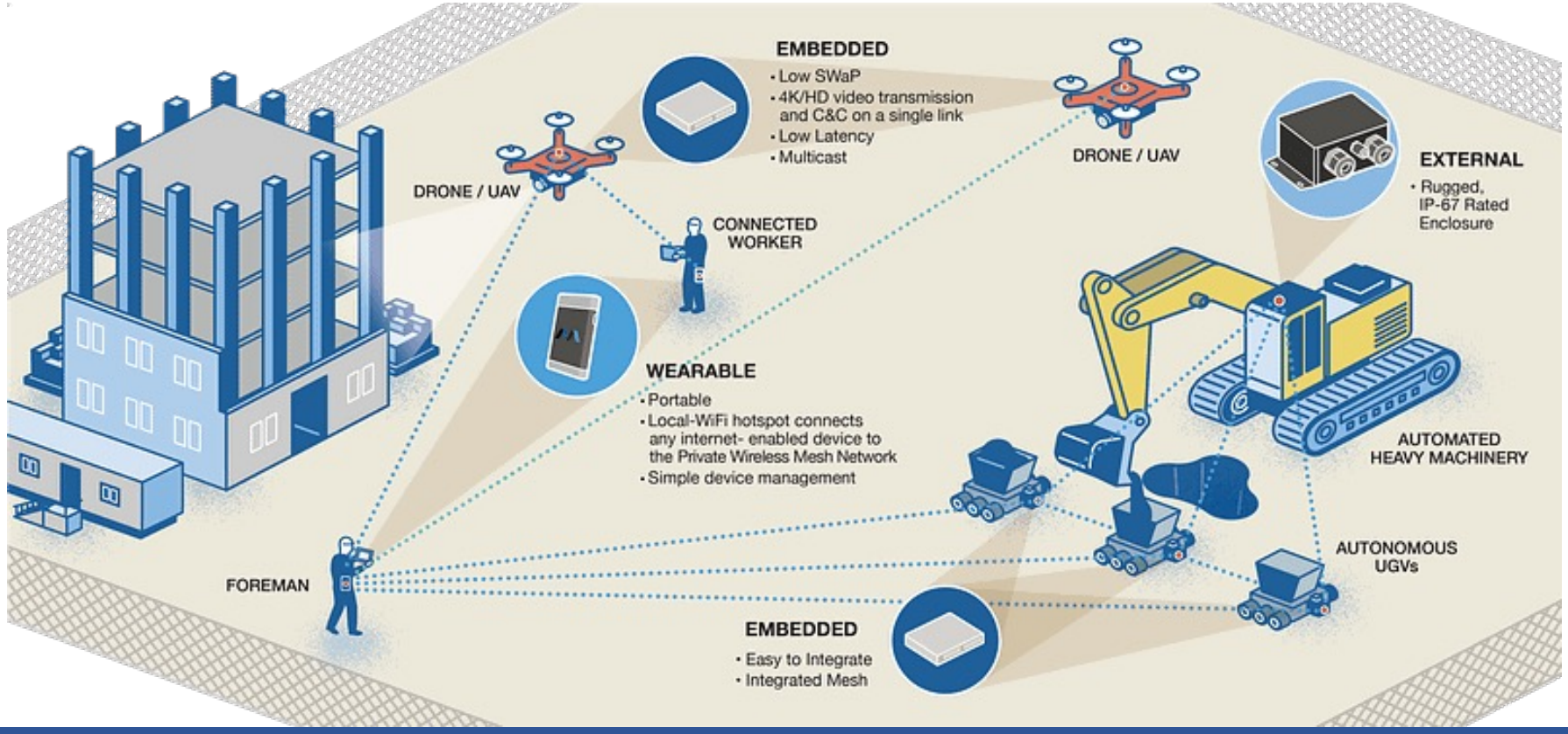
Nhat (Nick) Pham

UNIVERSITY OF OXFORD

Lecture 2: *"Localization, Routing, and Networking"*

(With grateful thanks to Andrew Markham)



Localization

Overview

- Acquiring sensor data without knowing where it came from is meaningless
- How do we know where a thing is?

Manual

- Simplest way is to manually record where a device was left
- No additional device hardware required
- Hugely labour intensive
- Not suitable for dense or mobile deployments

GPS/GNSS

- De facto outdoor positioning solution
- Typical accuracies of:
  - a few metres (standard mode)
  - a few millimetres (real-time kinematic mode)
- As a side-effect of navigation, also provide nanosecond level timing reference

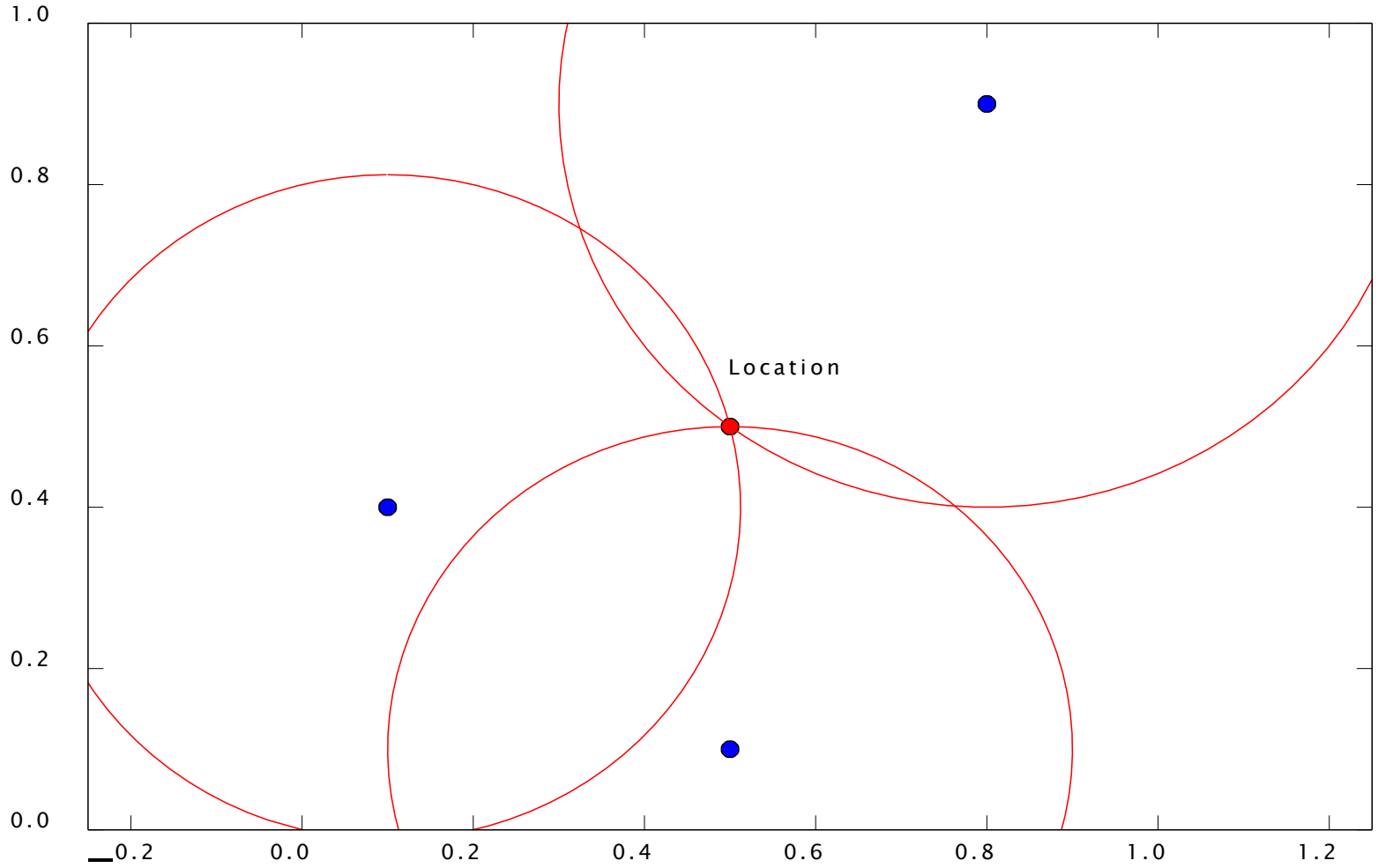
GPS/GNSS

- Not available indoors/underground/in cluttered environments
- High power consumption to obtain a fix
- Increases physical size
- Increases system cost by ~£10-20 per unit

WiFi

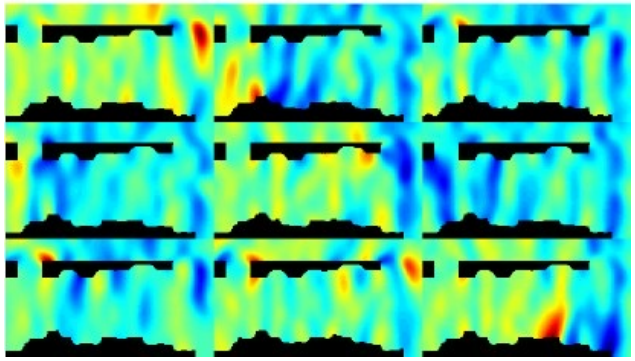
- WiFi access points are ubiquitous
- Recall our channel modelling
- If we assume free-space model, we can relate received signal strength (RSSI) to distance
- Given three or more distance measurements, simple trilateration can determine the position of the receiver

Trilateration



Radio map

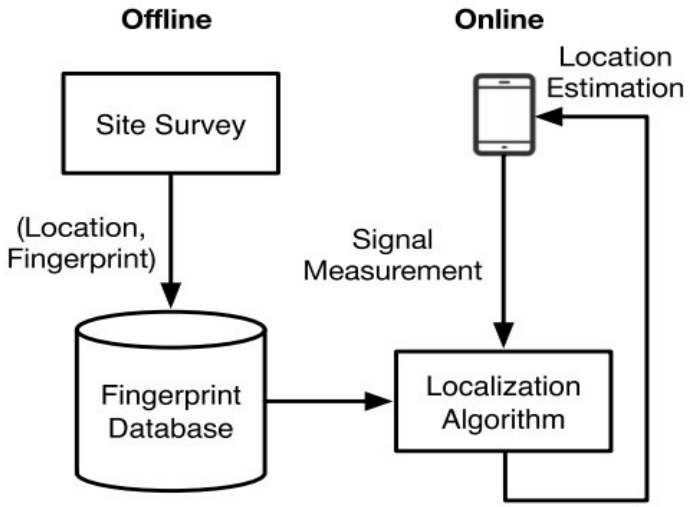
In reality, especially indoors, trilateration does not work, as the channel is too complex to model, so we use techniques such as fingerprinting[1] instead.



[1]Christian Beder, Alan McGibney, and Martin Klepal. "Predicting the expected accuracy for fingerprinting based WiFi localisation systems". In: *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on*. IEEE. 2011, pp. 1–6.

Fingerprinting

- Rather than trying to model the variability, a simple way is to exploit these variations to determine location
- *Offline Phase:* Survey (measure) WiFi fingerprints at multiple locations to build a fingerprint map
- *Online Phase:* Given a measurement, determine the most likely location that would have produced it



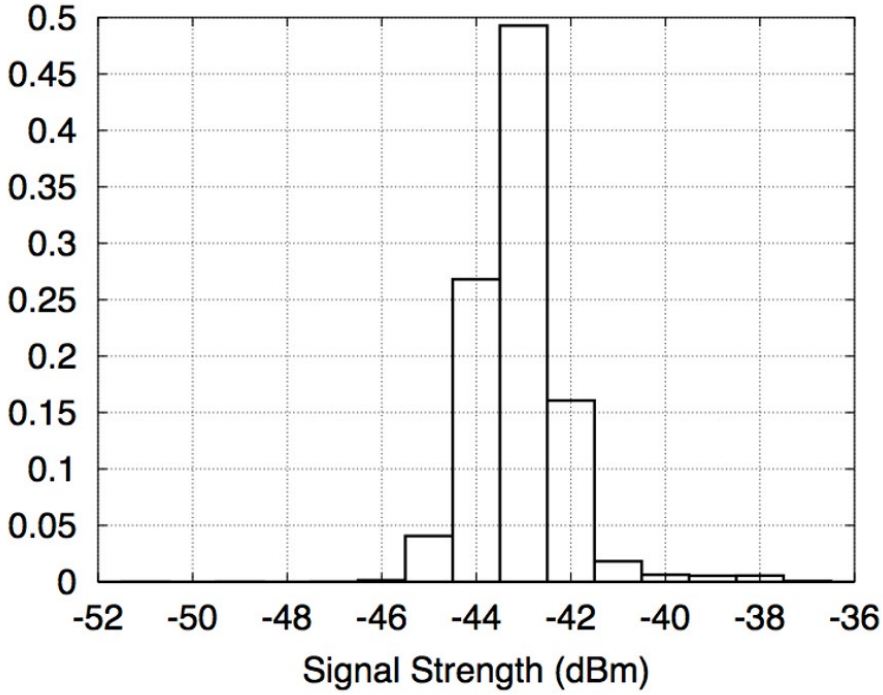
The Horus Localization System

- Horus[2] is a simple, probabilistic WiFi localization system
- It builds a map that captures:
  - the mean signal strength
  - the variance (variability of readings)
  - the temporal stability (autocorrelation of readings)
- It uses these surveyed parameters to estimate location

[2]Moustafa Youssef and Ashok Agrawala. "The Horus WLAN location determination system". In: *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM. 2005, pp. 205–218.

