Distributed Systems

Mobile & Sensor Computing

Rik Sarkar

University of Edinburgh
Fall 2020

Mobile and Ubiquitous computing

- Devices (computers) are carried by people (mobile)
  - Laptops, phones, watches …

- They are everywhere
  - Carried by people (mobile)
  - Embedded in the environment
    - Coffee machines, cameras, sensors for light control, elevators…
  - Produce large amounts of data
    - Usage, sensing…

Ubiquitous

- Advantages:
  - There are computers everywhere
  - Everything is “smart”
  - Potentially use collaborative distributed computations on these to make them even smarter

- Challenges:
  - There are more things to go wrong
  - Not easy to make things work well coherently
  - Consistent platforms for managing ubiquitous devices do not exist (yet)
  - Devices do not interoperate easily

Mobile

- Advantages:
  - The same device is carried by the person – easy to give consistent service
  - Information whenever, wherever they need
  - Devices have sensors – potential for sensing the environment and adapting
- Disadvantages:
  - Connectivity is challenge: data is costly; network does not work the same way; mobility interferes with comunication
  - Limited battery: can’t do too much communication
  - How to make use of sensors, not so well understood

Context aware computing

- Adapt computations to the circumstances
  - Time of day
  - Is the user present?
  - Is the phone in hand or in pocket
  - Scan for wifi only when *indoors*
  - Turn off ring when in *cinema, meeting…*
  - Recognize activity and bring up relevant information
  - …

Context aware computing

- Adapt computations to the circumstances
- Basic contexts are easy to identify, but it is not always clear how to adapt
  - Turn down volume at night… but what if it is an important call?
- Many contexts are very hard to detect reliably

Context detection examples

- Detect if user is indoor or outdoor
- Use sound to detect user in a meeting
- Detect transport mode (walking, car, bus, tram..)
  - Using accelerometer
- Detect presence of other users nearby from wifi activity

Context detection

- Generally hard
- Concerns about privacy: you do not want to send context information to a server
- Perhaps distributed computation can help
  - Use data from many phones to detect context
  - But again, do not want to send all data to server
  - Do as much of it as possible on device – filter/process data at source

Networking in mobile systems

- Difficulty:
  - The network graph changes
  - A node is not always connected to the same router

- Example system: Mobile ad-hoc networks
  - Ad-hoc: Unplanned
  - Devices simply connect to nearby devices and route packets
  - Also applies to sensor networks

Routing in ad hoc wireless networks

- Find route between pairs of nodes wishing to communicate.
- Proactive protocols: maintain routing tables at each node that is updated as changes in the network topology are detected.
  - Heavy overhead with high network dynamics (caused by link/node failures or node movement).
  - Not practical for networks that change frequently

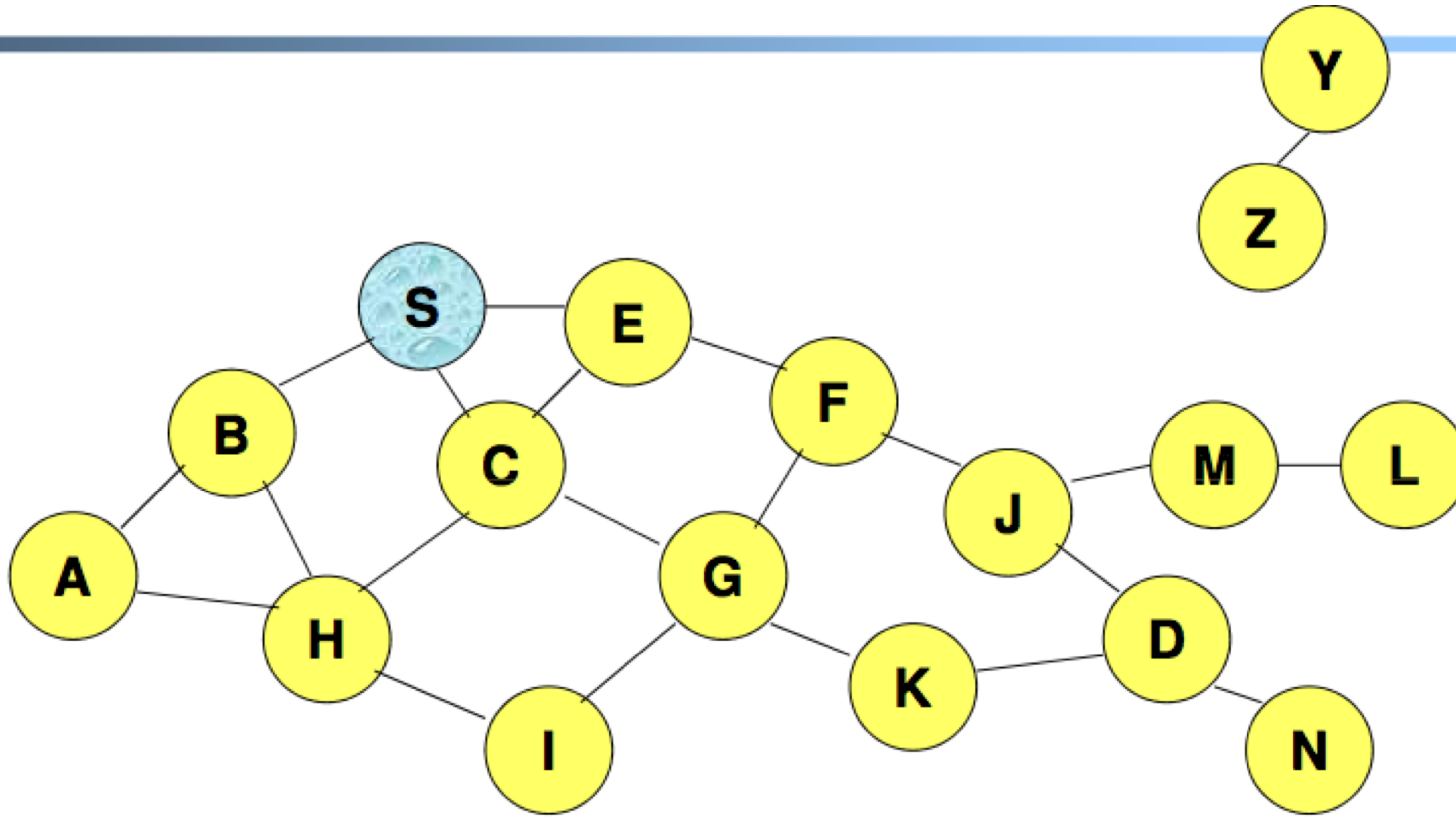
Routing in ad hoc wireless networks

- Reactive protocols: routes are constructed on demand. No global routing table is maintained.
- More appropriate for networks with high rate of changes
  - Ad hoc on demand distance vector routing (AODV)
  - Dynamic source routing (DSR)

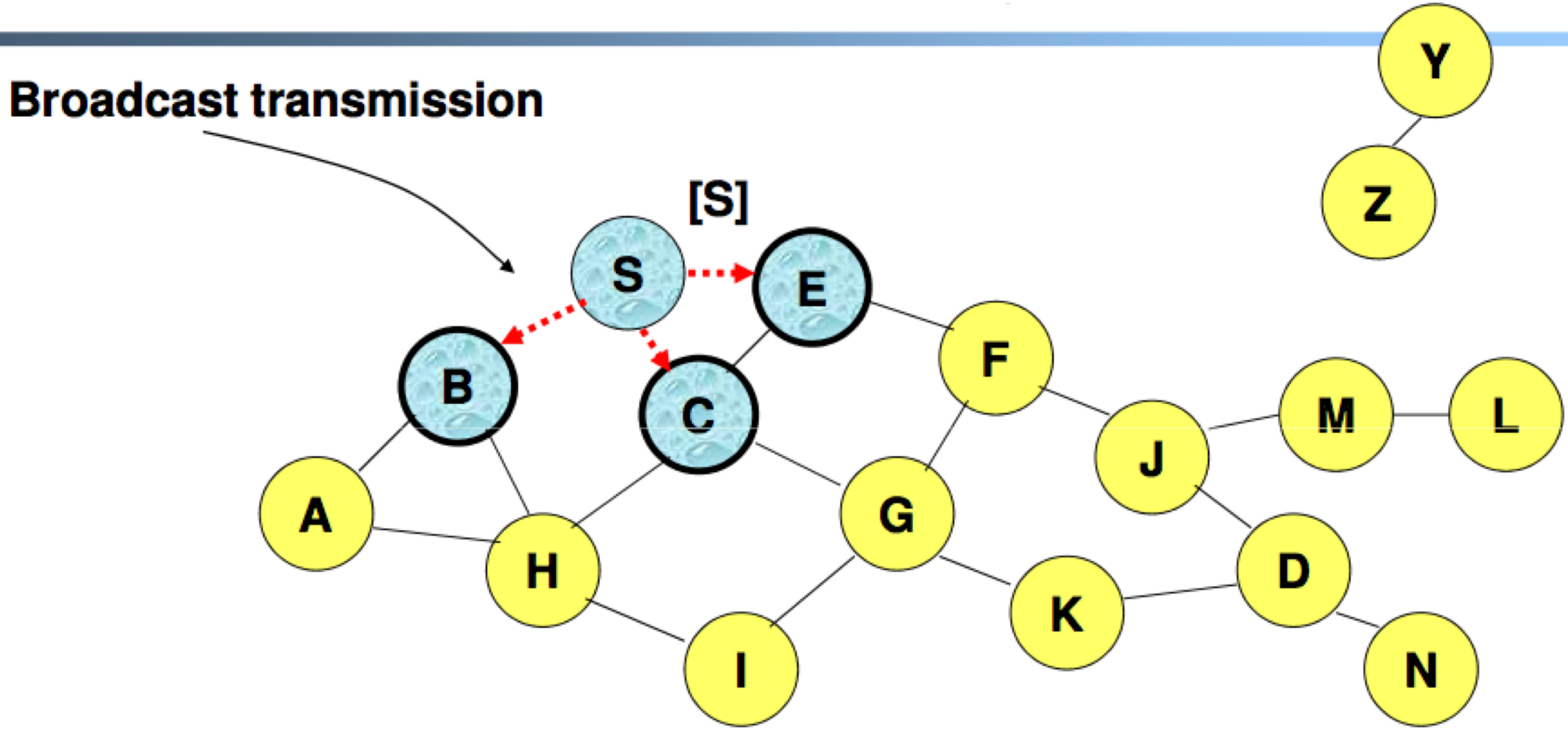
Dynamic Source Routing (DSR)

