Ad-hoc On-Demand Distance Vector Routing

Conference Paper · January 1999

DOI: 10.1109/MCSA.1999.749281 · Source: DBLP

2 authors:

Charles Edward Perkins
Institute of Electrical and Electronics Engineers
210 PUBLICATIONS **52,375** CITATIONS

SEE PROFILE

Elizabeth M. Belding
University of California, Santa Barbara
173 PUBLICATIONS **29,711** CITATIONS

SEE PROFILE

Ad-Hoc On-Demand Distance Vector (AODV) Routing

Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Das,
Proceedings of IEEE WMCSA'99,New Orleans, LA,
February 1999

Presented by 민 병 호
(**Byungho Min**)
17 April, 2006

Contents

- Ad-Hoc Network Routing Constraints
- Ad-Hoc On-Demand Distance Vector (AODV)
- AODV Algorithm
  - Path Discovery
  - Route Table Management
  - Path Maintenance
  - Local Connectivity Management
- Simulation and Results
- Current Status and Future Work
- Conclusion

Ad-Hoc Network Routing Constraints

- No Infrastructure Support
  - No Specialized Routers
  - (Physically) No Fixed Routers
- Frequent Topology Change
  - Mobility
- Problems Due to Wireless Media
  - Bandwidth
  - Range of Communication
  - Collisions Due to Broadcasting
  - Asymmetric One-Way Links

Ad-Hoc On-Demand Distance Vector (AODV)

- A Source Initiated Routing Protocol
- Premise
  - Symmetric Links
- Features
  - Pure On-Demand Acquisition
    - Broadcast only when necessary
  - Notion of 'Active' Paths
    - No centralized topological knowledge
    - Neighborhood detection
  - Topological Maintenance
    - Uses 'Hello' to identify neighbors

AODV Algorithm

- Path Discovery
  - Reverse Path Setup
  - Forward Path Setup
- Route Table Management
- Path Maintenance
- Local Connectivity Management

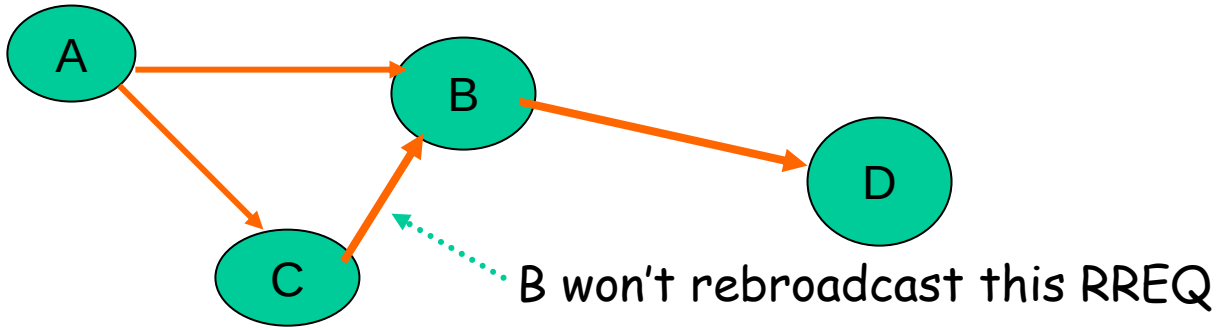
Path Discovery

- When?
  - No Information on Some Node to Which Communication Is Requested
- Each Node Has Two Counters

  node sequence number broadcast_id
- Path Discovery Process
  - Source Node Broadcasts a Route Request (RREQ) Packet to Its Neighbors
  - Neighbors 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

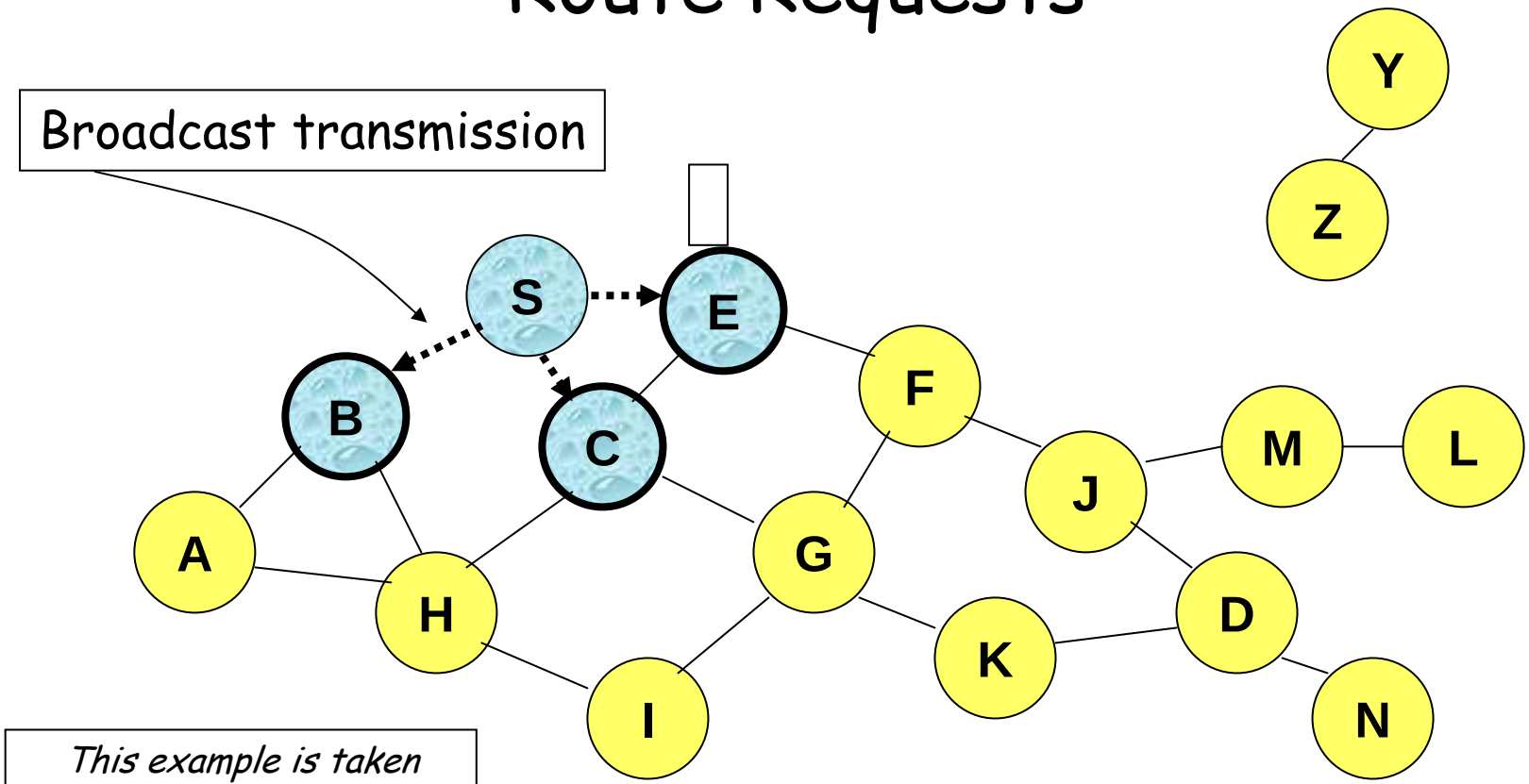
More on RREQ

- *broadcast_id* Is Incremented for New RREQ
- RREQ from Same Node with Same *broadcast_id* Will Not Be Broadcasted More than Once



