Autonomous Networking

Gaia Maselli
Dept. of Computer Science

Some slides in this course are readapted from lecture slides from ***Prof. Tommaso Melodia*** *(Northeastern University, Boston)*

Today's plan

- MAC protocols
  - S-MAC
- Routing protocols
  - proactive
  - Reactive
  - Geographic

Communication patterns

- **Broadcast** or **interest dissemination (1 to all)**
  - A broadcast pattern is generally used by a base station (sink) to transmit some information to all the sensor nodes of the network.
  - All nodes of the network are intended receivers
  - Broadcasted information may include *queries, program updates* for sensor nodes, or *control packets* for the whole system.
- **Convergecast** or **data gathering** (**All/many to 1**)
  - All or a group of sensors communicate to the sink.
  - Typically used to collect sensed data

Properties of a well-defined mac protocol

- To design a good MAC protocol for wireless sensor networks, the following attributes must be considered
- **Energy efficiency:** energy-efficient protocols in order to prolong the network lifetime must be defined
- **Scalability** and **adaptability to changes**: changes in network size, node density, and topology should be handled rapidly and effectively for successful adaptation
- Latency, throughput, and bandwidth utilization may be secondary in sensor networks, but desirable

Reasons of energy waste

- ***Collision***: When a node receives more than one packet at the same time, these packets are termed collided, even when they coincide only partially. All packets that cause the *collision* have to be discarded and retransmissions of these packets are required, which increase the energy consumption.
- ***Overhearing***: meaning that a node receives packets that are destined to other nodes.
- ***Control-packet overhead***: A minimal number of control packets should be used to make a data transmission.
- ***Idle listening***: listening to an idle channel in order to receive possible traffic.
- ***Overemitting***: caused by the transmission of a message when the destination node is not ready.

Techniques for WSN MAC

Contention based

- On-demand allocation for those that have frames for transmission
- Sensing the carrier before attempting a transmission
- **Scalable / no need for central authority**
- **Idle listening / Interference / Collisions / Traffic fluctuations -> Energy consumption**
- **Multi-hop topologies (hidden / exposed terminal problem)**

Scheduled based:
Fixed assignment or on demand

- Schedule that specifies when, and for how long, each node may transmit over the shared medium
- **Energy efficient**
- **Interference, collisions are not a problem**
- **Synchronization**
- **Central authority**

Contention-Based Mac Protocols: S-MAC

W. Ye, J. Heidemann, D. Estrin, , “Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks,'‘ IEEE/ACM Trans. on Networking, June 2004.

S-MAC

- S-MAC tries to reduce energy waste from all the above sources (collision, overhearing, control-packet overhead, etc.)

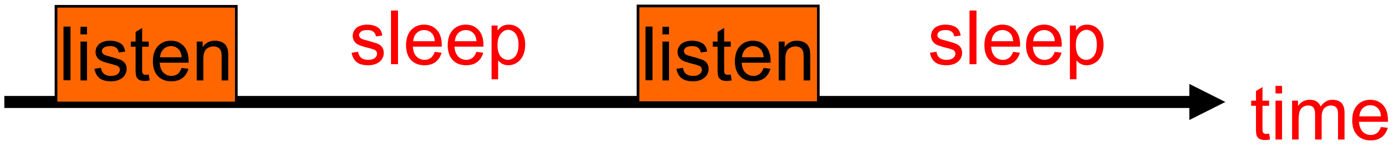
- **In exchange it accepts some performance reduction in latency**

- Basic idea: In S-MAC nodes become more active when there is traffic in the network, they do not have to listen all the time

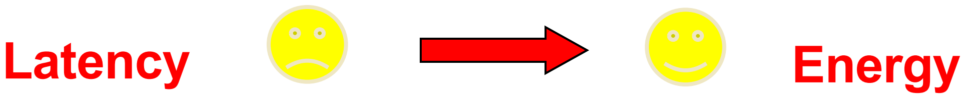
S-MAC: Sleep MAC

- Problem: “**Idle Listening**” consumes significant energy (Measurements have shown that idle listening consumes 50%–100% of the energy required for receiving)

- Solution: **Periodic listen and sleep**



- During sleeping, **radio is turned off**
- Reduce duty cycle to ~ 10% (Listen for 200ms and sleep for 2s)



Periodic listen and sleep

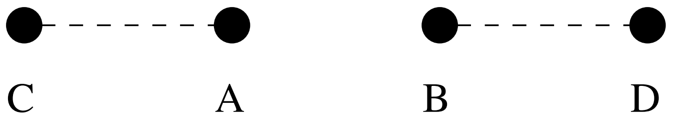
- Each node sleeps for some time, and then wakes up and listens to see if any other node wants to talk to it
- During sleeping the node turns off its radio, and sets a timer to awake itself later
- The listen interval is fixed (a priori)
- **Duty cycle**: defined as the ratio of the listen interval to the frame length (listen+sleep)
- if in each second a node sleeps for half second and listens for the other half, its duty cycle is reduced to 50%. So we can achieve close to 50% energy savings.

S-MAC

- **Each node** goes into periodic sleep mode during which it **switches the radio off** and sets a timer to awake later
- When the timer expires it wakes up and listens to see if any other node wants to talk to it
- The duration of the sleep and listen cycles are application dependent and they are set the same for all nodes
- What if a node wants to talk to another node that is sleeping?
- **periodic synchronization** among nodes

Syncronized Periodic Sleep and Listen

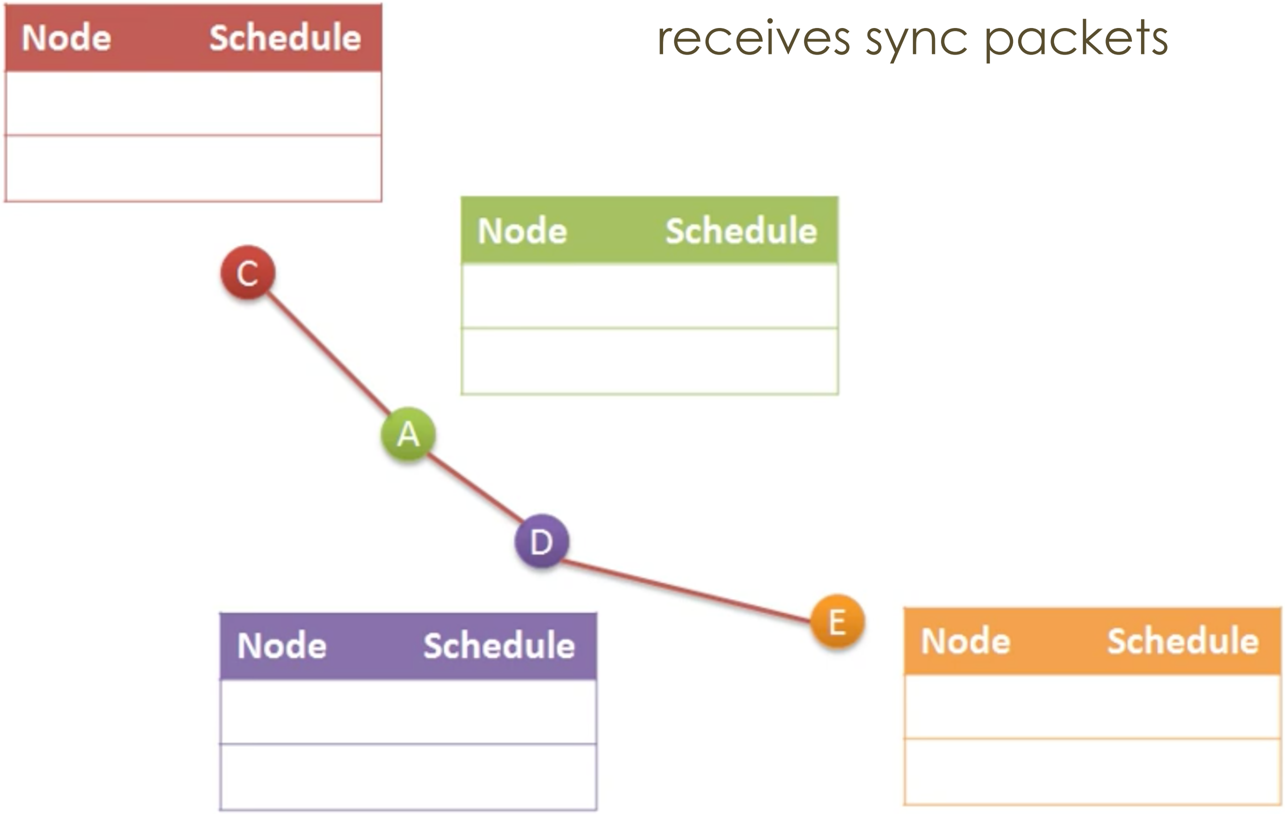
- All nodes are free to choose their own listen/sleep schedules
- To reduce control overhead, **neighboring nodes are synchronized together** (they listen at the same time and go to sleep at the same time)
- Neighboring nodes **form virtual clusters** so as to set up a common sleep schedule.



Neighboring nodes A and B have different schedules. They synchronize with nodes C and D respectively.

Schedule maintainance

- Each node maintaines a table with neighbors' schedule
- Table entries are filled when the node receives sync packets