- Node S wants to send a message to node D
- S initiates a a route discovery
- S floods the network with route request (RREQ) message
- Each node appends its own id to the message

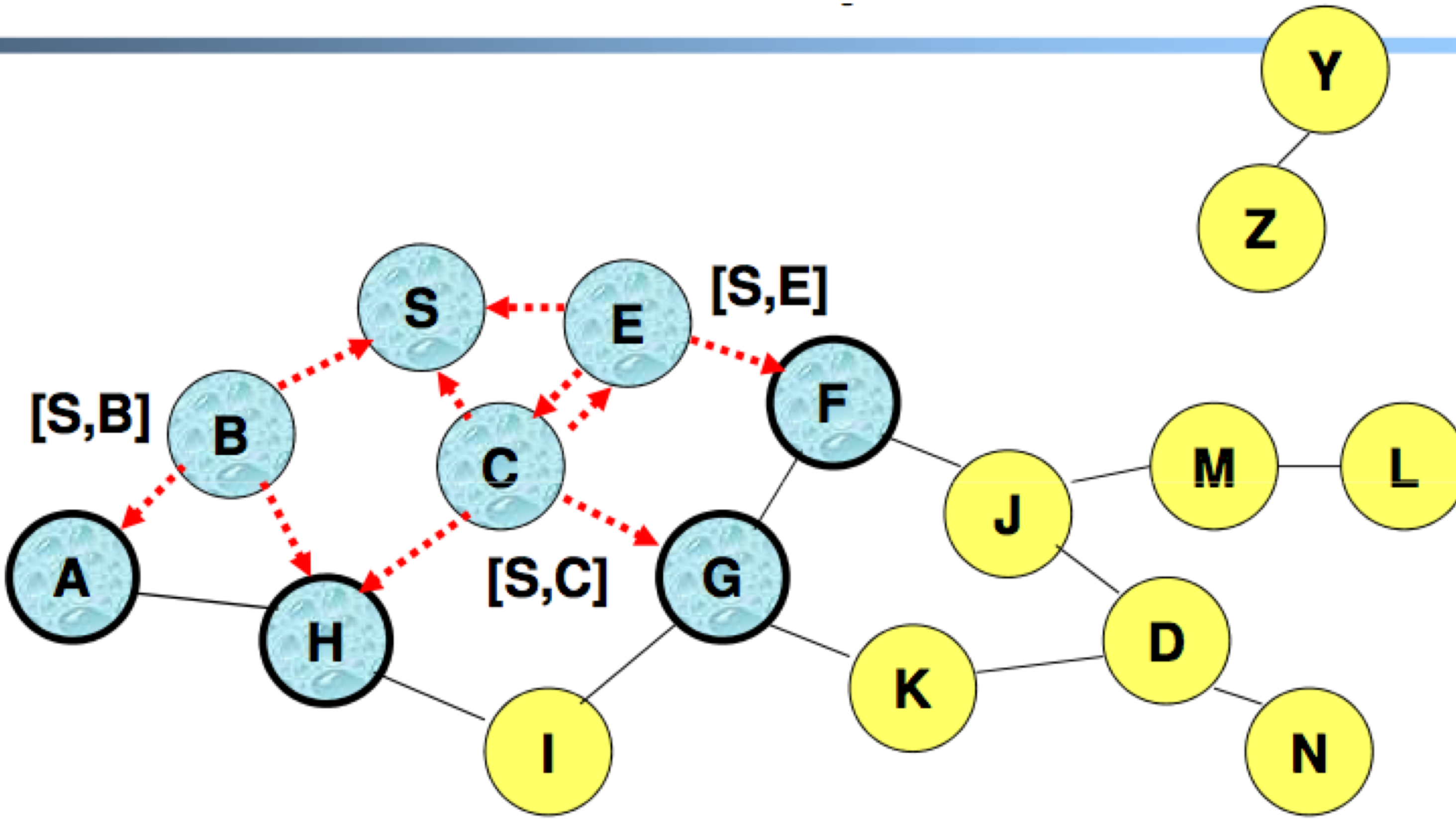
Route Discovery: RREQ



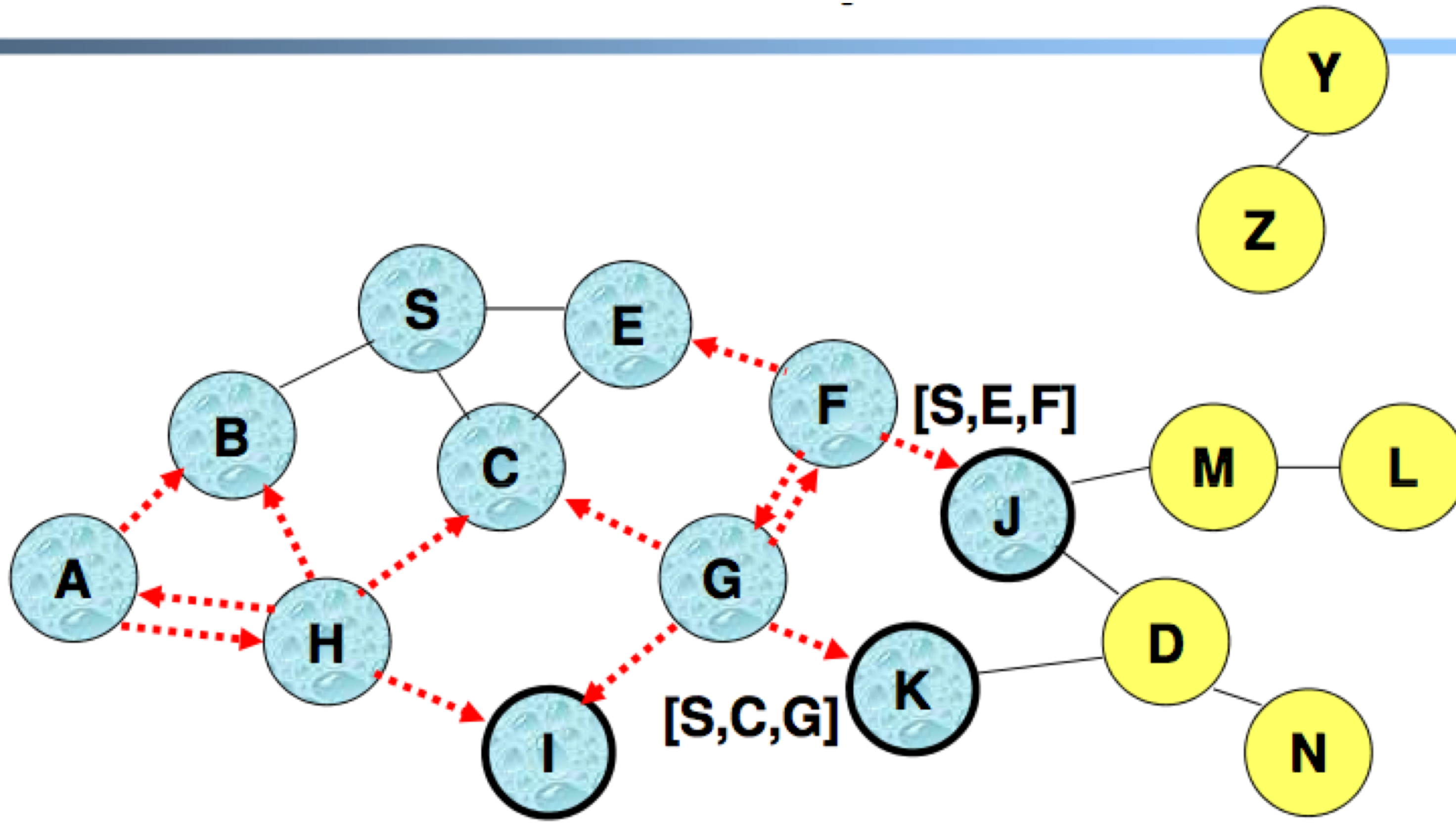
Route Discovery: RREQ



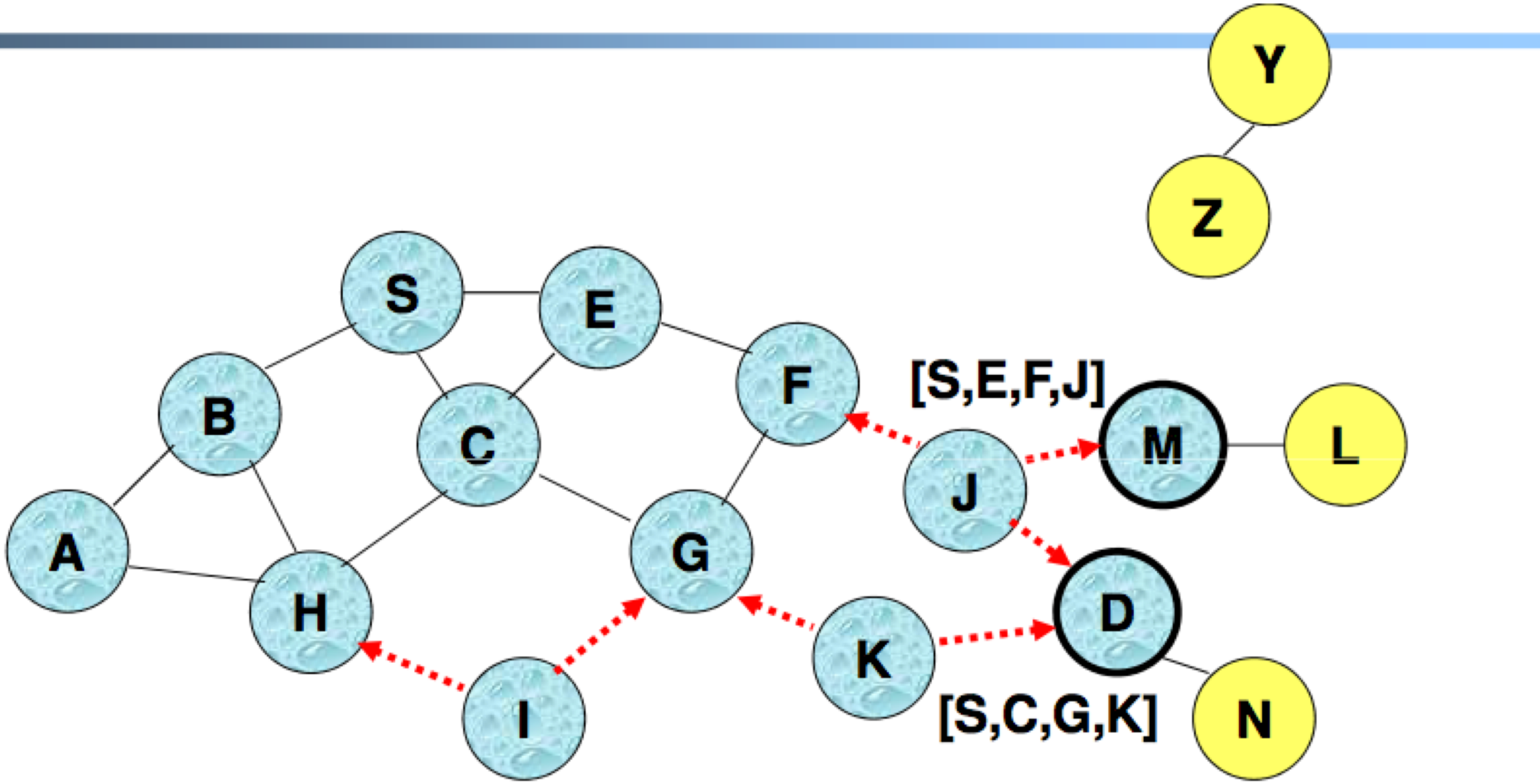
Route Discovery: RREQ



