

# Wireless Networks

(CSC-7602)

## Lecture 9

(29 Oct. 2007)

Seung-Jong Park (Jay)

<http://www.csc.lsu.edu/~sjpark>

1

CSC7602 – S.J. Park

## Paper Presentation

- Procedure
  - 6 papers have been uploaded at class website
  - Each student gives a preference list by marking a rank for each paper and send it via e-mail by this week Wednesday
- Presentation
  - One hour and 30 minutes presentation including Q&A
  - Each student should ask two technical questions for each paper presentation
  - Please write questions for each paper and turn in before a class starts
- Evaluation (10 points per paper)
  - Presentation: 8 points
    - Contents (definition, motivation, algorithms, criticism) = 4 points
    - Delivery (clear and well organized presentation & format) = 2 points
    - Q & A = 2 points
  - Questions: 2 points

2

## Today

- Routing over wireless networks
  - Routing over ad-hoc networks
  - Data diffusion over sensor networks

3

## Unicast Routing Protocol for Mobile Ad-hoc Networks\* (MONET)

\*Based on Dr. Vaidya(@UIUC)'s tutorial

4

## Terminology

- Unicast
- Multicast
- Broadcast
- Anycast

5

## Why is Routing in MANET different ?

- Host mobility
  - link failure/repair due to mobility may have different characteristics than those due to other causes
- Rate of link failure/repair may be high when nodes move fast
- New performance criteria may be used
  - route stability despite mobility
  - energy consumption

6

## Unicast Routing Protocols

- Many protocols have been proposed
- Some have been invented specifically for MANET
- Others are adapted from previously proposed protocols for wired networks
- No single protocol works well in all environments
  - some attempts made to develop adaptive protocols

7

## Routing Protocols

- Proactive protocols
  - Determine routes independent of traffic pattern
  - Traditional link-state and distance-vector routing protocols are proactive
- Reactive protocols
  - Maintain routes only if needed
- Hybrid protocols

8

## Trade-Off

- Latency of route discovery
  - Proactive protocols may have lower latency since routes are maintained at all times
  - Reactive protocols may have higher latency because a route from X to Y will be found only when X attempts to send to Y
  
- Overhead of route discovery/maintenance
  - Reactive protocols may have lower overhead since routes are determined only if needed
  - Proactive protocols can (but not necessarily) result in higher overhead due to continuous route updating
  
- Which approach achieves a better trade-off depends on the traffic and mobility patterns