Synchronization

- SYNC packets are exchanged periodically to maintain schedule synchronization

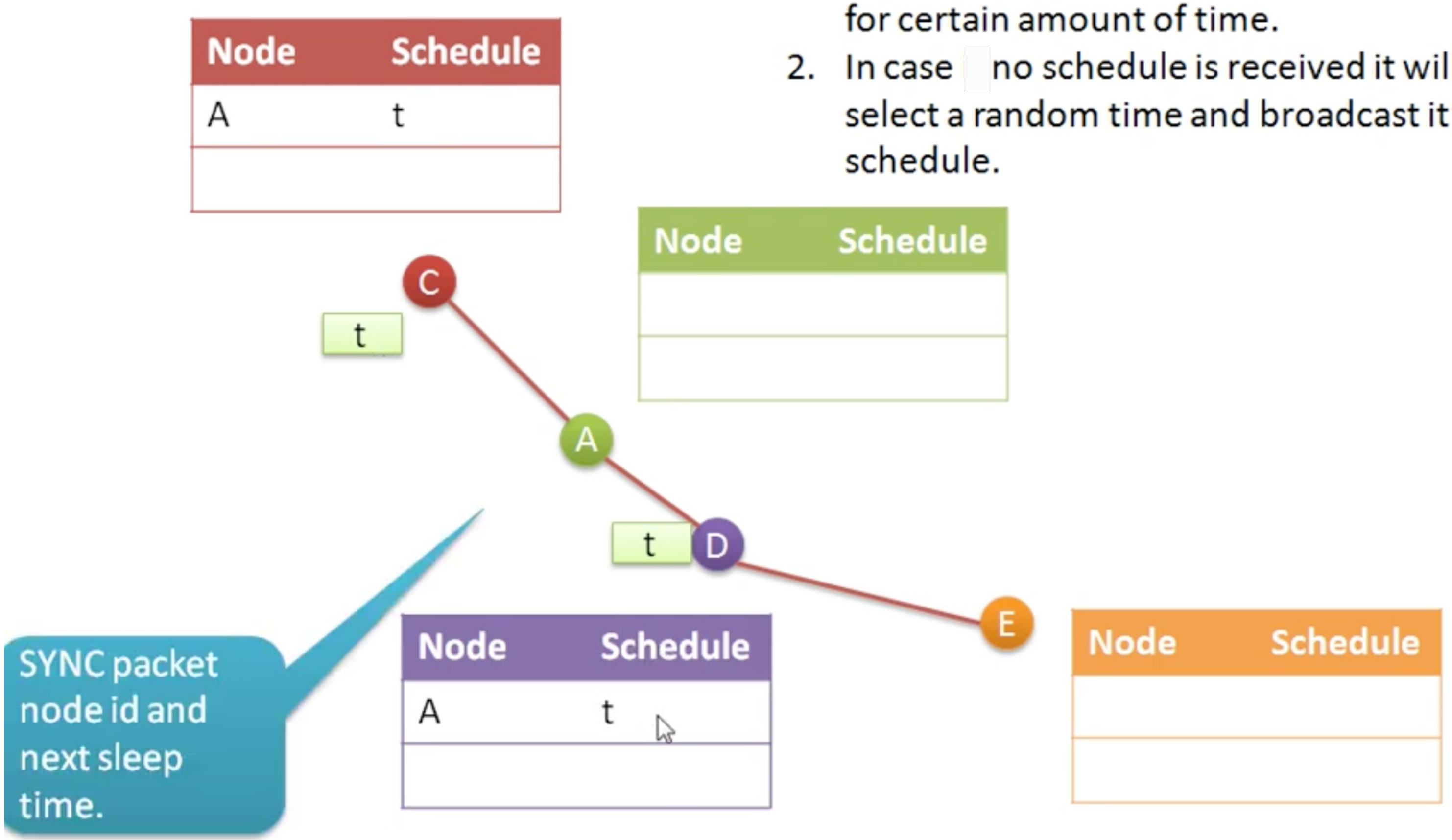
SYNC PACKET

Sender Node ID	Next Sleep Time

- SYNCHRONIZATION PERIOD: Period for a node to send a SYNC packet
- Receivers will adjust their timer counters immediately after they receive the SYNC packet

Schedule maintainance

1. Node will listen for incoming schedules for certain amount of time.
2. In case no schedule is received it will select a random time and broadcast it's schedule.



Node	Schedule
A	t

Node	Schedule

Node	Schedule
A	t

Node	Schedule

SYNC packet node id and next sleep time.

Choosing and Maintaining Schedules

- Each node maintains a **schedule table** that stores schedules of all its known neighbors

- For initial schedule, do:

1. A node first listens to the medium for a certain amount of time (at least the synchronization period)

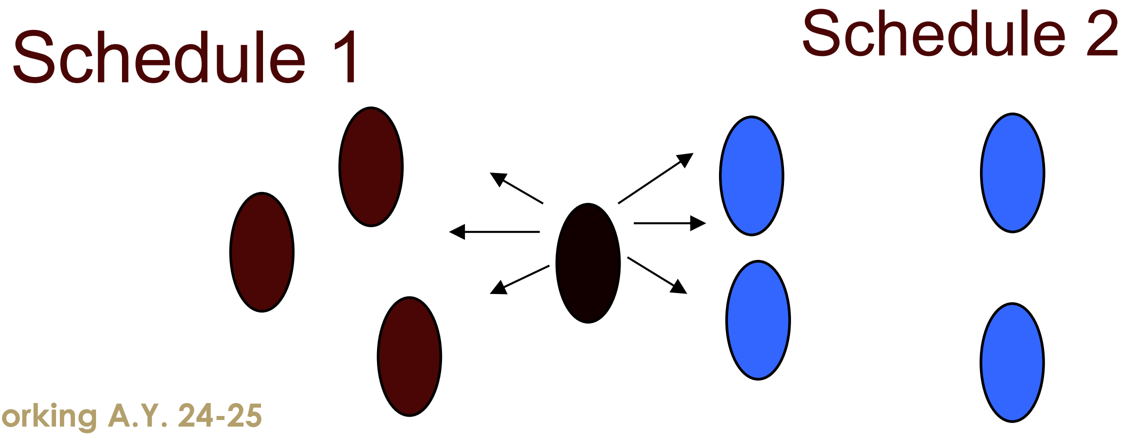
2. If it does not hear a schedule from another node, it randomly chooses a schedule and broadcasts its schedule with a SYNC packet immediately

3. This node is called a Synchronizer

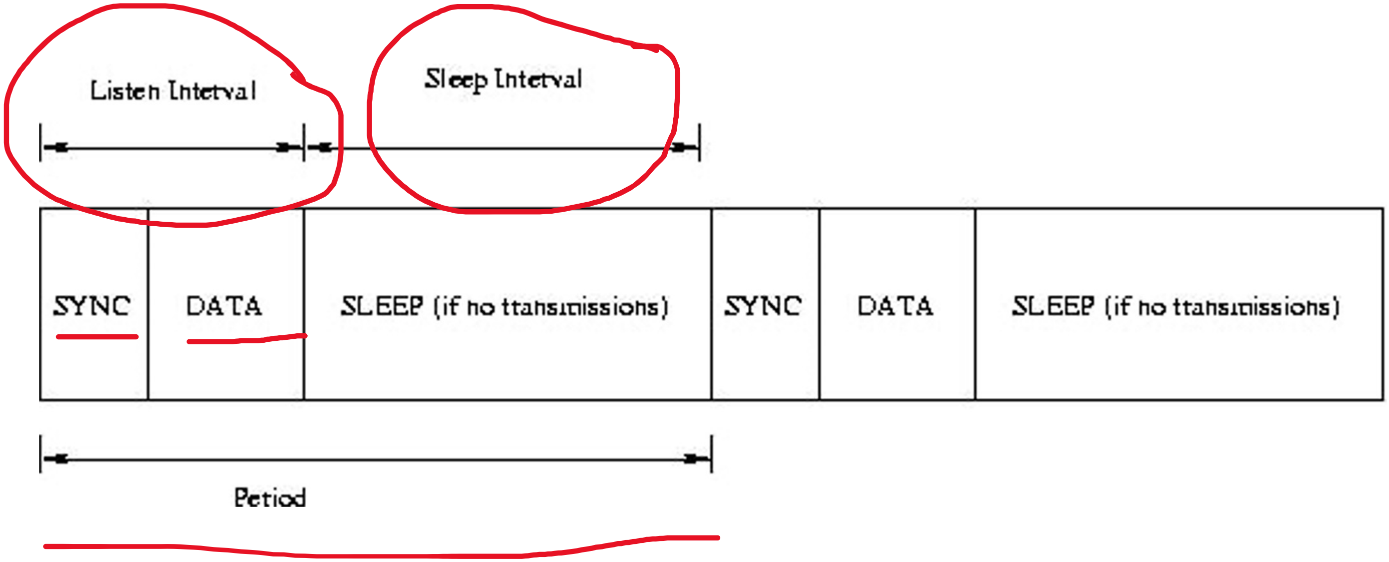
4. If a node receives a schedule from a neighbor before choosing its own schedule, it just follows this neighbor's schedule, i.e. becomes a Follower, waits for a random delay and broadcasts its schedule

Coordinated Sleeping

- Of course, in a large network, we cannot guarantee that all nodes follow the same schedule
- The node on the border will follow both schedules
- When it broadcasts a packet, it needs to do it twice, first for nodes on schedule 1 and then for those on schedule 2



Periodic Listen and Sleep



Collision Avoidance

- S-MAC is based on **contention**, i.e., if multiple neighbors want to talk to a node at the same time, they will try to send when the node starts listening

- Similar to IEEE 802.11, i.e. use RTS/CTS mechanism to address the hidden terminal problem

- Perform **carrier sense** before initiating a transmission

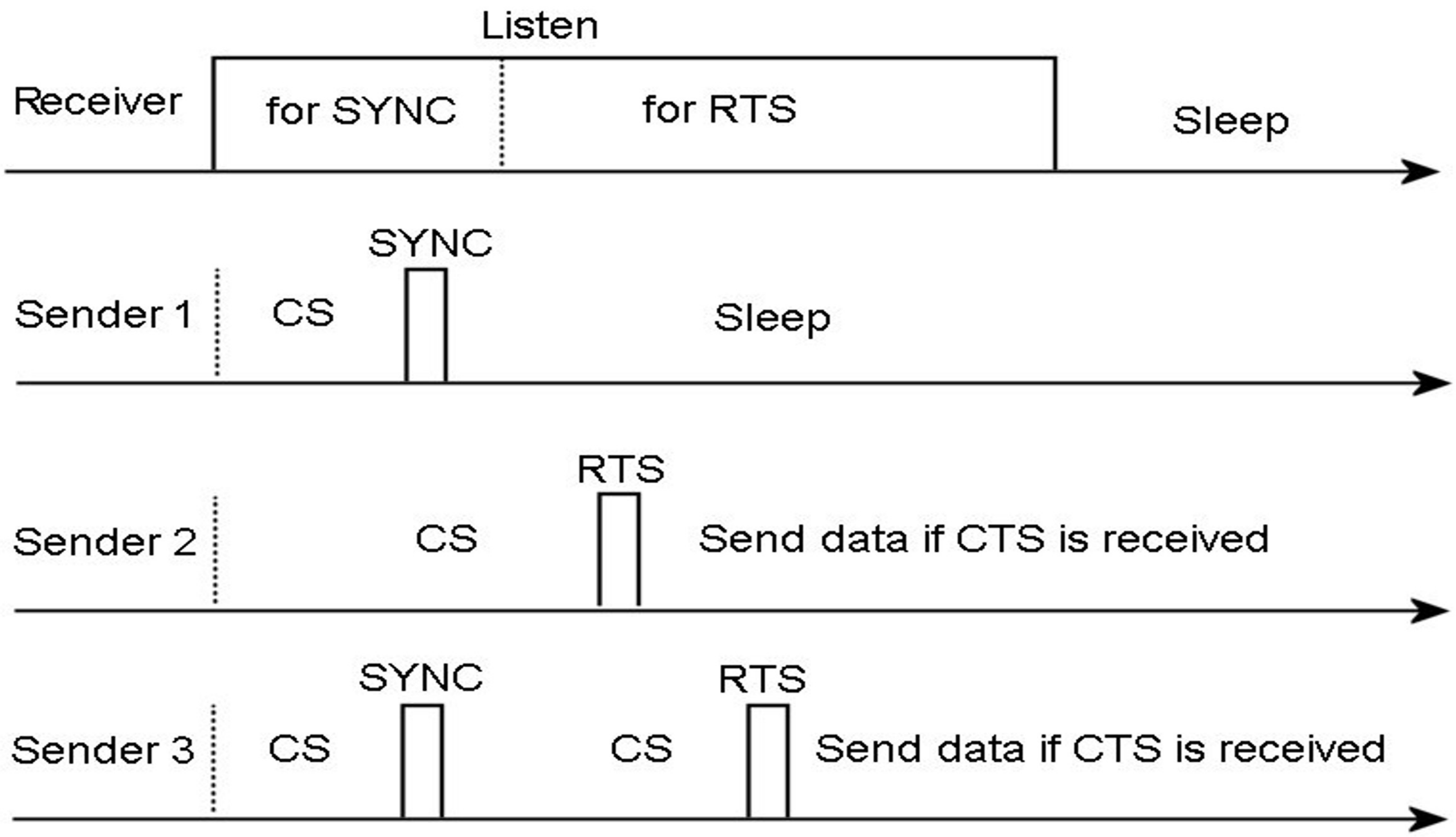
Collision Avoidance

- If a node fails to get the medium, it goes to sleep and wakes up when the receiver is free and listening again
- Broadcast packets are sent without using RTS/CTS
- Unicast data packets follow the sequence of RTS/CTS/DATA/ACK between the sender and receiver
- Duration field in each transmitted packet indicates how long the remaining transmission will be so **if a node receives a packet destined to another node, it knows how long it has to keep silent**
- The node records this value in **network allocation vector (NAV)** and sets a timer for it

