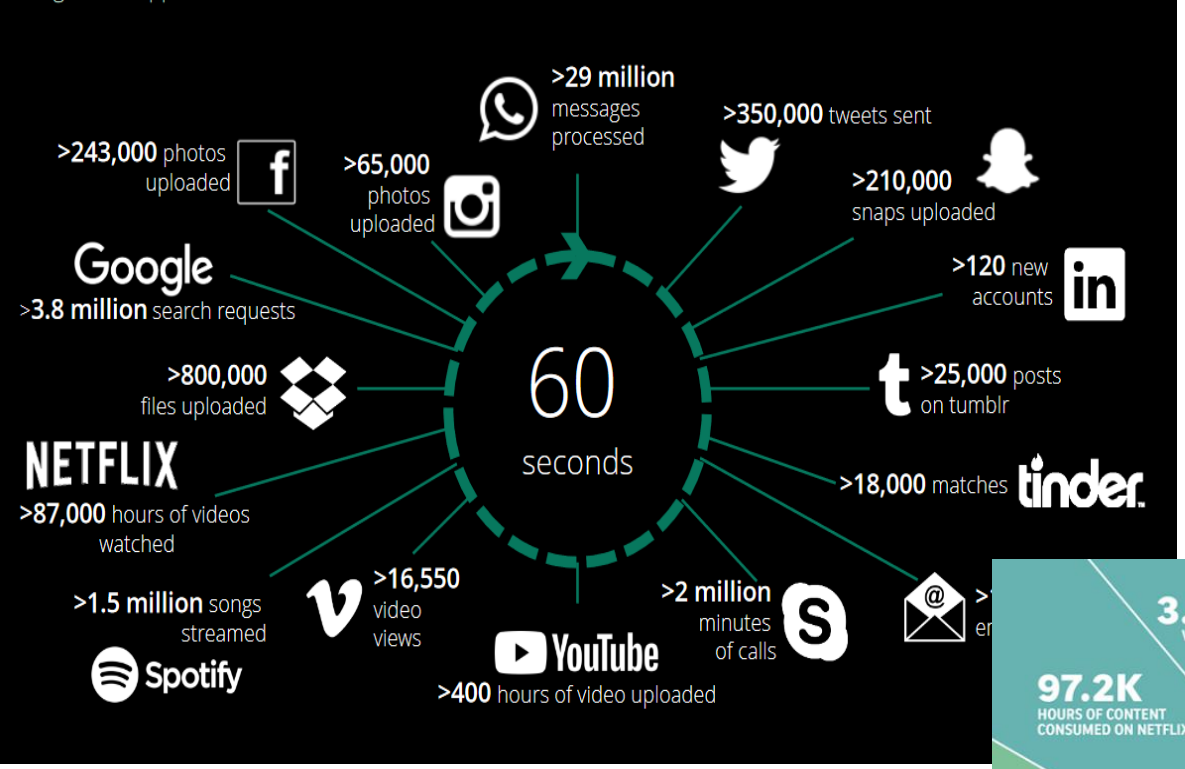
Peer-to-Peer Systems and Networks

Mestrado em Engenharia de Computadores e Telemática

2022/2023


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


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
4.2MM GOOGLE SEARCHES
1.3K PRODUCT-RICH PINS PINNED
3K ACTIVE USERS
4.1K CLICKS ON SPONSORED INSTAGRAM POSTS
350K TWEETS SENT
510K COMMENTS POSTED ON FB
7K ACTIVE USERS ON LINKEDIN
12.5K RIDE-SHARES TAKEN

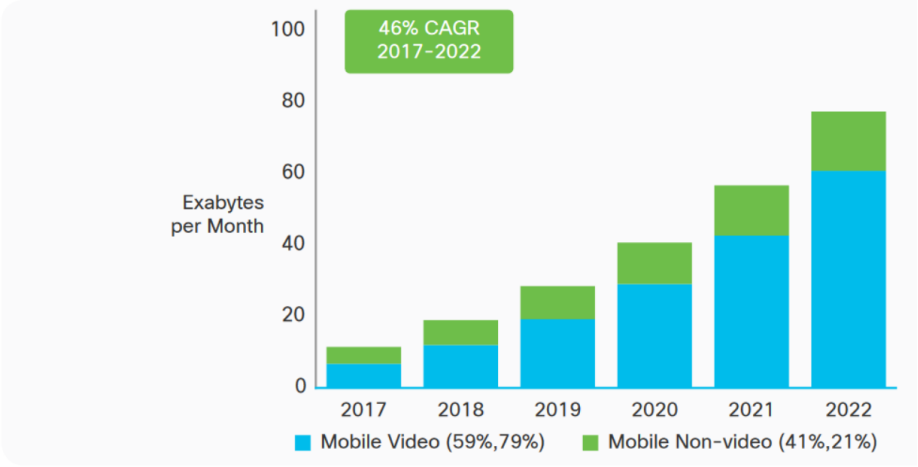
Motivation

- 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

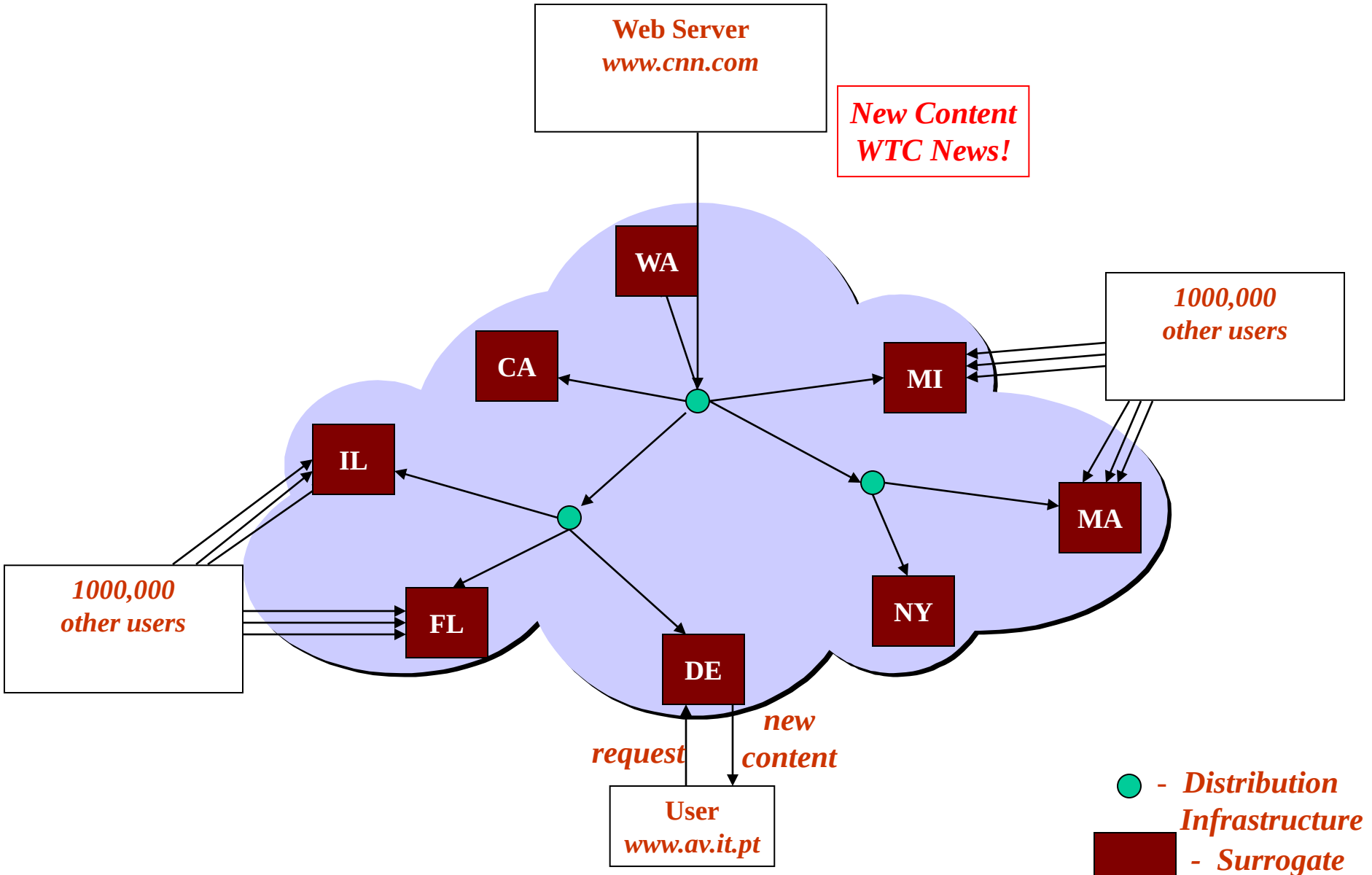
What is a CDN?

- 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



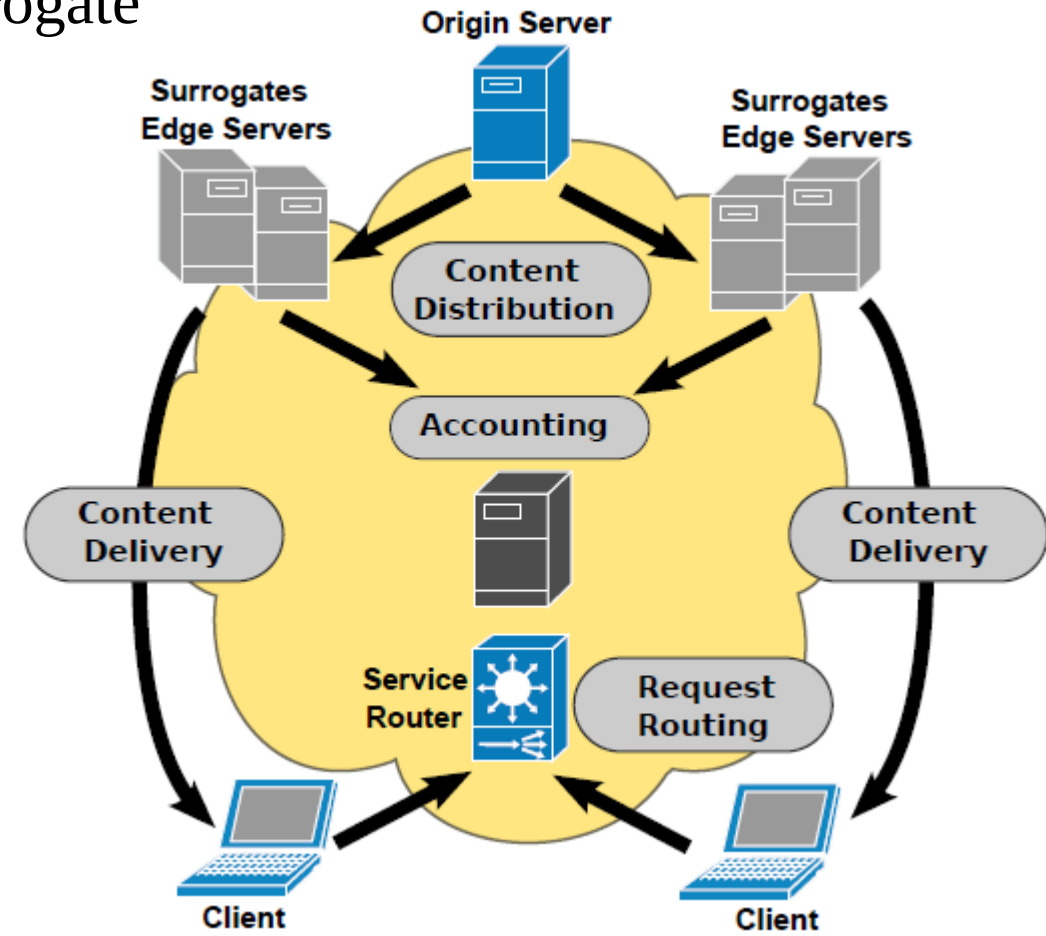
CDN Components

- *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



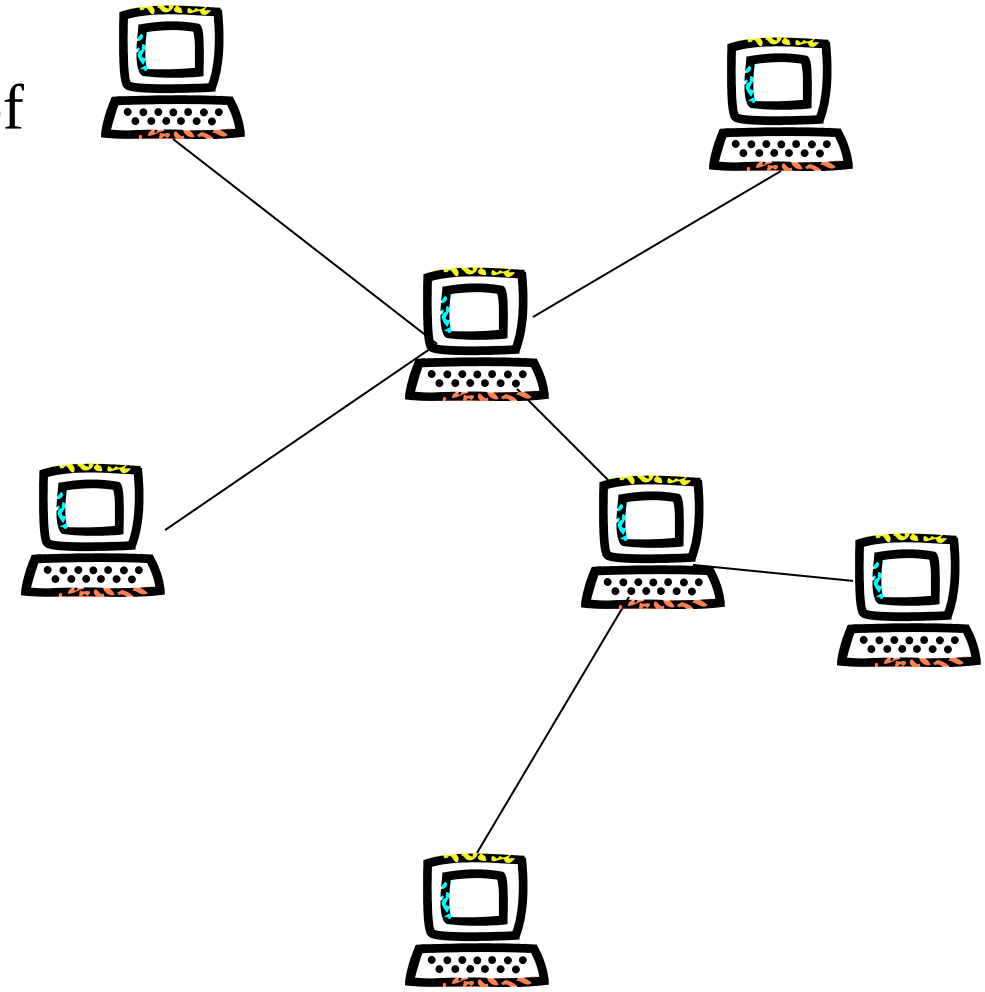
Peer-to-peer networks

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.

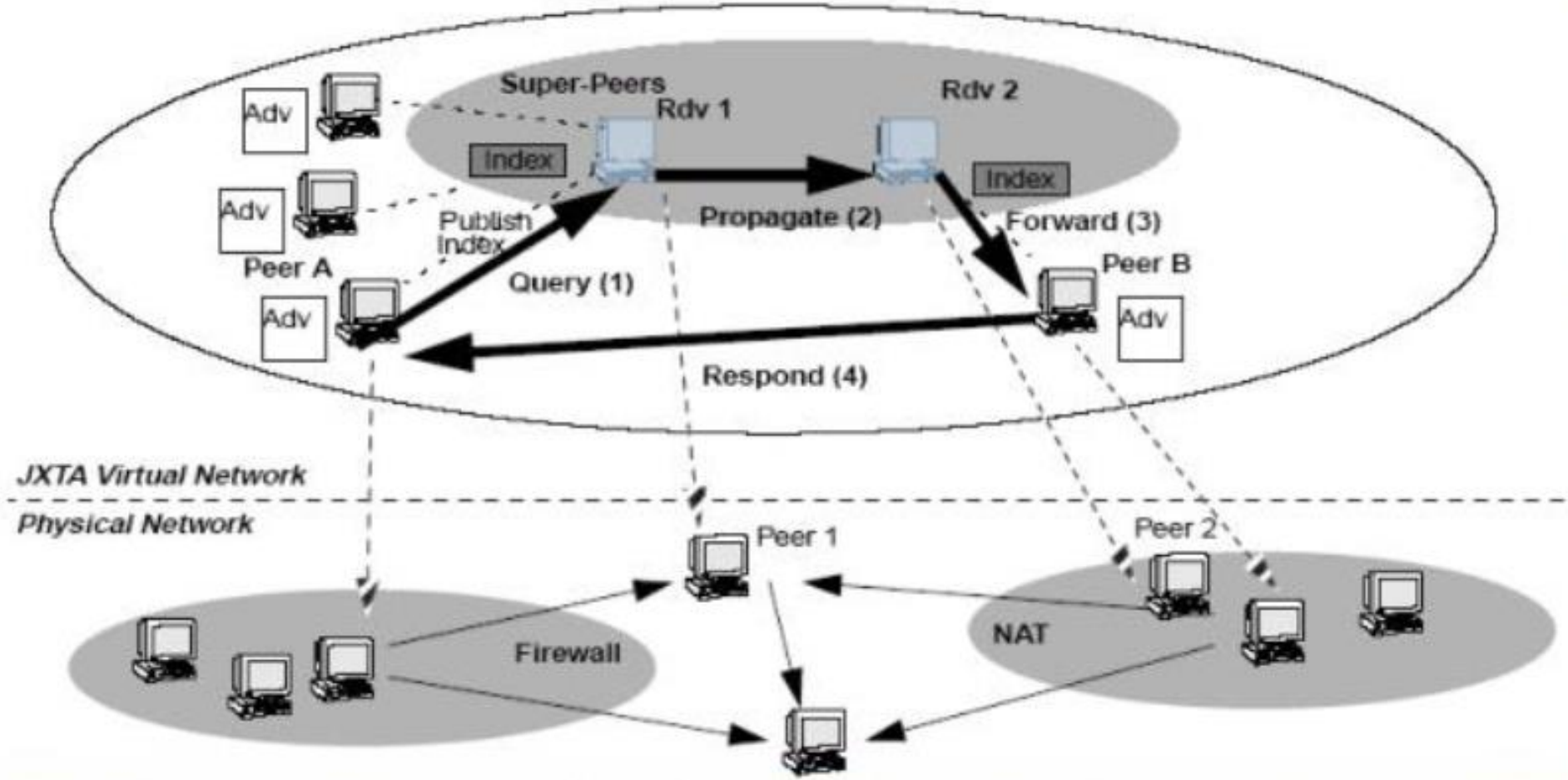
- They are not responsible for handling communication on behalf of other peers.

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.

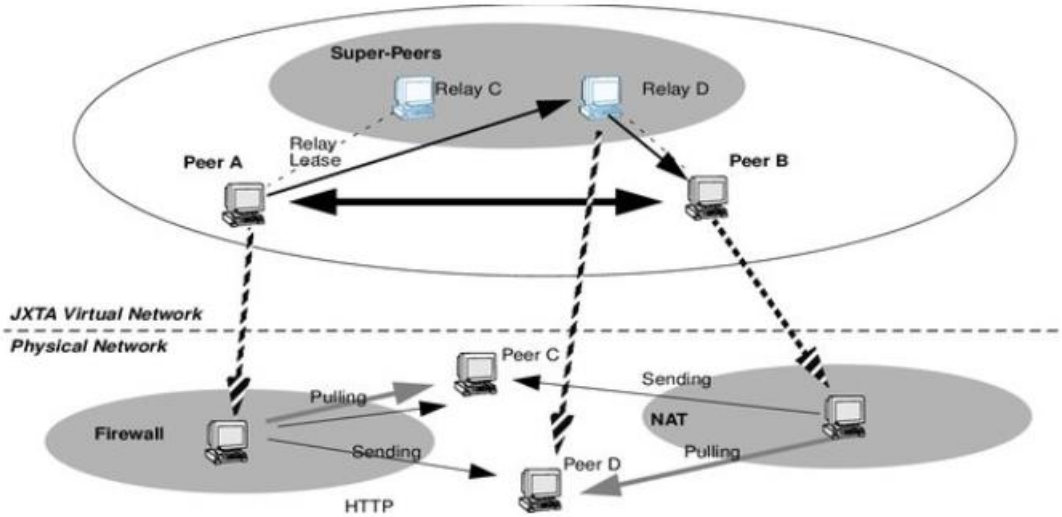
Rendez-vous Peers



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).



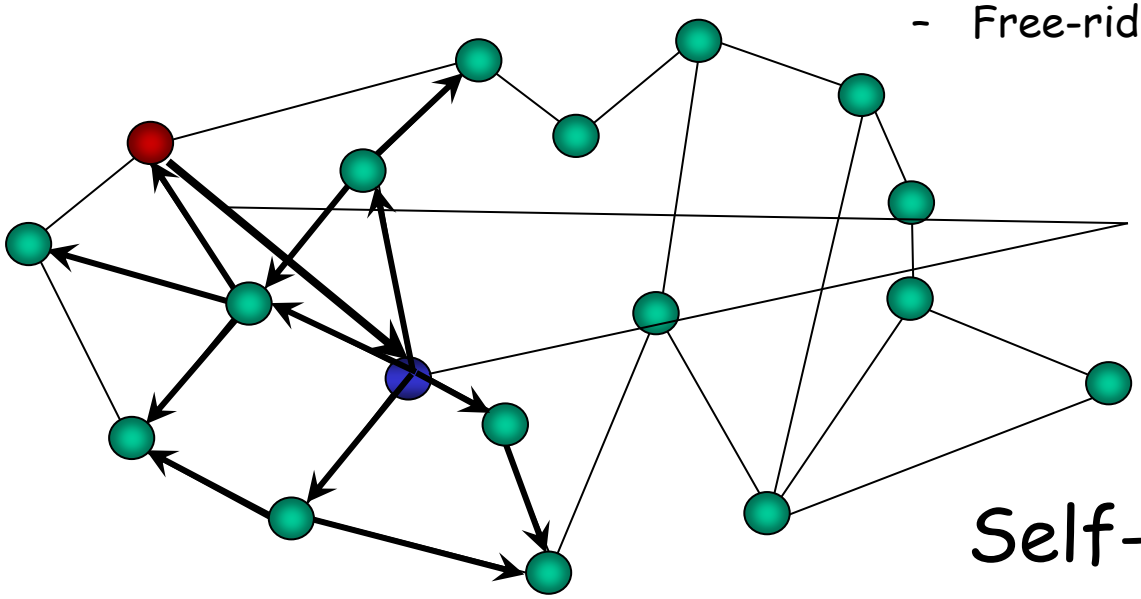
Structured vs Unstructured

- 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

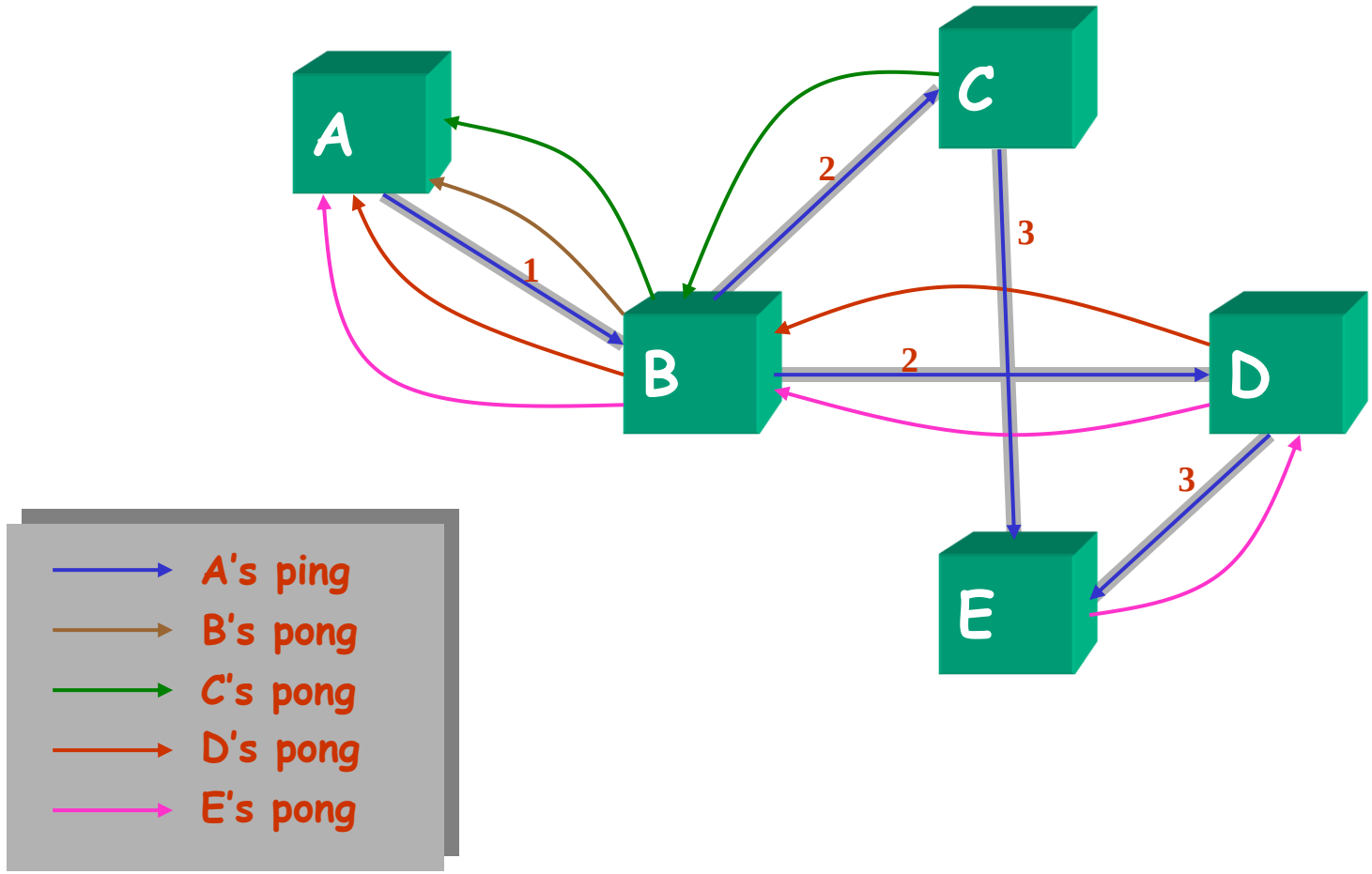


I have "brick_in_the_wall.mp3" ….