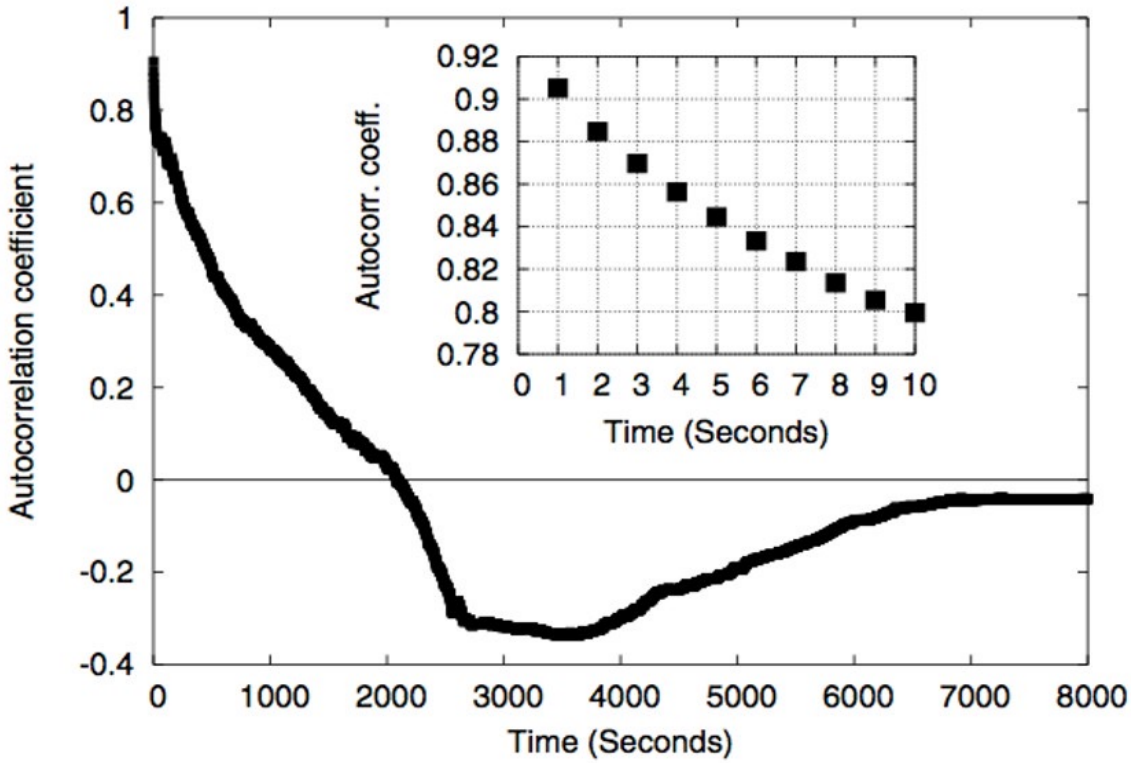
Temporal variation

- 5 minute WiFi signal variability histogram at a single location



Temporal correlation

- Long term predictability of signal from a single location



Localization

Weird and wonderful positioning techniques

Where in the world am I?

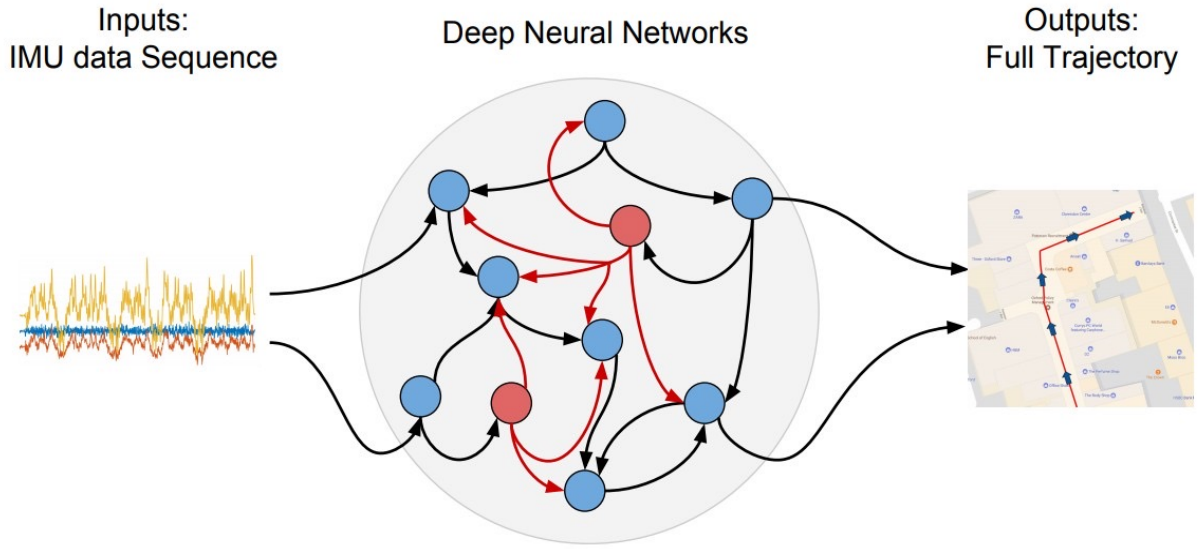
- ❑ We are not limited to using radio signals for localization
- ❑ We can use any physical property that captures either absolute location or relative motion
- ❑ We often use more than one modality (sensor fusion) to enhance robustness and accuracy

Inertial Measurement Units

- Force must be applied to a device to move it around
- By measuring the body acceleration, the device's trajectory can be reconstructed
- This can be done through double integration:
  - acceleration → velocity → displacement
- Double integration suffers from accumulative drift
- IMU sensors are present in all modern smartphones

Inertial Measurement

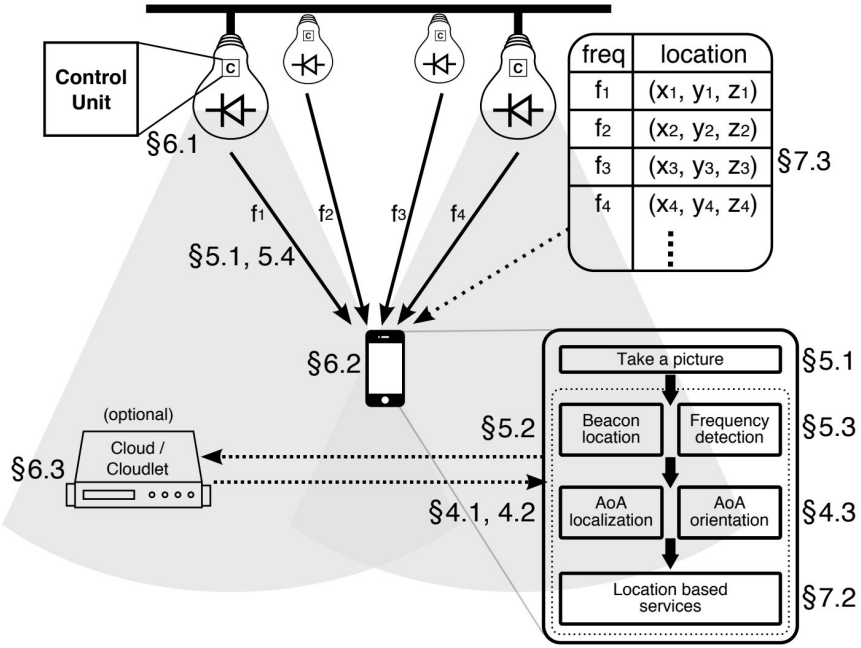
- Deep-learning the relationship between inertial sensors and resultant trajectory[3]



[3]Changhao Chen et al. "IONet: Learning to Cure the Curse of Drift in Inertial Odometry". In: *IEEE Transactions on Mobile Computing 2020.*

LiFi[4]

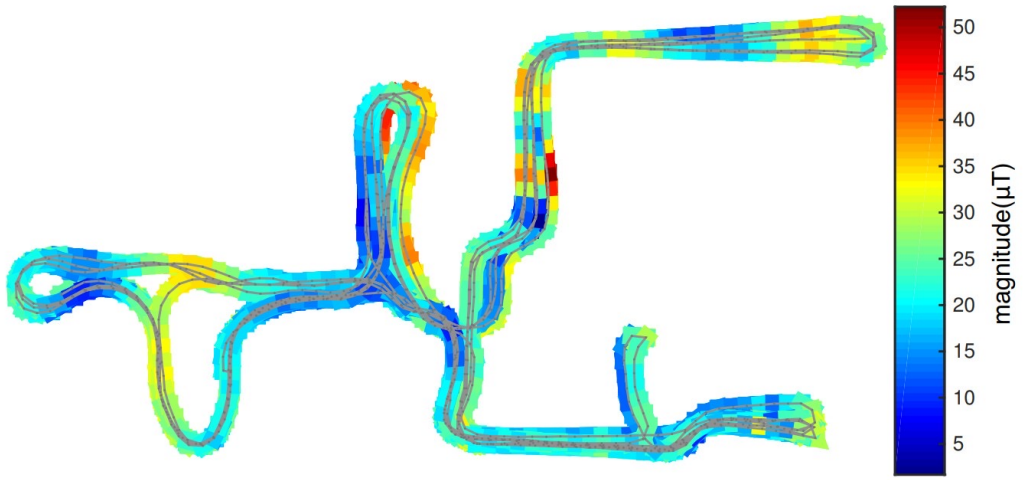
- Use constellation of LEDs to determine 3-D location



[4]Ye-Sheng Kuo et al. "Luxapose: Indoor positioning with mobile phones and visible light". In: *Proceedings of the 20th annual international conference on Mobile computing and networking*. ACM. 2014, pp. 447–458.

Geomagnetic[5]

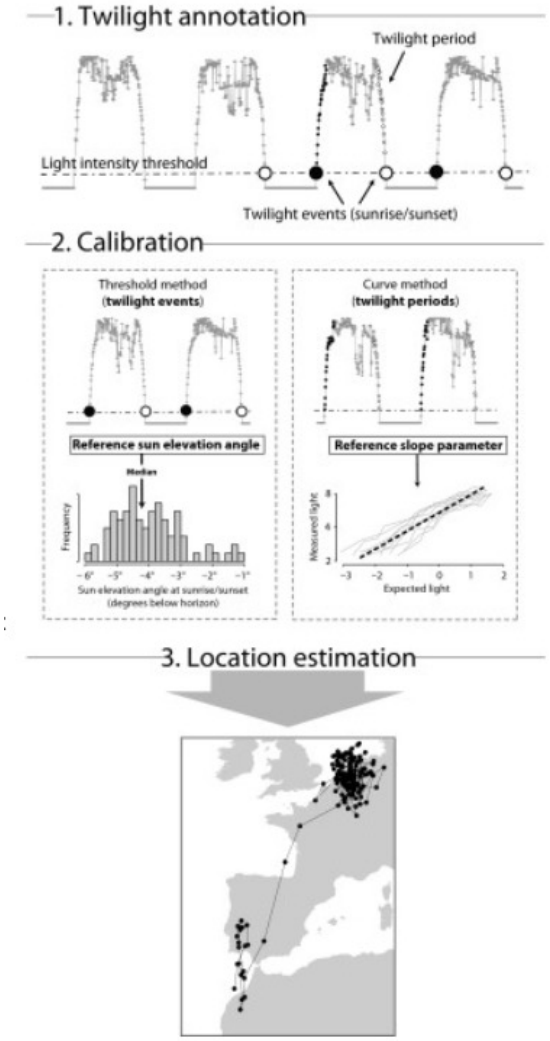
- Use distortion of Earth's magnetic field caused by building itself



[5]Sen Wang et al. "Keyframe based large-scale indoor localisation using geomagnetic field and motion pattern". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 1910–1917.

Light-level geolocators

- Use length of day and time of sunrise/sunset to determine global location



The steps required to obtain location estimates from raw light-level geolocator data. Schematic from Lisovski et al. (2020).

http://migratoryconnectivityproject.org/geolocators/

Further reading

Suining He and S-H Gary Chan. "Wi-Fi fingerprint-based indoor positioning: Recent advances and comparisons". In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 466–490

Pavel Davidson and Robert Piché. "A survey of selected indoor positioning methods for smartphones". In: *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 1347–1370

Trong-Hop Do and Myungsik Yoo. "An in-depth survey of visible light communication based positioning systems". In: *Sensors* 16.5 (2016), p. 678

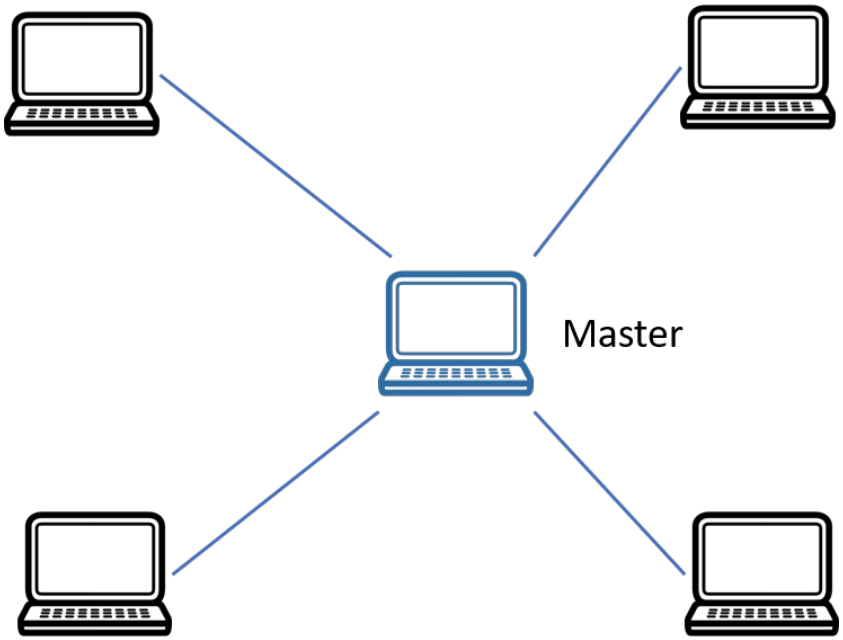
Routing and Networking

Overview

- How can devices form a network?
- How do we deal with factors such as mobility or limited power?