Collision Avoidance

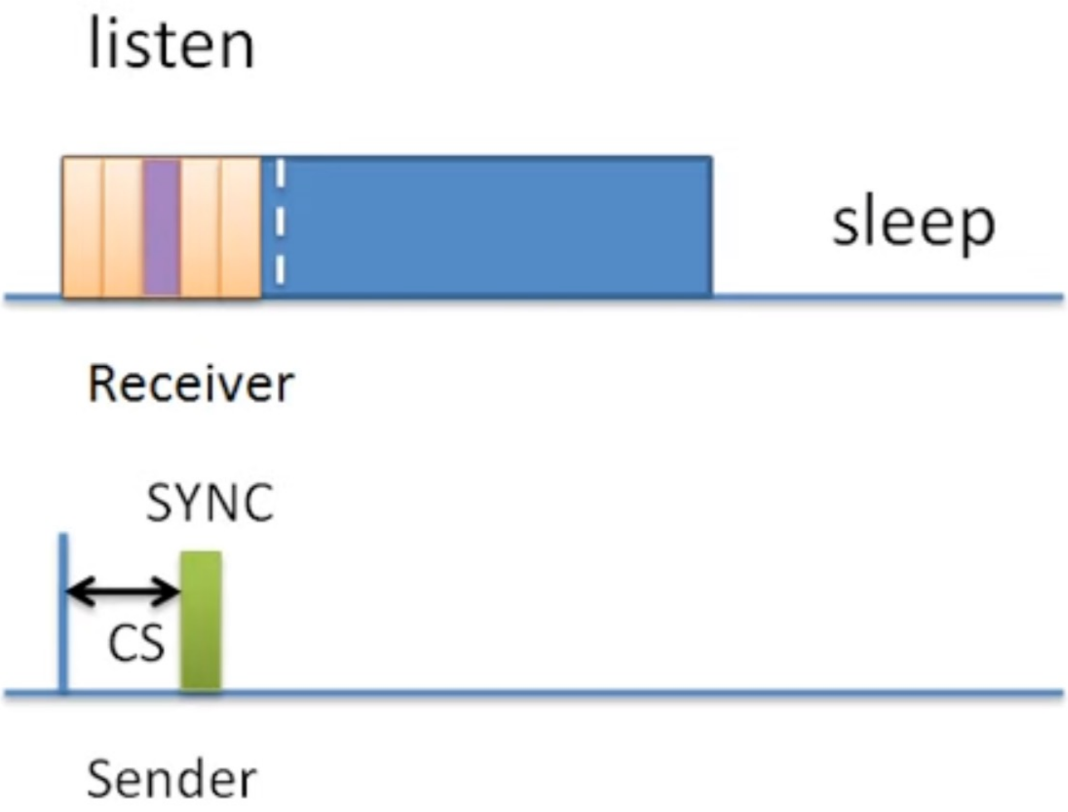
- When a node has data to send, it first looks at NAV
- If this value is not zero, then medium is busy (**virtual carrier sense**)
- The medium is determined as free if **both virtual and physical carrier sense indicate the medium is free**
- All immediate neighbors of both the sender and receiver should sleep after they hear RTS or CTS packet until the current transmission is over

Maintaining Synchronization



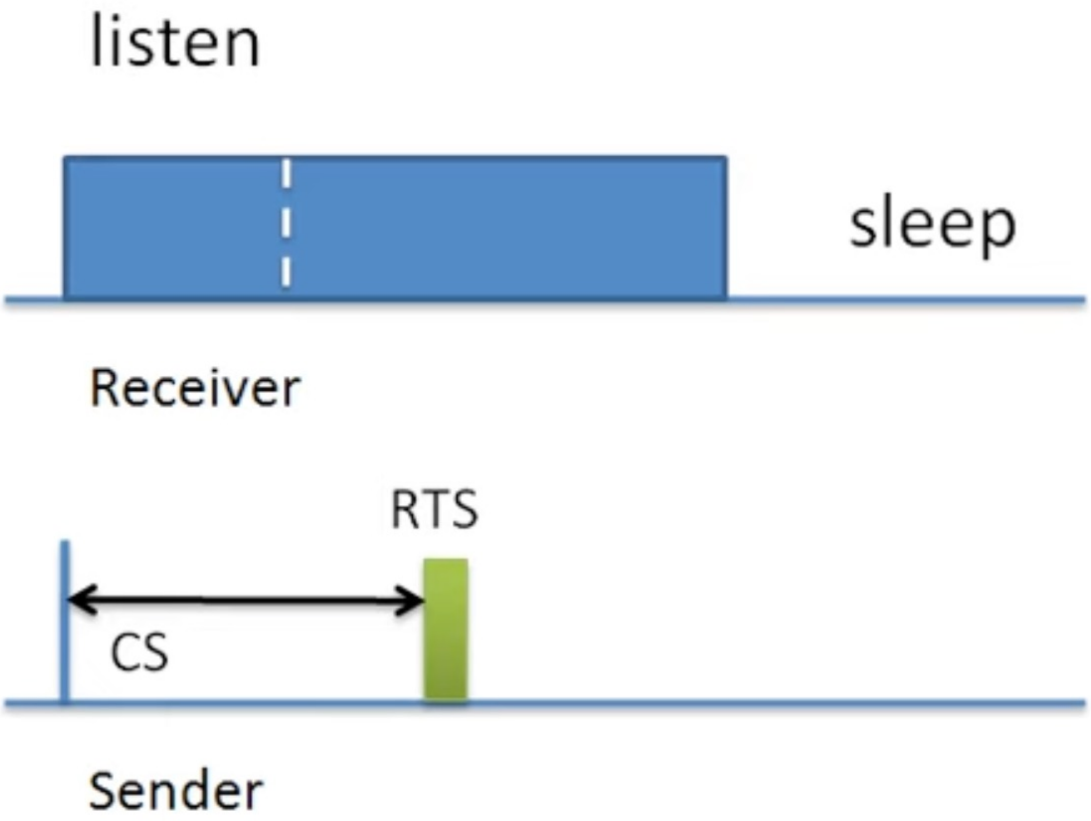
CS: carrier sense (to avoid collisions)

SYNC and data sending



- Time is divided into minislots
- Sender select a minislot to end carrier sense
- If chanel is free senser will transmit SYNC in the next minislot

Data sending

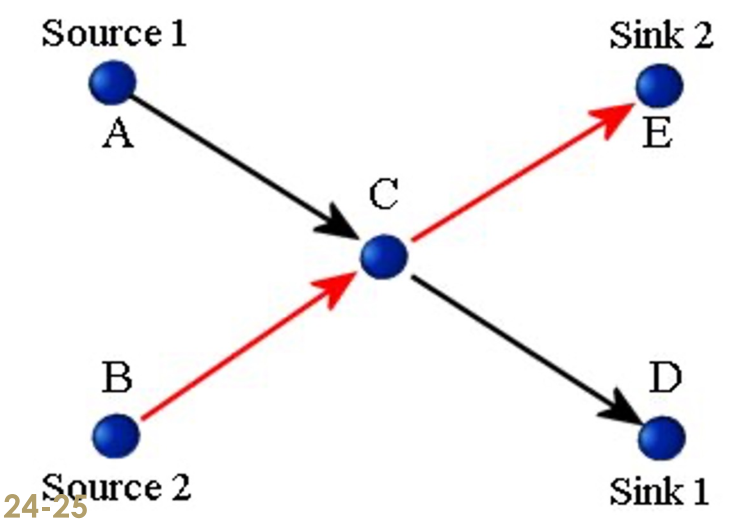


S-MAC energy efficiency

- Collision: RTS/CTS
- Idle listening: by periodically putting nodes in sleep state
- Overhearing avoidance: Nodes go to sleep after they hear an RTS or CTS not addressed to them

S-MAC
Performance Evaluation

- Topology: Two-hop network with two sources and two sinks
- Sources periodically generate a sensing message which is divided into fragments
- Traffic load is changed by varying the inter-arrival period of the messages: (for inter-arrival period of 5s, a message is generated every 5s by each source. Here it varies between 1-10s)

