Ad Hoc Networking

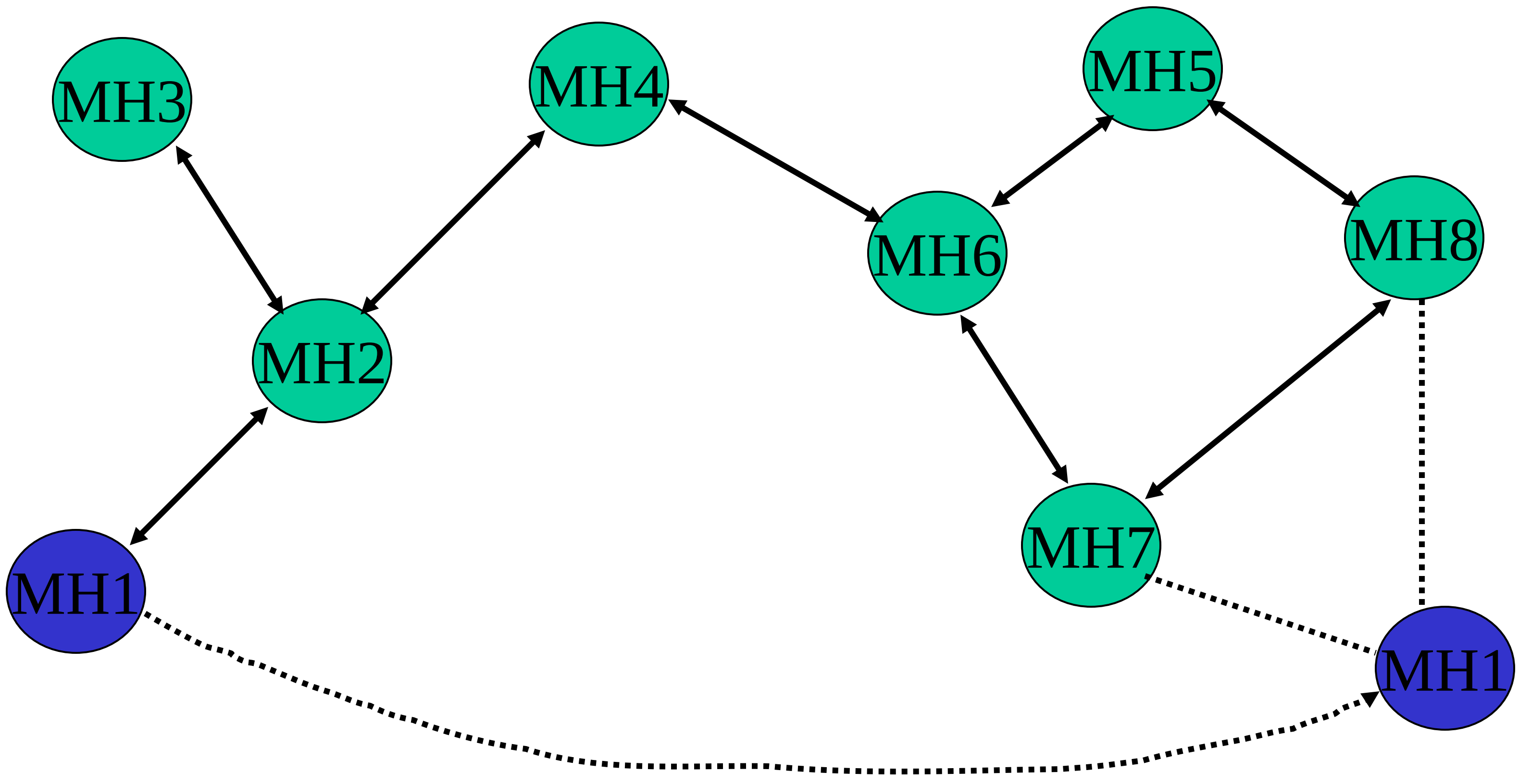
Main source

Ad Hoc Networking, C. E. Perkins, Addison-Wesley

What is an ad hoc network?

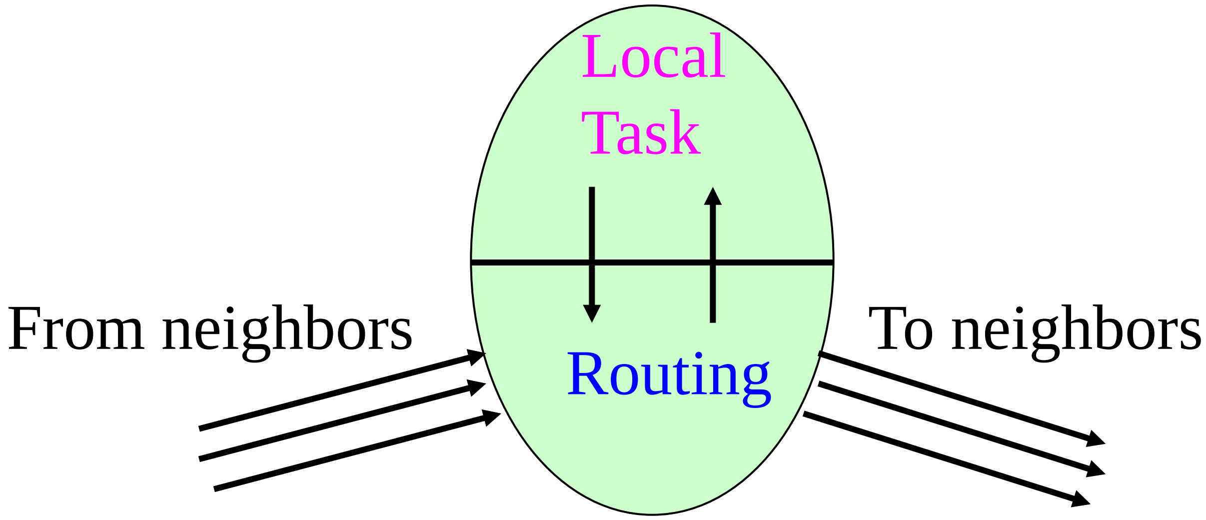
- A short lived network just for the communication needs of the moment
- Infrastructure-less network
- Commercial Applications
  - Emergency services
    - Damaged infrastructure
    - Difficult to set up infrastructure
  - Vehicular networks

Model of operation



A node in an ad hoc network

Two main components of a node



Routing methods

- Recall: 2 routing algorithms on fixed networks
  - Distance-Vector (DV): Routing Information Protocol (RIP)
  - Link State: Open Shortest Path First (OSPF)
- Destination-Sequenced Distance Vector (DSDV)
- Dynamic Source Routing (DSR)
- Location Aided Routing (LAR)
- Ad-hoc On-demand Distance Vector (AODV)

Two routing algorithms for fixed networks

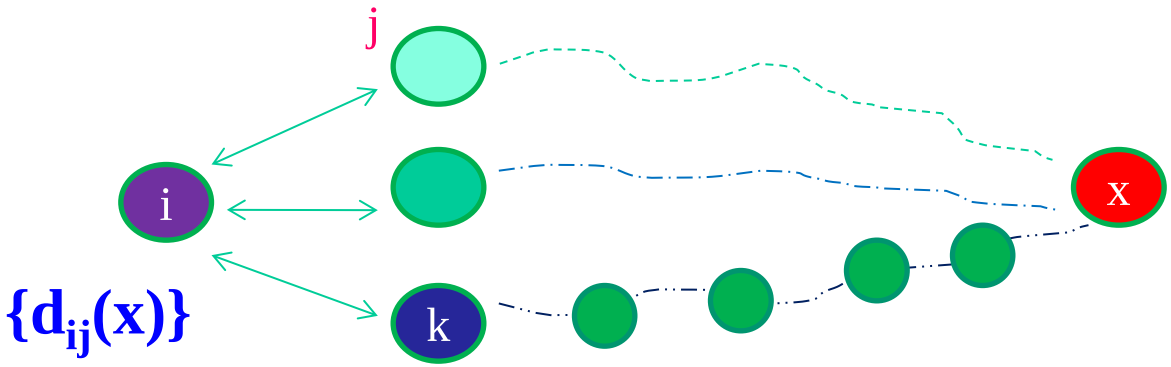
Distance Vector (RIP: Routing Info. Protocol)

Link State (OSPF: Open Shortest Path First)

Distance Vector

– Based on: Distributed Bellman-Ford algorithm

- Node **i** maintains, for each dest **x**, a set of distances $\{d_{ij}(x)\}$, where **j** is a neighbor of **i**.
- Node **i** treats neighbor **k** as a next hop for a packet for **x** if $d_{ik}(x) = \min_j\{d_{ij}(x)\}$.
- To keep $\{d_{ij}(x)\}$ up to date,
  - Node i monitors the **cost** of its outgoing links.
  - Periodically broadcast your estimate of the shortest distance to every other node.