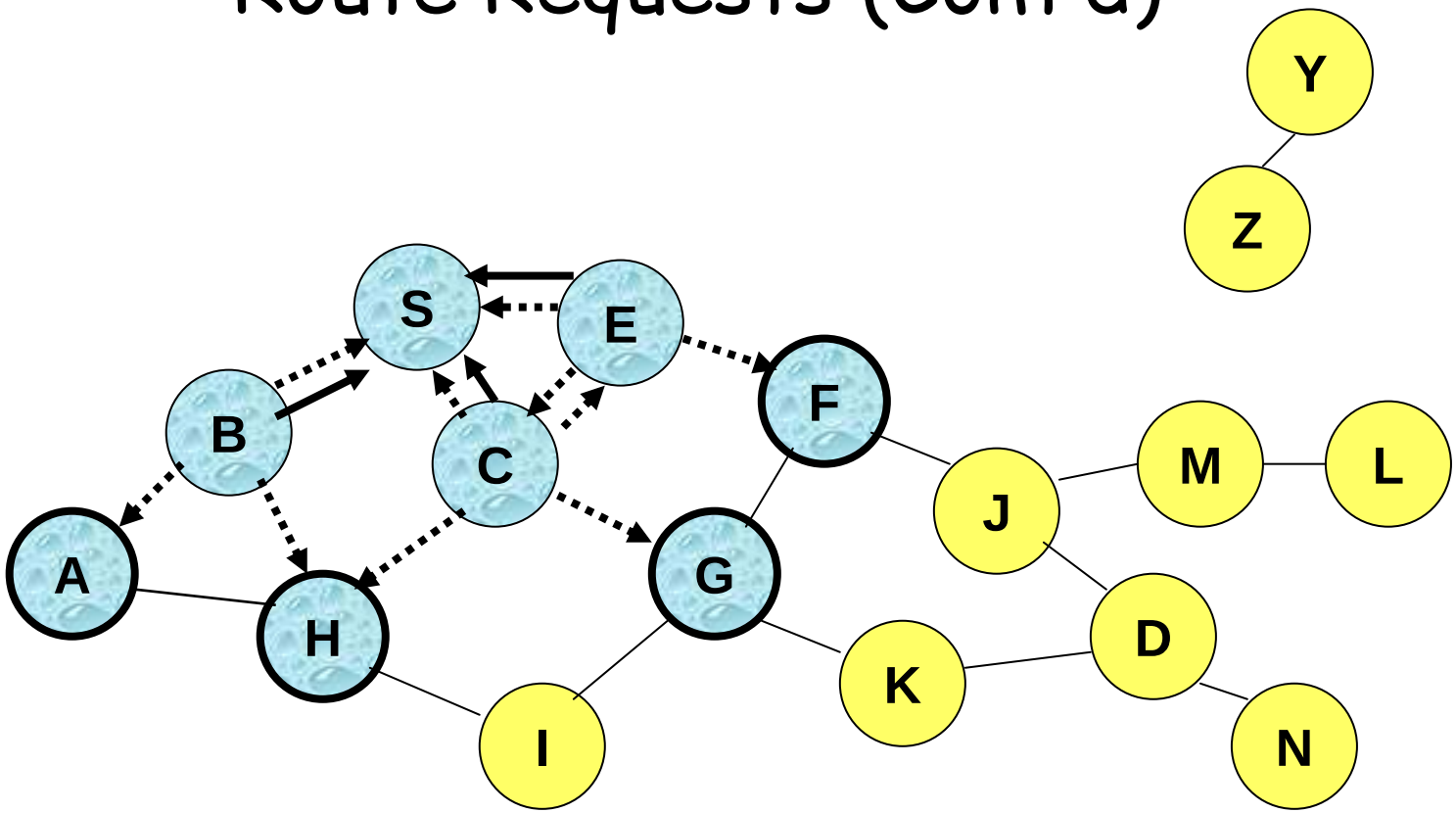
Node A wants to contact node D,

Route Requests

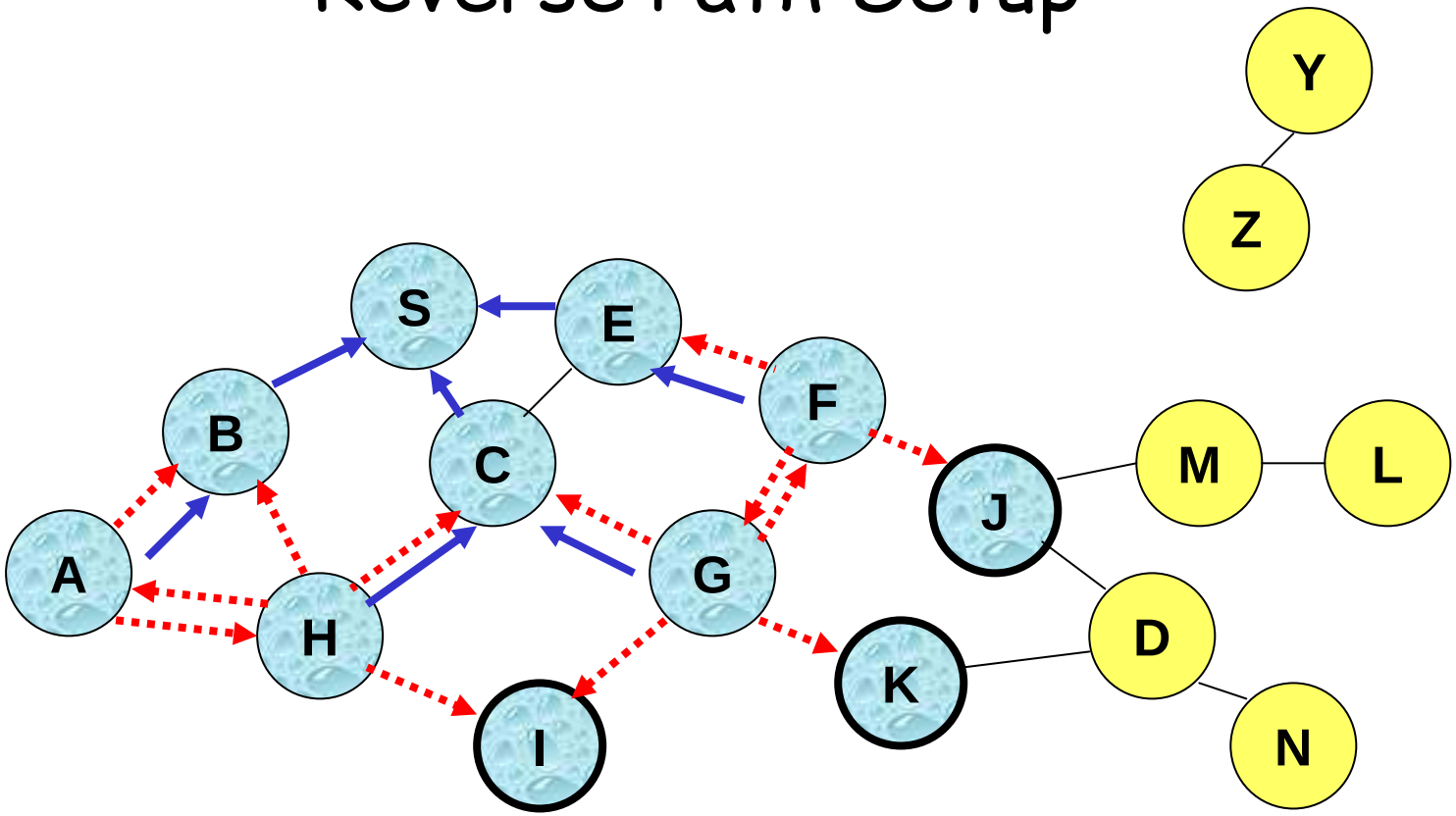


Represents transmission of RREQ

Route Requests (Cont'd)