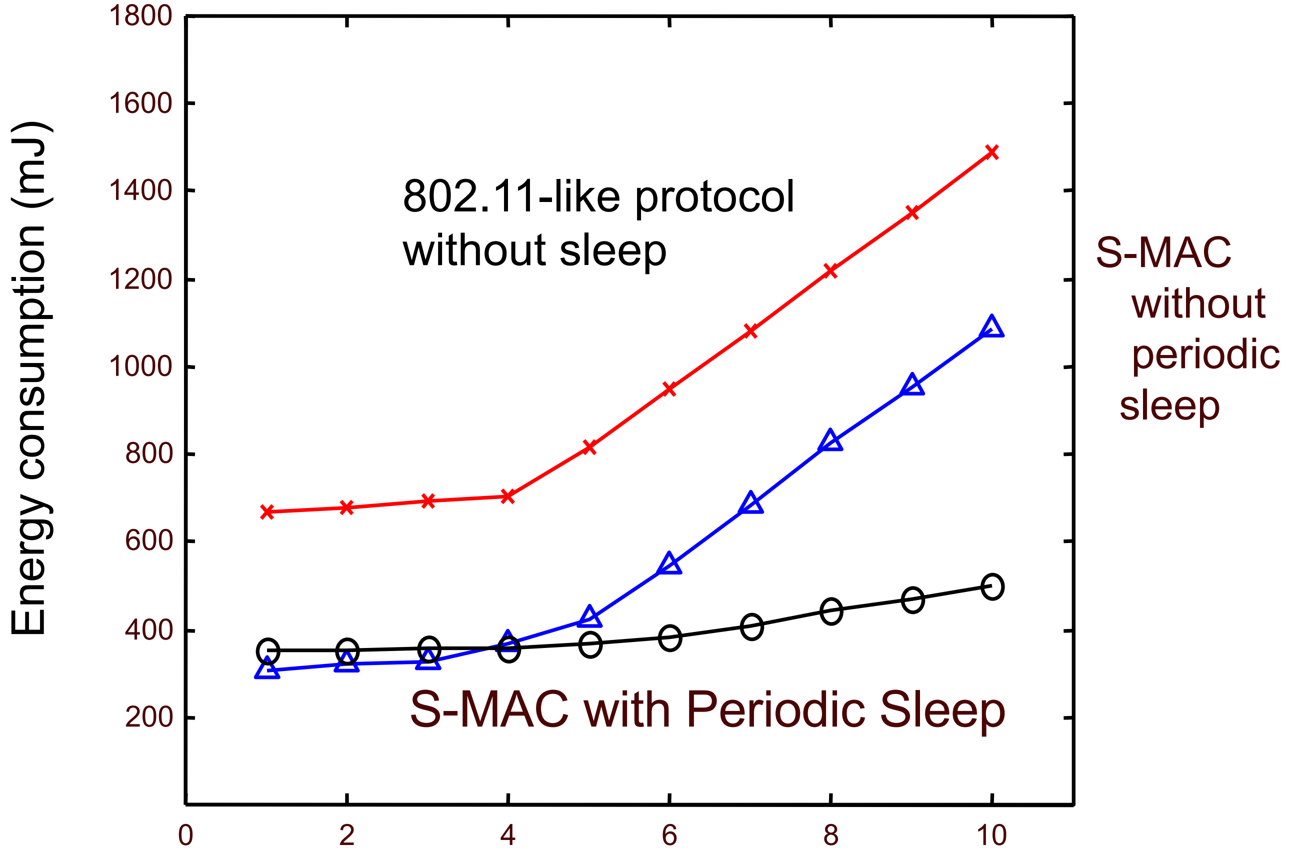


S-MAC – Example

- In each test, there are 10 messages generated on each source node
- Each message has 10 fragments, and each fragment has 40 bytes (200 data packets to be passed from sources to sinks)
- The total energy consumption of each node is measured for sending this fixed amount of data

Experiments

Average energy consumption in the source nodes A&B



Message inter-arrival period (traffic load) (second)
(small value → heavy traffic load)

Experiments

- S-MAC consumes much less energy than 802.11-like protocol without sleeping
- At heavy load, idle listening rarely happens, energy savings from sleeping is very limited.
- At light load, periodic sleeping plays a key role

S-MAC - Conclusions

- A mainly static network is assumed
- Trades off latency for reduced energy consumption
- Redundant data is still sent with increased latency

Routing

Routing

- Routing technique is needed for sending the data between the sensor nodes and the base stations, so as to establish **multi-hop** communication
- The applied routing strategy should ensure the **minimum of the energy consumption** and hence **maximization of the lifetime of the network**

Ad Hoc Routing Protocols – Classification

- Network Topology
  - Flat, Hierarchical

- Which data is used to identify nodes?
  - An arbitrary identifier?
  - The position of a node?
    - Can be used to assist in geographical routing protocols because choice of next hop neighbor can be computed based on destination address
    - Scalable and suitable for sensor networks

Flat routing protocols

- Flat Networks Routing Protocols for WSNs can be classified, according to the **routing strategy**, into three main different categories depending on:
- ***When*** does the routing protocol operate?
- ***Proactive*** Protocols
  - Routing protocol ***always*** tries to keep its routing data up-to-date
  - Active before tables are actually needed
  - Also known as ***table-driven***
- ***Reactive*** Protocols
  - Route is only determined when actually needed
  - Protocol operates ***on demand***
- ***Hybrid*** protocols
  - Combine these behaviors

Flat routing: proactive

- **Pro-active or Table-Driven Routing Protocols** work in a way similar to wired networks.

- **Each node maintains routes to all reachable destinations at all times**, whether or not there is currently any need to deliver packets to those destinations

- They also **respond to any changes in network topology** by sending updates throughout the wireless network and thus maintaining a consistent network view.

↑ When a path to some destination is needed the route is already known and there is no extra delay due to route discovery.

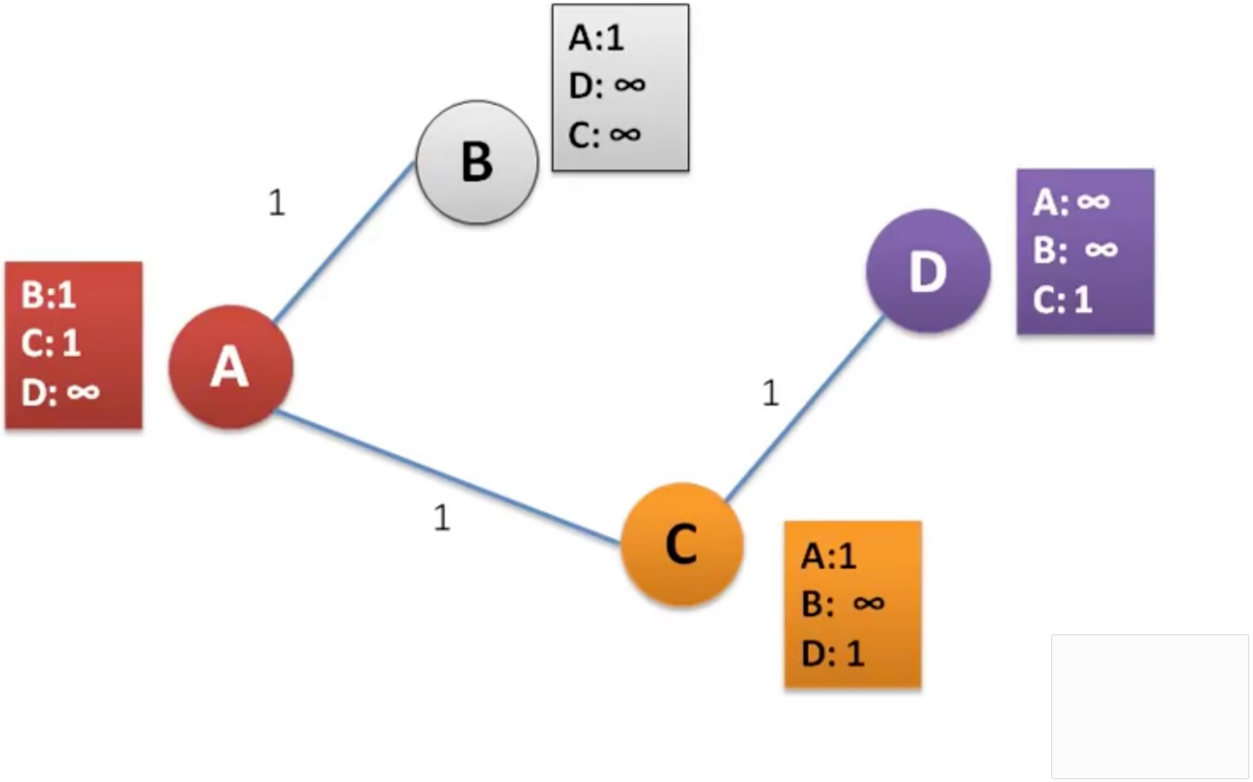
↓ Keeping the information up-to-date may require a lot of bandwidth and extra battery power.

↓ The information may still be out-of-date.

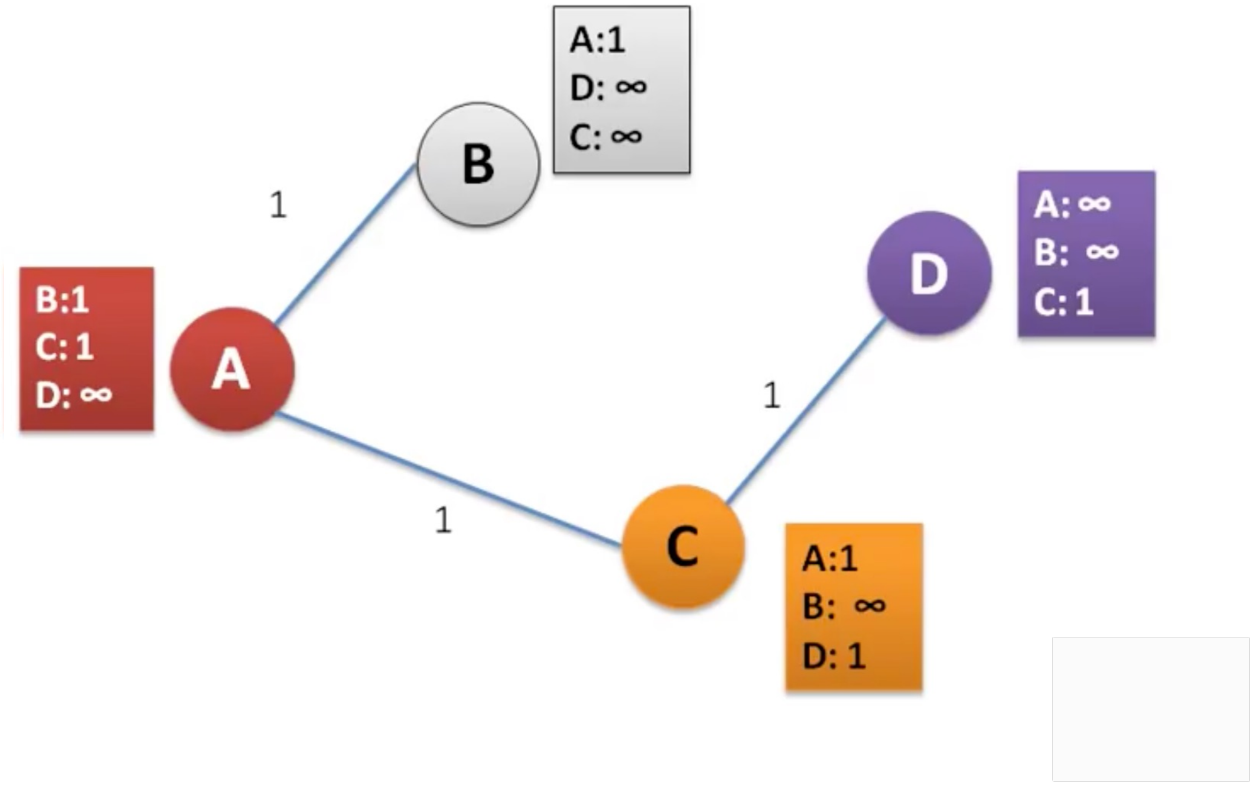
Destination Sequence Distance Vector (DSDV)

- **Adapted DISTANCE VECTOR:**
  Destination Sequence Distance Vector (DSDV)
  - Based on distributed Bellman Ford procedure
  - Periodically send full route updates
  - **Add aging information** to route information propagated by distance vector exchanges; helps to avoid routing loops
  - On topology change, send incremental route updates
  - Unstable route updates are delayed