Distance Vector (contd.)

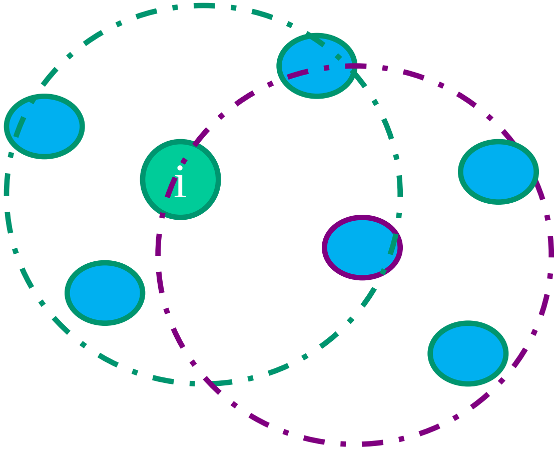
- Advantages: efficient, easier to implement, less storage.

- Drawback: short/long-lived **loops**.
  - Two-node and three-node instability problems

- Ad hoc networks ➔ rapid topological changes ➔ loops.

Link State

- Each node has its own **view of the net topology** and **link costs.**



- For **consistent view**
  - Each node periodically **broadcasts** outgoing link **costs.**
  - Based on received info, nodes **update** their **view** of net.
- Apply **shortest-path algorithm** to **choose** its **next hop** for each destination.

DSDV Routing Algorithm

(Enhancement of Distance Vector routing)

DSDV Protocol Overview

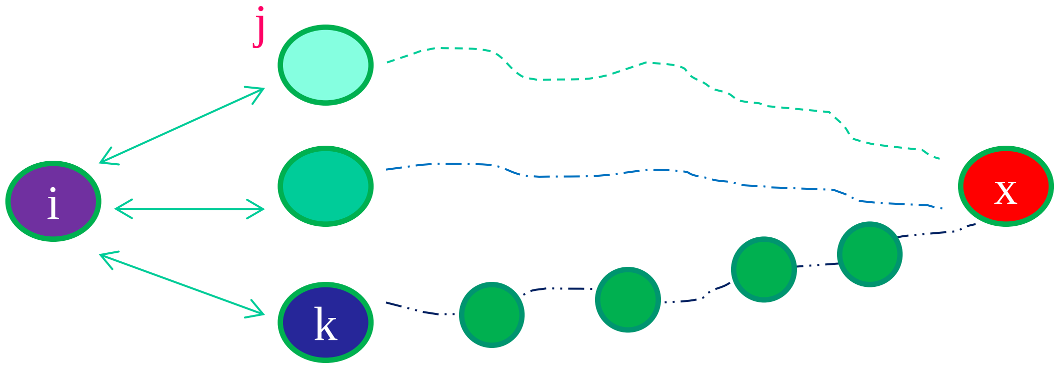
- **Each node** constructs and maintains a **route table.**
- Route table entry of node i:

Destination (node j)	Next hop (NH)	# of hops	**Sequence #**

Sequence #: originated by the destination node.

Seq #: Sometime before, node j made a broadcast with a “seq #” that was propagated to i via NH.

DSDV – Sequence number



Overview (contd.)

- To **maintain consistency** of route tables, each node
  - Periodically transmits **updates.**
  - Transmits **immediately** when **significant** new info is available.
- Update information consists of
  - **Which nodes** are **accessible** from the node.
  - **Number of hops** necessary to reach them.
  - (Sequence #)

Route Advertisement

- Structure of a packet **broadcast** by each node

Seq #1 (new)	Dest (j)	# of hops to dest	Seq #2

…

Generated by broadcasting node i

Seq #2: Sometime before, node j made a broadcast with a sequence # that was propagated to i via NH and i chose NH to reach j.

Route selection criteria

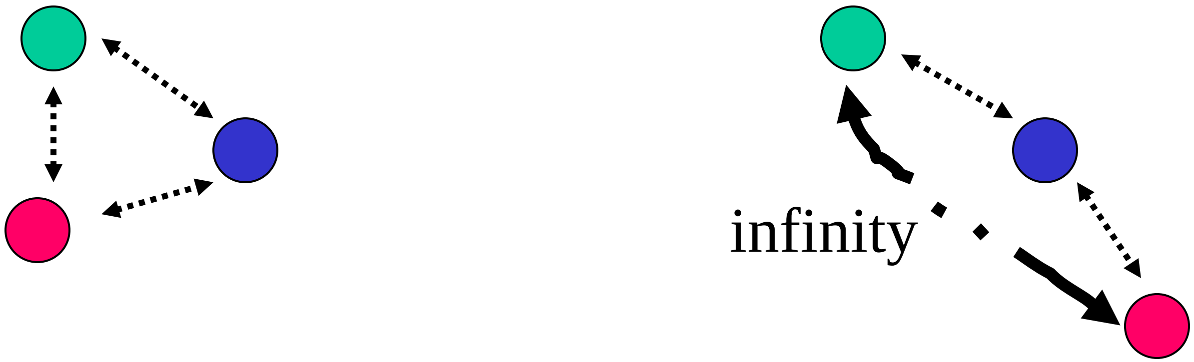
- When a node receives info. about a new route to a node
  - Compare it with existing info. of a route to the same node.
  - Use a route with a **more recent sequence number**.
  - Use a route with an **identical** sequence number, if it has a **better metric**.
    - » The existing route is discarded or saved as a less preferable route.
  - Increment the metrics for routes chosen from the newly received broadcast info by one hop.

Responding to **topology change**

- What is a topology change?
  - Discover a new node from a received advertisement
  - Learn about a broken link with a neighbor.
    - » How: No communication from the neighbor for a long time.
    - » Update the route table as follows:
    - » Assign an **infinity** metric to that destination.
    - » Assign a ***new*** sequence number. (last seq # received from that node + 1) 🡰 As if the node told us that it moved away
- What to do next because of this **substantial** change:
  - Disclose this new info in a **broadcast** packet.

Responding to topology change

- When a node **receives** an **infinity** metric and it has a "=" or ">" sequence # with a **finite** metric
  - Trigger a route update broadcast.



Efficient Response to topology change

- To **reduce** the amount of info in broadcast packets, two types of broadcasts are used:
  - **Full dump**: All available routing info., periodic
  - **Incremental**: Info changed since last full dump

Dynamic Source Routing (DSR)

Note: It is the responsibility of the source node to find and maintain routes.

Source Routing

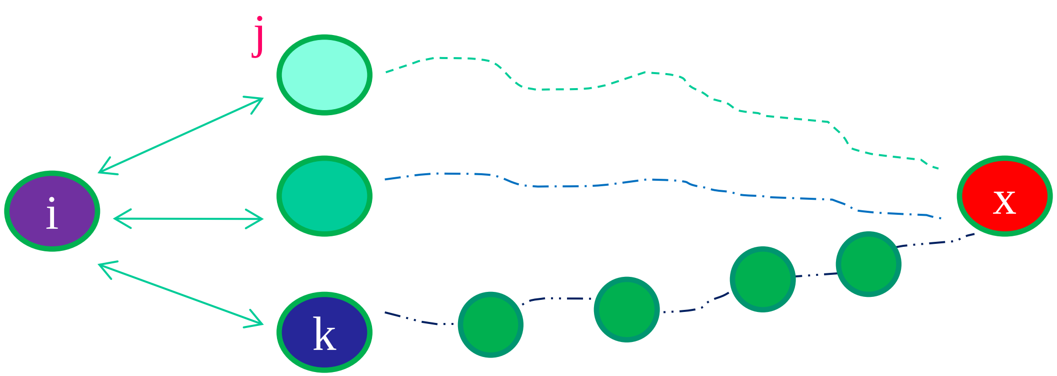
- The source of a packet determines the complete sequence of nodes (i.e. the route) for the packet.
- The route is put in the packet's header.

- There is no periodic route advertisement.
- When a node needs one, it dynamically determines one based on
  - Cached information
  - Results of a route discovery protocol

Basic Operation

- To send a packet, construct a source route in the packet header.
- The sender transmits the packet to the first node on the **route.**
- When a node receives a packet, if it is not the final destination, it simply transmits the packet to the next node on the route.
- Each node maintains a **route cache** in which it caches source routes that it has **learned**.
- When a source wants to send a packet to another node, it first checks its route cache:
  - If a route is found, use the route.
  - If no route is found, discover one.

