Course Overview

- Introduction and History
- Data in Wireless Cellular Systems
- Data in Wireless Local Area Networks
- Internet Protocols
- Routing and Ad-Hoc Networks
  - some slides in this section are from the Tutorial on Mobile Ad Hoc Networks: Routing, MAC and Transport Issues, prepared by Nitin Vaidya, see http://www.cs.tamu.edu/faculty/vaidya/presentations.html
- TCP over Wireless Link
- Services and Service Discovery
- System Support for Mobile Applications

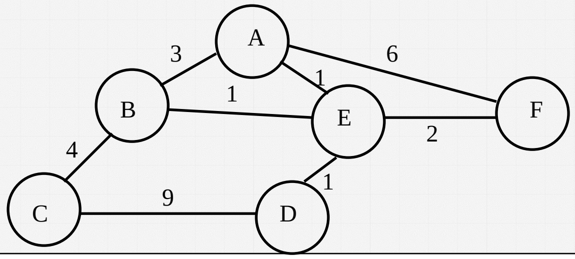
Routing

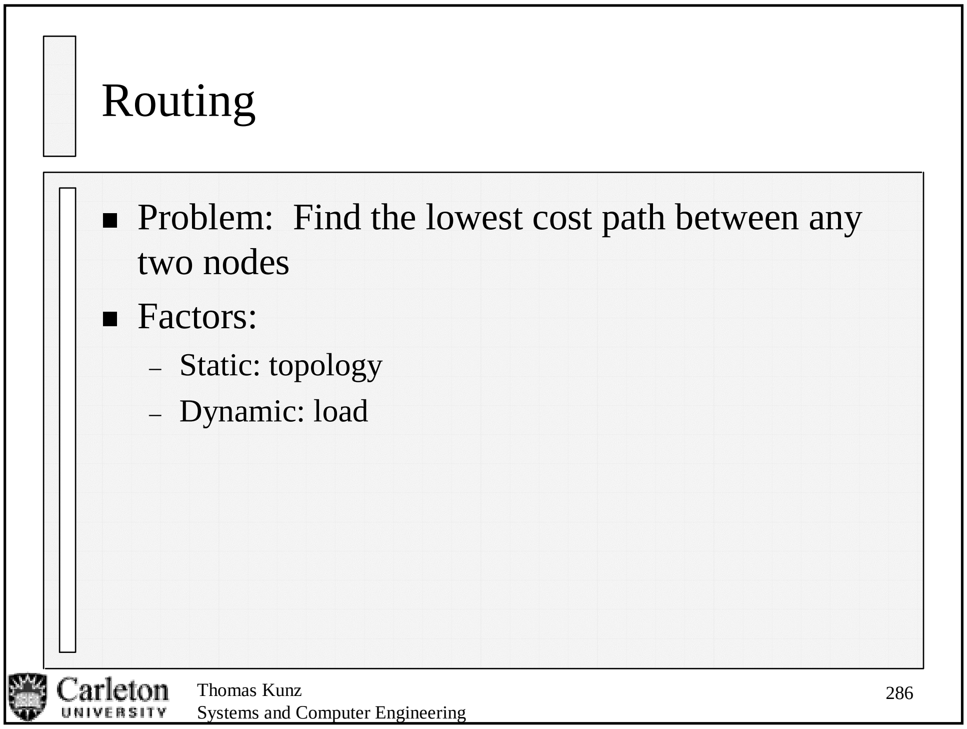
Forwarding versus Routing:

forwarding: to select an output port based on destination address and routing table

routing: process by which routing table is built

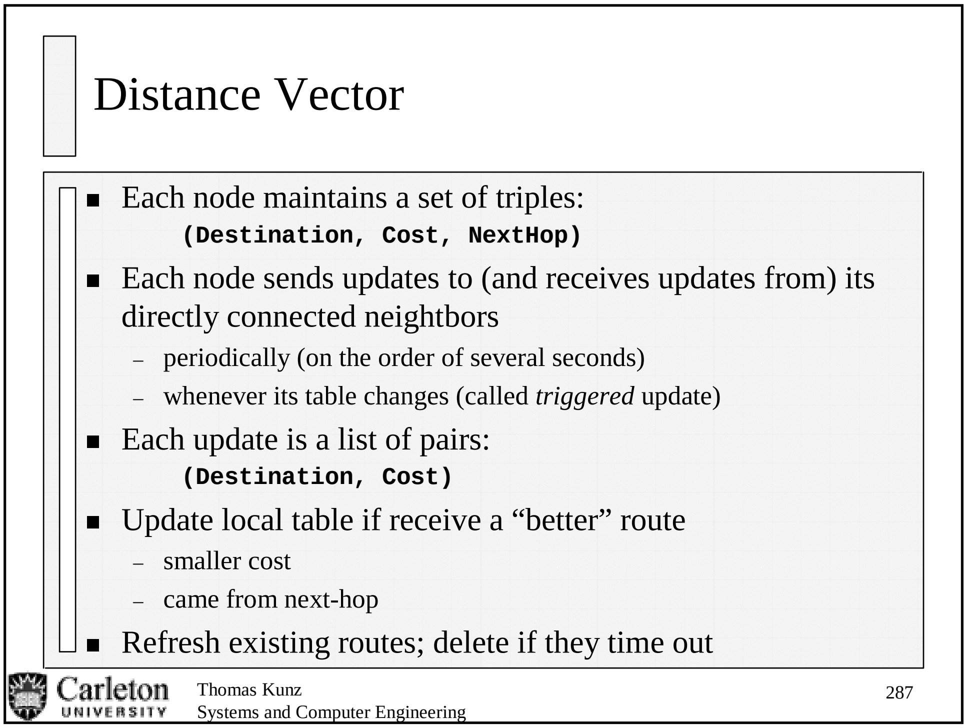
Network as a Graph:





Routing

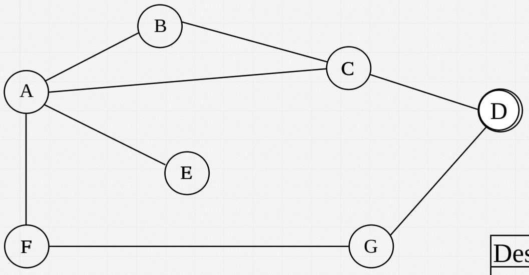
- Problem: Find the lowest cost path between any two nodes
- Factors:
  - Static: topology
  - Dynamic: load



Distance Vector

- Each node maintains a set of triples:
  `(Destination, Cost, NextHop)`
- Each node sends updates to (and receives updates from) its directly connected neightbors
  - periodically (on the order of several seconds)
  - whenever its table changes (called *triggered* update)
- Each update is a list of pairs:
  `(Destination, Cost)`
- Update local table if receive a “better” route
  - smaller cost
  - came from next-hop
- Refresh existing routes; delete if they time out

Example

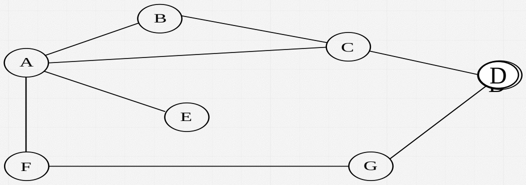


Routing table at node B

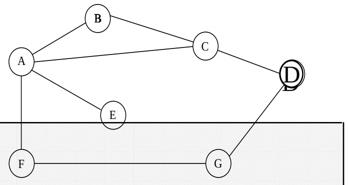
Destination	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

Routing Example

- Example 1
  - F detects that link to G has failed
  - F sets distance to G to infinity and sends update to A
  - A sets distance to G to infinity since it uses F to reach G
  - A receives periodic update from C with 2-hop path to G
  - A sets distance to G to 3 and sends update to F
  - F decides it can reach G in 4 hops via A



Routing Loops



- Example 2
  - Link from A to E fails
  - A advertises distance of infinity to E
  - B and C advertise a distance of 2 to E
  - B decides it can reach E in 3 hops; advertises this to A
  - A decides it can reach E in 4 hops; advertises this to C
  - C decides that it can reach E in 5 hops......
- Heuristics to break routing loops
  - set infinity to 16
  - split horizon
  - split horizon with poison reverse

Link State

Strategy: Send to all nodes (not just neighbors) information about directly connected links (not entire routing table).

- Link State Packet (LSP)
  - id of the node that created the LSP
  - cost of link to each directly connected neighbor
  - sequence number (SEQNO)
  - time-to-live (TTL) for this packet
- Reliable Flooding
  - store most recent LSP from each node
  - forward LSP to all nodes but one that sent it
  - generate new LSP periodically; increment SEQNO
  - start SEQNO at 0 when reboot
  - decrement TTL of each stored LSP; discard when TTL=0

Route Calculation

- Dijkstra's shortest path algorithm
- *N* denotes set of nodes in the graph
- *l(i,j)* denotes non-negative cost (weight) for edge *(i,j)*
- *s /in N* denotes this node
- *M* denotes the set of nodes incorporated so far
- *C(n)* denotes cost of the path from *s* to node *n*

```
M = {s}
for each n  in N - {s}
    C(n) = l(s,n)
while (N ° M)
    M = M union {w} such that C(w)
    is the minimum for all w in (N-M)
    for each n in (N-M)
        C(n) = MIN (C(n), C(w)+l(w,n))
```

Routing Example

Step	Confirmed	Tentative
1.	(D,0,-)	
2.	(D,0,-)	(B,11,B) (C,2,C)
3.	(D,0,-) (C,2,C)	(B,11,B)
4.	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)

Step	Confirmed	Tentative
5.	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)
6.	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)
7.	(D,0,-) (C,2,C) (B,5,C) (A,10,C)	

Route Propagation

Idea: Impose a second hierarchy on the network that limits what routers talk to each other. (The first hierarchy is the address hierarchy that governs how packets are forwarded.)

- Autonomous System (AS)
  - corresponds to an administrative domain
  - examples: University, company, backbone network
  - assign each AS a 16-bit number
- Two-level route propagation hierarchy
  - interior gateway protocol (each AS selects its own)
  - exterior gateway protocol (Internet-wide standard)

Popular Interior Gateway Protocols

- RIP: Route Information Protocol
  - developed for XNS
  - distributed with Unix
  - distance-vector algorithm
  - based on hop-count
- OSPF: Open Shortest Path First
  - recent Internet standard
  - uses link-state algorithm
  - supports load balancing
  - supports authentication

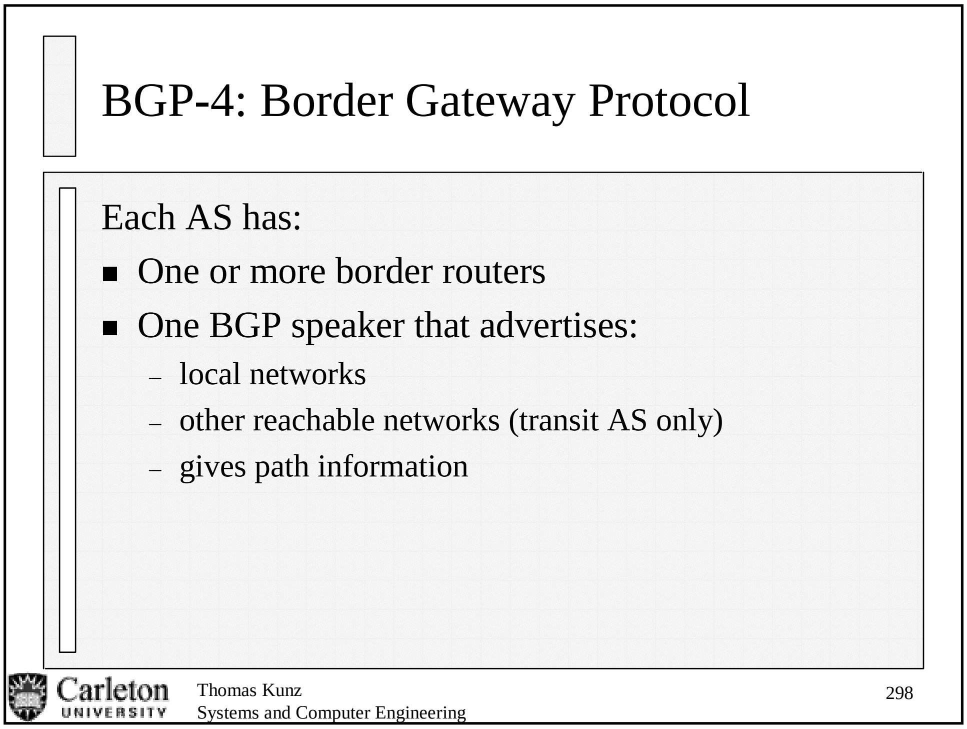
EGP: Exterior Gateway Protocol

- Overview
  - designed for tree-structured Internet
  - concerned with *reachability*, not optimal routes
- Protocol messages
  - neighbor acquisition: one router requests that another be its peer; peers exchange reachability information
  - neighbor reachability: one router periodically tests to see if the other router is still reachable; exchange HELLO/ACK messages; uses a k-out-of-n rule
  - routing updates: peers periodically exchange their routing tables (distance-vector)

BGP-4: Border Gateway Protocol

Assumes the Internet is an arbitrarily interconnected set of AS's. Define *local traffic* as traffic that originates at or terminates on nodes within an AS, and *transit traffic* as traffic that passes through an AS, we can classify AS's into three types:

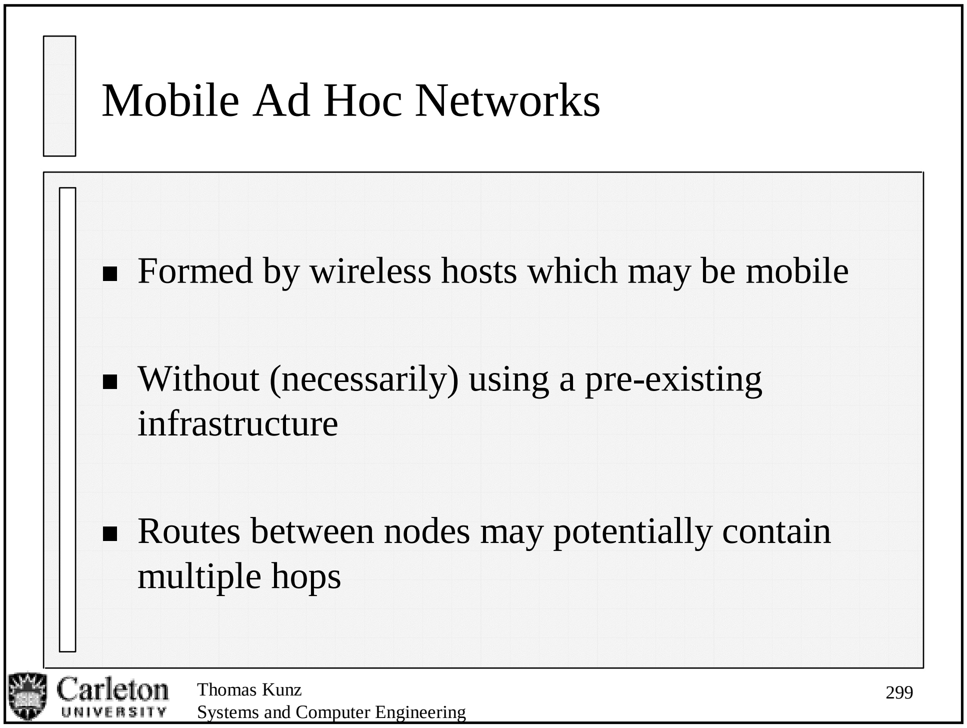
- Stub AS: an AS that has only a single connection to one other AS; such an AS will only carry local traffic.
- Multihomed AS: an AS that has connections to more than one other AS, but refuses to carry transit traffic.
- Transit AS: an AS that has connections to more than one other AS, and is designed to carry both transit and local traffic.