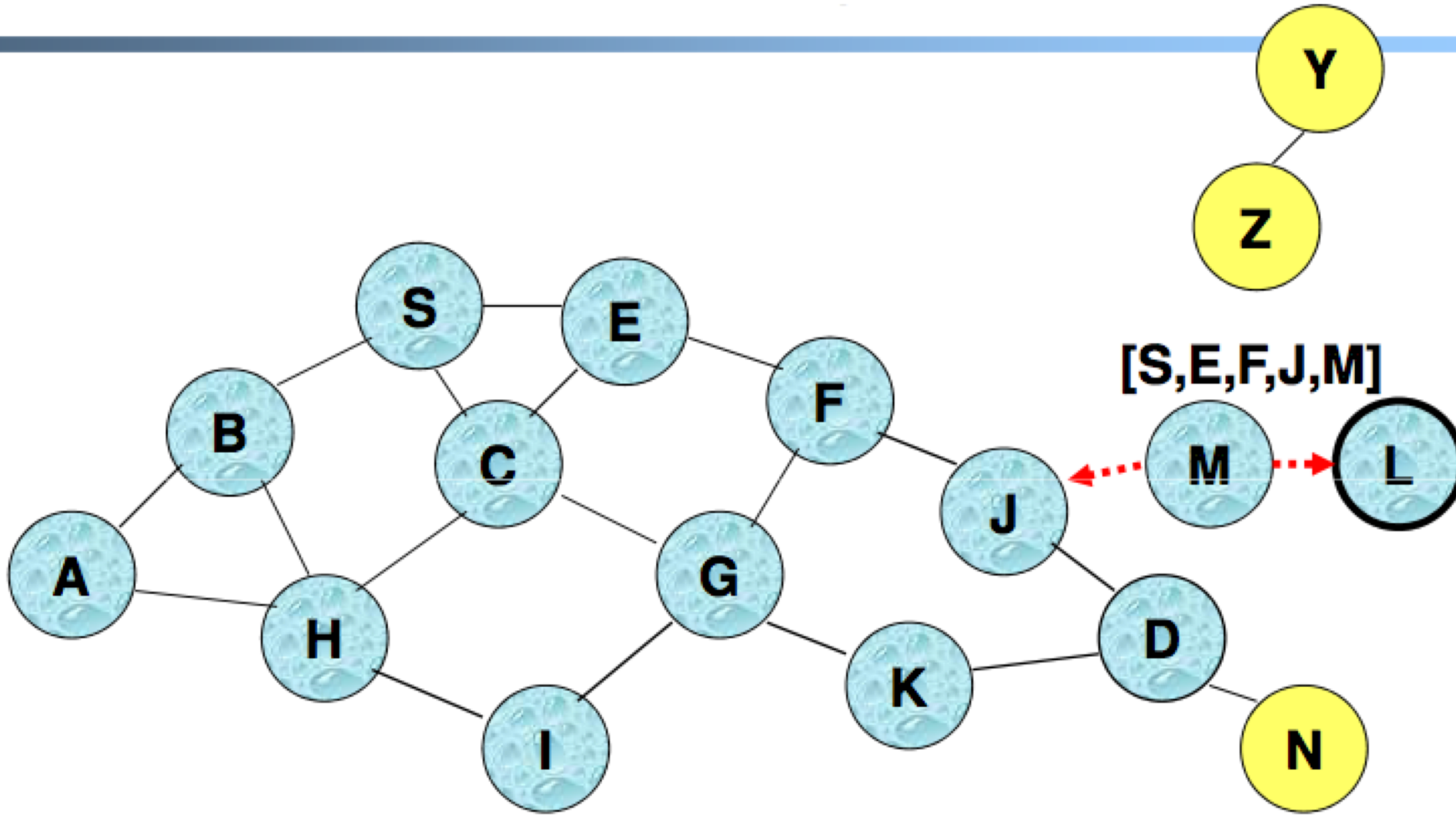
Route Discovery: RREQ



Route Discovery: RREQ



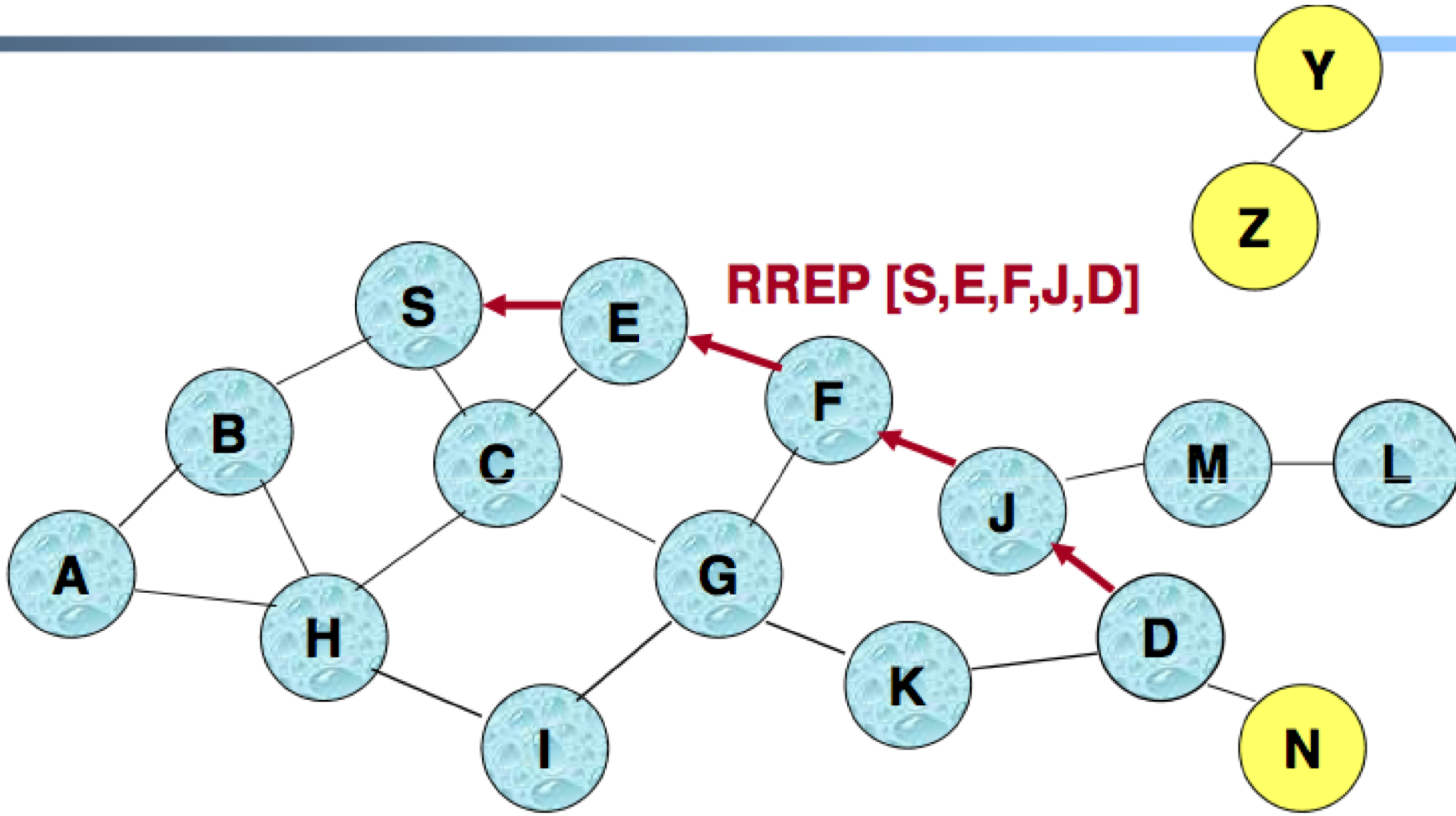
Route Discovery: RREQ



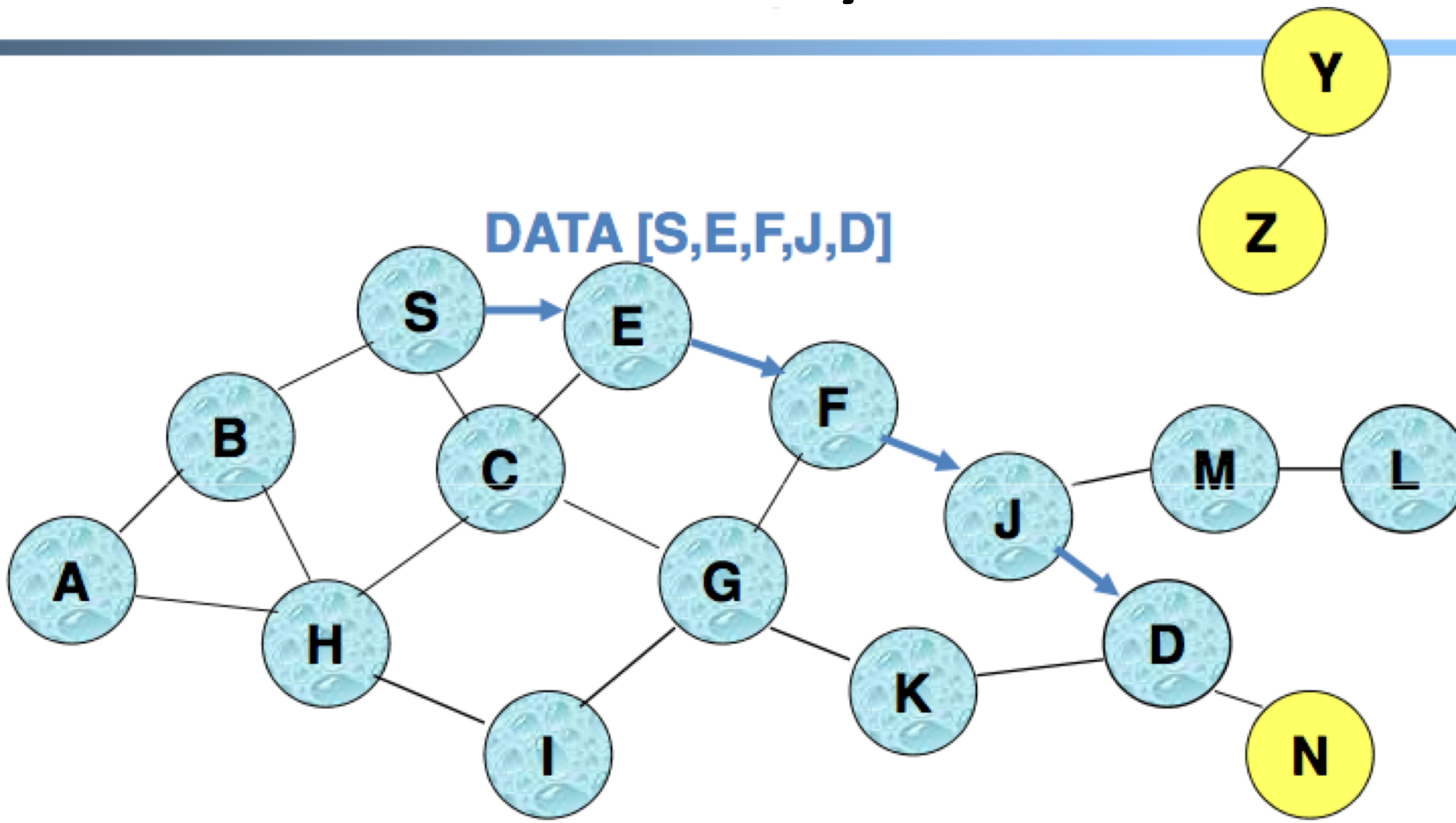
Route Discovery in DSR

- Destination D on receiving the first RREQ sends a *route reply* (RREP)
- RREP is sent on a route obtained by reversing the route in received RREQ