Learning routes from others



Route Discovery (find a route)

- A node initiating a route discovery broadcasts a **route request** (RR) packet (to all its neighbors).
  - RR header: Sequence#, source, destination.
- A node receiving an RR *is NOT* the destination:
  - The node further broadcasts it after inserting its ID in the RR header. ➔ Incrementally construct a route.
- A node receiving an RR *is* the destination:
  - It sends a **route reply** packet containing the route (sequence of nodes from the source to the destination).
  - A route reply takes the reveres of the defined path just discovered by the RR. 🡐 Assumption: Symmetric communication between nodes.

Route Discovery (efficiency)

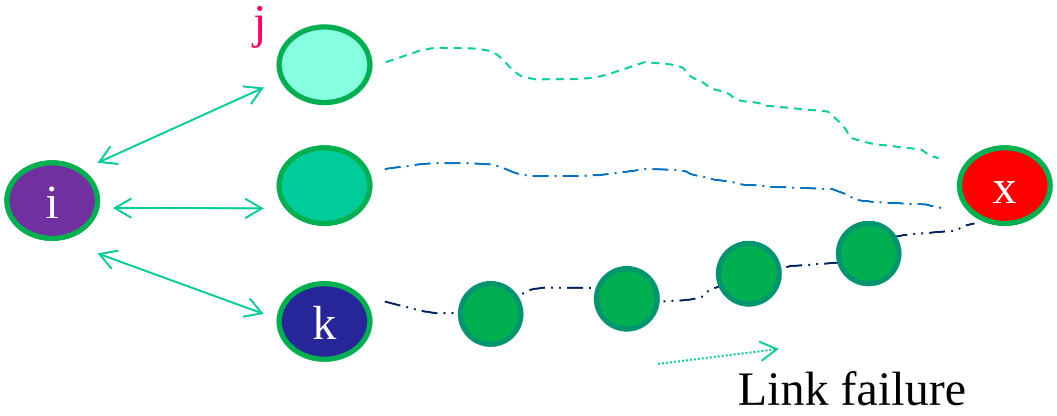
- Each node maintains a **list** of

  <source ID, RRseq. #> for **all** sources
- When a node receives a route request, it does this:
  - If the pair <source ID, RRseq. #> from the RR is already on the above list, discard the packet.
  - If the node's address is already in the route record in the request, discard the packet.
  - If the target of the request matches this node's address, send a route reply.
  - Else, insert this node's own address in the route record, and rebroadcast the packet.

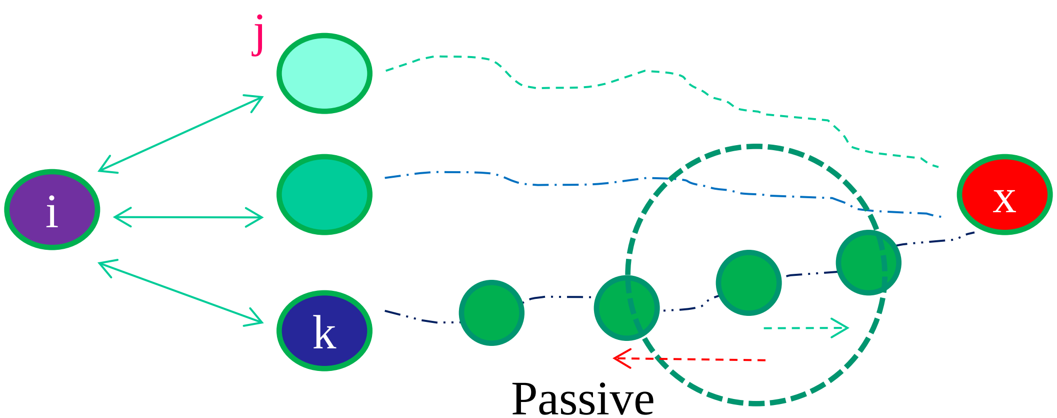
Route Maintenance

- While a route is in use, a route maintenance procedure monitors the operation of the route and informs the sender of any routing error.
  - Wireless networks generally utilize hop-by-hop ACK. If the data-link layer in a node on a route reports a problem, the node sends a route error packet to the original sender.
  - If link-level error detection is not available, you may use the idea of passive ACK (i.e. being able to hear that node transmitting the packet again.)
  - Worst case: Ask for an explicit ACK.

Route maintenance



Route maintenance



Passive
ACK:

Location Aided Routing (LAR)

1. Obtain location information from a GPS receiver

2. **Utilize** location information to improve the performance of routing protocols

Motivation

- Host mobility causes topology change
  ➔Task of discovering and maintaining routes is non-trivial

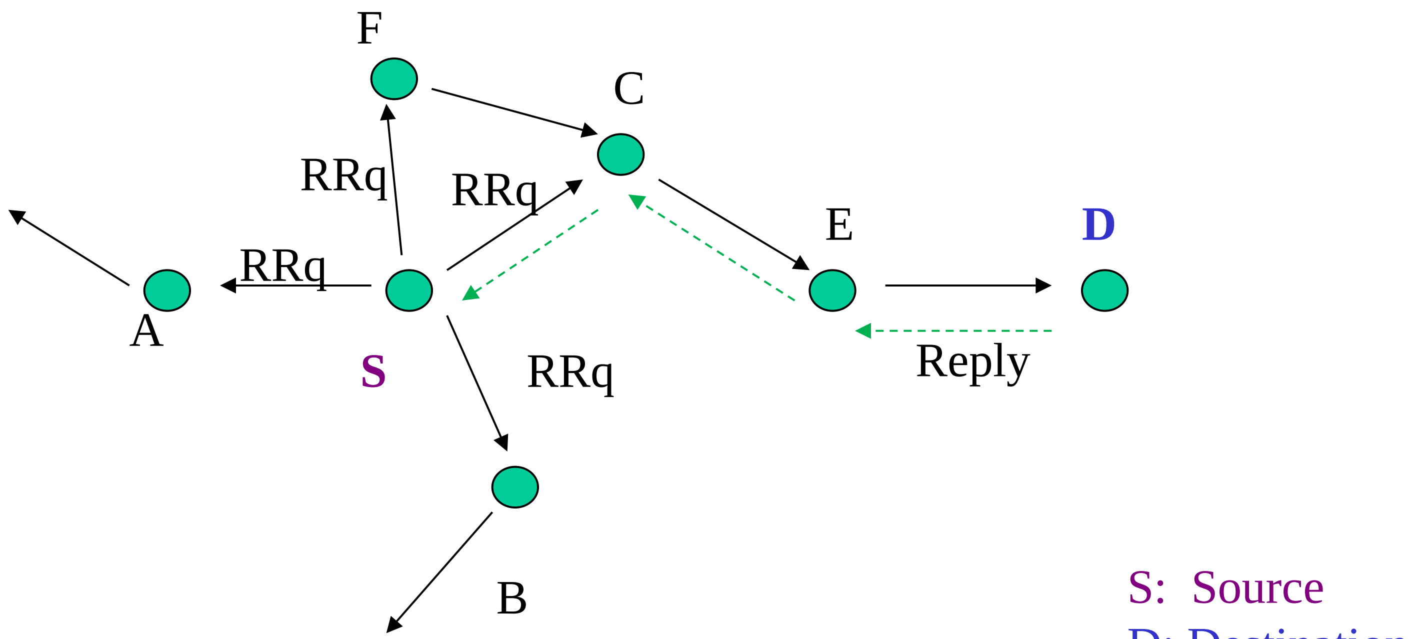
- Flooding (ex. DSR) is a brute force way of discovering routes.

- LAR: Reduces the full impact of flooding by forwarding route discovery messages to a selected region by using location information.

Route discovery using flooding

- A sender broadcasts a route request (RRq) to all its neighbors
- If a node *is NOT* the destination of an RRq, it broadcasts the RRq.
  - Subsequent copies of an RRq are not rebroadcast: save recent <Sender, sequence #> pairs.
- If a node *is* the destination of an RRq, it sends a reply back. The reply takes the reverse path taken by the RRq.
- Timeout: If the sender does not receive a reply within a certain timeout period, it reinitiates the process.

Route discovery using flooding



LAR: Preliminaries

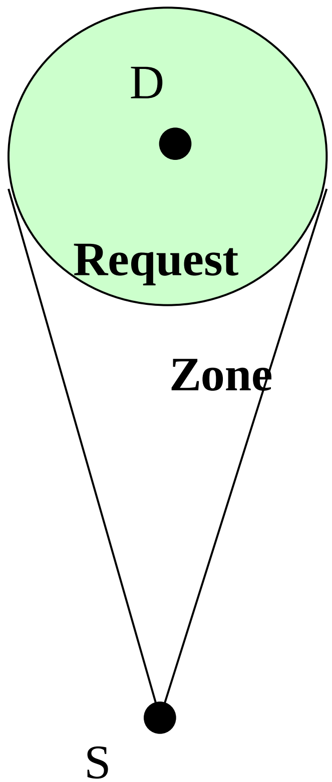
- Assumptions
  - Node S needs to find a route to D.
  - S knew D's location "L" at time t0 (in the past)
  - Current time is t1.