BGP-4: Border Gateway Protocol

Each AS has:

- One or more border routers
- One BGP speaker that advertises:
  - local networks
  - other reachable networks (transit AS only)
  - gives path information

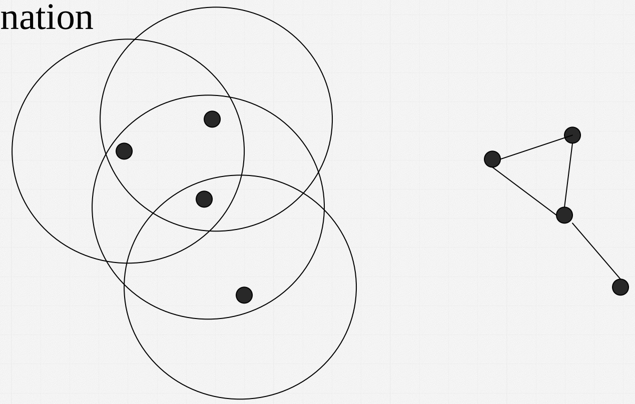


Mobile Ad Hoc Networks

- Formed by wireless hosts which may be mobile
- Without (necessarily) using a pre-existing infrastructure
- Routes between nodes may potentially contain multiple hops

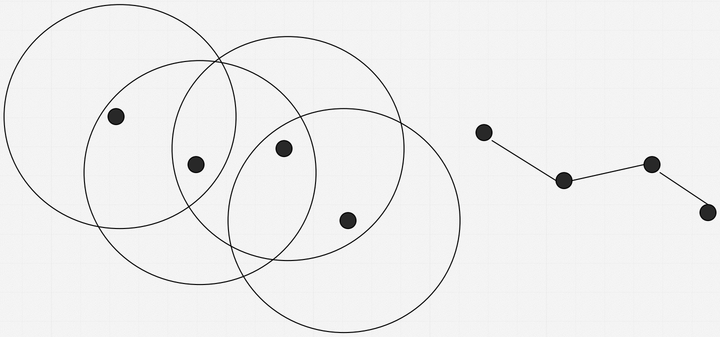
Mobile Ad Hoc Networks

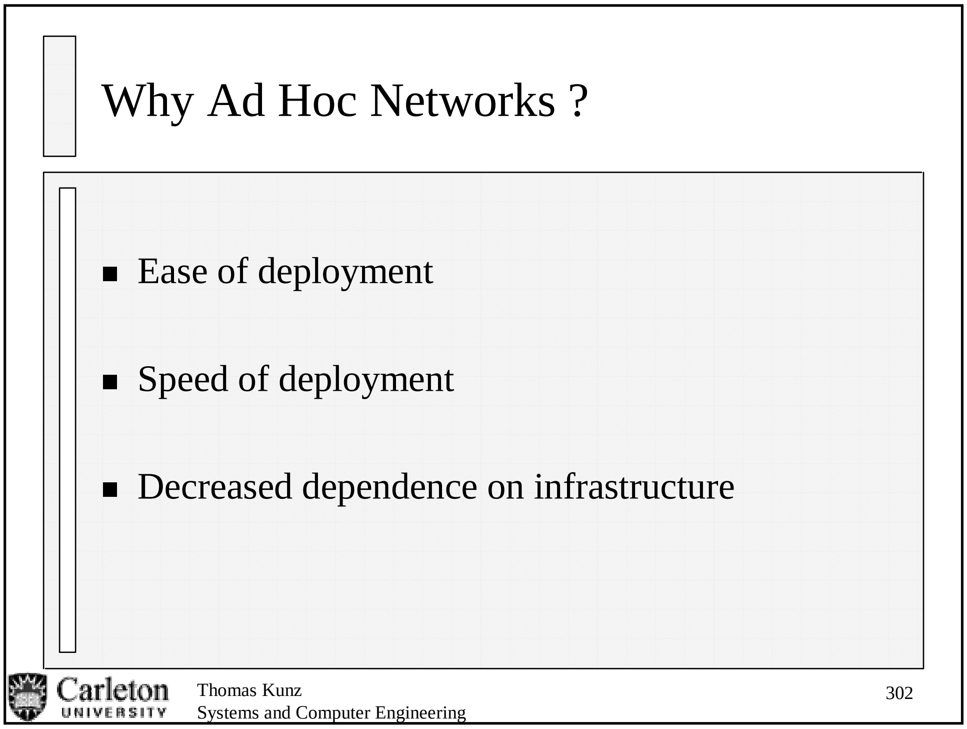
- May need to traverse multiple links to reach a destination



Mobile Ad Hoc Networks (MANET)

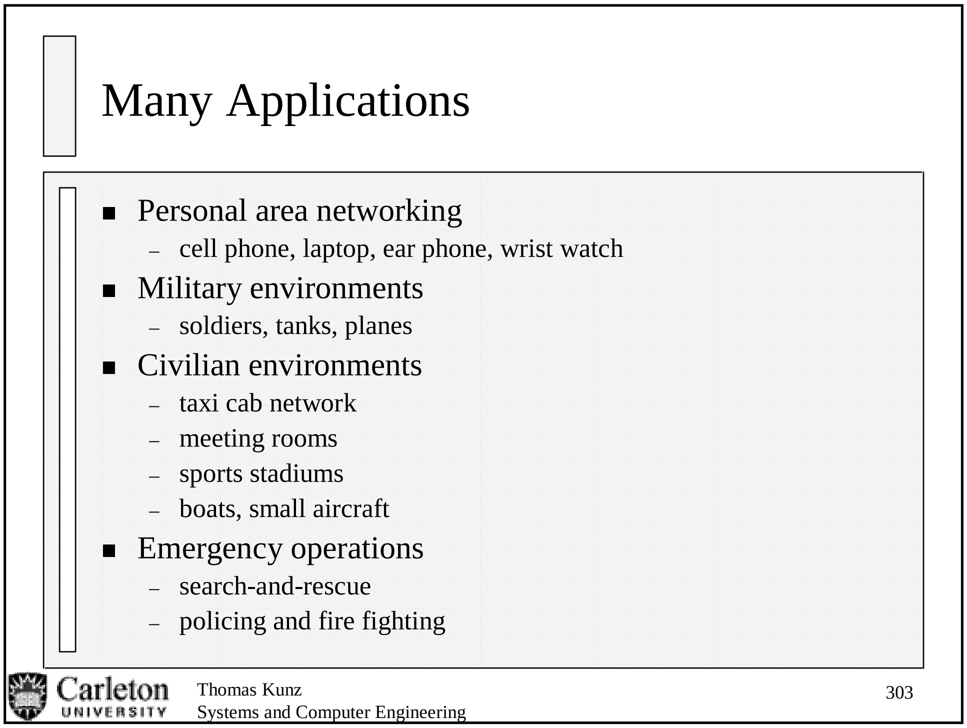
- Mobility causes route changes





Why Ad Hoc Networks ?

- Ease of deployment
- Speed of deployment
- Decreased dependence on infrastructure



Many Applications

- Personal area networking
  - cell phone, laptop, ear phone, wrist watch
- Military environments
  - soldiers, tanks, planes
- Civilian environments
  - taxi cab network
  - meeting rooms
  - sports stadiums
  - boats, small aircraft
- Emergency operations
  - search-and-rescue
  - policing and fire fighting

Many Variations

- Fully Symmetric Environment
  - all nodes have identical capabilities and responsibilities
- Asymmetric Capabilities
  - transmission ranges and radios may differ
  - battery life at different nodes may differ
  - processing capacity may be different at different nodes
  - speed of movement
- Asymmetric Responsibilities
  - only some nodes may route packets
  - some nodes may act as leaders of nearby nodes (e.g., cluster head)

Many Variations

- Traffic characteristics may differ in different ad hoc networks
  - bit rate
  - timeliness constraints
  - reliability requirements
  - unicast / multicast / geocast
  - host-based addressing / content-based addressing / capability-based addressing
- May co-exist (and co-operate) with an infrastructure-based network

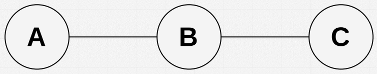
Many Variations

- Mobility patterns may be different
  - people sitting at an airport lounge
  - New York taxi cabs
  - kids playing
  - military movements
  - personal area network
- Mobility characteristics
  - speed
  - predictability
    - direction of movement
    - pattern of movement
  - uniformity (or lack thereof) of mobility characteristics among different nodes

Challenges

- Limited wireless transmission range
- Broadcast nature of the wireless medium
  - Hidden terminal problem (see next slide)
- Packet losses due to transmission errors
- Mobility-induced route changes
- Mobility-induced packet losses
- Battery constraints
- Potentially frequent network partitions
- Ease of snooping on wireless transmissions (security hazard)

Hidden Terminal Problem



Nodes A and C cannot hear each other

Transmissions by nodes A and C can collide at node B

Nodes A and C are hidden from each other

Research on Mobile Ad Hoc Networks

Variations in capabilities & responsibilities

X

Variations in traffic characteristics, mobility models, etc.

X

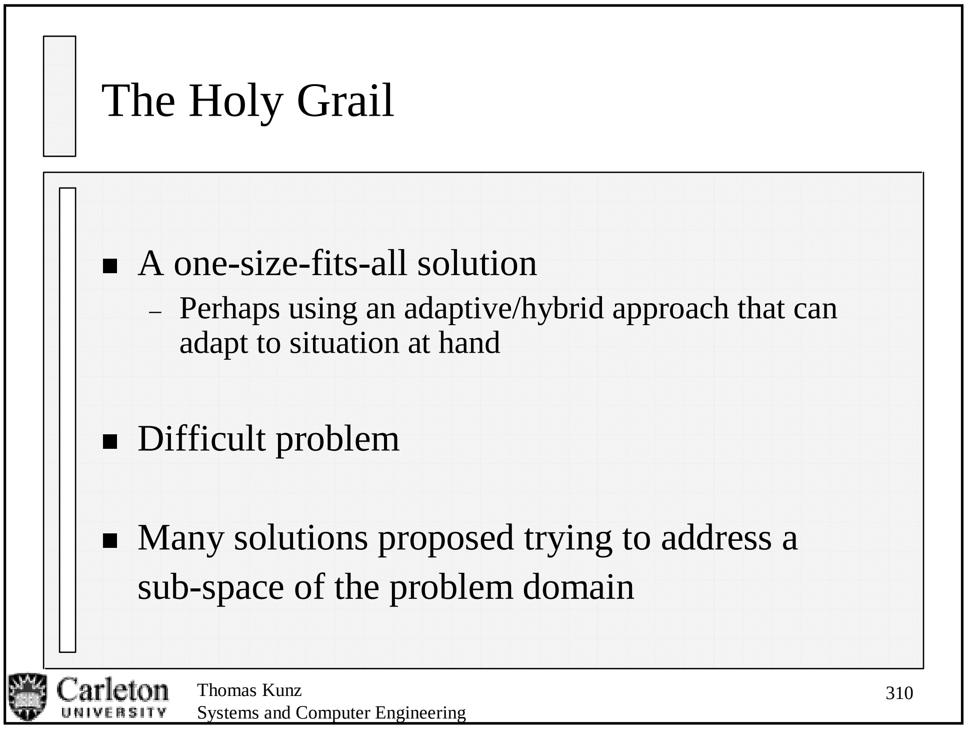
Performance criteria (e.g., optimize throughput, reduce energy consumption)

+

Increased research funding

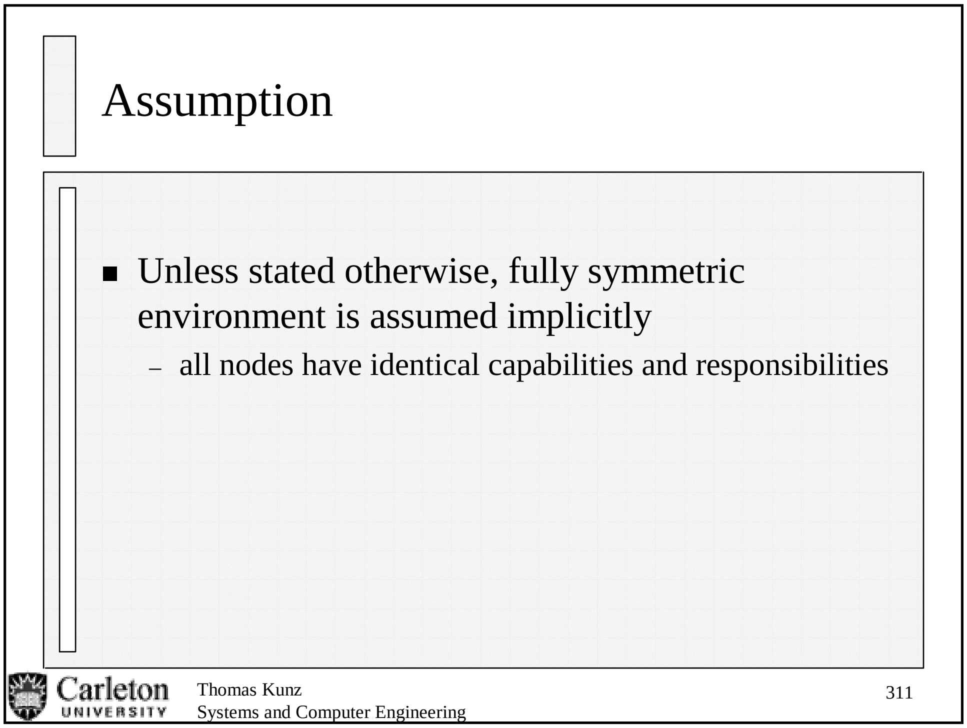
=

Significant research activity



The Holy Grail

- A one-size-fits-all solution
  - Perhaps using an adaptive/hybrid approach that can adapt to situation at hand
- Difficult problem
- Many solutions proposed trying to address a sub-space of the problem domain



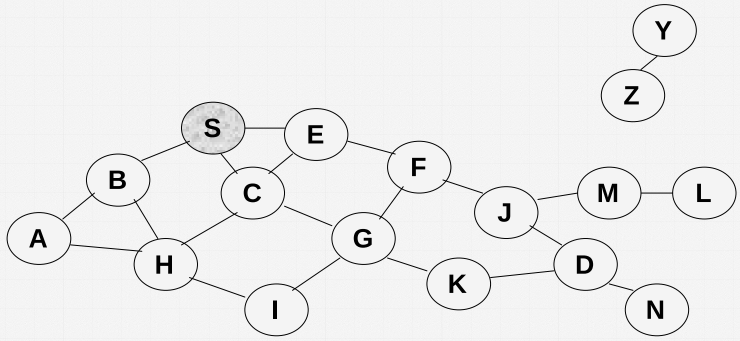
Assumption

- Unless stated otherwise, fully symmetric environment is assumed implicitly
  - all nodes have identical capabilities and responsibilities

Flooding for Data Delivery

- Sender S broadcasts data packet P to all its neighbors
- Each node receiving P forwards P to its neighbors
- Sequence numbers used to avoid the possibility of forwarding the same packet more than once
- Packet P reaches destination D provided that D is reachable from sender S
- Node D does not forward the packet

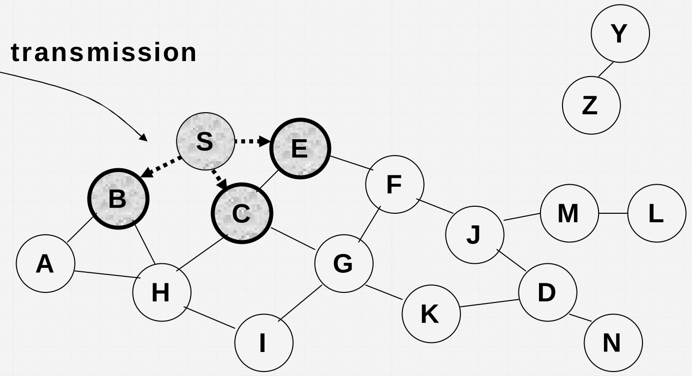
Flooding for Data Delivery



● Represents a node that has received packet P

— Represents that connected nodes are within each other’s transmission range

Flooding for Data Delivery

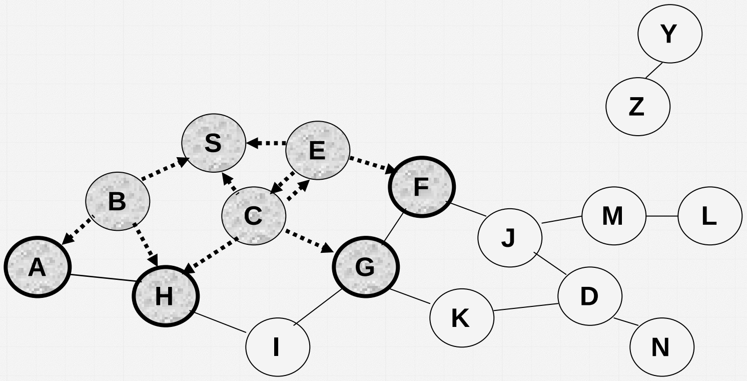


Broadcast transmission

● Represents a node that receives packet P for the first time

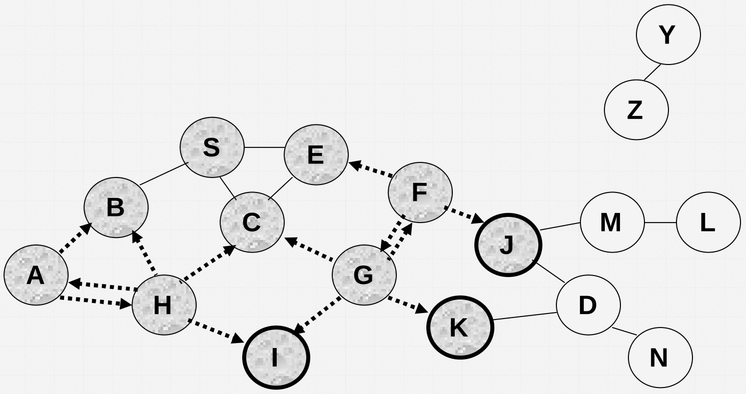
⋯▸ Represents transmission of packet P

Flooding for Data Delivery



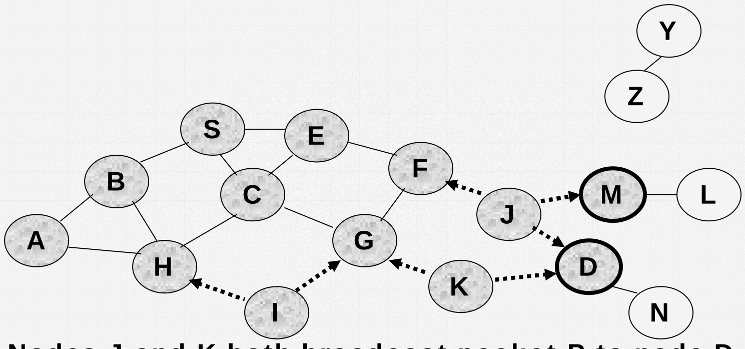
- Node H receives packet P from two neighbors: potential for collision