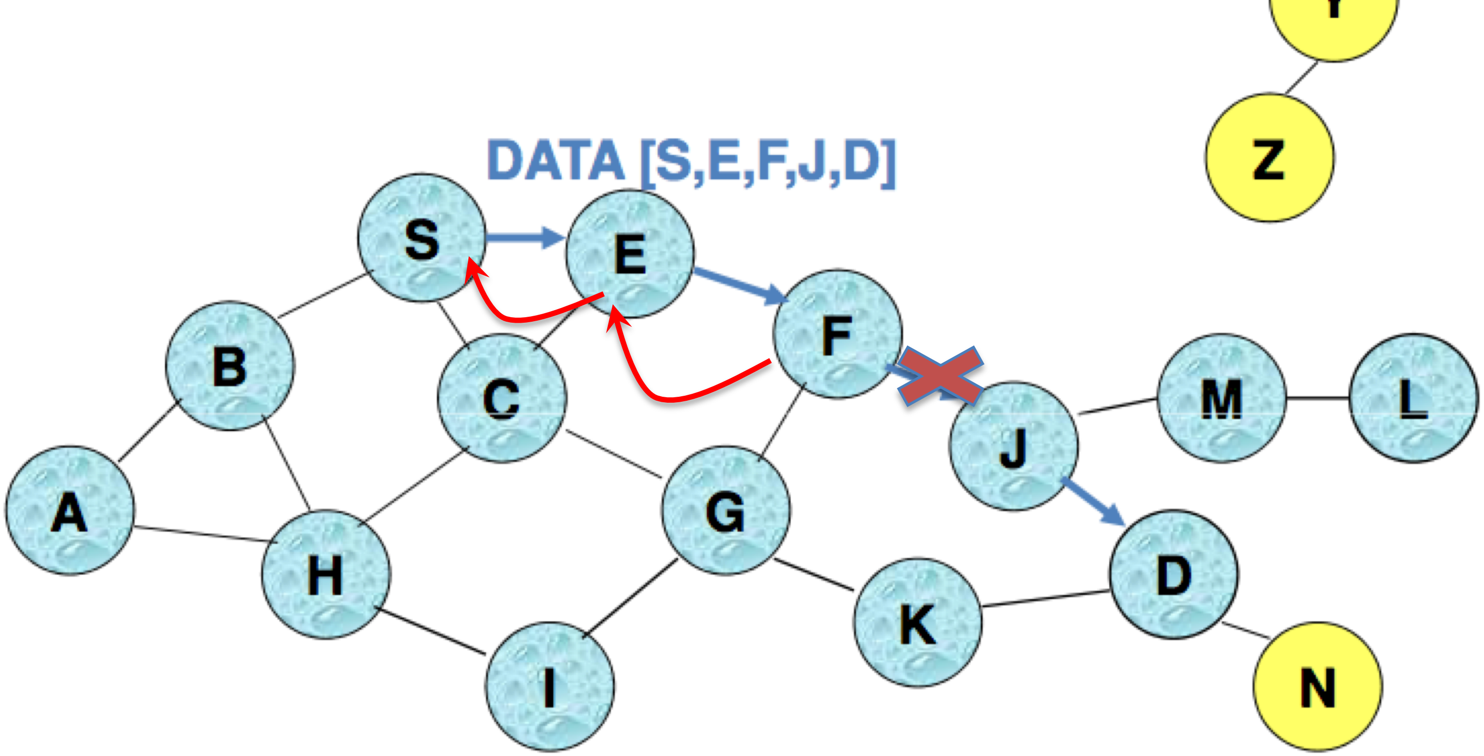
Route Discovery: RREQ



Route Discovery: RREQ



When node S sends a data packet to D, the entire route is included in the packet header, hence the name source routing



- When a link fails, an error message with the link name is sent back to S.
- S deletes any route using that link and starts discovery.