Self-organizing System

Gnutella: no servers

Gnutella: Meeting Peers (Ping/Pong)

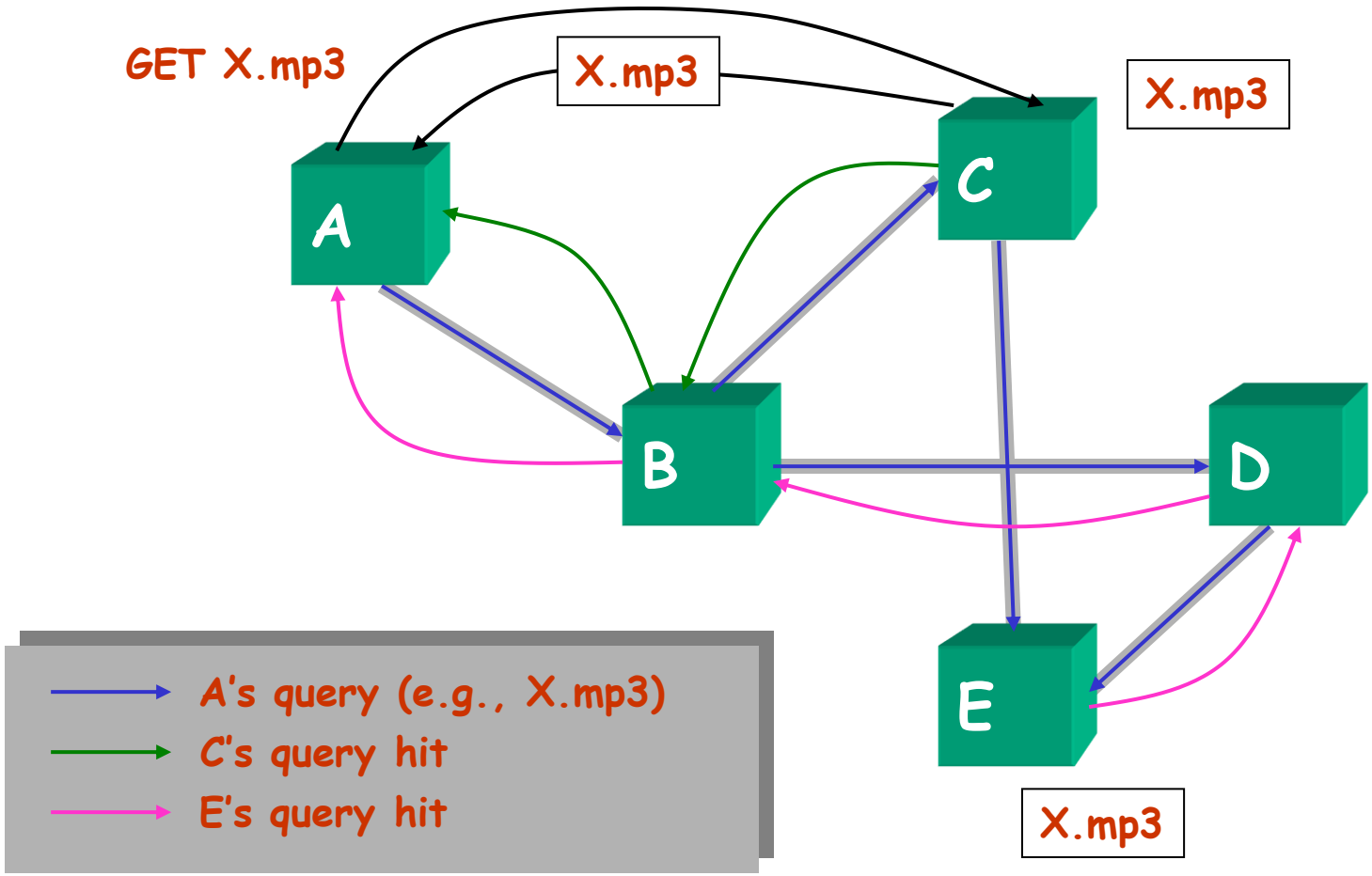


Gnutella: Protocol Message Types

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

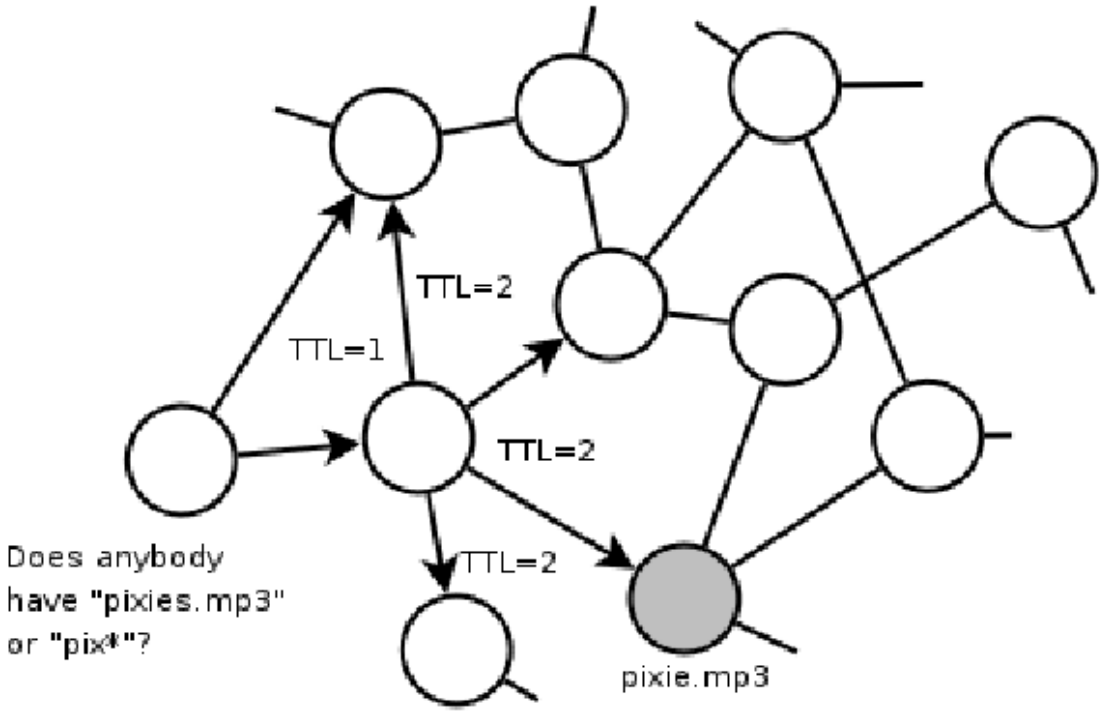
Gnutella: Searching

(Query/QueryHit/GET)



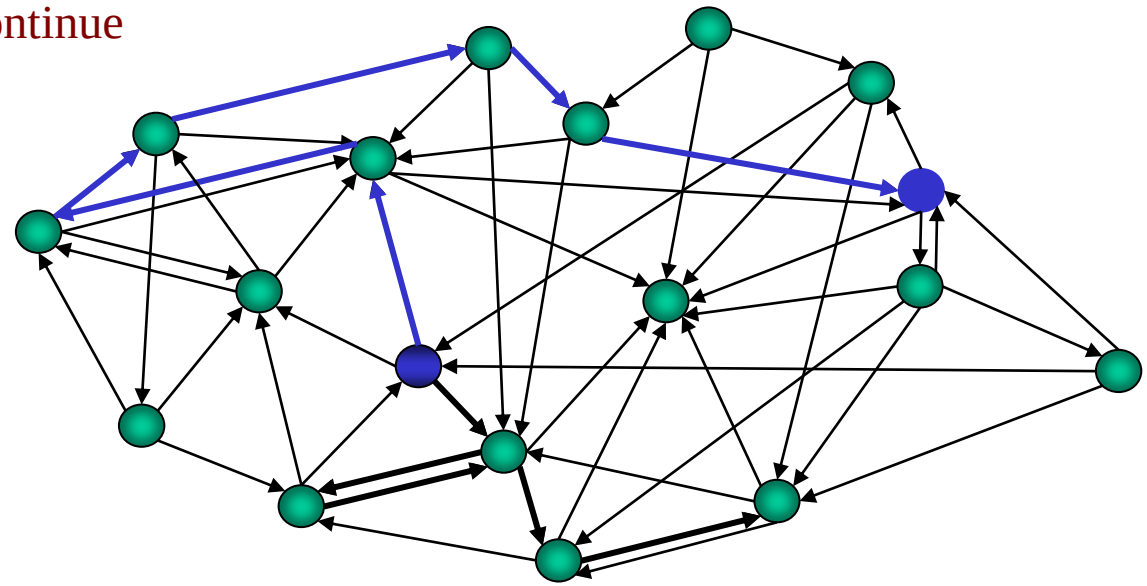
Searching in Gnutella (structureless)

- 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?



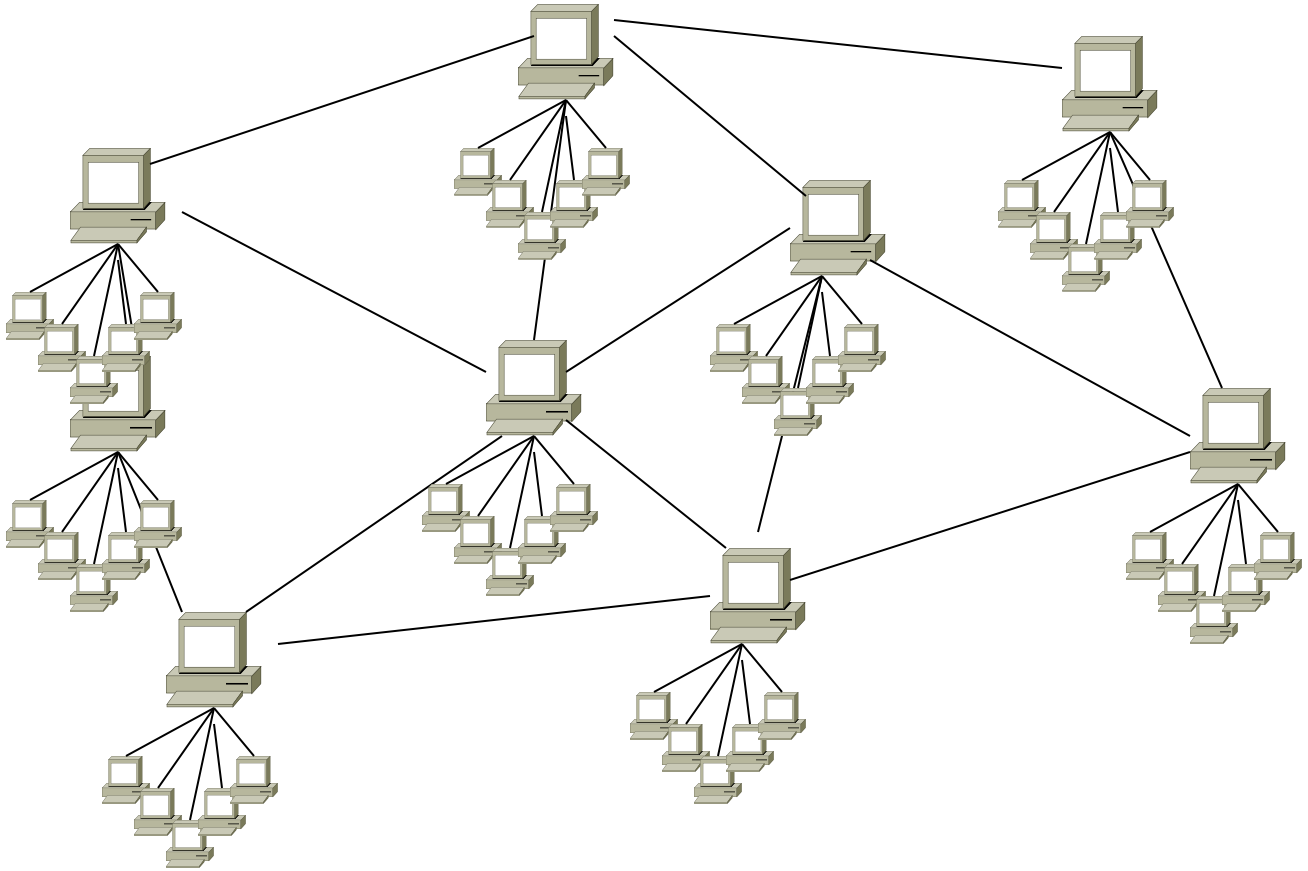
Improvements of Message Flooding

- 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



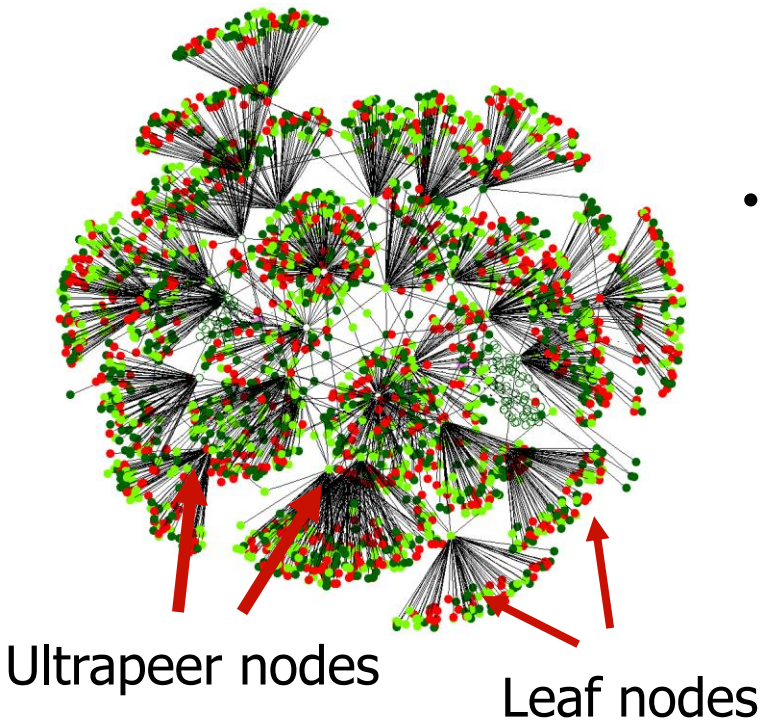
Hybrid Gnutella: “Ultrapeers”

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



Real Gnutella Network

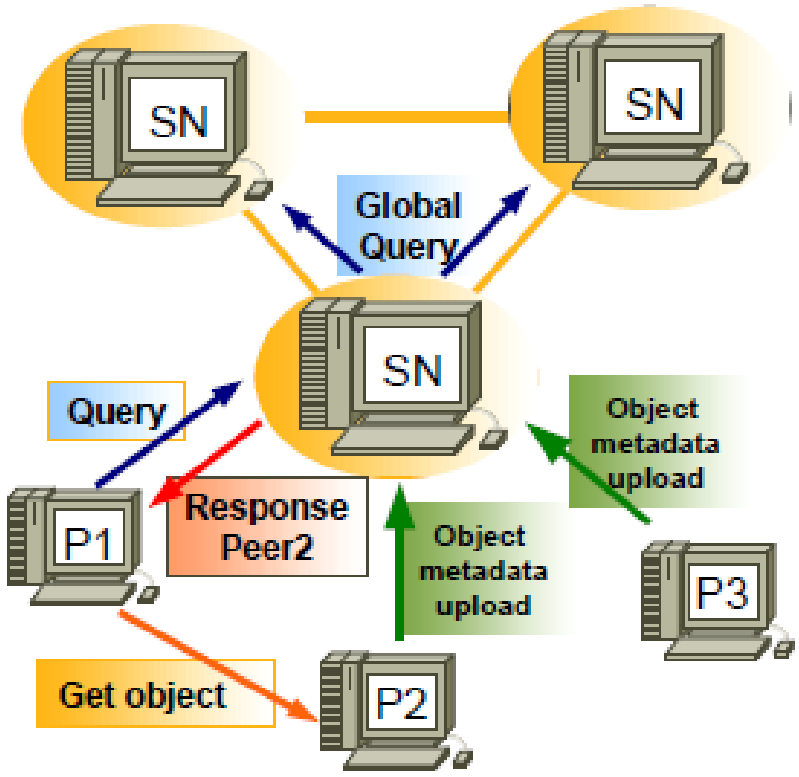
Oct 2003 Crawl of public gnutella (v.2)



- 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

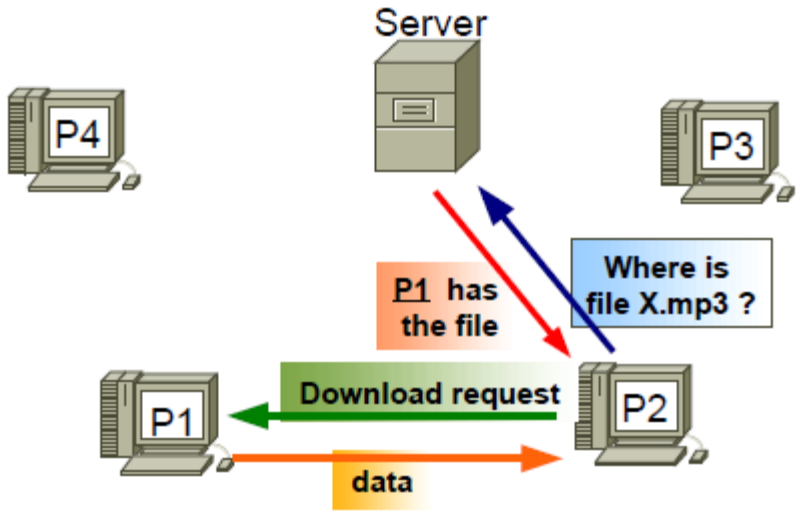
FastTrack/KaZaA

- 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

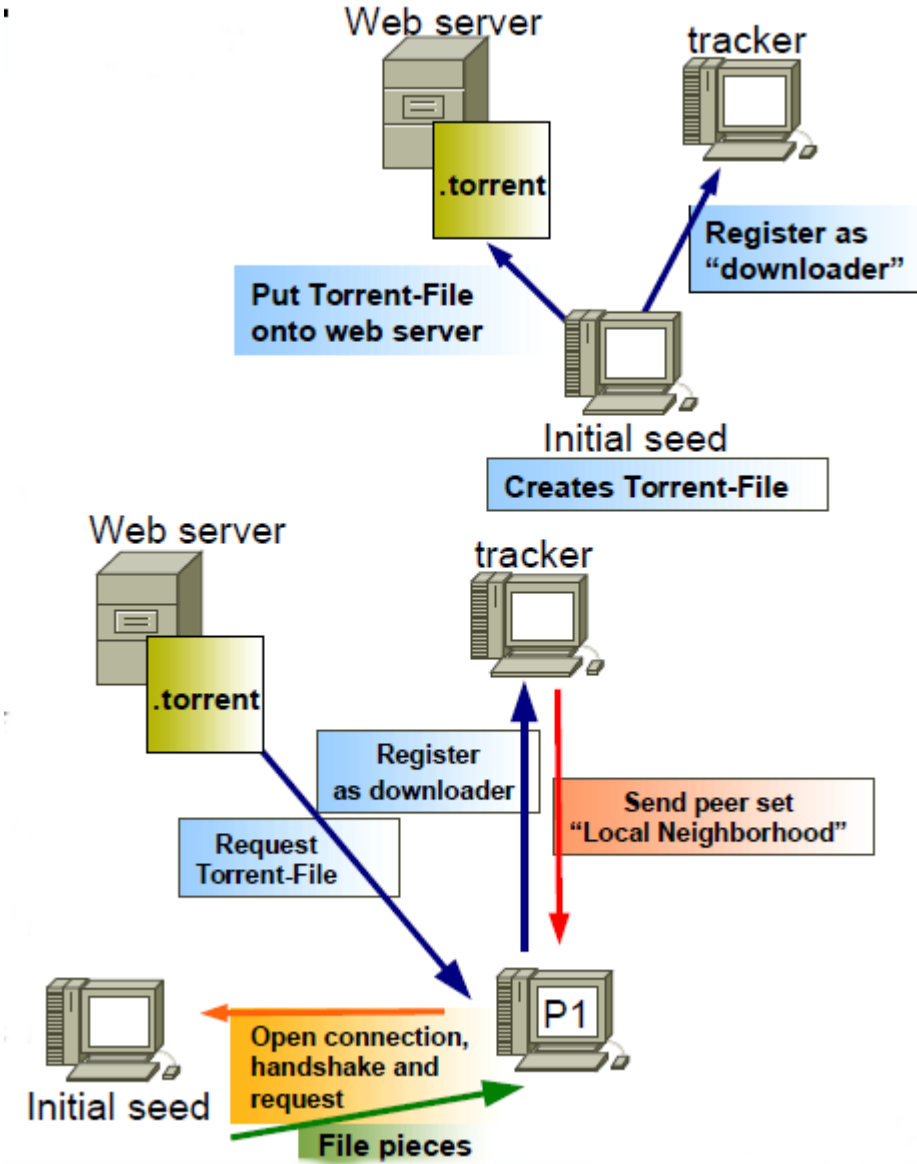
- .torrent: pointer file that directs the computer to the file it wants to download
- Swarm: group of computers simultaneously downloading or uploading the same file
- Tracker: server that manages the BitTorrent file transfer process

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

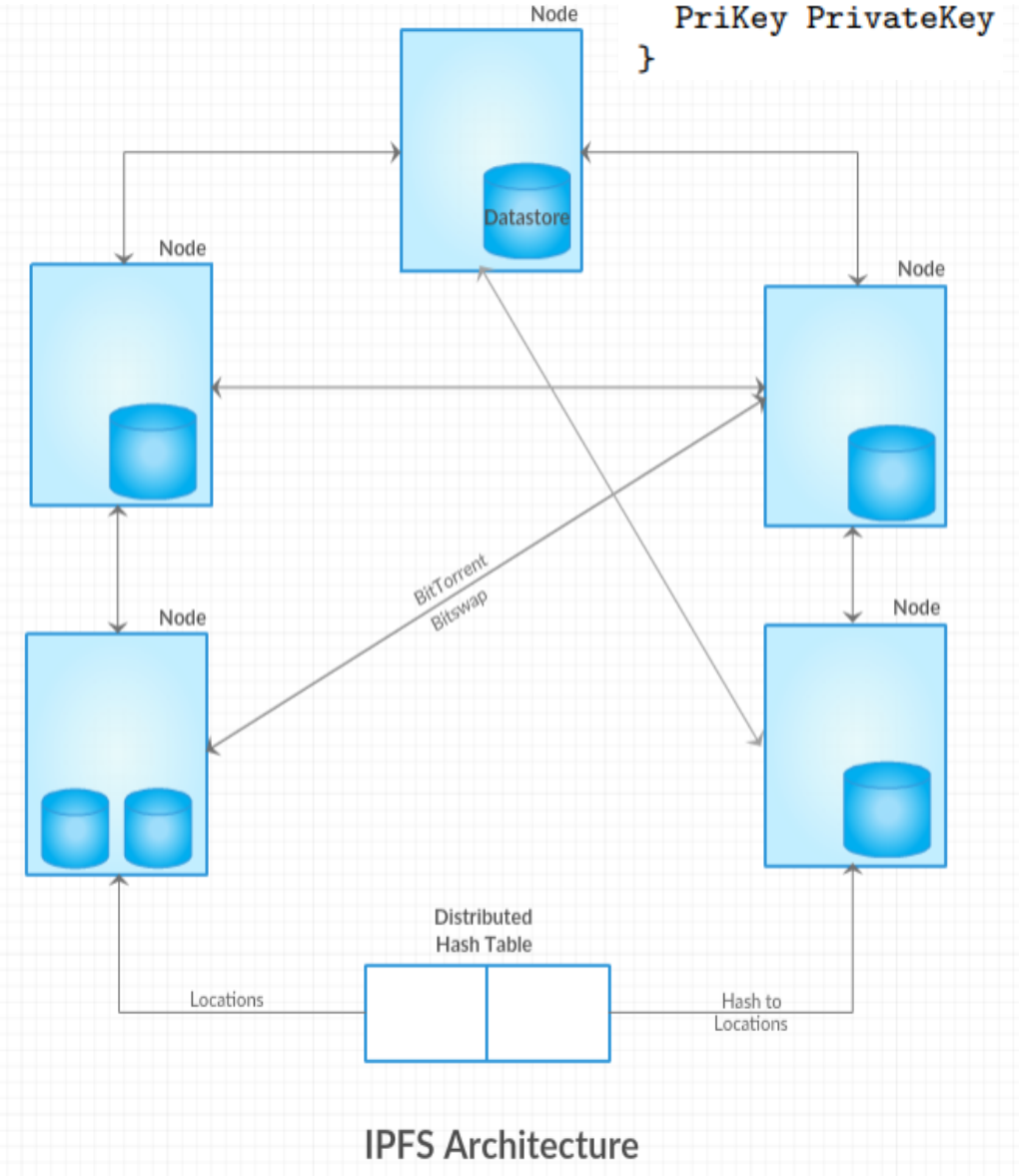
- Global distributed file system: IPFS is about “distribution” decentralization
- Content-based identification with secure hash of contents
- Resolving locations using Distributed Hash Table (DHT)
- Block exchanges using popular Bittorrent peer-to-peer file distribution protocol
- Incentivized block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

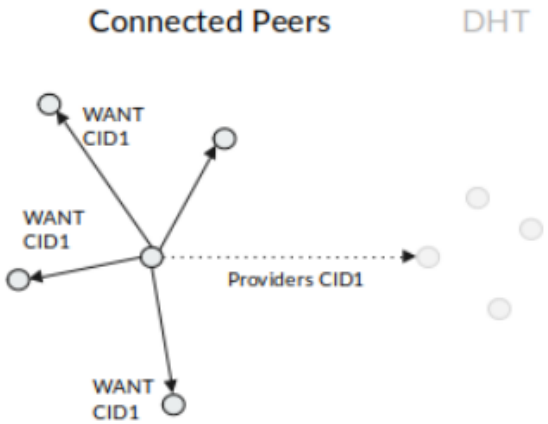
```
// Protocol interface:
interface Peer {
  open (nodeid :NodeId, ledger :Ledger);
  send_want_list (want_list :WantList);
  send_block (block :Block) -> (complete :Bool);
  close (final :Bool);
}
```

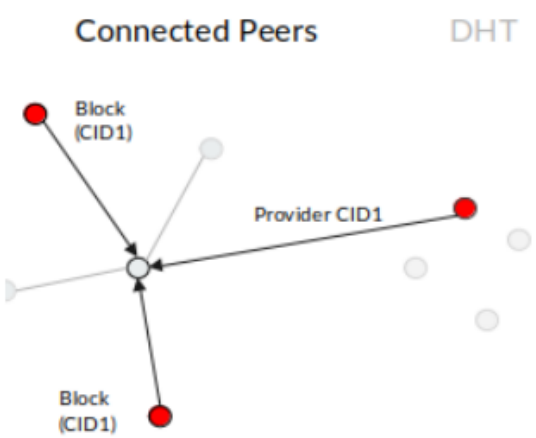
Content ID

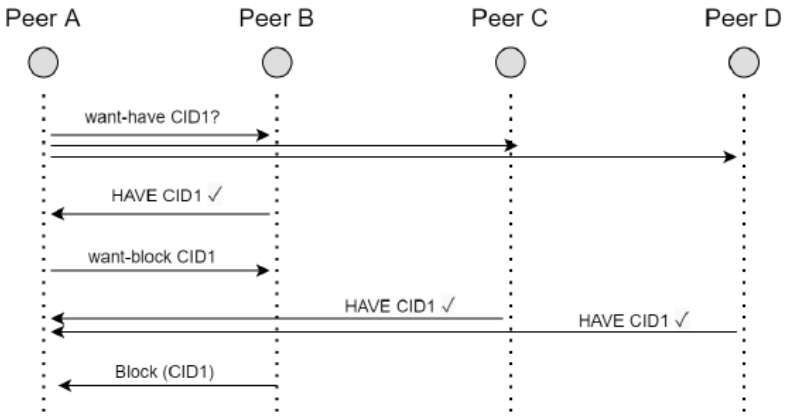
- In IPFS, every file or directory has a CID
  - Unique SHA256 hash used to identify the file.
- Whenever the content changes, the CID also changes.
- To keep track of files after being altered, IPFS uses the InterPlanetary Name System (IPNS) where the name is the hash of a public key, stored in the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

Bitswap examples

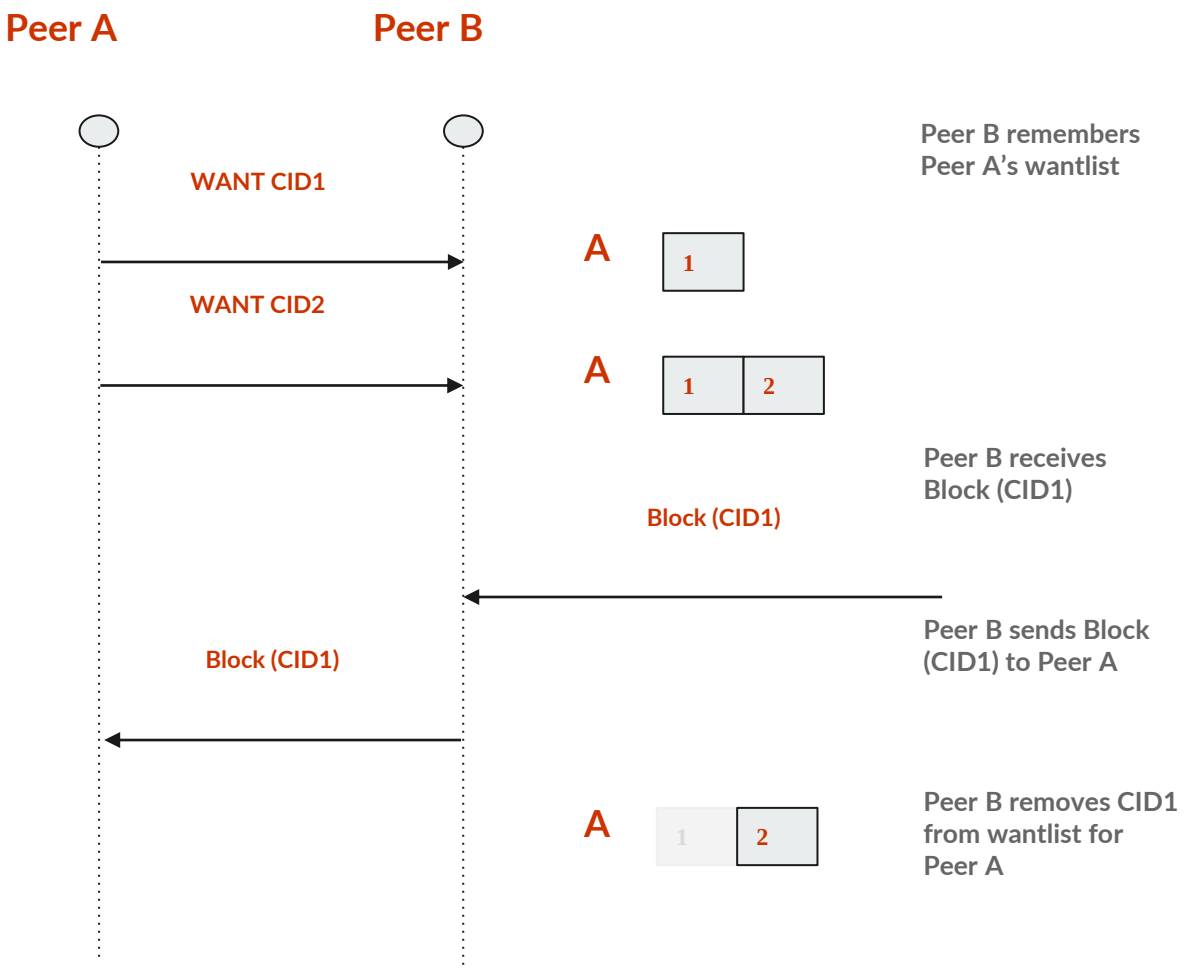
- When a peer wants a block, it broadcasts a Want to all connected peers
- If there is no response, it asks the DHT who has root CID. Peers who respond are added to the session and subsequent requests are sent only to peers in the session





