

# Decentralized Systems Engineering

CS-438 – Fall 2024

DEDIS

Pierluca Borsò-Tan and Bryan Ford

EPFL

Credits: P. Tennage, C. Basescu, et al.

# So far...

- Decentralized communication & search
- Focusing on (mostly) unstructured networks

Characteristics:

- (Nearly) stateless
- Simple to engineer
- Expressive search
- Optimizations require (true) random sampling (hard)
- ☹ inefficient,  $O(\sqrt{n})$  search at best

Can we aim for  $O(\log n)$  efficiency ?

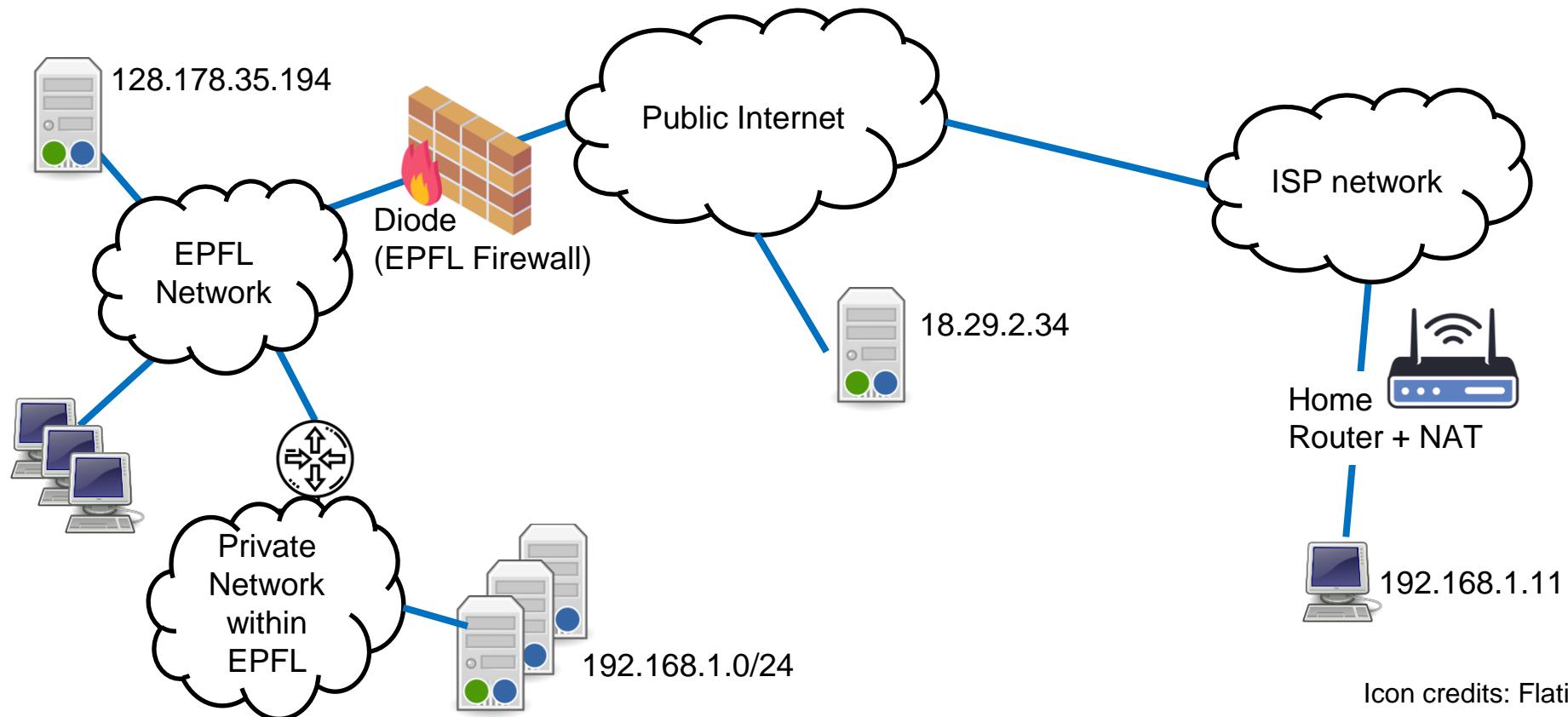
→ later today

# Ad-hoc Routing Protocols

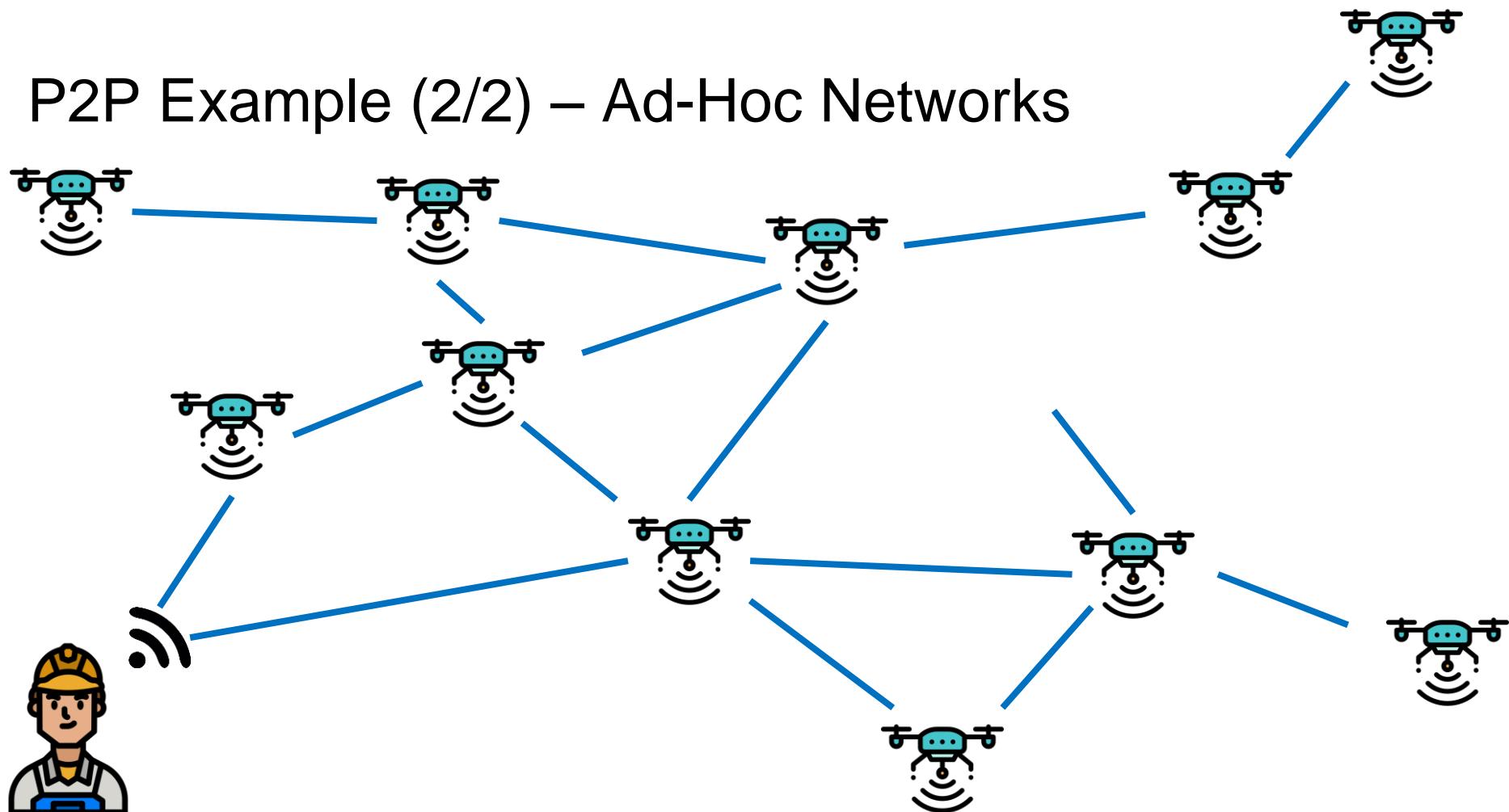
Finding your way through ad-hoc networks

(Homework 1)