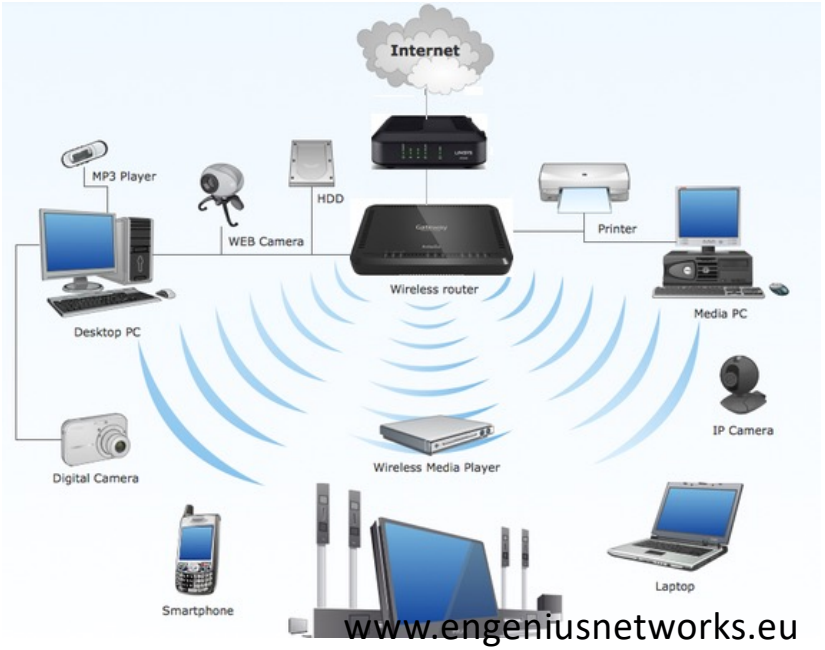
Star topology

- Simplest topology
- Master controls entire 1-hop network

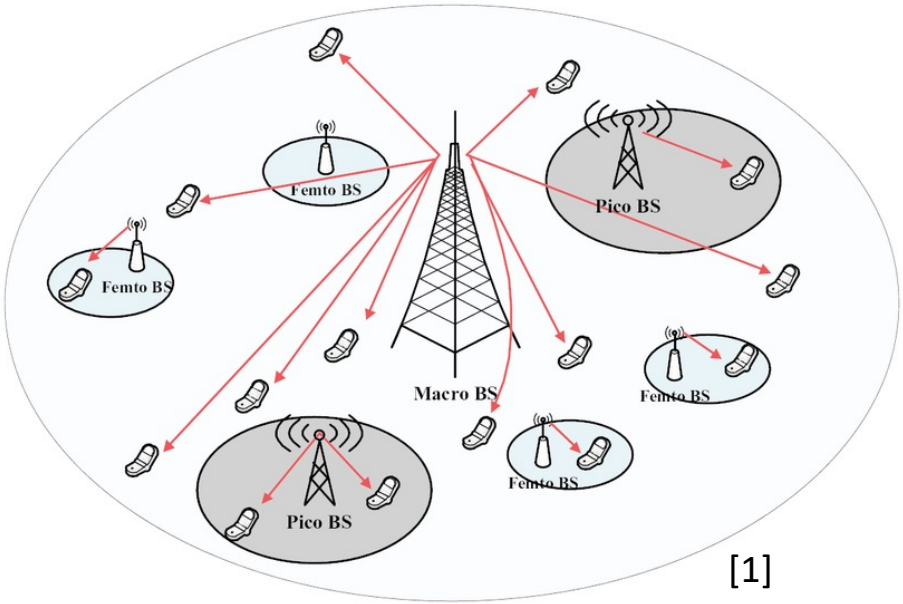


Star topology examples:

- 802.11 PCF: WiFi Access Points
- Cellular connectivity
- LoRaWAN: low-power nodes communicate with a single basestation



www.engeniusnetworks.eu



[1]

Star topology

- Expensive, complex master
- Simple, low-power end-devices
- All slaves must be in range of the master
- Communication between slaves must go through the master
- Typical low latency and no collisions

Routing and Networking

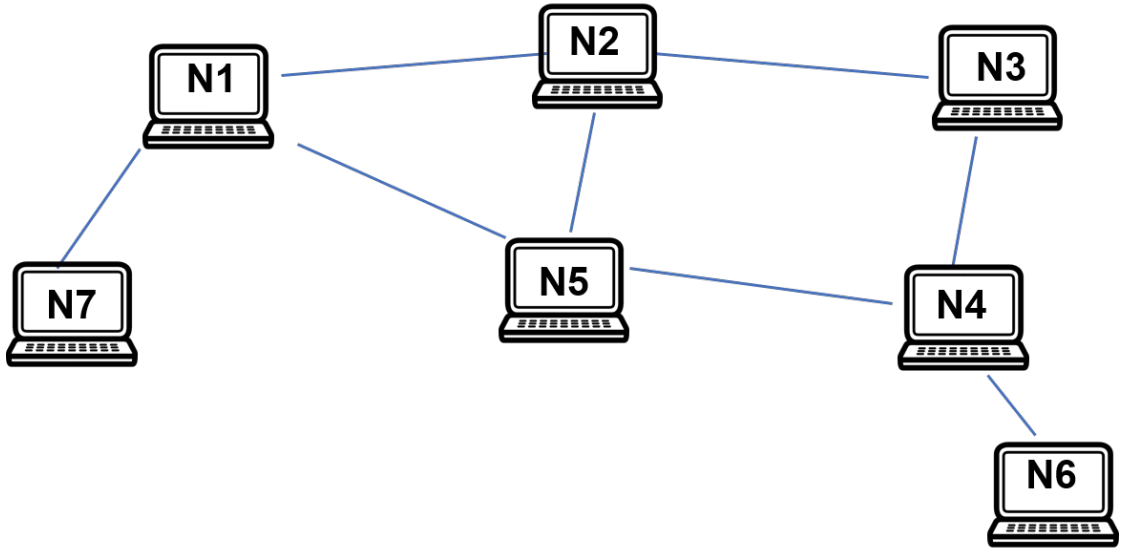
Routing in mobile ad-hoc networks

Mobile Ad-hoc Network (MANET)

- Mobile devices opportunistically form a network
- Vehicle-to-vehicle (V2V) networks
- Distributed comms (battlefield)
- Peer-to-peer (pocket switched networks)

Why do we need network routing?

- Not possible to directly communicate with all other nodes
- Packets must travel over multiple hops

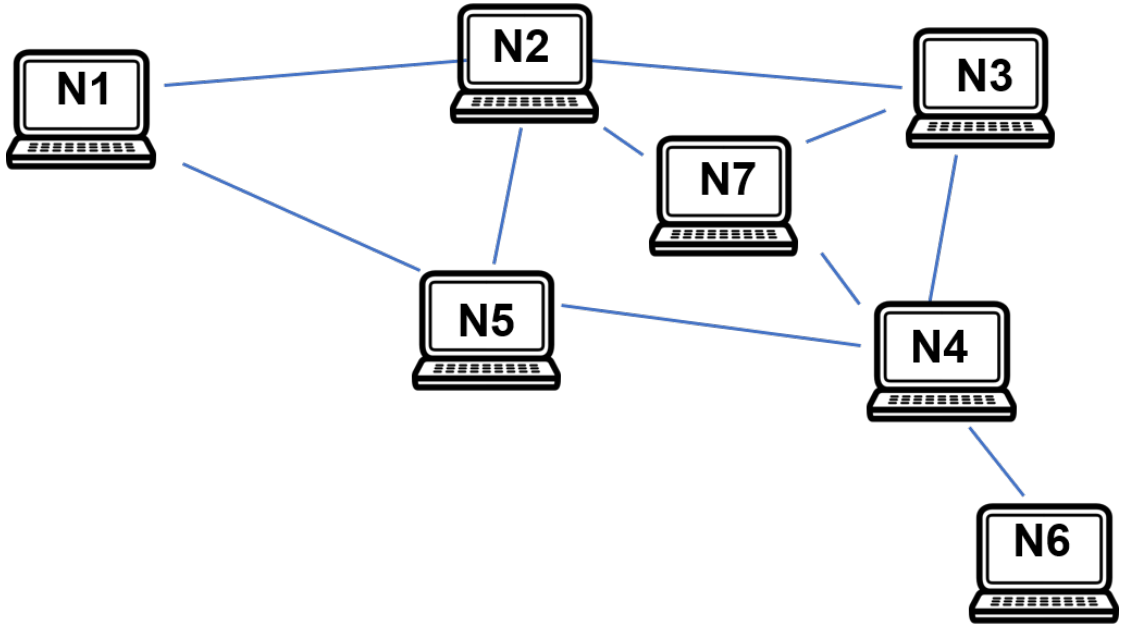


Routing problem

- For each node find the best path to the destination
- 'Best' can be:
  - Shortest (fewest hops)
  - Fastest (lowest latency)
  - Most robust
  - Combination of the above
- Distributed routing problem: no master that performs routing

Challenges: Dynamic Connectivity

- Nodes can move, be added or removed, breaking existing links and forming new links
- Links may be intermittent (see fading)



Challenges

- Mobile nodes often have stringent energy and memory constraints
- Wireless links have limited bandwidth, and time-varying link error probability
- Some applications require real-time data delivery
- Some applications require robust data delivery

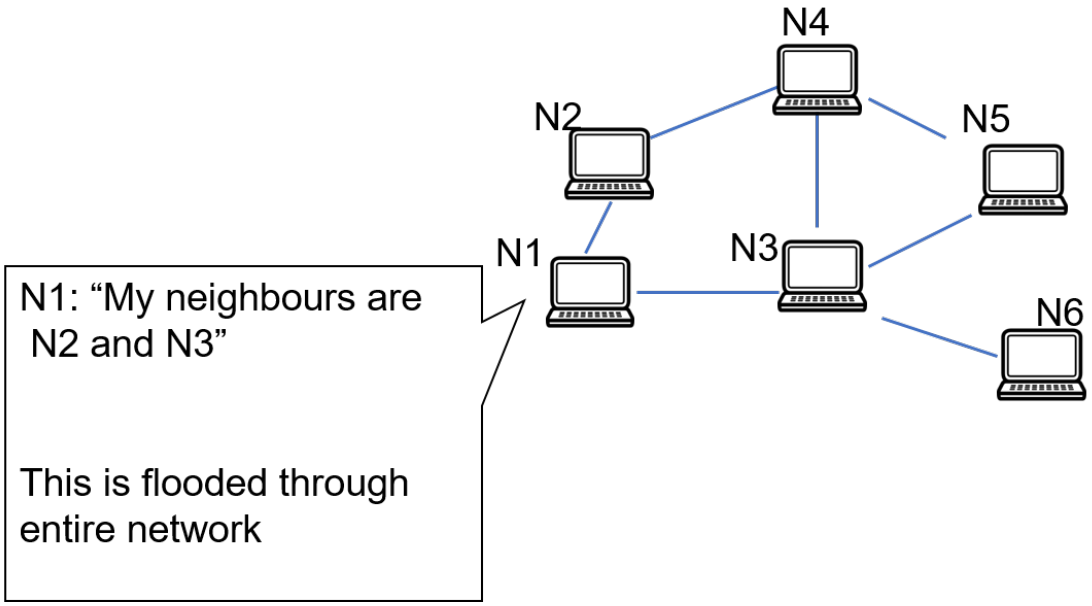
Design goals

- ❑ Fully distributed - no centralized control
- ❑ Adaptivity to topology changes
- ❑ Loop-free paths
- ❑ Fast route discovery (for real-time applications)
- ❑ Low control overhead (for bandwidth-/energy-constrained applications)
- ❑ Fault-tolerance (for security/emergency applications)

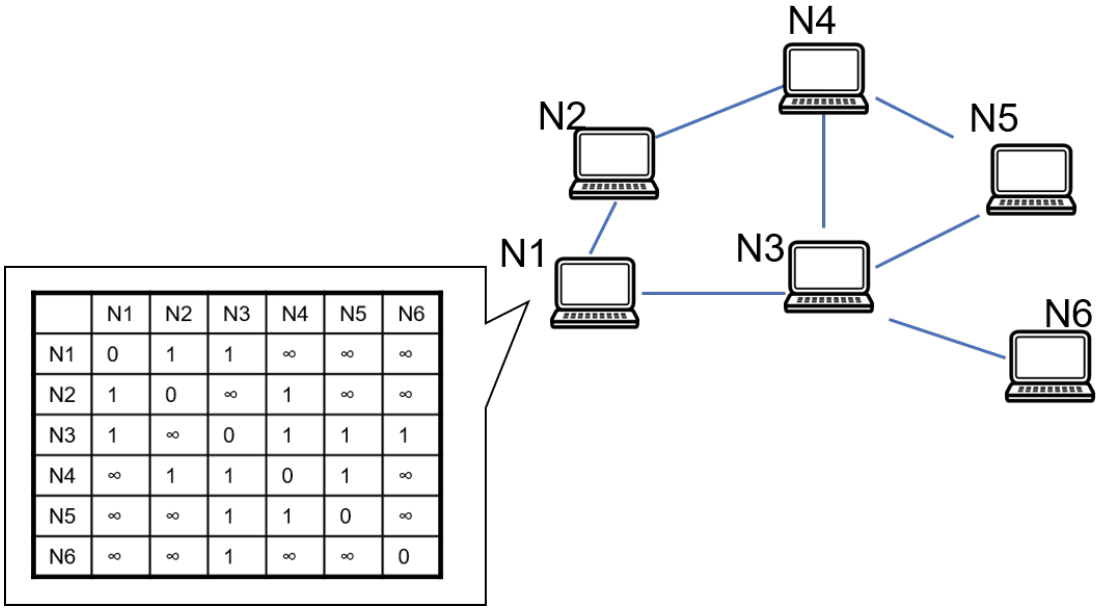
Link-State (Source) Routing

- Each node obtains global knowledge about which nodes have links between them
- This is done by periodically exchanging packets with neighbours
- A table (adjacency matrix) is populated and shared
- Adjacency matrix is a binary matrix which indicates whether a pair of nodes have a link or not