Route caching

- When a node receives or forwards a message, it learns routes to all nodes on the path
- Advantage:
  - S may not need to send RREQ
  - Intermediate node on receiving RREQ, can respond with complete route
- Disadvantage:
  - Caches may be stale: S tries many cached routes before starting a discovery. Or, intermediate nodes return outdated information.

DSR: Summary

Advantages:

- Routes computed only when needed – good for changing networks
- Caching can make things efficient
- Does not create loops

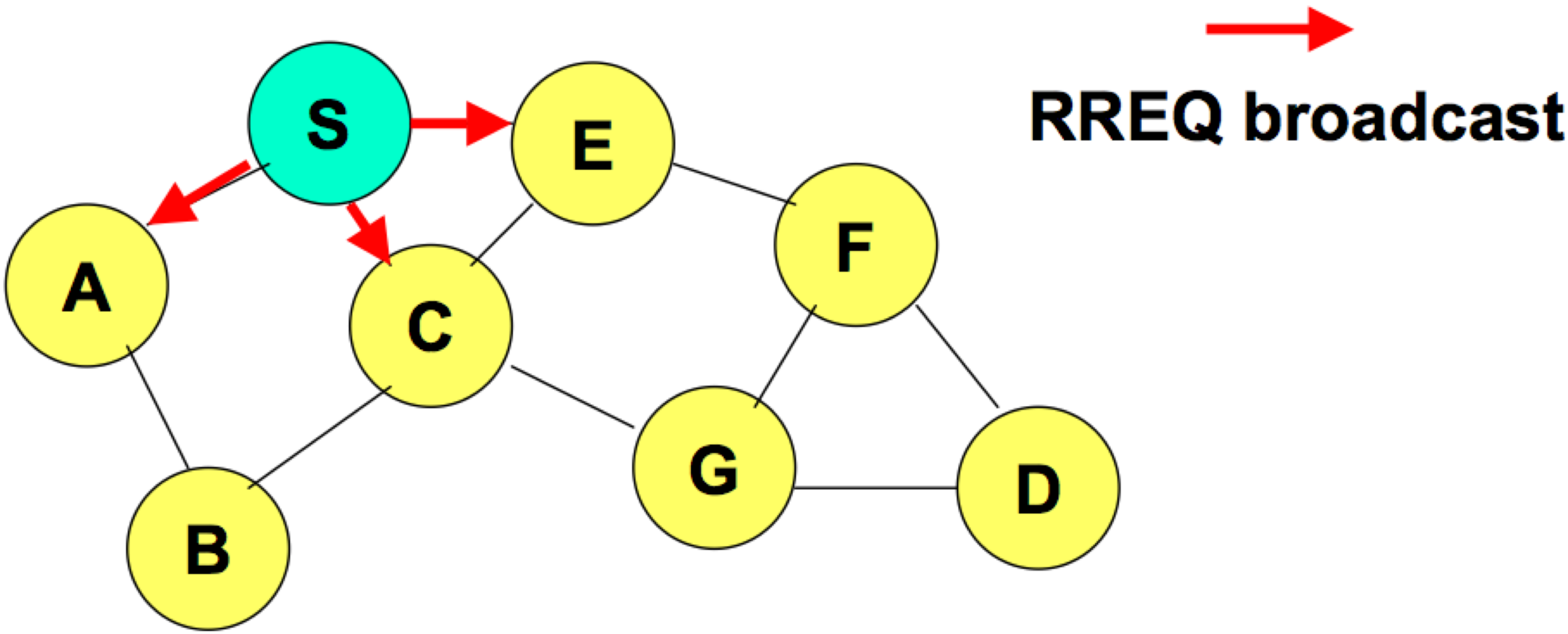
Disadvantages

- Entire route must be contained in message: can be long for large networks
- Flooding causes communication to many nodes
- Stale caches can be a problem
- Not suitable for networks where changes are too frequent

Ad hoc On-Demand Distance Vector Routing (AODV)

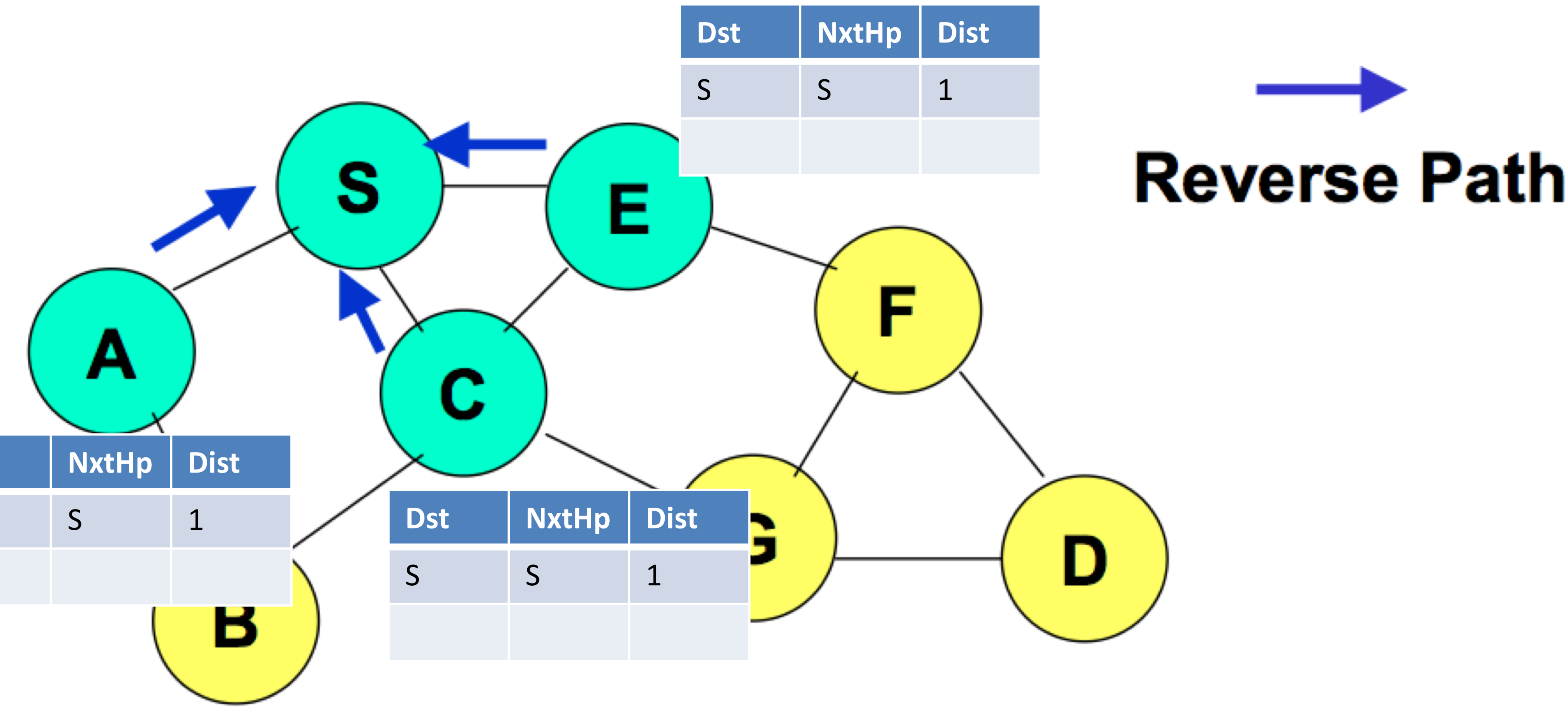
- Maintains routing tables at nodes so that the route need not be stored in the message
- No Caches: Only one route per destination