9

## Overview of Unicast Routing Protocols

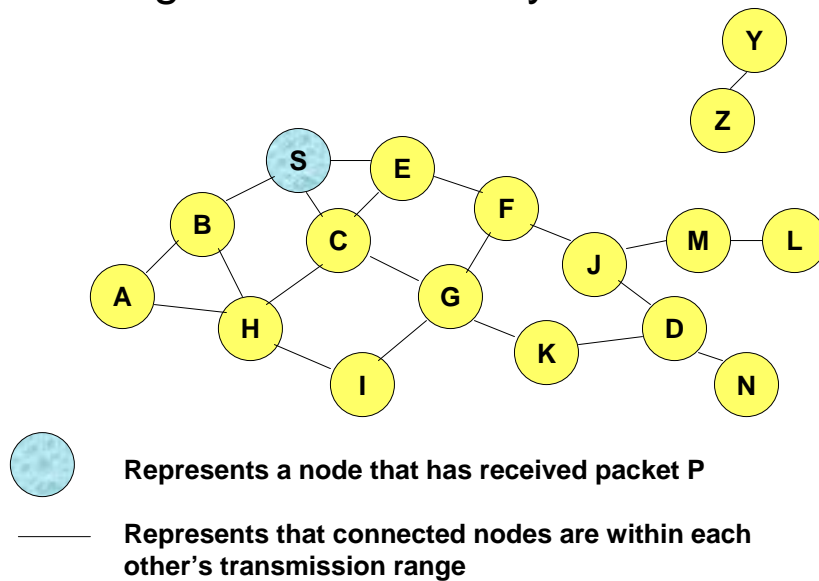
10

## 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

11

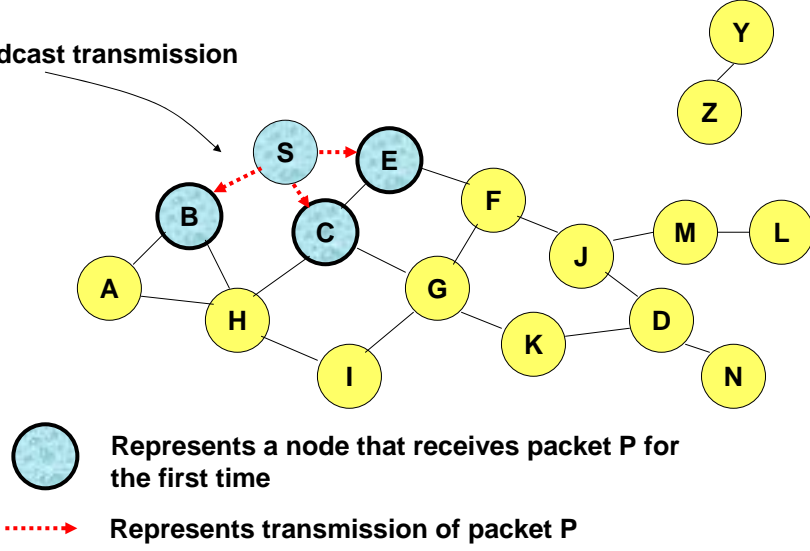
## Flooding for Data Delivery



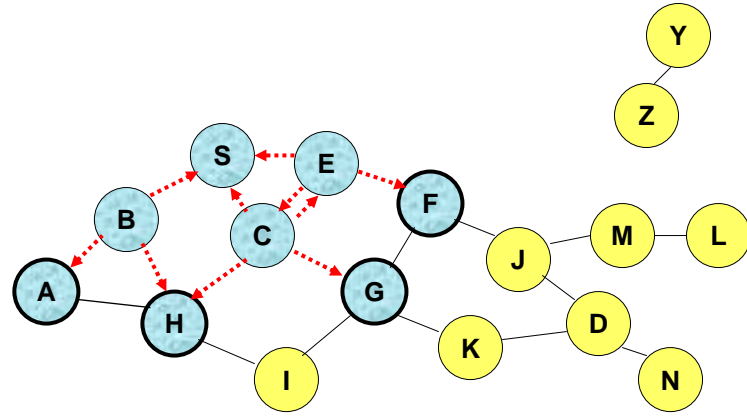
12

# Flooding for Data Delivery

Broadcast transmission

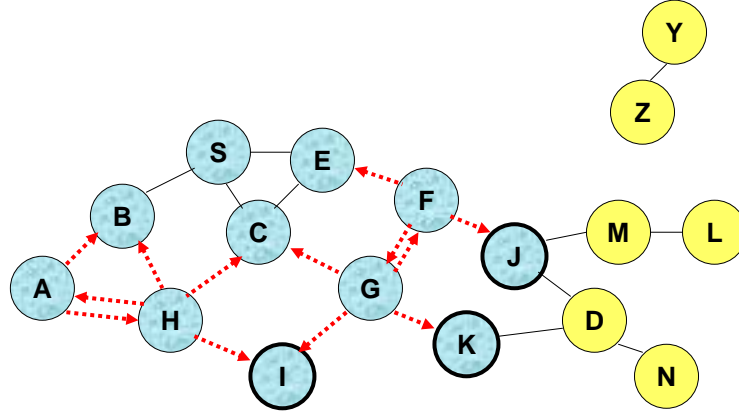


# Flooding for Data Delivery



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

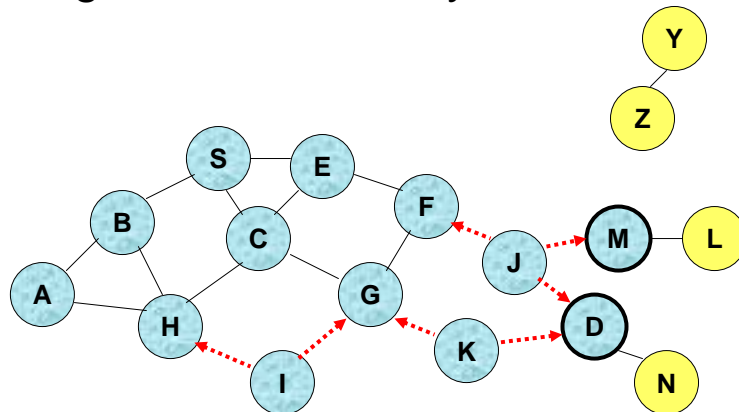
## Flooding for Data Delivery



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

15

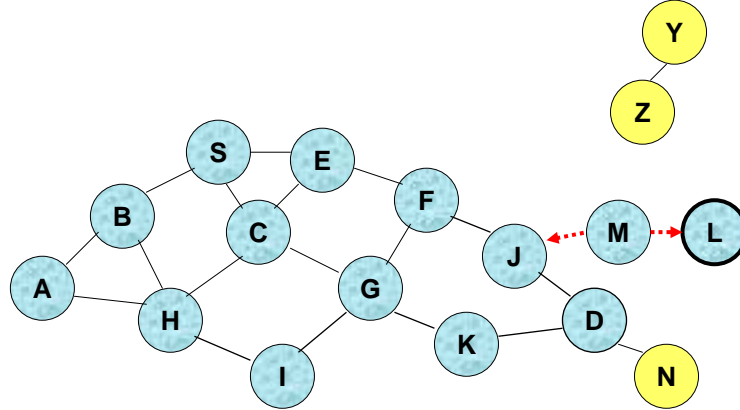
## Flooding for Data Delivery



- 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

16

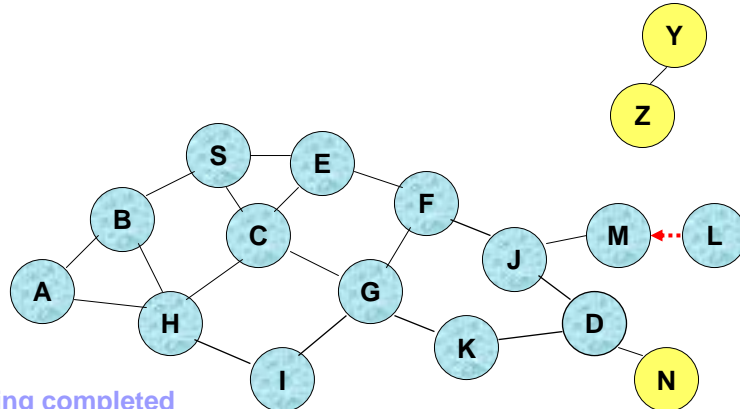
## Flooding for Data Delivery



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

17

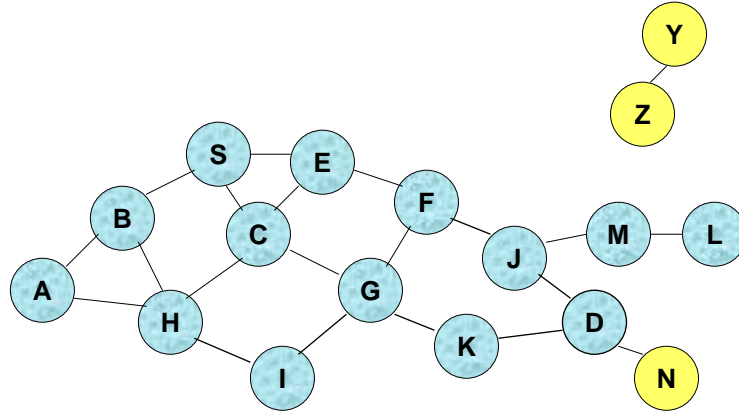
## Flooding for Data Delivery



- **Flooding completed**
- Nodes **unreachable** from S do not receive packet P (e.g., node Z)
- Nodes for which all paths from S go through the destination D also do not receive packet P (example: node N)