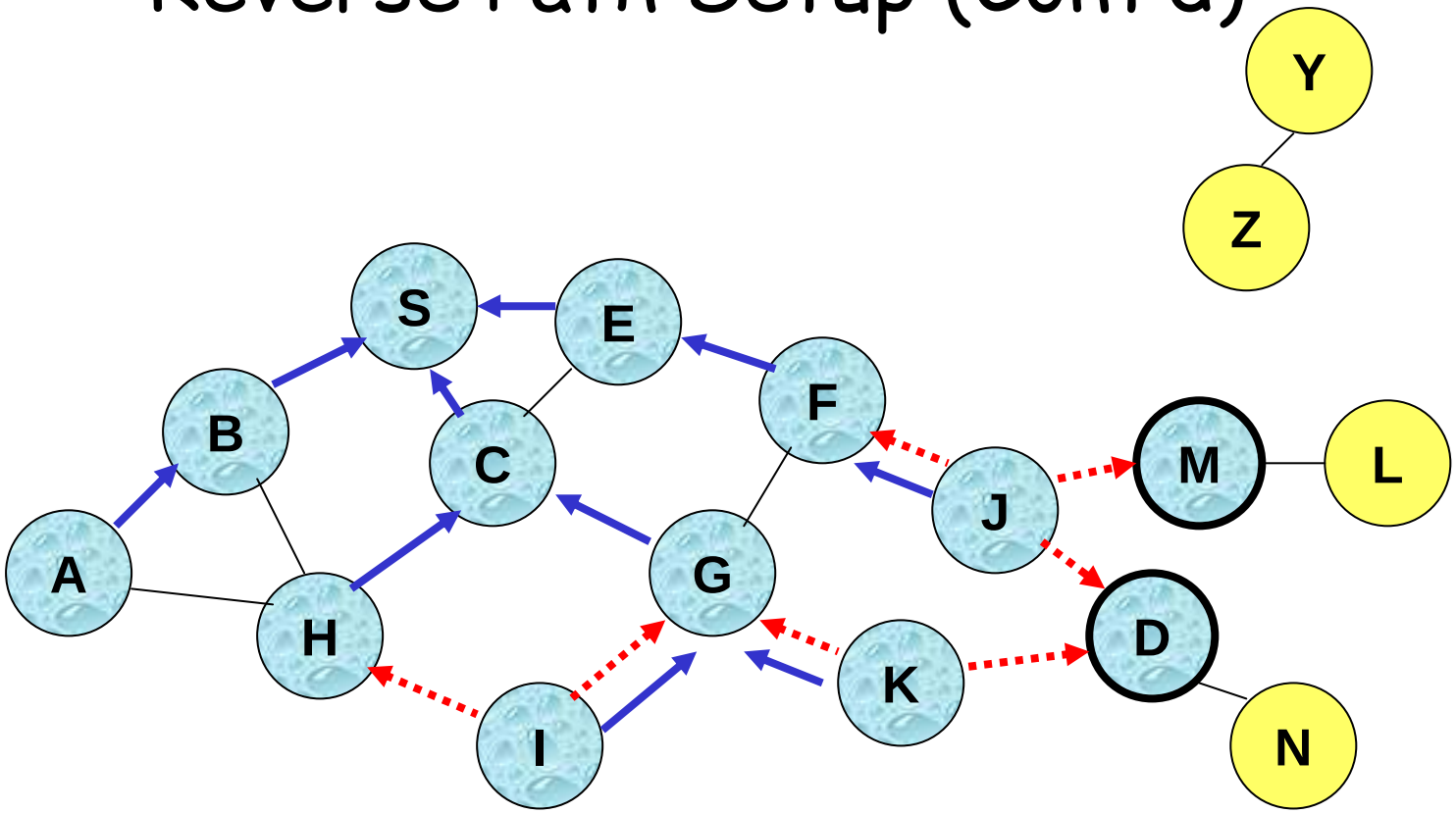


Reverse Path Setup

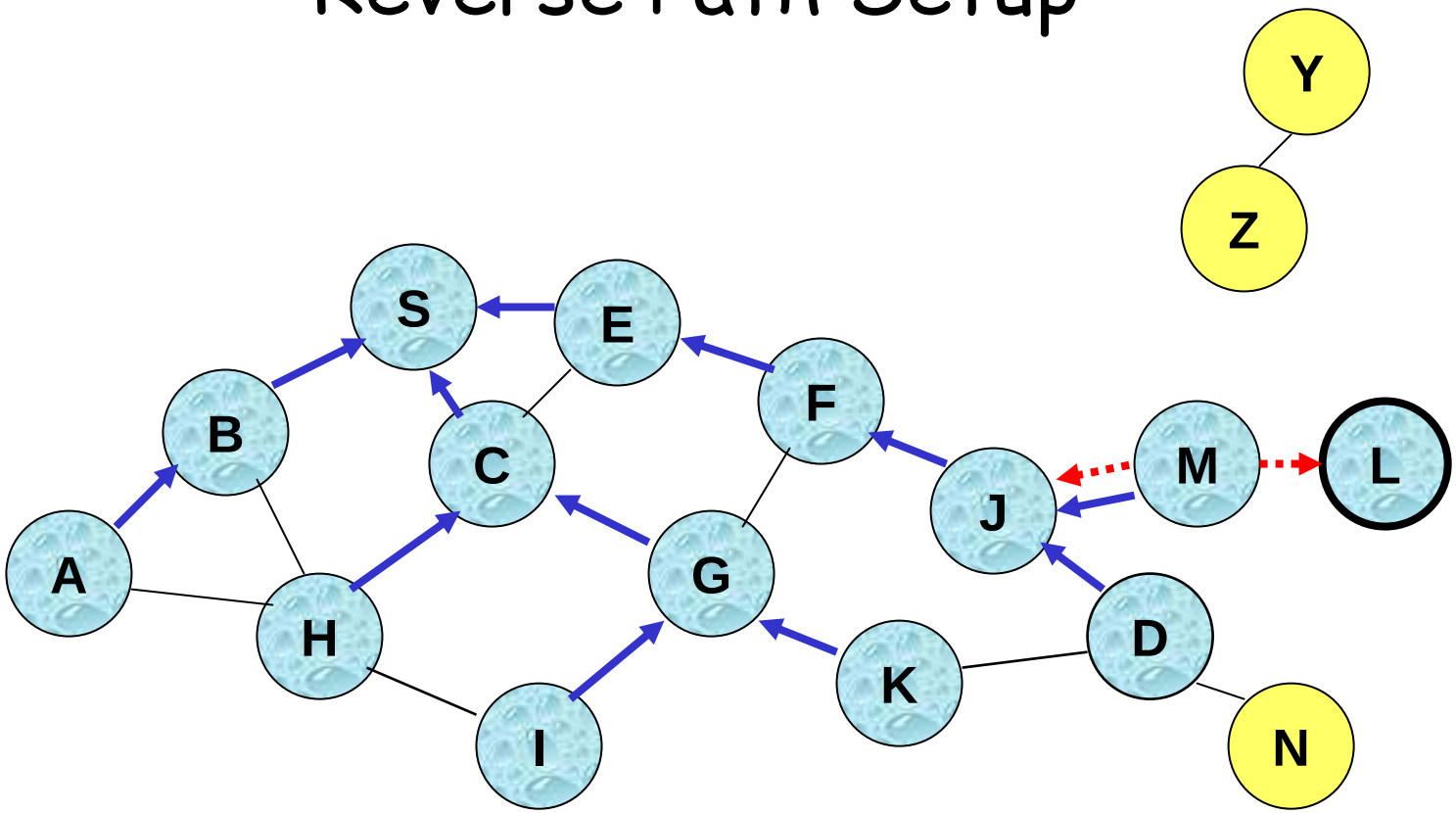


Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once

Reverse Path Setup (Cont'd)