AODV Route Discovery



- Source floods the network

AODV Route Discovery



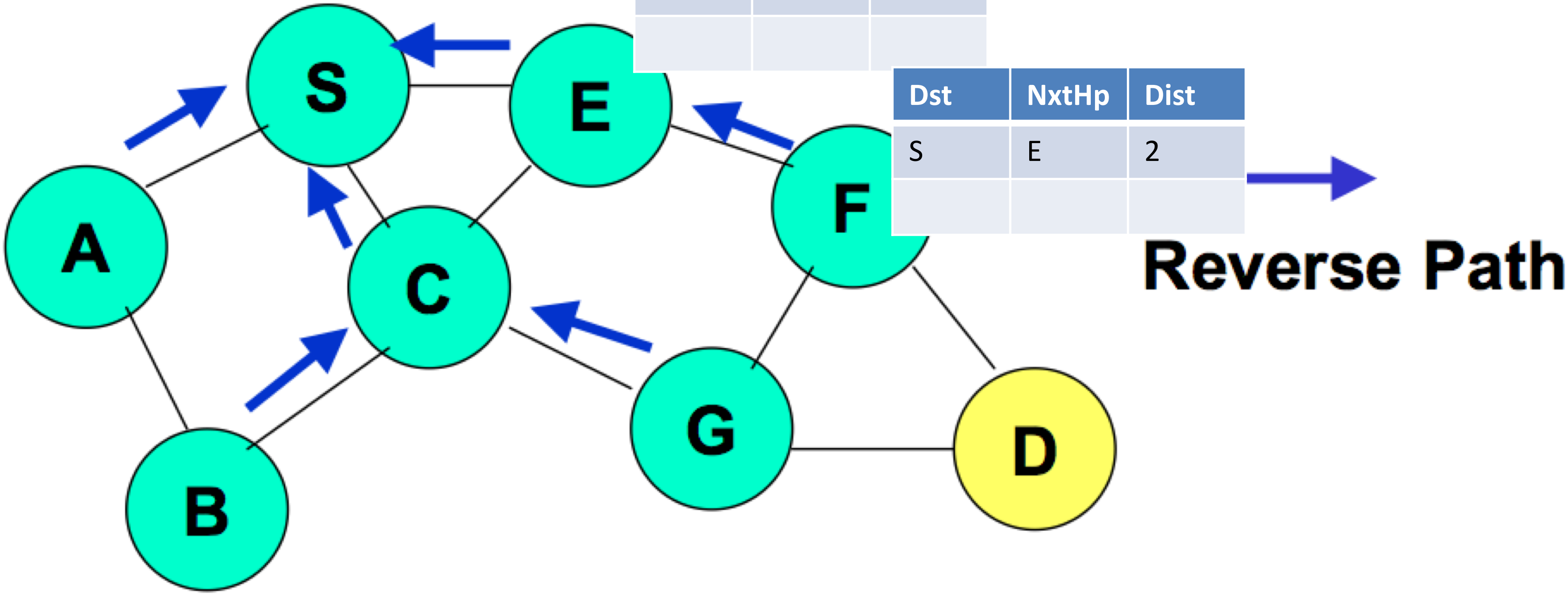
Dst	NxtHp	Dist
S	S	1

NxtHp	Dist
S	1

Dst	NxtHp	Dist
S	S	1

- Other nodes create parent pointer
- A node forwards a RREQ only once

AODV Route Discovery

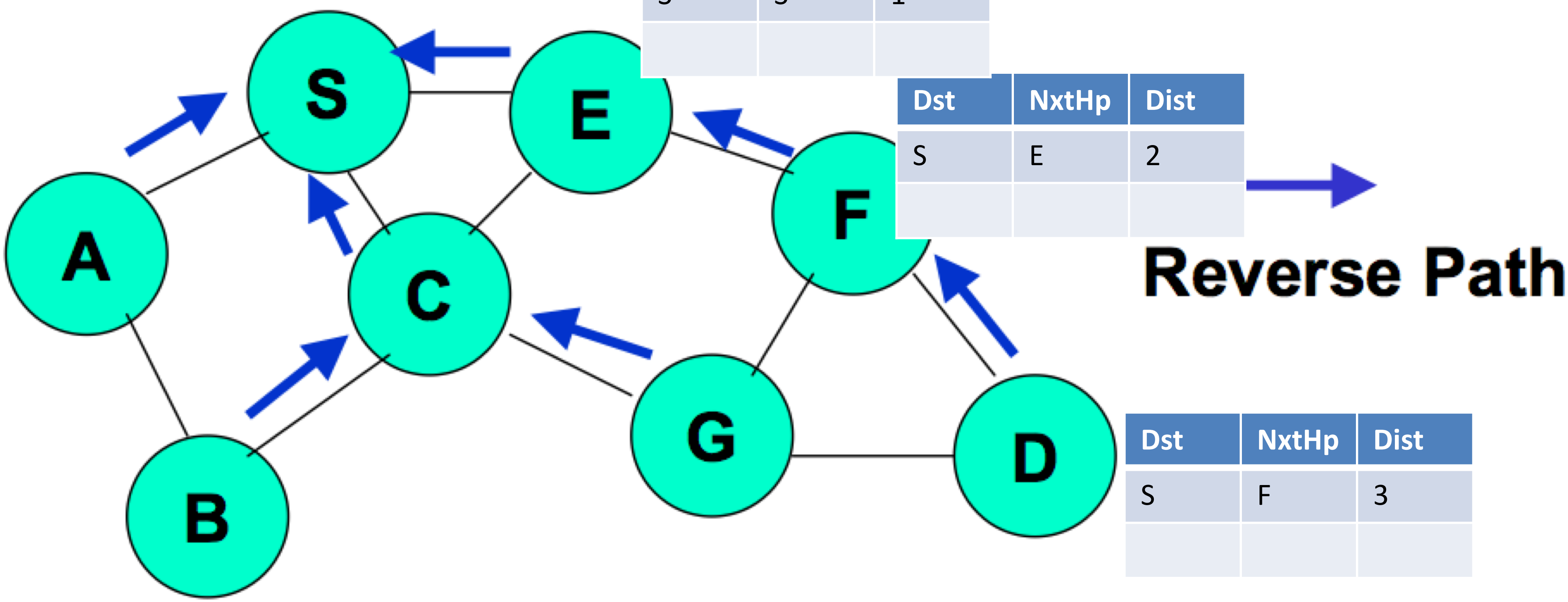


Dst	NxtHp	Dist
S	S	1

Dst	NxtHp	Dist
S	E	2

- Other nodes create parent pointer
- A node forwards a RREQ only once

AODV Route Discovery



Dst	NxtHp	Dist
S	S	1

Dst	NxtHp	Dist
S	E	2

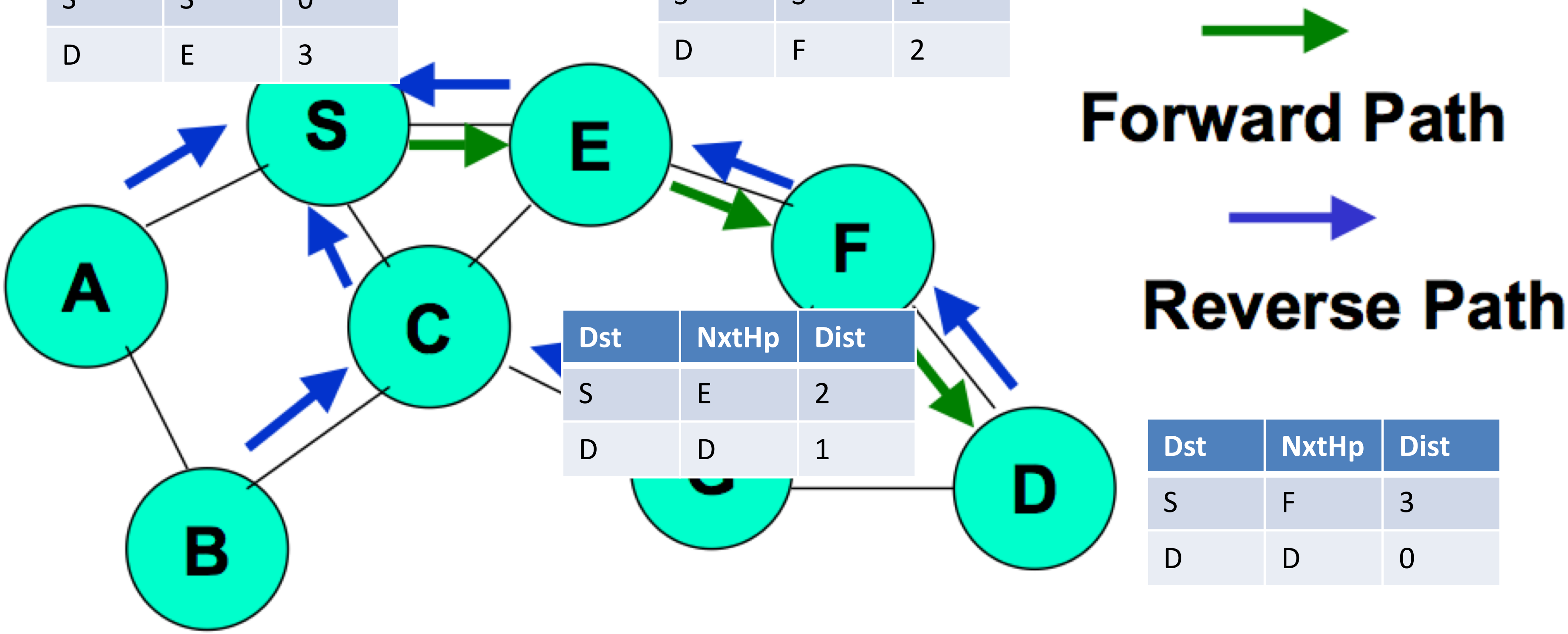
Dst	NxtHp	Dist
S	F	3

- RREP is forwarded via reverse path

AODV Route Discovery

Dst	NxtHp	Dist
S	S	0
D	E	3

Dst	NxtHp	Dist
S	S	1
D	F	2



Dst	NxtHp	Dist
S	E	2
D	D	1

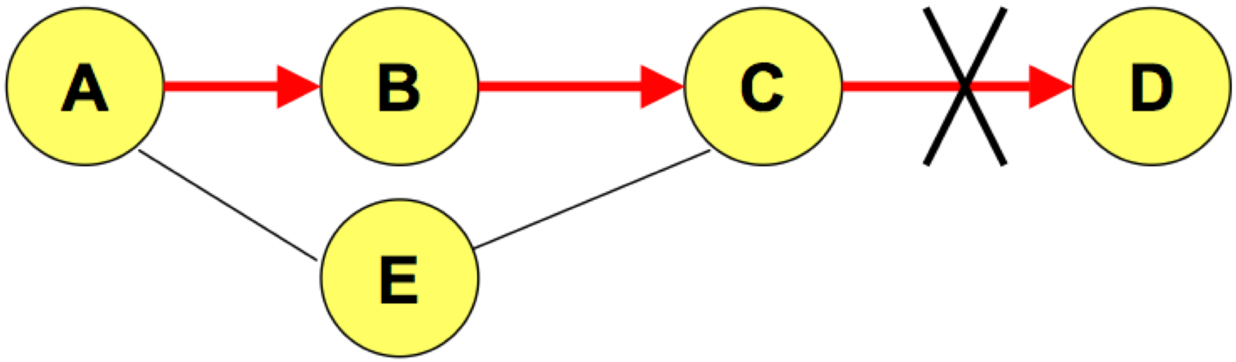
Dst	NxtHp	Dist
S	F	3
D	D	0

- RREP is forwarded via reverse path
- Creates a forward path

Route expiry

- A path expires if not used for a certain time.
- If a node sees that a routing table entry has not been used by this time, it removes this entry
- Even if the path itself is valid
- Good for networks with frequent changes
- Bad for static and stable networks