Flooding for Data Delivery



- Node C receives packet P from G and H, but does not forwa it again, because node C has already forwarded packet P o

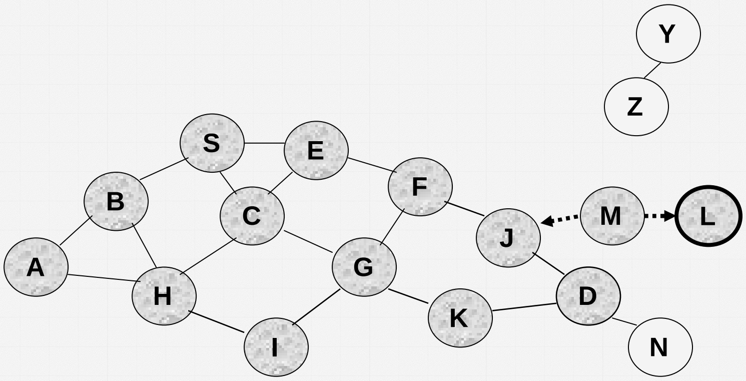
Flooding for Data Delivery



- Nodes J and K both broadcast packet P to node D
- Since nodes J and K are hidden from each other, their transmissions may collide

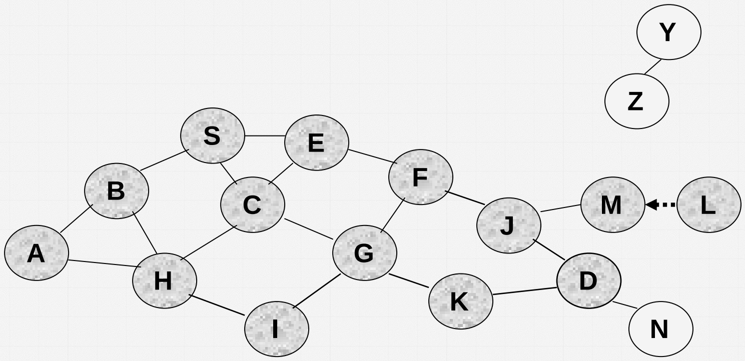
  => Packet P may not be delivered to node D at all, despite the use of flooding

Flooding for Data Delivery



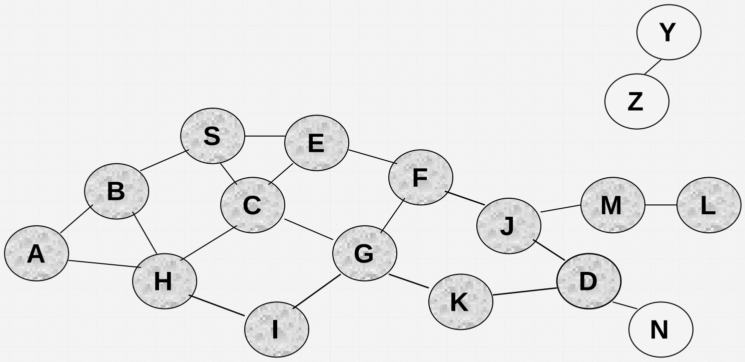
- Node D does not forward packet P, because node D is the intended destination of packet P

Flooding for Data Delivery



- Flooding completed
- Nodes unreachable from S do not receive packet P (e.g., no
- Nodes for which all paths from S go through the destination also do not receive packet P (example: node N)

Flooding for Data Delivery



- **Flooding may deliver packets to too many nodes (in the worst case, all nodes reachable from sender may receive the packet)**

Flooding for Data Delivery: Advantages

- Simplicity
- May be more efficient than other protocols when rate of information transmission is low enough that the overhead of explicit route discovery/maintenance incurred by other protocols is relatively higher
  - this scenario may occur, for instance, when nodes transmit small data packets relatively infrequently, and many topology changes occur between consecutive packet transmissions
- Potentially higher reliability of data delivery
  - Because packets may be delivered to the destination on multiple paths

Flooding for Data Delivery: Disadvantages

- Potentially, very high overhead
  - Data packets may be delivered to too many nodes who do not need to receive them
- Potentially lower reliability of data delivery
  - Flooding uses broadcasting -- hard to implement reliable broadcast delivery without significantly increasing overhead
    - Broadcasting in IEEE 802.11 MAC is unreliable
  - In our example, nodes J and K may transmit to node D simultaneously, resulting in loss of the packet
    - in this case, destination would not receive the packet at all

Flooding of Control Packets

- Many protocols perform (potentially *limited*) flooding of control packets, instead of data packets
- The control packets are used to discover routes
- Discovered routes are subsequently used to send data packet(s)
- Overhead of control packet flooding is amortized over data packets transmitted between consecutive control packet floods

Ad-Hoc Networking: MANET

- Mobile Ad-hoc Networks (manet) at IETF: http://www.ietf.org/html.charters/manet-charter.html
  - A "mobile ad hoc network" (MANET) is an autonomous system of mobile routers (and associated hosts) connected by wireless links
  - routers are free to move randomly and organize themselves arbitrarily; thus, the network's wireless topology may change rapidly and unpredictably
- The primary focus of the working group is to develop and evolve MANET routing specification(s) and introduce them to the Internet Standards track. The goal is to support networks scaling up to hundreds of routers.

Ad-Hoc Networking: DSDV

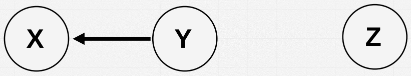
- DSDV: Destination-Sequenced Distance Vector protocol
- basic idea: destination provides freshness indication, not intermediate nodes
  - each mobile host is a router
  - periodic broadcasts with sequence numbers
  - significant new information triggers broadcast updates
  - link breakages (metric ∞) are significant
  - two criteria for route selection:
    - sequence number guarantee maximal freshness (top priority)
    - metric comparison for lowest cost
  - protocol supports incremental and full updates
  - some routes in forwarding table might not be advertised

Destination-Sequenced Distance-Vector (DSDV)

- Each node maintains a routing table which stores
  - next hop towards each destination
  - a cost metric for the path to each destination
  - a destination sequence number that is created by the destination itself
  - Sequence numbers used to avoid formation of loops
- Each node periodically forwards the routing table to its neighbors
  - Each node increments and appends its sequence number when sending its local routing table
  - This sequence number will be attached to route entries created for this node

Destination-Sequenced Distance-Vector (DSDV)

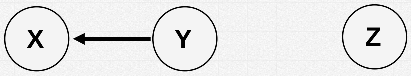
- Assume that node X receives routing information from Y about a route to node Z



- Let S(X) and S(Y) denote the destination sequence number for node Z as stored at node X, and as sent by node Y with its routing table to node X, respectively

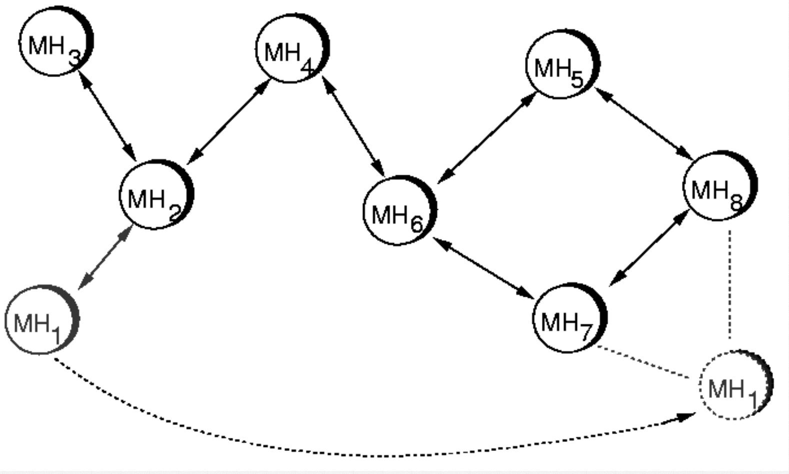
Destination-Sequenced Distance-Vector (DSDV)

- Node X takes the following steps:

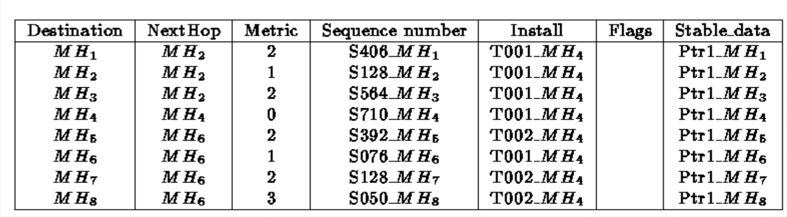


  - If S(X) > S(Y), then X ignores the routing information received from Y
  - If S(X) = S(Y), and cost of going through Y is smaller than the route known to X, then X sets Y as the next hop to Z
  - If S(X) < S(Y), then X sets Y as the next hop to Z, and S(X) is updated to equal S(Y)

Ad-Hoc Networking: DSDV Example

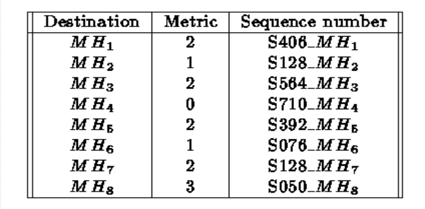


Ad-Hoc Networking: DSDV Example



Destination	NextHop	Metric	Sequence number	Install	Flags	Stable_data
MH_1	MH_2	2	S406_MH_1	T001_MH_4		Ptr1_MH_1
MH_2	MH_2	1	S128_MH_2	T001_MH_4		Ptr1_MH_2
MH_3	MH_2	2	S564_MH_3	T001_MH_4		Ptr1_MH_3
MH_4	MH_4	0	S710_MH_4	T001_MH_4		Ptr1_MH_4
MH_5	MH_6	2	S392_MH_5	T002_MH_4		Ptr1_MH_5
MH_6	MH_6	1	S076_MH_6	T001_MH_4		Ptr1_MH_6
MH_7	MH_6	2	S128_MH_7	T002_MH_4		Ptr1_MH_7
MH_8	MH_6	3	S050_MH_8	T002_MH_4		Ptr1_MH_8

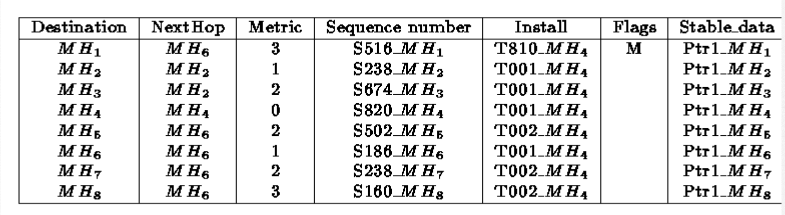
Routing Table for MH4



Destination	Metric	Sequence number
MH_1	2	S406_MH_1
MH_2	1	S128_MH_2
MH_3	2	S564_MH_3
MH_4	0	S710_MH_4
MH_5	2	S392_MH_5
MH_6	1	S076_MH_6
MH_7	2	S128_MH_7
MH_8	3	S050_MH_8

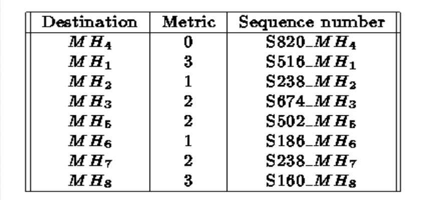
Advertised Routes by MH4

Ad-Hoc Networking: DSDV Example



Destination	NextHop	Metric	Sequence number	Install	Flags	Stable_data
MH_1	MH_6	3	S516_MH_1	T810_MH_4	M	Ptr1_MH_1
MH_2	MH_2	1	S238_MH_2	T001_MH_4		Ptr1_MH_2
MH_3	MH_2	2	S674_MH_3	T001_MH_4		Ptr1_MH_3
MH_4	MH_4	0	S820_MH_4	T001_MH_4		Ptr1_MH_4
MH_5	MH_6	2	S502_MH_5	T002_MH_4		Ptr1_MH_5
MH_6	MH_6	1	S186_MH_6	T001_MH_4		Ptr1_MH_6
MH_7	MH_6	2	S238_MH_7	T002_MH_4		Ptr1_MH_7
MH_8	MH_6	3	S160_MH_8	T002_MH_4		Ptr1_MH_8

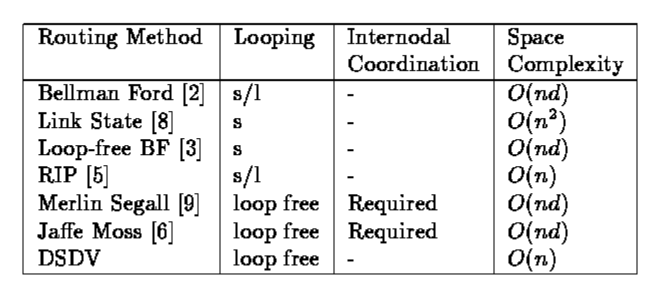
Routing Table at MH4 (updated)



Destination	Metric	Sequence number
MH_4	0	S820_MH_4
MH_1	3	S516_MH_1
MH_2	1	S238_MH_2
MH_3	2	S674_MH_3
MH_5	2	S502_MH_5
MH_6	1	S186_MH_6
MH_7	2	S238_MH_7
MH_8	3	S160_MH_8

Advertised Routes by MH4

DSDV Evaluation



Routing Method	Looping	Internodal Coordination	Space Complexity
Bellman Ford [2]	s/l	-	$O(nd)$
Link State [8]	s	-	$O(n^2)$
Loop-free BF [3]	s	-	$O(nd)$
RIP [5]	s/l	-	$O(n)$
Merlin Segall [9]	loop free	Required	$O(nd)$
Jaffe Moss [6]	loop free	Required	$O(nd)$
DSDV	loop free	-	$O(n)$

- DSDV good for small ad-hoc networks
- Brute-force approach to larger networks
  - requires periodic advertisements and global dissemination of connectivity information
  - each node has to keep complete list of routes

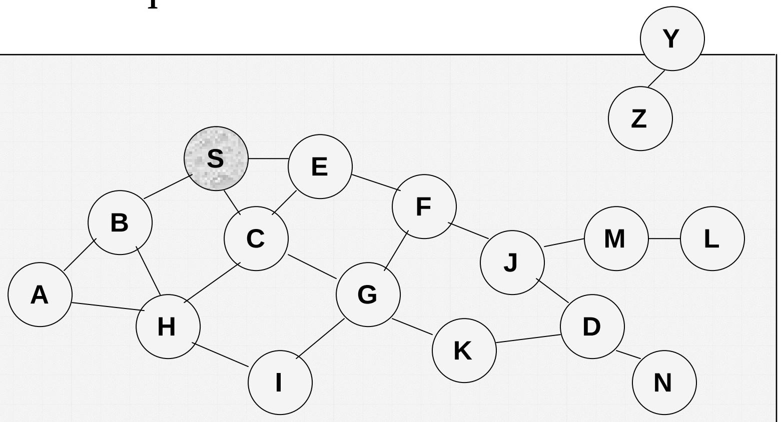
AODV: Improvement on DSDV

- AODV intends to improve on DSDV shortcomings
- basic idea: create routes only “on demand” to eliminate overhead of periodic broadcasts
- problem: if purely “on demand”, could lead to long latencies before first packet gets transmitted
- AODV assumes symmetric links between two nodes
- discover new nodes in neighborhood using local *hello* messages
- primary objectives of AODV:
  - broadcast discovery packets only when necessary
  - distinguish between local connectivity changes and general topological maintenance
  - disseminate information about local changes only to those neighboring mobiles that are likely to need information

AODV

- Route Requests (RREQ) are forwarded via flooding
- When a node re-broadcasts a Route Request, it sets up a reverse path pointing towards the source
  - AODV assumes symmetric (bi-directional) links
- When the intended destination receives a Route Request, it replies by sending a Route Reply
- Route Reply travels along the reverse path set-up when Route Request is forwarded