Link-State Routing



Link-State Routing



	N1	N2	N3	N4	N5	N6
N1	0	1	1	∞	∞	∞
N2	1	0	∞	1	∞	∞
N3	1	∞	0	1	1	1
N4	∞	1	1	0	1	∞
N5	∞	∞	1	1	0	∞
N6	∞	∞	1	∞	∞	0

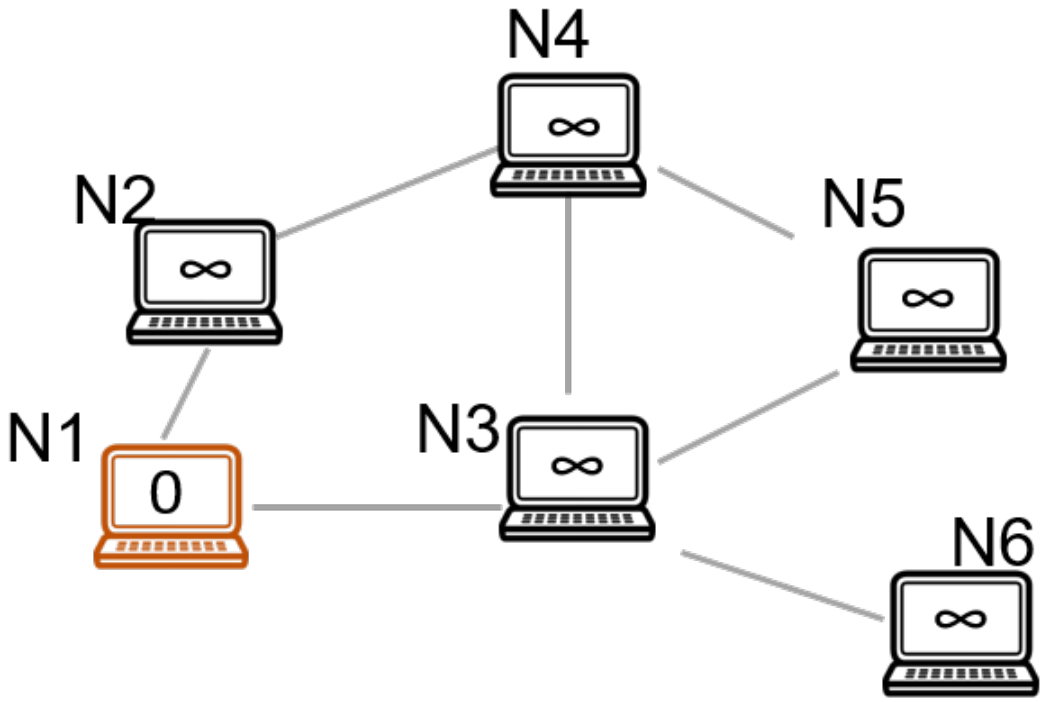
Dijkstra's algorithm

- Dijkstra's algorithm is an efficient, shortest path algorithm[1]
- It calculates the min-cost tree from a single source throughout the network

[1]By example: https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html

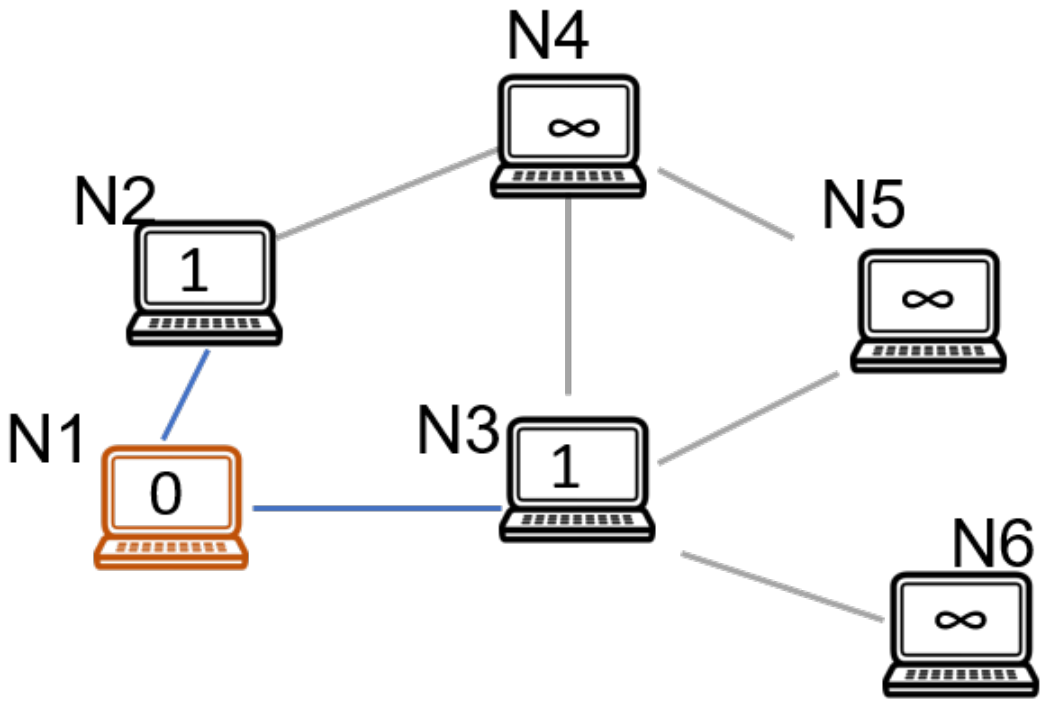
Dijkstra's algorithm example

- Start from self. All other nodes are unknown, hence infinite costs



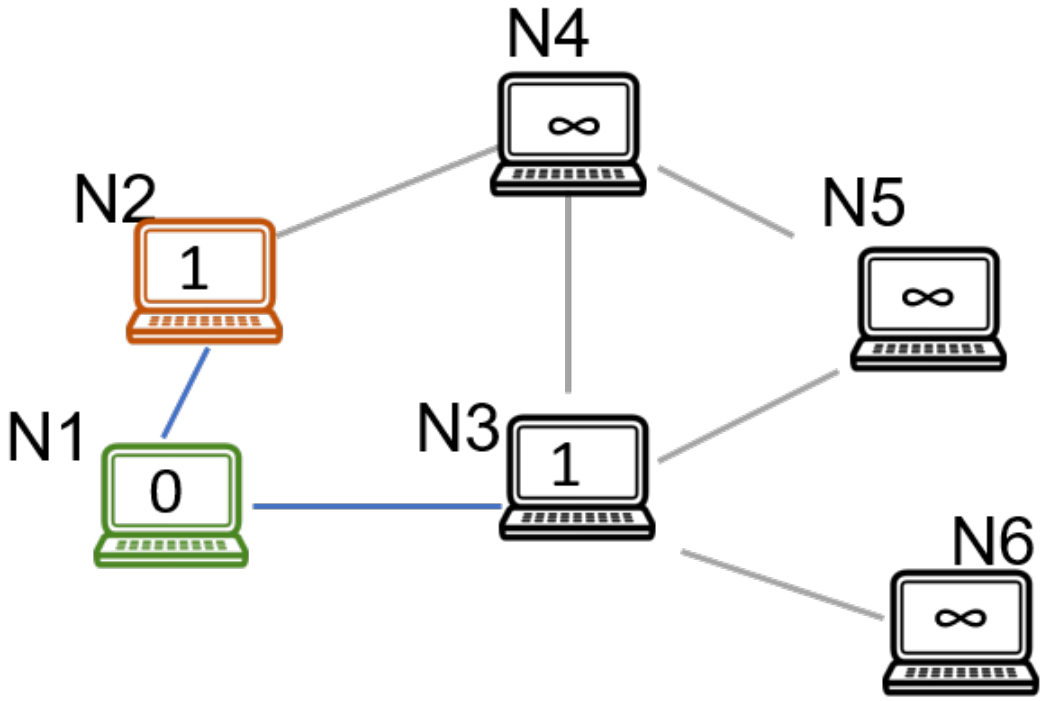
Dijkstra's algorithm example

- Update neighbours with their costs



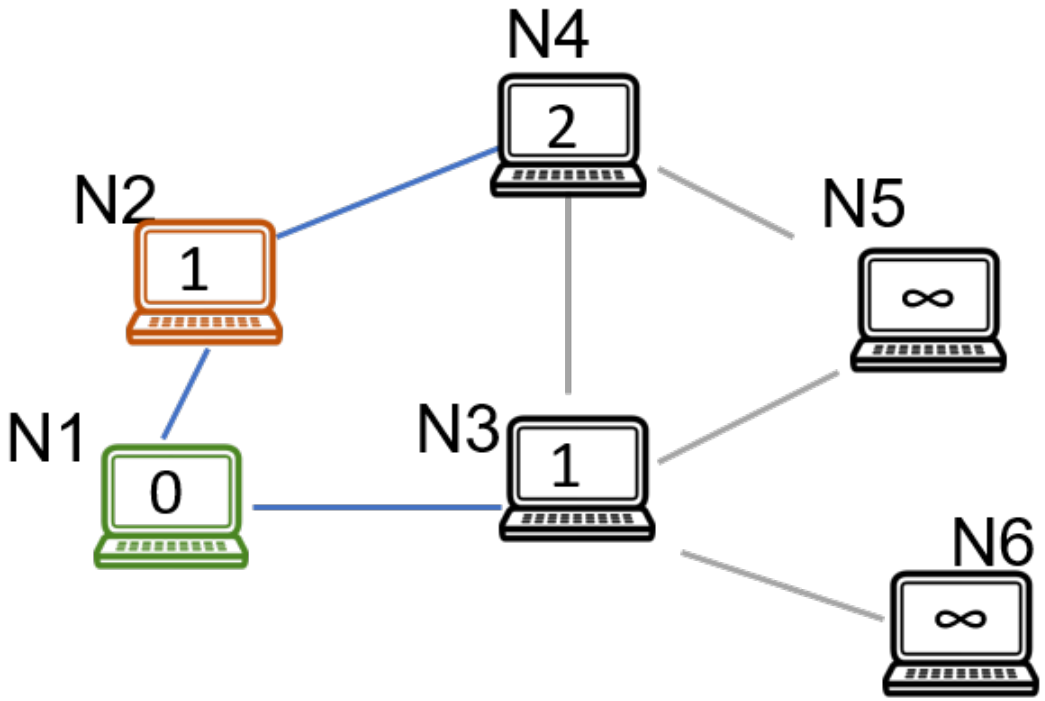
Dijkstra's algorithm example

- N1 is marked as done. Choose next lowest cost (either N2 or N3 at random)