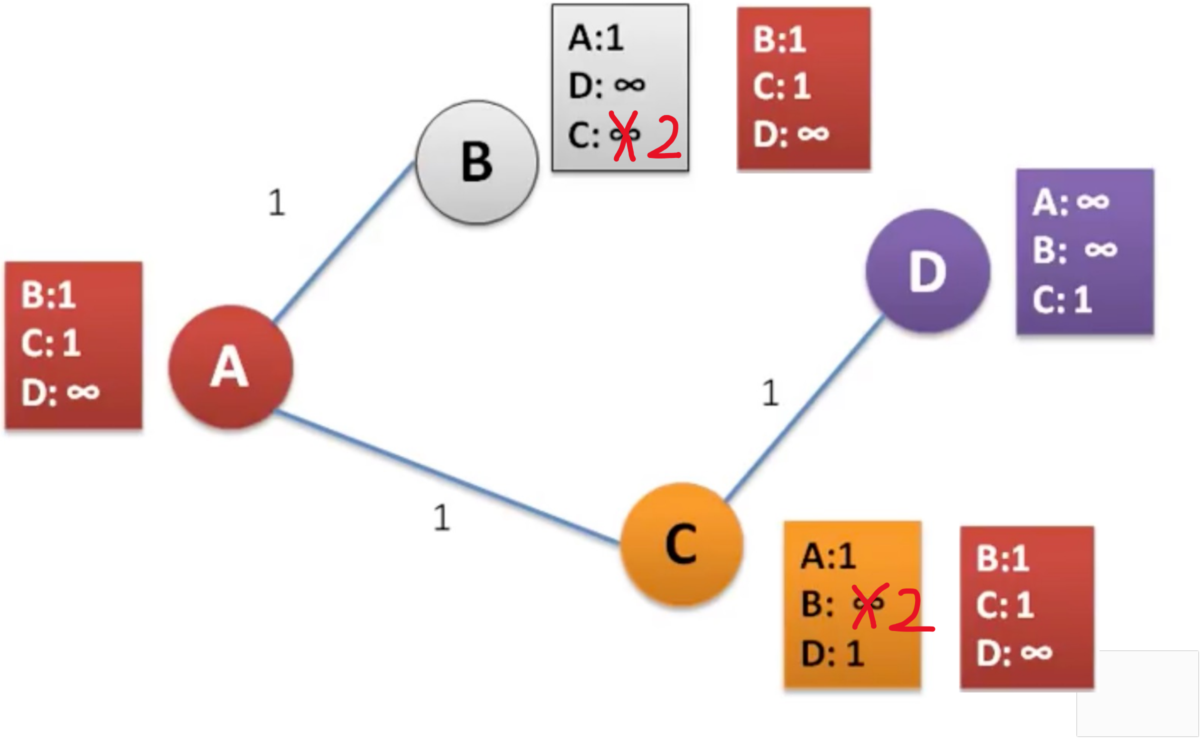
DSDV



DSDV: distance vector



DSDV: distance vector update



A:1
D: ∞
C: ~~∞~~ 2

B:1
C: 1
D: ∞

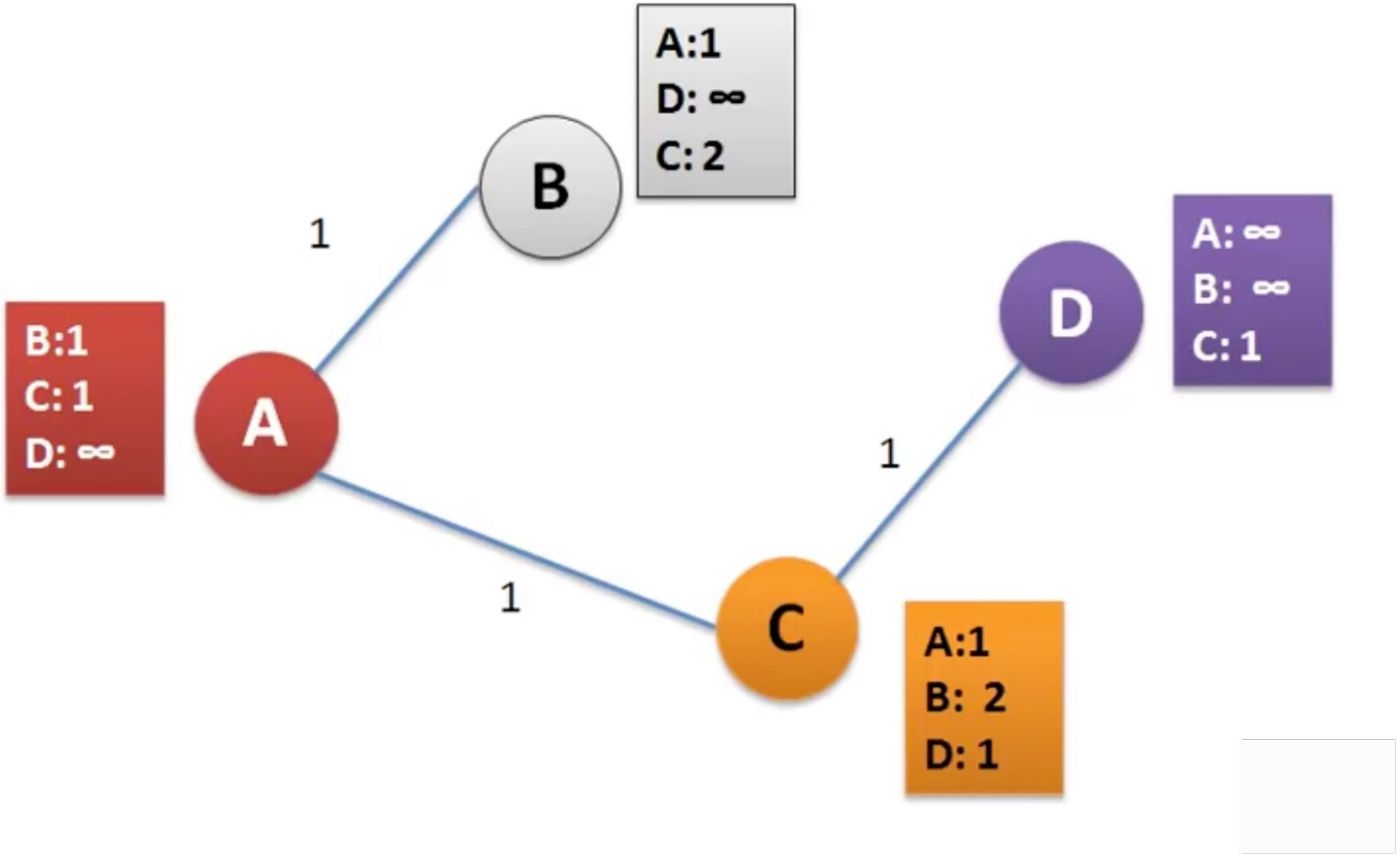
A: ∞
B: ∞
C: 1

B:1
C: 1
D: ∞

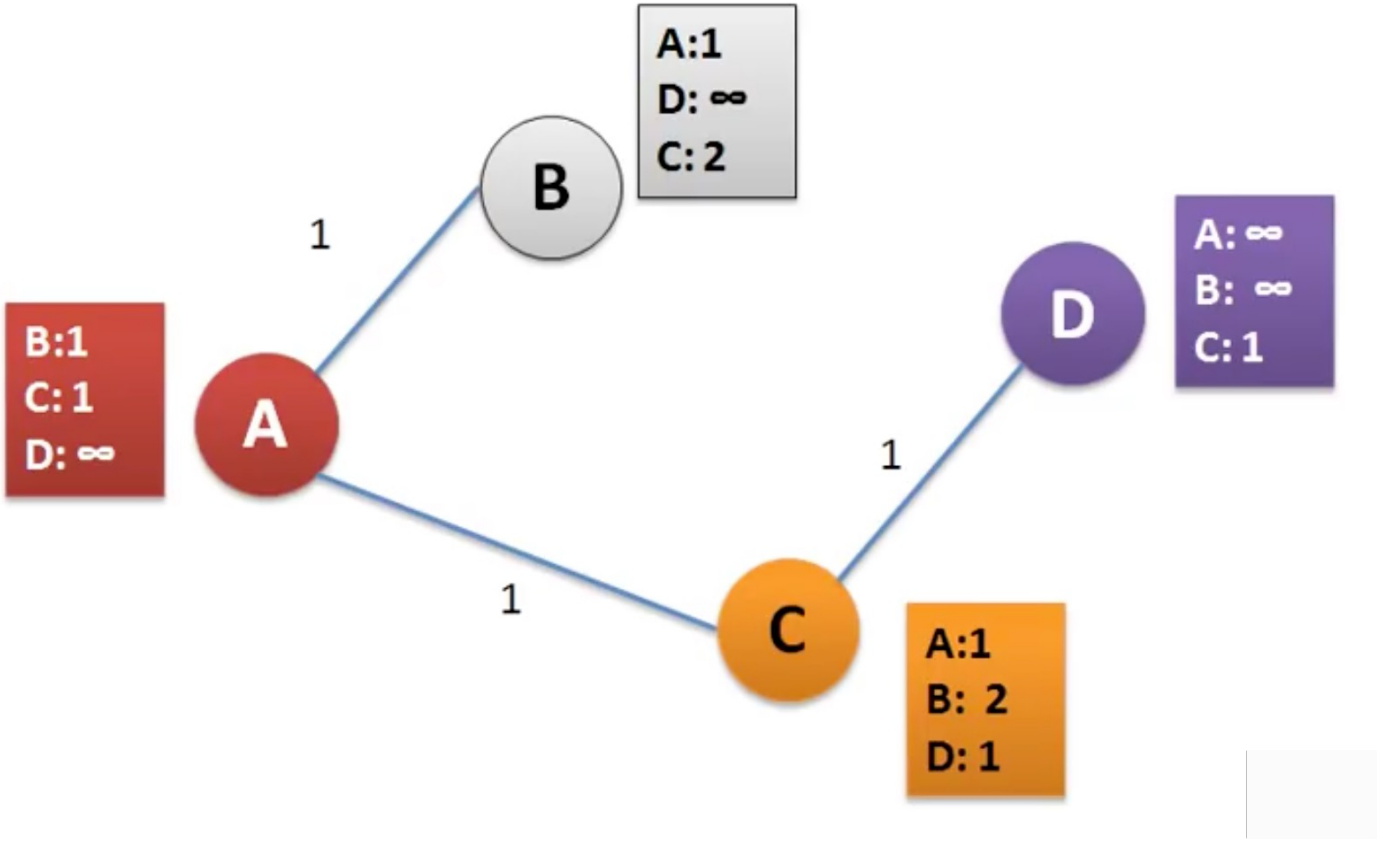
A:1
B: ~~∞~~ 2
D: 1

B:1
C: 1
D: ∞

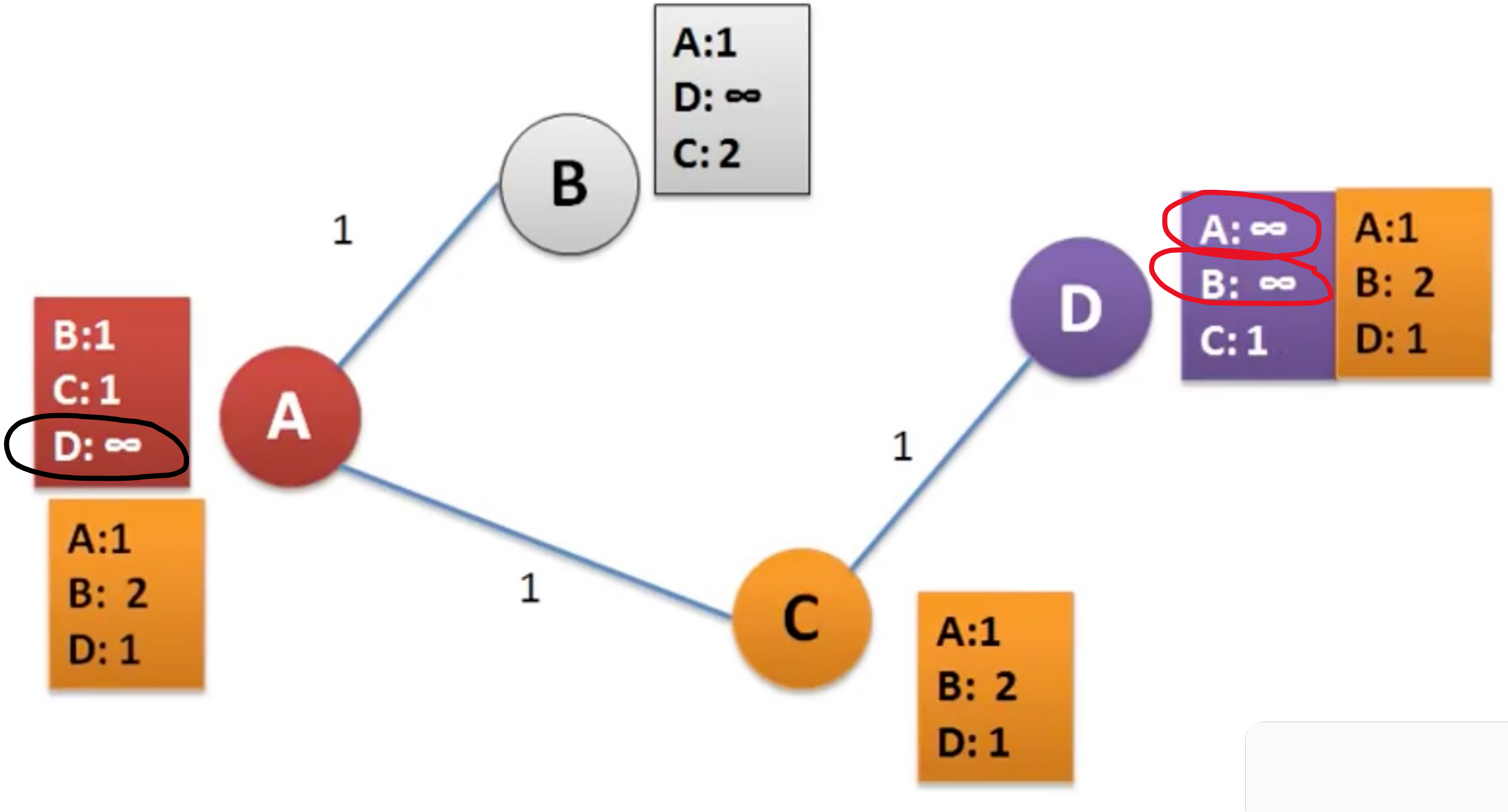
DSDV: distance vector update



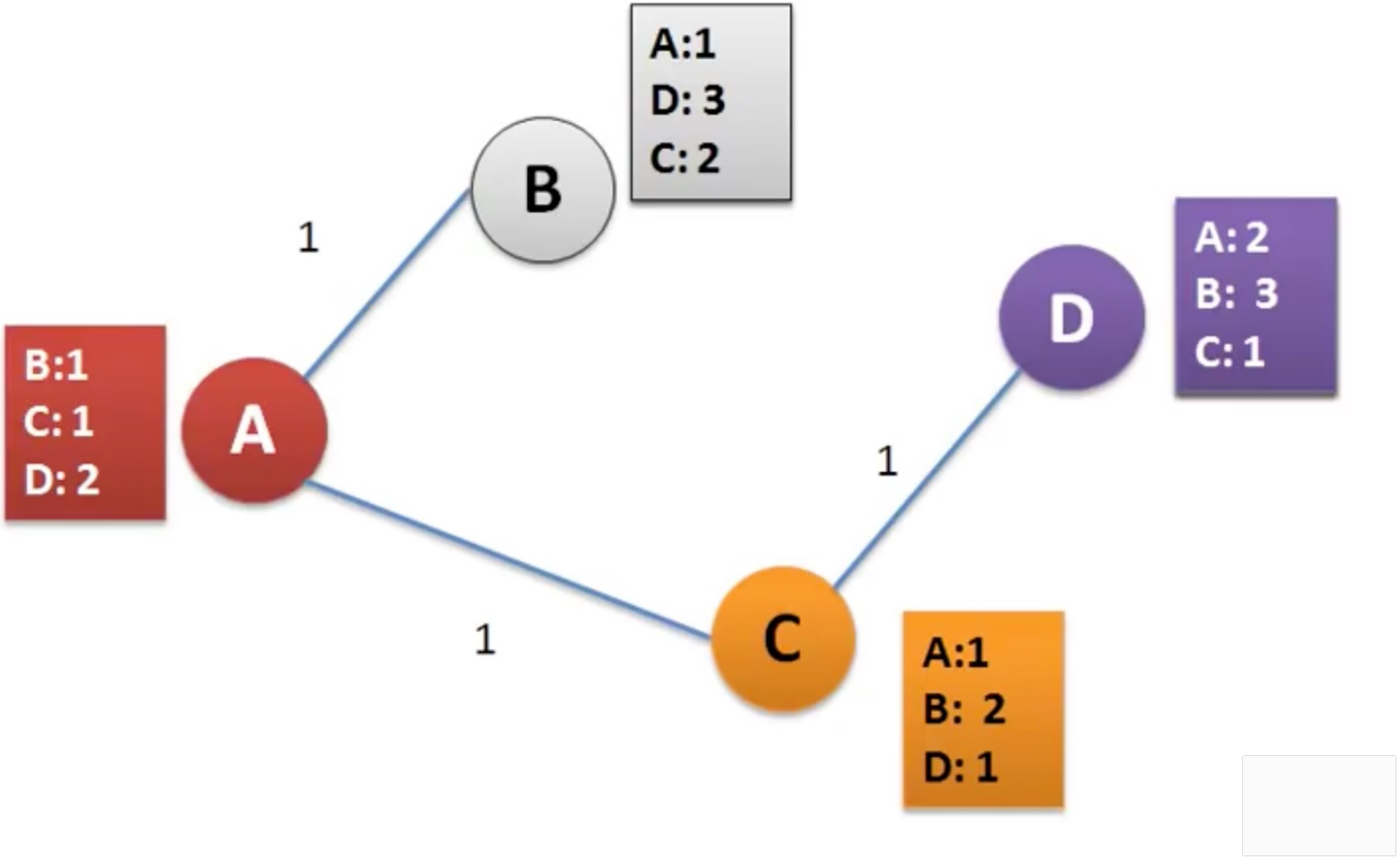
DSDV: distance vector update



DSDV: distance vector update



DSDV: distance vector update



Destination Sequence Distance Vector

- **Sequence Numbers:** To avoid loops and ensure that the routes are fresh, DSDV adds a **sequence number to each routing table entry**, which is periodically updated. Routes with higher sequence numbers are preferred as they represent more recent information.

Flat routing: reactive

- **Re-active or Source-Initiated On-Demand Routing Protocols** only start a route discovery procedure when needed
- **When a route from a source to a destination is needed**, a kind of global search procedure is started.