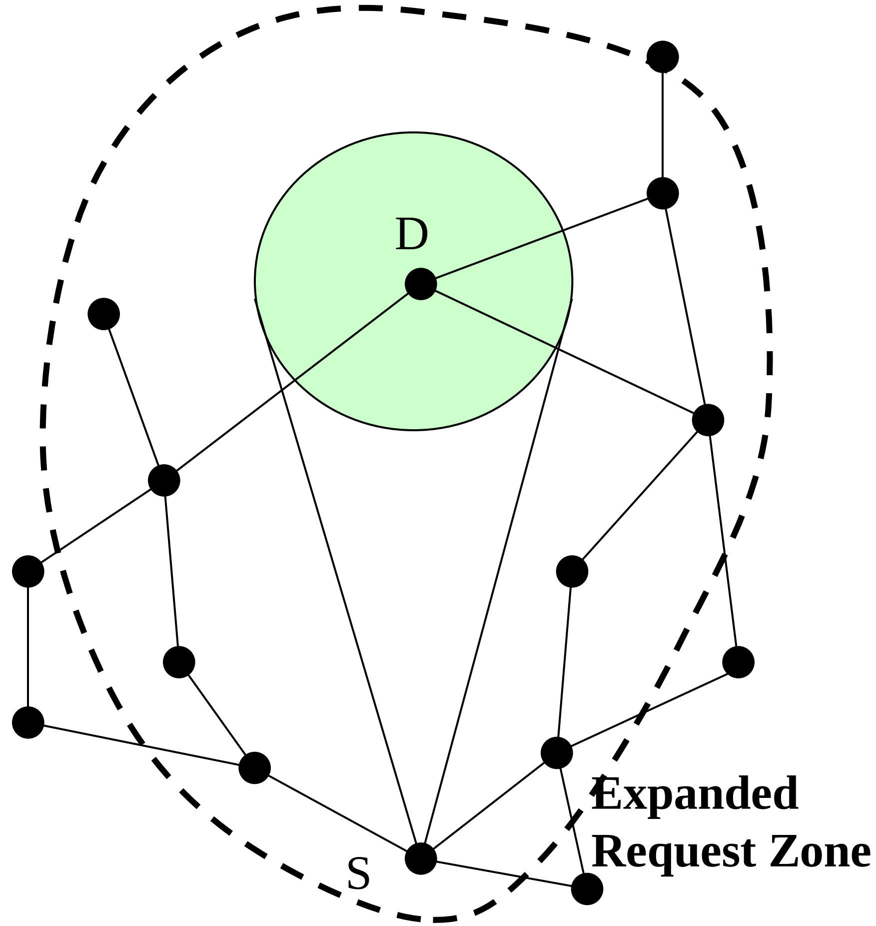
- **Expected Zone** of D from the viewpoint of S at t1
  - This is the region that S expects to contain D at time t1.
    - Zero knowledge: the entire network area is the expected zone.
    - Having more information regarding mobility of a destination node can result in a smaller expected zone.

LAR: Preliminaries

- **Request Zone**
  - Node S defines a <u>request zone</u> for the route req.
  - To increase the probability that the route request will reach node D, the <u>request zone includes the expected zone.</u>
  - Additional regions must be included in the request zone so that S and D belong to the request zone.
  - A node forwards a route request only if it lies in the request zone. 🡐 Cost reduction
  - If a route is not discovered within a suitable time period, S reinitiates route discovery with an EXPANDED request zone.

Request Zone





Tradeoff between <u>probability</u> of finding a route and <u>discovery overhead.</u>

The LAR Algorithm

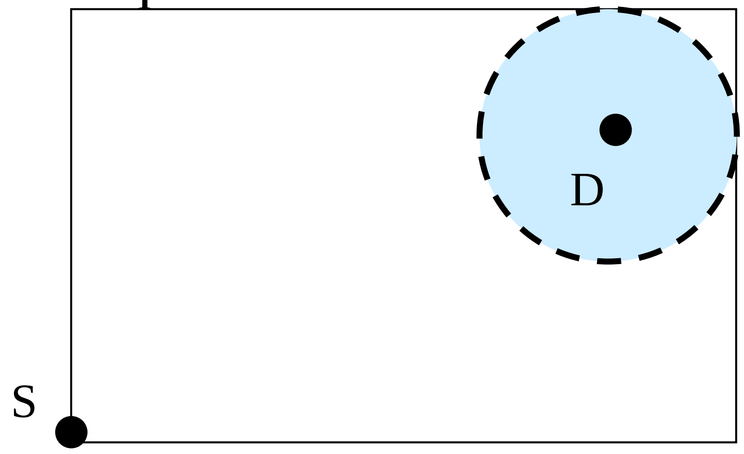
- It is the FLOODING algorithm.
- A node that is NOT in the request zone does not forward a route request to its neighbors.

Two ways of computing a Request Zone

- LAR Scheme 1
- LAR Scheme 2

LAR Scheme 1

- A request zone is the smallest rectangle that includes the current location of S and the expected zone of D, such that the sides of the rectangle are parallel to the X- and Y-axis.
- The source explicitly specifies the request zone in its route request.

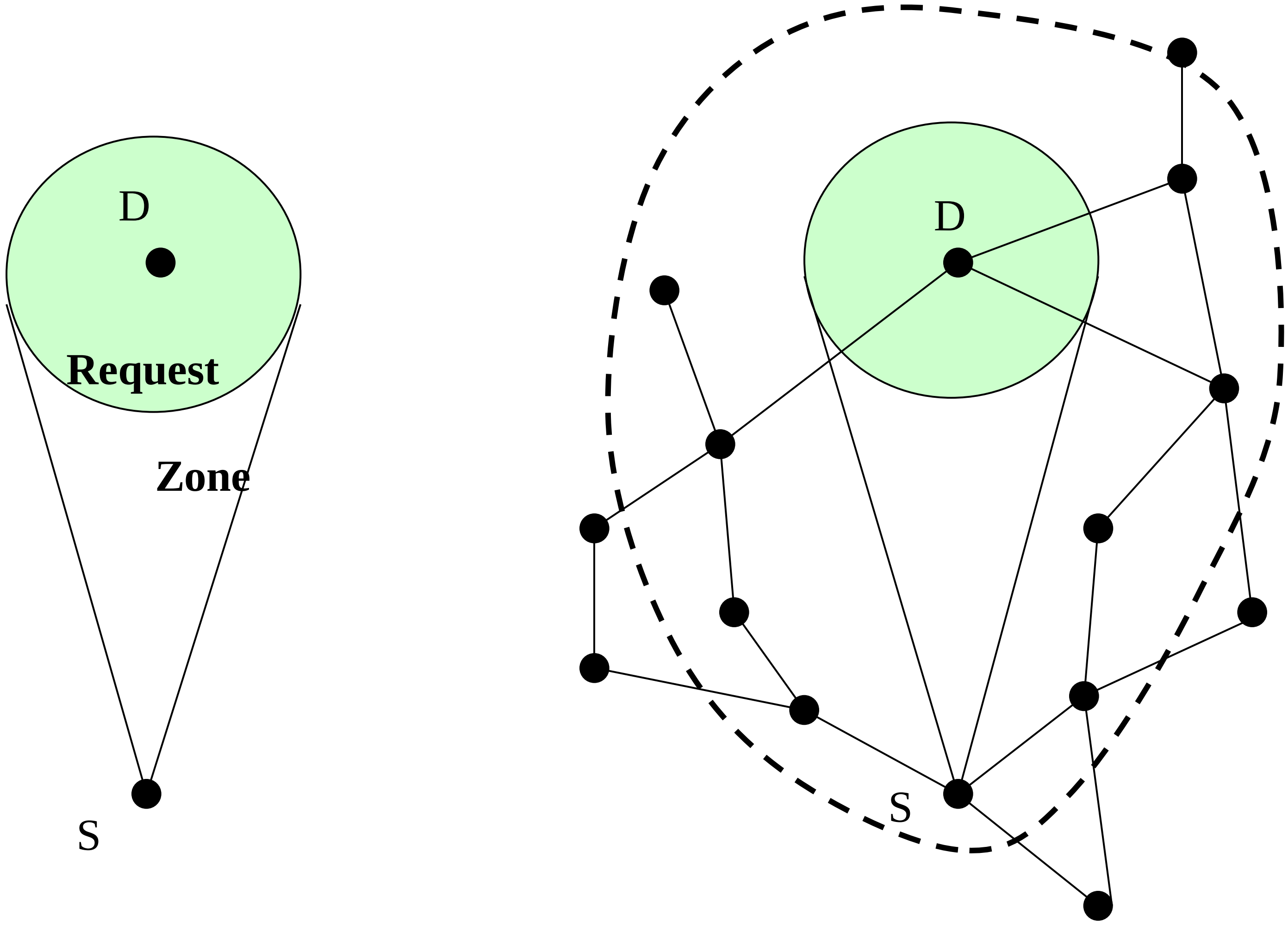


When D receives a Route request, it sends a route reply which includes its **current location** and the **current time**.