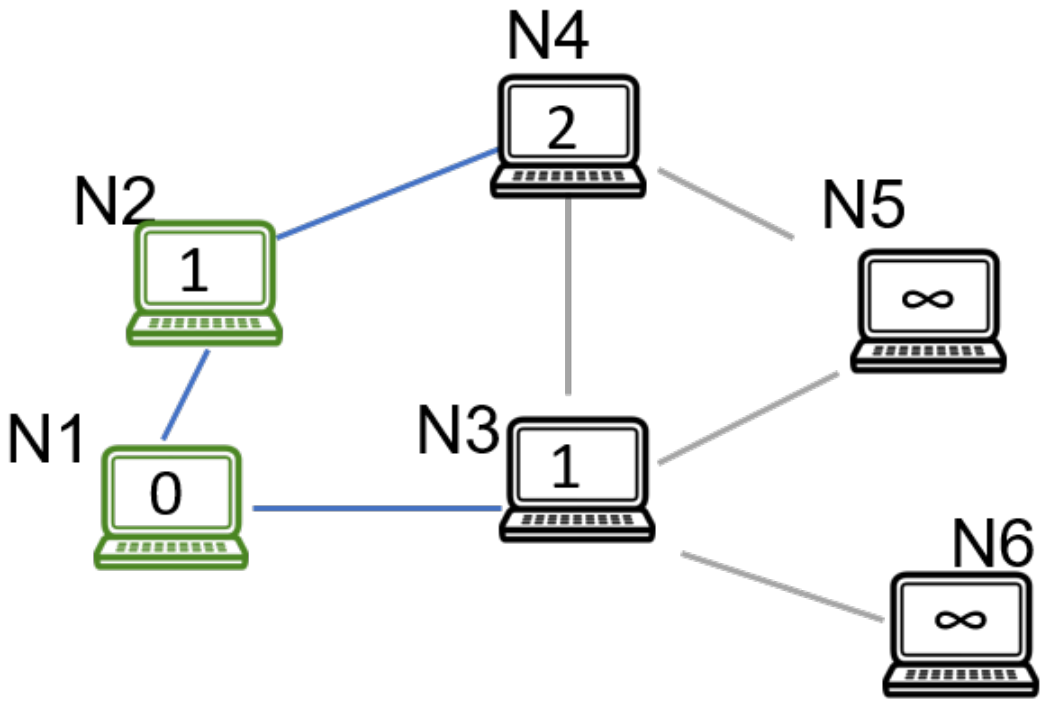
Dijkstra's algorithm example

- Update cost of reachable nodes that are not done



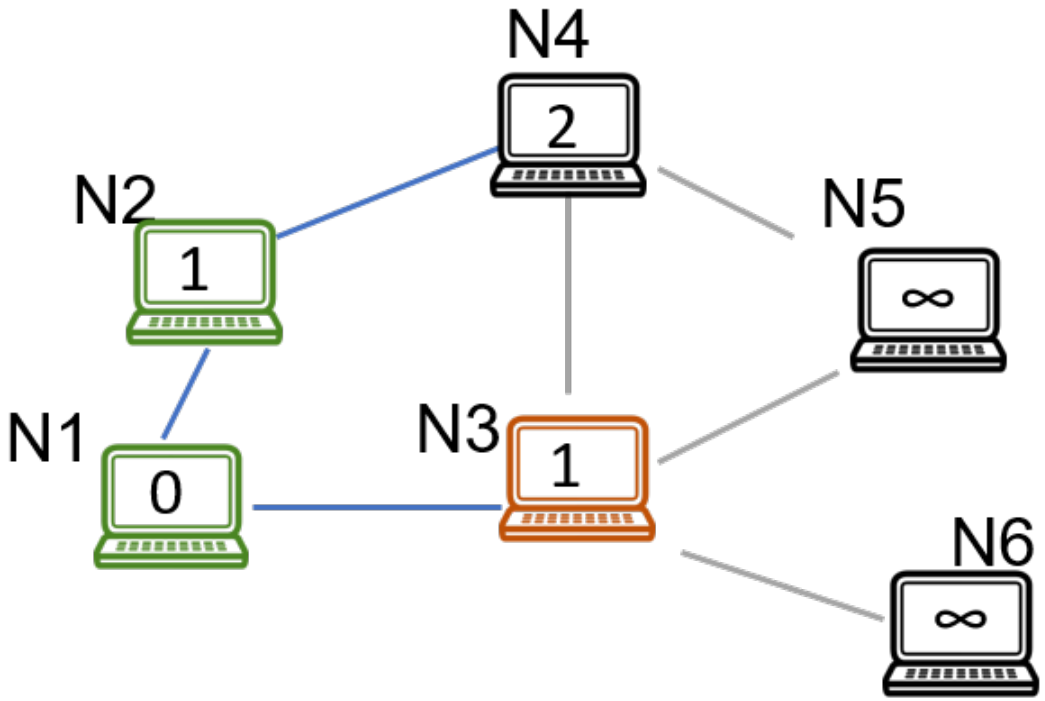
Dijkstra's algorithm example

- Mark N2 as done



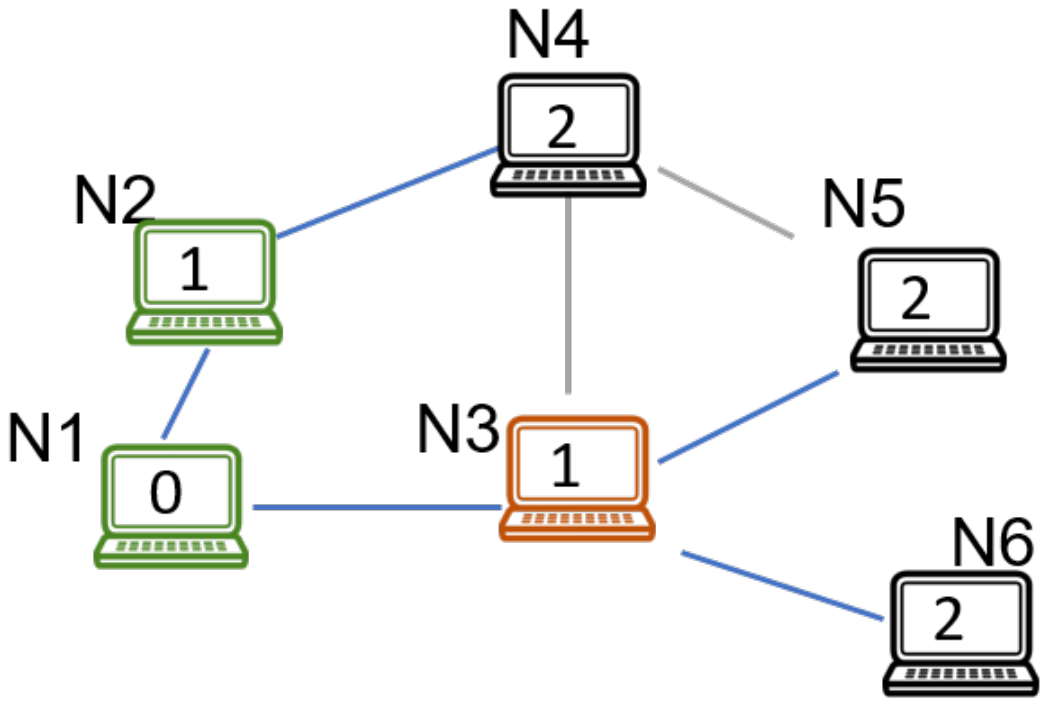
Dijkstra's algorithm example

- Choose next lowest cost, N3 in this case



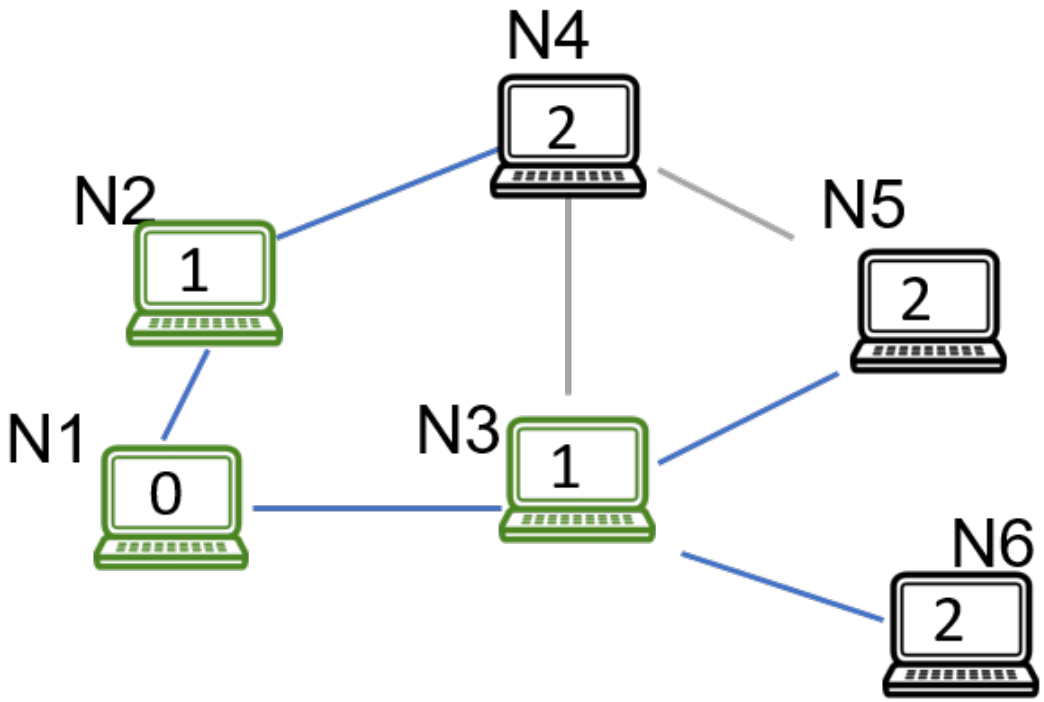
Dijkstra's algorithm example

- Update reachable nodes which are not done



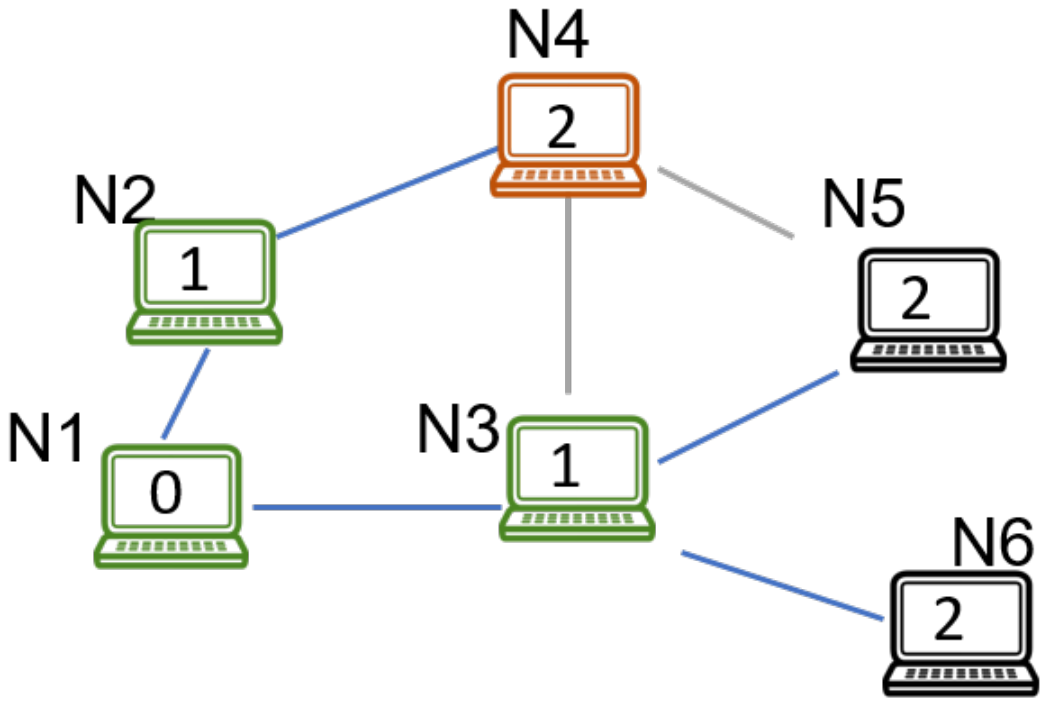
Dijkstra's algorithm example

- Mark N3 as done



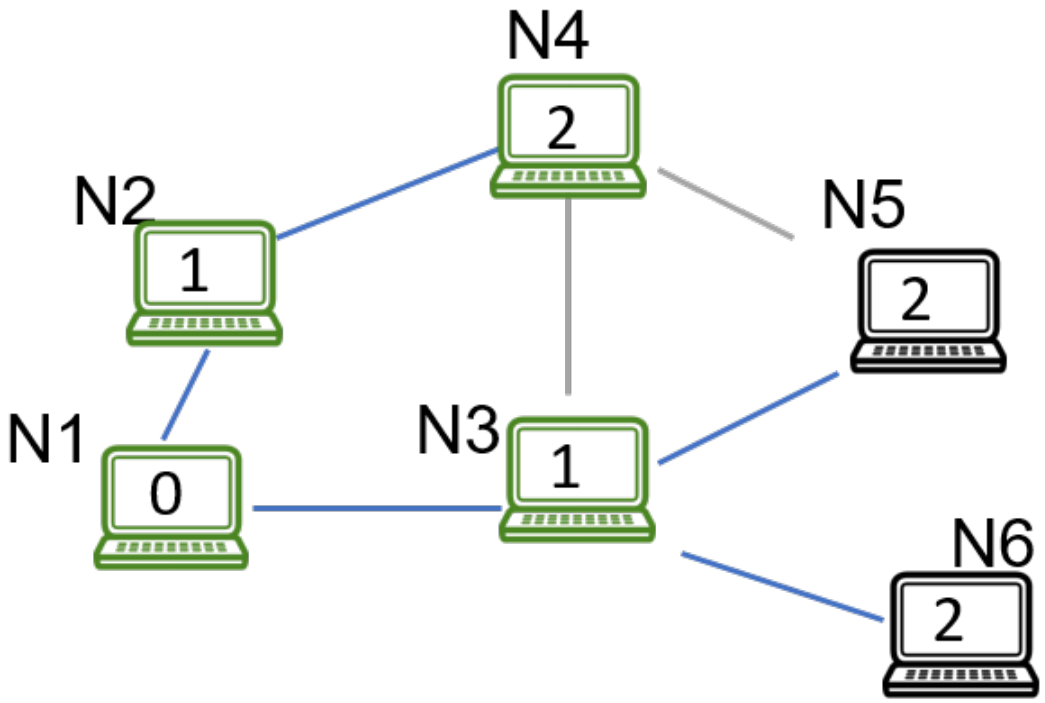
Dijkstra's algorithm example

- Choose next lowest cost, N4 in this case



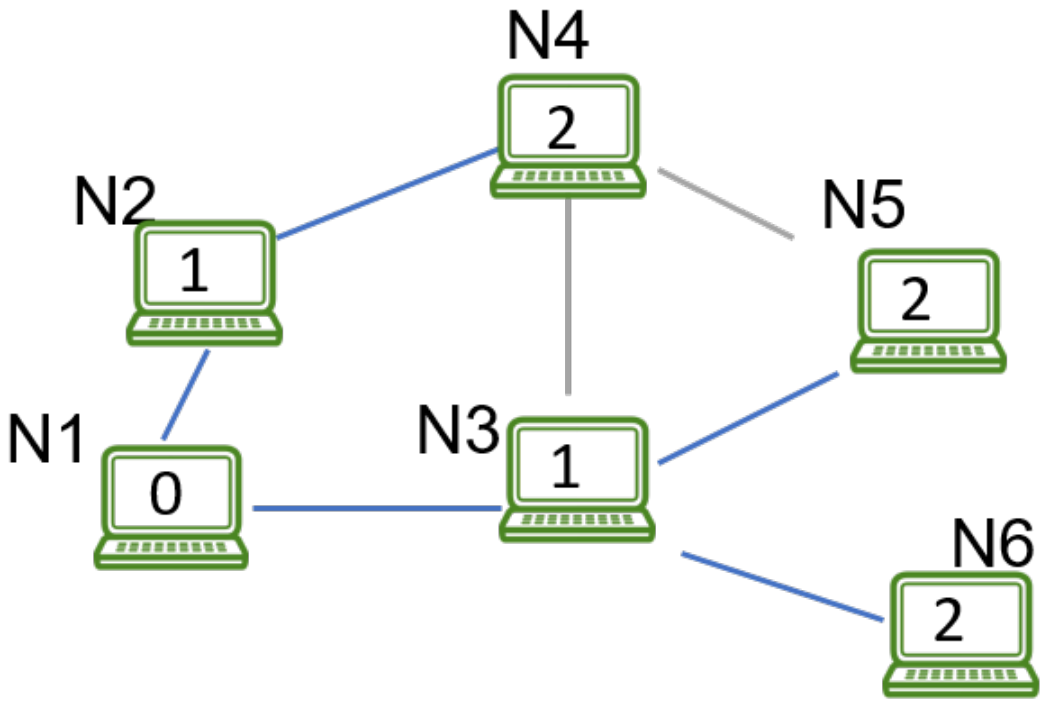
Dijkstra's algorithm example

- N4 can't improve anything, so mark as done



Dijkstra's algorithm example

- Eventually, the entire topology routed at N1 is computed



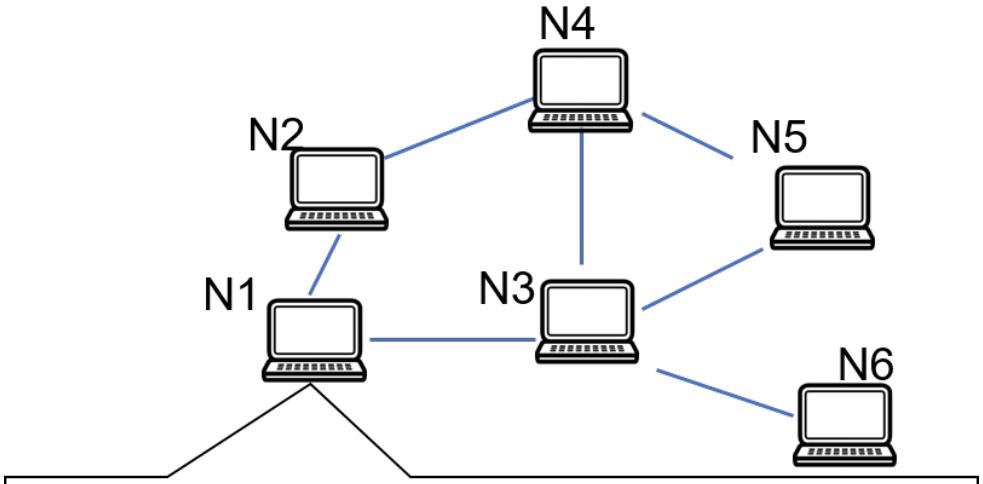
Link-State Routing

- All nodes know the exact path the packet should take at source
- Obtaining global information can be expensive
- Small network changes lead to new routing tables
- This ‘churn’ can be expensive, especially with high mobility