18

## 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)

19

## 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

20

## 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

21

## 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

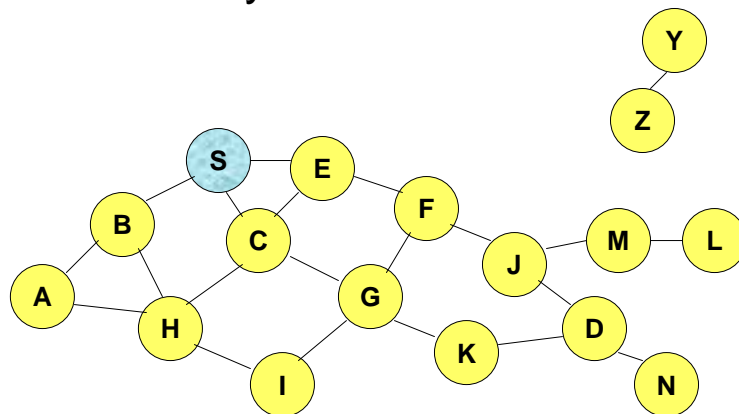
22

## Dynamic Source Routing (DSR) [Johnson96]

- 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

23

## Route Discovery in DSR

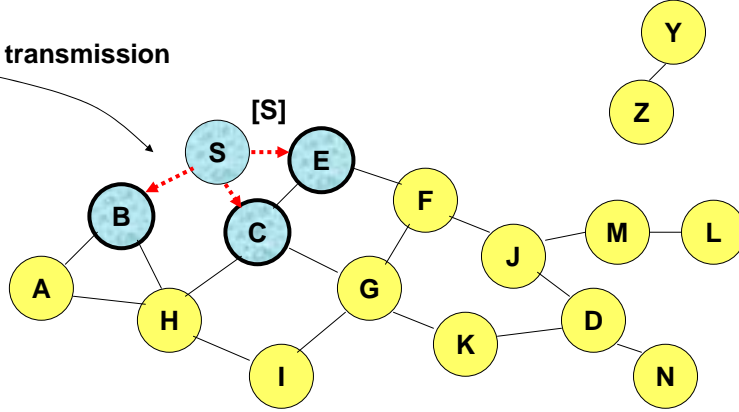


Represents a node that has received RREQ for D from S

24

# Route Discovery in DSR

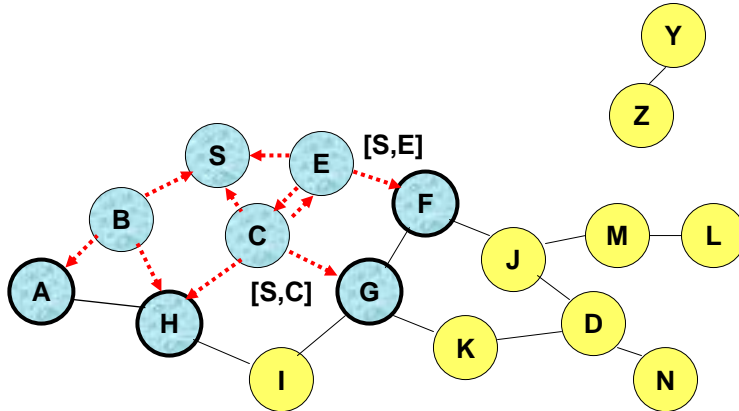
Broadcast transmission



.....> Represents transmission of RREQ

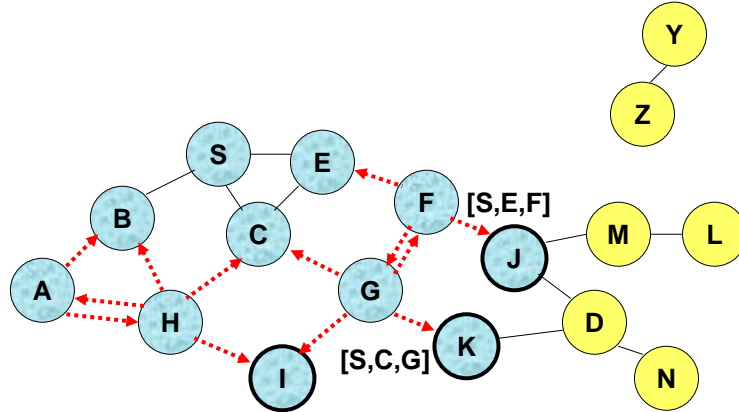
[X,Y] Represents list of identifiers appended to RREQ