LAR Scheme 2

- Sender S includes these info <u>with a route request</u>.
  - $DIST_S$: Distance of S from D
  - (Xd, Yd): Coordinates of D
- When a node I receives the route request from S, I calculates its distance $DIST_I$ from D and acts as follows:
  - If $DIST_S + \delta >= DIST_I$ I forwards the request to its neighbors <u>after replacing</u> $DIST_S$ with $DIST_I$.
  - If $DIST_S + \delta < DIST_I$ I discards the request.
- Idea: A node (I) forwards a request forwarded by another node (S), if I is "at most δ farther".

Request Zone

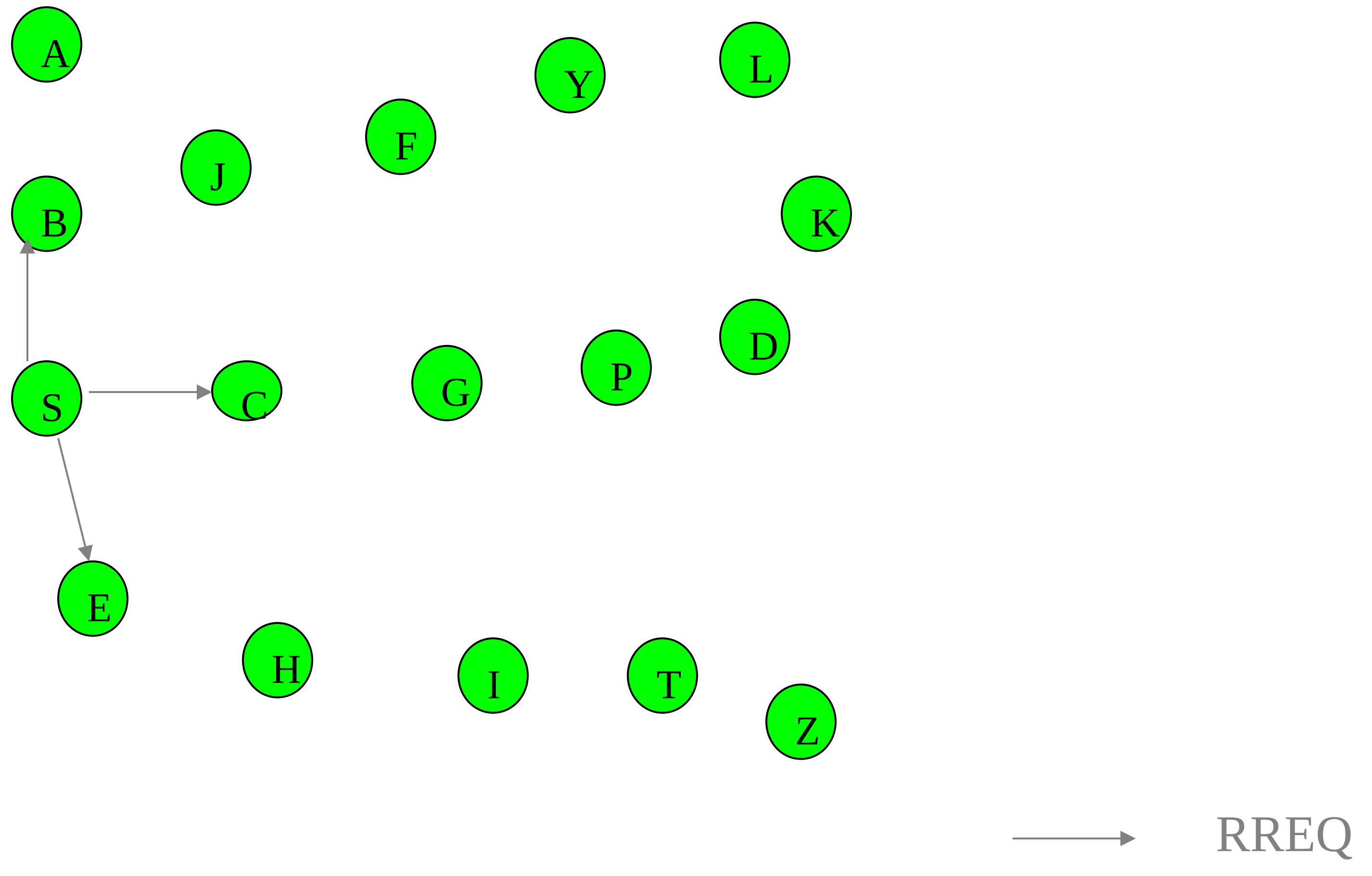


AODV: AD HOC ON-DEMAND DV ROUTING PROTOCOL

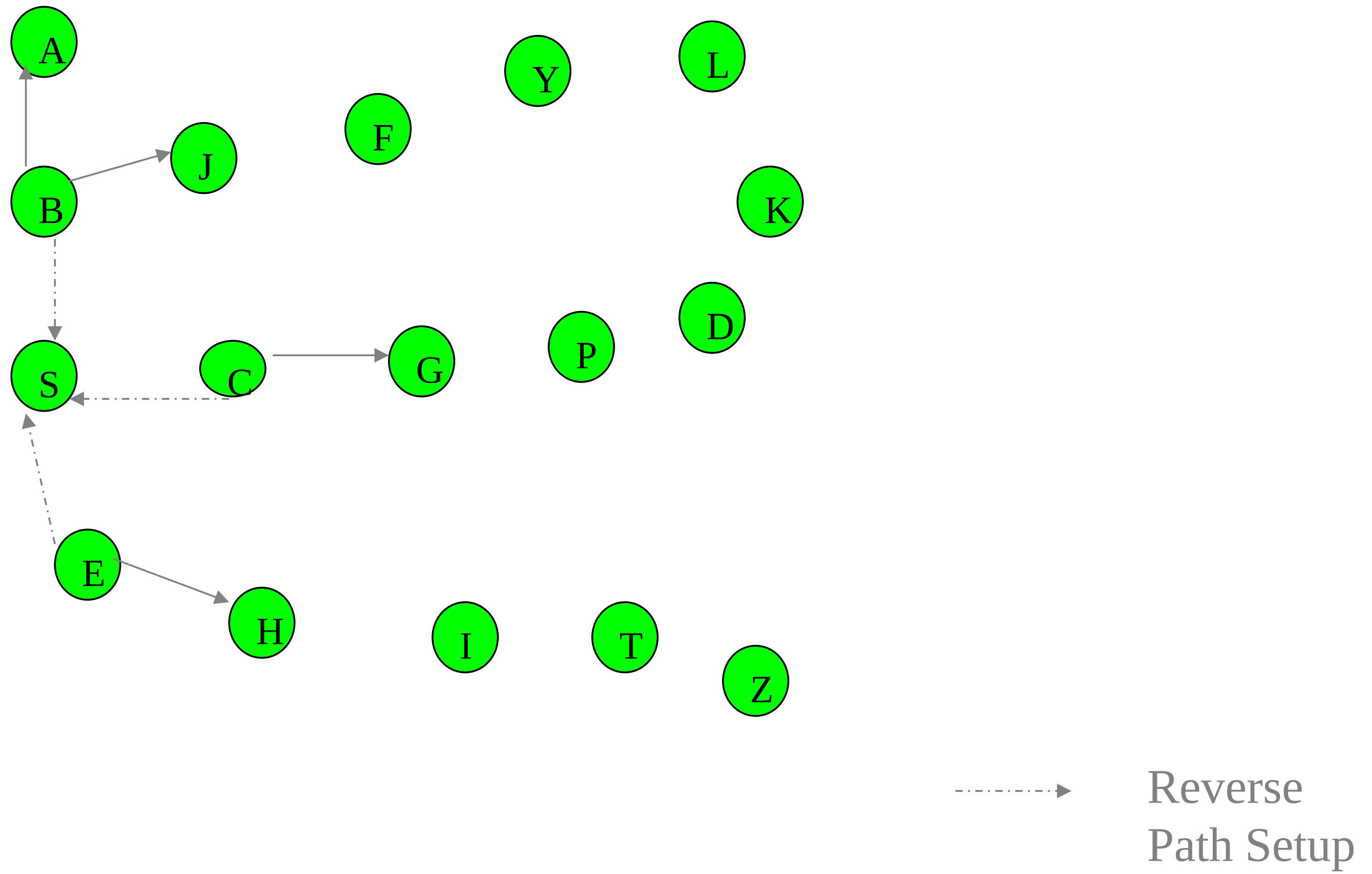
AODV: Ad Hoc On-demand DV

- Does not attempt to maintain routes from every node to every other node.
- Routes are discovered on an **as-needed basis** and are maintained as needed.
- The protocol uses three kinds of messages:
  - RREQ, RREP, RERR
- **Need:** A node wants to send a message to a destination
  - If a route exists in its RT to that dest., forward the pkt to the next hop; otherwise, start a **route discovery process**.