Distance Vector Routing

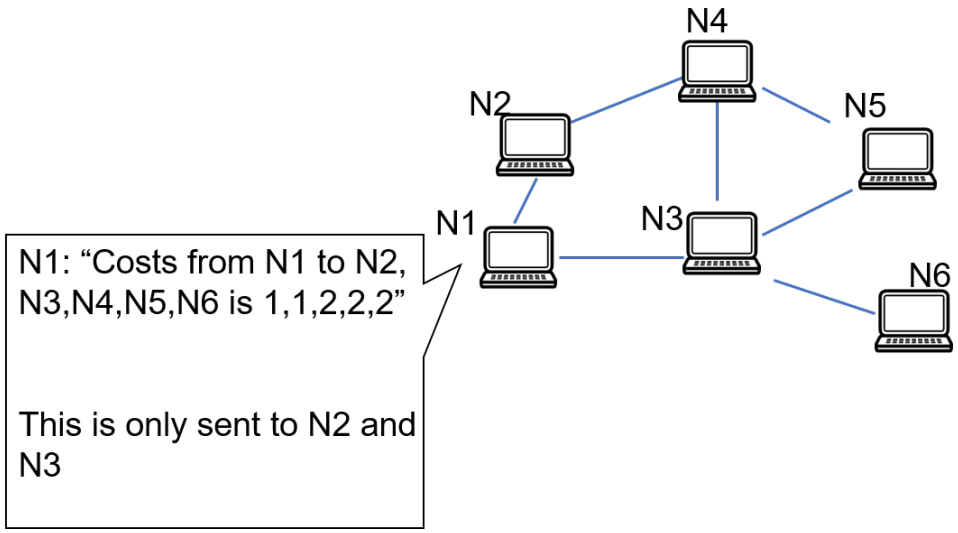
- Store next hop and which edge (vector) to take
- No need for global topology knowledge
- Nodes share routing tables with direct neighbours
- Use their information to update paths

Distance Vector Routing



Dest.	Cost via N2	Cost via N3	Min cost	Next-hop
N2	**1**	3	**1**	N2
N3	3	**1**	**1**	N3
N4	**2**	**2**	**2**	N2 / N3
N5	3	**2**	**2**	N3
N6	4	**2**	**2**	N3

Distance Vector Routing



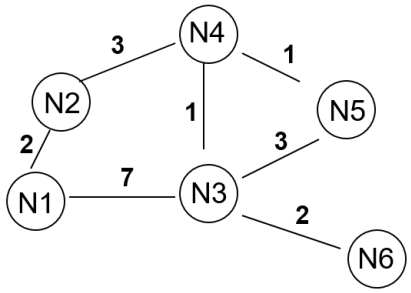
Distributed Bellman-Ford Algorithm

- Initially, nodes only know their neighbours' costs

Distance to node …

Info stored at node …

	N1	N2	N3	N4	N5	N6
N1	0	2 (N2)	7 (N3)	∞	∞	∞
N2	2 (N1)	0	∞	3 (N4)	∞	∞
N3	7 (N1)	∞	0	1 (N4)	3 (N5)	2 (N6)
N4	∞	3 (N2)	1 (N3)	0	1 (N5)	∞
N5	∞	∞	3 (N3)	1 (N4)	0	∞
N6	∞	∞	2 (N3)	∞	∞	0



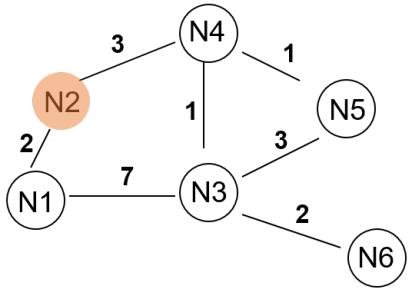
Distributed Bellman-Ford Algorithm

- N2 broadcasts

Distance to node …

Info stored at node …

	N1	N2	N3	N4	N5	N6
N1	0	2 (N2)	7 (N3)	∞	∞	∞
N2	2 (N1)	0	∞	3 (N4)	∞	∞
N3	7 (N1)	∞	0	1 (N4)	3 (N5)	2 (N6)
N4	∞	3 (N2)	1 (N3)	0	1 (N5)	∞
N5	∞	∞	3 (N3)	1 (N4)	0	∞
N6	∞	∞	2 (N3)	∞	∞	0



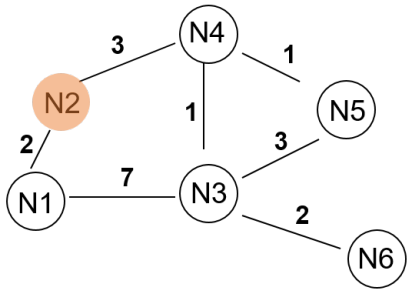
Distributed Bellman-Ford Algorithm

- N2 sends its routing table

Distance to node …

Info stored at node …

	N1	N2	N3	N4	N5	N6
N1	0	2 (N2)	7 (N3)	∞	∞	∞
N2	2 (N1)	0	∞	3 (N4)	∞	∞
N3	7 (N1)	∞	0	1 (N4)	3 (N5)	2 (N6)
N4	∞	3 (N2)	1 (N3)	0	1 (N5)	∞
N5	∞	∞	3 (N3)	1 (N4)	0	∞
N6	∞	∞	2 (N3)	∞	∞	0



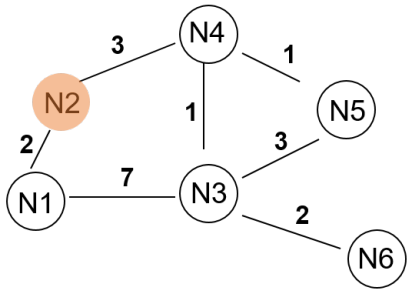
Distributed Bellman-Ford Algorithm

- N1 and N4 can update their routing tables

Distance to node …

Info stored at node …

	N1	N2	N3	N4	N5	N6
N1	0	2 (N2)	7 (N3)	5 (N2)	∞	∞
N2	2 (N1)	0	∞	3 (N4)	∞	∞
N3	7 (N1)	∞	0	1 (N4)	3 (N5)	2 (N6)
N4	5 (N2)	3 (N2)	1 (N3)	0	1 (N5)	∞
N5	∞	∞	3 (N3)	1 (N4)	0	∞
N6	∞	∞	2 (N3)	∞	∞	0



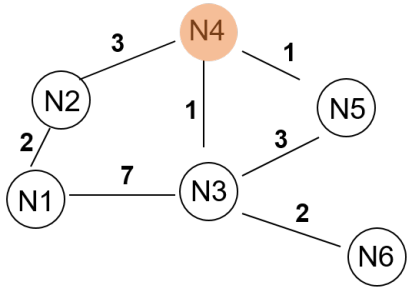
Distributed Bellman-Ford Algorithm

- N4 sends its routing table

Distance to node …

Info stored at node …

	N1	N2	N3	N4	N5	N6
N1	0	2 (N2)	7 (N3)	5 (N2)	∞	∞
N2	2 (N1)	0	∞	3 (N4)	∞	∞
N3	7 (N1)	∞	0	1 (N4)	3 (N5)	2 (N6)
N4	5 (N2)	3 (N2)	1 (N3)	0	1 (N5)	∞
N5	∞	∞	3 (N3)	1 (N4)	0	∞
N6	∞	∞	2 (N3)	∞	∞	0



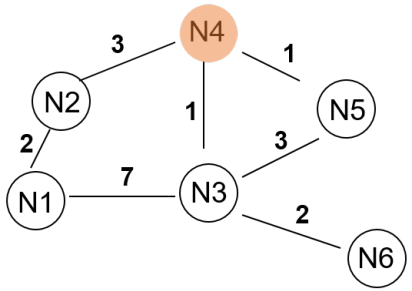
Distributed Bellman-Ford Algorithm

- Entire table starts to become populated

Distance to node …

Info stored at node …

	N1	N2	N3	N4	N5	N6
N1	0	2 (N2)	7 (N3)	5 (N2)	∞	∞
N2	2 (N1)	0	4 (N4)	3 (N4)	4 (N4)	∞
N3	6 (N4)	4 (N4)	0	1 (N4)	2 (N4)	2 (N6)
N4	5 (N2)	3 (N2)	1 (N3)	0	1 (N5)	∞
N5	6 (N4)	4 (N4)	3 (N3)	1 (N4)	0	∞
N6	∞	∞	2 (N3)	∞	∞	0

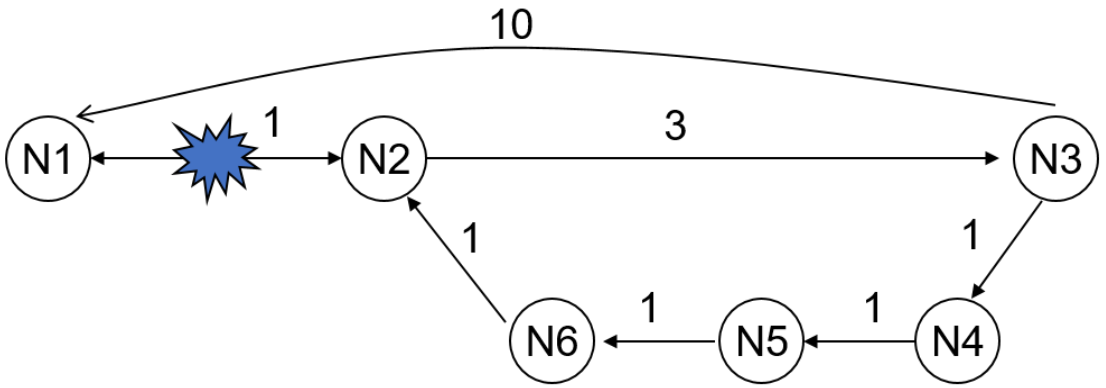


Distance Vector Routing

- Only need to know link state and routing tables of neighbours
- Less sensitive to network perturbations
- Network convergence can take a long time
- Can suffer from instability and routing loops

Routing loops

- Path from N1 to N2 breaks
- N3 updates that it has a path of cost 5 to N1, which is stale information
- N2 updates its routing table with a cost of 8 to N1



Routing loops

- A packet sent will now traverse
  N6→N2→N3→N4→N5→N6→
- The routing costs will slowly climb to infinity
- Use of Distributed Bellman-Ford as-is cannot handle mobility and link changes
- Modified versions are used in practice

Practical Distance Vector Routing

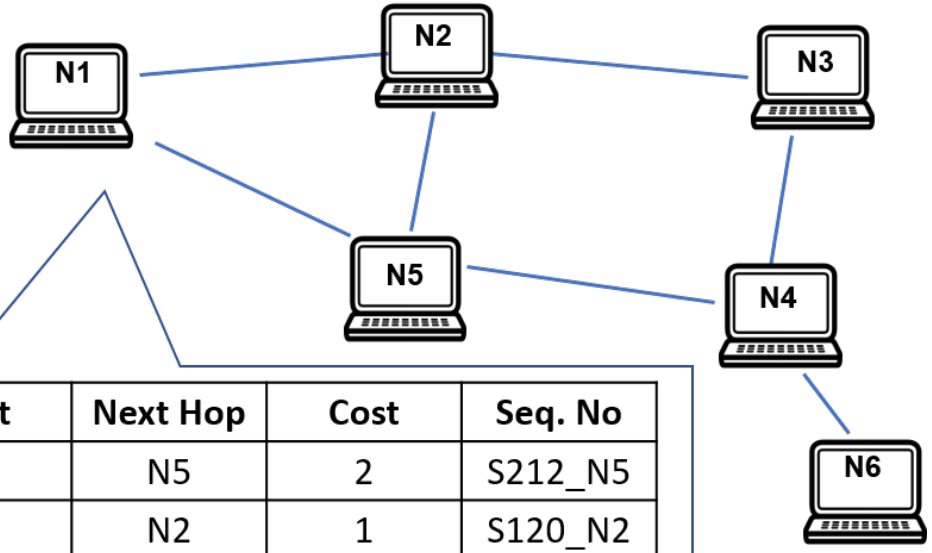
- *Proactive protocols* identify optimal paths to destination nodes in advance, so that they are already there whenever needed e.g. DSDV
- *Reactive protocols* identify optimal paths to destination nodes on-demand (when needed) e.g. AODV
- *Hybrid protocols* use the proactive approach to find paths to nodes within a nearby zone, and the reactive approach to find paths to nodes beyond this zone e.g. ZRP

