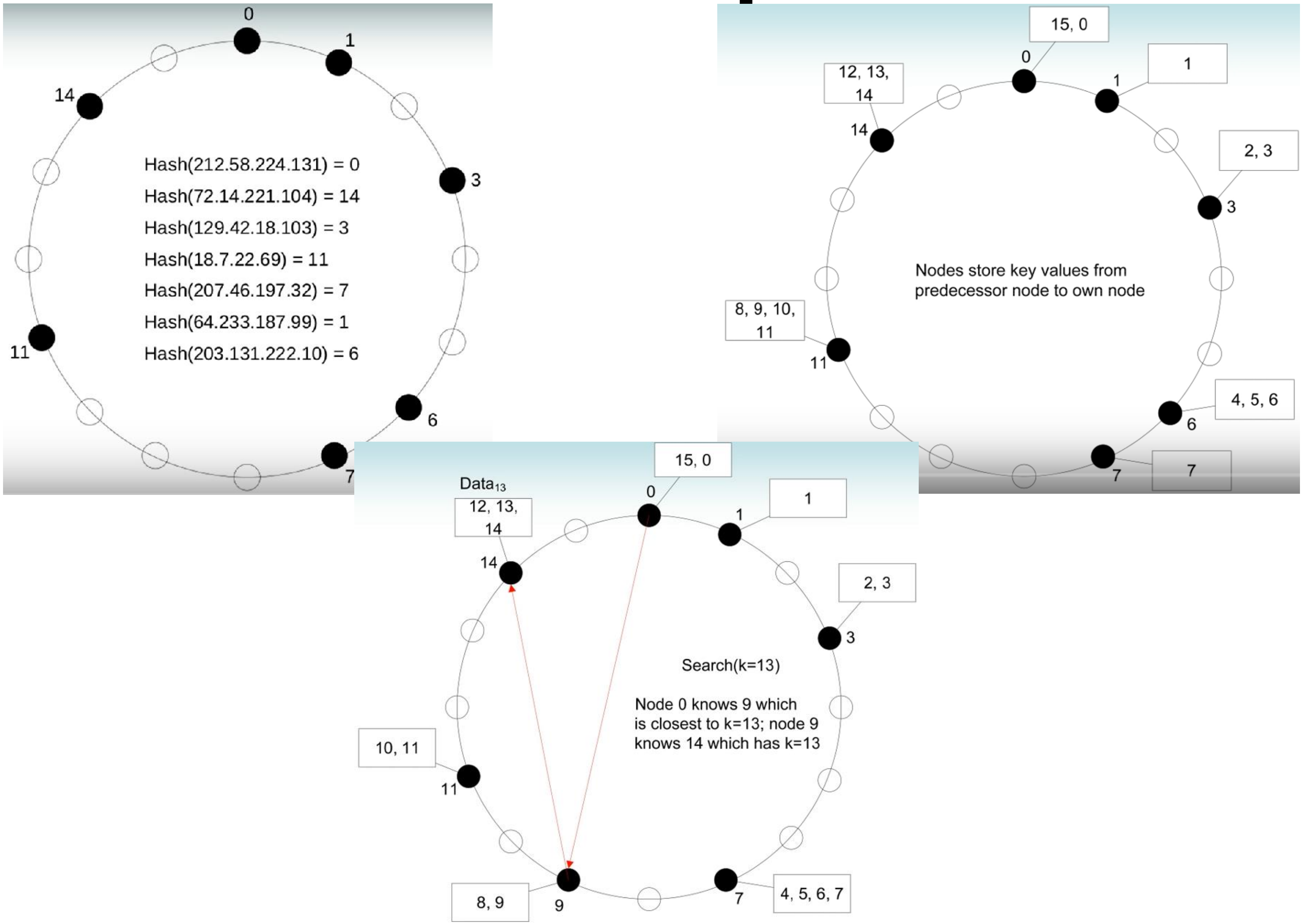


Finger table

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



Example



Hash(212.58.224.131) = 0

Hash(72.14.221.104) = 14

Hash(129.42.18.103) = 3

Hash(18.7.22.69) = 11

Hash(207.46.197.32) = 7

Hash(64.233.187.99) = 1

Hash(203.131.222.10) = 6

Nodes store key values from predecessor node to own node

Search(k=13)