# Route Discovery in DSR



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

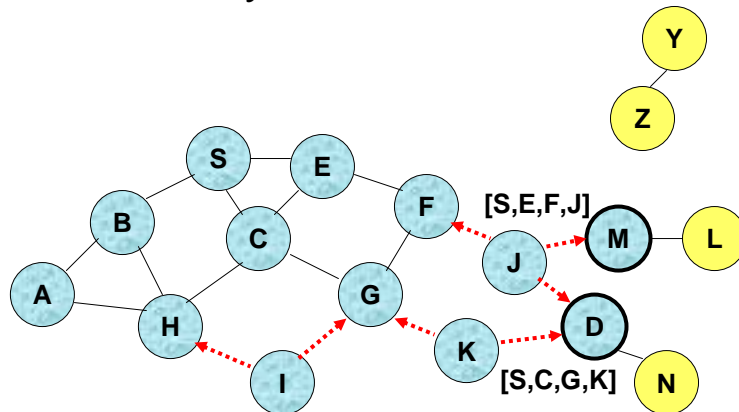
## Route Discovery in DSR



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

27

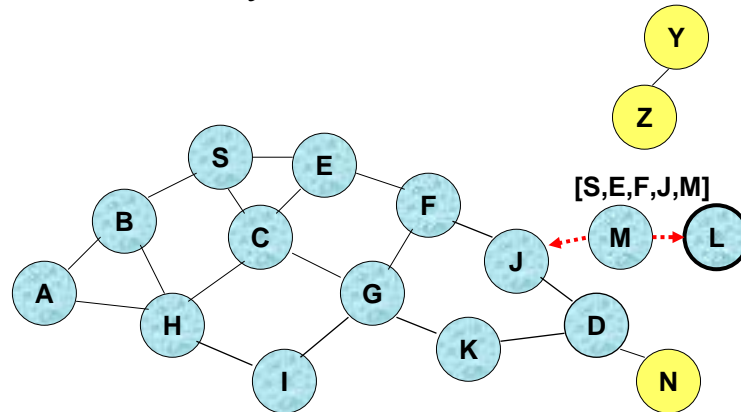
## Route Discovery in DSR



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

28

## Route Discovery in DSR



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

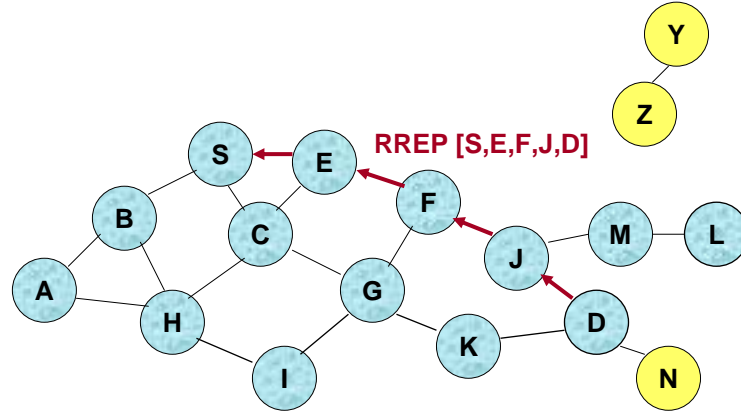
29

## 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

30

## Route Reply in DSR



← Represents RREP control message

31

## 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, RREP 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)

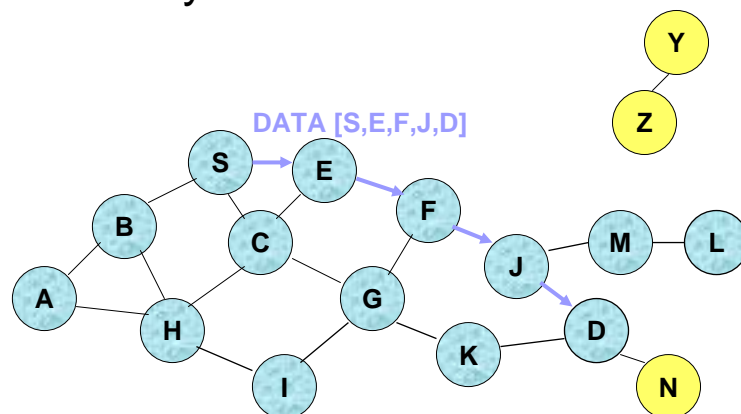
32

## 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

33

## Data Delivery in DSR



**Packet header size grows with route length**

34

## 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

35

## 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

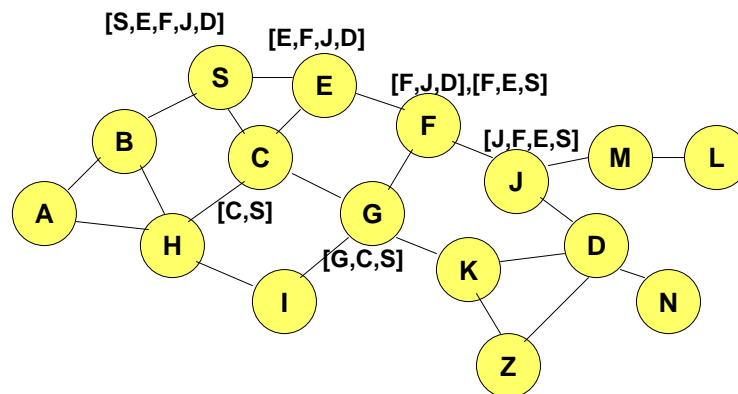
36

## 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

37

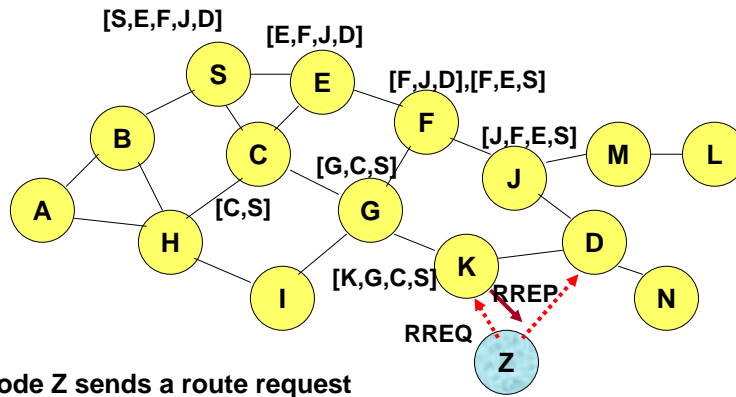
## Use of Route Caching



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

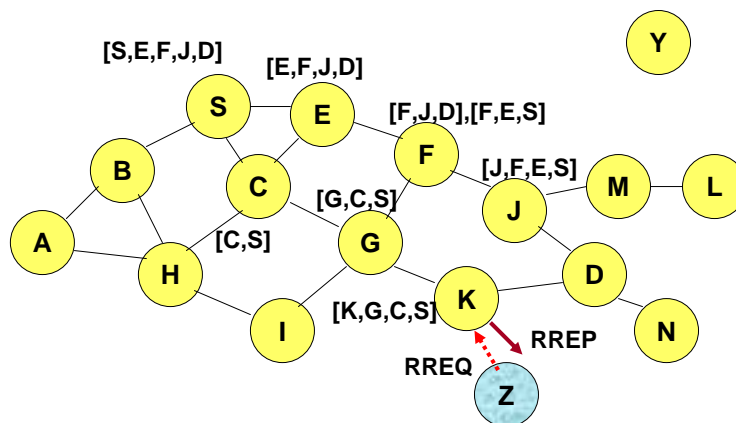
38

## Use of Route Caching: Can Speed up Route Discovery



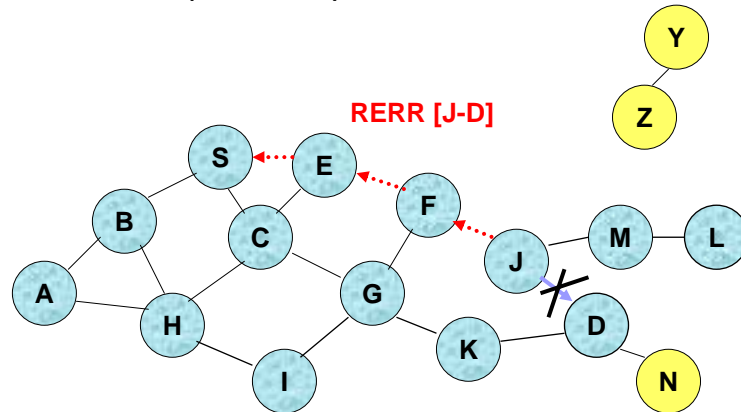
When node Z sends a route request for node C, node K sends back a route reply [Z, K, G, C] to node Z using a locally cached route

## Use of Route Caching: Can 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 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

41

## 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
- An illustration of the adverse impact on TCP will be discussed later in the tutorial [[Holland99](#)]

42

## 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

43

## 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

44

## 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.
- For some proposals for cache invalidation, see [Hu00Mobicom]
  - Static timeouts
  - Adaptive timeouts based on link stability

45

## Flooding of Control Packets

- How to reduce the scope of the route request flood ?
  - LAR [Ko98Mobicom]
  - Query localization [Castaneda99Mobicom]
- How to reduce redundant broadcasts ?
  - The Broadcast Storm Problem [Ni99Mobicom]