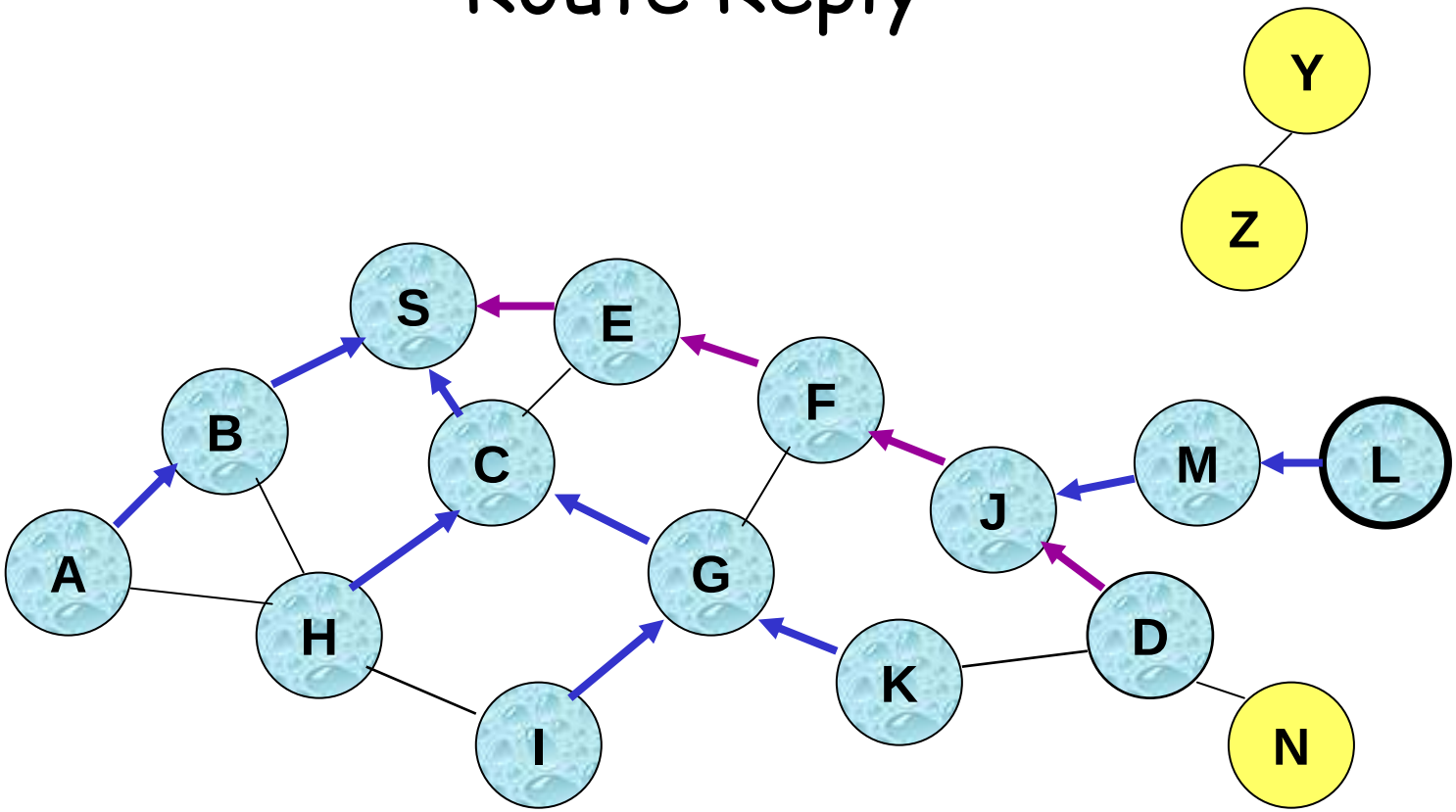


Reverse Path Setup



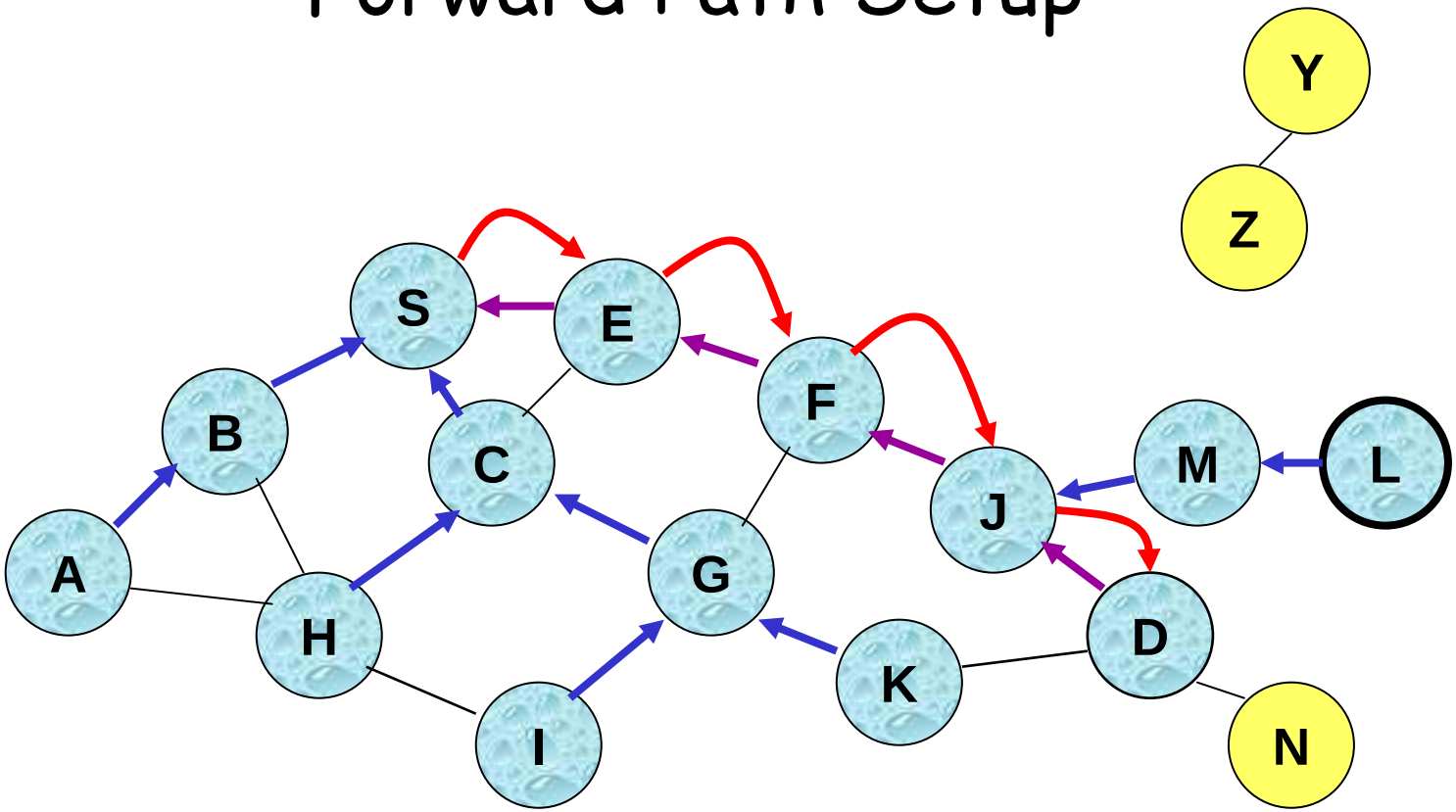
Node D does not forward RREQ, because node D is the intended target of the RREQ