# P2P Example (1/2) – Your Home & EPFL Networks



## P2P Example (2/2) – Ad-Hoc Networks



# P2P Examples – Quick analysis

- Peers may not be directly accessible
- Peers may join or leave the network at arbitrary times
- We need to route packets through the system

Some differences:

- Churn
- Node mobility / network reconfiguration
- Protocols / Physical Layer / etc.
- Bandwidth

# Naive routing – don't do this at home

Nodes advertise a distance to other nodes

$$B \rightarrow D = 1$$

$$A \rightarrow D = 2 \text{ (through B or C)}$$

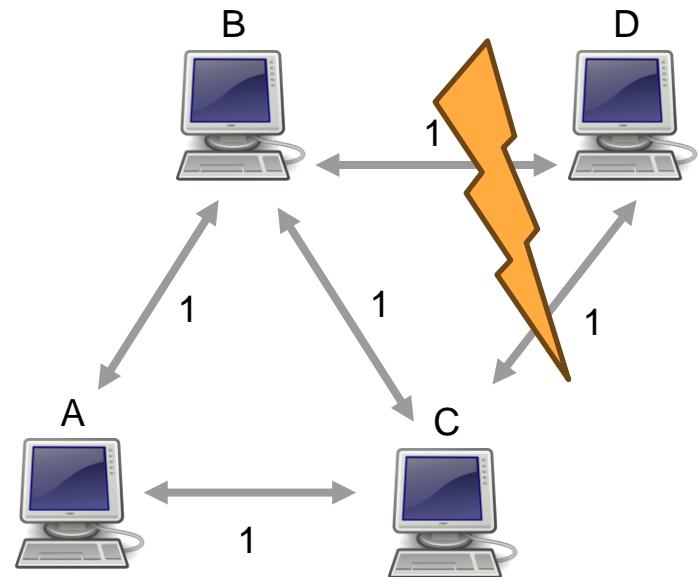
$$C \rightarrow D = 1$$

On link failure, B updates:

$$B \rightarrow D = 2 \text{ (through C)}$$

Then C updates:

$$C \rightarrow D = 3 \text{ (through B)}$$



# Reaching arbitrary peers in a network : AODV

Ad-hoc On-demand Distance Vector

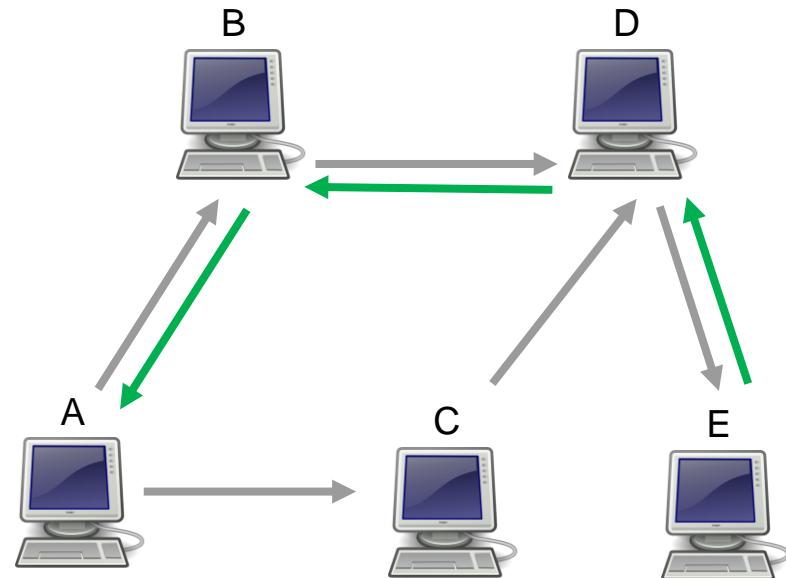
Key idea: Flooding search for a node (e.g. E)

Nodes remember where the search came from  
... And build a return path

$A \rightarrow B \rightarrow D \rightarrow E$

$A \leftarrow B \leftarrow D \leftarrow E$

Reactive (on-demand) routing, cached  
Used in the ZigBee wireless protocol



# Reaching arbitrary peers in a network : DSDV

Destination-Sequenced Distance Vector

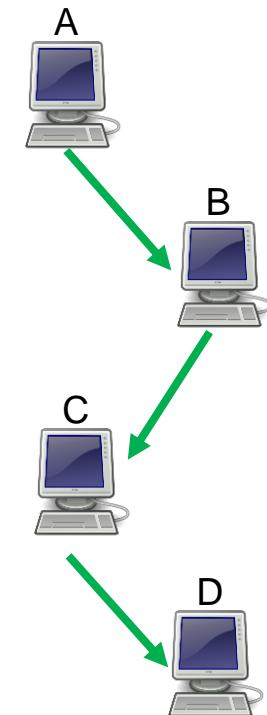
Key idea:

- Store next hop for any destination (  $O(n)$  )
- Version (“sequence”) routing table entries

Each node periodically broadcasts its existence:

Flood the network, with increasing sequence numbers

$O(n^2)$  traffic, superseded by newer protocols,  
versioning idea lives on !



# Quality factors in ad-hoc routing

- Speed of convergence
- Loop-free
- Traffic at rest (maintenance)
- Traffic during updates
- Robustness to churn & movement

# Compact Routing & Structured Search

$O(\log n)$ , here we come !

# General Approach

- Build a structured *overlay network*
- Enables significant efficiency gains

We'll pay a price:

- More engineering effort
- Nodes will need local state
- Constant fight against churn
- Loss of generality

# Distributed Hash Table

Local hash tables need:

- “Good” hash function
- Random-access memory
- Not too full

Distributed hash tables considerations:

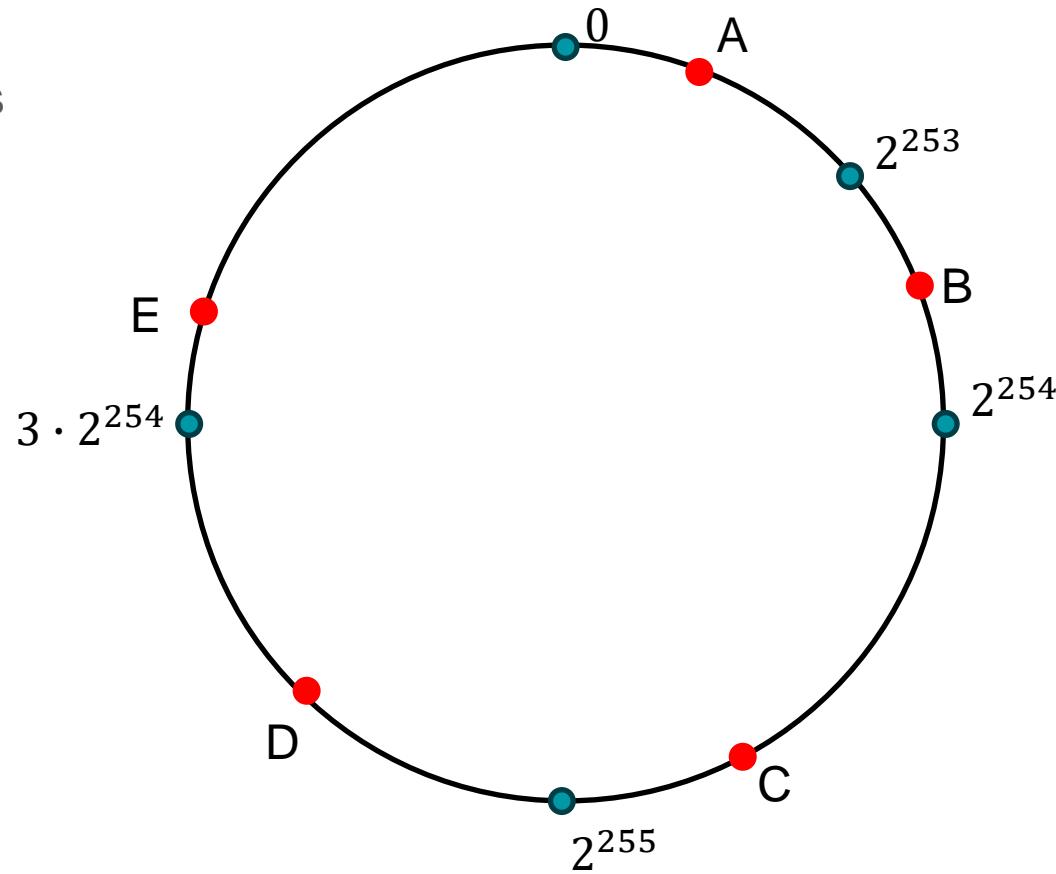
- What do we need from the hash function ?
  - Avoid collisions, not time-sensitive
  - Cryptographic hash, well distributed
- What are we missing ?
  - RAM

# Chord DHT

- Hash into a collection of RAMs
  - Circular hash ID space  
(e.g., SHA-256)
  - Each node has a  
pseudo-random hash ID

## What should go into that ID ?

# Public Key !



# Chord DHT

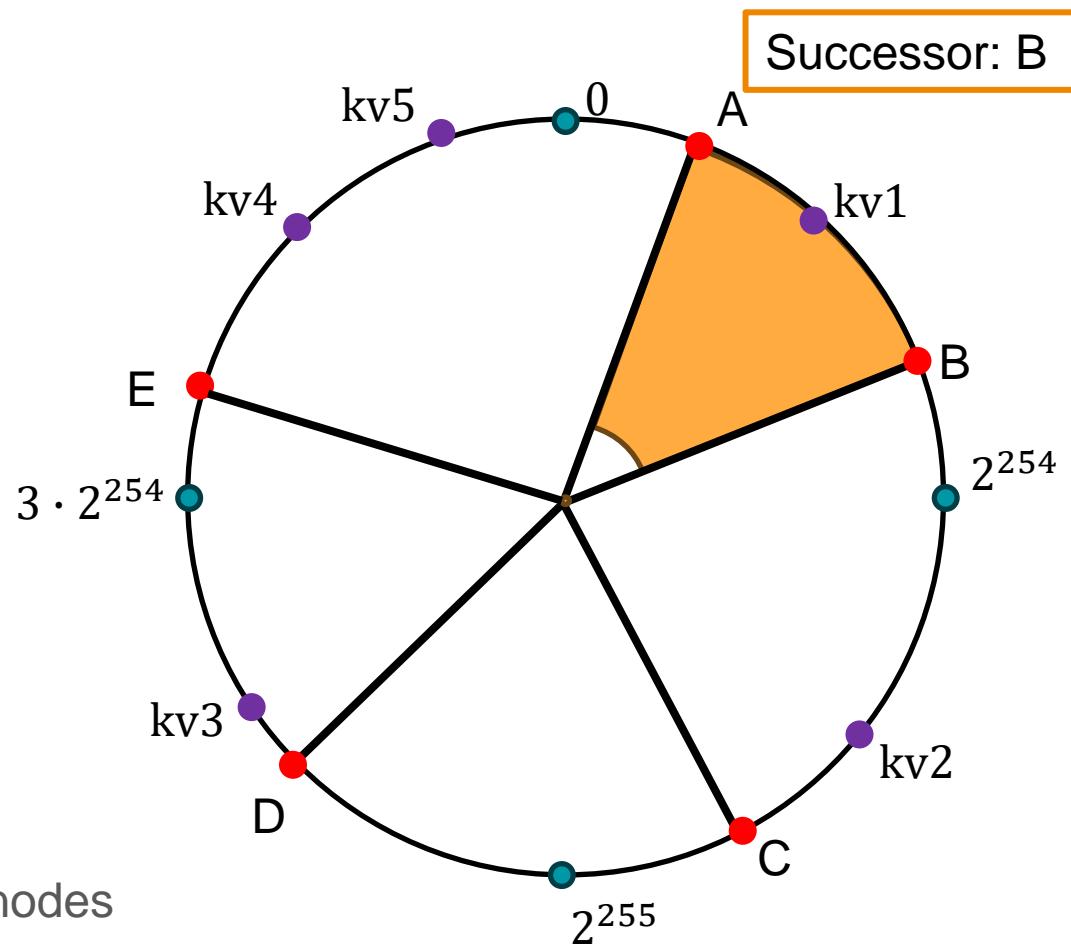
How do we approximate RAM ?