Can create loops

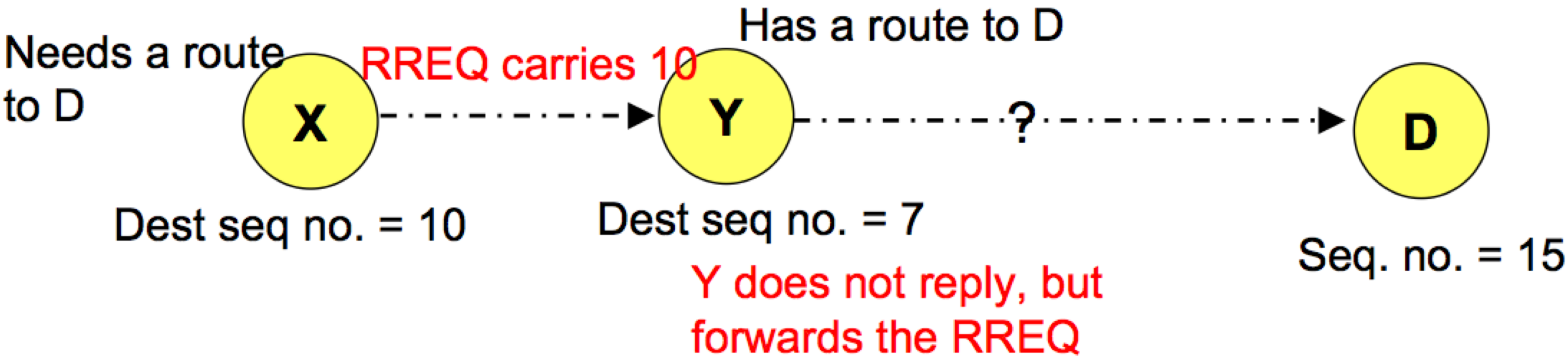


- Assume C->D link has failed, but A does not know because the ERR message was lost
- C is now trying to find path to D
- A responds since A thinks it has a path
- Creates loop: C-E-A-B-C

Sequence numbers in AODV

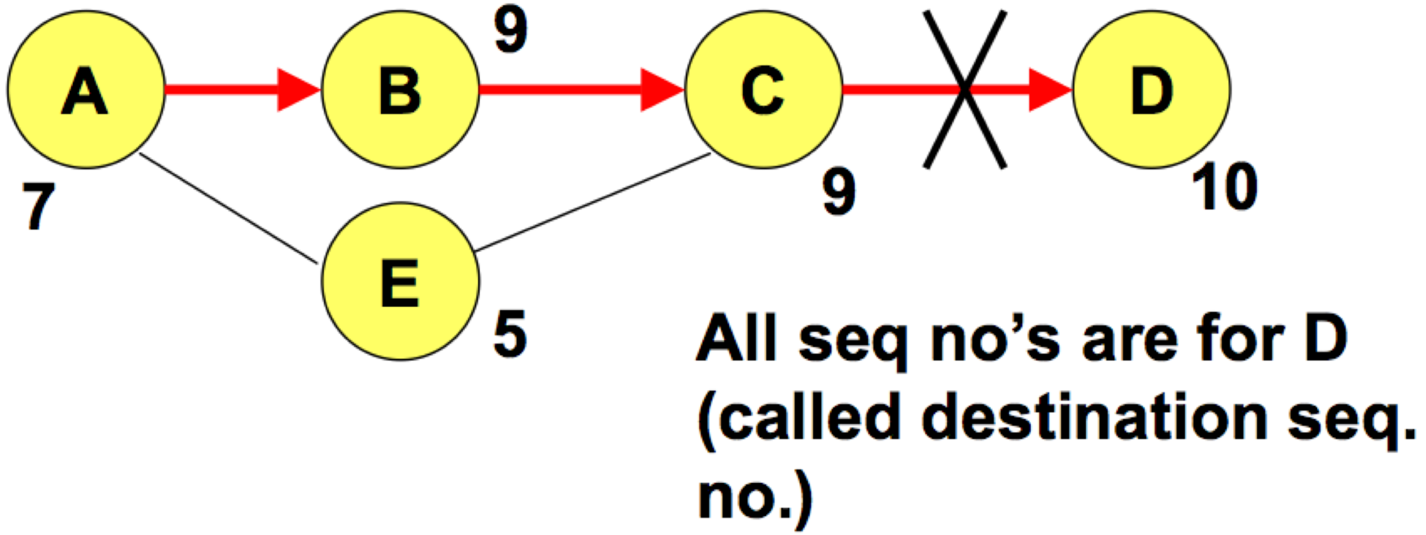
- If A has a route to D, A keeps a sequence number.
- A increments this number periodically: tells how old the information is

Using sequence numbers



- Rule : sequence number must increase along any route

Sequence number rule avoids loop



- A does not reply, since its sequence no. is less than that of C

AODV

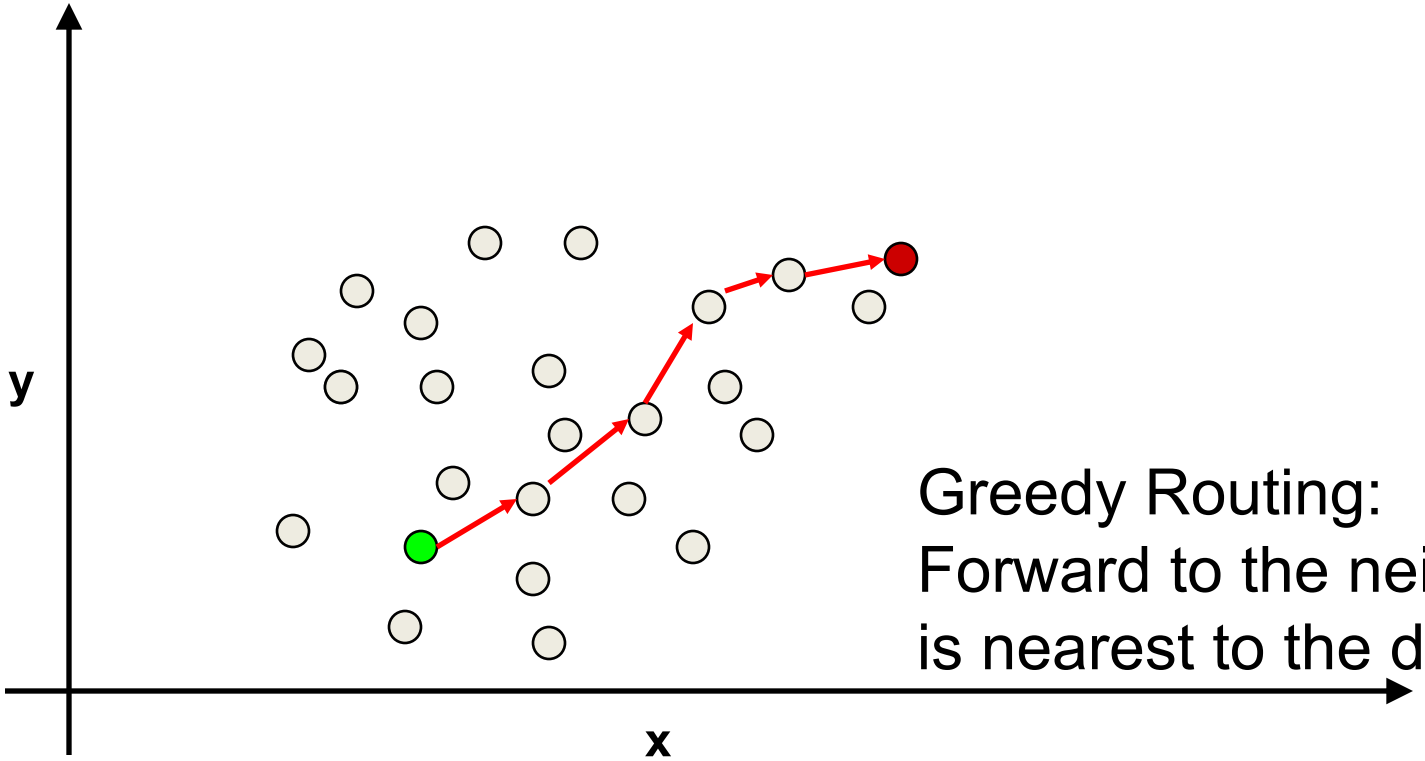
- Routing tables, message does not contain route
- Fresh routes preferred
- Old unused routes expire
- Stale routes less problematic
- Needs sequence numbers to prevent loops
- Better for more dynamic, changing environments

Routing in ad hoc networks

- Reactive protocols: routes are constructed on demand. No global routing table is maintained.
- More appropriate for networks with high rate of changes
  - Ad hoc on demand distance vector routing (AODV)
  - Dynamic source routing (DSR)
- Need flooding
  - Inefficient in large networks

Geographical routing: Using location

- Geographical routing uses a node's location to discover path to that node.