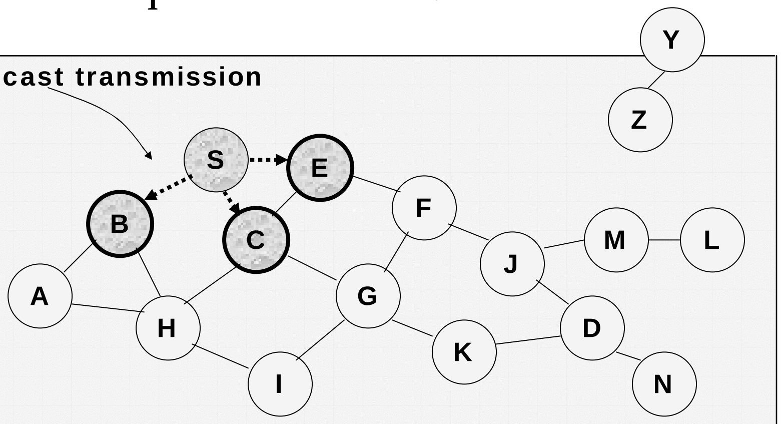
Route Requests in AODV



● **Represents a node that has received RREQ for D from**

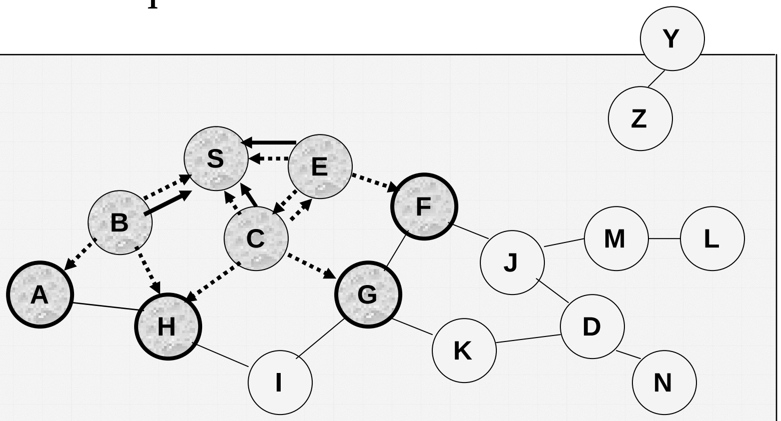
Route Requests in AODV

Broadcast transmission



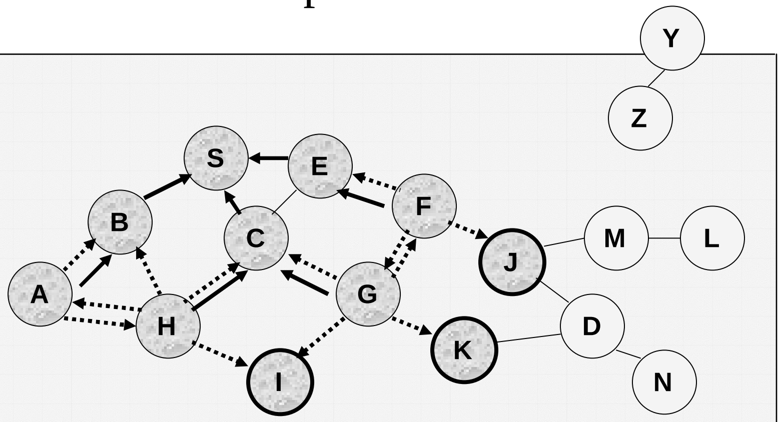
Represents transmission of RREQ

Route Requests in AODV



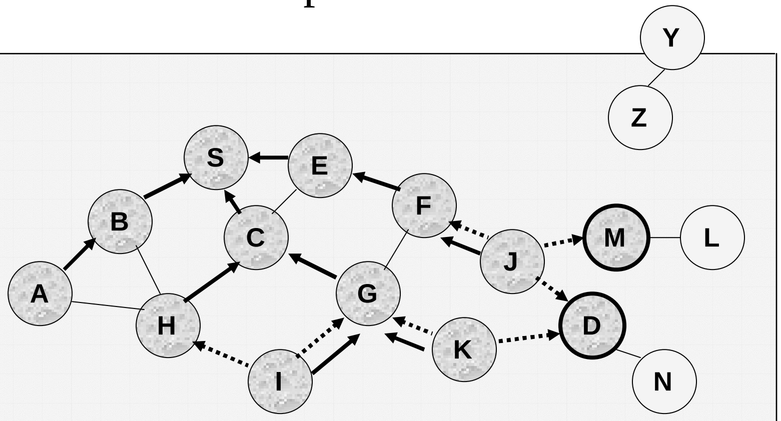
Represents links on Reverse Path

Reverse Path Setup in AODV

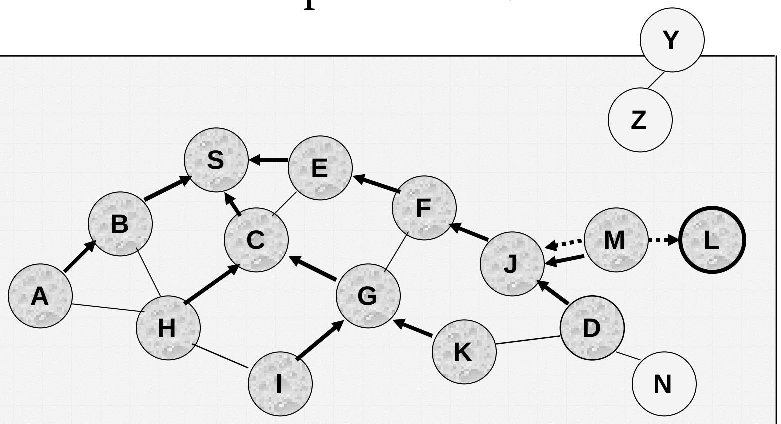


- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ onc

Reverse Path Setup in AODV

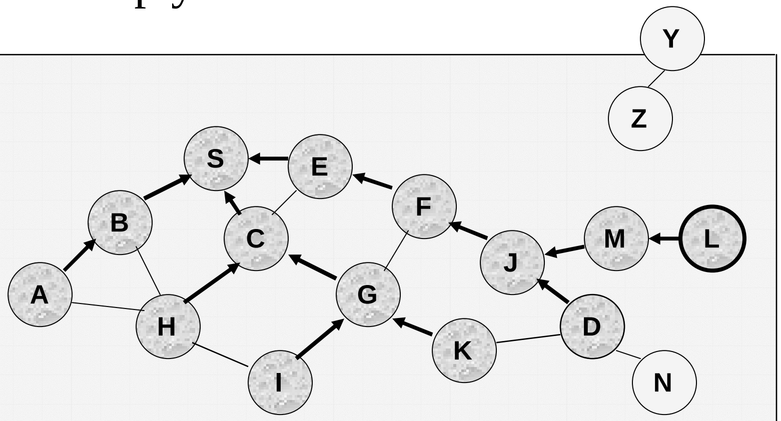


Reverse Path Setup in AODV



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

Route Reply in AODV

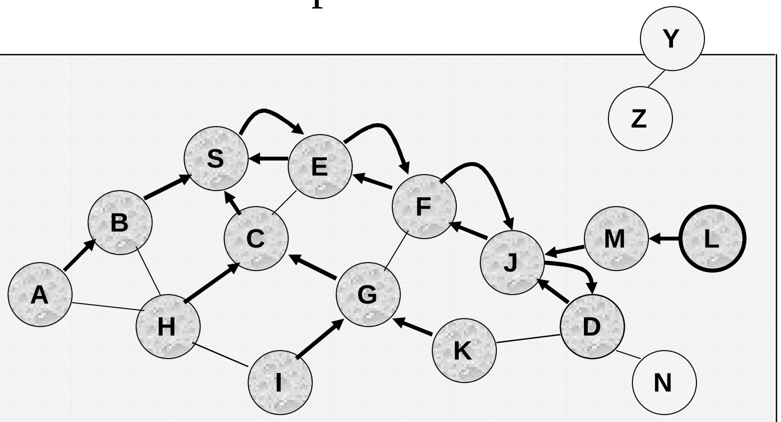


← Represents links on path taken by RREP

Route Reply in AODV

- An intermediate node (not the destination) may also send a Route Reply (RREP) provided that it knows a more recent path than the one previously known to sender S
- To determine whether the path known to an intermediate node is more recent, destination sequence numbers are used
- The likelihood that an intermediate node will send a Route Reply when using AODV not very high
  - A new Route Request by node S for a destination is assigned a higher destination sequence number. An intermediate node which knows a route, but with a smaller sequence number, cannot send Route Reply

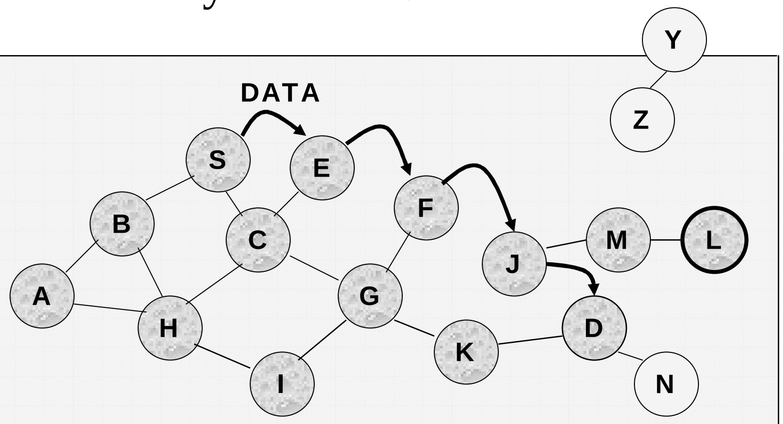
Forward Path Setup in AODV



Forward links are setup when RREP travels alon the reverse path

↷ Represents a link on the forward path

Data Delivery in AODV

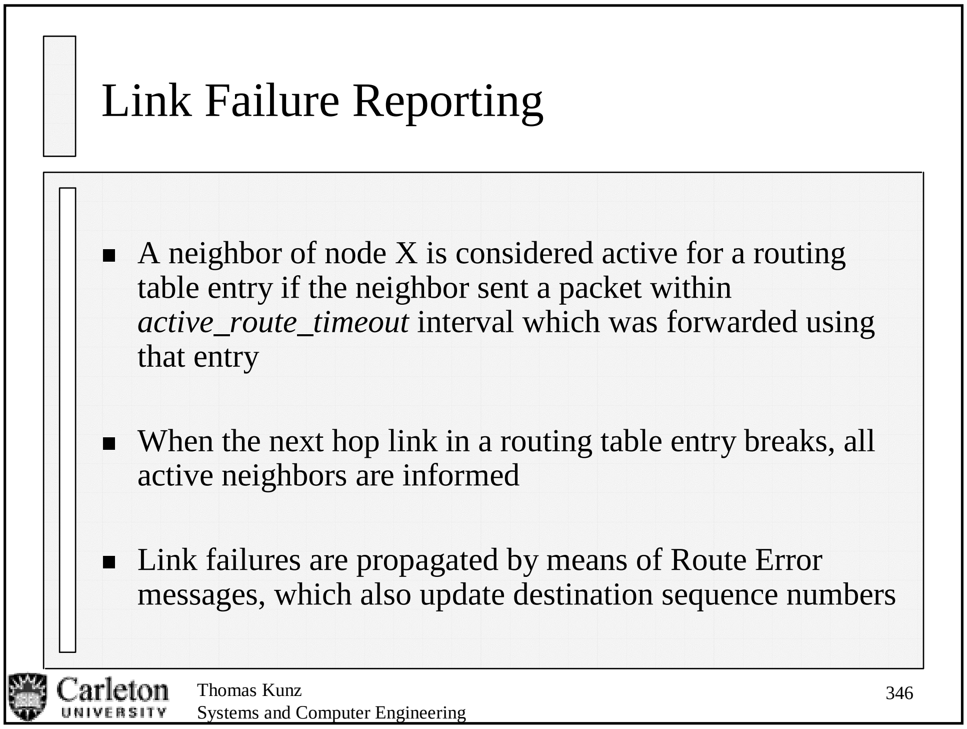


Routing table entries used to forward data packet.

Route is *not* included in packet header.

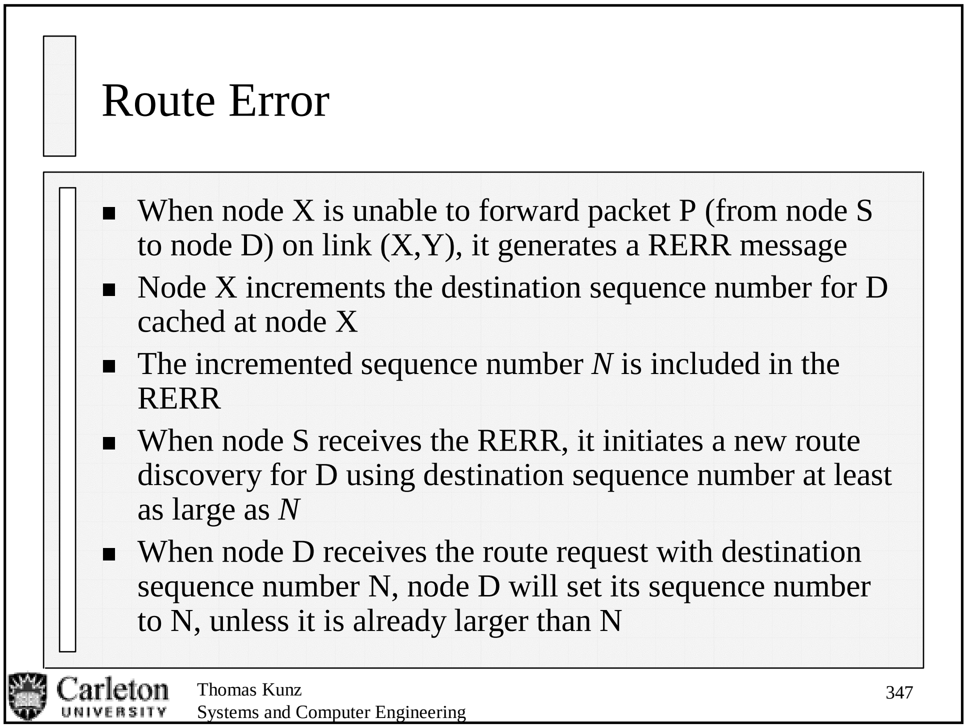
Timeouts

- A routing table entry maintaining a reverse path is purged after a timeout interval
  - timeout should be long enough to allow RREP to come back
- A routing table entry maintaining a forward path is purged if *not used* for a *active_route_timeout* interval
  - if no is data being sent using a particular routing table entry, that entry will be deleted from the routing table (even if the route may actually still be valid)



Link Failure Reporting

- A neighbor of node X is considered active for a routing table entry if the neighbor sent a packet within *active_route_timeout* interval which was forwarded using that entry
- When the next hop link in a routing table entry breaks, all active neighbors are informed
- Link failures are propagated by means of Route Error messages, which also update destination sequence numbers



Route Error

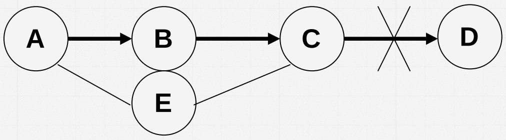
- When node X is unable to forward packet P (from node S to node D) on link (X,Y), it generates a RERR message
- Node X increments the destination sequence number for D cached at node X
- The incremented sequence number *N* is included in the RERR
- When node S receives the RERR, it initiates a new route discovery for D using destination sequence number at least as large as *N*
- When node D receives the route request with destination sequence number N, node D will set its sequence number to N, unless it is already larger than N

Link Failure Detection

- *Hello* messages: Neighboring nodes periodically exchange hello message
- Absence of hello message is used as an indication of link failure
- Alternatively, failure to receive several MAC-level acknowledgement may be used as an indication of link failure

Why Sequence Numbers in AODV

- To avoid using old/broken routes
  - To determine which route is newer
- To prevent formation of loops



  - Assume that A does not know about failure of link C-D because RERR sent by C is lost
  - Now C performs a route discovery for D. Node A receives the RREQ (say, via path C-E-A)
  - Node A will reply since A knows a route to D via node B
  - Results in a loop (for instance, C-E-A-B-C)

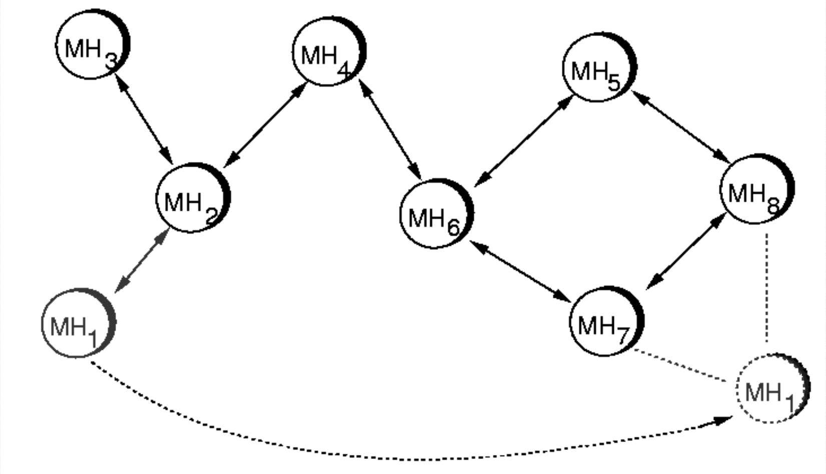
Optimization: Expanding Ring Search

- Route Requests are initially sent with small Time-to-Live (TTL) field, to limit their propagation
  - Common approach to limit flooding
- If no Route Reply is received, then larger TTL tried