- This task does not request the constant updates to be sent through the network, as in pro-active protocols, but this process does **cause delays**, since the requested routes are not available and have to be found.
- In some cases, the desired routes are still in the route cache maintained by the sensor nodes. When this is the case, there is no additional delay since routes do not have to be discovered.
- The whole process is completed as soon as a route is found or all possible route combinations have been examined.

Reactive protocols

- Flooding
- Gossiping

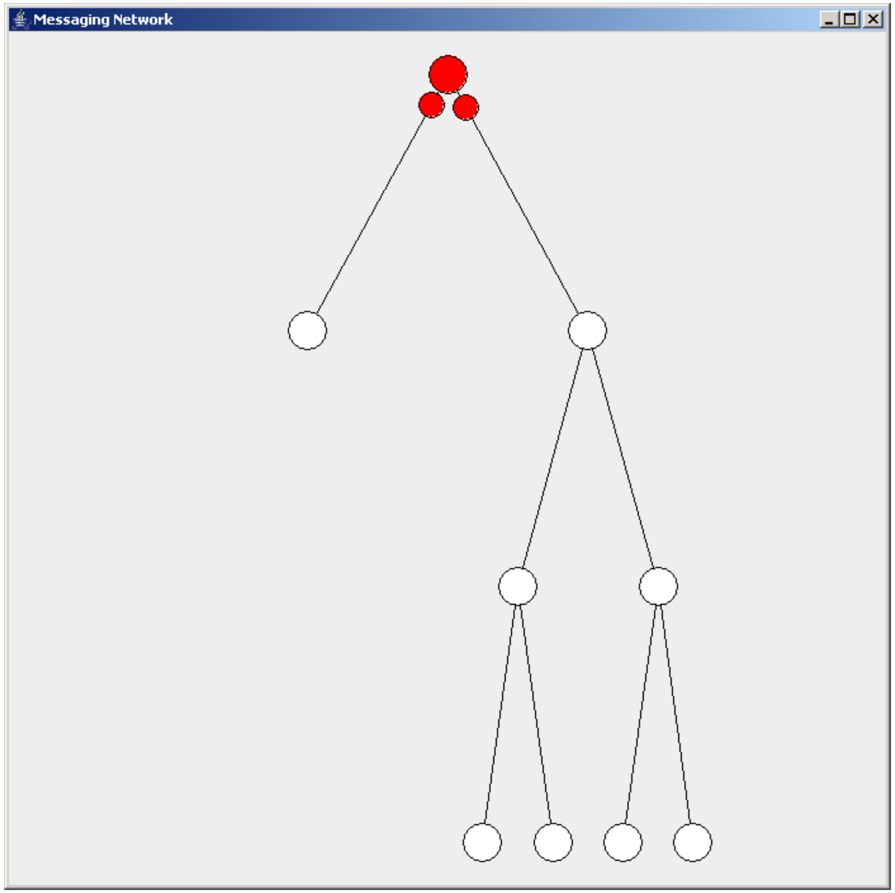
No routes

- Dynamic Source Routing (DSR)
- Ad hoc On Demand Distance Vector (AODV)

Flooding

- Flooding is an old and very simple technique which can be also used for routing in WSNs
- **Copies of incoming packets are sent by every link except the one by which the packets arrived**.
- This procedure generates an enormous amount of superfluous traffic.
- Flooding is a reactive technique, and does not require costly topology maintenance and complex route discovery algorithms.