- Divide the space up !
- Each node owns the space to its successor

API:

- PUT(key, value)
- GET(key) → value / error

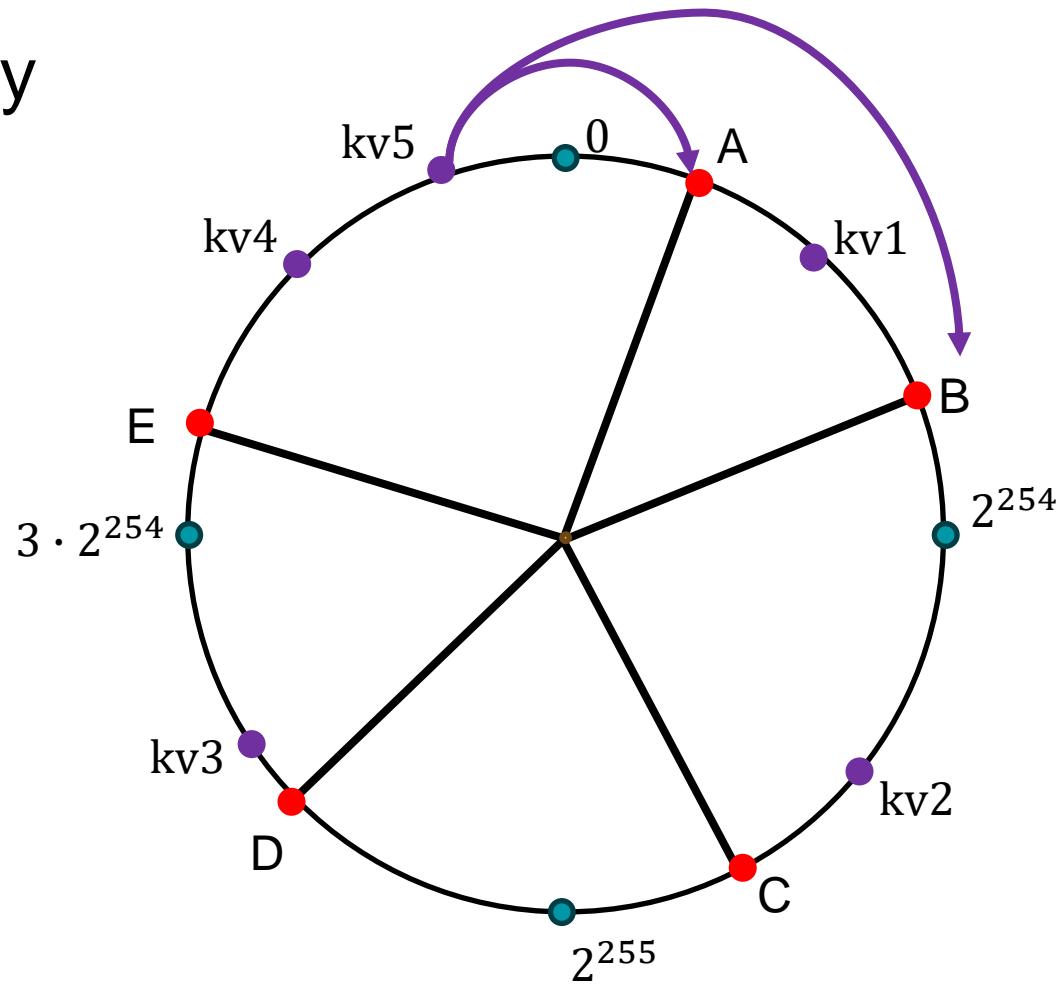
Keys use same hash function as nodes



# Chord DHT – Reliability

How do we prevent data loss ?

- Redundancy – factor  $r$
- Copies are stored by “owner” node +  $(r - 1)$  successors