Summary: AODV

- Routes not included in packet headers
- Nodes maintain routing tables containing entries only for routes that are in active use
- At most one next-hop per destination maintained at each node
  - Other protocols may maintain several routes for a single destination
- Unused routes expire even if topology does not change

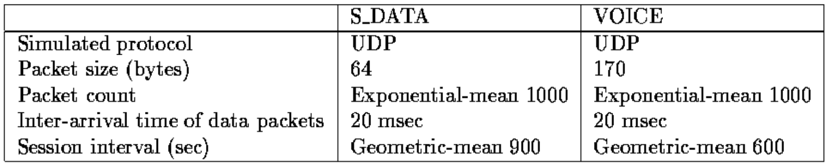
AODV Example



AODV Example

- Suppose MH1 moves away from MH2 towards MH7 and has active sessions with MH3 and MH6:
  - MH2 notices that its link to MH1 is broken
  - MH2 checks its routing table and finds that its link to MH1 was actively in use by MH3 and MH4
  - MH2 unicasts an ∞-metric route update, with an incremented destination sequence number, to MH3 and MH4. MH3 may subsequently issue a new route request for MH4
  - MH4 also notes that its route to MH1 was actively in use and forwards a ∞ -metric route update to MH6
  - the ∞-metric route update for MH1 may also be included in the next periodic limited broadcast issued by MH2
  - MH6 may subsequently issue a new route request for MH1
  - any subsequent route request for MH1 which is satisfied by a route reply through MH2 may cause MH2 to update its route table

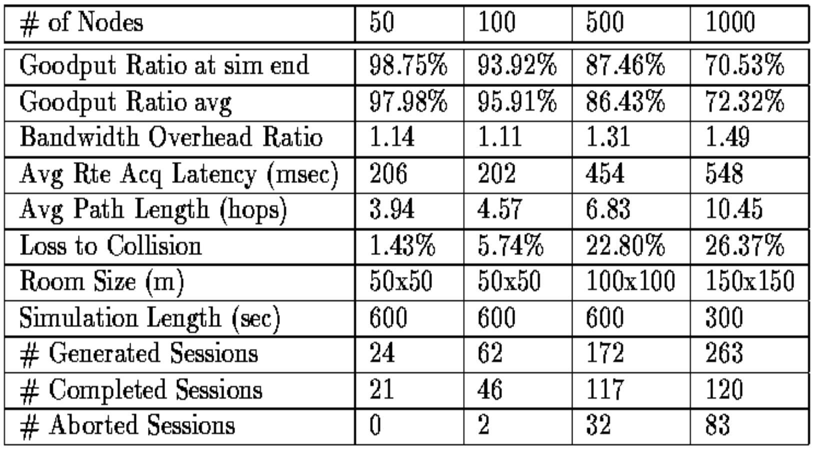
AODV Performance Evaluation



	S_DATA	VOICE
Simulated protocol	UDP	UDP
Packet size (bytes)	64	170
Packet count	Exponential-mean 1000	Exponential-mean 1000
Inter-arrival time of data packets	20 msec	20 msec
Session interval (sec)	Geometric-mean 900	Geometric-mean 600

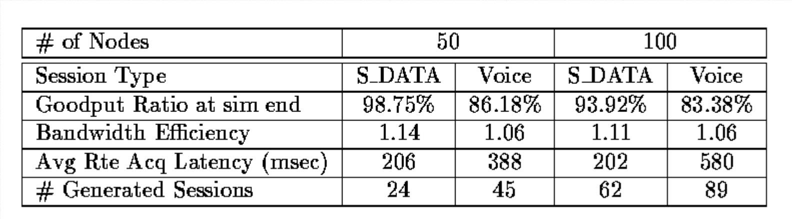
- Nodes move randomly with geographic region
  - speed chosen from uniform distribution between 0.4 and 0.8 meters/second
  - once they arrive at random location, rest for period chosen from uniform distribution between 60 and 300 seconds
- 2 nodes can communicate directly (are neighbors) if they are less than 10 meters apart

AODV Performance Evaluation



# of Nodes	50	100	500	1000
Goodput Ratio at sim end	98.75%	93.92%	87.46%	70.53%
Goodput Ratio avg	97.98%	95.91%	86.43%	72.32%
Bandwidth Overhead Ratio	1.14	1.11	1.31	1.49
Avg Rte Acq Latency (msec)	206	202	454	548
Avg Path Length (hops)	3.94	4.57	6.83	10.45
Loss to Collision	1.43%	5.74%	22.80%	26.37%
Room Size (m)	50x50	50x50	100x100	150x150
Simulation Length (sec)	600	600	600	300
# Generated Sessions	24	62	172	263
# Completed Sessions	21	46	117	120
# Aborted Sessions	0	2	32	83

AODV Performance Evaluation



# of Nodes	50		100	
Session Type	S_DATA	Voice	S_DATA	Voice
Goodput Ratio at sim end	98.75%	86.18%	93.92%	83.38%
Bandwidth Efficiency	1.14	1.06	1.11	1.06
Avg Rte Acq Latency (msec)	206	388	202	580
# Generated Sessions	24	45	62	89

- Voice: longer packets, resulting in more collisions and longer queue lengths at intermediate nodes, which in turn delays RREQ and RREP messages and increases route acquisition latency
- Bandwidth efficiency slightly higher, since number of “control” messages the same, but more data being send

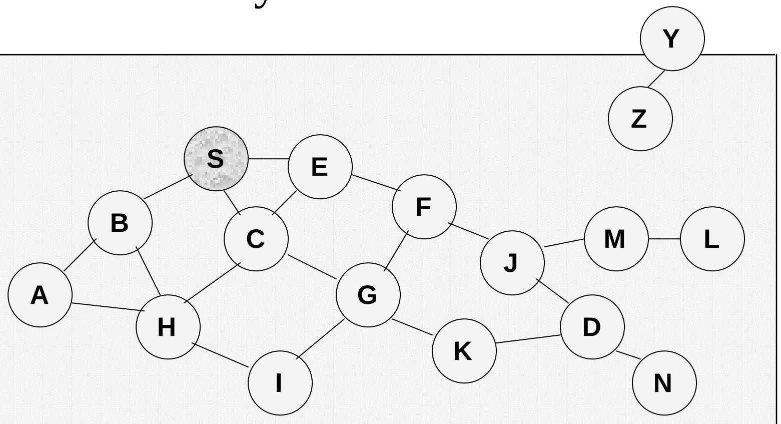
DSR: Dynamic Source Routing

- Source Routing: sender of packet determines complete sequence of nodes along path and lists them in packet header
- use dynamic route discovery to determine path
- advantages:
  - no periodic routing advertisement messages
    - saves bandwidth when there is little change in network
    - saves battery power (no need to send/receive messages)
  - ad-hoc networks have many redundant links, which cause flooding of routing messages
  - no assumption of link symmetry
  - possible to react to changes faster than state-based or distance-vector based protocols
  - better opportunities for route caching and maintenance of alternative routes, compared to AODV

Dynamic Source Routing (DSR)

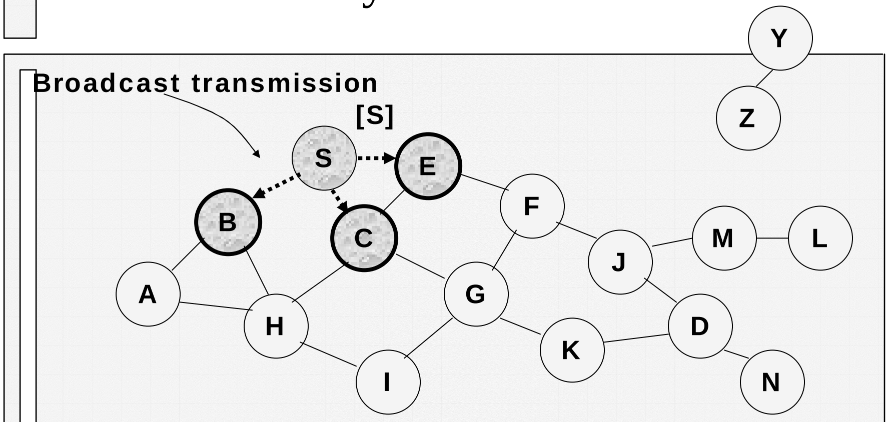
- When node S wants to send a packet to node D, but does not know a route to D, node S initiates a route discovery
- Source node S floods Route Request (RREQ)
- Each node appends own identifier when forwarding RREQ

Route Discovery in DSR



● Represents a node that has received RREQ for D from

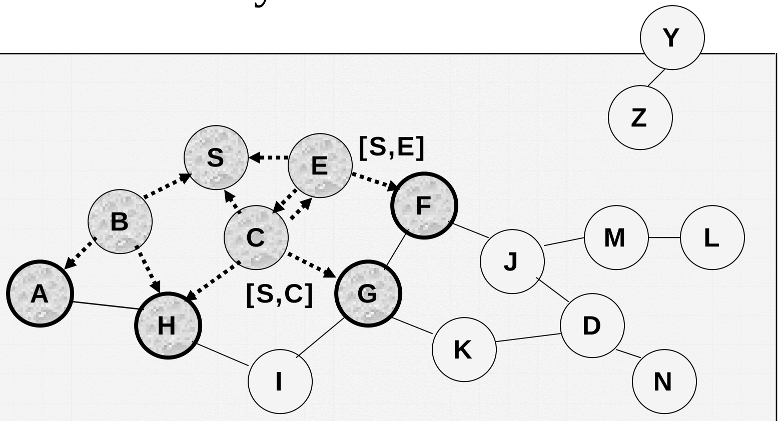
Route Discovery in DSR



⇢ Represents transmission of RREQ

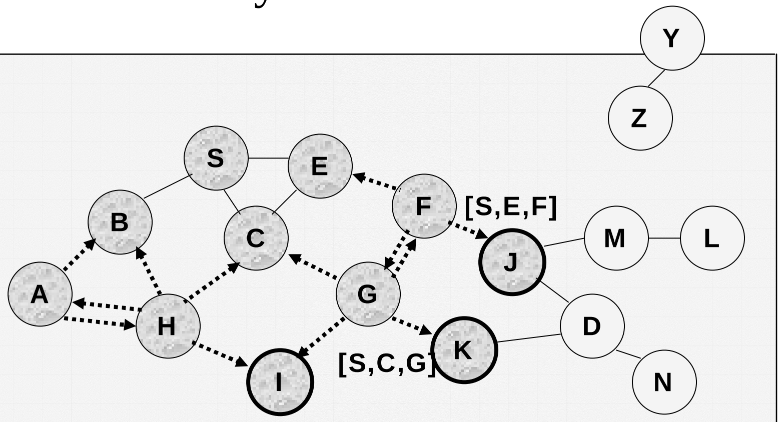
[X,Y] Represents list of identifiers appended to RREQ

Route Discovery in DSR



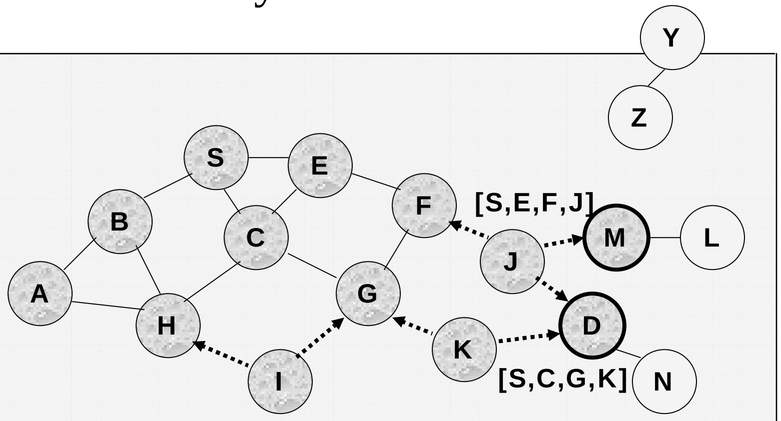
- Node H receives packet RREQ from two neighbors: potential for collision

Route Discovery in DSR



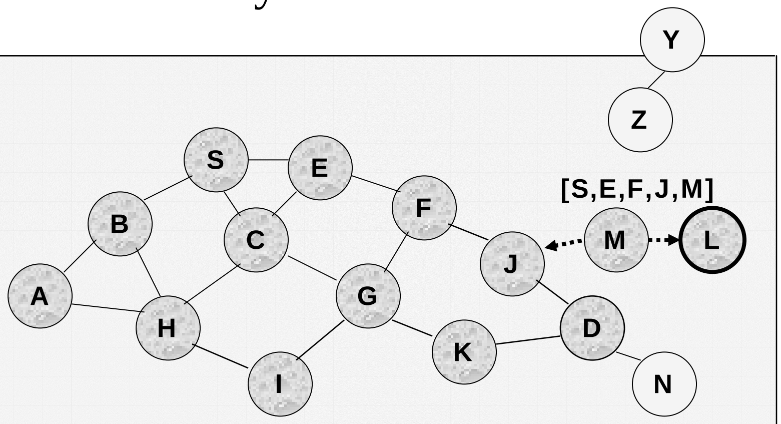
- Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ onc

Route Discovery in DSR



- Nodes J and K both broadcast RREQ to node D
- Since nodes J and K are hidden from each other, their transmissions may collide

Route Discovery in DSR

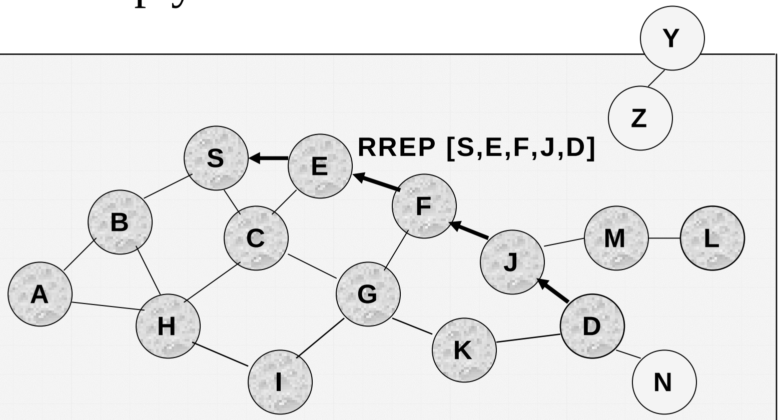


- Node D does not forward RREQ, because node D is the intended target of the route discovery

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 appended to received RREQ
- RREP includes the route from S to D on which RREQ was received by node D

Route Reply in DSR



← Represents RREP control message

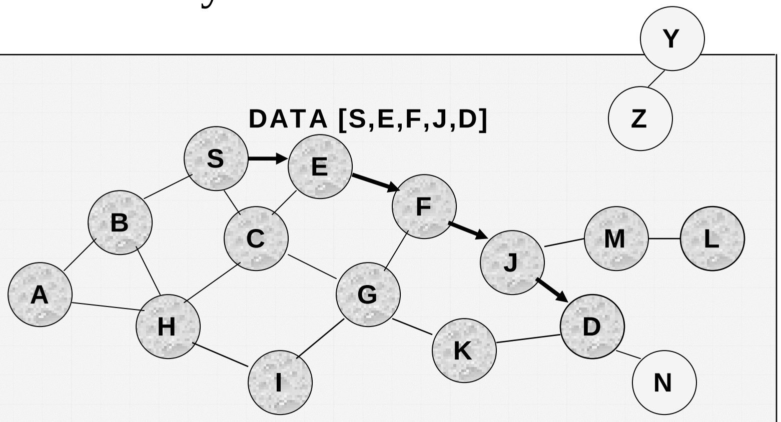
Route Reply in DSR

- Route Reply can be sent by reversing the route in Route Request (RREQ) only if links are guaranteed to be bi-directional
  - To ensure this, RREQ should be forwarded only if it received on a link that is known to be bi-directional
- If unidirectional (asymmetric) links are allowed, then RREP may need a route discovery for S from node D
  - Unless node D already knows a route to node S
  - If a route discovery is initiated by D for a route to S, then the Route Reply is piggybacked on the Route Request from D.
- If IEEE 802.11 MAC is used to send data, then links have to be bi-directional (since Ack is used)

Dynamic Source Routing (DSR)

- Node S on receiving RREP, caches the route included in the RREP
- When node S sends a data packet to D, the entire route is included in the packet header
  - hence the name source routing
- Intermediate nodes use the source route included in a packet to determine to whom a packet should be forwarded