Route Reply



Represents links on path taken by RREP

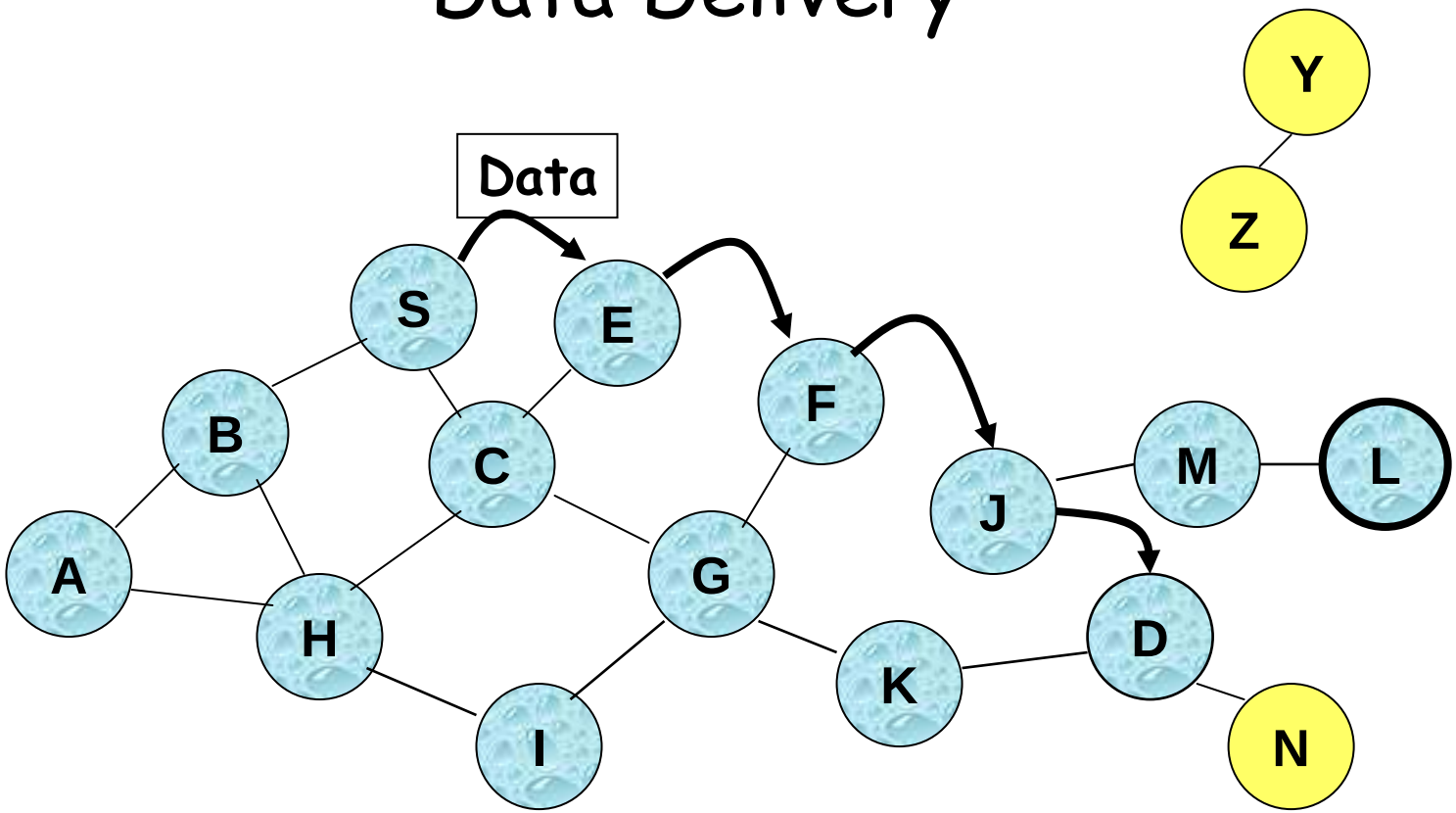
Forward Path Setup



Forward links are setup when RREP travels along the reverse path

Represents a link on the forward path