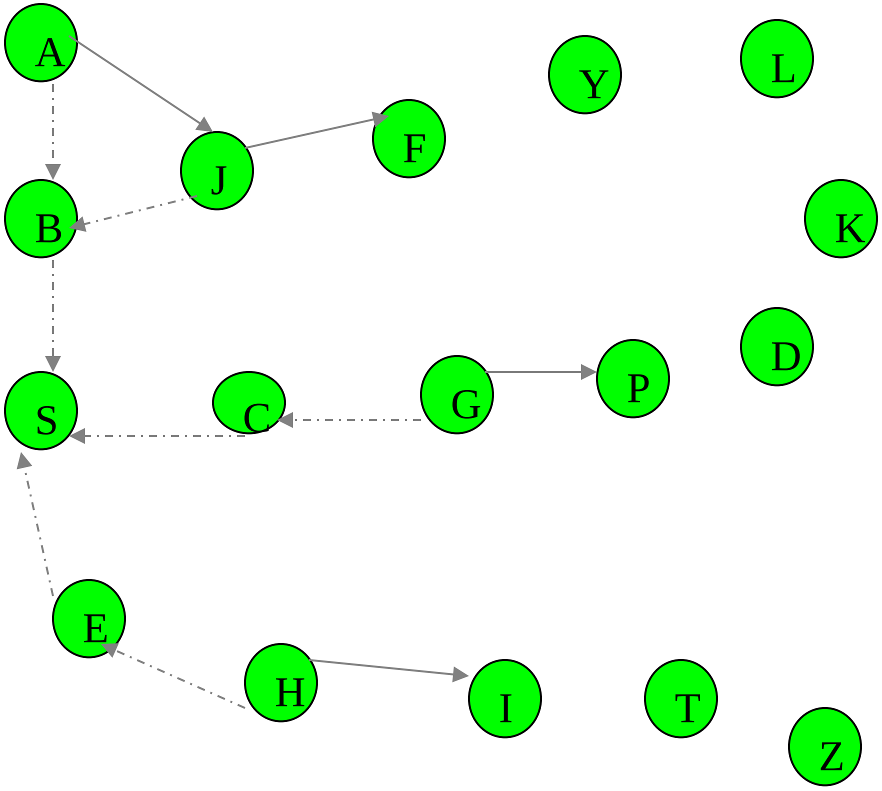
AODV (Example)



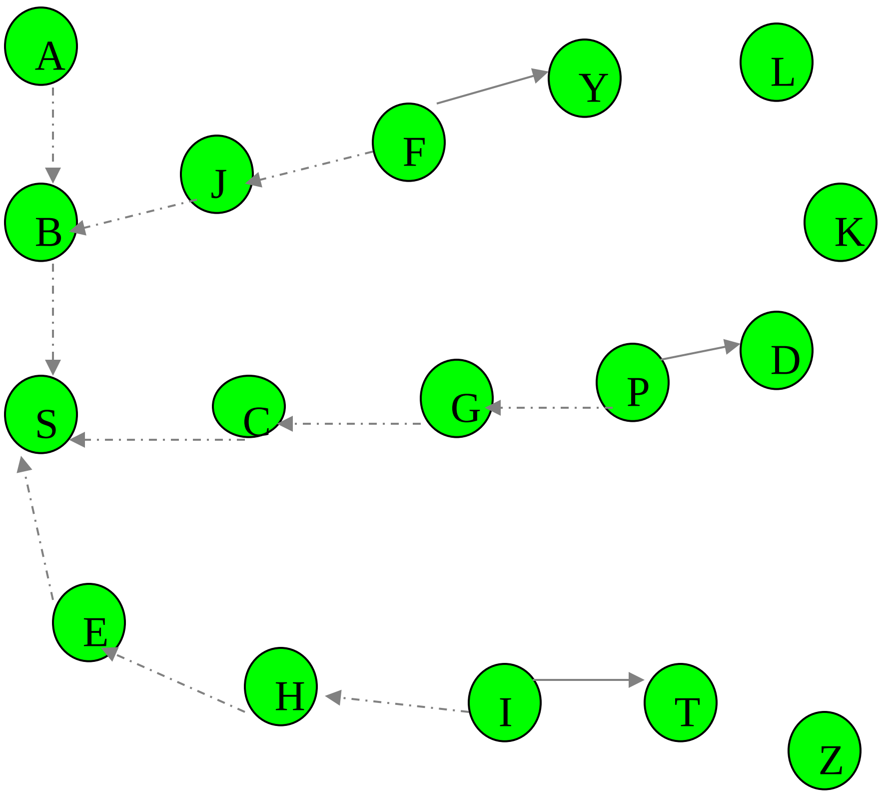
AODV (Example)



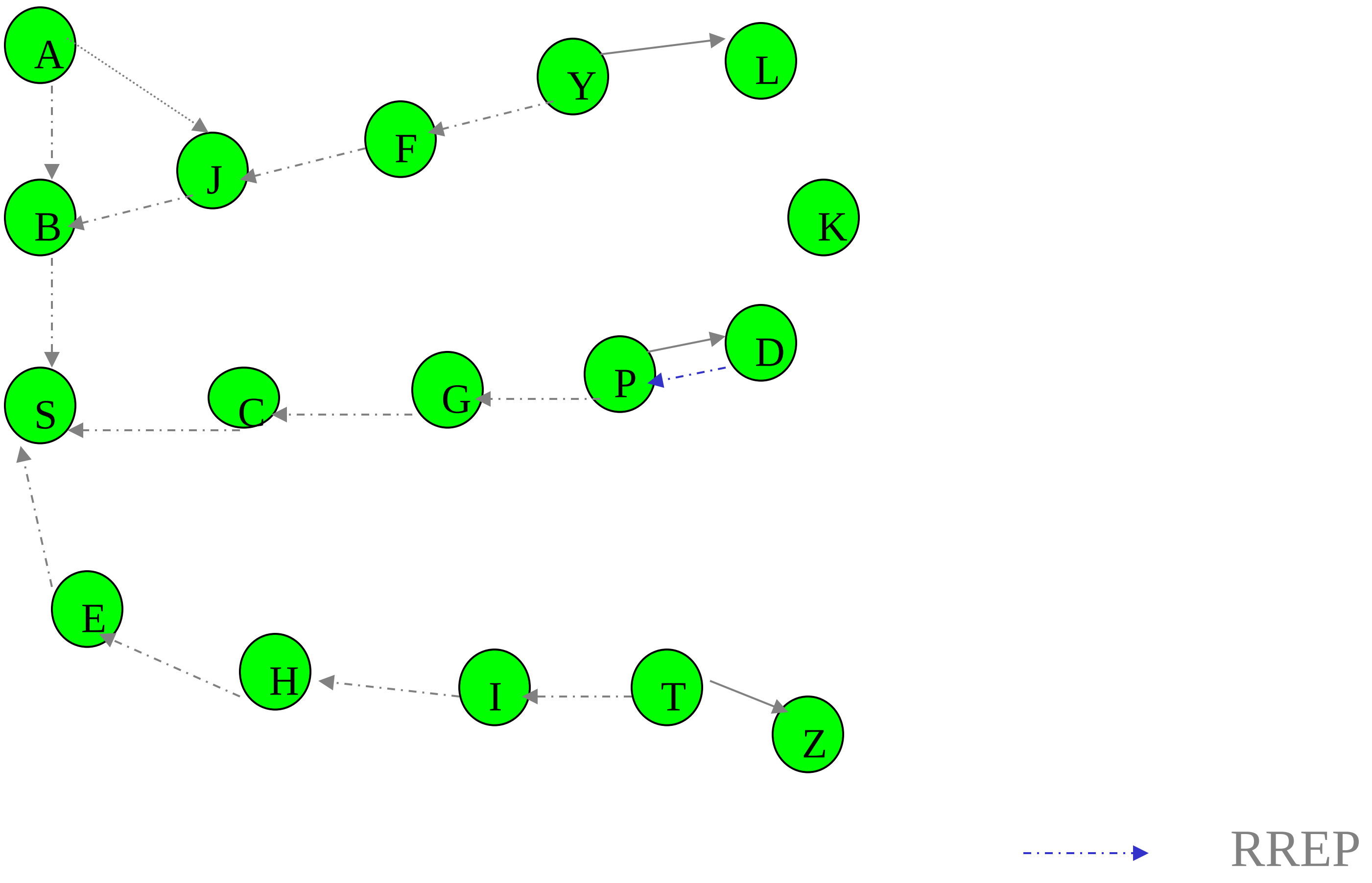
AODV (Example)