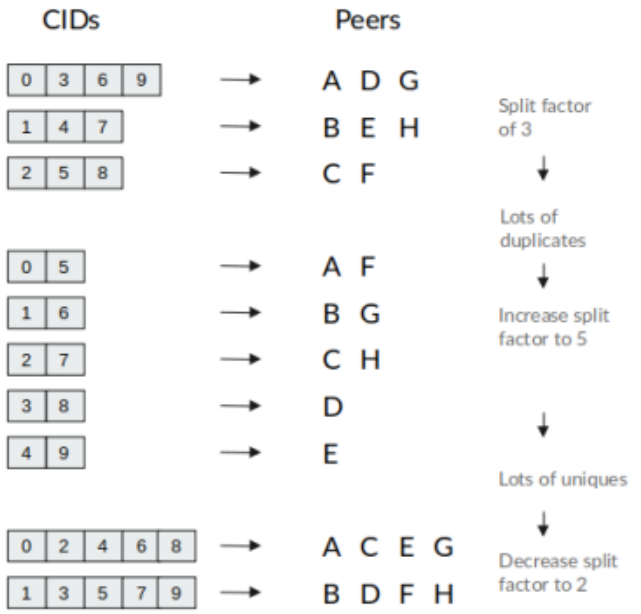


Bitswap examples



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**

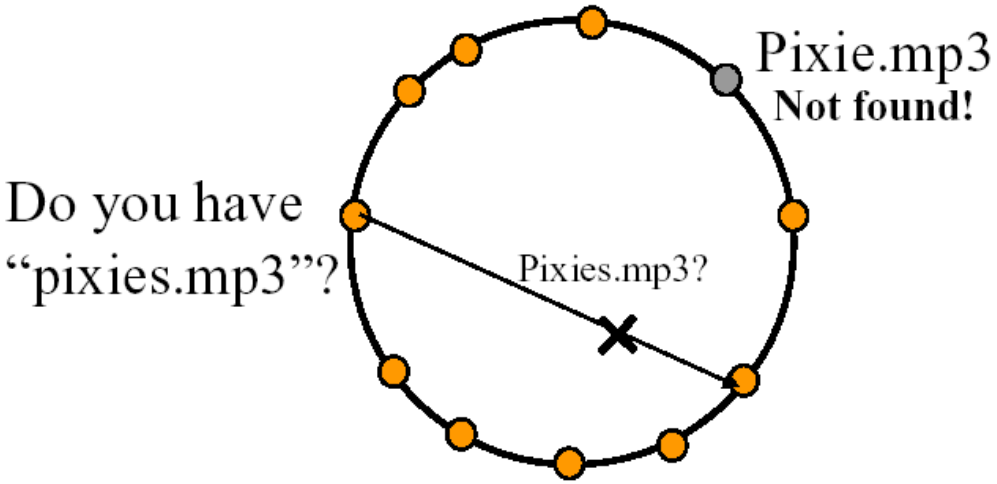
Distributed Hash Tables (DHTs)

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

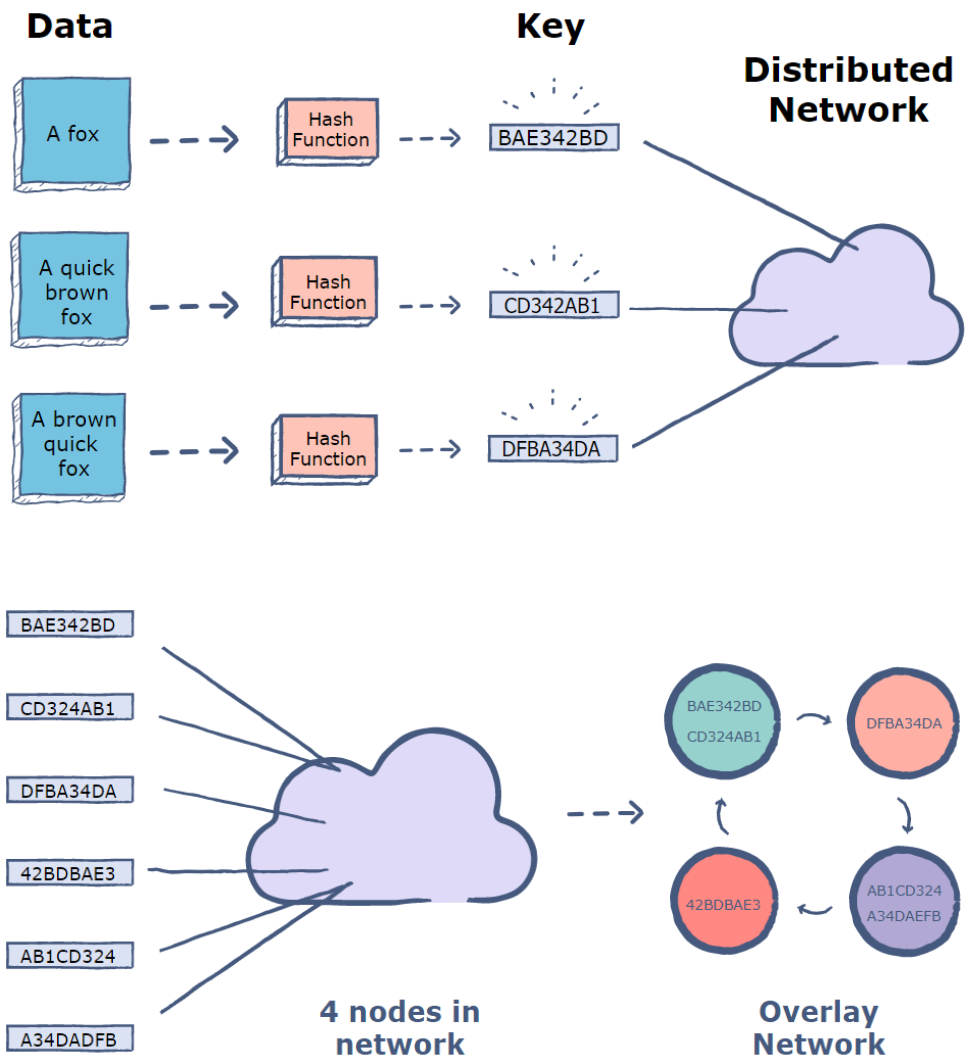
https://pdos.csail.mit.edu/papers/ton:chord/paper-ton.pdf

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
A quick brown fox
A brown quick fox
Hash Function
Hash Function
Hash Function
BAE342BD
CD342AB1
DFBA34DA
BAE342BD
CD324AB1
DFBA34DA
42BDBAE3
AB1CD324
A34DADFB
4 nodes in network
BAE342BD
CD324AB1
DFBA34DA
42BDBAE3
AB1CD324
A34DAEFB
Overlay Network

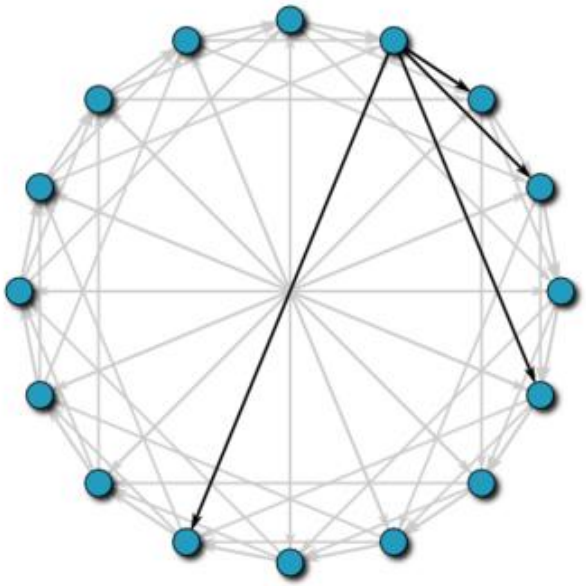
CHORD: DHT Algorithm

- 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

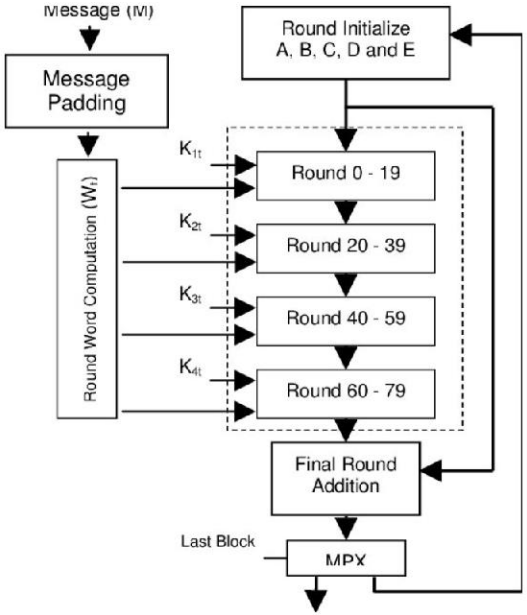
and then send a request using this key.

- 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



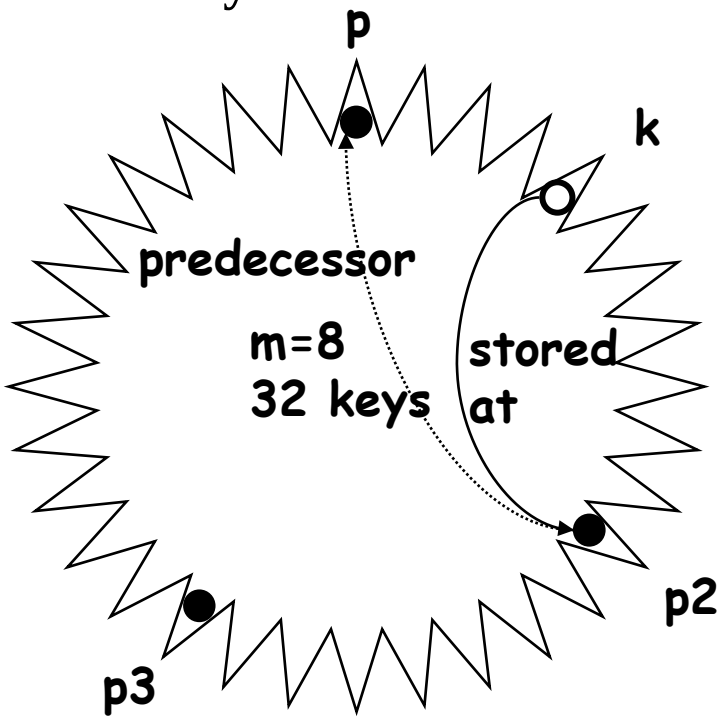
CHORD: DHT Algorithm

- 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

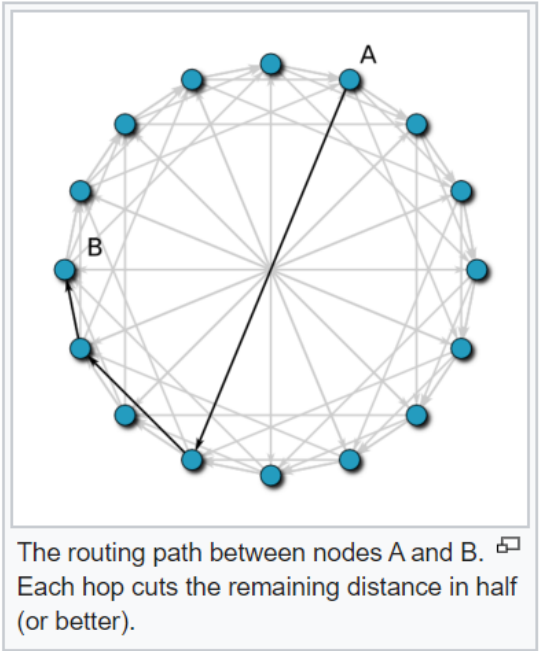
- **Hashing of search keys AND peer addresses on binary keys of length m**
  - **e.g. m=8, key("jingle-bells.mp3")=17, key(196.178.0.1)=3**
- **Calculates hash of data to get *k***
- **Routing is used to find the node that stores key *k* (*$node_k$*)**
- **Data keys are stored at next larger node key**



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

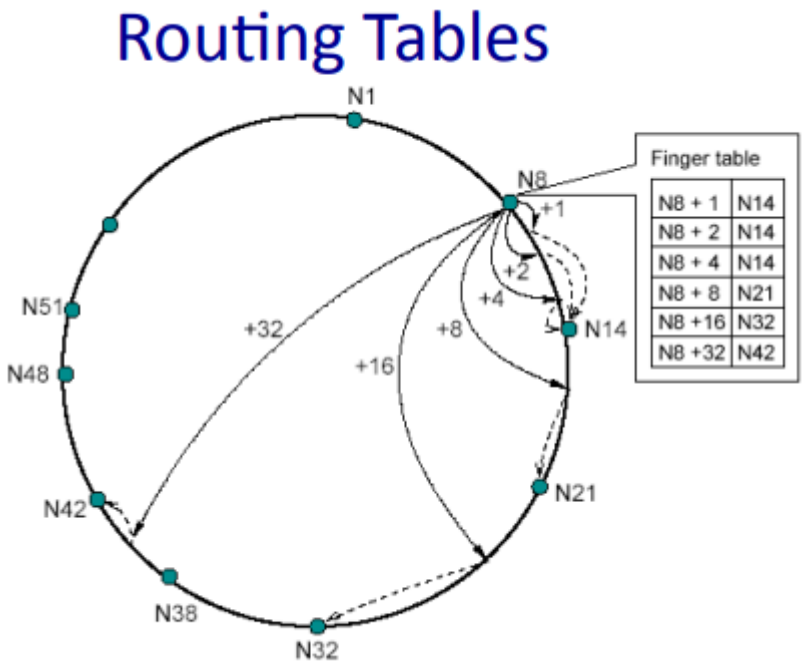
DHT: Search Information

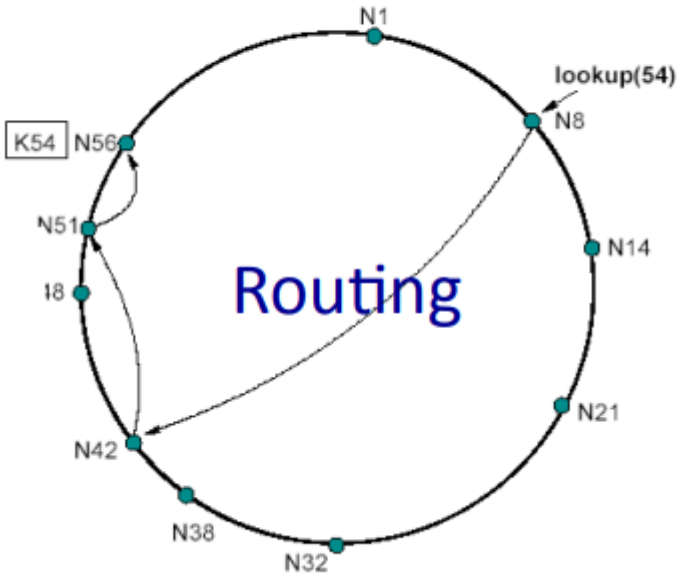
- When a node wants a content
  - Hashes data identifier and sends a request to successor(*k*)
  - Reply with the IP of the node that holds the actual data
  - How does a node request information from successor(*k*), when it doesn't know its IP, but only the key?
    - Every node holds what is known 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*



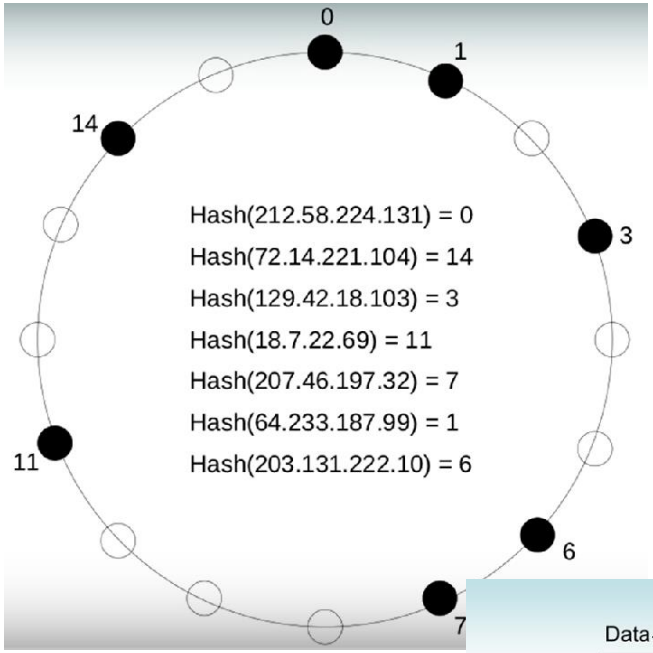
The routing path between nodes A and B. Each hop cuts the remaining distance in half (or better).

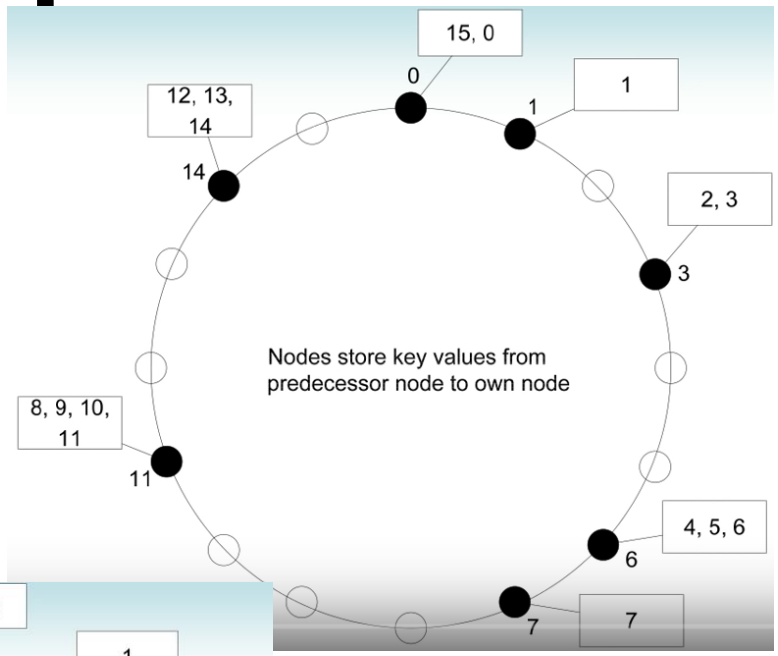
Search information: Finger Table

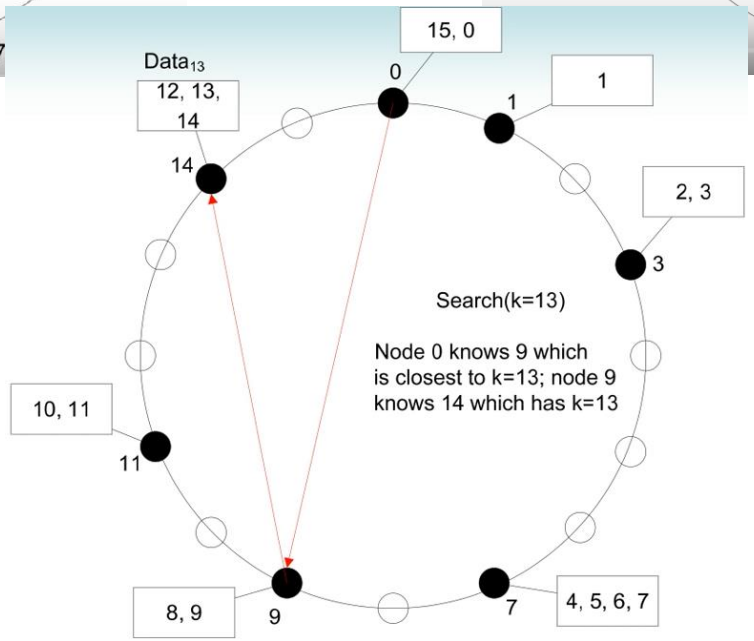




Example


0
1
14
3
11
6
7
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


15, 0
0
1
12, 13, 14
1
14
2, 3
3
Nodes store key values from predecessor node to own node
8, 9, 10, 11
11
4, 5, 6
6
7
7

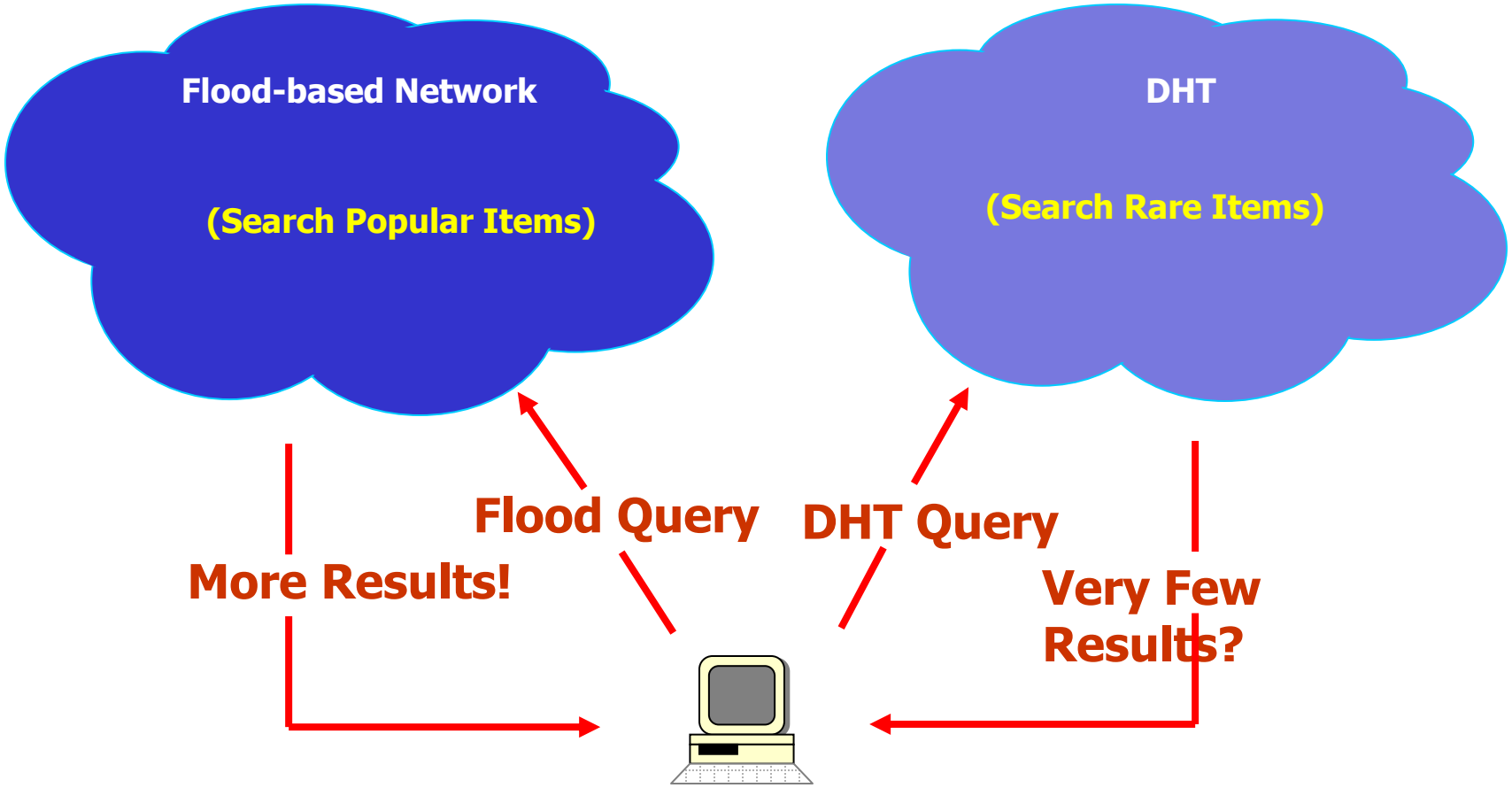

Data13
15, 0
12, 13, 14
0
1
1
14
2, 3
3
Search(k=13)
Node 0 knows 9 which is closest to k=13; node 9 knows 14 which has k=13
10, 11
11
8, 9
9
7
4, 5, 6, 7

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

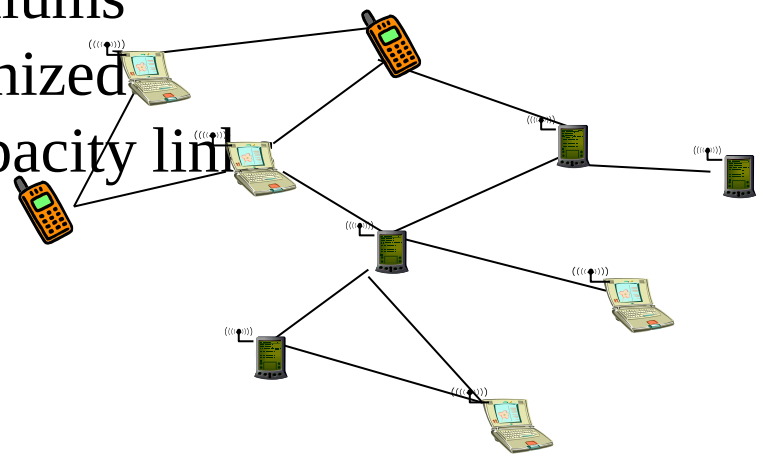
Ad-Hoc Networks

Mestrado em Engenharia de Computadores e Telemática

2022/2023

Mobile Ad-hoc networks

- Terminals may appear and disappear anywhere and anytime, and may freely move
- Nodes can act as routers or terminals
- Networks independently formed, can be merged and splitted anytime
- Dynamic topologies
- Coexistence of different access mediums
- Network is intelligent and self-organized
- Bandwidth constrained, variable capacity link
- Energy constrained operation
- Limited physical security



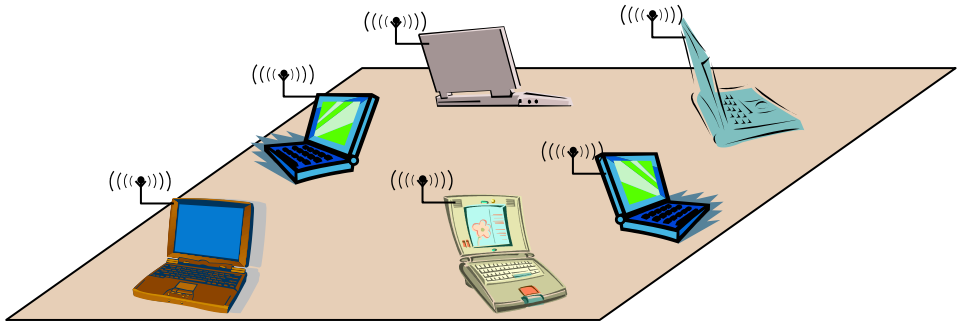
Challenges in Mobile Environments – Ad-hoc increases them

- Limitations of the wireless network
  - Lack of central entity for organization available
  - Limited range of wireless communication
  - Packet loss due to transmission errors
  - Variable capacity links
  - Frequent disconnections/partitions
  - Limited communication bandwidth
  - Broadcast nature of the communications
- Limitations imposed by mobility
  - Dynamically changing topologies/routes
  - Lack of mobility awareness by system/applications
- Limitations of the mobile computer
  - Short battery lifetime
  - Limited capacities

Build a wireless ad-hoc network

- Try to build a network without infrastructure, using networking abilities of the participants
  - This is an ad-hoc network – a network constructed "for a special purpose"

- Simplest example: Laptops in a conference room – a single-hop ad-hoc network



Application Scenarios

Ad-hoc applications

- Personal area networking
  - Cell phone, laptop, ear phone, wrist watch
- Military environments
  - Soldiers, tanks, planes
- Civilian environments
  - Taxi cab network
  - Meeting rooms
  - Sports stadiums
  - Boats, small aircraft
- Emergency operations
  - Search-and-rescue
  - Policing and fire fighting

Usage scenarios – in general

- Setting up of fixed access points and backbone infrastructure is not always viable
  - Infrastructure may not be present in a disaster area or war zone
  - Infrastructure may not be practical for short-range radios; Bluetooth (range ~ 10m)
- Ad-hoc networks
  - Do not need backbone infrastructure support
  - Are easy to deploy
  - Useful when infrastructure is absent, destroyed or impractical
- Or when the objective is to have
  - Self-adapting and self-sufficient networks
  - Networks with constant changes
  - Networks that require mobility
  - Moving networks
  - Requirement to absent any external configuration and management process