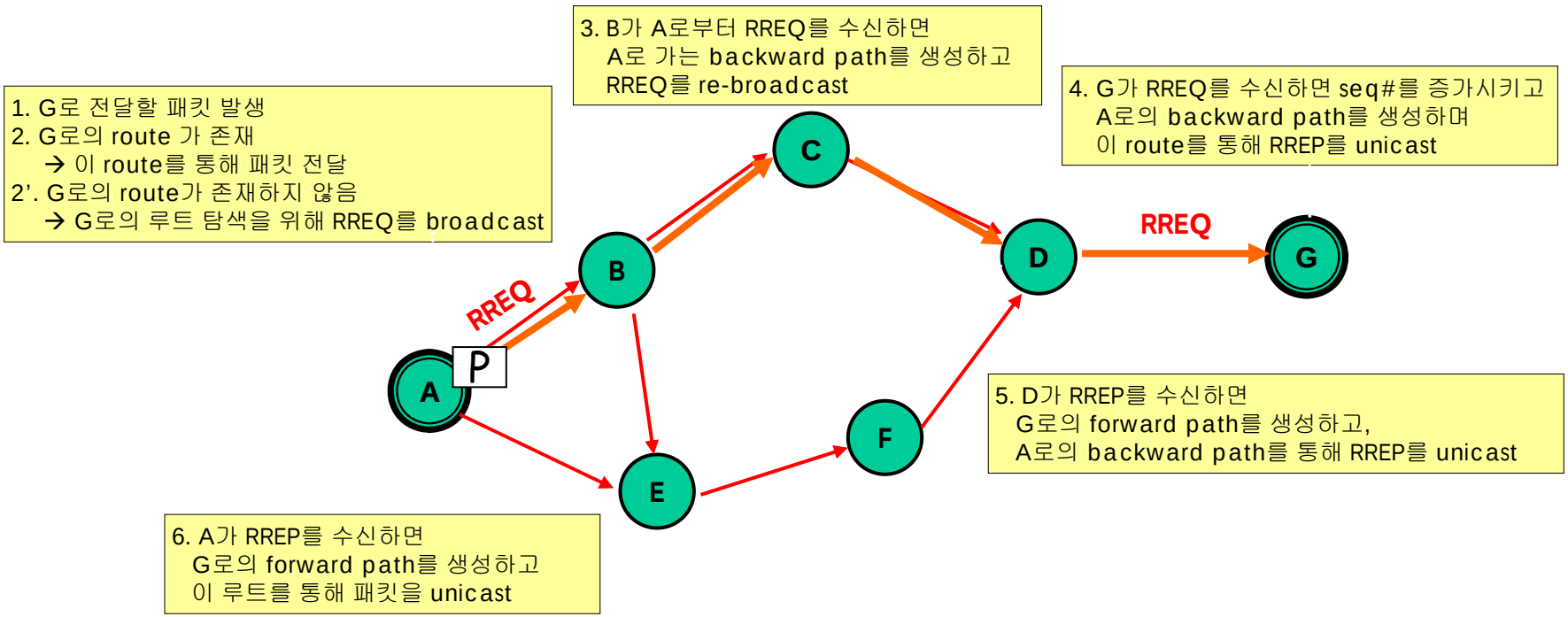
Data Delivery



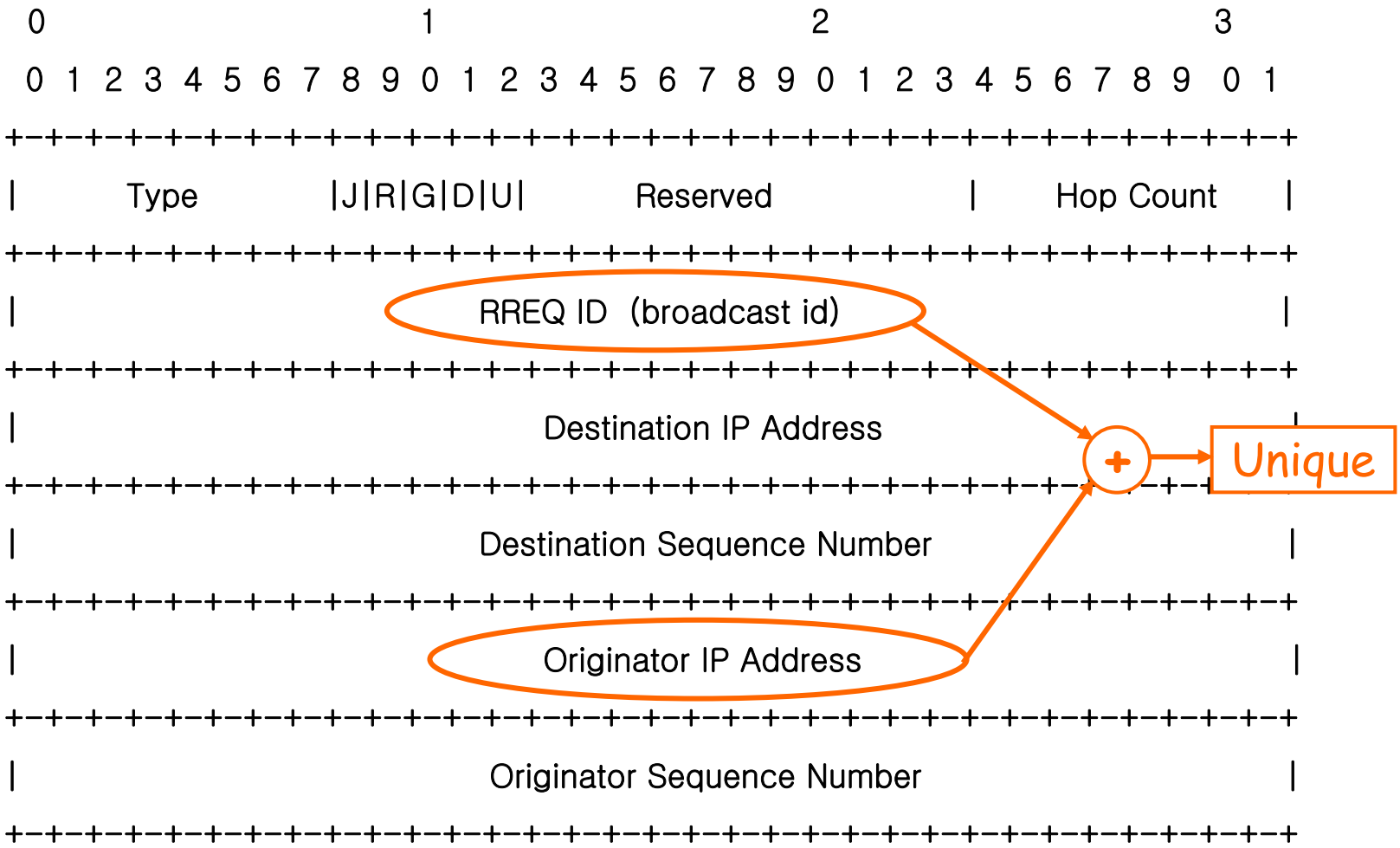
Routing table entries used to forward data packet.
Route is *not* included in packet header

Path Discovery (Cont'd)