Destination Sequence Distance Vector[2]

- DSDV is an enhanced version of the Bellman-Ford algorithm
- When nodes send routing table updates, they also include a monotonically increasing sequence number
- Nodes will trigger a routing table update if they identify a failed link in their neighbourhood
- Tables compare the sequence number with their cached sequence number and only update if it is fresher
- This helps to address problems with loops and stale information

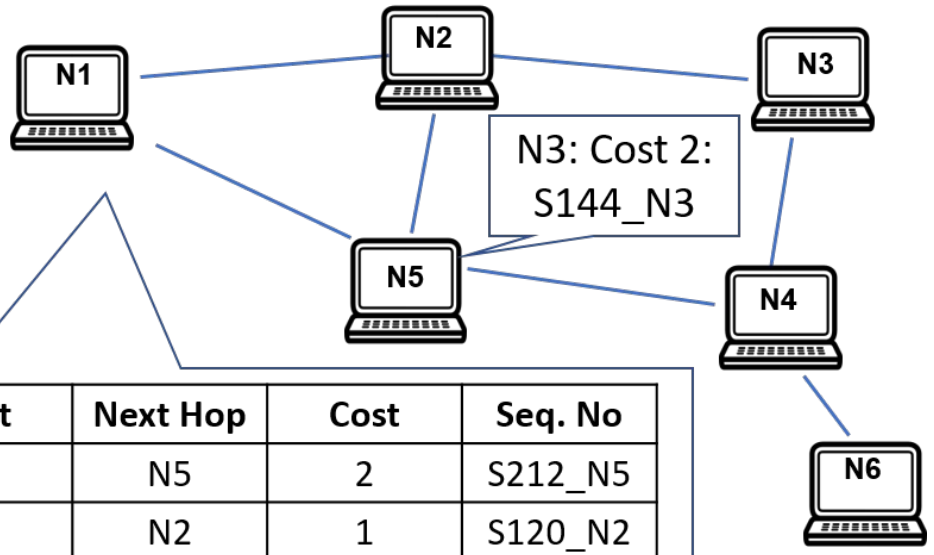
[2]Charles E Perkins and Pravin Bhagwat. "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers". In: *ACM SIGCOMM computer communication review*. Vol. 24. 4. ACM. 1994, pp. 234–244.

DSDV by example



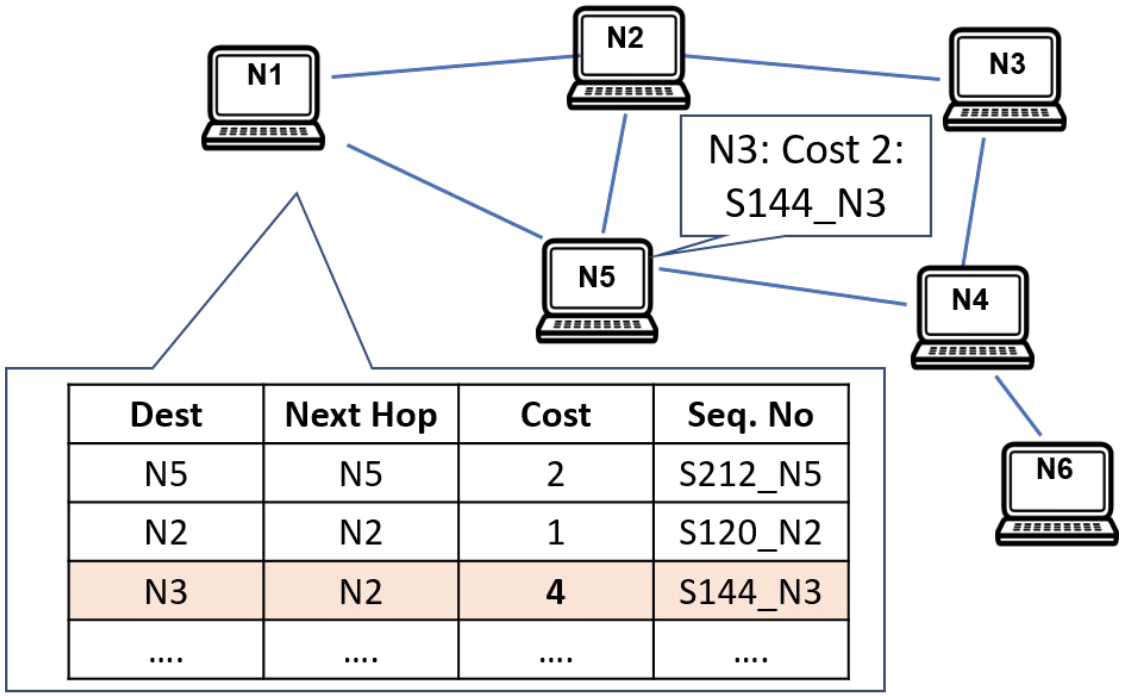
Dest	Next Hop	Cost	Seq. No
N5	N5	2	S212_N5
N2	N2	1	S120_N2
N3	N2	7	S144_N3
….	….	….	….

DSDV by example



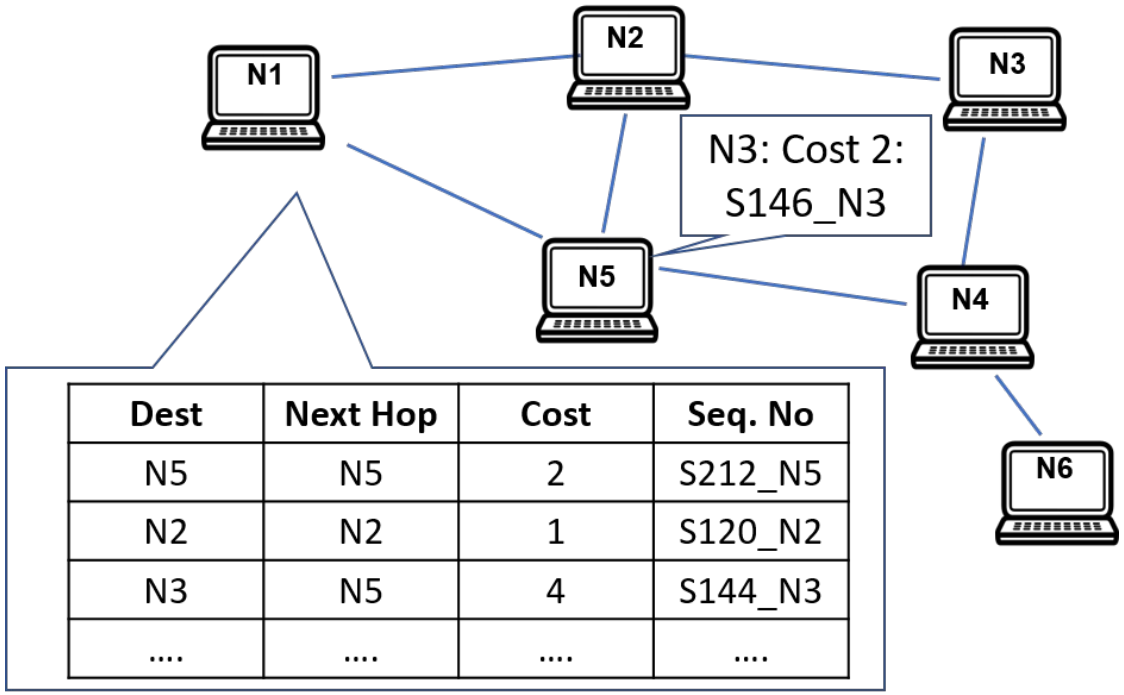
Dest	Next Hop	Cost	Seq. No
N5	N5	2	S212_N5
N2	N2	1	S120_N2
N3	N2	7	S144_N3
....	....	....	....

DSDV by example



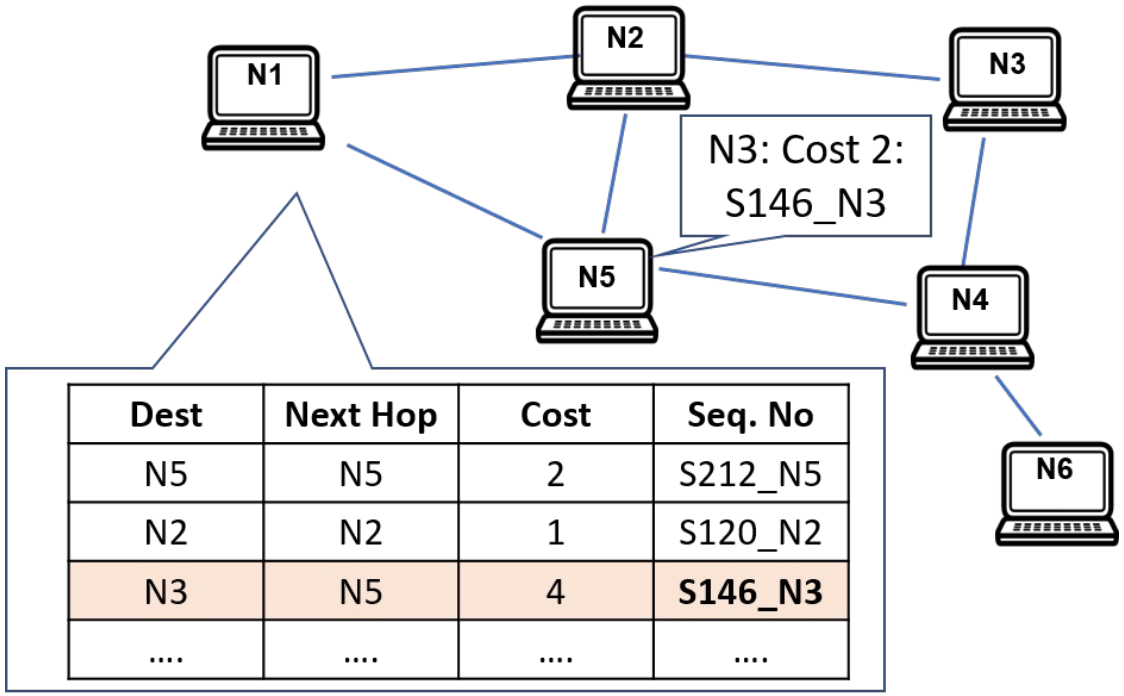
Dest	Next Hop	Cost	Seq. No
N5	N5	2	S212_N5
N2	N2	1	S120_N2
N3	N2	**4**	S144_N3
….	….	….	….

DSDV by example



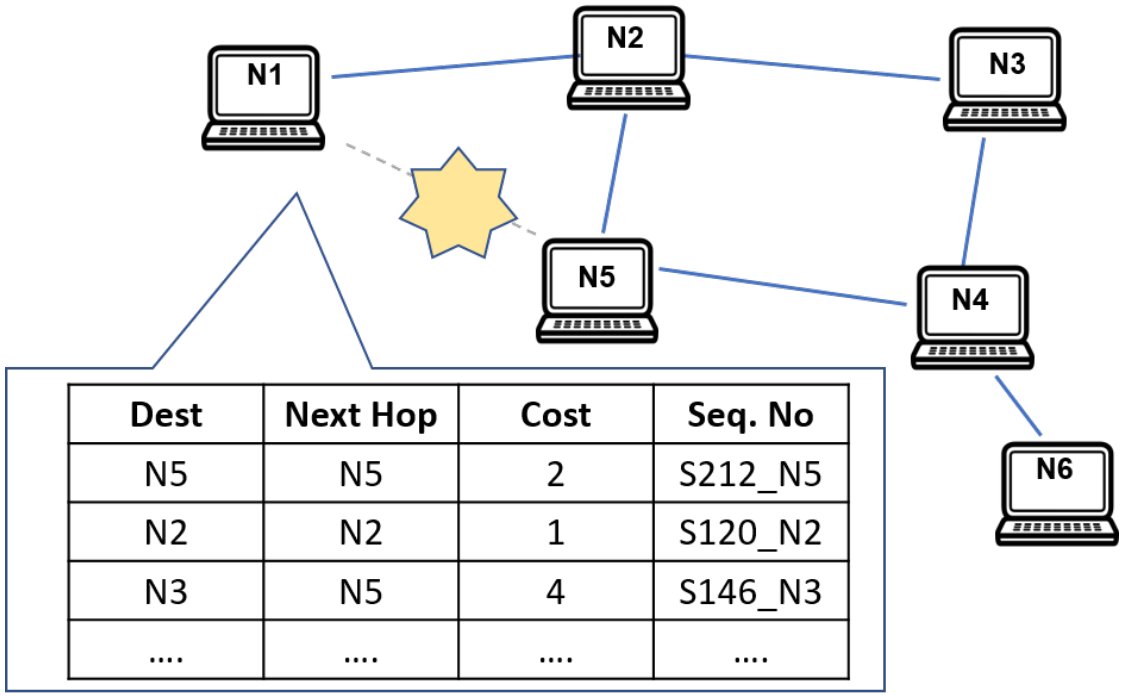
Dest	Next Hop	Cost	Seq. No
N5	N5	2	S212_N5
N2	N2	1	S120_N2
N3	N5	4	S144_N3
….	….	….	….