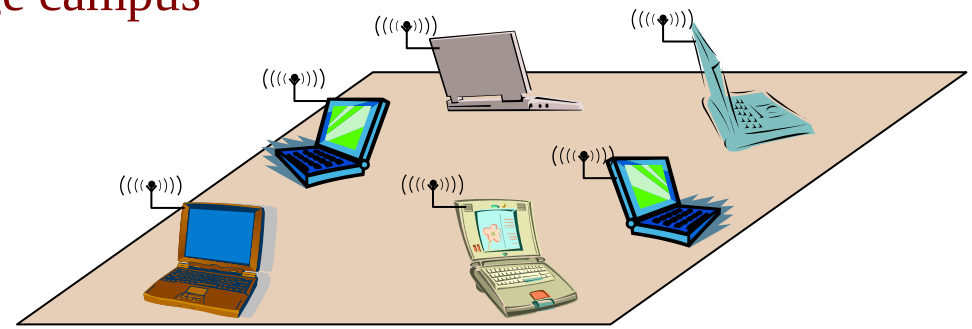
Civilian environments

- Computer science classroom
  - Ad-hoc network between student laptops
- Conference
  - Users in different rooms accessing services through other users
- Shopping mall, restaurant, coffee shops
  - Customers spend part of the day in a networked mall of speciality shops, coffee shops, and restaurants
- Large campus
  - Employees of a company

  moving within a large campus

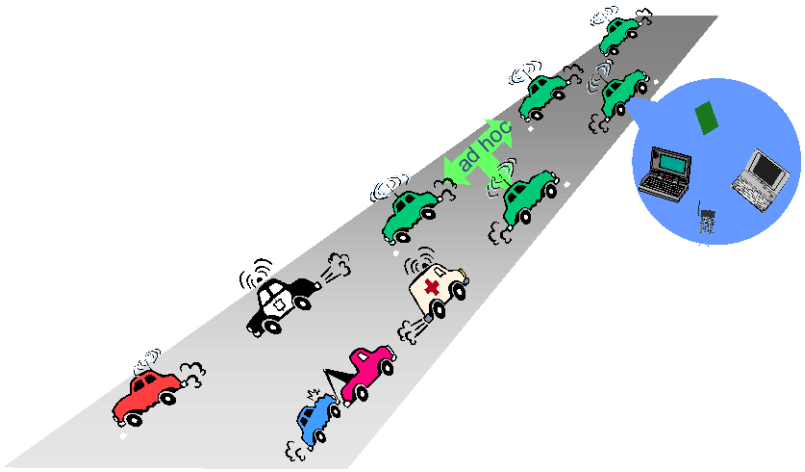
  with laptops,

  and cellphones



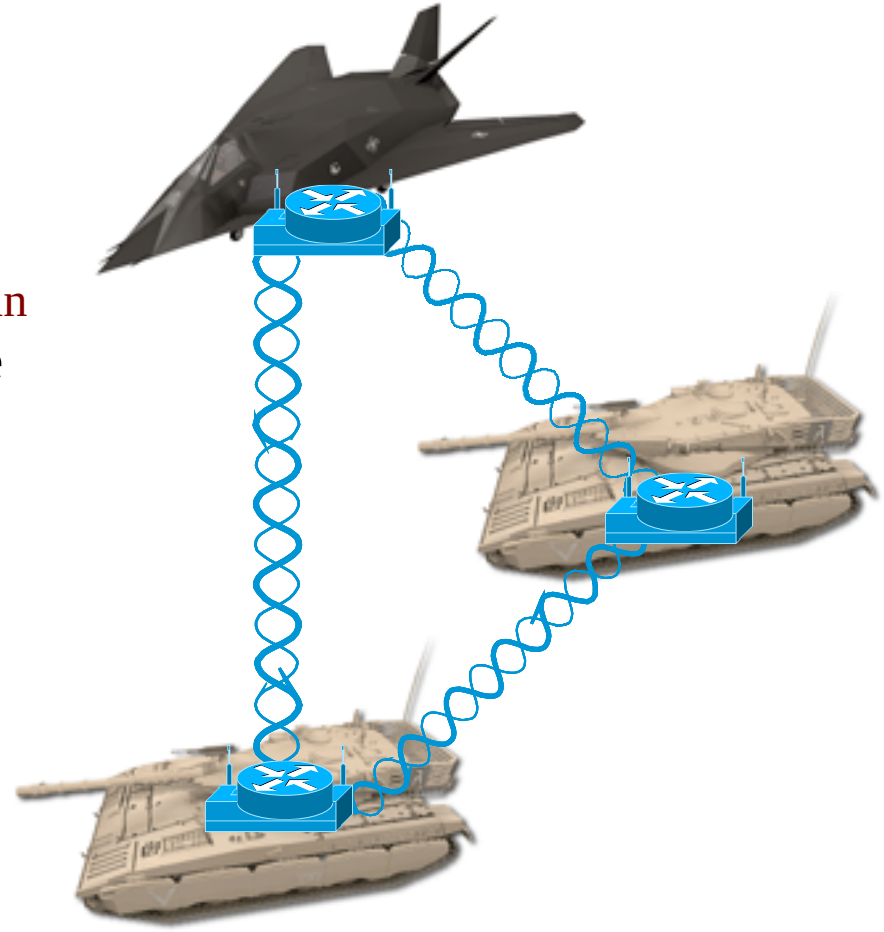
Civilian environments

- Traffic networks (smart cars and smart roads)
- Board systems talk with the road
  - Map delays and blocks
  - Obtain maps
  - Inform the road about its actions
- Finding out empty parking lots in a city, without asking a server
- Car-to-car communication



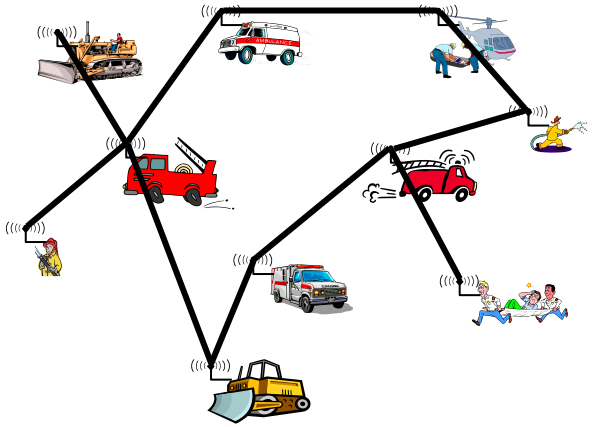
Military environments

- Combat regiment in the field
  - Around 4000-8000 objects in constant and umpredictable movement
- Force intercommunication
  - Proximity, function, battle plan
- Moving soldiers with wearable computers
  - Eavesdropping, denial-of-service and impersonation attacks can be launched
- Advantages
  - Low detection probability
  - Random topology and association between nodes

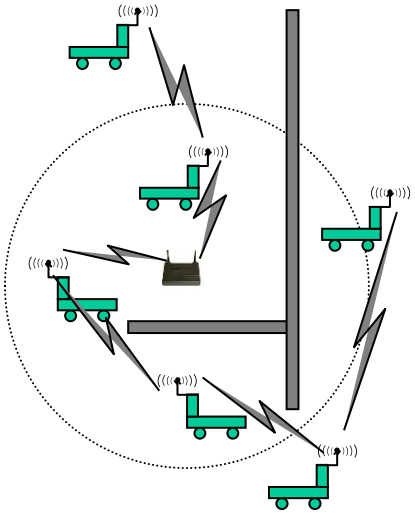


Others...

- **Disaster recovery**



- **Factory floor automation**



Routing

Routing: Challenges and Requirements

- Major challenges
  - Mobility – path breaks, packet collisions, transient loops
  - Bandwidth constraint – channel shared by all nodes in the broadcast region
  - Error-prone and shared channel – take into account the larger BERs in wireless ad-hoc
  - Location-dependent contention – high when the number of nodes increases
- Major requirements
  - Minimum route acquisition delay
  - Quick route reconfiguration (handle path breaks)
  - Loop-free routing (avoid waste of resources)
  - Distributed routing approach (reduce bandwitdth consumed)
  - Minimum control overhead (bandwidth, collisions)
  - Scalability (scale with large network – minimize control overhead)
  - Provisioning of QoS (provide QoS levels) - support for time-sensitive traffic
  - Security and privacy (resilient to threats and vulnerabilities)

Proactive and Reactive Protocols

- Proactive protocols
  - Always maintain routes
  - Little or no delay for route determination
  - Consume bandwidth to keep routes up-to-date
  - Maintain routes which may never be used
- Reactive protocols
  - Lower overhead since routes are determined on demand
  - Significant delay in route determination
  - Employ flooding (global search)
  - Control traffic may be bursty
- Which approach achieves a better trade-off depends on the traffic and mobility patterns

Reactive routing protocols

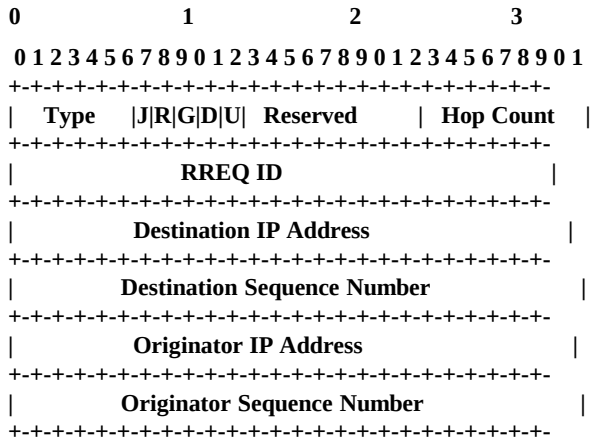
AODV - Ad Hoc On-Demand Distance Vector Routing

Ad Hoc On-Demand Distance Vector Routing (AODV)

- AODV maintains routing tables at the nodes, so that data packets do not have to contain routes
- Routes are maintained only between nodes which need to communicate

AODV operation

- Route Requests (RREQ)
- When a node re-broadcasts a Route Request, it sets up a reverse path pointing towards the source
  - AODV assumes symmetric (bi-directional) links
- When the destination receives a Route Request, it replies by sending a Route Reply (RREP)
- Route Reply travels along the reverse path set-up when Route Request is forwarded

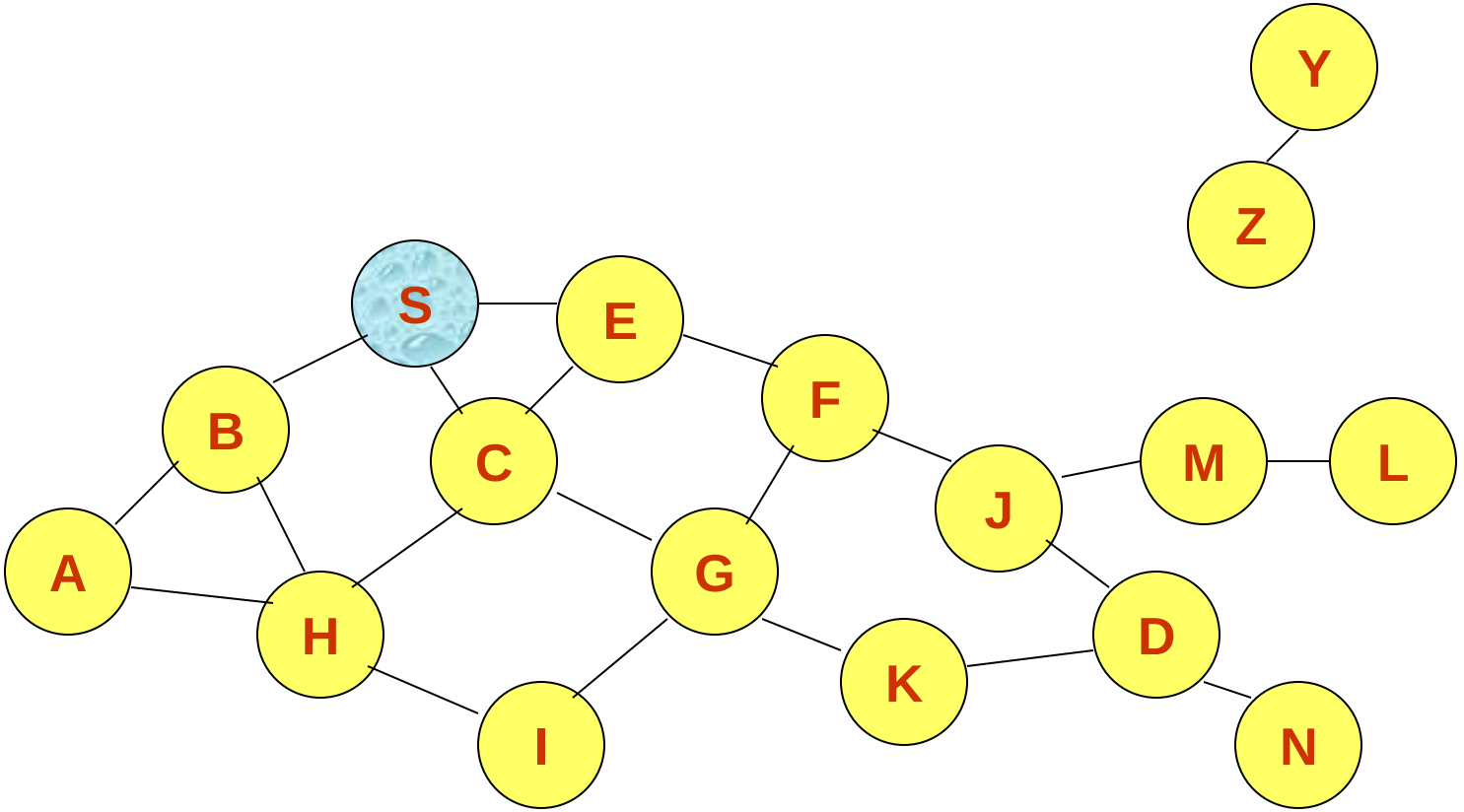


AODV operation

- Each node maintains non-decreasing sequence numbers
  - Sent in RREQ, RREP messages; incremented with each new message
  - Used to “timestamp” routing table entries for “freshness” comparison
- Intermediate node may return RREP if it has routing table entry for destination which is “fresher” than source’s (or equal with lower hop count)

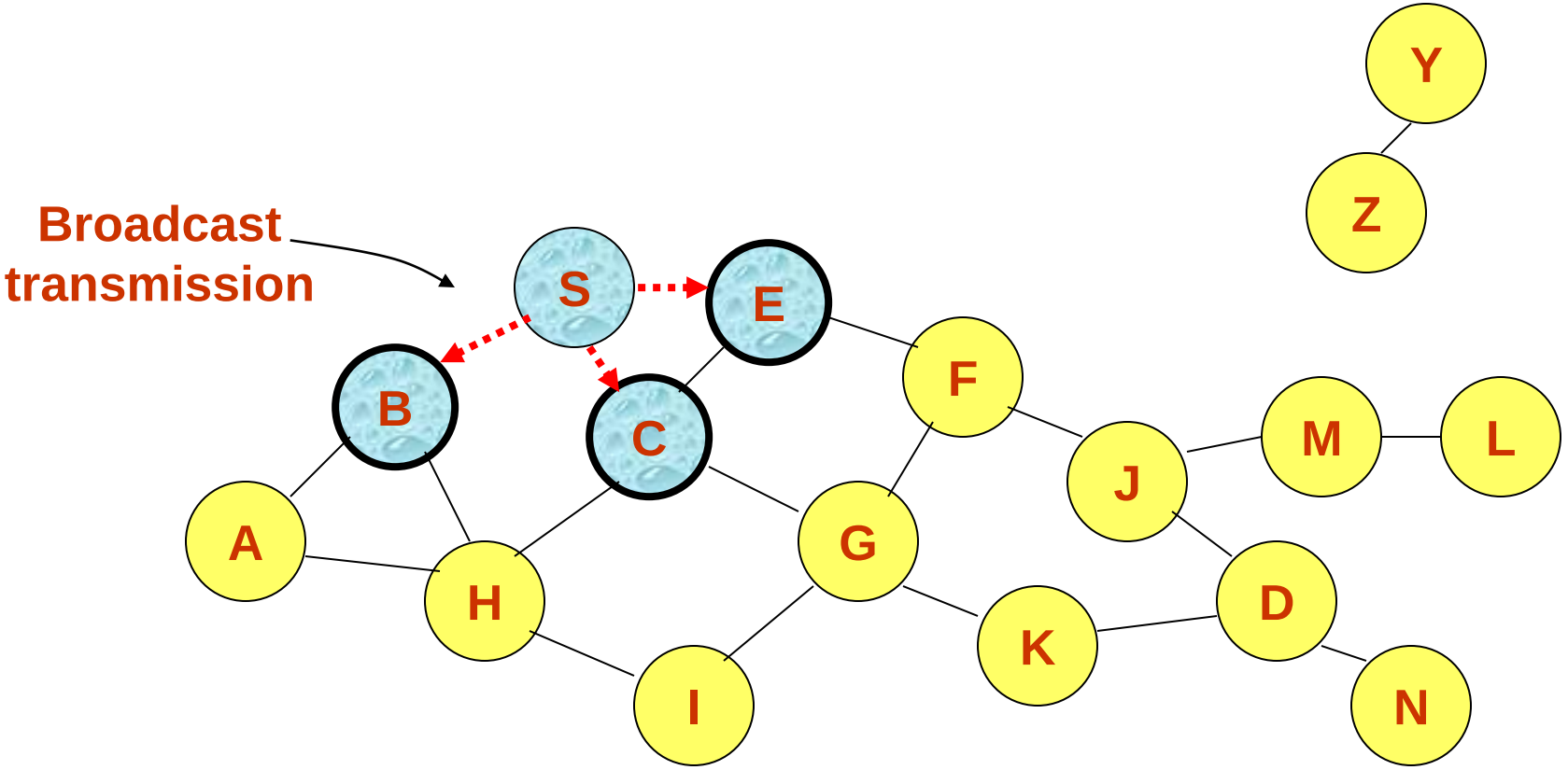
- Routing table entries assigned “lifetime”, deleted on expiration
- Unique ID included in RREQ for duplicate rejection

Route Requests in AODV



Represents a node that has received RREQ for D from S

Route Requests in AODV



Represents transmission of RREQ

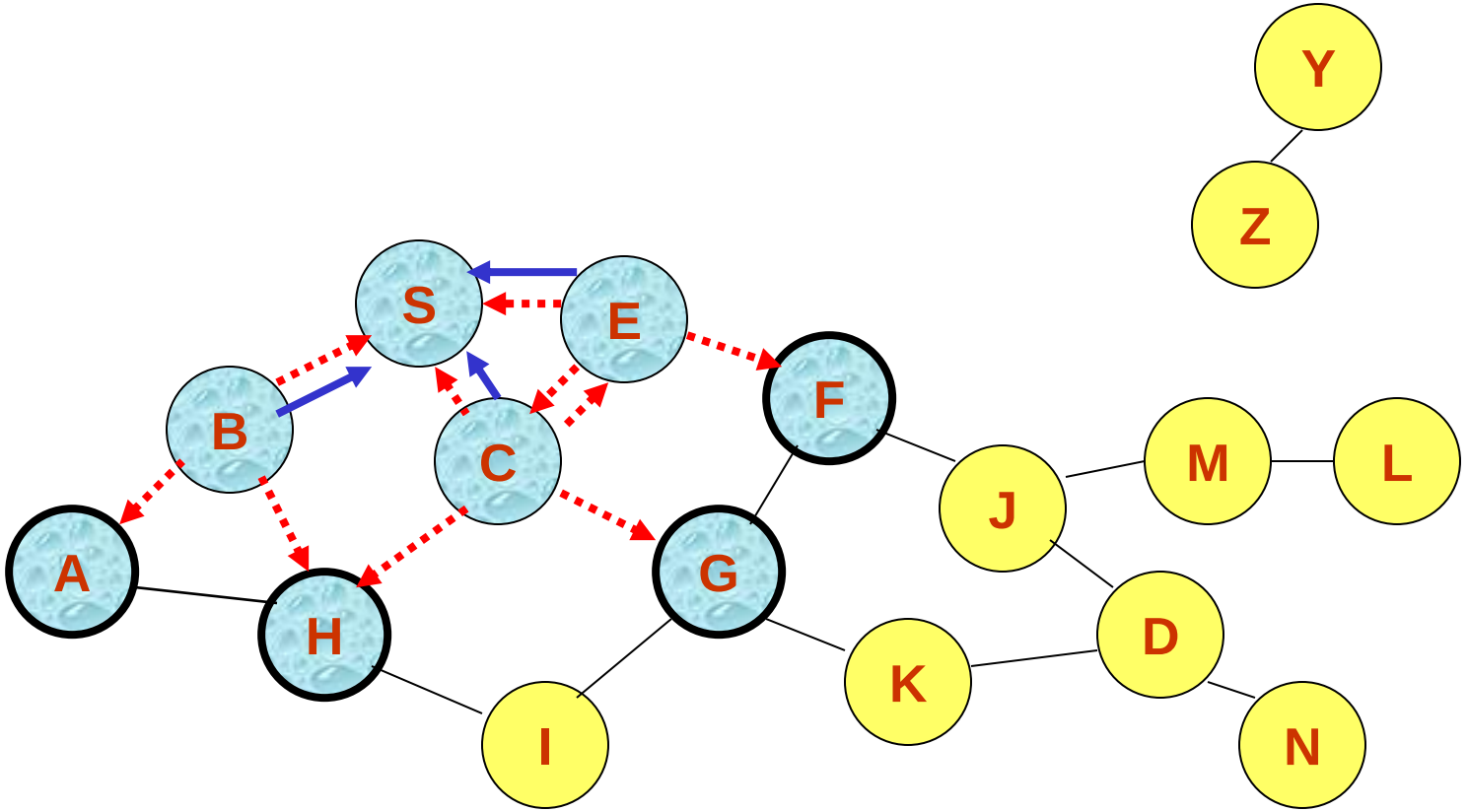
Node E receives RREQ

Makes reverse route entry for S

dest = S, next hop = S, hop cnt = 1

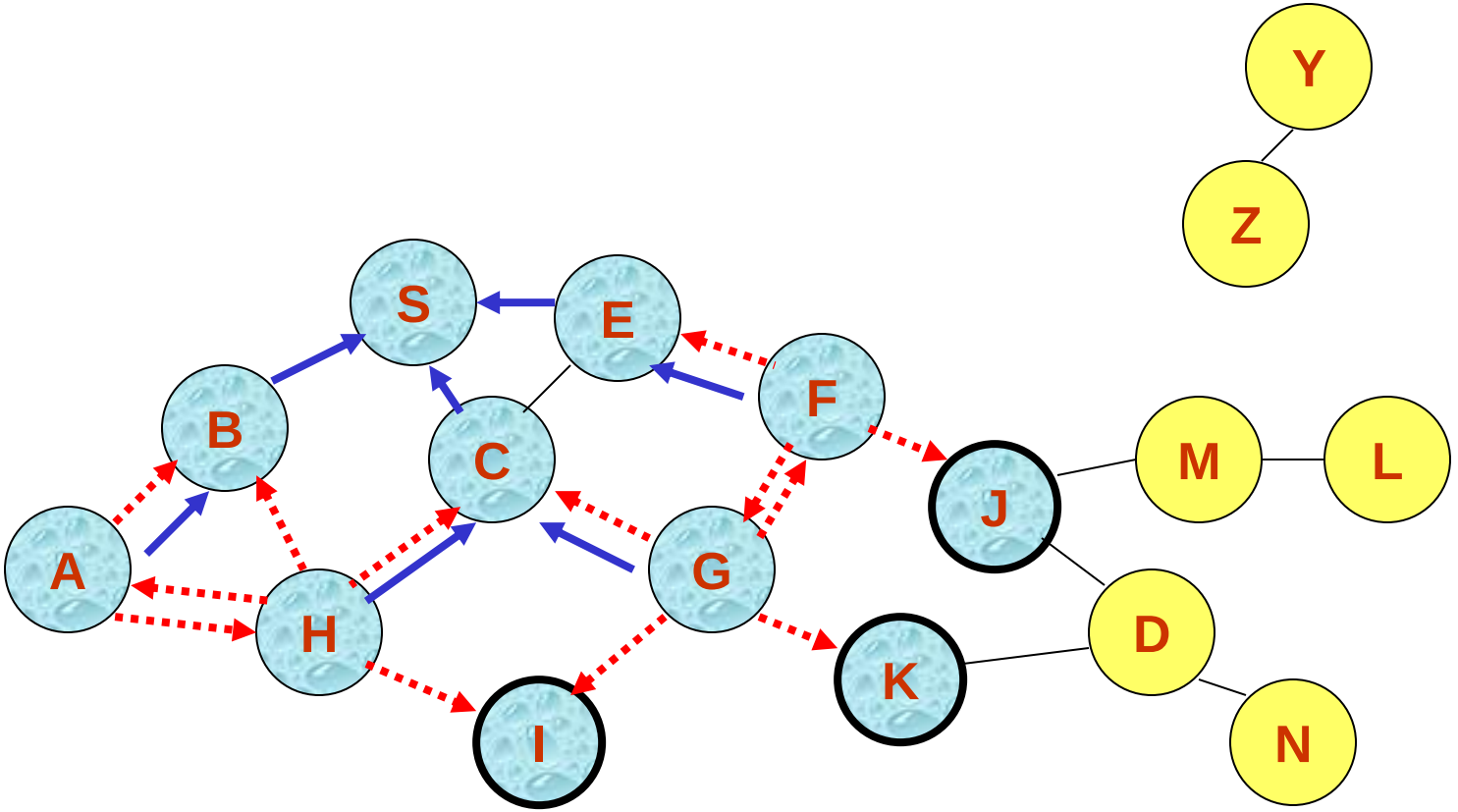
It has no route to D, so it rebroadcasts RREQ