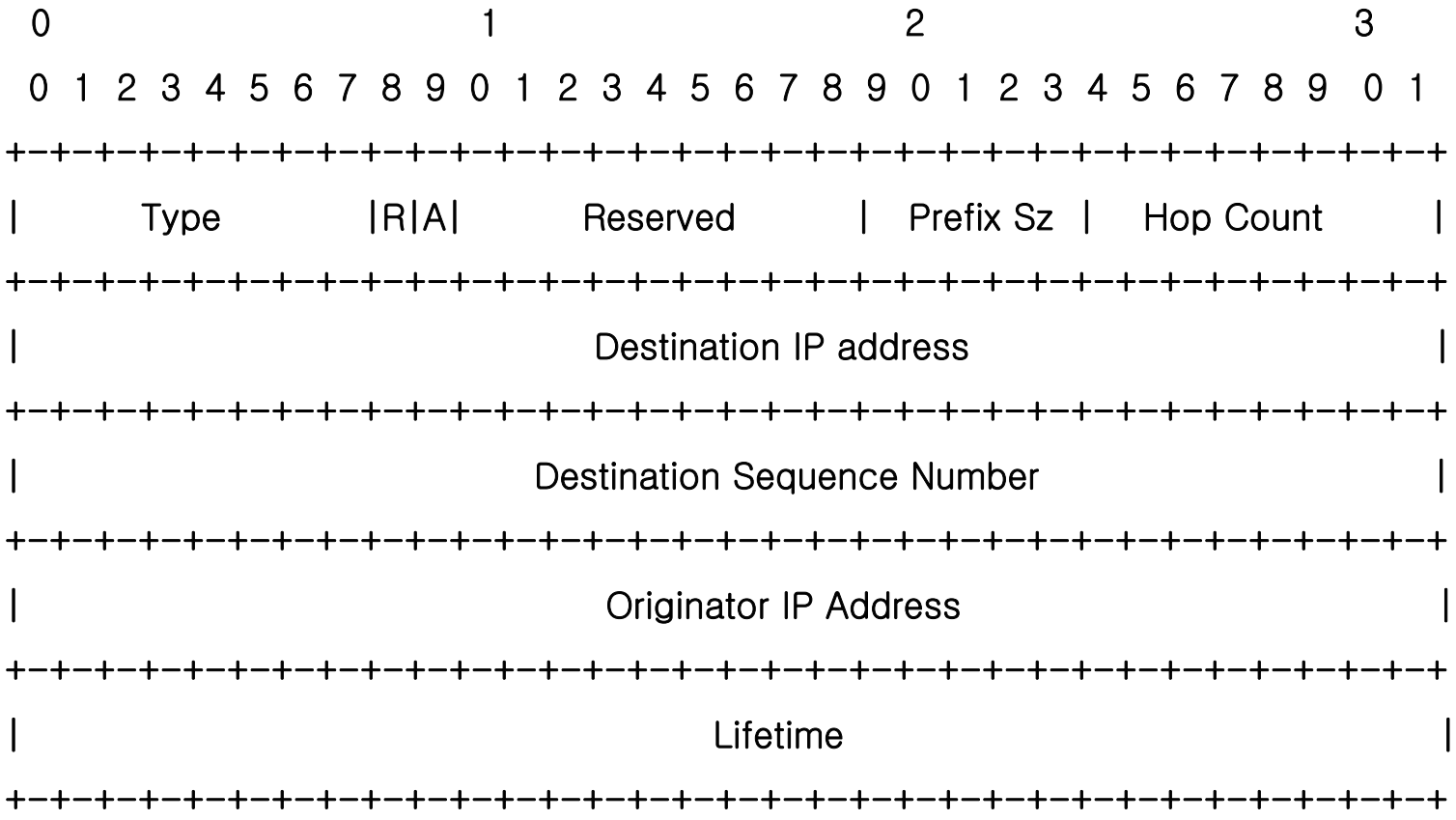
- RREQ Generates Backward Path to Source
- RREP Generates Forward Path to Destination



Route Request (RREQ) Message Format



Route Reply (RREP) Message Format



Route Table Management

- *Route Request Expiration Timer*
  - Purges Reverse Paths That Do Not Lie on Active Route
- *active_route_timeout*
  - Is Used to Determine If Neighboring Node Is Active i.e. Sends at Least One Packet in This Time
- *Route Cache Timer*
  - Purges Inactive Routes

Route Table Management (Cont'd)

- Each Route Table Entry Contains
  - Destination
  - Next Hope
  - Number of Hops
  - Sequence Number for the Destination
  - Active Neighbors for This Route
  - Expiration Time for the Route Table Entry

Route Table Management (Cont'd)

- When a Route Entry Is Used to Transmit Data from Source to Destination, *Timeout* for Each Entry Is Reset to Current Time + *active_route_timeout*

- When a New Route Is Available, Route Table Will Be Updated Only If New Route Has
  - Larger *dest_sequence_#*

  Or
  - Same *dest_sequence_#* but with Smaller *hop_cnt* to the Destination

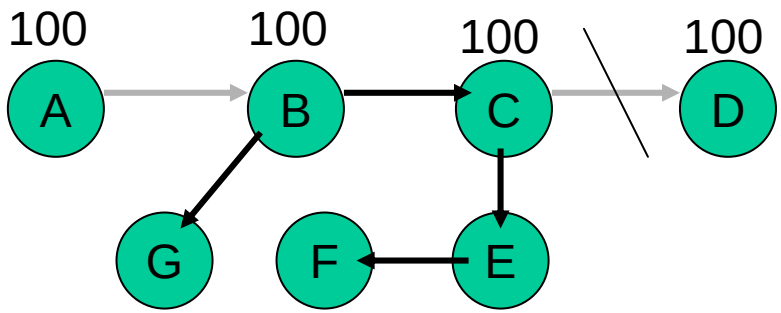
Path Management

If Source Node Moves during an Active Session
→ It Can Redo Path Discovery

- If Destination or Intermediate Nodes Move
→ A Special RREP Is Sent to Affected Source Nodes
  - On link breakage, affected node propagates an unsolicited RREP <*dest_sequence_#*+1, ∞> to all active neighbors
→ Source May Restart Route Discovery Process

Path Management Example

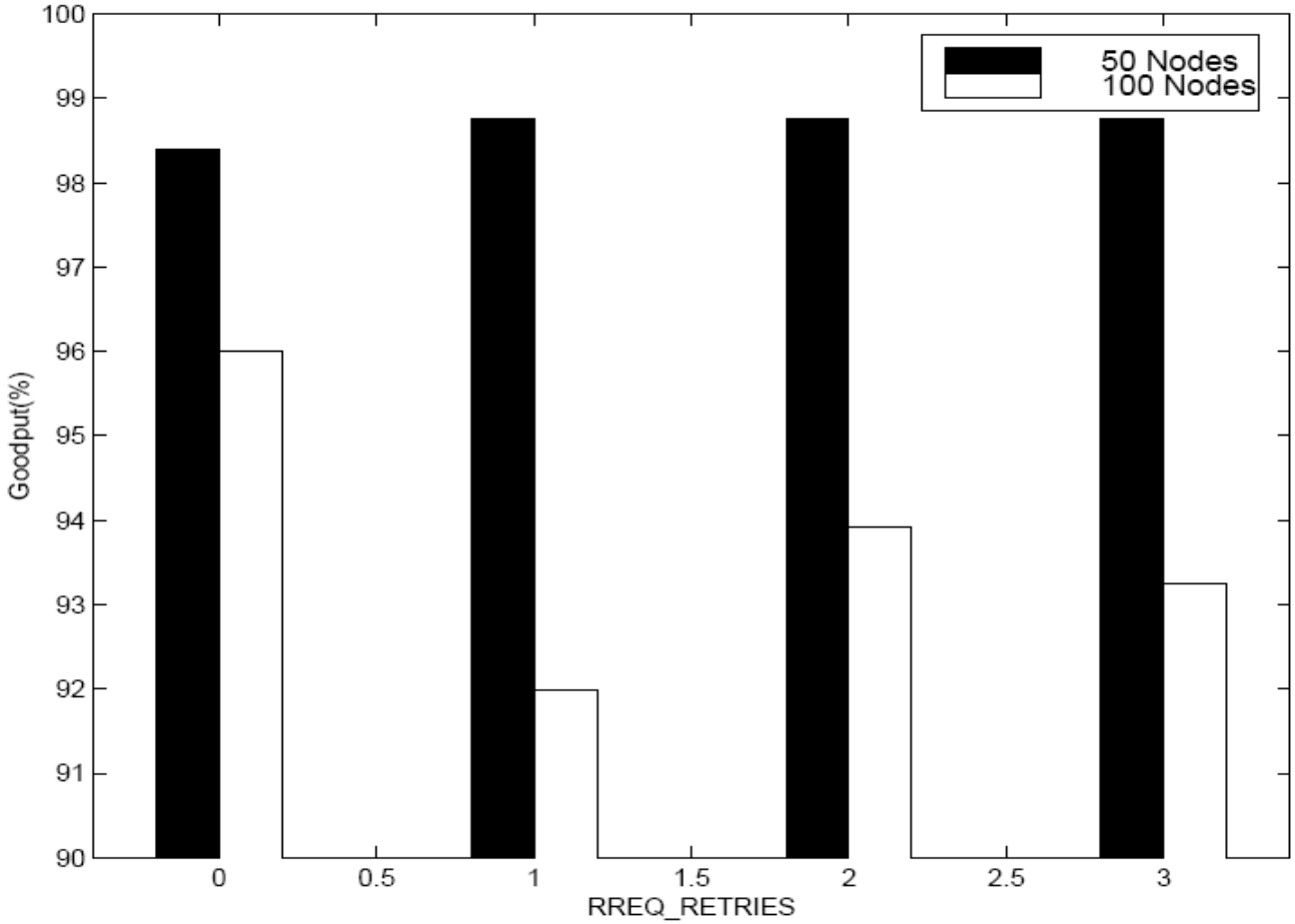
- If Link between C and D Is Broken, a Special RREP Will Be Sent to B and E by C
  - C Will Increase Its *dest_sequence_#* by 1 and Set *hop_count* to ∞)
- Then B and E Will Send This RREP to Its Active Neighbors F and G
- At the End, All Active Source Nodes Are Notified
- When Source Finds Out a Link to Destination Is Broken, and It Likes to Rebuild the Path to Destination, Source Will Send a RREQ with Last *dest_squence_#* + 1