Data Delivery in DSR

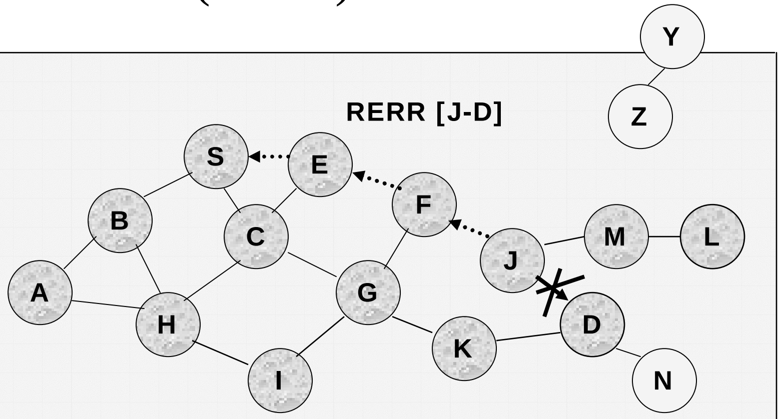

DATA [S,E,F,J,D]

Packet header size grows with route length

When to Perform a Route Discovery

- When node S wants to send data to node D, but does not know a valid route node D

Route Error (RERR)



J sends a route error to S along route J-F-E-S when its attempt to forward the data packet S (with route SEFJD) on J-D fails

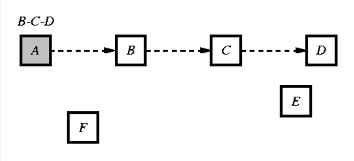
Nodes hearing RERR update their route cache to remove link J-D

DSR Optimization: Route Caching

- Each node caches a new route it learns by *any means*
- When node S finds route [S,E,F,J,D] to node D, node S also learns route [S,E,F] to node F
- When node K receives Route Request [S,C,G] destined for node, node K learns route [K,G,C,S] to node S
- When node F forwards Route Reply RREP [S,E,F,J,D], node F learns route [F,J,D] to node D
- When node E forwards Data [S,E,F,J,D] it learns route [E,F,J,D] to node D
- A node may also learn a route when it overhears Data packets

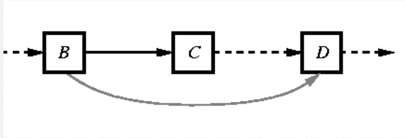
DSR Optimizations

- Route cache
  - intermediate nodes learn about routes by “snooping” on route request/reply messages and/or routing data
  - intermediate nodes may learn routes by listening to route discovery through neighboring nodes



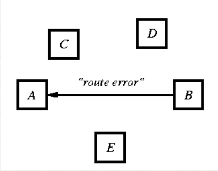
DSR Optimizations

- piggy-backing data onto route discovery to reduce latency
  - if node that is not target replies to route request, special attention is needed to make sure that the data is forwarded to final destination
- reflecting shorter routes
  - usually, as nodes move closer, they will also move out of range in existing route, so route maintenance will take care of this
  - alternatively, run network interfaces in promiscuous mode, listen to who is in your neighborhood and see whether routes in cache can be shortened



DSR Optimizations

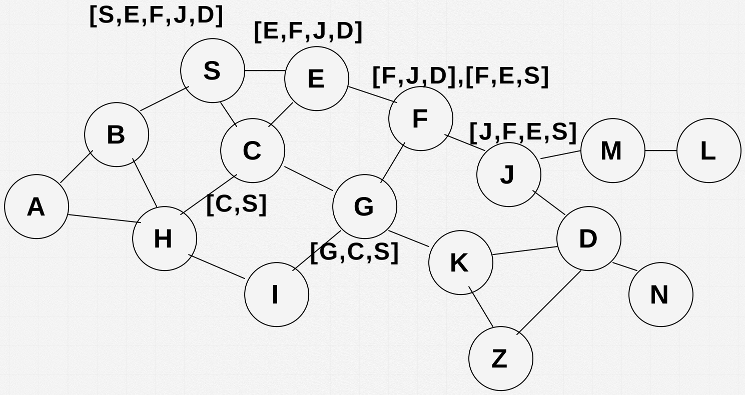
- handling network partitions
  - if nodes cannot communicate, lots of failed route request messages, with corresponding overhead
  - limit route requests using exponential backoff algorithm
- eavesdrop on route error packets (promiscuous mode)
  - error packets detail hop that is down
  - other nodes can update/invalidate routes that use this hop in their cache
  - keep negative information (hops that are down) in cache for a brief period to make sure new routes do not use outdated hop



Use of Route Caching

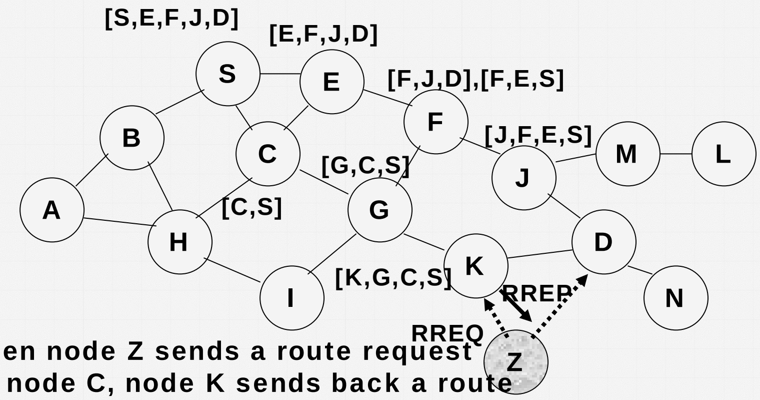
- When node S learns that a route to node D is broken, it uses another route from its local cache, if such a route to D exists in its cache. Otherwise, node S initiates route discovery by sending a route request
- Node X on receiving a Route Request for some node D can send a Route Reply if node X knows a route to node D
- Use of route cache
  - can speed up route discovery
  - can reduce propagation of route requests

Use of Route Caching



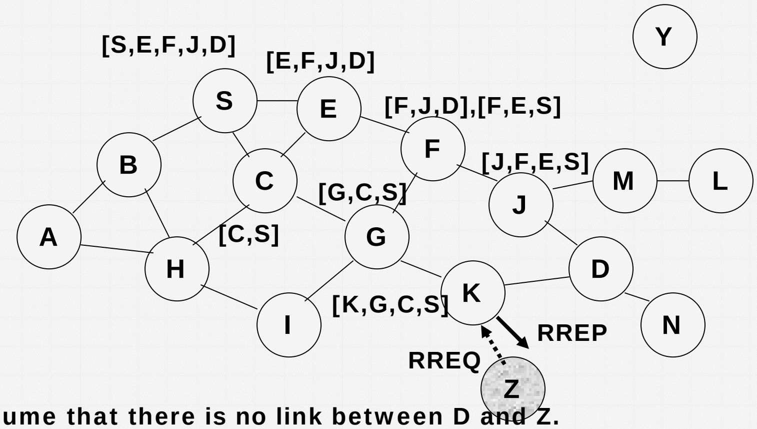
[P,Q,R] Represents cached route at a node
(DSR maintains the cached routes in a tree format)

Use of Route Caching: Speed up Route Discovery



When node Z sends a route request for node C, node K sends back a route reply [Z,K,G,C] to node Z using a locally cached route

Use of Route Caching: Reduce Propagation of Route Requests



Assume that there is no link between D and Z.
Route Reply (RREP) from node K limits flooding of RREQ.
In general, the reduction may be less dramatic.

Route Caching: Beware!

- Stale caches can adversely affect performance
- With passage of time and host mobility, cached routes may become invalid
- A sender host may try several stale routes (obtained from local cache, or replied from cache by other nodes), before finding a good route
- This is particularly bad for protocols such as TCP, where the sender will timeout, assume severe congestion, and drastically reduce the rate at which data is sent.

Dynamic Source Routing: Advantages

- Routes maintained only between nodes who need to communicate
  - reduces overhead of route maintenance
- Route caching can further reduce route discovery overhead
- A single route discovery may yield many routes to the destination, due to intermediate nodes replying from local caches

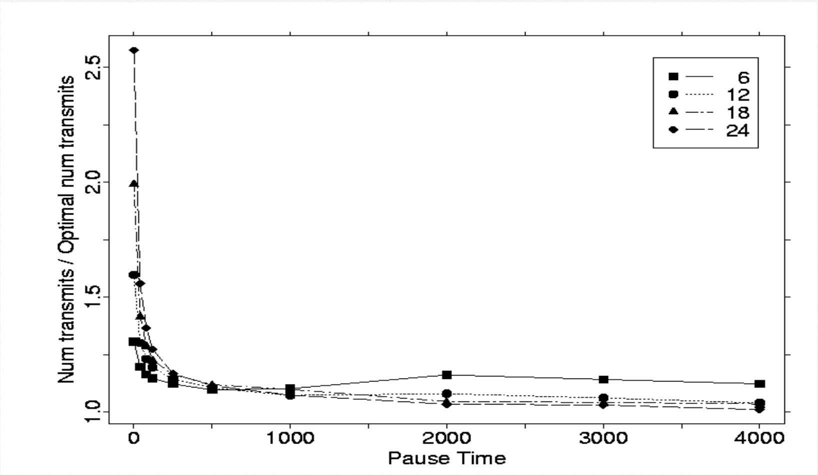
Dynamic Source Routing: Disadvantages

- Packet header size grows with route length due to source routing
- Flood of route requests may potentially reach all nodes in the network
- Care must be taken to avoid collisions between route requests propagated by neighboring nodes
  - insertion of random delays before forwarding RREQ
- Increased contention if too many route replies come back due to nodes replying using their local cache
  - Route Reply Storm problem
  - Reply storm may be eased by preventing a node from sending RREP if it hears another RREP with a shorter route

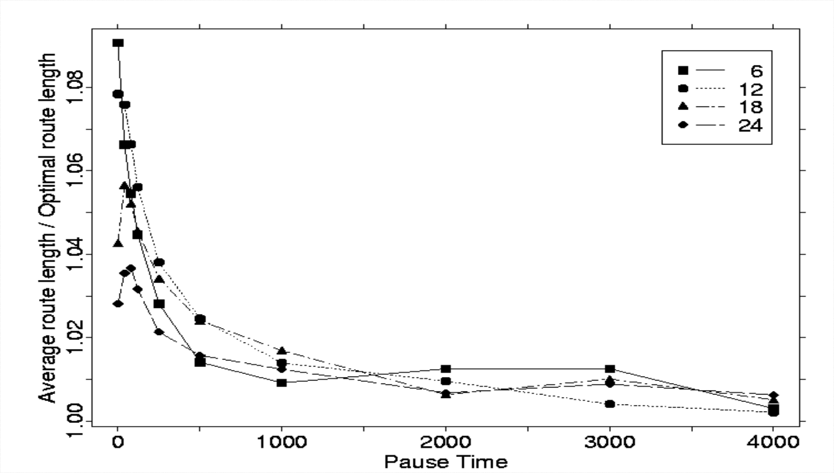
Dynamic Source Routing: Disadvantages

- An intermediate node may send Route Reply using a stale cached route, thus polluting other caches
- This problem can be eased if some mechanism to purge (potentially) invalid cached routes is incorporated.

DSR Performance

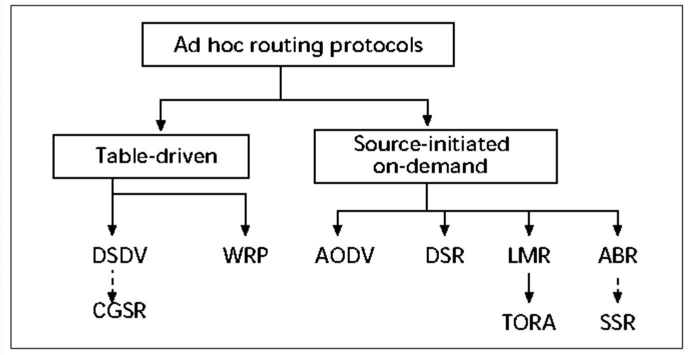


DSR Performance



Ad-Hoc Routing Schemes

(see also `http://alpha.ece.ucsb.edu/~eroyer/txt/review.ps`)



Ad hoc routing protocols

Table-driven

Source-initiated on-demand

DSDV

WRP

AODV

DSR

LMR

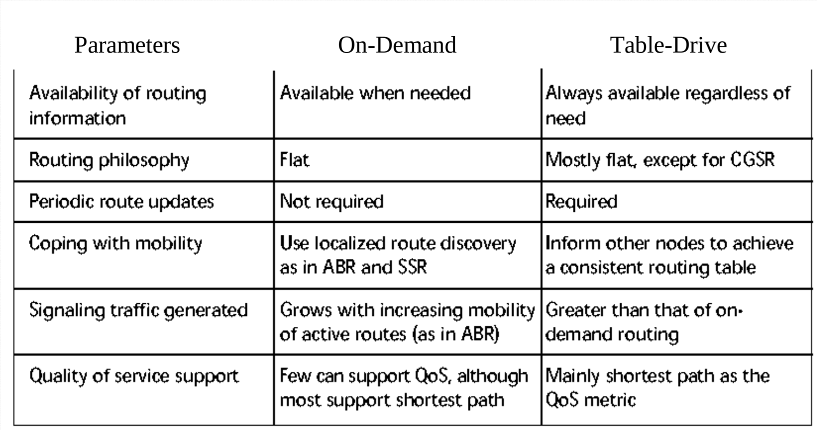
ABR

CGSR

TORA

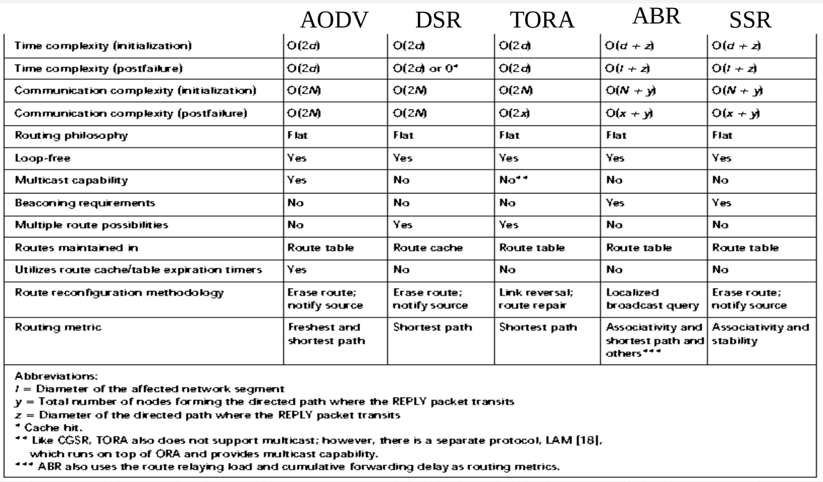
SSR

Comparison On-Demand vs. Table-Driven



Parameters	On-Demand	Table-Drive
Availability of routing information	Available when needed	Always available regardless of need
Routing philosophy	Flat	Mostly flat, except for CGSR
Periodic route updates	Not required	Required
Coping with mobility	Use localized route discovery as in ABR and SSR	Inform other nodes to achieve a consistent routing table
Signaling traffic generated	Grows with increasing mobility of active routes (as in ABR)	Greater than that of on-demand routing
Quality of service support	Few can support QoS, although most support shortest path	Mainly shortest path as the QoS metric

Comparison of Source-Initiated Protocols



	AODV	DSR	TORA	ABR	SSR
Time complexity (initialization)	$O(2d)$	$O(2d)$	$O(2d)$	$O(d + z)$	$O(d + z)$
Time complexity (postfailure)	$O(2d)$	$O(2d)$ or 0*	$O(2d)$	$O(l + z)$	$O(l + z)$
Communication complexity (initialization)	$O(2N)$	$O(2N)$	$O(2N)$	$O(N + y)$	$O(N + y)$
Communication complexity (postfailure)	$O(2N)$	$O(2N)$	$O(2x)$	$O(x + y)$	$O(x + y)$
Routing philosophy	Flat	Flat	Flat	Flat	Flat
Loop-free	Yes	Yes	Yes	Yes	Yes
Multicast capability	Yes	No	No**	No	No
Beaconing requirements	No	No	No	Yes	Yes
Multiple route possibilities	No	Yes	Yes	No	No
Routes maintained in	Route table	Route cache	Route table	Route table	Route table
Utilizes route cache/table expiration timers	Yes	No	No	No	No
Route reconfiguration methodology	Erase route; notify source	Erase route; notify source	Link reversal; route repair	Localized broadcast query	Erase route; notify source
Routing metric	Freshest and shortest path	Shortest path	Shortest path	Associativity and shortest path and others***	Associativity and stability

Abbreviations:
l = Diameter of the affected network segment
y = Total number of nodes forming the directed path where the REPLY packet transits
z = Diameter of the directed path where the REPLY packet transits
* Cache hit.
** Like CGSR, TORA also does not support multicast; however, there is a separate protocol, LAM [18], which runs on top of ORA and provides multicast capability.
*** ABR also uses the route relaying load and cumulative forwarding delay as routing metrics.