Route Requests in AODV



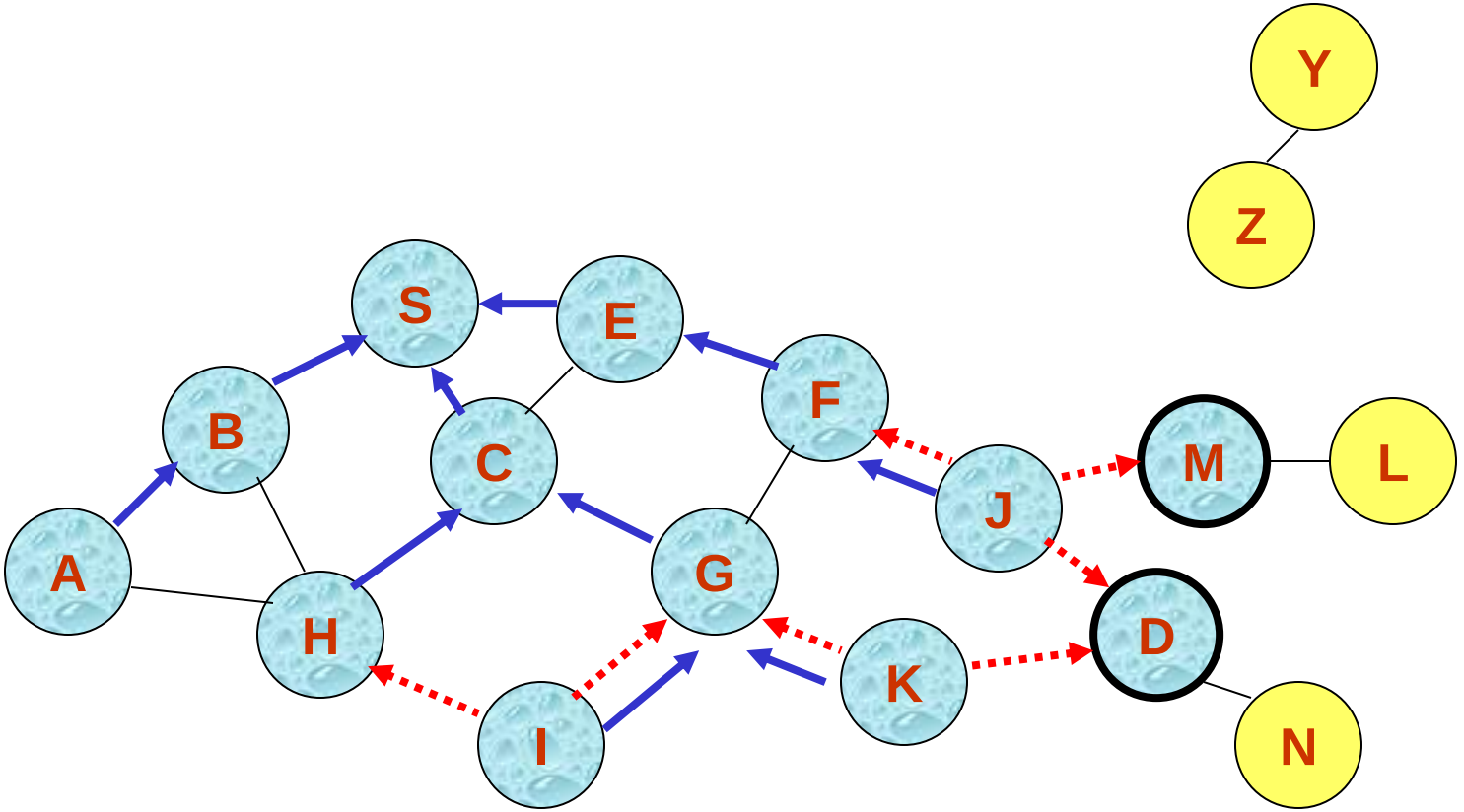
Represents links on Reverse Path

Reverse Path Setup in AODV



- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once

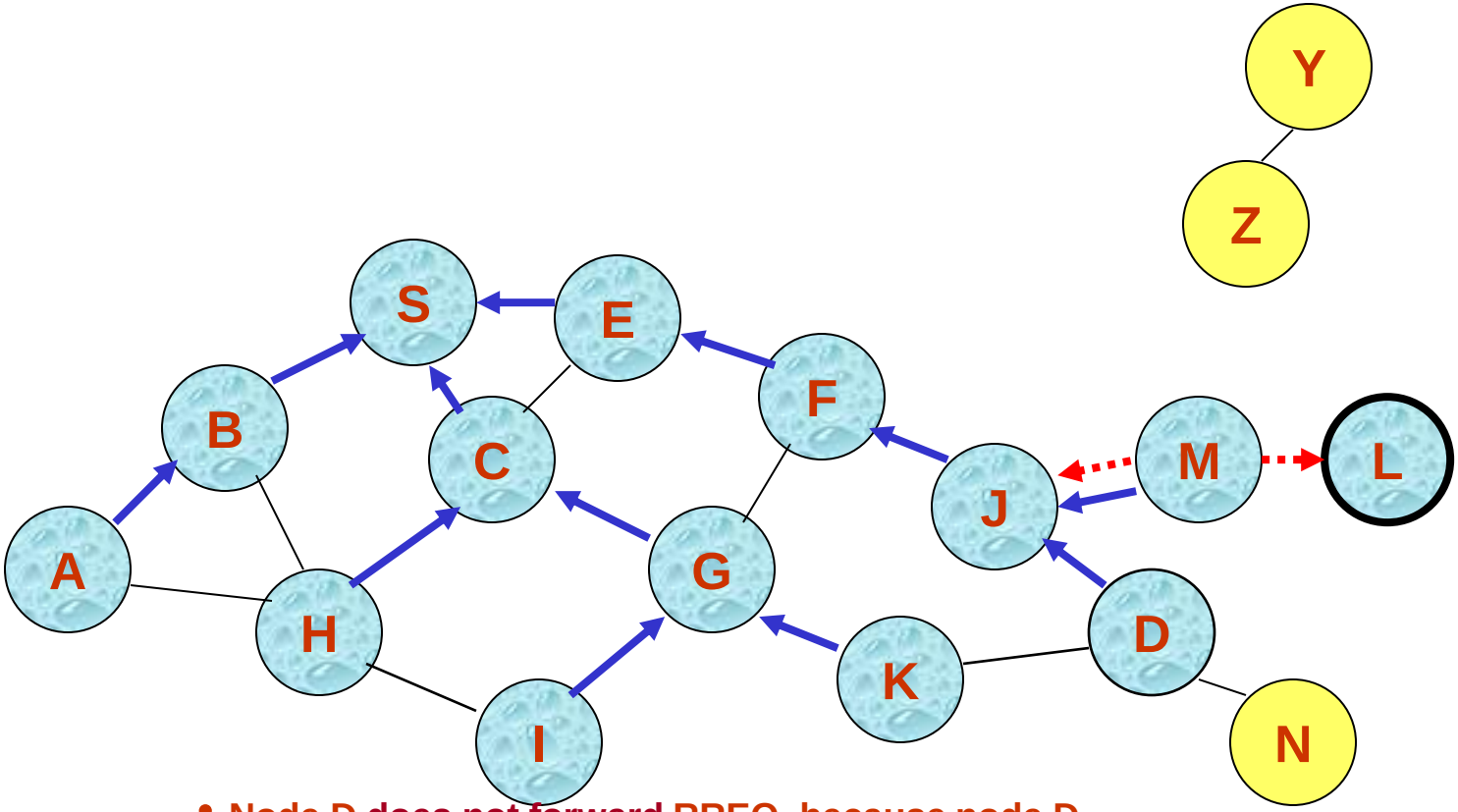
Reverse Path Setup in AODV



Node J receives RREQ

Makes reverse route entry for S, dest = S, next hop = F, hop cnt = 3
It has a route to D, and the seq# for route for D is <D's seq# in RREQ
Or
Makes reverse route entry for S, dest = S, next hop = F, hop cnt = 3
It has a route to D, and the seq# for route for D is > = D's seq# in RREQ

Reverse Path Setup in AODV



- Node D does not forward RREQ, because node D is the target of the RREQ

Node D sends RREP

D creates a Route Reply (RREP), Enters D's IP addr, seq #S's IP addr, hop count to D (=0)

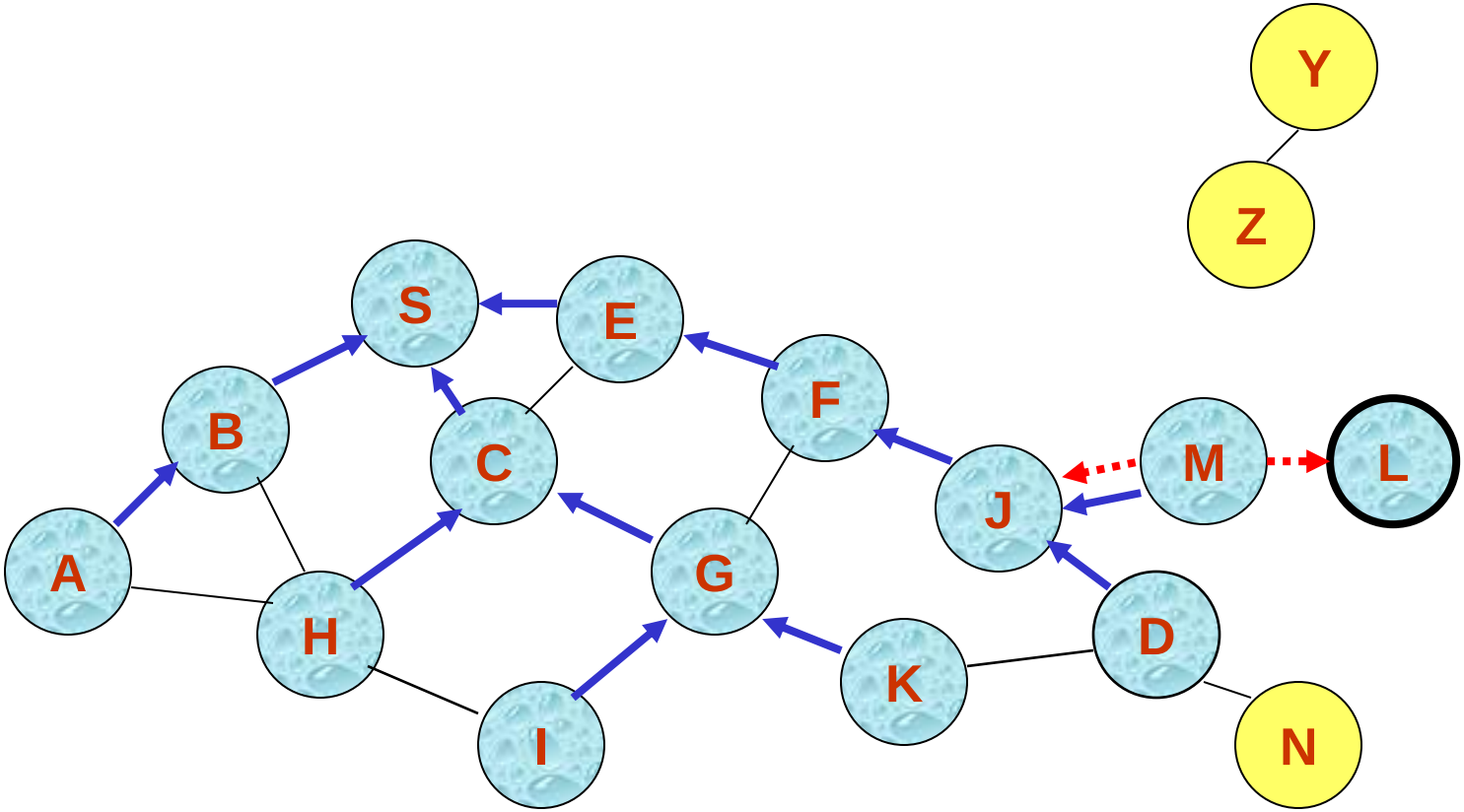
Unicasts RREP towards J

Or Node J sends RREP

J creates a Route Reply (RREP), Enters D's IP addr, seq #S's IP addr, hop count to D (=1)

Unicasts RREP towards F

Reverse Path Setup in AODV



Node E receives RREP

Makes forward route entry to D

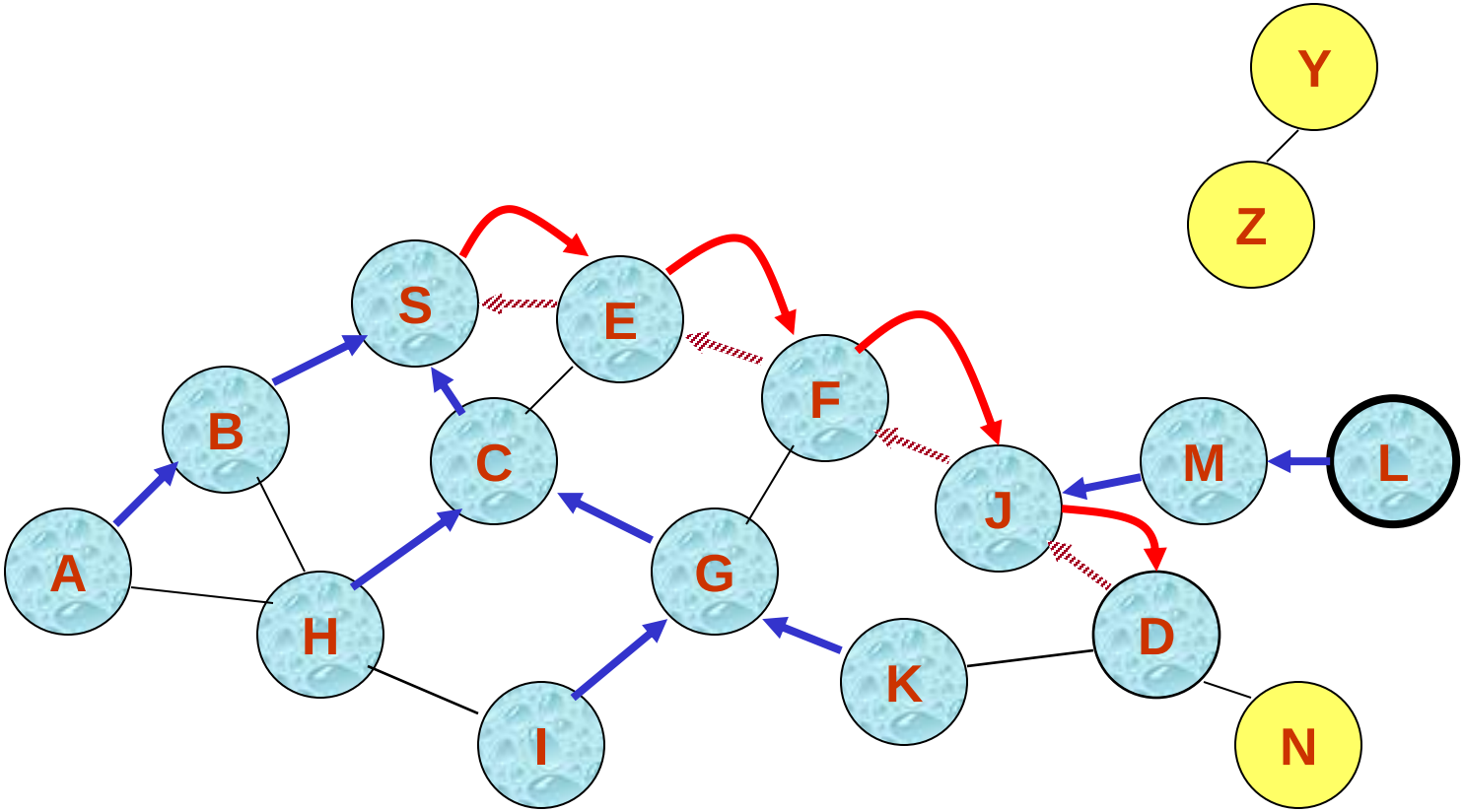
dest = D, next hop = F, hop count = 3, Lifetime, Unicasts RREP to S

Node S receives RREP

Makes forward route entry to D

dest = D, next hop = E, hop count = 4, Lifetime

Forward Path Setup in AODV



Forward links are setup when RREP travels along the reverse path

If multiple replies, uses one with lowest hop count

Represents a link on the forward path

Route Request and Route Reply

- Route Request (RREQ) includes the last known sequence number for the destination
- An intermediate node may also send a Route Reply (RREP) provided that it knows a more recent path than the one previously known to sender
- Intermediate nodes that forward the RREP, also record the next hop to destination
- A routing table entry maintaining a reverse path is purged after a timeout interval
- A routing table entry maintaining a forward path is purged if *not used* for a *active_route_timeout* interval

Link Failure

- A neighbor of node X is considered active for a routing table entry if the neighbor sent a packet within *active_route_timeout* interval which was forwarded using that entry
- Neighboring nodes periodically exchange hello messages
- Periodic route response to neighbors acts as hello, installing and refreshing route
- When the next hop link in a routing table entry breaks, all active neighbors are informed
- Link failures are propagated by means of Route Error (RERR) messages, which also update destination sequence numbers

Route Error

- When node X is unable to forward packet P (from node S to node D) on link (X,Y), it generates a RERR message
- Node X increments the destination sequence number for D cached at node X
- The incremented sequence number *N* is included in the RERR
- When node S receives the RERR, it initiates a new route discovery for D using destination sequence number at least as large as *N*
- When node D receives the route request with destination sequence number *N*, node D will set its sequence number to *N*, unless it is already larger than *N*

Local RERR

- Used when link breakage occurs
  - Link breakage detected by link-layer ACK, “passive ACK”, AODV “Hello” messages
- Detecting node may attempt “local repair”
  - Send RREQ for destination from intermediate node
- Route Error (RERR) message generated
  - Contains list of unreachable destinations
  - Sent to “precursors”: neighbors who recently sent packet which was forwarded over broken link
    - Propagated recursively

AODV: Summary

- Routes need not be included in packet headers
- Nodes maintain routing tables containing entries only for routes that are in active use
- At most one next-hop per destination maintained at each node
- Sequence numbers are used to avoid old/broken routes
- Sequence numbers prevent formation of routing loops
- Unused routes expire even if topology does not change

Proactive routing protocols

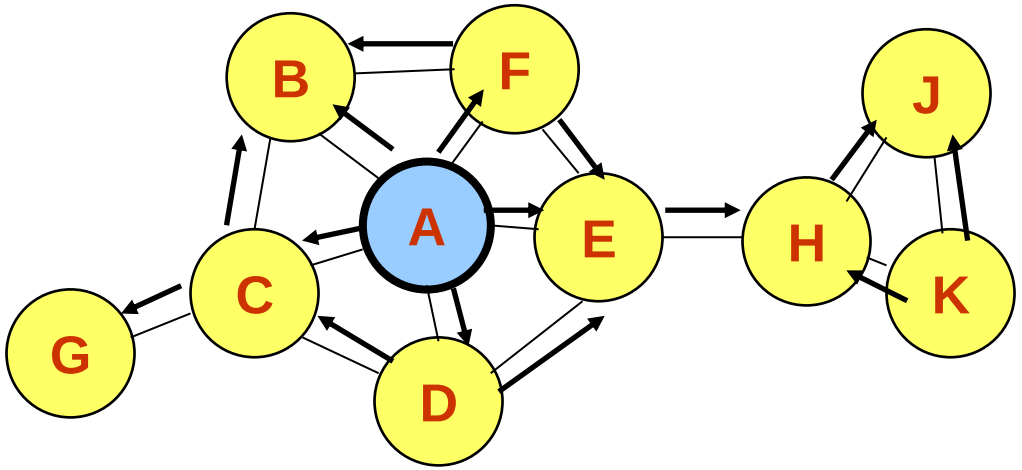
OLSR - Optimized Link State Routing Protocol

Optimized Link State Routing Protocol (OLSR)

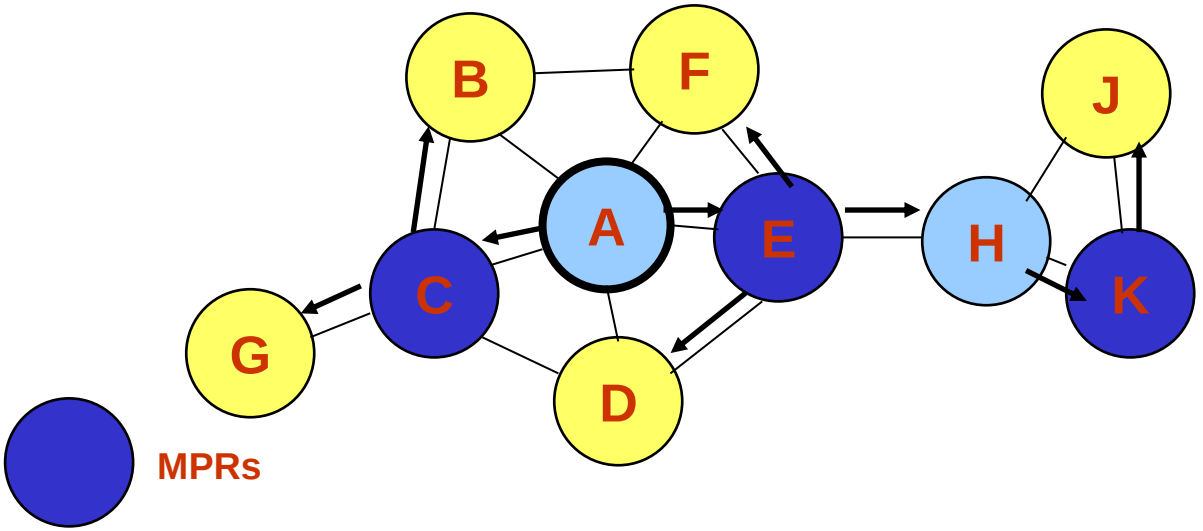
- **Proactive protocol**
- **Efficient link state packet forwarding mechanism**
  - **Multipoint relaying**
    - Reduced size of the control packets
      - Only a subset of the links in the link state updates
        - » Packet forwarding performed only by multipoint relays
    - Reduced number of links used for forwarding the link state packets
      - Multipoint relays

Example of MPR in OLSR

- **Simple flooding**

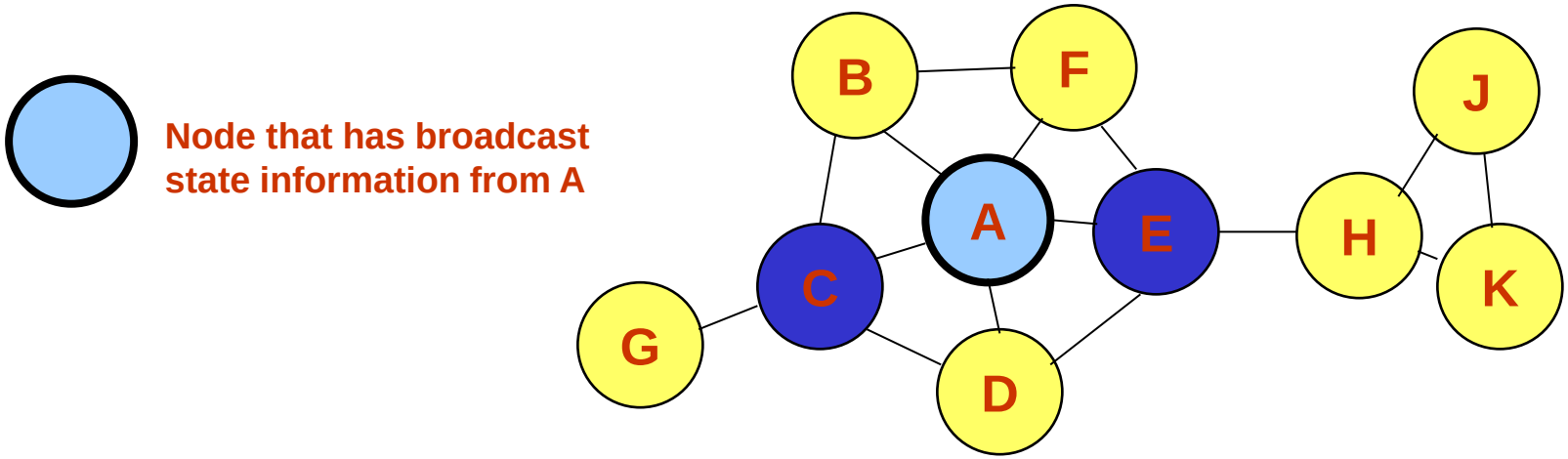


- **OLSR**



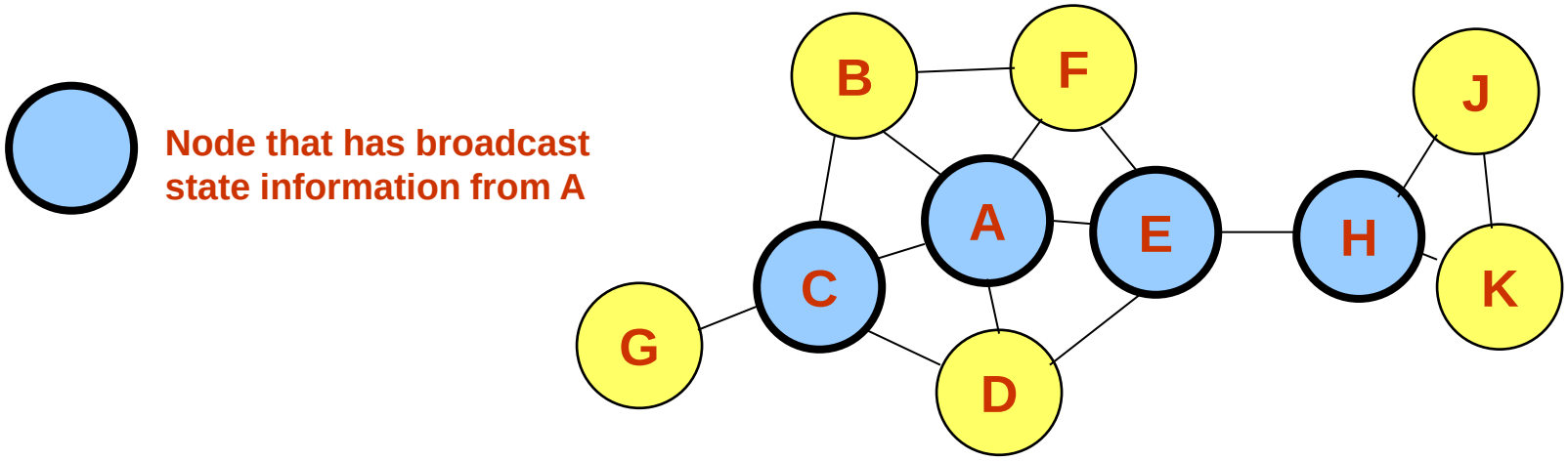
Link state forwarding

- Nodes C and E are multipoint relays of node A
  - Multipoint relays of A are its neighbors such that each two-hop neighbor of A is a one-hop neighbor of one multipoint relay of A
  - Nodes exchange neighbor lists to know their 2-hop neighbors and choose the multipoint relays

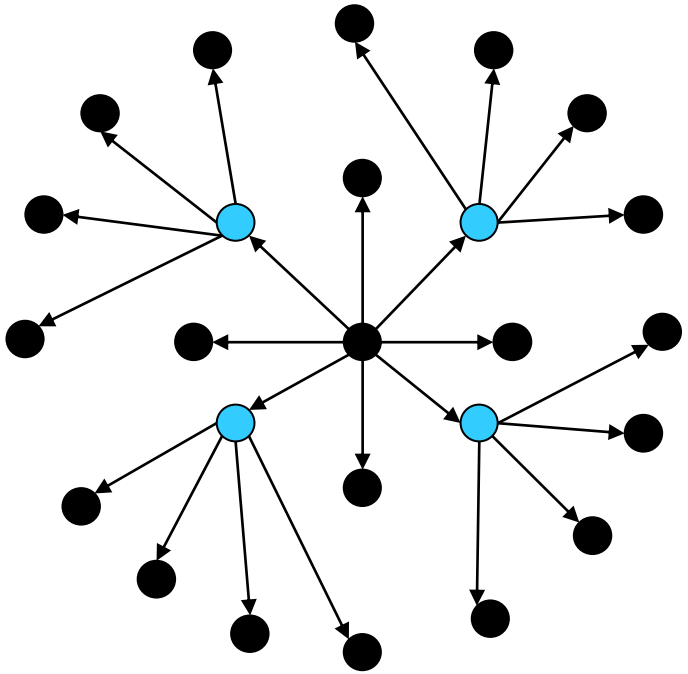


Link state forwarding

- Nodes C and E forward information received from A
- Nodes E and K are multipoint relays for node H
- Node K forwards information received from H



OLSR: Example



MPR

4 retransmission to diffuse a message up to 2 hops

MPR sets and MPR selectors

- MPR sets
  - Set of nodes that are multipoint relays
  - Each node selects an MPRset to process and forward every link state packet originated by it
  - Other nodes process the link state packets but do not forward them
- MPR selectors
  - Set of neighbors that have selected the node as multipoint relay
  - Node forwards packets received from nodes MPR selectors
- Members of MPR sets and MPR selectors change over time – efficient selection mechanisms

Selection of MPR

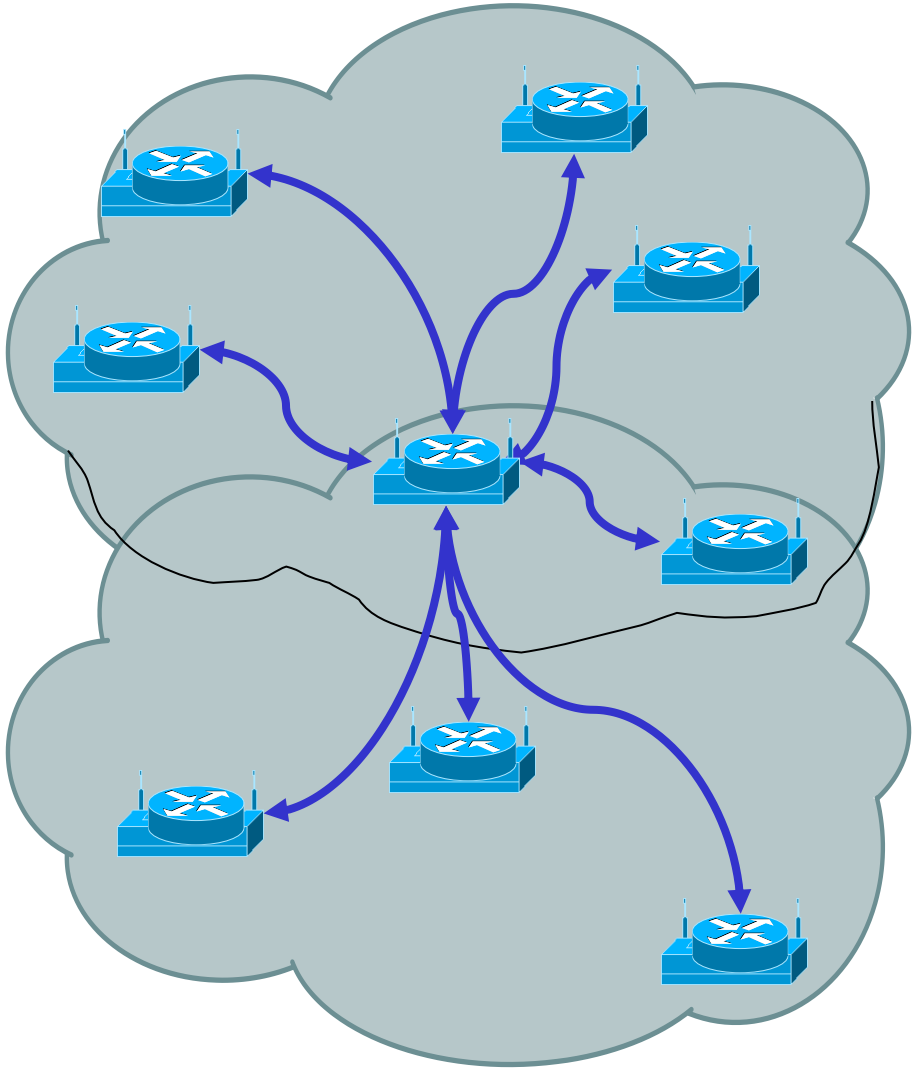
- Impact in performance
  - Node calculates routes to all destinations through the members of MPR set
- Decide on the membership of nodes in the MPR set
  - Node sends Hello messages
    - List of neighbors with which the node has bidirectional links
    - List of neighbors with whose transmissions were received in the recent past (do not know if there is bidirectionality)
  - Node receiving Hello messages
    - Update two-hop topology tables
    - Selection of multipoint relays