46

## Routing Protocols

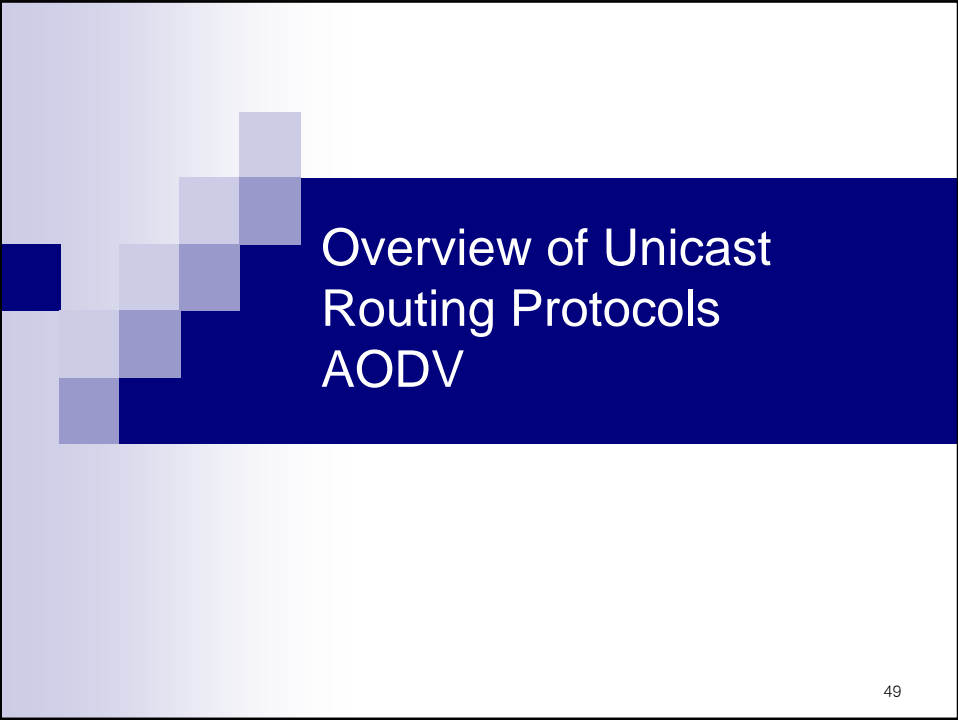
- Reactive approaches
  - DSR (dynamic source routing), AODV (ad-hoc on-demand distance vector)
- Proactive approaches
  - DSDV (destination sequenced distance vector), LSR (link state routing), OLSR (optimized link state routing)
- Hybrid approaches
  - ZRP (zone routing protocol), CEDAR (Core extraction distributed ad-hoc routing)

47

## Flooding of Control Packets

- How to reduce the scope of the route request flood ?
  - LAR [[Ko98Mobicom](#)]
  - Query localization [[Castaneda99Mobicom](#)]
- How to reduce redundant broadcasts ?
  - The Broadcast Storm Problem [[Ni99Mobicom](#)]

48



## Overview of Unicast Routing Protocols AODV

49

CSC7602 – S.J. Park

### Ad Hoc On-Demand Distance Vector Routing (AODV) [Perkins99Wmcsa]

- DSR includes source routes in packet headers
- Resulting large headers can sometimes degrade performance
  - particularly when data contents of a packet are small
- AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes
- AODV retains the desirable feature of DSR that routes are maintained only between nodes which need to communicate

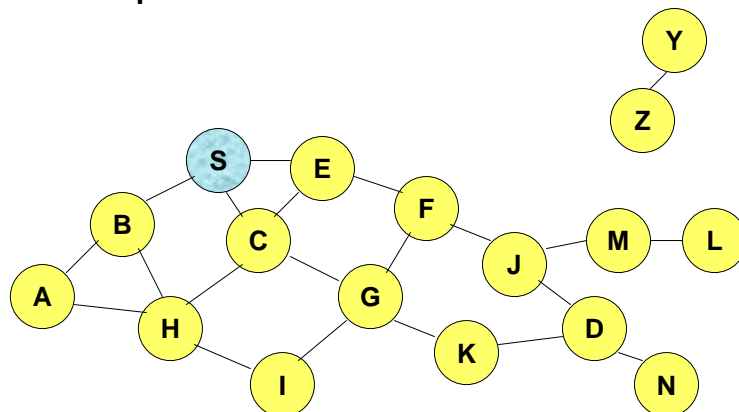
50

## AODV

- Route Requests (RREQ) are forwarded in a manner similar to DSR
- 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