# Chord DHT – Load

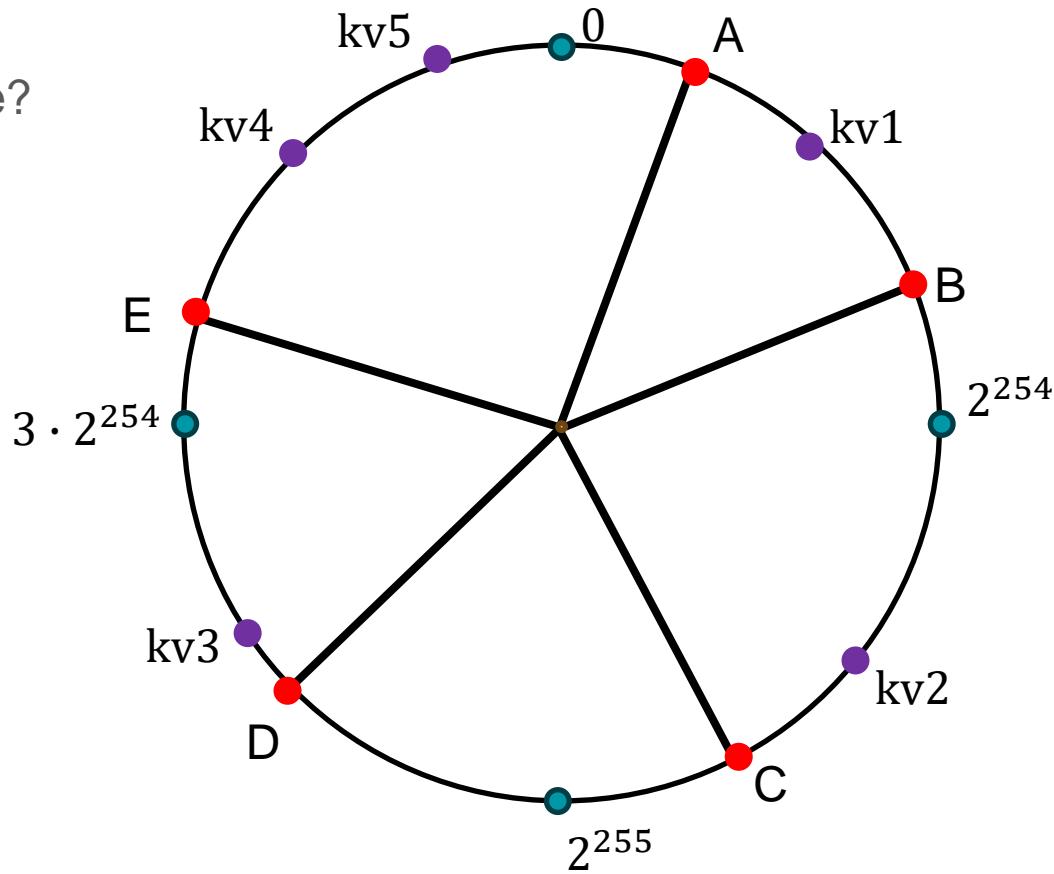
What is the expected load per node?

$n$  – # of nodes

$m$  – # of key-value pair

$r$  – redundancy factor

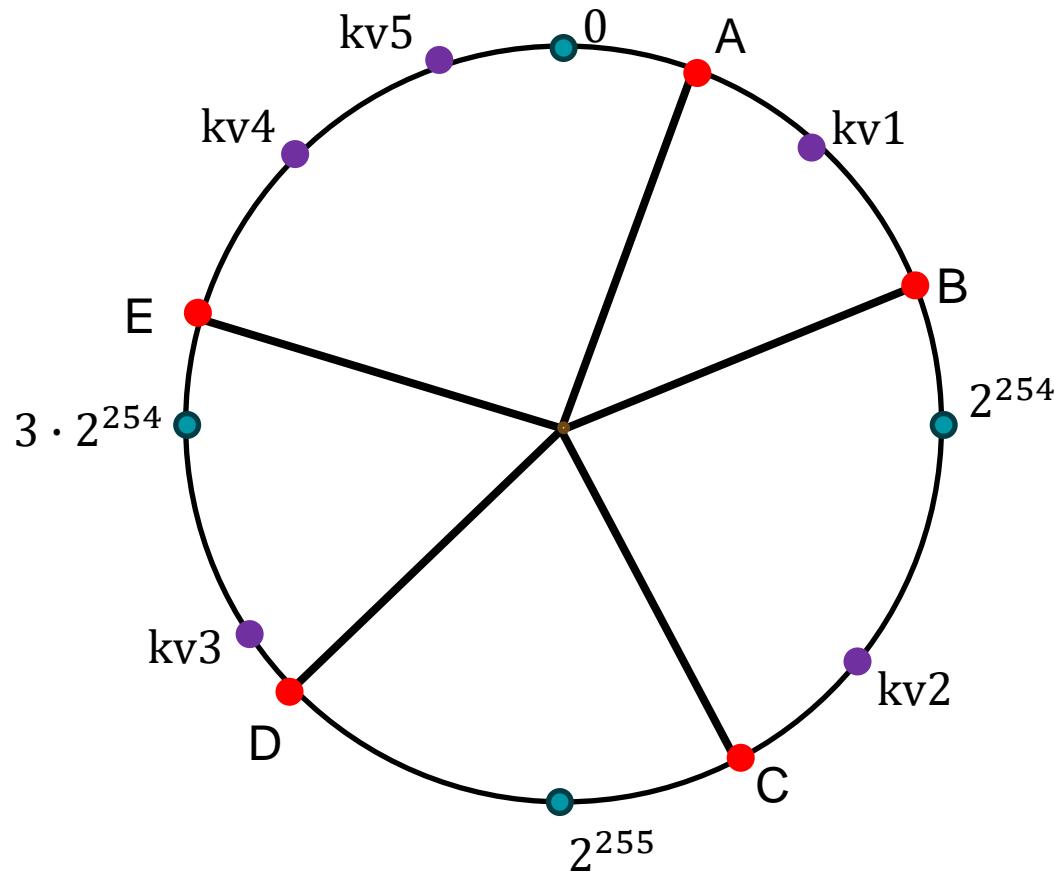
$$\text{Load} \sim \frac{r \cdot m}{n}$$