Greedy Routing:
Forward to the neighbor that is nearest to the destination

Geographical routing

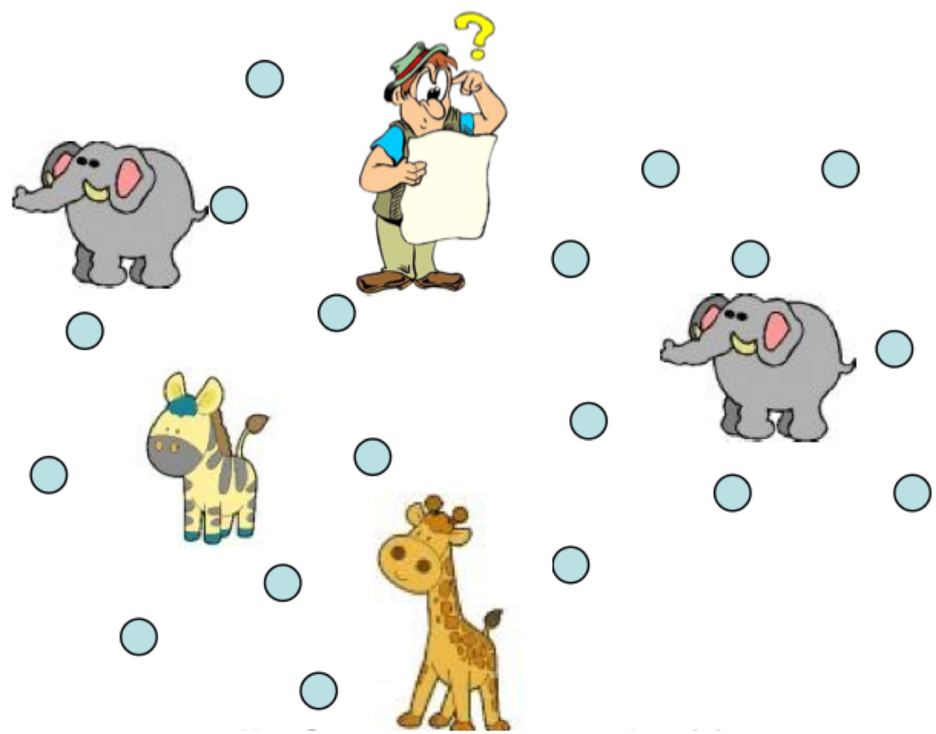
- Assumptions:
  - Nodes know their own geographical location
  - Nodes know their 1-hop neighbors
  - Routing destinations are specified geographically (a location, or a geographical region)
  - Each packet can hold a small amount of routing information.

Sensor network

- Sensors enabled with wireless
  - Can communicate with nearby sensors
  - Communication to server relatively costly
- Low power, but lots of data
  - Not worth sending everything to server
- Try use the data directly inside the network
  - In-network distributed computing

Problem: How to find the relevant data?

- A tourist in a park asks
- "Where is the elephant?"
- Out of all the sensors/cameras which one is close to an elephant?



Data centric routing

- Traditional networks try to route to an IP address
- Find path to the node with a particular ID
- But what if we try to find data, not specific nodes?
- After all, delivering data is the ultimate goal of routing and networks
- Data centric storage
  - Storage depends on the data (elephant, giraffe,song…)
- Data centric routing (search)
  - Route to the data

Distributed Database

- Information Producer
  - Can be anywhere in the network
  - May be mobile
  - Many producers may generate data of the same type
- User or Information Consumer
  - Can be anywhere
  - May be many

Distributed Database: Challenges

- Consumer does not know where the producer is, and vice versa
- Need to search : Must be fast, efficient

Basic methods:

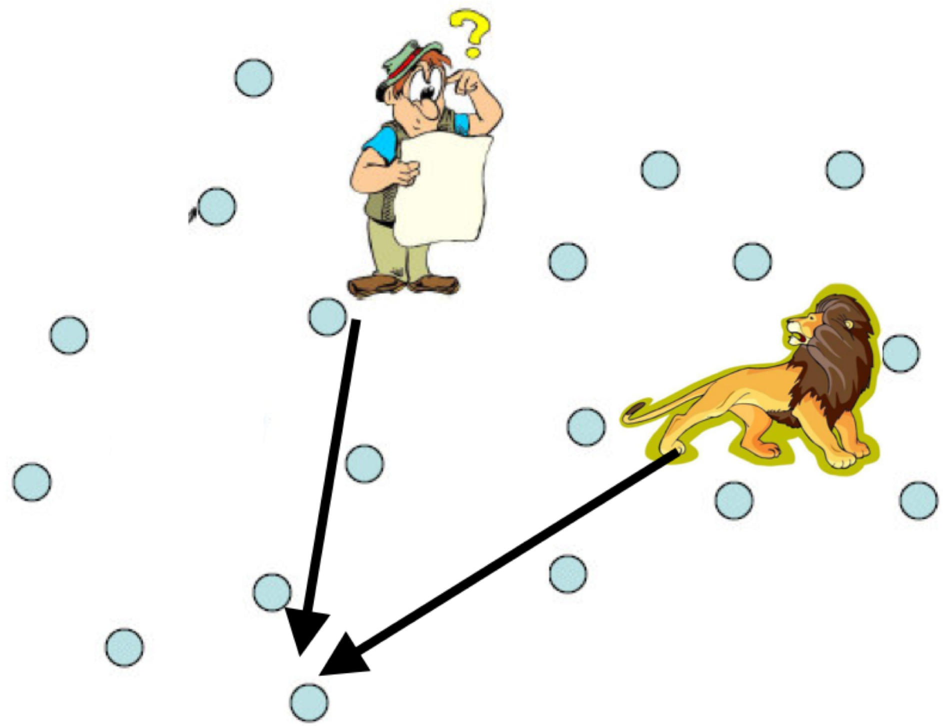
- Push: Producer disseminates data (flood)
- Pull: Consumer looks for the data (flood the query)
- Push-pull: Both producer, consumer search for each-other

Distributed hash tables

- Use a hash on the data: **h**(song1.mp3) = node#26
- Anyone that has song1.mp3 informs node#26
- Anyone that needs Song1.mp3 checks with node#26
- Used in peer to peer systems like Chord, pastry etc

Geographic Hash Tables

- Content based hash gives coordinates:
  - **h**(lion) = (12, 07)
- Producer sends msg to (12, 07) by geographic routing and stores data
- Consumer sends msg to (12, 07) by geographic routing and gets data



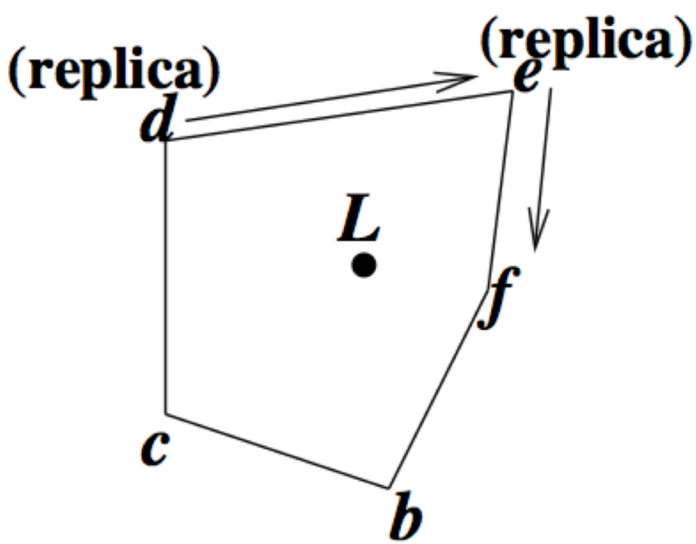
GHT

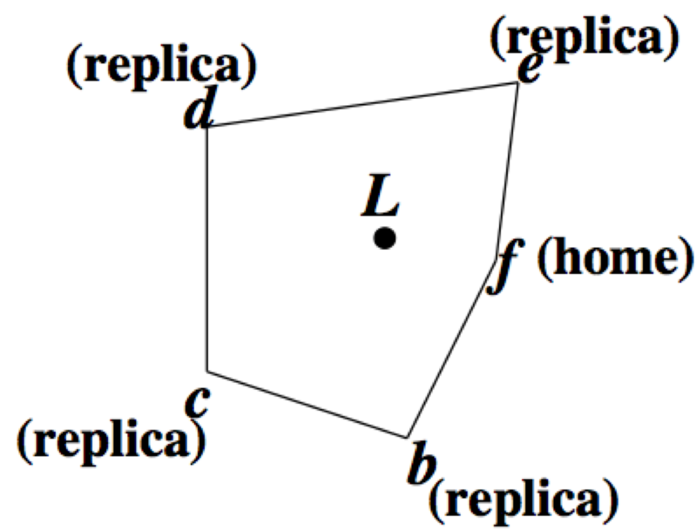
- What if there is no sensor at (12, 07) ?

- Use the sensor nearest to it

Fault handling

- What if home node a dies?
- Replicas have a timer that triggers a new check
- A new node becomes home





GHT

- Advantages
  - Simple
  - Handles load balancing and faults
- Disadvantages
  - Not distance sensitive: everyone has to go to hash node even if producer and consumer are close
  - If a data is queried or updated often, that node has a lot of traffic – bottleneck

Mobile, Ad-hoc and Sensor network

- A difficult model – least infrastructure, low power nodes, communication/computation expensive
- Not entirely realistic
- However, it makes least number of assumptons
  - useful as a basis for developing distributed protocols/algorithms
  - Which can then be enhanced using available infrastructure in specific cases