51

## Route Requests in AODV

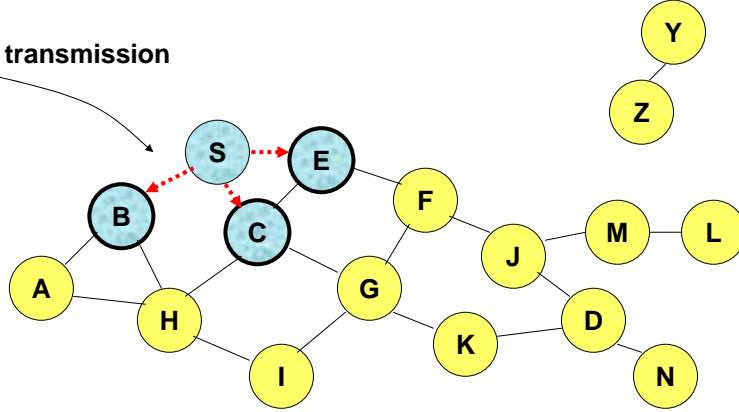


Represents a node that has received RREQ for D from S

52

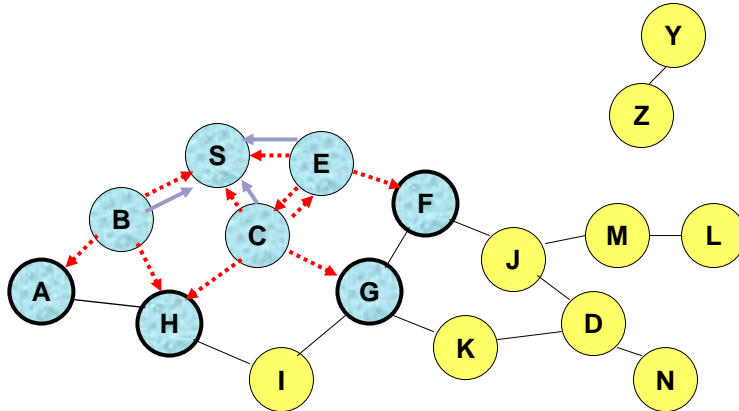
# Route Requests in AODV

Broadcast transmission



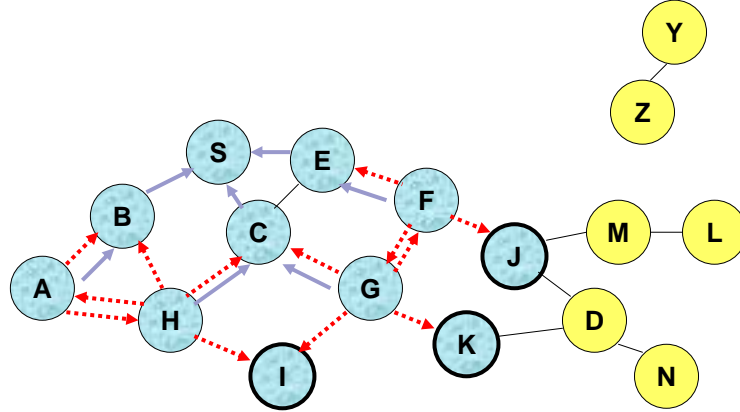
..... Represents transmission of RREQ

# Route Requests in AODV



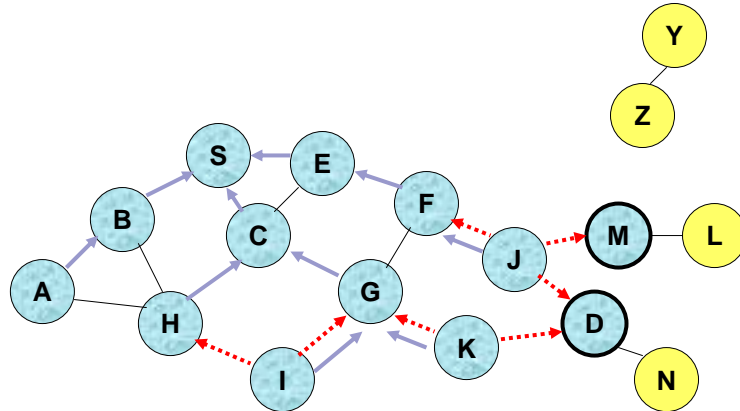
← Represents links on Reverse Path

## Reverse Path Setup in AODV

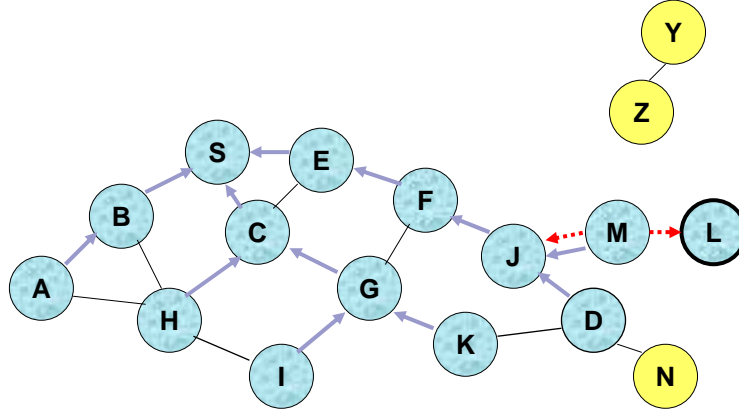


- Node C receives RREQ from G and H, but does not forward it again, because node C has **already forwarded RREQ once**

## Reverse Path Setup in AODV

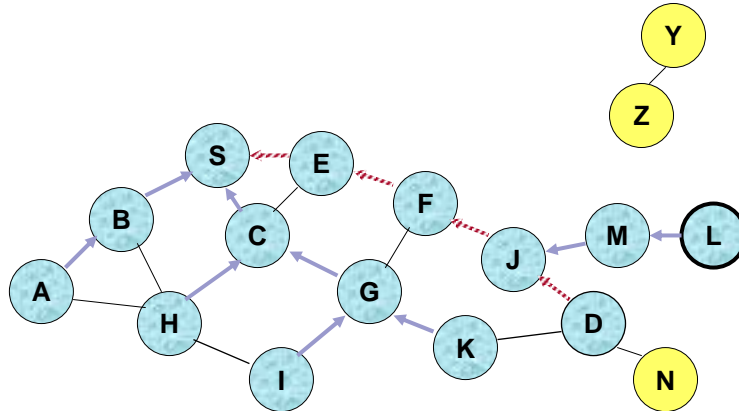


## Reverse Path Setup in AODV



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

## Route Reply in AODV



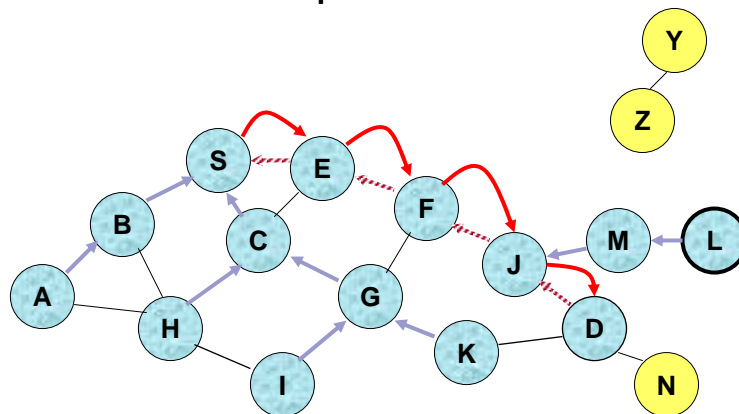
----- Represents links on path taken by RREP

## 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 as high as DSR
  - 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

59

## Forward Path Setup in AODV

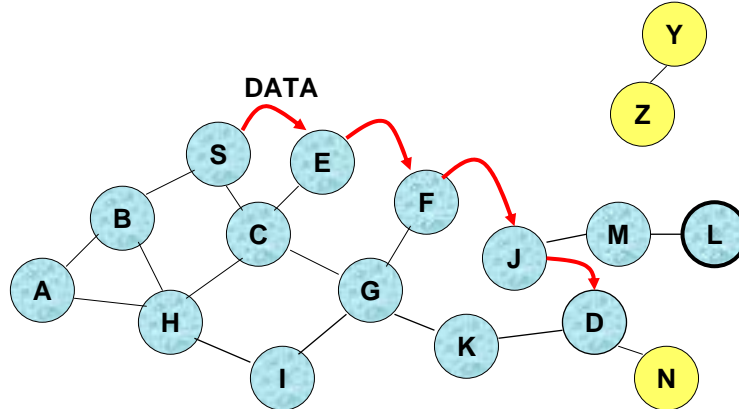


Forward links are setup when RREP travels along the reverse path

 Represents a link on the forward path

60

## Data Delivery in AODV



Routing table entries used to forward data packet.

Route is *not* included in packet header.