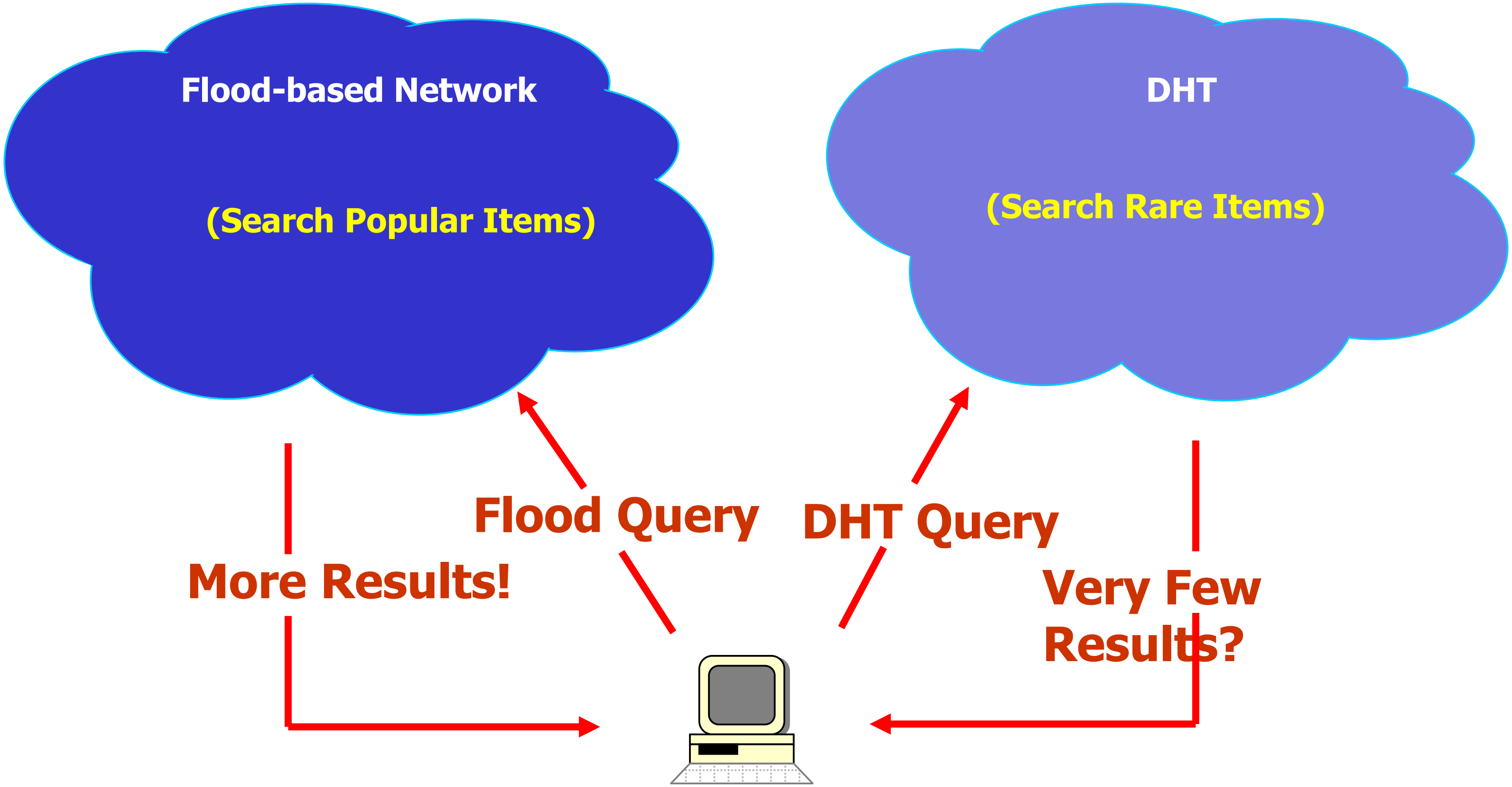
Node 0 knows 9 which is closest to k=13; node 9 knows 14 which has k=13

File Search: Flooding vs. DHTs

- Recall
  - Flooding can miss files
  - DHTs should never
- Query complexity
  - Flooding can handle arbitrary single-site logic
  - DHTs can do equijoins, selections, aggregates, etc.
    - But not so good at fancy selections like wildcards
- Query Performance
  - Flooding can be slow to find things, uses lots of BW
  - DHTs: expensive to publish documents with lots of terms
  - DHTs: expensive to intersect really long term lists
    - Even if output is really small!
- Hybrid solution!

Hybrid Search

Hybrid = “Best of both worlds”



Security

Security - attacks

- Poisoning attacks
  - e.g. providing files whose contents are different from the description
- Polluting attacks
  - e.g. inserting "bad" chunks/packets into an otherwise valid file on the network
- Freeloaders
  - Users or software that make use of the network without contributing resources to it
- Insertion of viruses to carried data
  - e.g. downloaded or carried files may be infected with viruses or other malware
- Malware in the peer-to-peer network software itself
  - e.g. distributed software may contain spyware
- Denial of service attacks
  - Attacks that may make the network run very slowly or break completely
- Filtering
  - Network operators may attempt to prevent peer-to-peer network data from being carried
- Identity attacks
  - e.g. tracking down the users of the network and harassing or legally attacking them
- Spamming
  - e.g. sending unsolicited information across the network- not necessarily as a denial of service attack

Security

- **Most attacks can be defeated or controlled by careful design of the peer-to-peer network and through the use of encryption**
  - **However, almost any network will fail when the majority of the peers are trying to damage it**
- **Anonymity**
  - **Some peer-to-peer protocols (such as Freenet) attempt to hide the identity of network users by passing all traffic through intermediate nodes**
- **Encryption**
  - **Some peer-to-peer networks encrypt the traffic flows between peers**
    - Make it harder for an ISP to detect that peer-to-peer technology is being used (as some artificially limit bandwidth)
    - Hide the contents of the file from eavesdroppers
    - Impede efforts towards law enforcement or censorship of certain kinds of material
    - Authenticate users and prevent 'man in the middle' attacks on protocols
    - Aid in maintaining anonymity