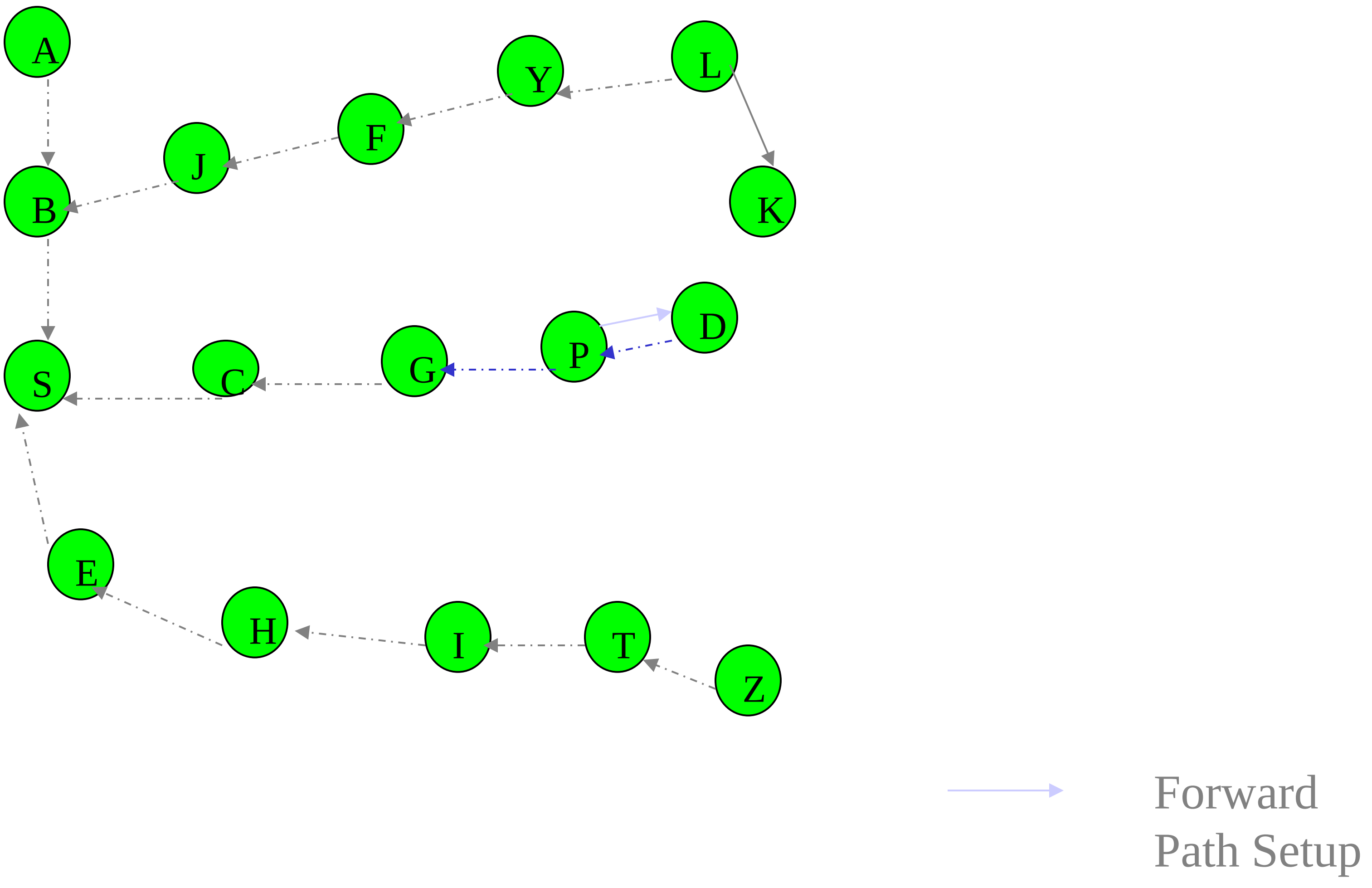
AODV (Example)



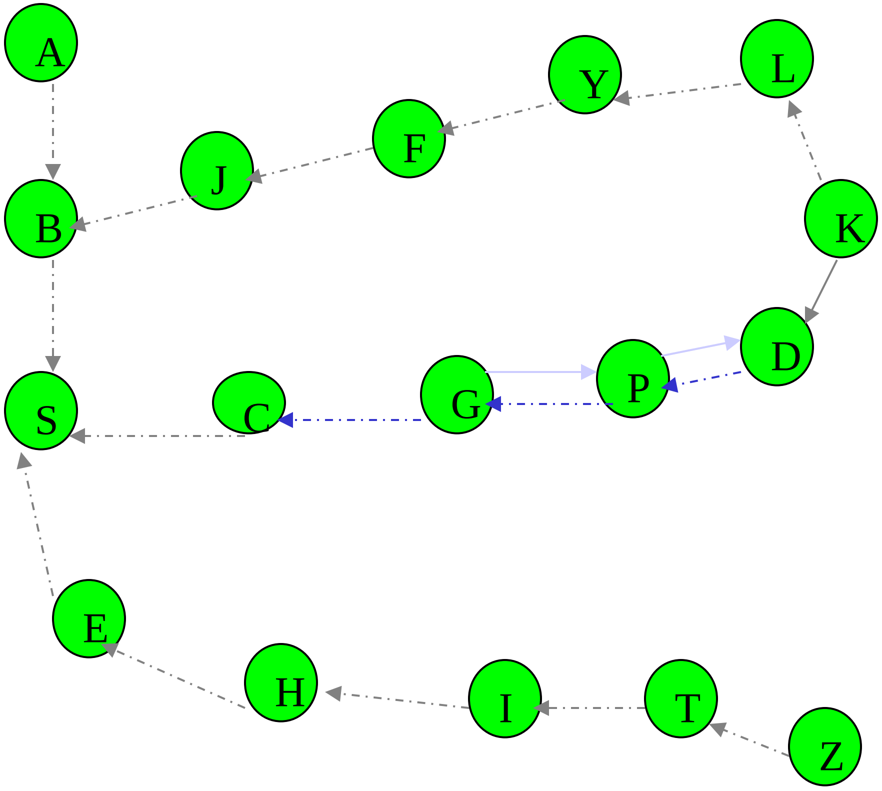
AODV (Example)



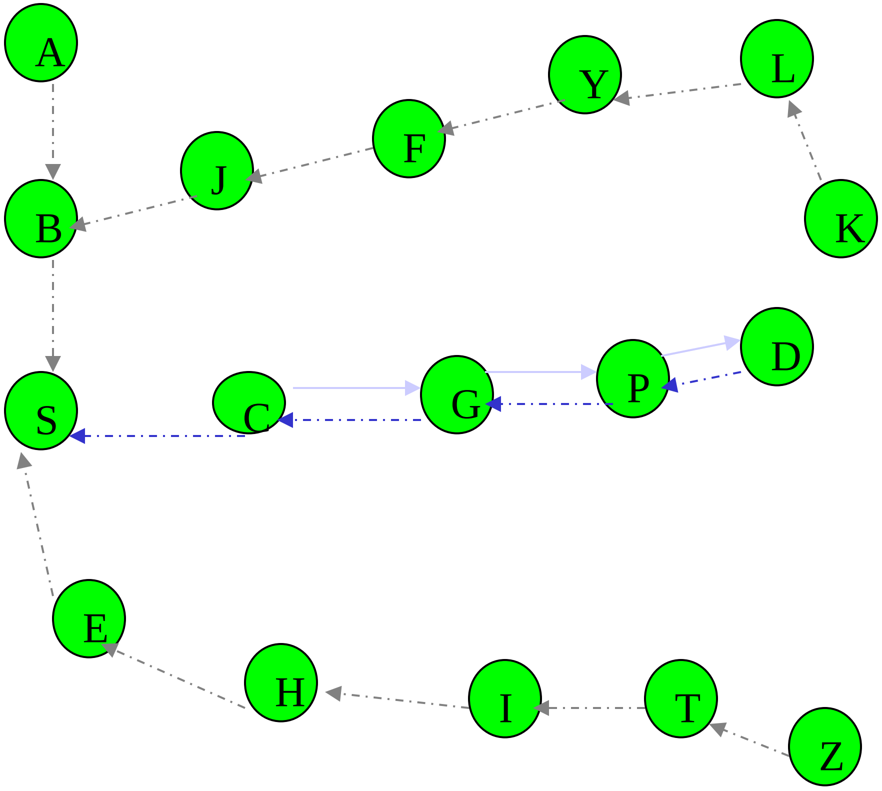
AODV (Example)