61

## Optimization: Expanding Ring Search

- Route Requests are initially sent with small Time-to-Live (TTL) field, to limit their propagation
  - DSR also includes a similar optimization
- If no Route Reply is received, then larger TTL tried

62

## Summary: AODV

- Routes need not be 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
  - DSR may maintain several routes for a single destination
- Unused routes expire even if topology does not change

63

## So far ...

- All protocols discussed so far perform some form of flooding
- Now we will consider protocols which try to reduce/avoid such behavior

64

## Destination-Sequenced Distance-Vector (DSDV) [Perkins94Sigcomm]

- 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

65

## 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

66

## 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  $S(Y)$

67

## 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

68

## Multicasting in MANET

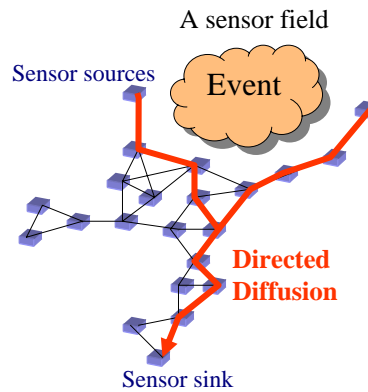
- Need to take topology change into account when designing a multicast protocol
- Several new protocols have been proposed for multicasting in MANET

## Directed Diffusion for Wireless Sensor Networking

C. Intanagonwiwat, R. Govindan, D. Estrin, et. al.  
IEEE/ACM Transactions on Networking, Feb 2003

## Sensor Networking

- A region requires event-monitoring (harmful gas, vehicle motion, seismic vibration, temperature, etc.)
- Deploy sensors forming a distributed network
- On event, sensed and/or processed information delivered to the inquiring destination
- Simple attribute-based naming as fundamental building block.
- Requests for information (*interests*) and relevant data (*reports*) are described as sets of value-attribute pairs.



71

## Elements of Directed Diffusion

- **Contribution**
  - *1<sup>st</sup> approach for data centric routing for sensor networks*
- **Naming**
  - *Data* is named using attribute-value pairs.
- **Interests**
  - A node requests data by sending *interests for named data* .
- **Gradients**
  - *Gradients* are set up within the network designed toward the sink to "draw" events, i.e. data matching interest.
- **Reinforcement**
  - Sink *reinforces* particular neighbors to draw higher quality ( higher data rate) events.

72

## Naming

- Content-based naming.
  - Tasks are named by a list of attribute - value pairs.
  - Task description specifies an *interest* for data matching the attributes.
  - Animal tracking:

### Request

#### Interest ( Task ) Description

Type = four-legged animal  
 Interval = 20 ms  
 Duration = 1 minute  
 Location = [-100, -100; 200, 400]

### Reply

#### Node data

Type =four-legged animal  
 Instance = elephant  
 Location = [125, 220]  
 Confidence = 0.85  
 Time = 02:10:35

73

## Gradient Set Up

- Inquirer (sink) broadcasts exploratory interest, *i1*
  - Intended to discover routes between source and sink
- Neighbors update interest-cache and forwards *i1*
- Gradient for *i1* set up to upstream neighbor
  - No source routes
  - Gradient – a weighted reverse link
  - Low gradient → Few packets per unit time needed

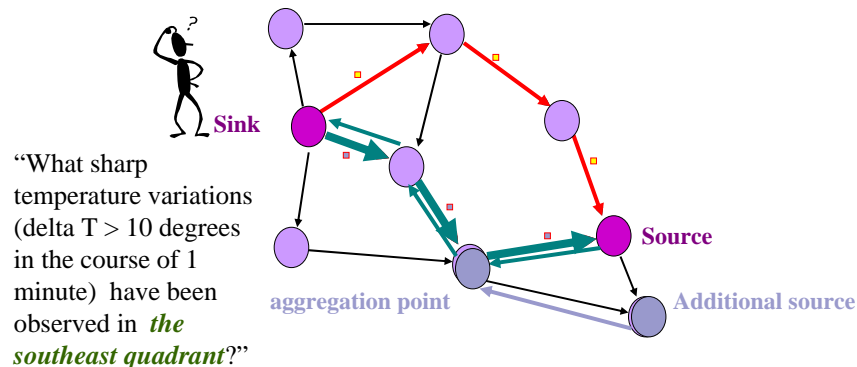
74

## Basic Algorithm

- Sink floods interest. (interest may be periodically repeated).
- Every node caches interest while valid, and creates local gradient towards neighboring nodes from which it heard interest.
- Sources with relevant data starts sending it according to local gradients.
- When sink starts receiving data, it reinforces one or some of the paths, pruning the rest.
- Negative reinforcements can be used for adjusting to changing conditions.

75

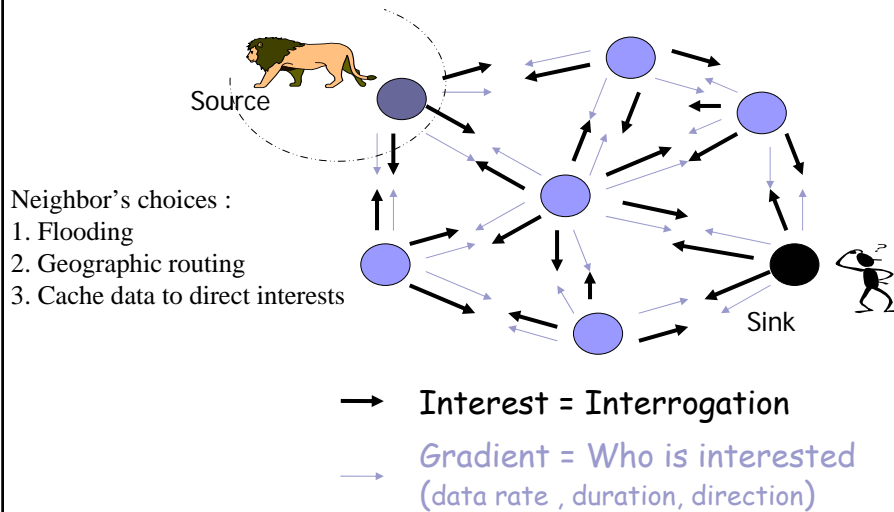
## Main Idea



- Robust, efficient data distribution in sensor networks
  - **name data** (not nodes), use physicality
  - **diffuse requests and responses** across network
  - optimize path with **gradient-based feedback**
  - additional data causes **in-network aggregation**

76

## Example



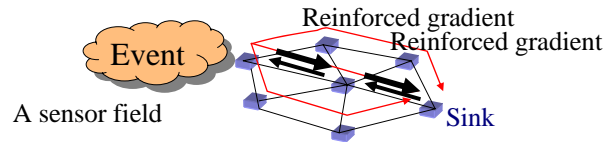
77

## Data Propagation

- Sensor node computes the highest requested event rate among all its outgoing gradients.
- When a node receives data:
  - Find a matching interest entry in its cache
    - Examine the gradient list, send out data by rate.
  - Cache keeps track of recent seen data items (loop prevention).
  - Data message is unicast individually to the relevant neighbors.