# Chord DHT – Performance

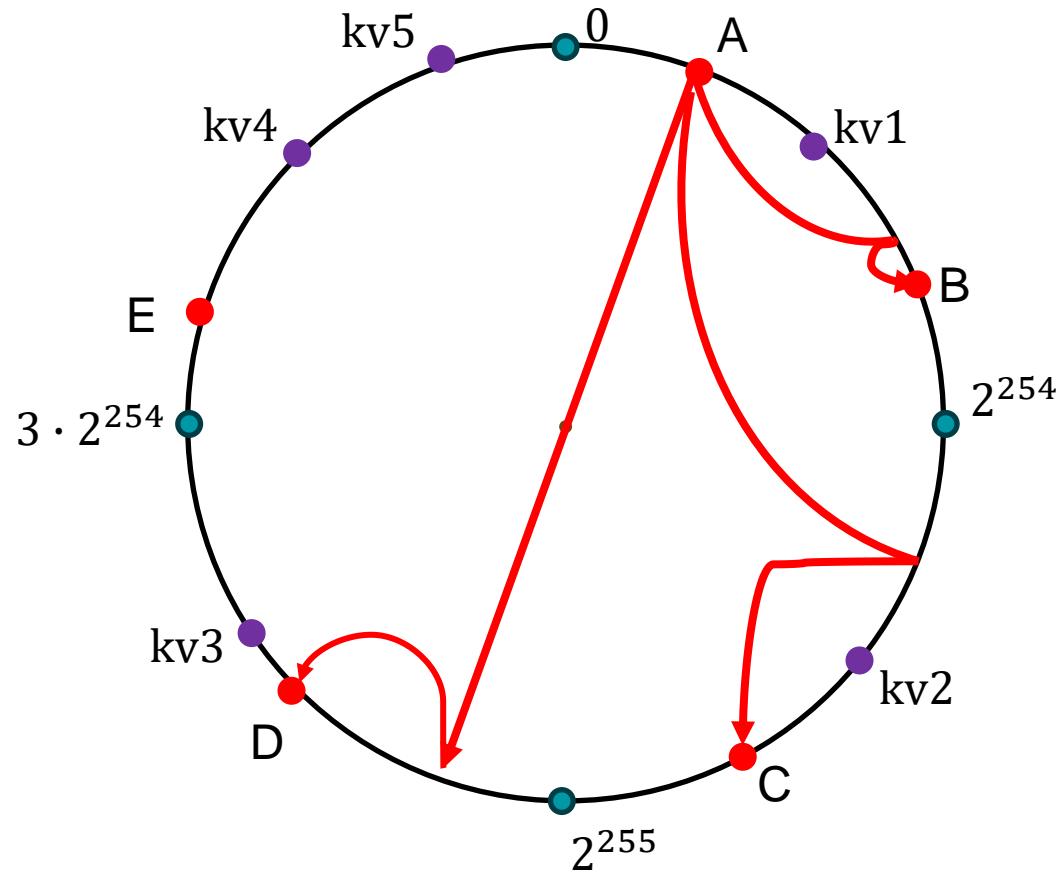
How do we make this  $O(\log n)$  ?  
(in storage, network, etc.)

- Using only successors:  
 $O(1)$  routing table size  
 $O(n)$  access ☹
  - Binary search ?  
Finger tables !



# Chord DHT – Finger tables

Distance	Bucket
1	(successor) = B
$\frac{1}{2}$ circle	D
$\frac{1}{4}$ circle	C
$\frac{1}{8}$ circle	B
...	...