Flooding



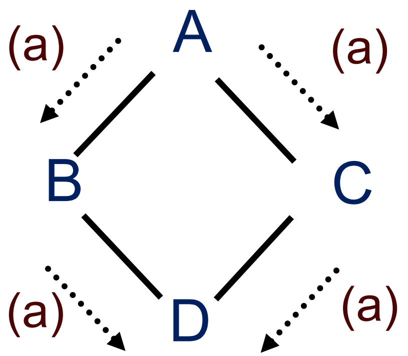
Flooding: characteristics

- Flooding has two interesting characteristics which arise from the fact that all possible routes are tried

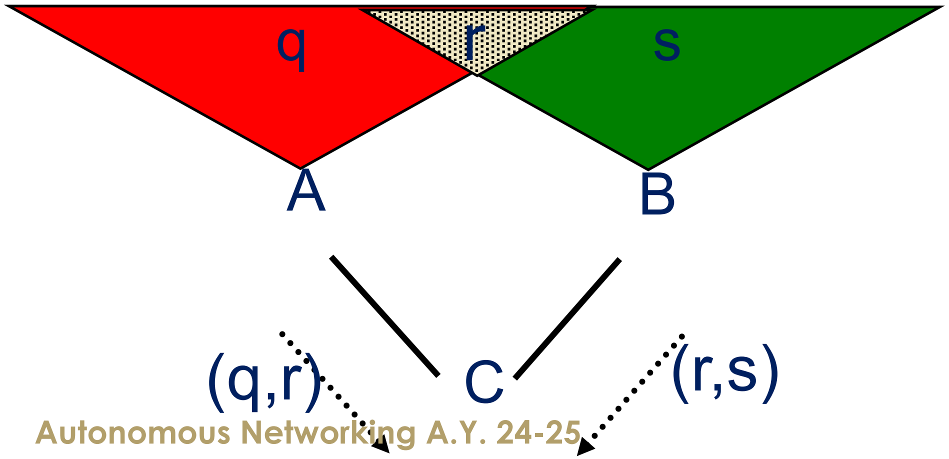
1. As long as there is a route from source to destination, the delivery of the packet is guaranteed.
2. One copy of the packet will arrive by the **quickest possible route**

Flooding: drawbacks

Implosion



Data Overlap



- Implosion: a situation where duplicated messages are broadcasted to the same node
- Overlap: If two nodes share the same under observation region, both of them may sense the same stimuli at the same time. As a result, neighbor nodes receive duplicated messages

Flooding: drawbacks

- **Resource blindness** (No knowledge about the available power of resources)
- The flooding protocol **does not take into consideration all the available energy resources.** An energy resource-aware protocol must take into account the amount of energy which is available to them all the time.
- **Flooding consumes much energy**, as for each data packet, all the nodes that are in the broadcast domain will receive packets that they will forward it to their neighbors. Thus, they require a large amount of power that causes a prohibitively short network lifetime.

Gossiping

- In gossiping, nodes send the incoming packets to a **randomly** selected neighbor.
- Although this approach avoids the implosion problem by just having one copy of a message at any node, it takes long to propagate the message to all sensor nodes in the network.

Dynamic Source Routing (DSR) – RFC 4728

- **Source routing: Each data packet sent carries in its header the complete, ordered list of nodes through which the packet will pass**
- The sender can select and control the routes used for its own packets and supports the use of multiple routes to any destination
- Including source route in the header of each data packet, other nodes forwarding or overhearing any of these packets can also easily cache this routing information for future use

Dynamic Source Routing (DSR)

- The DSR protocol is composed of two main mechanisms that work **(on demand)** together to allow the discovery and maintenance of source routes
- **Route Discovery** is the mechanism by which a node S wishing to send a packet to a destination node D obtains a source route to D
- Route Discovery is used only when S attempts to send a packet to D and does not already know a route to D
- **Route Maintenance** is the mechanism by which node S is able to detect, while using a source route to D, if the network topology has changed such that it can no longer use its route to D because a link along the route no longer works.
- When Route Maintenance indicates a source route is broken, S can attempt to use any other route it happens to know to D, or it can invoke Route Discovery again to find a new route for subsequent packets to D. Route Maintenance for this route is used only when S is actually sending packets to D.

DSR: route request

- Each Route Request contains
  - a **unique request identification**
  - a **record listing the address of each intermediate node through which this particular copy of the Route Request has been forwarded**.
- This route record is initialized to an empty list by the initiator of the Route Discovery
- When another node receives this Route Request, it **appends its own address to the route record** in the Route Request and propagates it by transmitting it as a local broadcast packet (with the same request identification)

```
   ^    "A"    ^   "A,B"   ^  "A,B,C"  ^ "A,B,C,D"
   |   id=2    |   id=2    |   id=2    |   id=2
+-----+     +-----+     +-----+     +-----+     +-----+
|  A  |---->|  B  |---->|  C  |---->|  D  |---->|  E  |
+-----+     +-----+     +-----+     +-----+     +-----+
   |           |           |           |
   v           v           v           v
```

DSR: route reply

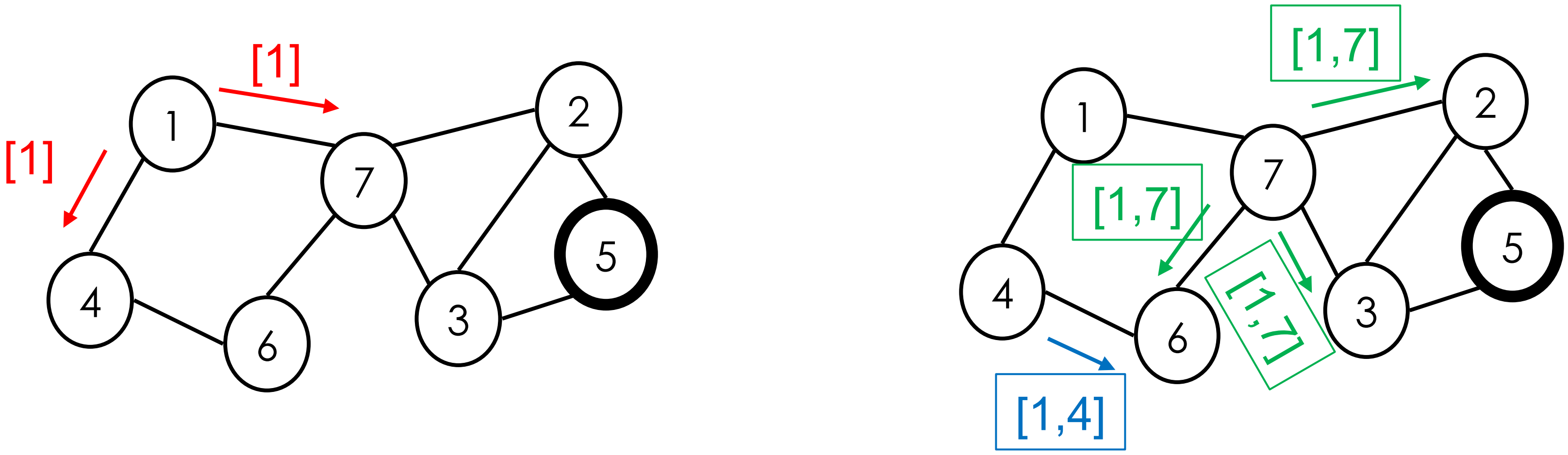
```
   ^     "A"    ^    "A,B"   ^   "A,B,C"  ^ "A,B,C,D"
   |    id=2    |    id=2    |    id=2    |    id=2
+------+     +------+     +------+     +------+     +------+
|  A   |---->|  B   |---->|  C   |---->|  D   |---->|  E   |
+------+     +------+     +------+     +------+     +------+
   |            |            |            |
   v            v            v            v
```

- When a node receives a Route Request (such as node E in this example), if it is **the target of the Route Discovery**, it **returns a "Route Reply"** to the initiator of the Route Discovery, giving a copy of the accumulated route record from the Route Request

- When the initiator receives this Route Reply, it caches this route in its Route Cache for use in sending subsequent packets to this destination.

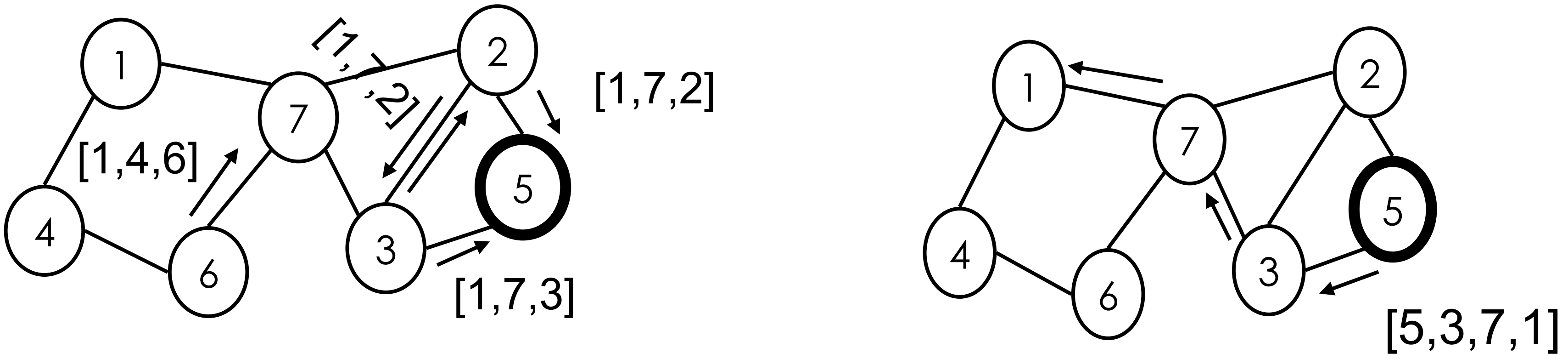
DSR Route Discovery Procedure

- Search for route from 1 to 5



- Each Route Request contains
  - a unique request identification
  - a record listing the address of each intermediate node through which this particular copy of the Route Request has been forwarded.

DSR Route Discovery Procedure



- Node 5 uses route information recorded in RREQ to send back, via ***source routing***, a route reply [5,3,7,1]

DSR: Multiple routes

- In response to a single Route Discovery a node may learn and cache multiple routes to any destination.
- This support for multiple routes allows the **reaction to routing changes to be much more rapid**, since a node with multiple routes to a destination can try another cached route if the one it has been using should fail.

Reactive Protocols – AODV

- **Ad hoc On Demand Distance Vector routing (AODV)**
  - Very popular routing protocol
  - Essentially same basic idea as DSR for discovery procedure
  - **Nodes maintain routing tables instead of source routing**
  - Sequence numbers added to handle stale caches
  - Nodes remember from where a packet came and populate routing tables with that information

AODV

- AODV is an improvement on DSDV because it typically minimizes the number of required broadcasts by creating routes **on** a **demand** basis, as opposed to maintaining a complete list of routes as in the DSDV algorithm
- When a source node desires to send a message to some destination node and does not already have a valid route to that destination, it initiates a ***path discovery*** process to locate the other node.
- Source node broadcasts a route request (RREQ) packet to its neighbors, which then forward the request to their neighbors, and so on, until either the destination or an intermediate node with a "fresh enough" route to the destination is located.