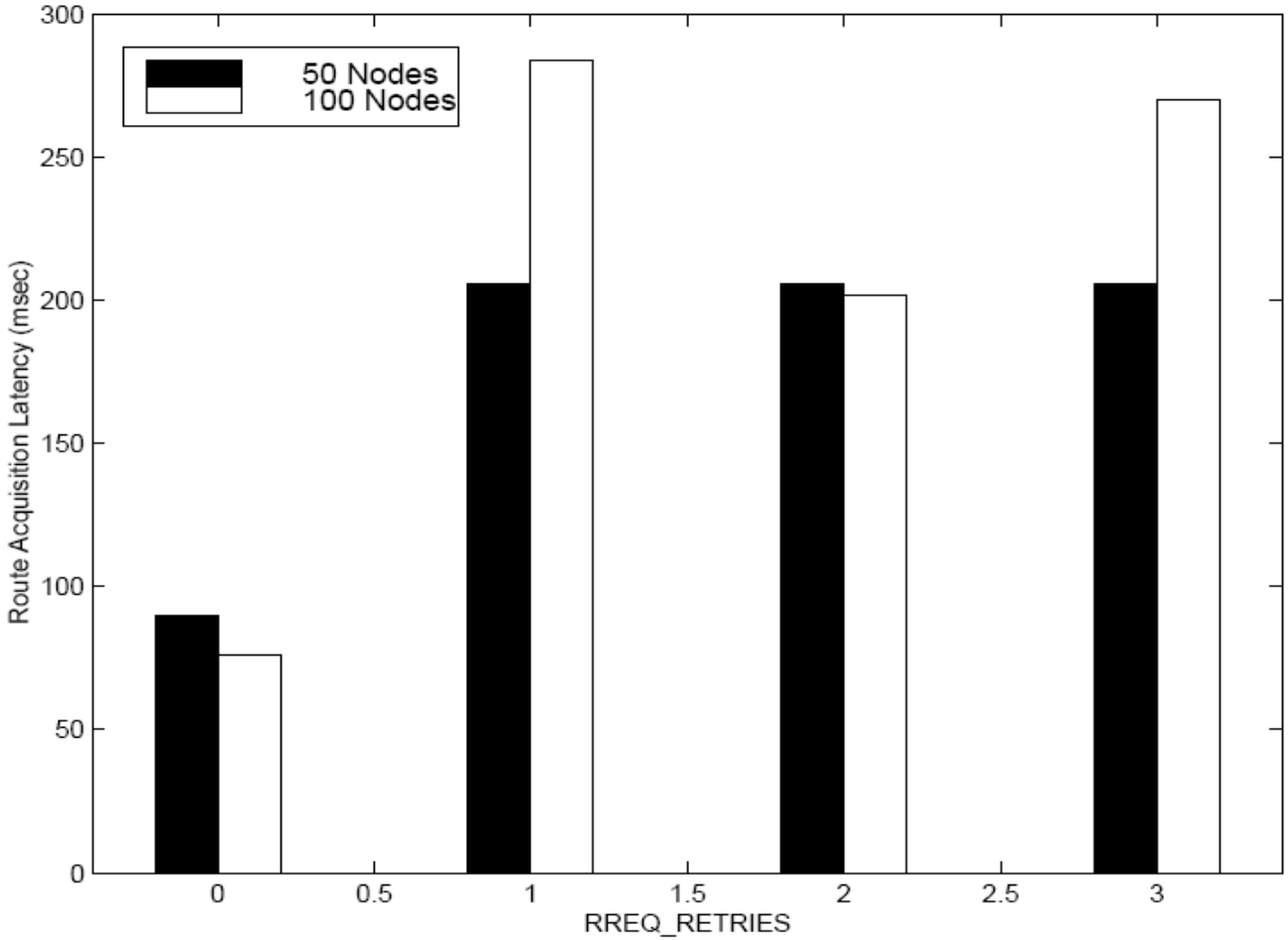
Local Connectivity Management

- Two Ways for a Node to Find Its Neighbors
  - Receiving Broadcast from Its Neighbors
  - Send Its Neighbors Hello Messages Containing Its Identity and Sequence Number
    - Sequence number is not changed for hello message
    - Nodes can not rebroadcast hello messages (TTL=1)
- Local Connectivity Is Changed If
  - Receiving a Broadcast or a Hello from a New Node
  - Failing to Receive *allowed_hello_loss* Consecutive Hello Message from a Node Previously in the Neighborhood
- Neighbors Only Communicate When Heard Each Other's Hello Message
  - It Ensures the Link Is Bidirectional

Simulations and Results



Simulations and Results



Current Status and Future Work

- Current Status
  - Multicast
    - RREQ, RREP, and Multicast validation message
- Future Works
  - Intermediate Node Route Rebuilding
    - Try 'repair' before link failure notification
  - Elimination of Hello Messages
  - Locality of Association and QoS
    - Improve bandwidth utilization by varying route propagation without indiscriminate relays

Conclusion

- Storing Only the Needed Routes
- Minimized Need for Broadcast
- Reduced Memory Requirements and Needless Duplications
- Quick Response to Link Breakage in Active Routes
- Loop-Free Routes Maintained by Use of Destination Sequence Numbers
- Scalable to Large Populations of Nodes