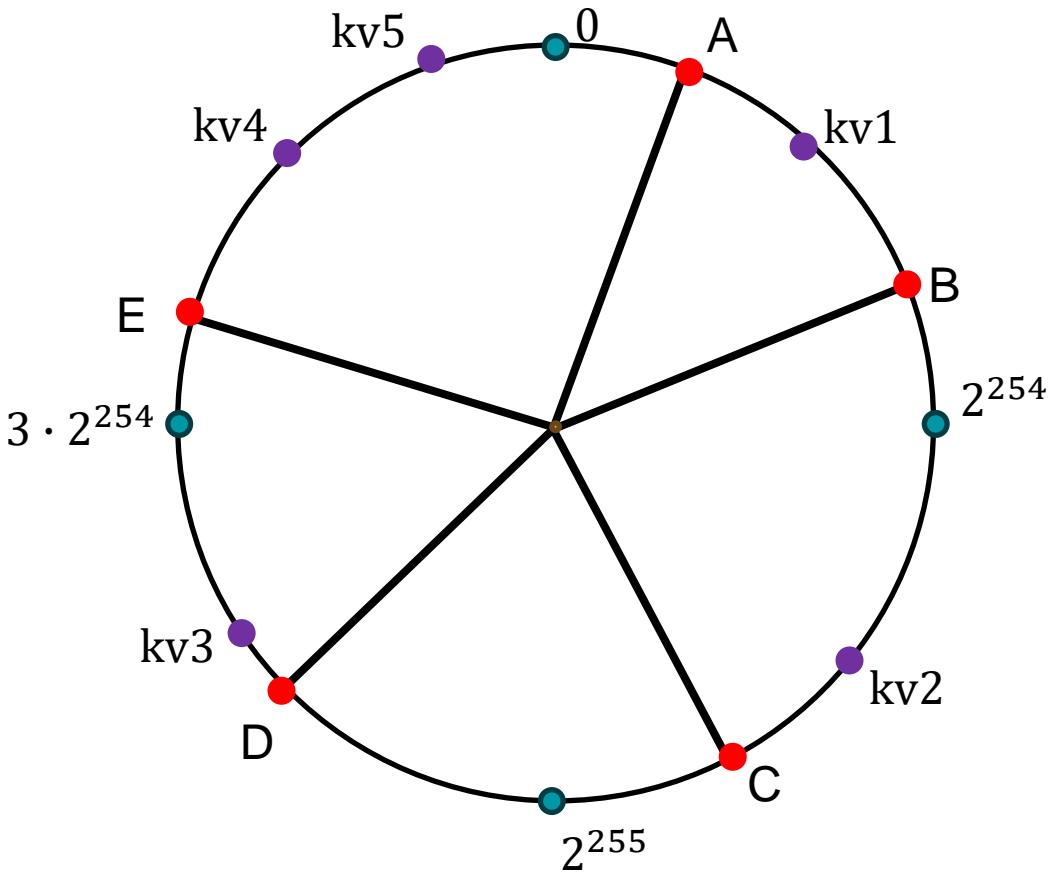
# Chord DHT – Churn

Need to handle:

- Concurrent joining
- Nodes leaving (gracefully)
- Nodes leaving (unresponsive)

Approach:

- Split correctness & performance
- Transient failures can be retried



# Chord DHT – Degenerate cases

Network partitions can lead to some (transient) degenerate cases.

Where:

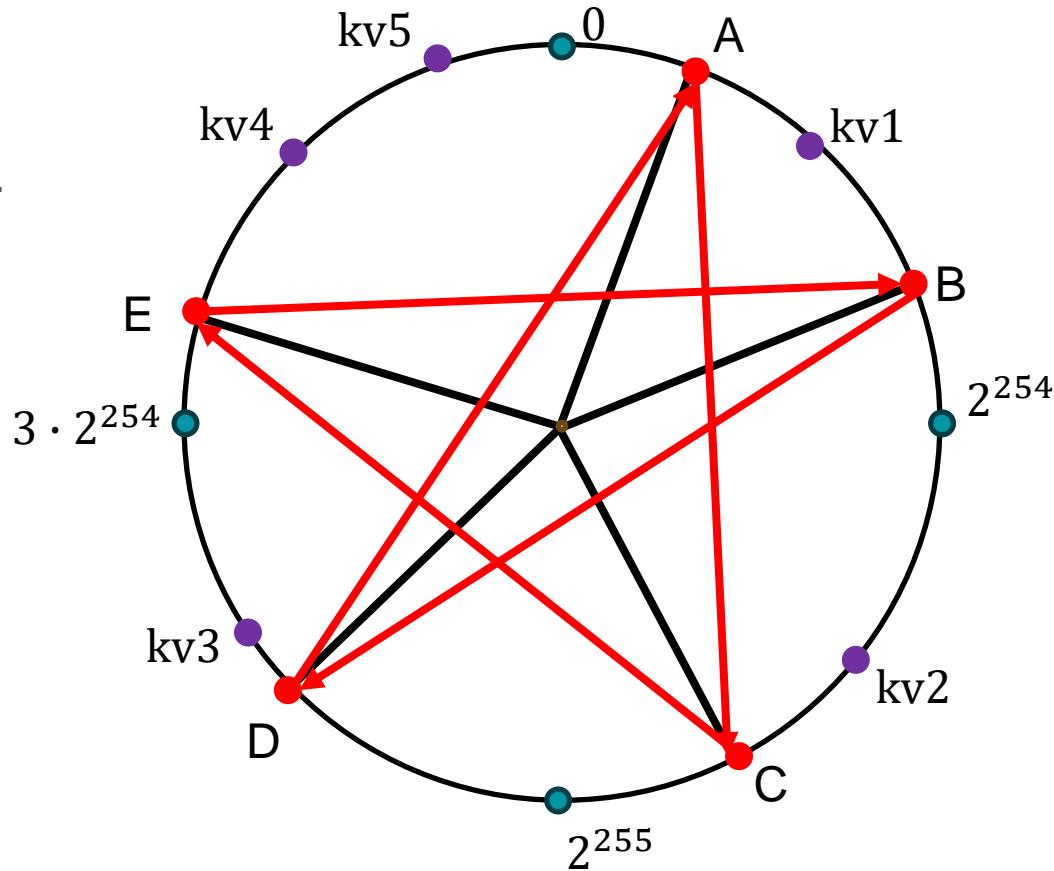
B's successor is D

D's successor is A

A's successor is C

C's successor is E

etc.



# Chord DHT – Possible attacks

- Sybil attacks
- **Eclipse attacks**
- Churn attacks
- Adversarial routing
- Denial of Service

# Next steps - Readings

Mandatory:

- DSDV: Routing over a Multihop Wireless Network of Mobile Computers
- Chord: A Scalable P2P Lookup Service for Internet Applications

Recommended (Engineering):

- The Babel Routing Protocol (RFC 8966)
- Kademlia: A Peer-to-Peer Information System Based on the XOR Metric

... and a few others for the curious among you ...

→ Use Friday's session to ask questions