Selection of MPR

- Select as MPR every node in the node's two-hop neighborhood has a bidirectional link with the node
- Select as MPR, the nodes covering "isolated" nodes, i.e. for which there is a neighbor which has another node as single parent
- Select as MPR the node which covers the maximal number of nodes

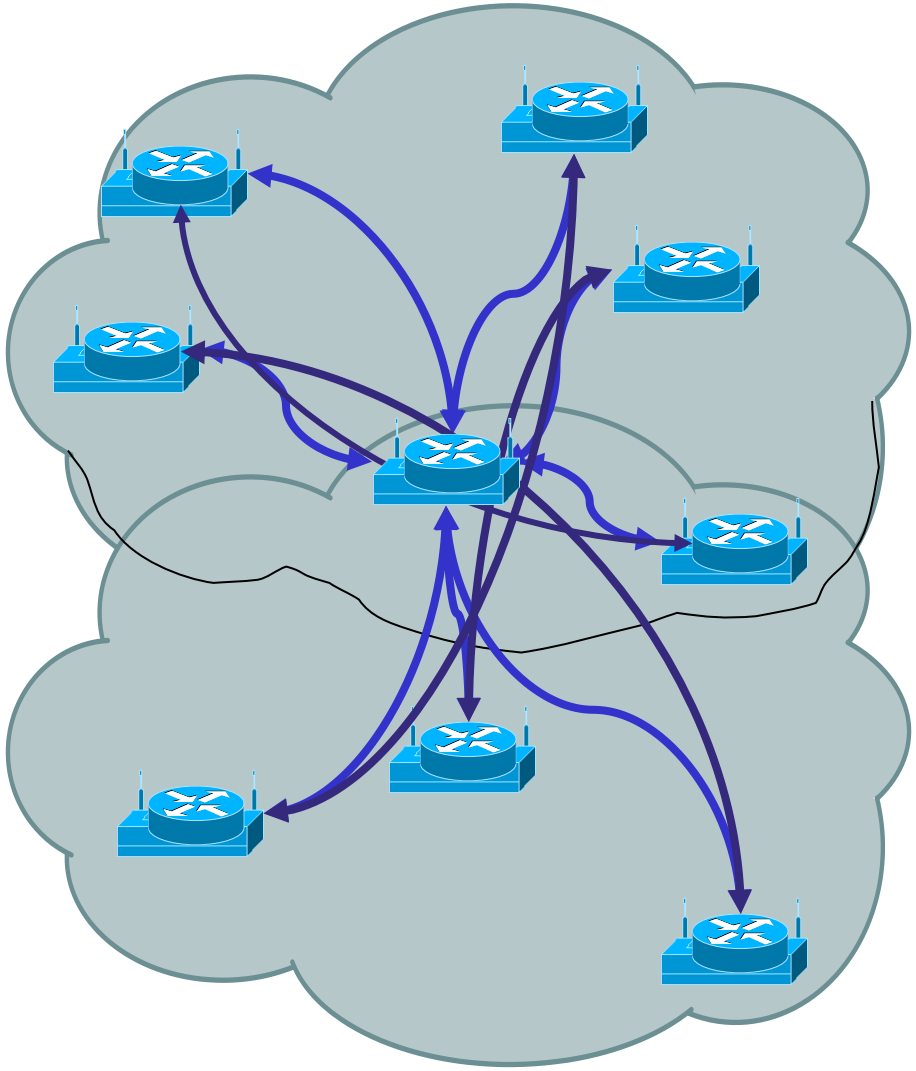
Neighbor relationships

- Each device emits a periodic “Hello”
  - –Advertise itself to its neighbors
  - –Determine who else is there
  - –Select some systems to act as MultiPoint Relays



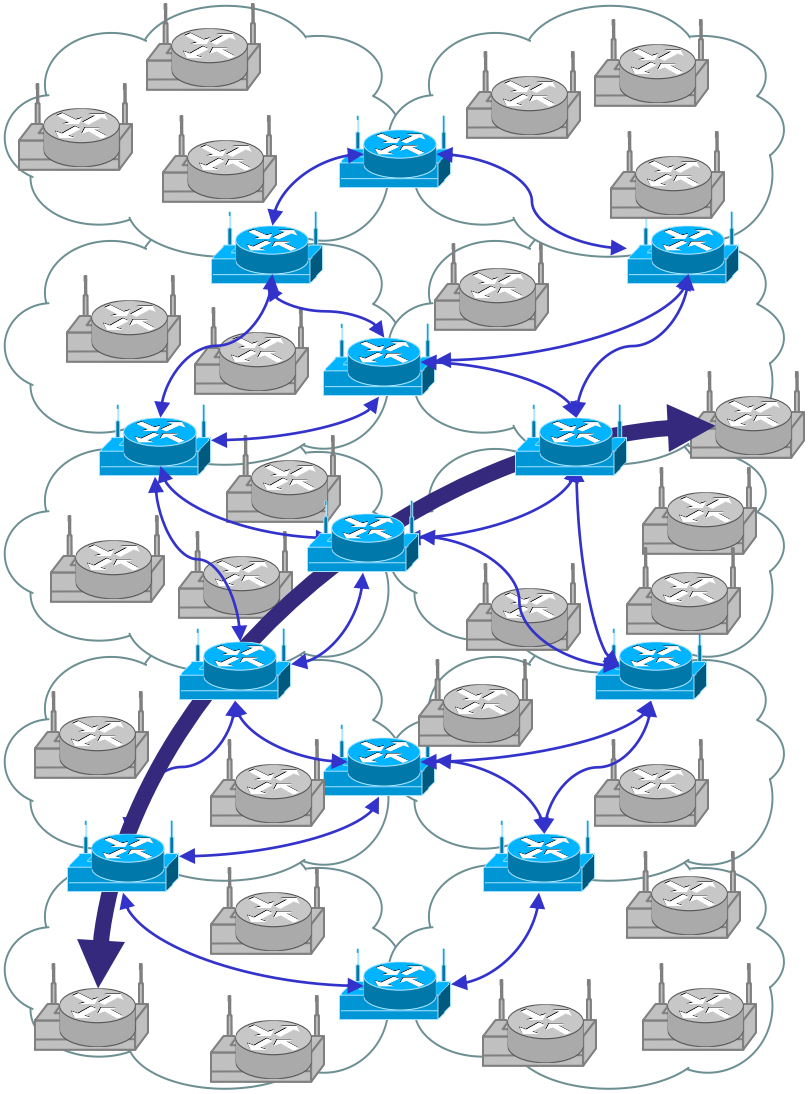
MultiPoint Relays

- Passes Topology Information
  - –Acts as router between hosts
  - –Minimizes information retransmission
  - –Forms a routing backbone



Structure of an OLSR Network

- MPRs form routing backbone
  - –Other nodes act as "hosts"
- As devices move
  - –Topological relationships change
  - –Routes change
  - –Backbone shape and composition changes



Location-based routing protocols

LAR – Location Aided Routing

The main problems of previous mechanisms

- Nodes location changes rapidly
- No information regarding
  - Current location
  - Speed
  - Direction
- Knowing the location
  - Minimizes the search zone
  - No need to flood the network
- Knowing the speed and/or direction
  - More minimization of the search zone
  - Increases the probability to find the necessary node

Location-Aided Routing (LAR)

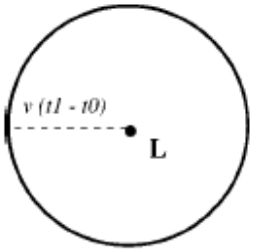
- Each node knows its location in every moment
- Using location information for route discovery
- Routing is done using the last known location + an assumption
- Route discovery is initiated when
  - S does not know a route to D
  - Previous route from S to D is broken
- Assumptions
  - Location knowledge
  - No error
  - 2D movement
  - Full cooperation

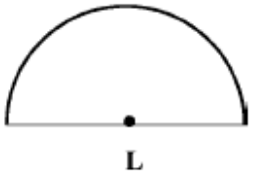
Location information

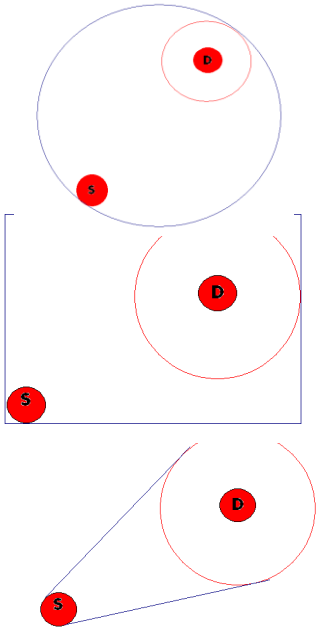
- Alignment of satellites and ground stations
- Global Positioning System (GPS) - USA
- Global Navigation Satellite System (GLONASS) - Russia
- Galileo – EU
- 3D positioning
- Accuracy 3-100 meters
- Can provide further information
  - Velocity
  - Time
- Cutting edge technology
- Already in use in many fields

LAR - Definitions

- Expected Zone (EZ)
  - S knows the location L of D in t_0
  - Current time t_1
  - The location of D in t_1 is the expected zone
  - Assume Max/Avg speed v
- Request Zone (RZ)
  - Flood with a modification
  - Node S defines a request zone for the route request
  - How to determine the size and shape of the request zone?
  - Several considerations
    - If the destination's EZ does not include the source node, other regions must be included in the RZ
    - Not always a route will be found using a certain RZ

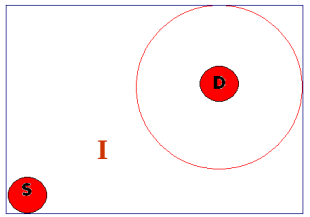






LAR – scheme 1 (Algorithm)

- Node I receives RREQ
  - Location of I – (X_i, Y_i)
  - If I is within the rectangular, I forwards the RREQ to its neighbors
  - Else I discards the RREQ
- Node D receives the RREQ
  - Replies RREP
  - Adds its current location

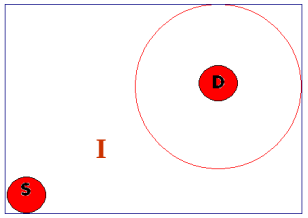


I

LAR – scheme 1 (some issues)

- The rectangular size is proportional to
  - Average speed (v)
  - Time elapsed (t_1-t_0)

  Therefore
  - Low speed ⇒ small v in the same (t_1-t_0) ⇒ smaller RZ
  - High speed ⇒ large v in the same (t_1-t_0) ⇒ larger RZ



I

- Improvements
  - D can add its speed/avg. speed in the RREP, this can help other nodes in future route discoveries
  - D can piggyback its location in other packets

BATMAN

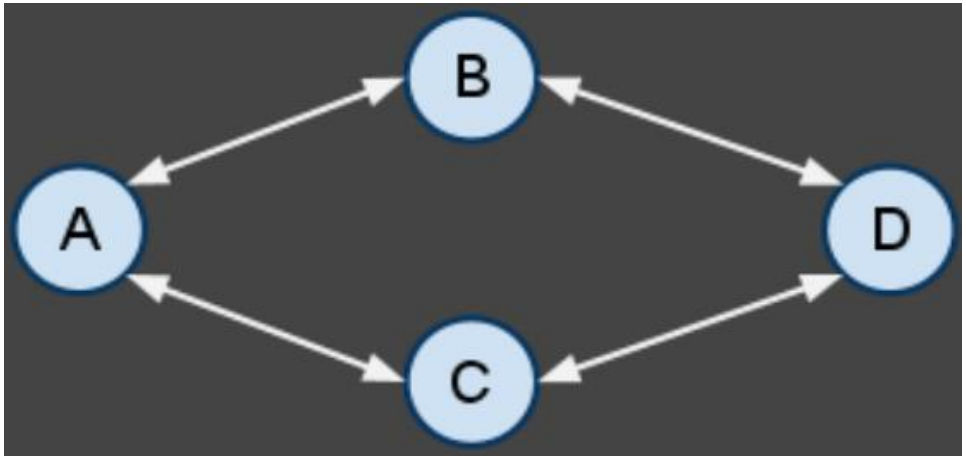
A Better Approach to Mobile Ad hoc Networking

https://www.researchgate.net/publication/320172464_Better_approach_to_mobile_ad-hoc_networking_BATMAN

https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_IV

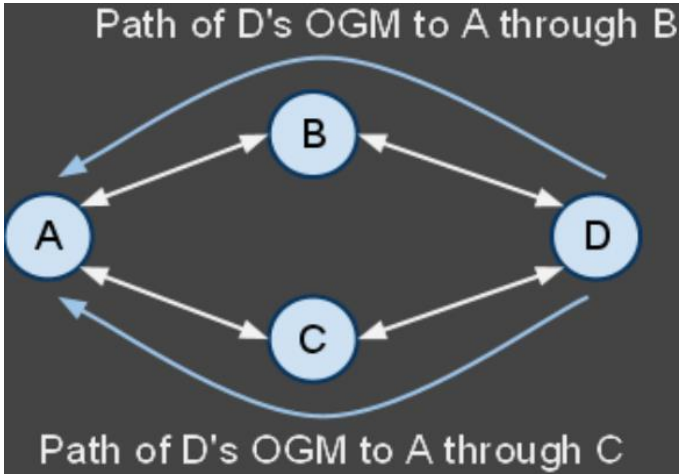
Batman

- Traditionally, nodes exchange control packets that contain information about link state (current link utilization, bandwidth, etc).
  - Nodes determine best paths based on control packets.
  - Every node must have near exhaustive information about the entire network
- BATMAN takes a very different approach:
  - The presence or absence of control packets is used to indicate link (and path) quality.



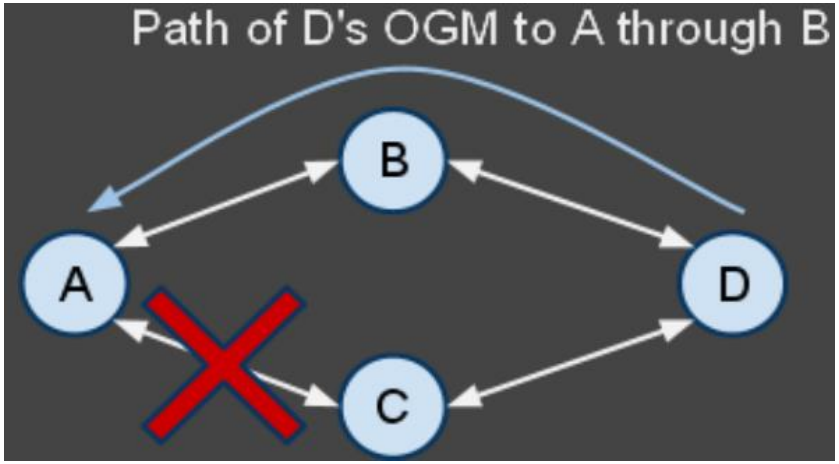
Batman Operation

- Each node has a set of direct-link neighbors
  - In the figure, node A has neighbors B and C. These are the nodes through which A sends and receives all its packets.
- Each node in the network sends an Originator Message (OGM) periodically, in order to inform all other nodes of its presence
  - OGMs include a sequence number
- If all shown links are perfect, Node A will receive node D's OGM through both of its neighbors B and C.
  - If all of D's OGMs arrive through both B and C, then when A needs to send something to D, it can use either B or C as the next hop towards the destination node D.



Batman Operation

- If the link between nodes A and C goes down
  - Node D's OGM will only arrive to A through node B.
  - Node A therefore considers node B as the best next hop neighbor for all packets destined for node D.
  - Further, Node C's OGMs will also only reach node A through node B. Node B is the best next hop for data destined for Node C.



Batman: sliding window

- If some but not all OGMs arrive through a link
  - Sliding window
- A sliding window indicates which of the last WINDOW_SIZE (in the example, 8) sequence numbers have been received
  - Uses the sequence numbers received through OGMs

	Out of Range			In Window Range								Out of Range			
Seq. Numbers:	...	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
Arrived:	...	-	-	-	1	1	1	0	1	0	1	1	-	-	-	...

Sequence numbers and sliding window

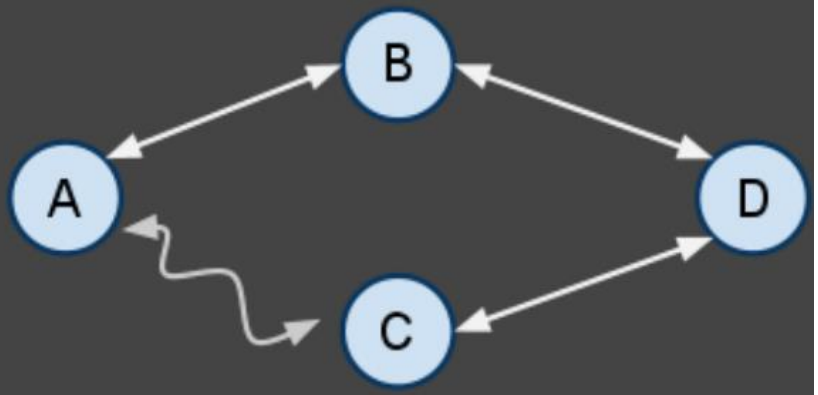
- When an out of range sequence number is received, in this case seq# 17, the window shifts up.
  - From 6 sequence numbers in-range to only 5.

Routing table

- All nodes have a sliding window for each originator (other node) in the network for each neighbor

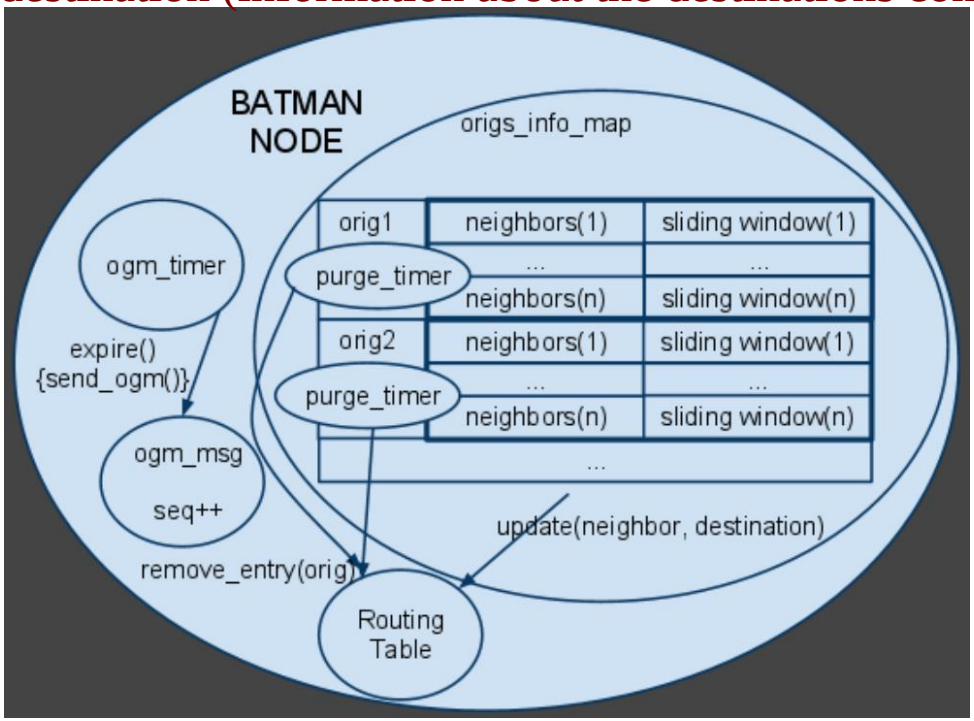
Originators	Neighbour	In Window Range Packet Count
B	B	8
	C	3
C	B	6
	C	2
D	B	7
	C	2

Information stored by node A in order to determine best next hop to each node in the network



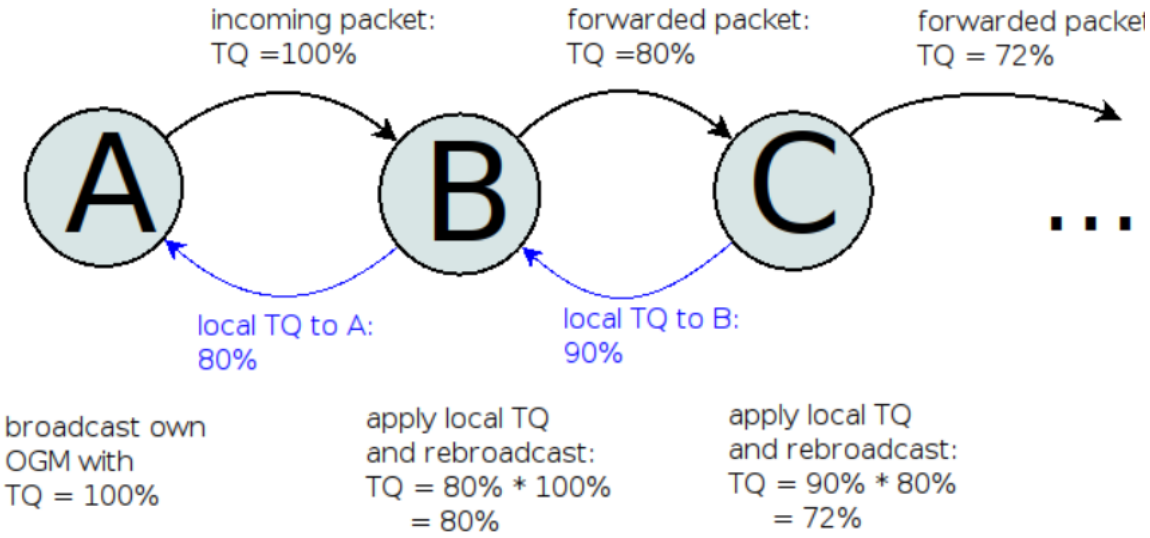
Batman Operation

- BATMAN receives information about link (and path) quality through the presence or absence of control packets.
  - Collective intelligence - retransmission of an OGM implies it arrived successfully through a best-link neighbour
  - No node needs to have exhaustive knowledge of the network, just the next hops to the destination (information about the destinations comes from OGMs)

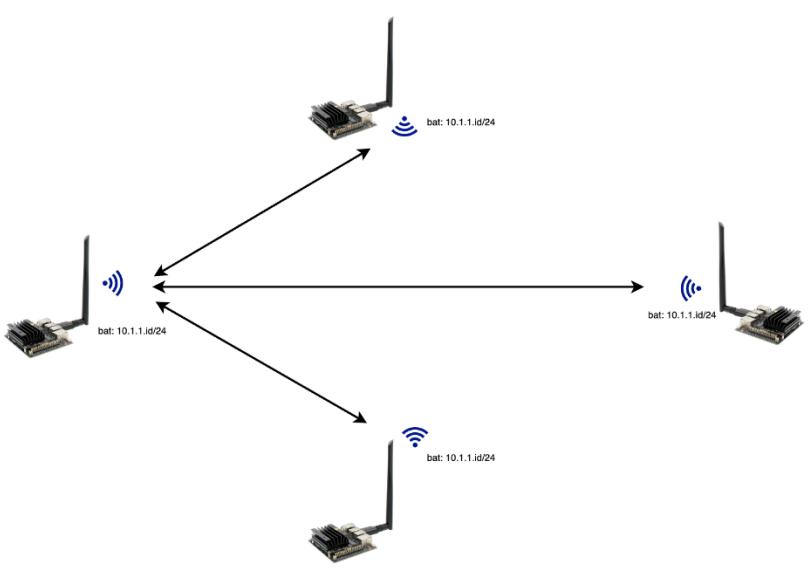


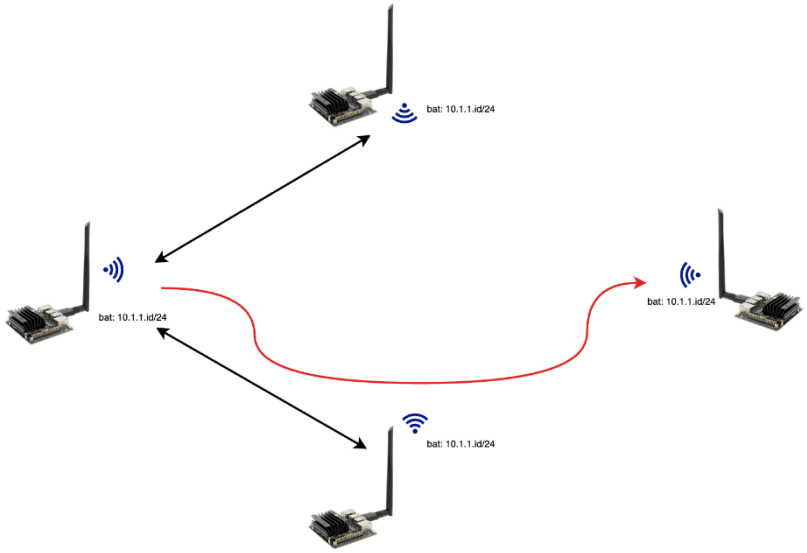
Transmission Quality (Batman v.3)

- To add the local link quality in the TQ value, the following calculation is performed:
- TQ = TQ_{incoming} * TQ_{local}
- Example: Node A broadcasts the packet with TQ max. Node B receives it, applies the TQ calculation and rebroadcasts it. When node C gets the packet it knows about the transmit quality towards node A.



Example


bat: 10.1.1.id/24
bat: 10.1.1.id/24
bat: 10.1.1.id/24
bat: 10.1.1.id/24


bat: 10.1.1.id/24
bat: 10.1.1.id/24
bat: 10.1.1.id/24
bat: 10.1.1.id/24

Comparison

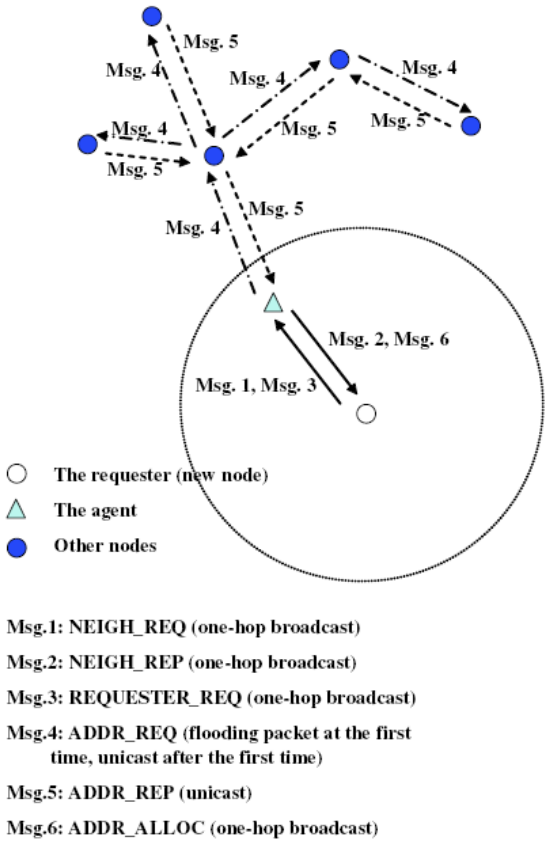
- AODV pros and cons
  - Low overhead
  - Slow discovery and recovery
- OLSR pros and cons
  - Medium overhead
  - Fast discovery and recovery
  - MPRs automation
- LAR pros and cons
  - Medium overhead
  - How to discover the location of destination?
- Batman pros and cons
  - Medium-high overhead
  - Implicit quality information
  - Fast discovery and recovery

IP Address Assignment

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1019354

https://docs.pycom.io/pymesh/

MANETconf



- Distributed mutual exclusion algorithm to check the uniqueness of the address
- Assigning an address to a new node requires an agreement from all the known nodes in the network
  - Each node has a global address allocation table maintaining currently in use addresses
  - Node (requester) broadcasts a NEIGH_REQ message to one-hop neighbors
  - Each neighbor answers back to the requester (NEIGH_REP)
  - Requester node selects one of neighbors as its agent, which performs address allocation on behalf of the requester
    - It then sends a REQUESTER_REQ to the agent to request an address
    - The agent picks a currently unused address from its table and floods an ADDR_REQ to obtain an agreement from all other configured nodes in the network
    - Each node in the network sends an ADDR_REP reply back to the agent
    - If the agent receives a reply from all the other nodes, it assigns the address to the requester by sending ADDR_ALLOC

Msg.1: NEIGH_REQ (one-hop broadcast)

Msg.2: NEIGH_REP (one-hop broadcast)

Msg.3: REQUESTER_REQ (one-hop broadcast)

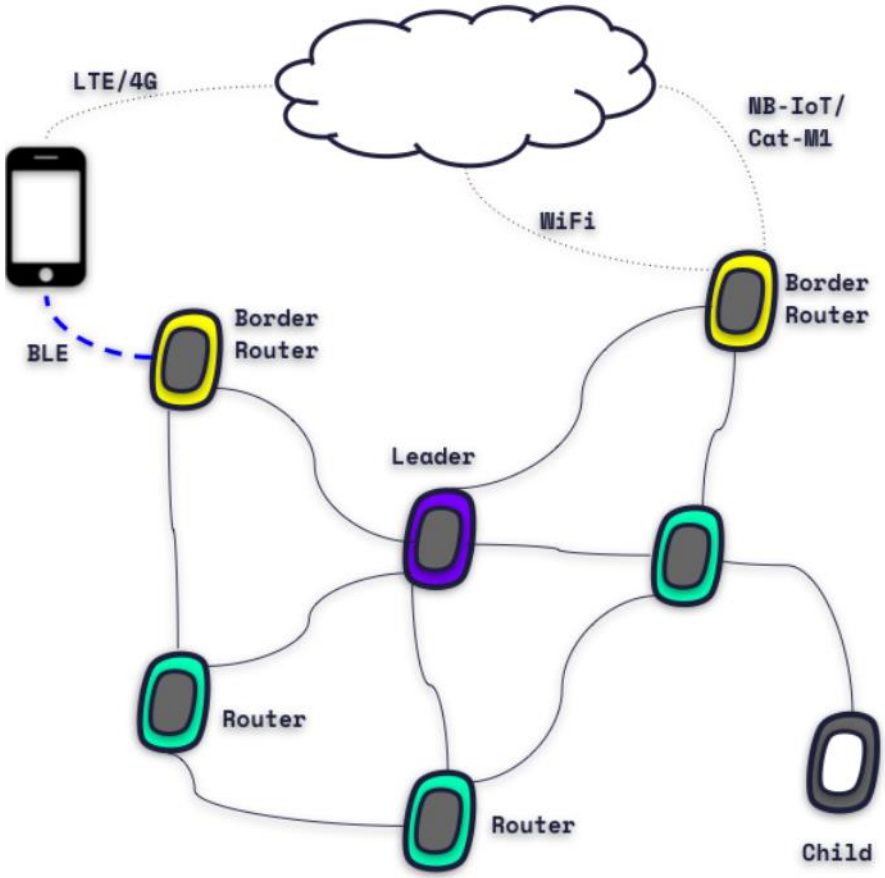
Msg.4: ADDR_REQ (flooding packet at the first time, unicast after the first time)

Msg.5: ADDR_REP (unicast)

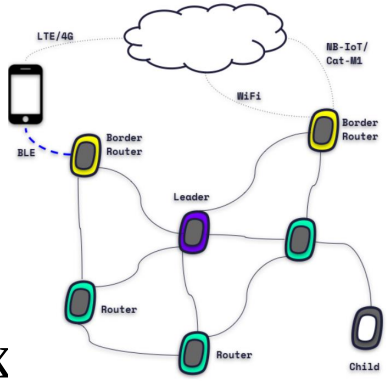
Msg.6: ADDR_ALLOC (one-hop broadcast)

PyMesh Addressing

- Ad-hoc communication network over raw-LoRa
- Multi-gateway (Border Router) Nodes that connect Mesh-internal data with the Cloud
- Each Node uses LBS - Listen Before Talk
- Security on multiple levels
- Any LoRa device (Lopy4/Fipy) can have any of the Pymesh Node Role: Leader, Router, Child or Border Router.