Ad-Hoc Routing Algorithm Performance

(see also `http://www.monarch.cs.cmu.edu/monarch-papers/mobicom98.ps`)

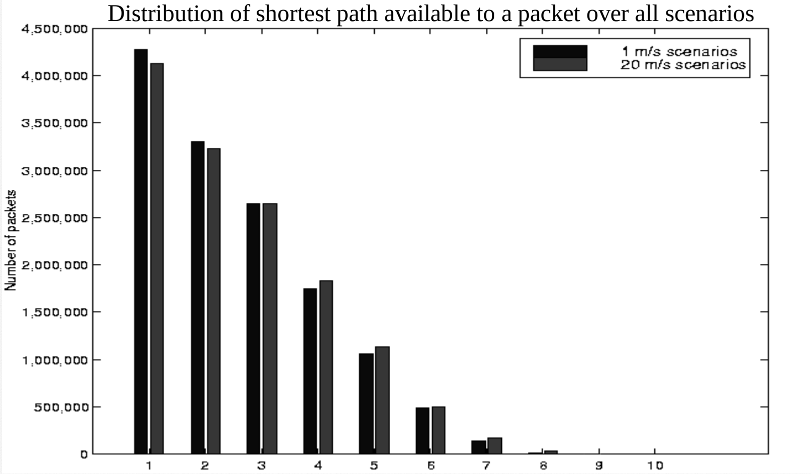
- each paper/proposal tends to come with its own set of performance evaluations, based on some simulated environment
- similar to location management proposals, no common performance evaluation framework exists
- one study (November 1998): implement a number of proposals in NS and evaluate them, compare relative advantages and disadvantages
- catch: study done by authors of DSR, which ends up being found superior

Ad-Hoc Routing: Performance Comparison

- Movement Model:
  - 1500m x 300m space
  - user picks random location, moves there at a speed randomly distributed between 0 and 20 meters/sec
  - stay at new location for *pause time* second, repeat
- Communication Model:
  - 10, 20, or 30 CBR sources, sending 4 packets/sec
  - packet size is fixed at 64 bytes
  - all communication peer-to-peer, starts at times uniformly distributed between 0 and 180 seconds
  - transmission range of 250 meters for each radio

Ad-Hoc Routing: Performance Comparison

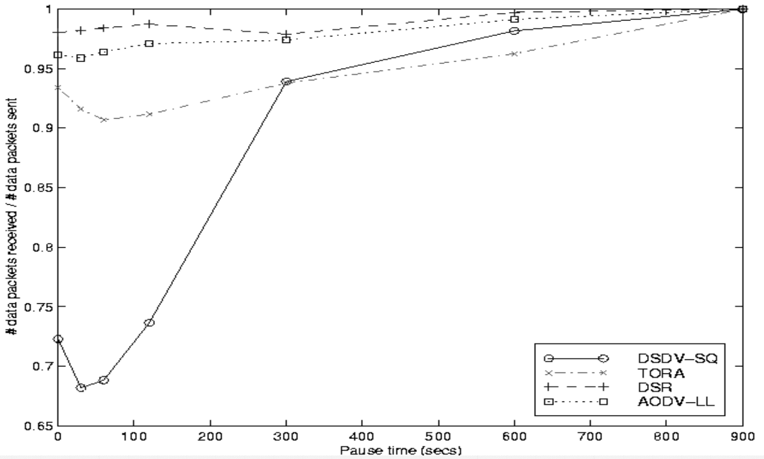
Distribution of shortest path available to a packet over all scenarios



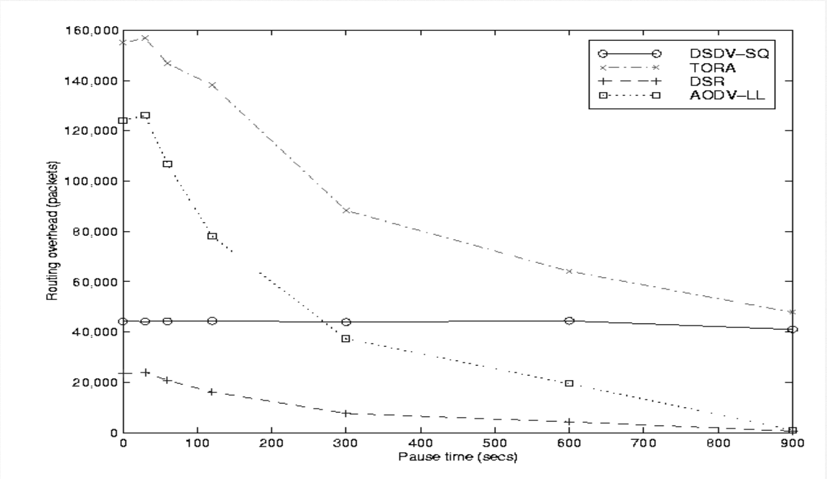
Ad-Hoc Routing: Performance Comparison

- Performance Metrics:
  - packet delivery ratio
  - routing overhead
  - path optimality
- Not considered:
  - latency to establish route
  - goodput
  - route failures (i.e., routes break and connection times out before new route can be established)

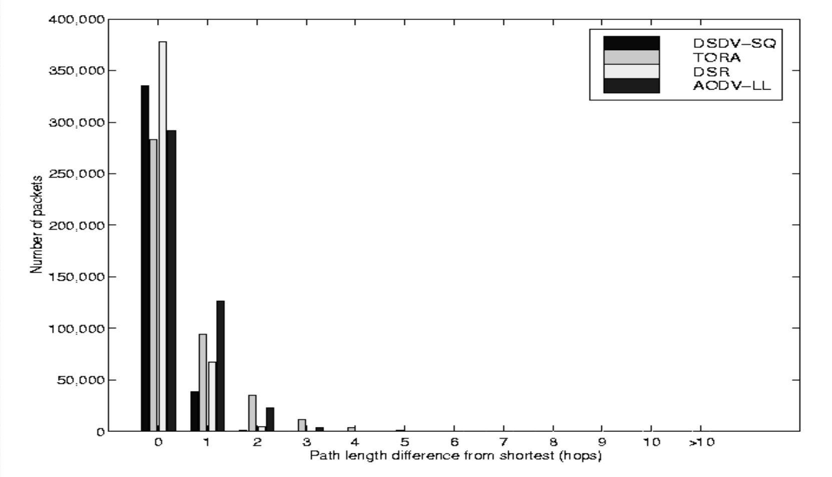
Ad-Hoc Routing: Performance Comparison



Ad-Hoc Routing: Performance Comparison



Ad-Hoc Routing: Performance Comparison



Ad-Hoc Routing: Performance Comparison

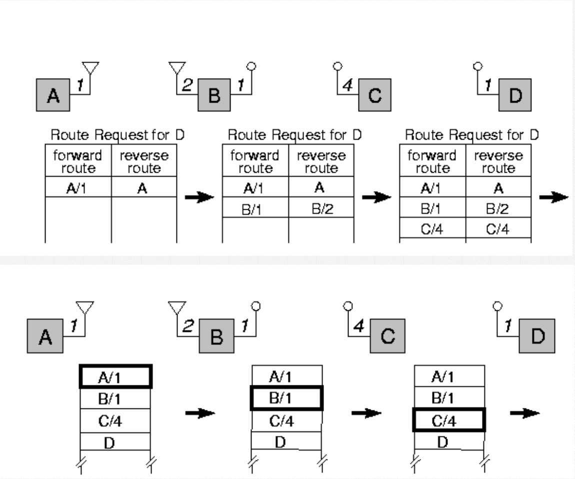
- Conclusions of study:
  - each protocol performs well in come cases and has drawbacks in others
  - DSR performs predictably
    - delivers virtually all data packets at low mobility rate and speed
    - fails to converge as node mobility increases
  - AODV performs nearly as well as DSR at all mobility rates and speeds
    - accomplishes its goal of eliminating source routing overhead
    - requires transmission of many routing overhead packets
    - is more expensive than DSR of high rates of node mobility

Future Issues: Heterogeneous Interfaces

(see also `http://www.monarch.cs.cmu.edu/monarch-papers/ispan99.ps`)

Route Discovery

Data Transmission



Route Request for D

forward route	reverse route
A/1	A

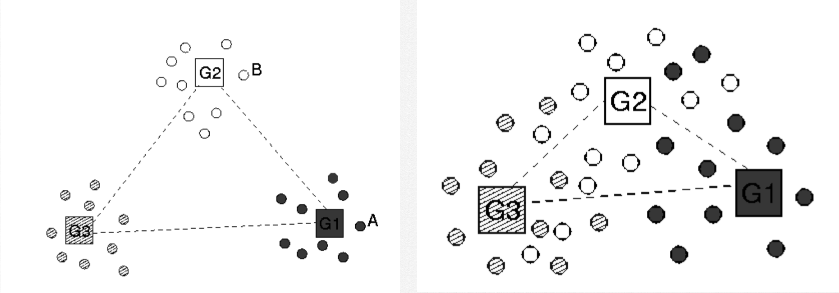
Route Request for D

forward route	reverse route
A/1	A
B/1	B/2

Route Request for D

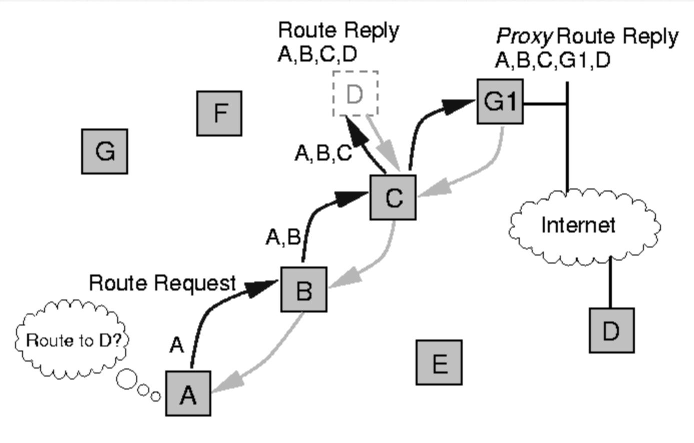
forward route	reverse route
A/1	A
B/1	B/2
C/4	C/4

Future Issues: Hierarchical Networks

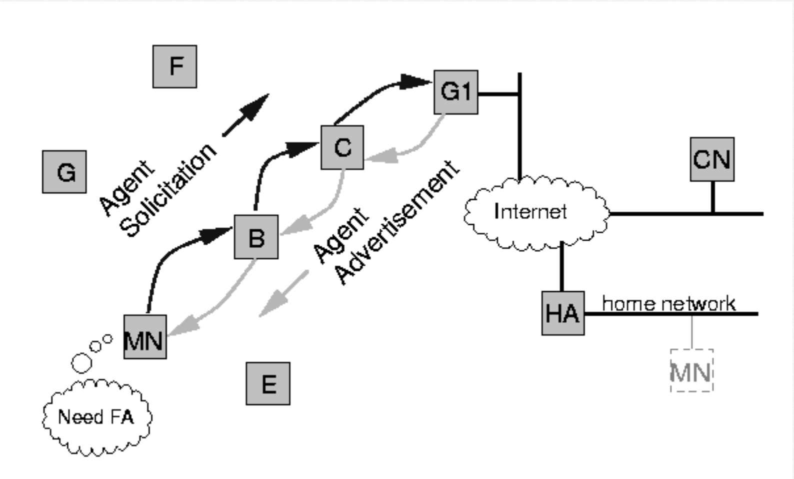


Hierarchy:
- reduces routing tables
- reduces routing overhead
- limits route discovery

Future Issues: Integration with Internet



Future Issues: Integration with Mobile IP



Multicasting

- A multicast group is defined with a unique *group identifier*
- Nodes may join or leave the multicast group anytime
- In traditional networks, the physical network topology does not change often
- In MANET, the physical topology can change often

Multicasting in MANET

- Need to take topology change into account when designing a multicast protocol
- Several new protocols have been proposed for multicasting in MANET

AODV Multicasting

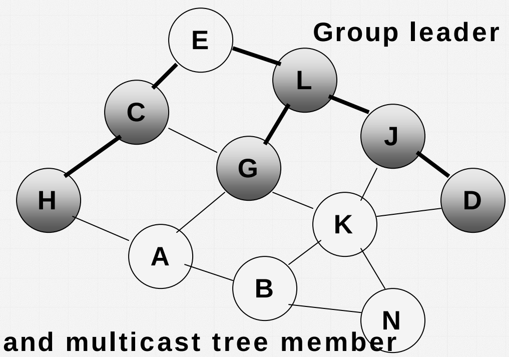
- Each multicast group has a group leader
- Group leader is responsible for maintaining group sequence number (which is used to ensure freshness of routing information)
  - Similar to sequence numbers for AODV unicast
- First node joining a group becomes *group leader*
- A node on becoming a group leader, broadcasts a *Group Hello* message

AODV Group Sequence Number

- In our illustrations, we will ignore the group sequence numbers
- However, note that a node makes use of information received only with *recent enough* sequence number

AODV Multicast Tree

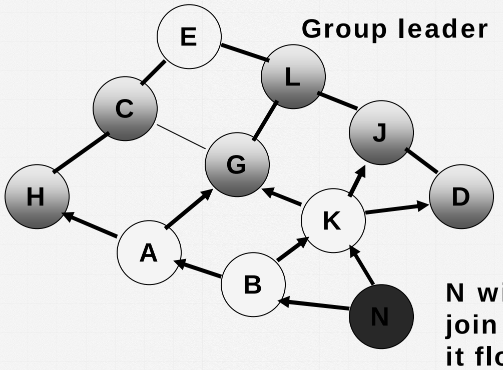
—— Multicast tree links



● Group and multicast tree member

○ Tree (but not group) member

Joining the Multicast Tree: AODV

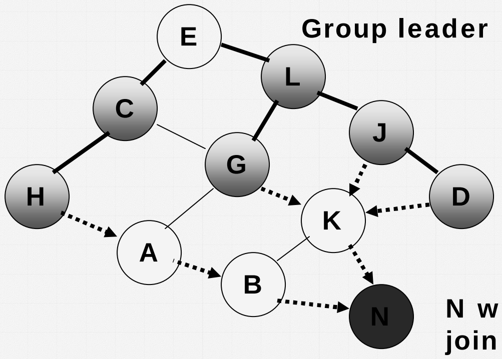


Group leader

N wishes to join the group: it floods RREQ

← Route Request (RREQ)

Joining the Multicast Tree: AODV

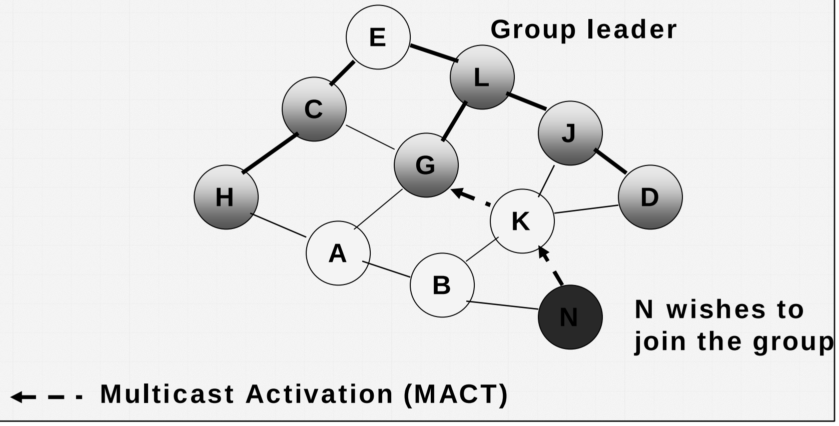


Group leader

N wishes to join the group

←······· Route Reply (RREP)

Joining the Multicast Tree: AODV

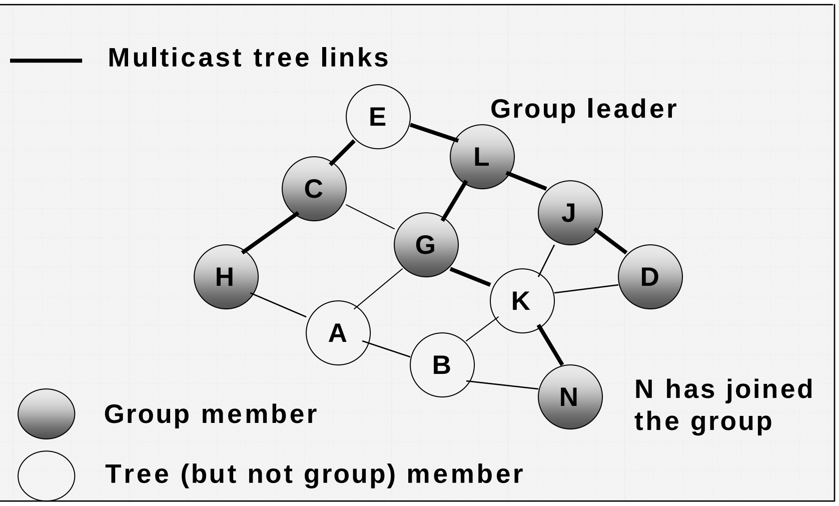


Group leader

N wishes to join the group

Multicast Activation (MACT)