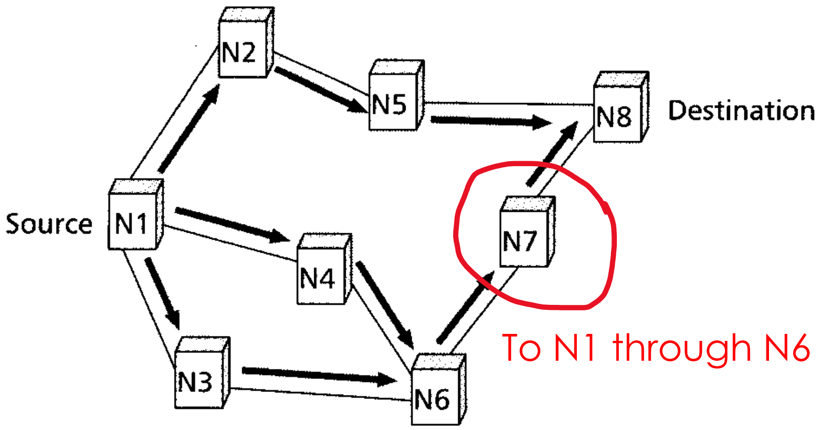
AODV

- During the process of forwarding the RREQ, **intermediate nodes record in their route tables the address of the neighbor from which the first copy of the broadcast packet is received**, thereby establishing a reverse path.

- **Once the RREQ reaches the destination** or an intermediate node with a fresh enough route, **the destination** intermediate node **responds by unicasting a route reply (RREP) packet back to the neighbor from which it first received the RREQ**

- As the RREP is routed back **along the reverse path**, **nodes** along this path **set up forward route entries** in their route tables which point to the node from which the RREP came.

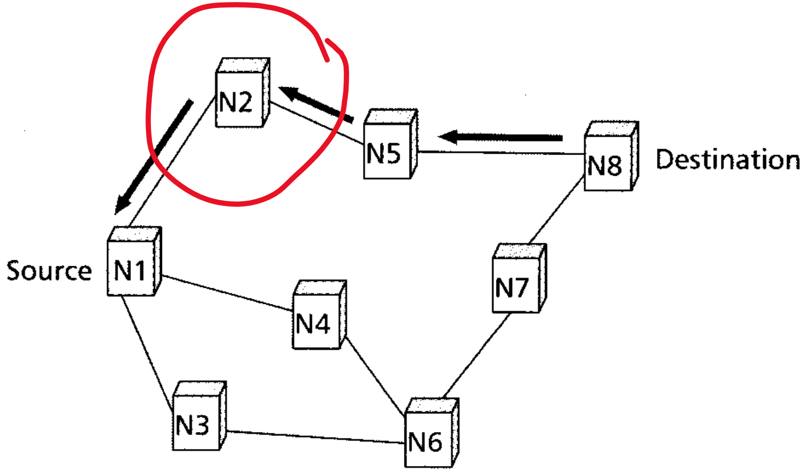
AODV



To N1 through N6

(a) Propagation of the RREQ

To N8 through N5



(b) Path of the RREP to the source

- Intermediate nodes learn reverse paths
- Intermediate nodes learn forward paths

Proactive vs reactive

- Which approach is best?

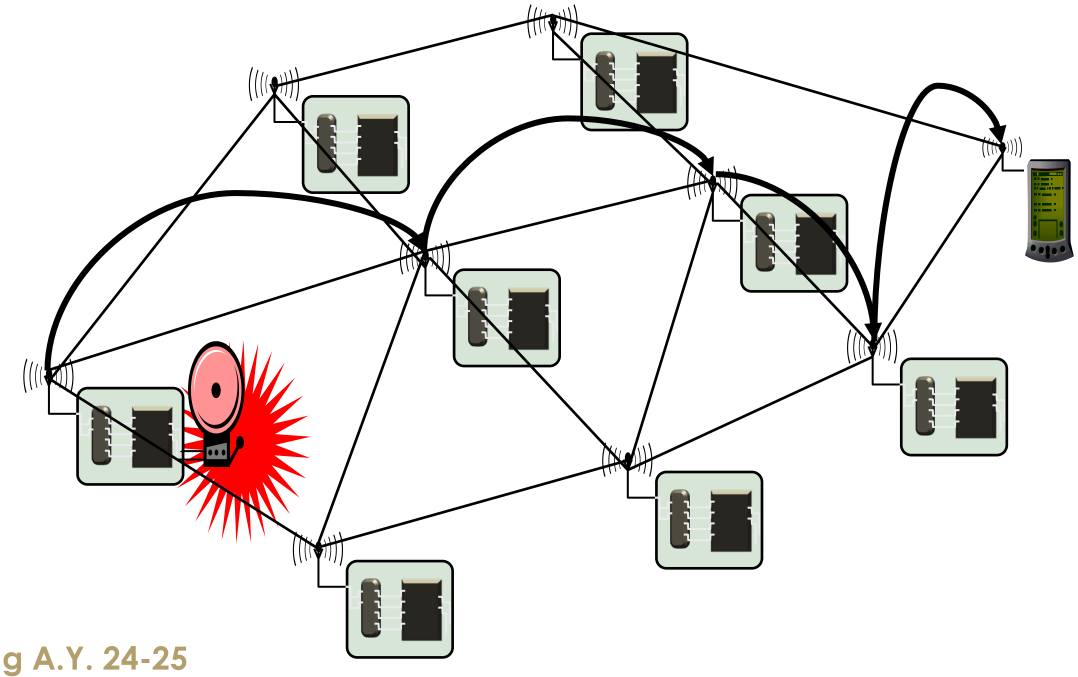
- Proactive approach is fast but involves overhead
- Reactive approach generate much less overhead but it is slower

Geographic Routing

- Routing tables contain information to which next hop a packet should be forwarded
  - Explicitly constructed
- Alternative: **Implicitly infer this information from physical placement of nodes**
  - Position of current node, current neighbors, destination known – send to a neighbor in the right direction as next hop
  - **Geographic routing** (also **position-based routing**)
- Options
  - Send to any node in a given area – geocasting
  - Use position information to aid in routing – position-based routing
    - Might need a location service to map node ID to node position

Basics of Position-based Routing

- “**Most forward within range r**” strategy
  - Send to that neighbor that realizes the most forward progress towards destination
  - NOT: farthest away from sender!



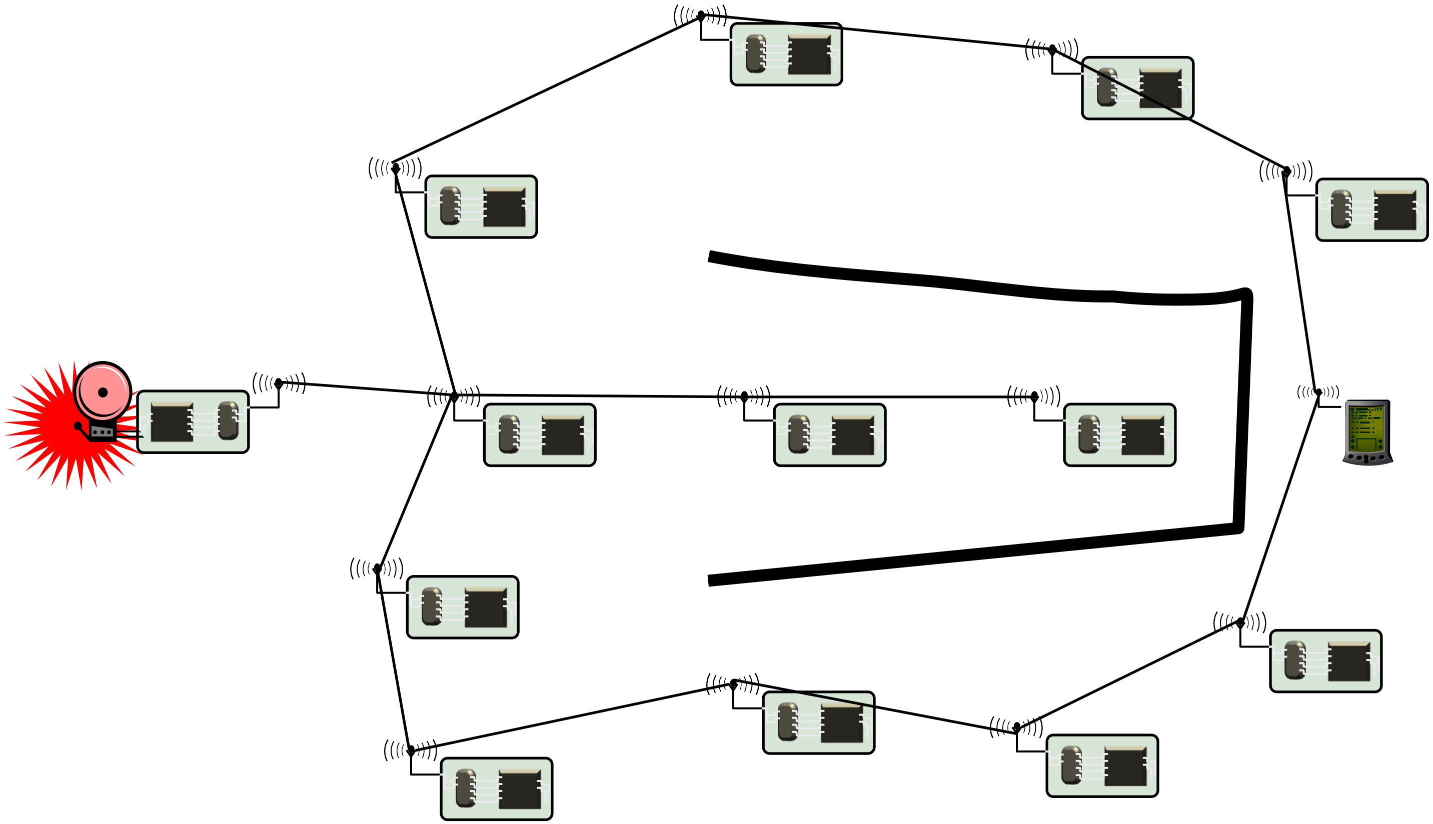
Alternative Strategies

- **Nearest node with (any) forward progress**
  - Idea: Minimize transmission power

- **Directional routing**
  - Choose next hop that is angularly closest to destination
  - Choose next hop that is closest to the connecting line to destination
  - Problem: Might result in loops!

Problem: Dead ends

- Simple strategies might send a packet into a dead end



Routing in WSN

- There are many other protocols
- What is the best solution?
- Depends on the network characteristics
  - Mobility
  - Node capabilites
- Geographic approach allows to save more energy
- Proactive approach is fast but involves overhead
- Reactive approach generate much less overhead but it is slower

What if

- The network topology is continuoulsy changing..

- Consider a network of drones flying over an area (topology continuously changes and nodes can be often disconnected)

- Which approach would be best?