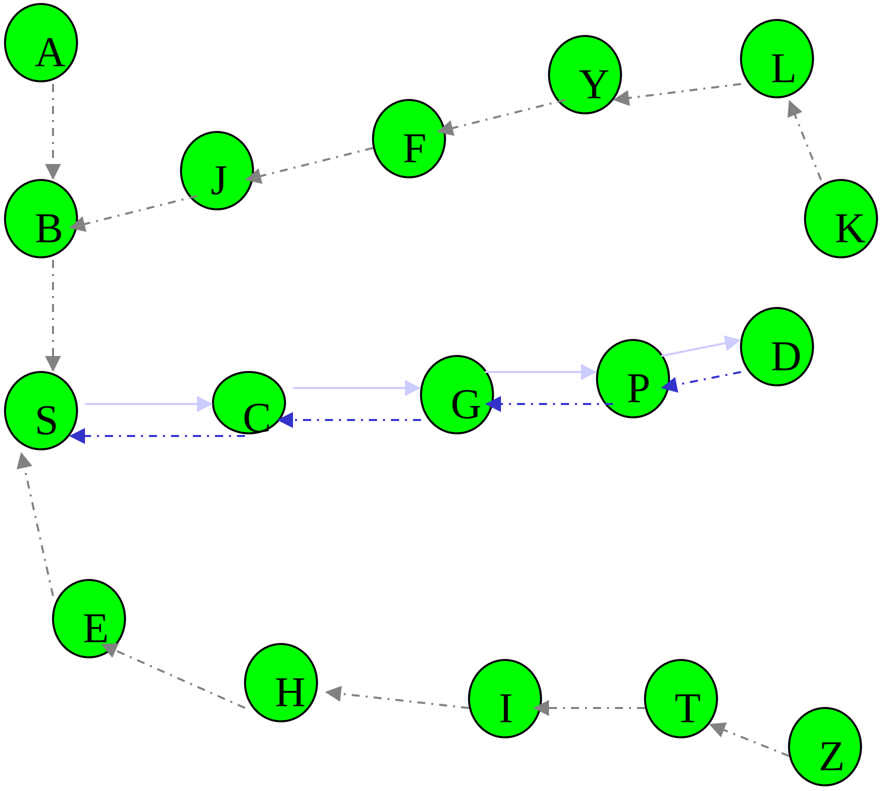
AODV (Example)



AODV (Example)



AODV (Example)



AODV: Route Discovery Process

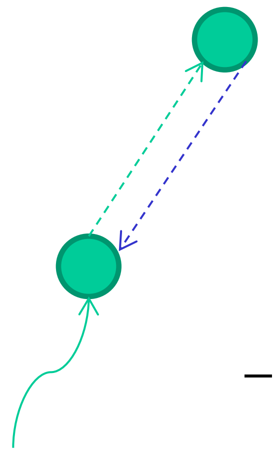
- Create a RREQ(BcastID, SrcIP, CurSeq#, DestIP, LastKnownSeq#)
  - BcastID is incremented for each **new** discovery.
  - (BcastID, SrcIP) uniquely identifies a discovery process.
  - Note: Every time a node sends out any type of message, it increases its own sequence number: **CurSeq#**.
- <u>Broadcast</u> RREQ and start a timer.
- When a node receives a RREQ, it checks if it has **seen** the RREQ before. If YES, discard the RREQ, else **process** it as follows.
- Processing a RREQ (see the next slide.)

AODV: Route Discovery Process (Processing RREQ)

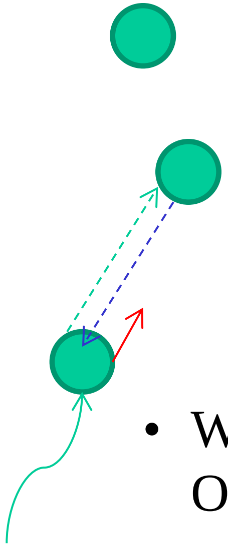
- The node sets up a **reverse route** entry for the source node in its RT. [Reverse Route: SrcIP, SrcCurSeq#, # of hops to Src, IP of the neighbor that gave it RREQ, Lifetime]
- To respond to the RREQ: **2 conditions** must hold.
  - The node must have an **unexpired entry** for the Dest.
  - The **Seq#** associated with that Dest must be **at least as great as** that indicated in the RREQ.
- (The above conditions **prevent the formation of loops** by ensuring that the route returned is never old enough.
- If the node satisfies the above 2 conditions, it responds by unicasting a RREP back to the source (See 6.2.3.)
- If the node is unable to satisfy the RREQ, it increments the RREQ's hop count and then broadcasts the RREQ to neighbors.
- **Naturally,** the Dest is always able to respond, eventually.

AODV: Route Discovery Process

- **Forward Path Setup**
  - **Current Enough Route**: Any route with a Seq# not smaller than that indicated in the RREQ.
  - When a node determines that it has a route current enough to respond to the RREQ, it creates a RREP.
    - » **If** the **Dest** is responding, it places its **CurSeq#** in RREP, initializes HopCount = 0, and initializes the LifeTime field of the route.
    - » **If** an **intermediate node** is responding, it places its record of the DestSeq# in the pkt, sets the HopCount to its distance to Dest, and calculates the amount of time for which its RT entry will still be valid.
  - It then unicasts the RREP toward the Src, using the node from which it received the RREQ as the next hop.



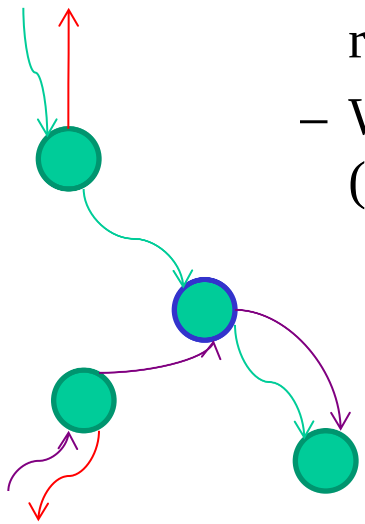
AODV: Route Discovery Process



- **Forward Path Setup (Contd.)**
  - When an intermediate node receives the RREP, it sets up a **forward path** entry to the Dest in its RT. The entry contains DestIP, NeighborIP who gave RREP, DestSeq#, HopCountToDest, LifeTime (as in RREP).
  - Note: Each time the route is used, its LifeTime is updated. If it is NOT used within the specified LifeTime, it is deleted.
- When a node receives a RREP for a given Dest from more than ONE neighbor:
  - It forwards the FIRST RREP and forwards a later RREP only if that RREP contains a larger DestSeq# or a smaller HopCount. Otherwise discard the RREP.
  - A source can set multiple routes to the same Dest.

AODV: Route Discovery Process

- **Route Maintenance**
  - Once a route has been discovered for a given Drc/Dest pair, it is maintained as long as needed.
  - Movement of nodes affect only the routes containing those nodes. (Such a path is called an **active path**.)
  - Movement not along an active path triggers no protocol action.
  - IF a **Src** moves during an active session, it can reinitiate route discovery to establish a new route to the **Dest**.
  - WHEN a **Dest/intermediate node** moves, a Route Error (RERR) is sent to the affected Src nodes.
    - » This is initiated by the node **upstream** of the break (closer to the Src.)
    - » It lists each of the Dest that are now unreachable because of the loss of the link.



AODV: Route Discovery Process

- **Route Maintenance (Contd.)**
  - When the neighbor(s) receive the RERR, they mark their routes to Dest "Invalid" by setting the distance to INFINITY and propagating RERR to their precursors. When a Src receives the RERR, it **can reinitiate** route discovery.

- **Local Neighborhood**
  - "Hello" messages (Unsolicited RREP) with own IP and Current Seq#.
  - Not rebroadcast beyond 1-hop.

AODV: Route Discovery Process

- **Actions after Reboot** (reboot: loss of Seq# info, neighbors)
  - Wait for delete_period, during which it does not respond to any packet.
  - If it receives a DATA packet, it broadcasts a RERR and resets the waiting timer (lifetime) to expire after the **current time PLUS delete_period.**
  - By the time the reboot node comes out of waiting and becomes active, **none of its neighbors will still be using it as an active next hop.**
  - Its own Seq# is updated once it receives a RREQ from any other node.