Joining the Multicast Tree: AODV



Multicast tree links

Group leader

N has joined the group

Group member

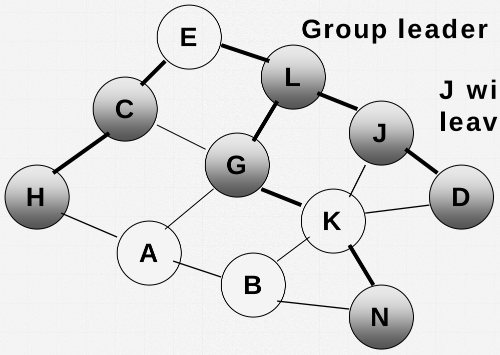
Tree (but not group) member

Sending Data on the Multicast Tree

- Data is delivered along the tree edges maintained by the Multicast AODV algorithm
- If a node which does not belong to the multicast group wishes to multicast a packet
  - It sends a *non-join* RREQ which is treated similar in many ways to RREQ for joining the group
  - As a result, the sender finds a route to a multicast group member
  - Once data is delivered to this group member, the data is delivered to remaining members along multicast tree edges

Leaving a Multicast Tree: AODV

Multicast tree links

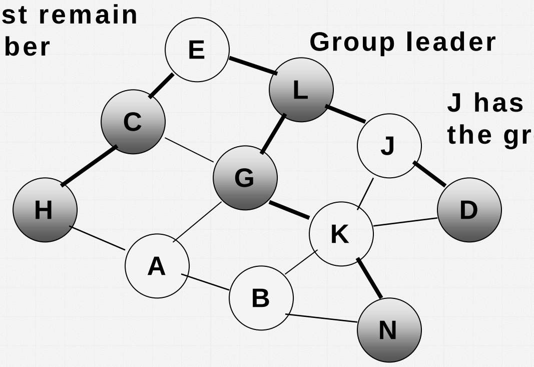


Group leader

J wishes to leave the group

Leaving a Multicast Tree: AODV

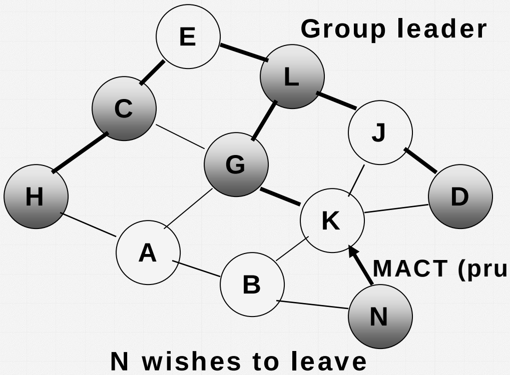
Since J is not a leaf node, it must remain a tree member



Group leader

J has left the group

Leaving a Multicast Tree: AODV

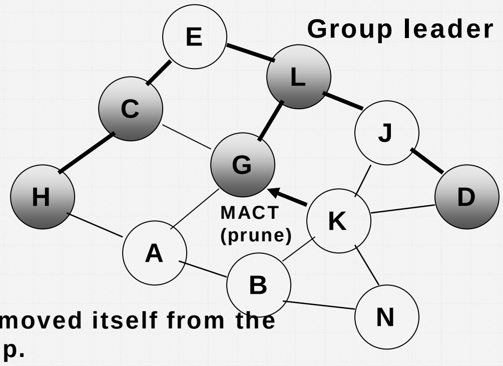


Group leader

MACT (prune)

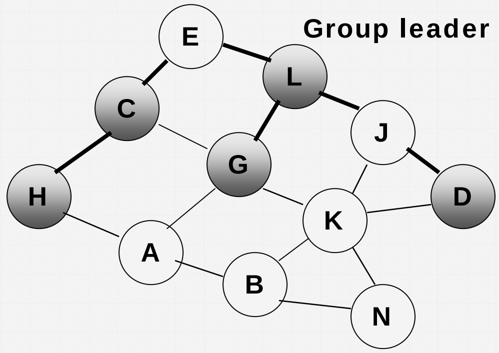
N wishes to leave the multicast group

Leaving a Multicast Tree: AODV



Node N has removed itself from the multicast group.
Now node K has become a leaf, and K is not in the group.
So node K removes itself from the tree as well.

Leaving a Multicast Tree: AODV



Nodes N and K are no more in the multicast tree.

Handling a Link Failure: AODV Multicasting

- When a link (X,Y) on the multicast tree breaks, the node that is further away from the leader is responsible to reconstruct the tree, say node X
- Node X, which is further downstream, transmits a Route Request (RREQ)
  - Only nodes which are closer to the leader than node X’s last known distance are allowed to send RREP in response to the RREQ, to prevent nodes that are further downstream from node X from responding

Handling Partitions: AODV

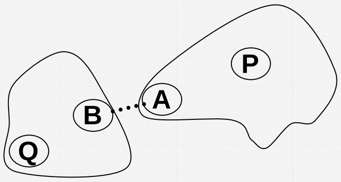
- When failure of link (X,Y) results in a partition, the downstream node, say X, initiates Route Request
- If a Route Reply is not received in response, then node X assumes that it is partitioned from the group leader
- A new group leader is chosen in the partition containing node X
- If node X is a multicast group member, it becomes the group leader, else a group member downstream from X is chosen as the group leader

Merging Partitions: AODV

- If the network is partitioned, then each partition has its own group leader
- When two partitions merge, group leader from one of the two partitions is chosen as the leader for the merged network
  - The leader with the larger identifier remains group leader

Merging Partitions: AODV

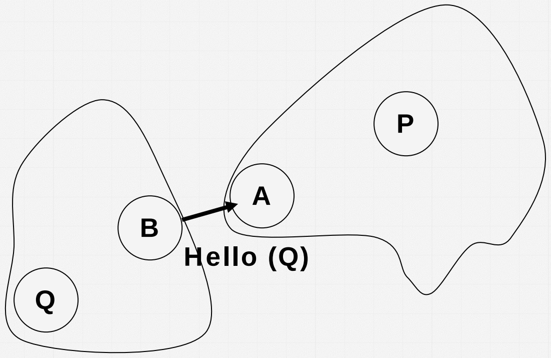
- Each group leader periodically sends Group Hello
- Assume that two partitions exist with nodes P and Q as group leaders, and let P < Q
- Assume that node A is in the same partition as node P, and that node B is in the same partition as node Q
- Assume that a link forms between nodes A and B



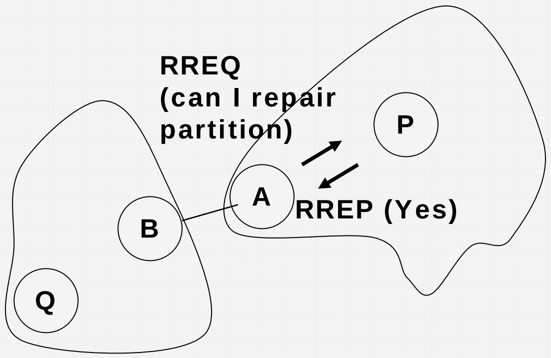
Merging Partitions: AODV

- Assume that node A receives Group Hello originated by node Q through its new neighbor B
- Node A asks exclusive permission from its leader P to merge the two trees using a special Route Request
- Node A sends a special Route Request to node Q
- Node Q then sends a Group Hello message (with a special flag)
- All tree nodes receiving this Group Hello record Q as the leader

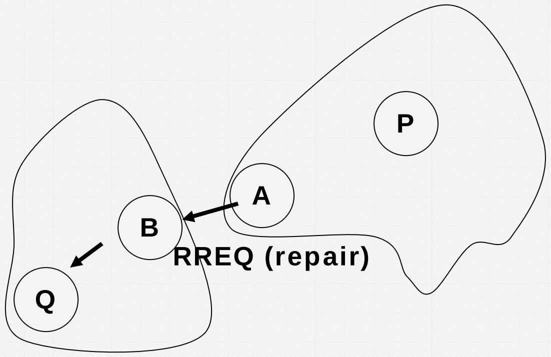
Merging Partitions: AODV



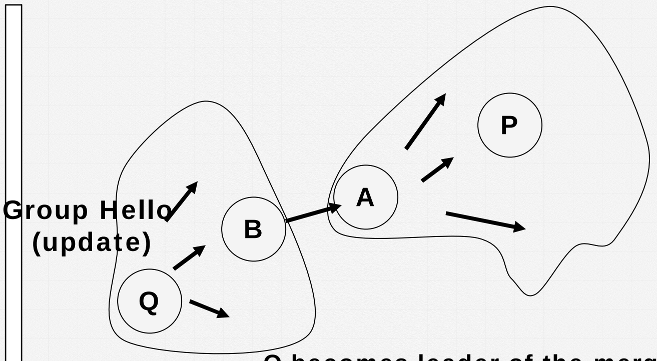
Merging Partitions: AODV



Merging Partitions: AODV



Merging Partitions: AODV



Q becomes leader of the merged multicast tree
New group sequence number is larger than most recent ones known to P and Q both

Summary: Multicast AODV

- Similar to unicast AODV
- Uses leaders to maintain group sequence numbers, and to help in tree maintenance

On-Demand Multicast Routing Protocol (ODMRP)

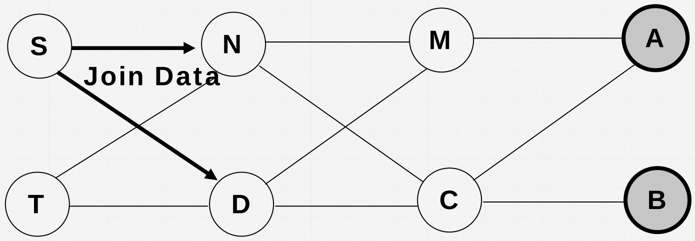
- ODMRP requires cooperation of nodes wishing to send data to the multicast group
  - To construct the multicast *mesh*
- A sender node wishing to send multicast packets *periodically* floods a Join Data packet throughtput the network
  - Periodic transmissions are used to update the routes

On-Demand Multicast Routing Protocol (ODMRP)

- Each multicast group member on receiving a Join Data, broadcasts a Join Table to all its neighbors
  - Join Table contains (sender S, next node N) pairs
  - next node N denotes the next node on the path from the group member to the multicast sender S
- When node N receives the above broadcast, N becomes member of the *forwarding group*
- When node N becomes a forwarding group member, it transmits Join Table containing the entry (S,M) where M is the next hop towards node S

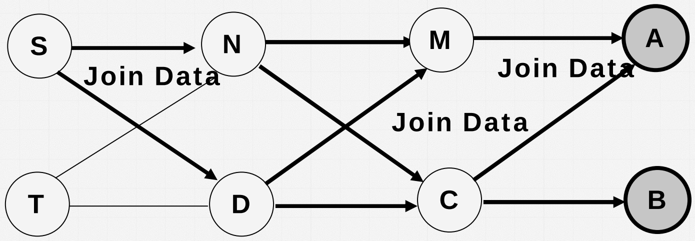
On-Demand Multicast Routing Protocol (ODMRP)

- Assume that S is a sender node



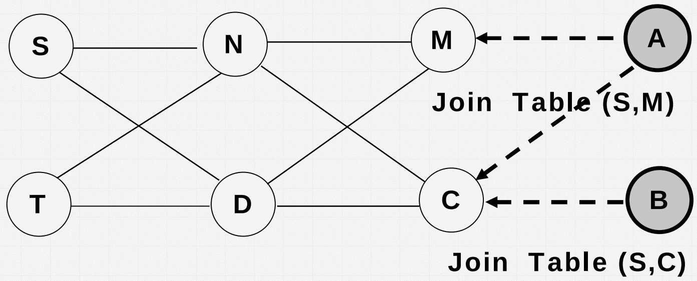
● Multicast group member

On-Demand Multicast Routing Protocol (ODMRP)



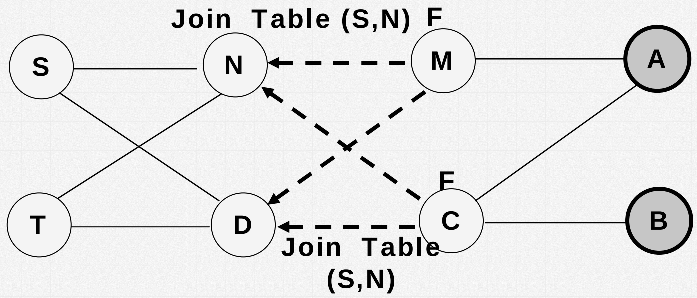
● Multicast group member

On-Demand Multicast Routing Protocol (ODMRP)



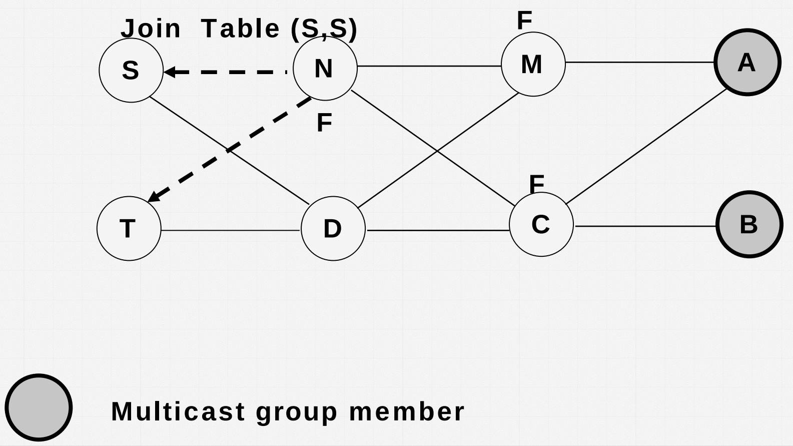
● Multicast group member

On-Demand Multicast Routing Protocol (ODMRP)

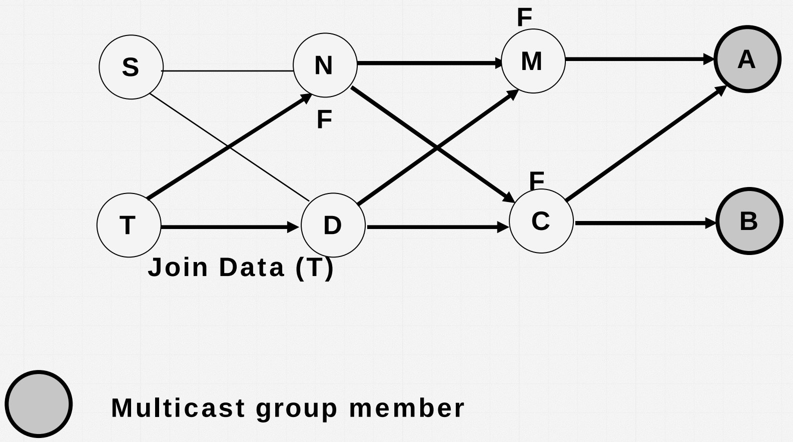


F marks a forwarding group member

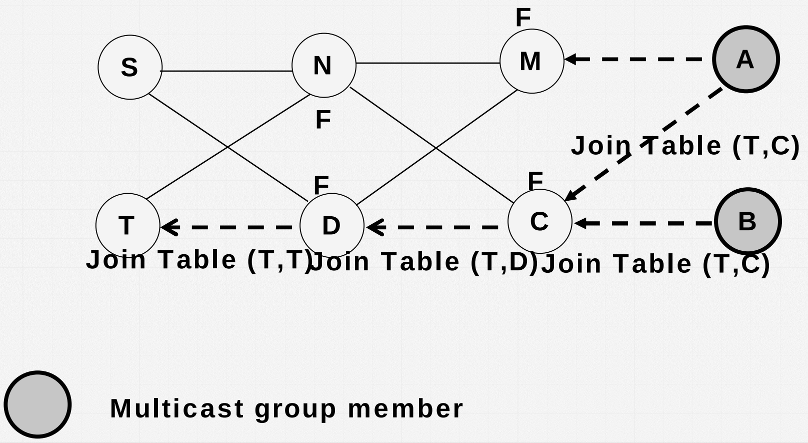
On-Demand Multicast Routing Protocol (ODMRP)



On-Demand Multicast Routing Protocol (ODMRP)



On-Demand Multicast Routing Protocol (ODMRP)



ODMRP Multicast Delivery

- A sender broadcasts data packets to all its neighbors
- Members of the forwarding group forward the packets
- Using ODMRP, multiple routes from a sender to a multicast receiver may exist due to the mesh structure created by the forwarding group members

ODMRP

- No explicit join or leave procedure
- A sender wishing to stop multicasting data simply stops sending Join Data messages
- A multicast group member wishing to leave the group stops sending Join Table messages
- A forwarding node ceases its forwarding status unless refreshed by receipt of a Join Table message
- Link failure/repair taken into account when updating routes in response to periodic Join Data floods from the senders