PyMesh Addressing



- Declare the Border Router network address prefix

for example 2001:dead:beef:cafe::/64

- The network address prefix is then sent to the Leader
- All the nodes will be assigned a random IPv6 unicast address with this network prefix (for example 2001:dead:beef:cafe:1234:5678:9ABC:DEF0)
- If a node sends data to an IPv6 which is external (prefix from non-existent networks in Pymesh), then the UDP datagram will be routed to the Border Router
- This UDP packet will have as source the random IPv6 from the border router network address

PyMesh Addressing and Communication

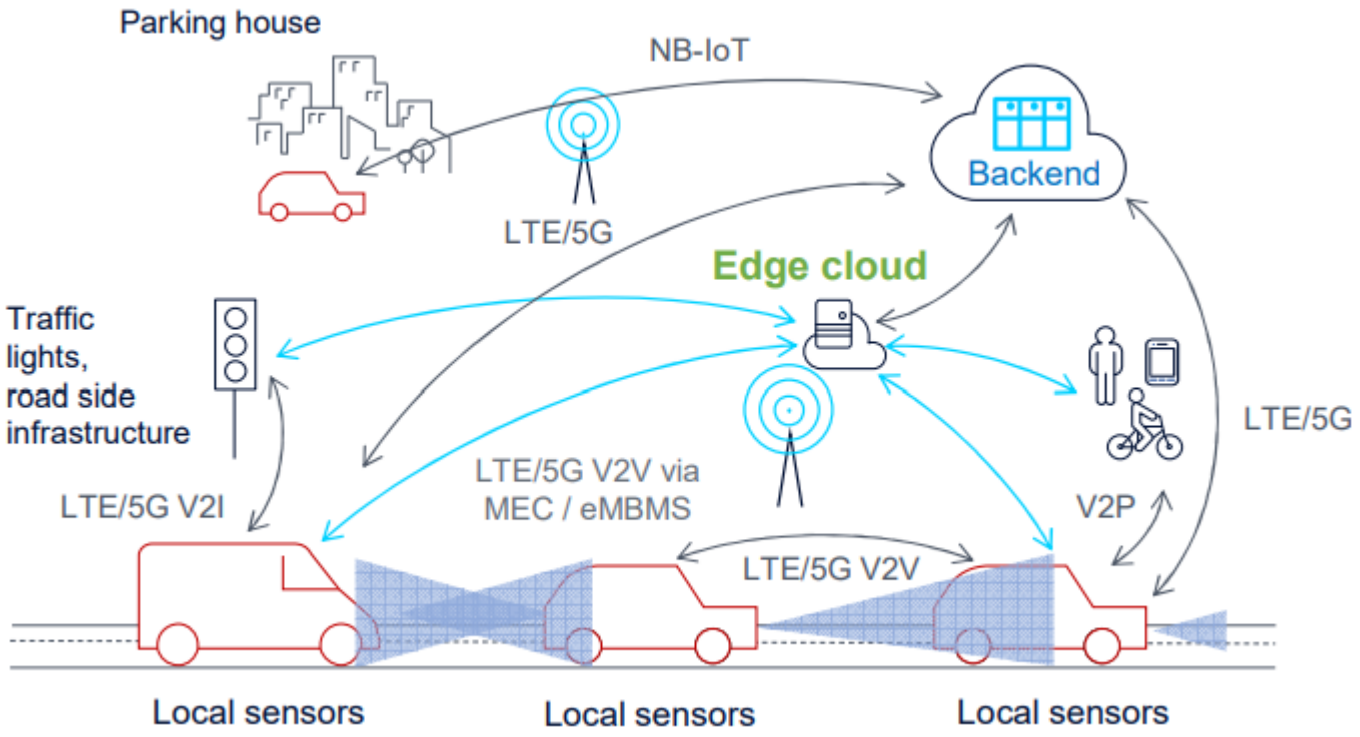
- The Border Router will receive the external UDP datagrams with an appended header, which contains:
  - MAGIC byte: 0xBB
  - IPv6 destination as bytearray (16 bytes)
    - bytearray([source, encoding, errors])
    - Returns the bytearray of the bytes array passed in
  - port destination as 2 bytes (1-65535 values).
  - The IPv6 destination is important, because it means that the Border Router can route (forward) the UDP datagram content to the correct interface (Wifi/BLE/cellular).

Self-organized systems: Data, learning and decisions

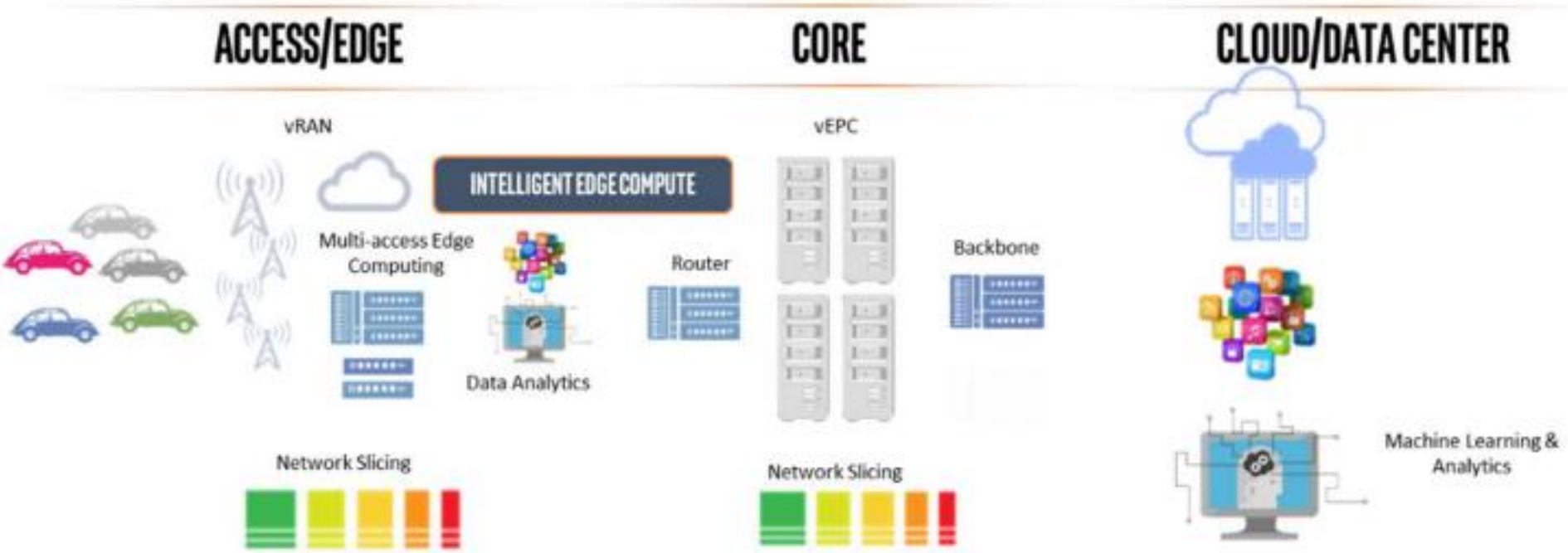
Mestrado em Engenharia de Computadores e Telemática

2022/2023

Use cases and data



Where to process the data?

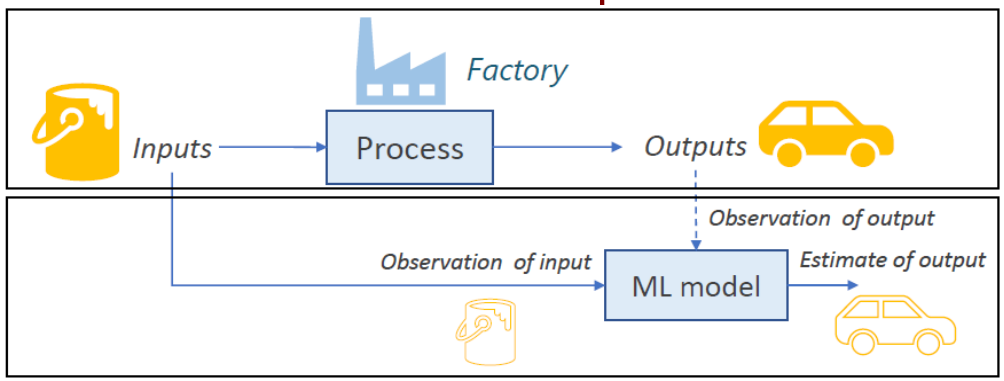


Data Analytics

- Processing data
- Get some decision with the data
- Network decisions with network and services data
  - Give more bandwidth?
  - Assign some special queue to reduce the delay?
- User decisions or element decisions
  - Obstacle in place?
  - Robot kicks the ball to the right?
- Network decisions with users' data
  - Predict handovers with location and velocity
  - Move the CDN content to the users' most near access point
  - Ambulance is on the way with network requirements
- User decisions with network data
  - Chose a path with great connectivity for gaming or video
  - Chose a place for remote augmented and virtual reality

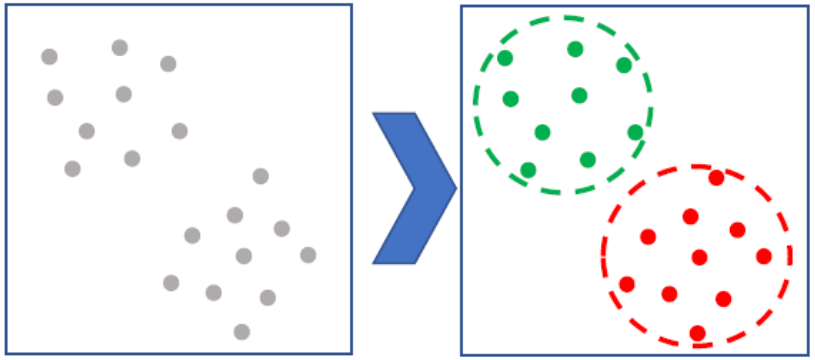
Machine Learning

- A pragmatic definition:
  - Collection of algorithms and statistical models (methods) for machines to carry out automated tasks ***based on the observation of inputs and/or outputs of a process***
- The goal of Machine Learning is to produce an estimation or a classification given a set of input values.
- We often distinguish:
  - ML method: the mechanism to train a model (neural network, support vector machine, etc.)
  - ML model: an instance of the method trained to replicate the behavior of the target process



Types of Learning

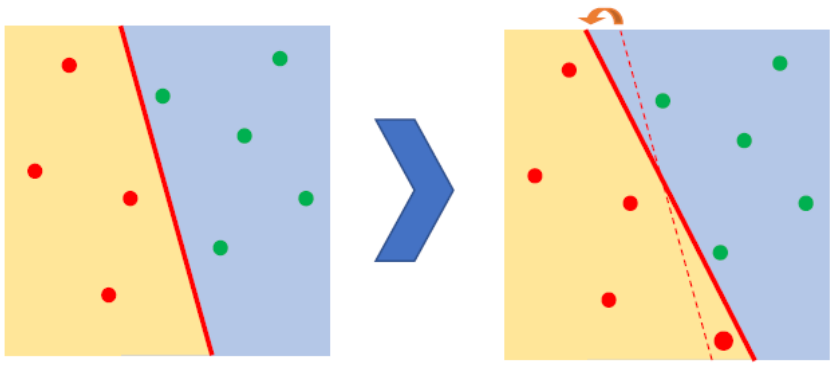
- Supervised: model is trained with a dataset of the target process
  - When training for a classification task, the historical dataset should contain the Ground Truth - the actual class of a given sample
- Unsupervised: classification or regression does not depend on prior knowledge



Unsupervised (classes are created by, e.g., finding clusters of similar data points)

Ground Truth
necessary for training

Historical Dataset		
Feature 1	Feature 2	Class
A	1	X
B	1	X
C	2	Y



Supervised (classification depends on historical inputs)

Supervised Learning

- Training stage

  A dataset with input data and corresponding output

  The input data is pre-processed to identify and/or extract relevant features

  The feature data is input to the ML method, typically one feature set (e.g., mean, median, std. dev.)

  Sometimes a blind set of features is produced, and then only the most relevant are selected (e.g., decision trees)

  For each input set, the method produces an estimate likely to have an error.

  The method compares the estimate with the actual process output (the Ground Truth), and updates the model's internal processes to improve the accuracy of the estimates.

  The process is repeated until performance of the method is within acceptable bounds.

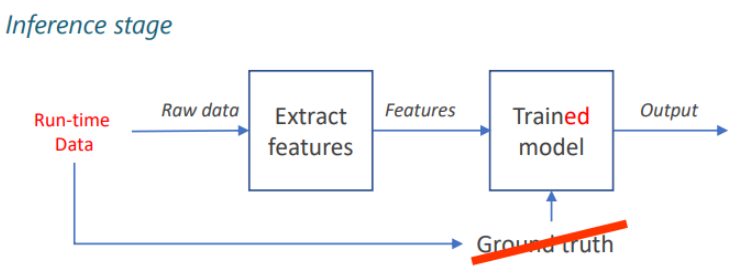
- Inference stage

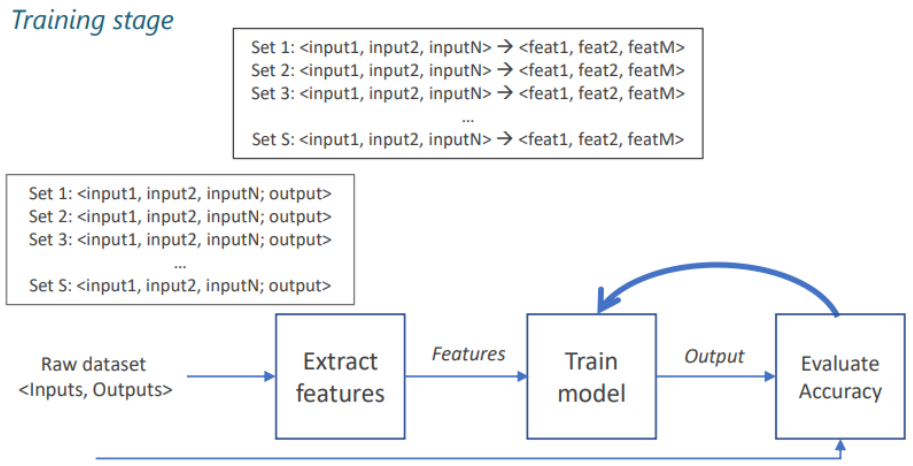
  The trained model is deployed in its target setting.

  Given inputs, it can produce estimates of the process output.

  However, the method no longer has access to the ground truth,

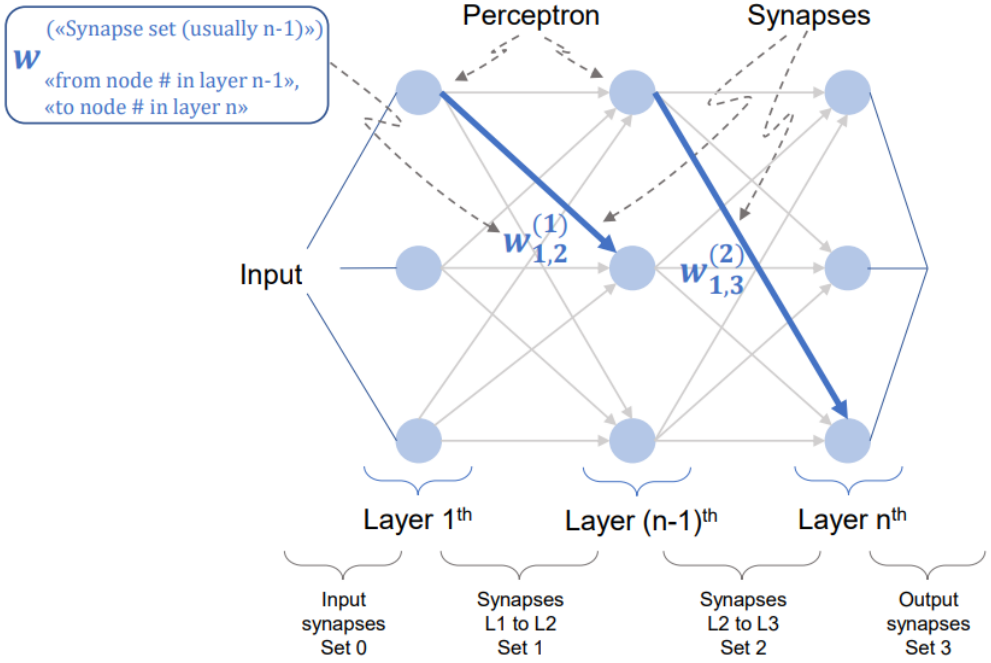
  and it thus enable of further learning.

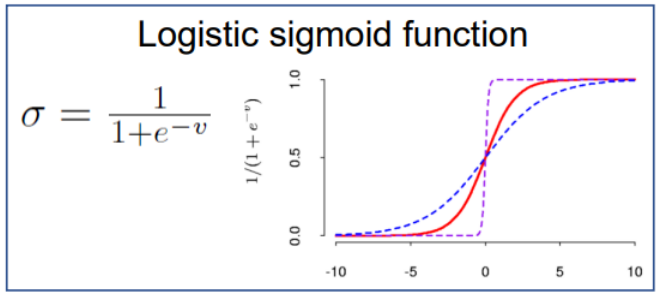




Neural networks

- One of the most successful methods of ML
- Building blocks: Perceptron and Synapse
- Perceptron: typically a function that maps the entire natural range into a bounded interval ([0,1] or [-1,1])
  - Example: Logistic sigmoid function, ReLU function, tanh, softmax, etc
- Synapses: connections from perceptrons of layer (n-1)th to perceptrons of layer nth, each applying a weight to the transmitted value
- Training Neural Networks is mostly about finding the weights of those synapses

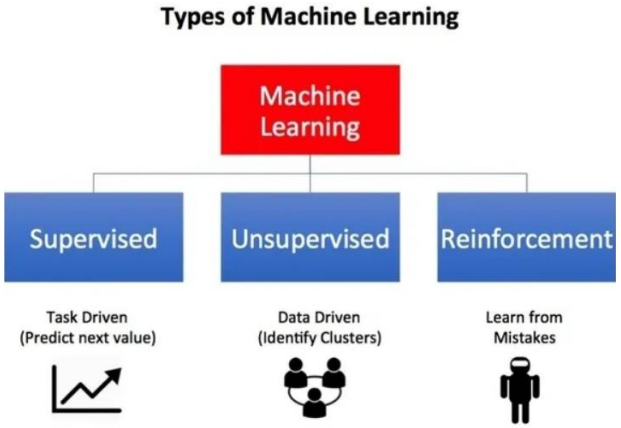




Reinforcement Learning

- Type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences

1. **Environment —** Physical world in which the agent operates
2. **State —** Current situation of the agent
3. **Reward —** Feedback from the environment
4. **Policy —** Method to map agent's state to actions
5. **Value —** Future reward that an agent would receive by taking an action in a particular state

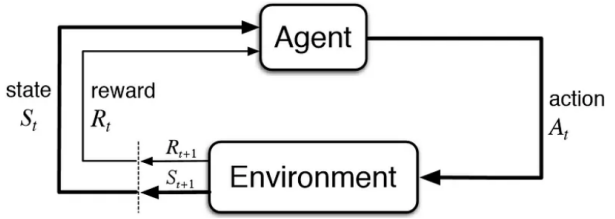


$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

- **Q-learning:** updates Q values which denotes value of performing action *a* in state *s*. The following value update rule is the core of the Q-learning algorithm.

Reward example: best path with resources: path bandwidth / path length

Learning rate and discount factor:]0 1[



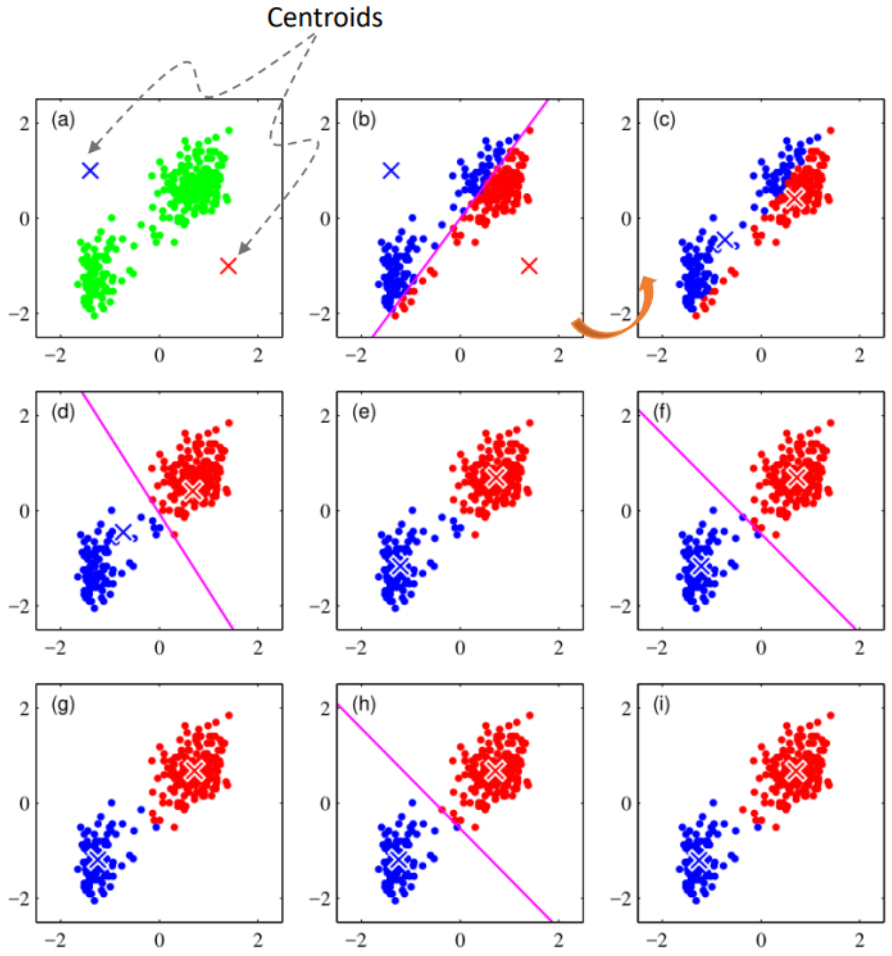
$$reward = \frac{\frac{100.0}{|P|} \times \sum_{l}^{l \in P} R_l}{E_{pl} \times K_{spl}}$$

Computer games (pacman example), robot automation (RL is used to enable the robot to create an efficient adaptive control system for itself which learns from its own experience and behavior**), …**

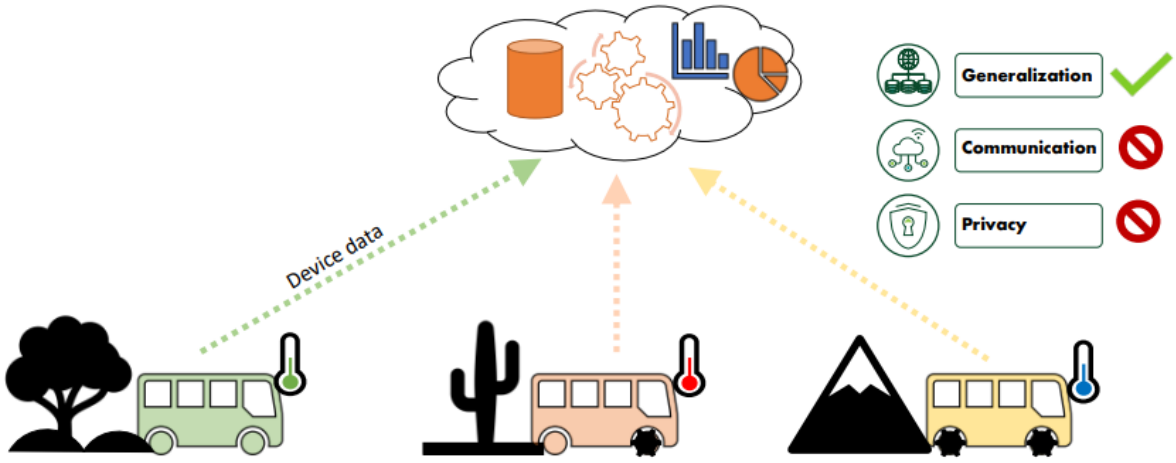
Unsupervised learning: K-means

- Centroid: non-data point that indicates center of cluster as identified by K-means

- Operation:
  1. Deploy N centroids randomly (N proportional to number of expected classes)
  2. Assign randomly data points to classes
  3. Repeat iteratively
     1. Compute center of gravity of each class;
     2. Centroid is repositioned in that center of gravity
     3. Update boundary
  4. Stop when updates become negligible



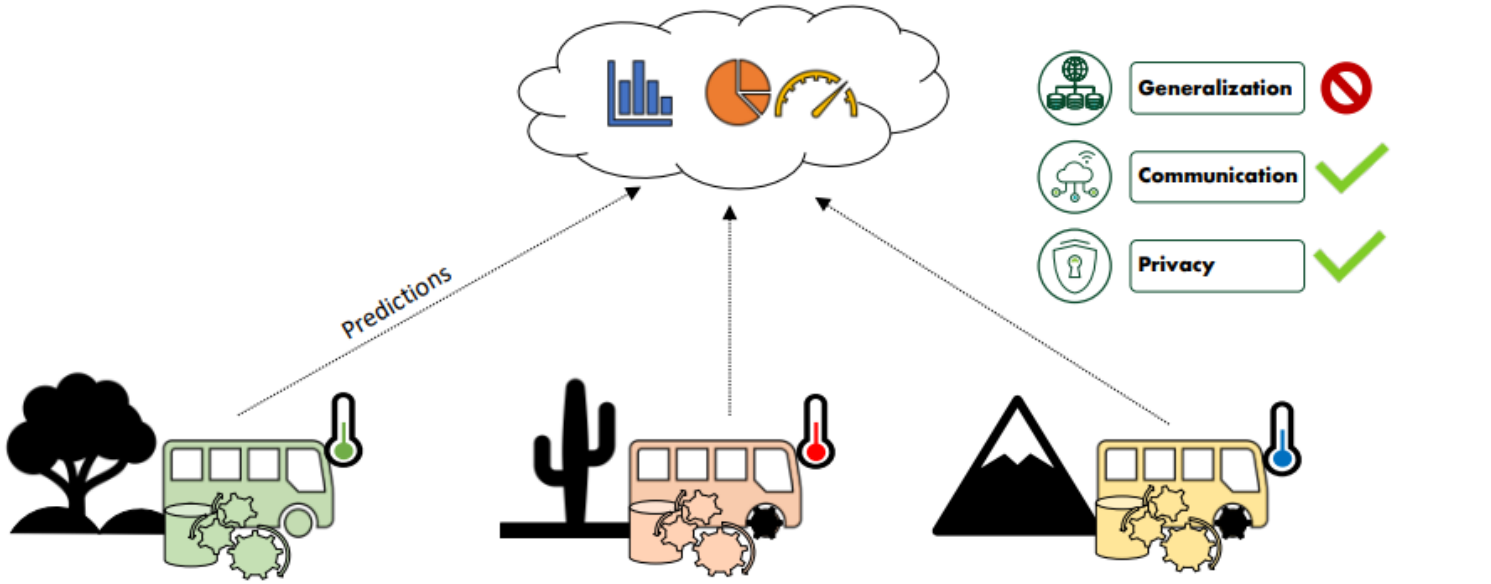
Learning: centralized



Traditional centralized learning – ML runs in the cloud, gathering info from all connected devices and sending back a model.