DSDV by example



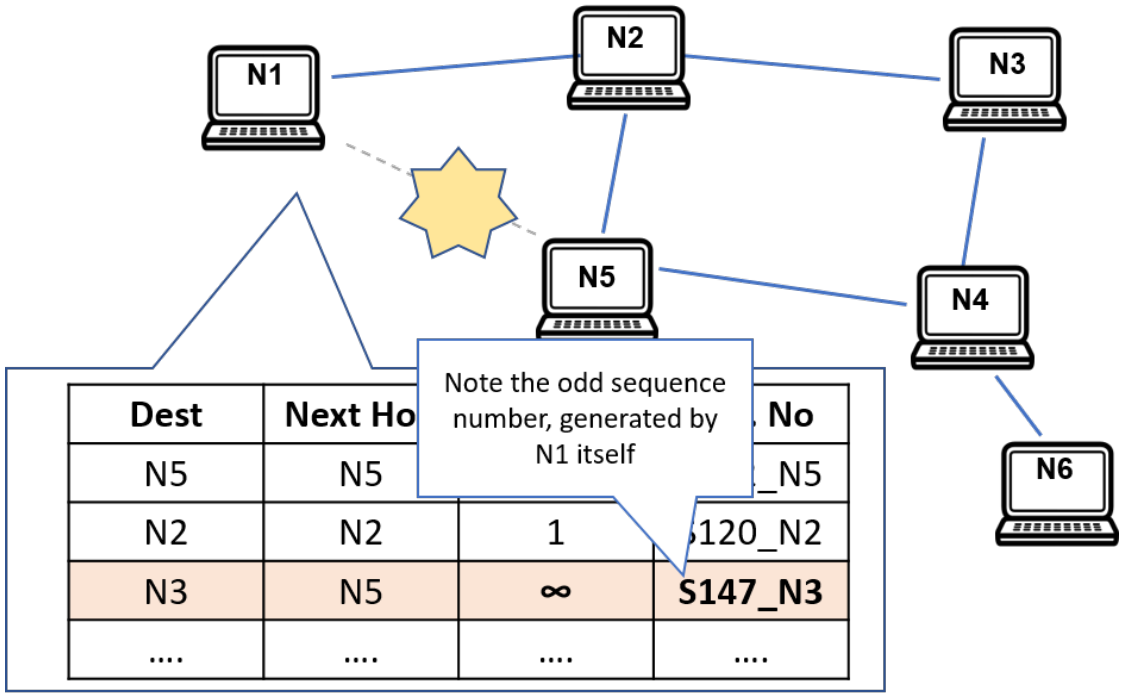
Dest	Next Hop	Cost	Seq. No
N5	N5	2	S212_N5
N2	N2	1	S120_N2
N3	N5	4	**S146_N3**
….	….	….	….

DSDV by example



Dest	Next Hop	Cost	Seq. No
N5	N5	2	S212_N5
N2	N2	1	S120_N2
N3	N5	4	S146_N3
….	….	….	….

DSDV by example



Dest	Next Ho		No
N5	N5		_N5
N2	N2	1	120_N2
N3	N5	∞	S147_N3
….	….	….	….

DSDV

- Handles issues with stale information
- Does not scale well to very large networks, as routing tables for all destinations need to be maintained
- As tables grow in size, they consume more network overhead

Routing and Networking

Resource-constrained routing

Resource-constrained routing

- Nodes may be energy or computationally limited
- Need more efficient techniques for routing data
- May not have space or resources to store entire routing tables

Mesh networks

- Densely connected networks with little or no mobility
- Data travels over multiple hops to save energy
- Long-lived, low-power networks

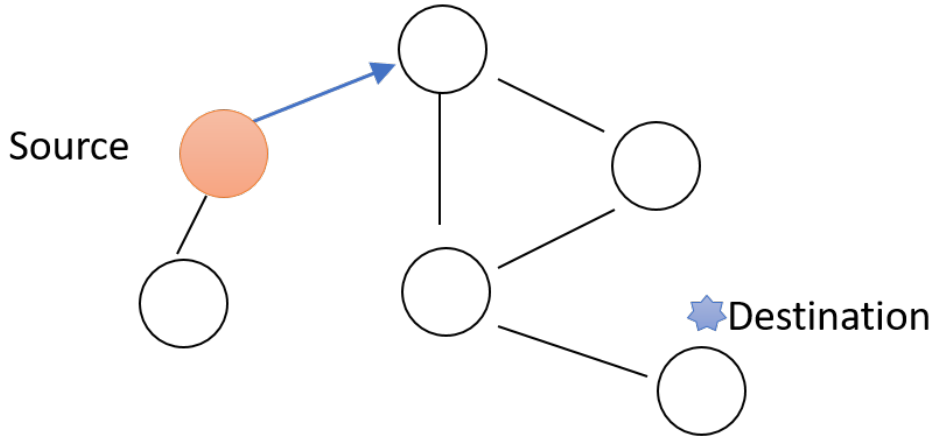
Geographical routing

- Rather than sending data to a particular destination by ID, data can be sent by location
- At each hop, the aim is to get (physically) closer to the target location
- Assume all nodes know their location
- Assume all nodes know their one-hop neighbours

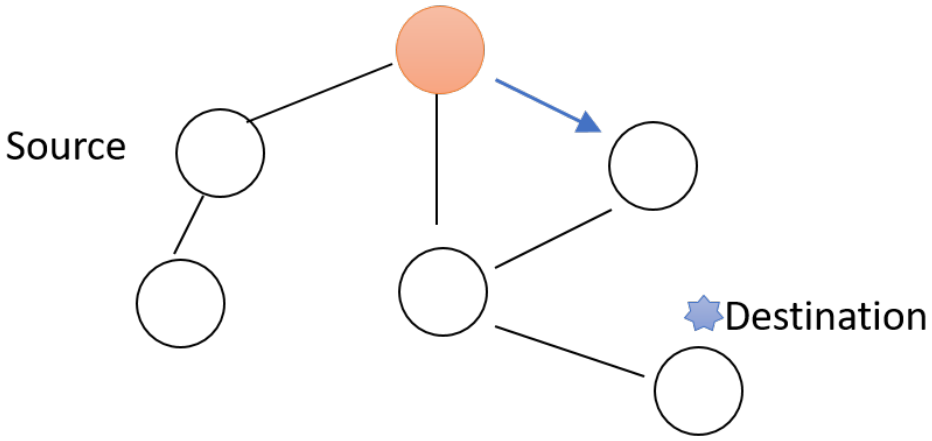
Geographical routing

- Simplest approach: greedy forwarding
- At each hop, forward to a neighbour which is closer to the destination

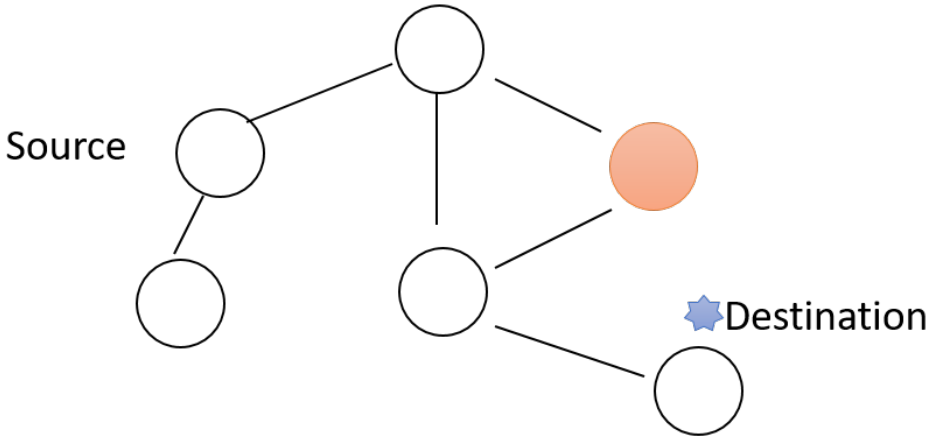
Greedy forwarding



Greedy forwarding



Greedy forwarding

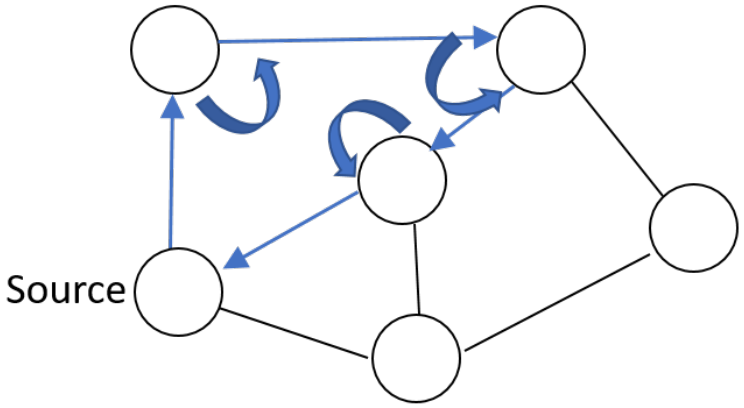


Face routing

- Greedy geographical forwading will get trapped in voids or local minima
- An alternative to greedy routing is to use *face routing*
- Face routing works correctly on planar graphs, that is, graphs with no crossing edges
- Face routing uses two primitive operations to route
  - Right-hand rule
  - Face changes

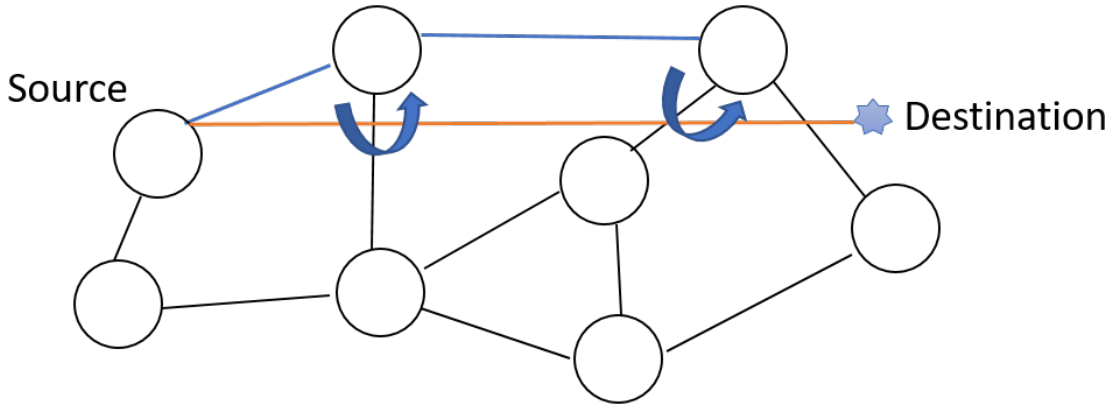
Face routing: Right-hand rule

- Right-hand rule - send to next edge counter-clockwise relative to incoming edge



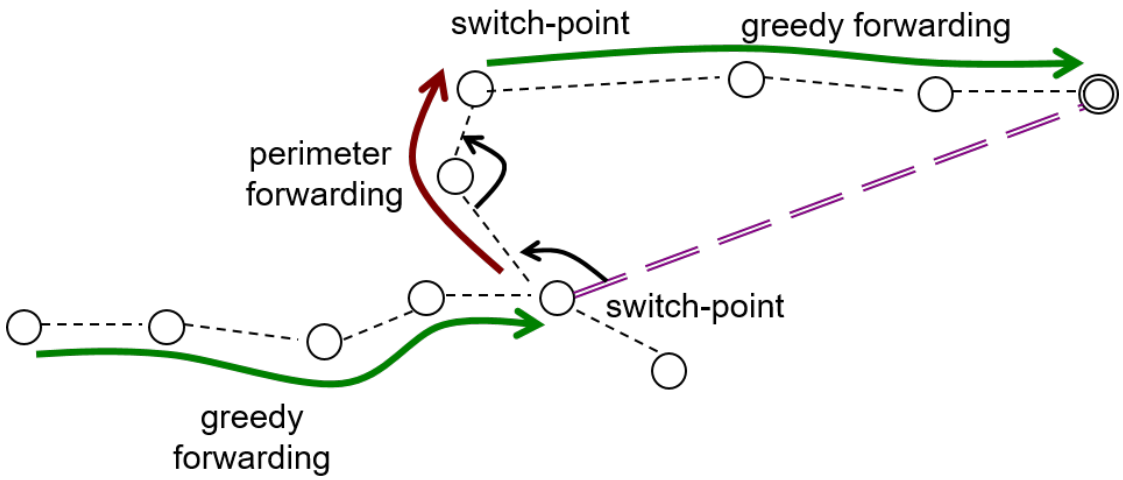
Face routing: Face changes

- A virtual line is drawn from source to destination
- Any edge which crosses this line switches the primitive from right-hand rule to face traversal
- Change to next face if this occurs



Greedy Perimeter Stateless Routing (GPSR)

- GPSR[4] combines greedy routing with face routing when local minima are encountered



[4]Brad Karp and Hsiang-Tsung Kung. "GPSR: Greedy perimeter stateless routing for wireless networks". In: *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM. 2000, pp. 243–254.

Geographical Routing: Overview

- Geographical routing is a way of routing without explicitly constructing a routing tree
- Decisions are taken locally based on co-ordinates of neighbouring nodes
- Handles node mobility elegantly
- Does require location of all nodes to be known

Routing Tree: Collection Tree Protocol[5]

- Degenerate form of a mesh network with a single destination
- All data concentrates at a single point (basestation)
- Basestation has a cost/metric of 0
- Nodes don't need to know basestation ID beforehand

[5]Omprakash Gnawali et al. "Collection tree protocol". In: *Proceedings of the 7th ACM conference on embedded networked sensor systems*. ACM. 2009, pp. 1–14.

Routing Tree Construction

- Basestation initiates construction by broadcasting its hop-count (0)
- Nodes increase hop-count and rebroadcast beacon
- Nodes choose a parent with lowest distance to root
- Increase hop-count if parent beacon is not received within timeout period

Routing tree metrics

- Different costs can be used
  - Hops to root
  - Remaining node energy
  - Link reliability
  - Some weighted combination of the above

Do we even need routing protocols?

- Glossy[9] is an emerging approach which does away with explicit routing protocols
- It achieves reliability of 99.99% for flooding data through a network
- Glossy relies on constructive interference through simultaneous rebroadcasts
- By exploiting the properties of the PHYsical layer, it provides an alternative network primitive

[9]Federico Ferrari et al. “Efficient network flooding and time synchronization with glossy”. In: *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*. IEEE. 2011, pp. 73–84.

Comparison of resource-constrained approaches

- In dense or resource-constrained networks, the identity of the node is often not as important as:
  - Location (geographical routing)
  - Distance from root (collection tree)
- These are akin to semantic overlays, where node identity is irrelevant