- The model can generalize based on data from a group of devices and thus instantly work with other compatible devices
- Data can explain all variations in the devices and their environment
- Connectivity - data must be transmitted over a stable connection
- Bandwidth – e.g. a new electrical substation could generate 5 GB/s
- Latency - real-time applications, e.g. automation, requires very low latency
- Privacy - sensitive operational data must remain on site

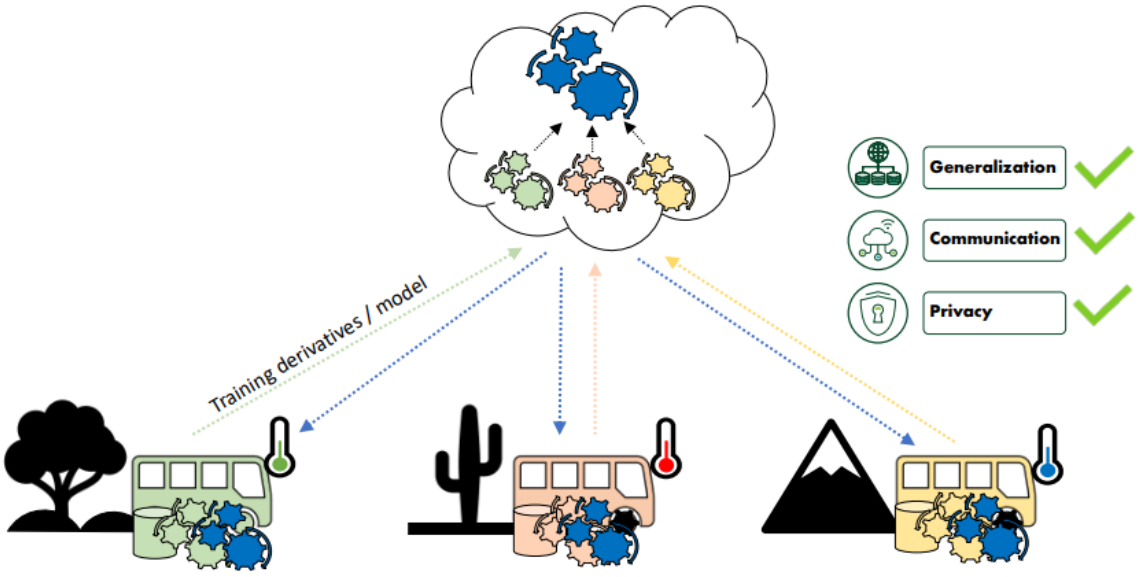
Learning: de-centralized



Edge/decentralized learning: ML continuously onboard each device at the edge of the network.

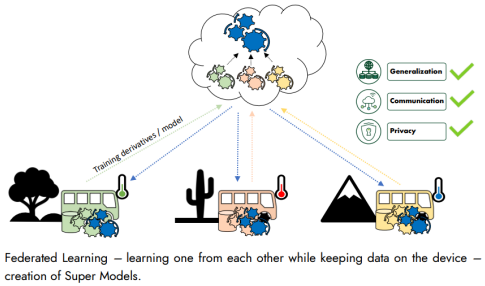
- ML that runs on site, onboard each connected device, by continuously training the ML model on streaming data, the devices learn an individual model for their environment.
- Each model only needs to be able to explain what is normal for itself and not how it varies compared to all other devices.
- Models adapt to changes over time, learning is not constrained by the internet connection and that no confidential information needs to be transferred to the cloud.
- It is not possible to get an overall view and learning

Learning: federated



Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

- ML technique to train algorithms across decentralized edge devices while holding data samples locally
- Google started as the main player
- Aim to train ML models on billions of mobile phones while respecting the privacy of the users
  - Only send fractions of training results, i.e. training derivatives, to the cloud
  - Never store anything on the device



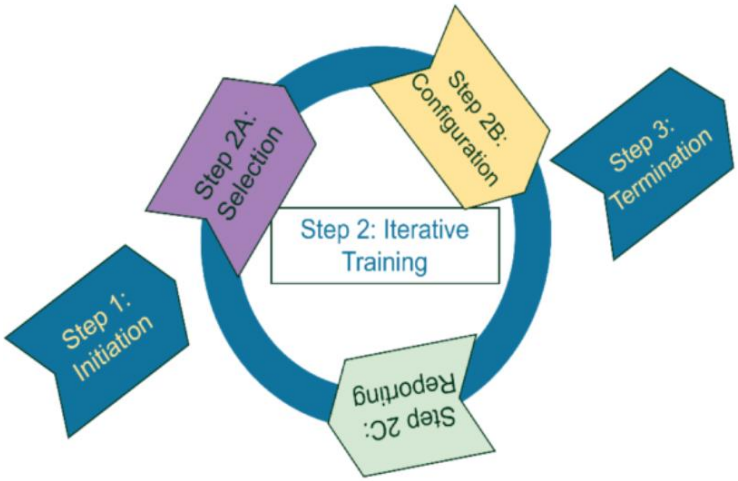
Federated Learning – learning one from each other while keeping data on the device – creation of Super Models.

Learning: federated

- When collected in the cloud, the partial training results can be assembled to a new supermodel that, in the next step, can be sent back to the devices
  - **Goolgle open source framework TensorFlow Federated**
- Model inspection – evaluation of device behavior through its model
- Model comparison – comparing models in the cloud to find outliers, super models
- Robust learning - learning can continue even if connection to the cloud is lost
- Tailored initialization – new devices can start with a model from a similar device, instead of a general super model

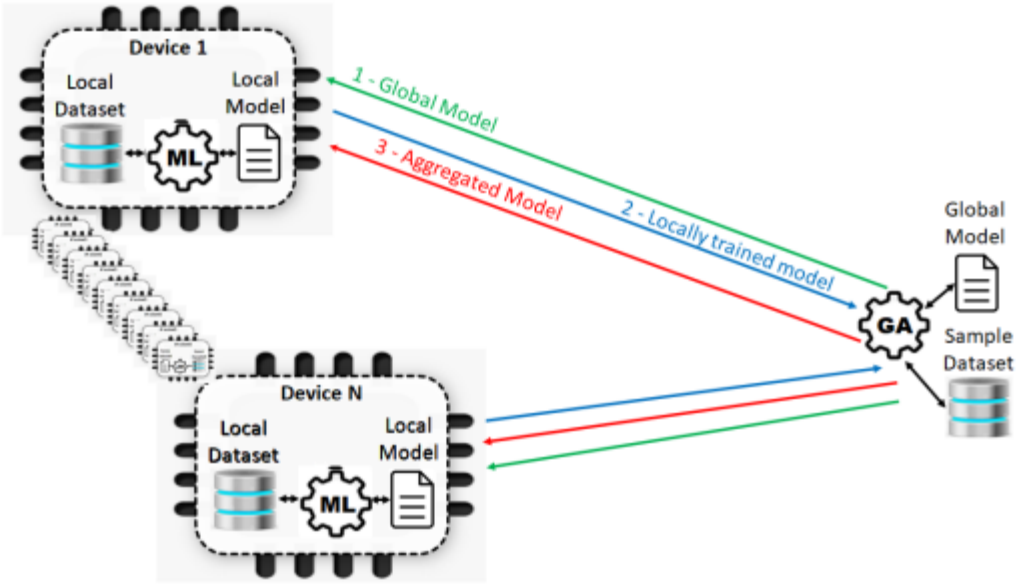
Federated learning: iterative

- FL employs an iterative method containing multiple client-server exchanges: federated learning round
  - Diffuse the current/updated global model state to the contributing nodes (participants)
  - Train the local models on those nodes to yield certain potential model updates from the nodes
  - Process and aggregate the updates from local nodes into an aggregated global update so that the central model can be updated accordingly
- FL server is used for this processing and aggregation of local updates to global updates
  - Local training is performed by local nodes with respect to the commands of FL server



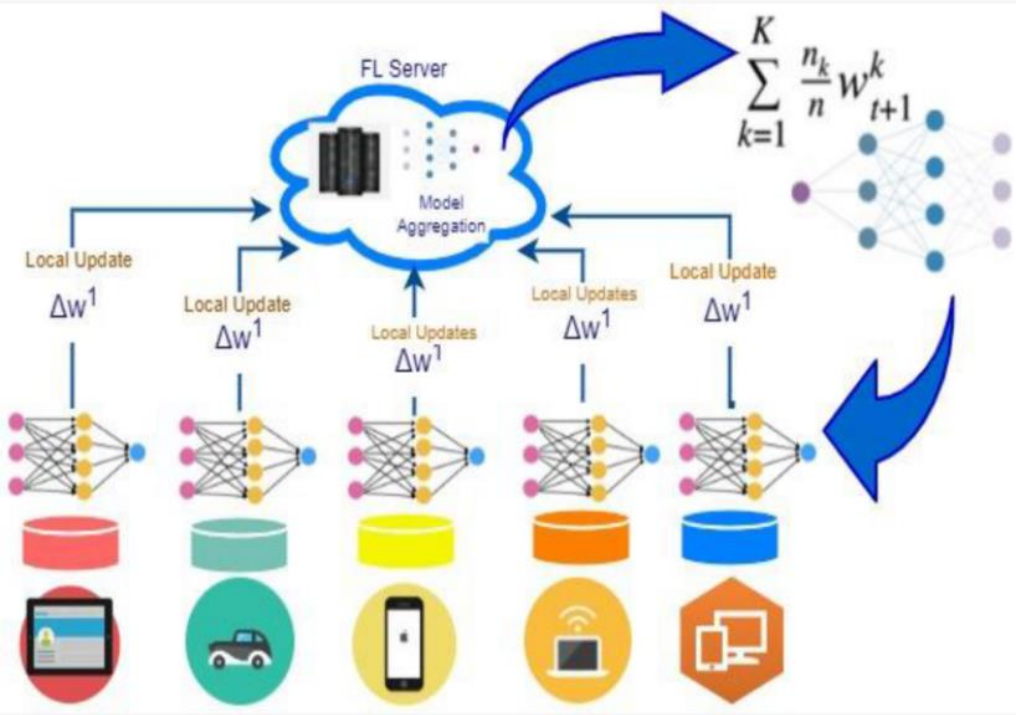
Learning: federated

- The FL approach allows for mass processing of data in a distributed way
- It can follow a client-server architecture
  - Server sends the model to be created to the clients (1 – green lines)
  - Results of the local computation are sent to the server, which aggregates them into the global model (2 – blue lines)
  - Returns the new aggregated model to the clients (3 – red lines)
  - This iteration, named federated learning round (FLR), occurs until some stopping criterion is reached, such as model convergence or maximum number of iterations reached.
  - Edge devices only send information from their local models (parameters, hyperparameters (before training), weights, etc).



Learning: federated

- Federated learning distributes deep learning by eliminating the necessity of pooling the data into a single place
- In FL, the model is trained at different sites in numerous iterations



Learning: model aggregation

- Effective aggregation of distributed models across devices is essential for creating a generalized global model.
- Its efficiency affects precision, convergence time, number of rounds and network overhead.
- Federated stochastic gradient descent (SGD): uses a single instance of the dataset to perform the local training on the client per round of communication. SGD requires a substantial number of training rounds to produce reliable models. This algorithm is the baseline of federated learning.
- FedAvg algorithm starts from the SGD, but each client locally performs a train using the local data at the current model with multiple steps of SGD before sending the models back to the server for aggregation

FedAvg reduces the communication overhead required to upload and download the FL model

It requires clients to perform more total computation during training.

Local epoch: one complete pass of the training dataset through the algorithm

Algorithm 1 Algorithm FedAvg. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and n is the learning rate [3].

function SERVER-SIDE:
  initialize w0
  for each round $t = 1,2,...$ **do**
    $m \leftarrow \max(C \cdot k,1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1} \leftarrow \text{ClientUpdate}(k, wt)$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  end for
end function

function CLIENT-SIDE:
  ClientUpdate(k,w): // Run on client k
  $B \leftarrow$ (split P_k into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in B$ **do**
      $w \leftarrow w - \eta \nabla l\,(w; b)$
    end for
  end for
  return w to server
end function

Learning: model aggregation

- Fault Tolerant Federated Average: ability of a computing system to continue working in the event of a failure
  - Can tolerate some nodes being offline during secure aggregation
- Q-Federated Average: re-weight the objective in order to achieve fairness in the global model
  - Gives higher weights to devices with poor performance
  - The network's accuracy distribution becomes more uniform
  - Fk(·) to the power of (q+1), q is a parameter that tunes the amount of fairness to impose.

$$\min_w f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)$$

- Federated Optimization: uses a client optimizer during the multiple training epochs and a server optimizer during model aggregation
  - ADAGRAD, ADAM, and Yogi

Algorithm 1 FEDOPT

Input: x_0, CLIENTOPT, SERVEROPT
for $t = 0, \cdots, T-1$ **do**
  Sample a subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = \text{CLIENTOPT}(x_{i,k}^t, g_{i,k}^t, \eta_l, t)$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $x_{t+1} = \text{SERVEROPT}(x_t, -\Delta_t, \eta, t)$

Algorithm 2 FEDADAGRAD, FEDYOGI, and FEDADAM

Initialization: $x_0, v_{-1} \geq \tau^2$, decay parameters $\beta_1, \beta_2 \in [0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $m_t = \beta_1 m_{t-1} + (1-\beta_1)\Delta_t$
  $v_t = v_{t-1} + \Delta_t^2$ **(FEDADAGRAD)**
  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{m_t}{\sqrt{v_t} + \tau}$

TensorFlow Federated

- Open source framework for experimenting with machine learning and other computations on decentralized data.
- Locally simulating decentralized computations into the hands of all TensorFlow users.
  - ML model architecture of our choice
  - Train it locally across data by all users
- Version of the NIST dataset that has been processed by the Leaf project to separate the digits written by each volunteer.

```python
# Load simulation data.
source, _ = tff.simulation.datasets.emnist.load_data()
def client_data(n):
  dataset = source.create_tf_dataset_for_client(source.client_ids[n])
  return mnist.keras_dataset_from_emnist(dataset).repeat(10).batch(20)

# Wrap a Keras model for use with TFF.
def model_fn():
  return tff.learning.from_compiled_keras_model(
      mnist.create_simple_keras_model(), sample_batch)

# Simulate a few rounds of training with the selected client devices.
trainer = tff.learning.build_federated_averaging_process(model_fn)
state = trainer.initialize()
for _ in range(5):
  state, metrics = trainer.next(state, train_data)
  print (metrics.loss)
```

federated_learning_for_image_classification.py hosted with ❤ by GitHub

TensorFlow Federated

- Training an ML model with federated learning is one example of a federated computation
- Evaluating it over decentralized data is another
  - Array of sensors capturing temperature readings
  - Compute the average temperature across these sensors
- Each client computes its local contribution
- Centralized coordinator aggregates all the contributions.

```
1  @tff.federated_computation(READINGS_TYPE)
2  def get_average_temperature(sensor_readings):
3    return tff.federated_average(sensor_readings)
```

get_average_temperature.py hosted with ❤ by GitHub

Flower: A Friendly Federated Learning Framework (https://flower.dev/)

- Open source framework for experimenting with machine learning and other computations on decentralized data.
- Able to use in containers in a federated framework, FedFramework

List containers:

- **http://10.0.22.37:8000/containers/list**

Containers:
/containers/list
/containers/create/server
/containers/create/client
/containers/start
/containers/stop
/containers/remove

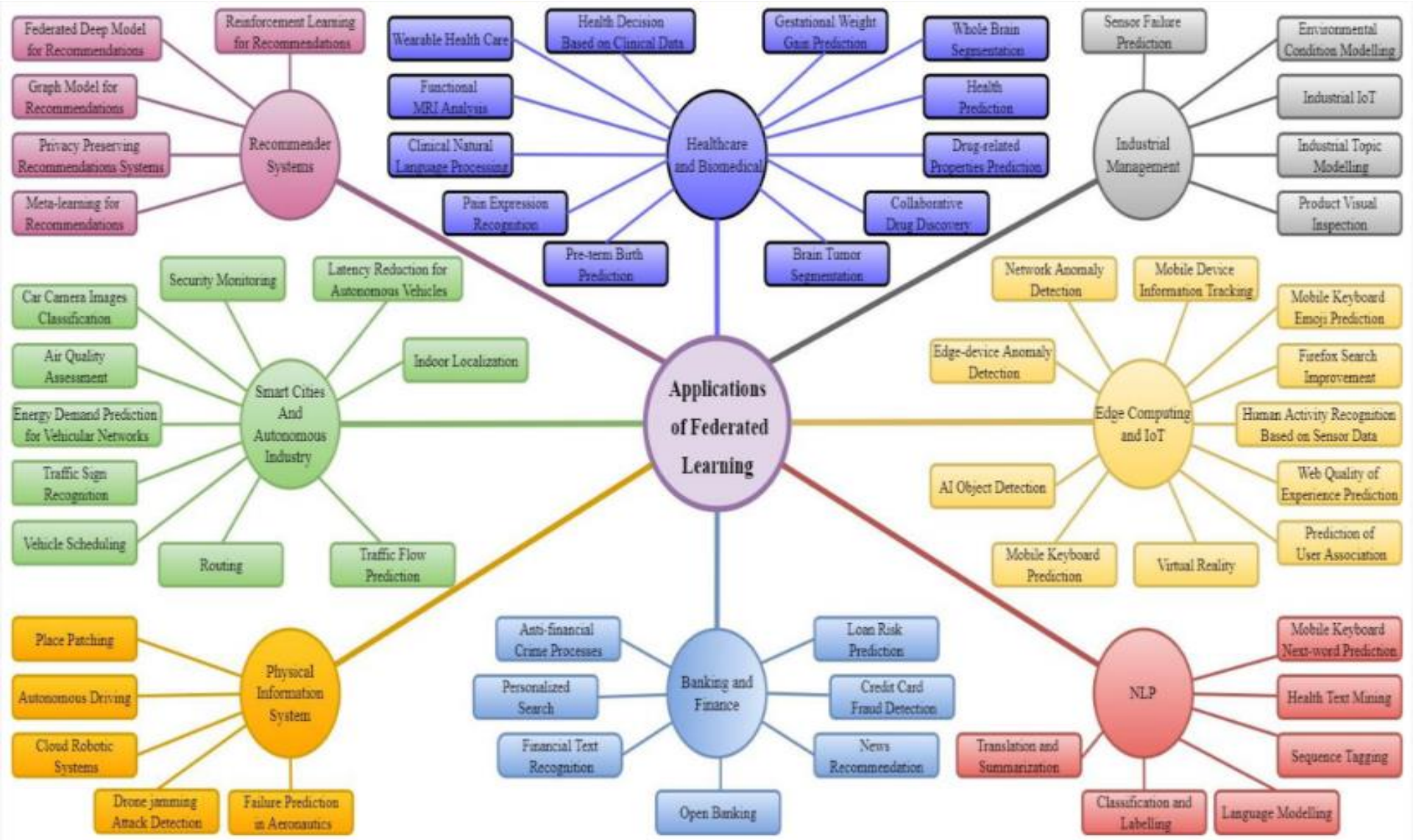
Server Creation:

- **http://10.0.22.37:8000/containers/create/server?img_server=fed-server&port=5010&id=10&clients=4&algorithm=FedAvg&model=cnn&rounds=10&epochs=5&predict=true**

Rapid deployment of a testbed and run tests:

- **http://10.0.22.37:8000/run?img_server=fed-server&img_client=fed-client&model=cnn&clients=4&rounds=10&epochs=5&predict=true&predict_client=true**

Applications for Federated

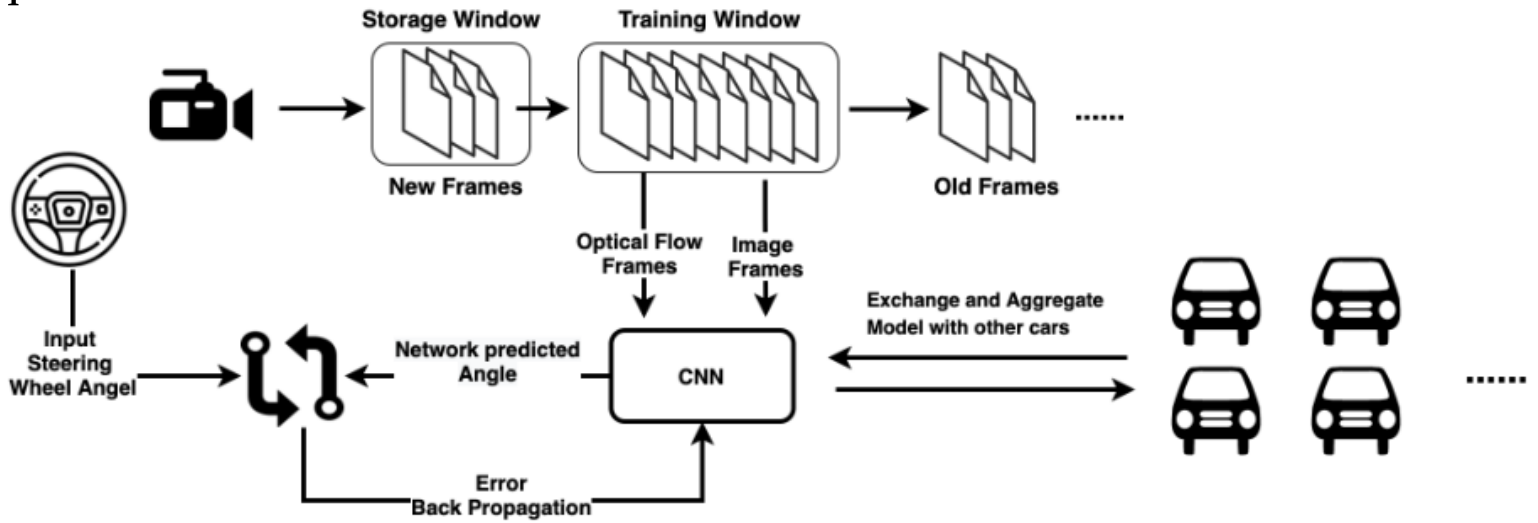


Applications for Federated

Domain	Applications
Edge computing	FL is implemented in edge systems using the MEC (mobile edge computing) and DRL (deep reinforcement learning) frameworks for anomaly and intrusion detection.
Recommender systems	To learn the matrix, federated collaborative filter methods are built utilizing a stochastic gradient approach and secured matrix factorization using federated SGD.
NLP	FL is applied in next-word prediction in mobile keyboards by adopting the FedAvg algorithm to learn CIFG [93].
IoT	FL could be one way to handle data privacy concerns while still providing a reliable learning model
Mobile service	The predicting services are based on the training data coming from edge devices of the users, such as mobile devices.
Biomedical	The volume of biomedical data is continually increasing. However, due to privacy and regulatory considerations, the capacity to evaluate these data is limited. By collectively building a global model for the prediction of brain age, the FL paradigm in the neuroimaging domain works effectively.
Healthcare	Owkin [31] and Intel [32] are researching how FL could be leveraged to protect patients‘ data privacy while also using the data for better diagnosis.
Autonomous industry	Another important reason to use FL is that it can potentially minimize latency. Federated learning may enable autonomous vehicles to behave more quickly and correctly, minimizing accidents and increasing safety. Furthermore, it can be used to predict traffic flow.
Banking and finance	The FL is applied in open banking and in finance for anti-financial crime processes, loan risk prediction, and the detection of financial crimes.

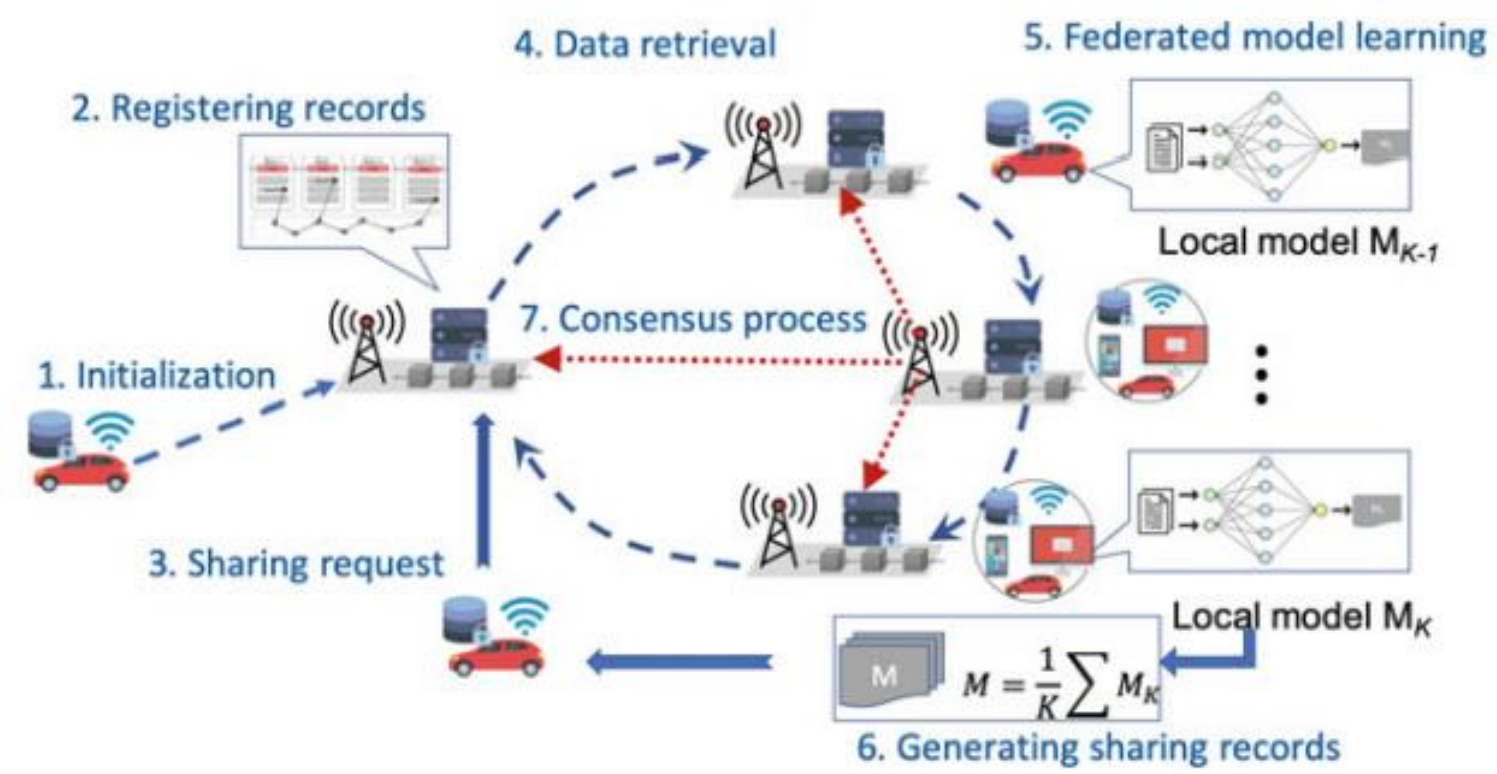
Federated learning in self-driving

- Edge vehicles compute the model locally; after completing each local training epoch, they retrieve the global model version and compare it to their local version.
- In order to form a global awareness of all local models, the central server performs aggregation based on the ratio determined by the global and local model versions.
- The aggregation server returns the aggregated result to the edge vehicles that request the most recent model.



Edge-based Federated

- MEC-empowered model sharing
  - **Edge intelligence to wireless edge networks and enhances the connected intelligence among end devices in 6G networks.**



- https://www.pdl.cmu.edu/SDI/2019/slides/2019-09-05Federated%20Learning.pdf
- https://wires.onlinelibrary.wiley.com/doi/epdf/10.1002/widm.1443
- https://medium.com/tensorflow/introducing-tensorflow-federated-a4147aa20041