78

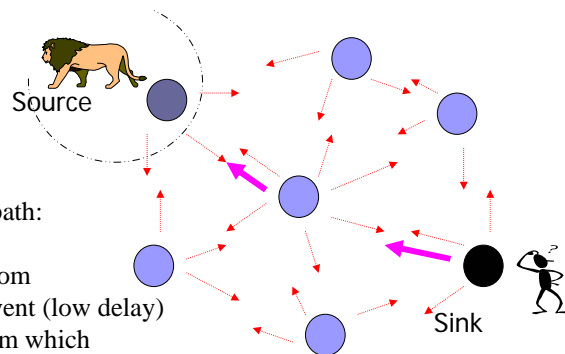
## Reinforcement



- From exploratory gradients, reinforce optimal path for high-rate data download → **Unicast**
  - By requesting higher-rate-*if* on the optimal path
  - Exploratory gradients still exist – useful for faults

79

## Reinforcing the Best Path



The neighbor reinforces a path:

1. At least one neighbor
2. Choose the one from whom it first received the latest event (low delay)
3. Choose all neighbors from which new events were recently received

→ Low rate event

→ Reinforcement = Increased interest

80

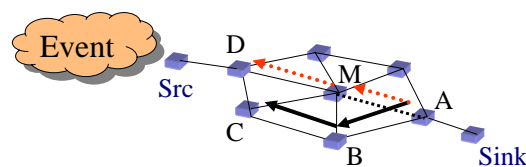
## Local Behavior Choices

- For data transmission
  - *Multi-path delivery with selective quality along different paths*
  - Probabilistic forwarding
  - Single-path delivery, etc.
- For reinforcement
  - *Reinforce paths based on observed delays*
  - Losses, variances etc.

81

## Path Failure / Recovery

- Link failure detected by reduced rate, data loss
  - *Choose next best link (i.e., compare links based on infrequent exploratory downloads)*
- Negatively reinforce lossy link
  - *Either send *i1* with base (exploratory) data rate*
  - *Or, allow neighbor's cache to expire over time*

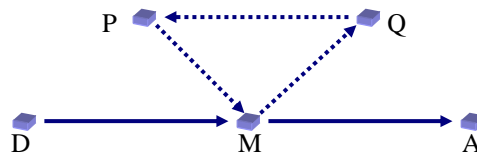


Link A-M lossy

A reinforces B  
 B reinforces C ...  
 D need not  
 A (-) reinforces M  
 M (-) reinforces D

82

## Loop Elimination



- M gets same data from both D and P, but P **always** delivers late due to looping
  - M negatively-reinforces (nr) P, P nr Q, Q nr M
  - Loop {M → Q → P} eliminated
- Conservative nr useful for fault resilience

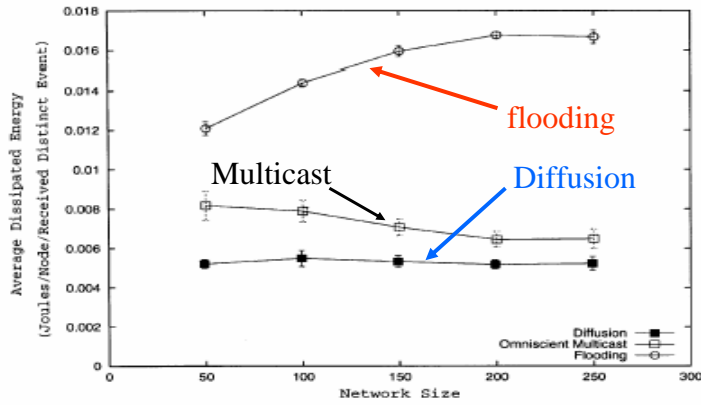
83

## Simulation Setup & Metrics

- ns2, 50 nodes in 160x160 sqm., range 40m
  - Node density maintained, 802.11 MAC
- Random 5 sources in 70x70, random 5 sinks
- Average Dissipated Energy
  - Per node energy dissipation / # events seen by sinks
- Average Delay
  - Latency of event transmission to reception at sink
- Distinct event delivery ratio
  - Ratio of # events sent to # events received by sink

84

## Average Dissipated Energy

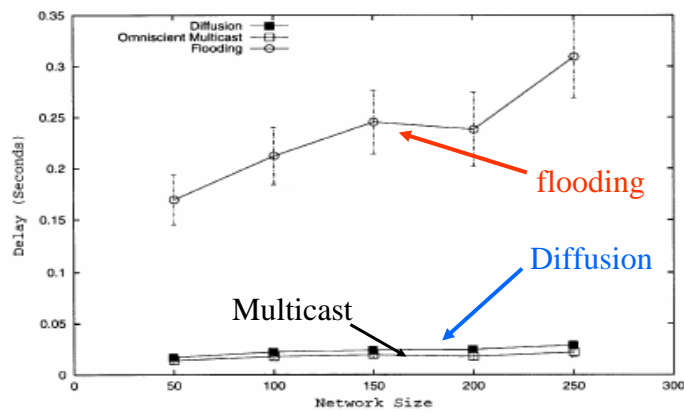


In-network aggregation reduces DD redundancy

- Flooding poor because of multiple paths from source to sink

85

## Delay

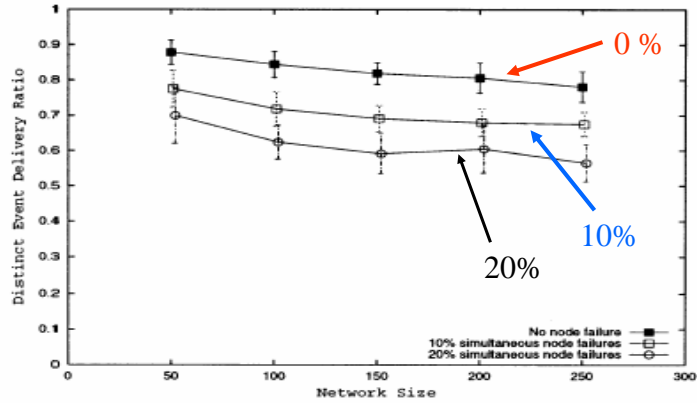


DD finds least delay paths, as OM – encouraging

- Flooding incurs latency due to high MAC contention, collision

86

## Event Delivery Ratio under node failures



Delivery ratio degrades with higher % node failures

- Graceful degradation indicates efficient negative reinforcement

87

## Conclusion

- Directed diffusion, a paradigm proposed for event monitoring sensor networks
- Energy efficiency achievable
- Diffusion mechanism resilient to fault tolerance
  - Conservative negative reinforcements proves useful
- A careful MAC protocol, designed for such specifics, can yield further performance gains

88

## Contribution

- Application-awareness – a beneficial tradeoff
  - Data aggregation can improve energy efficiency
  - Better bandwidth utilization
- Network addressing is data centric
  - Probably correct approach for sensor type applications
- Notion of gradient (exploratory and reinforced)
  - Flexible architecture – enables configuration based on application requirements, tradeoffs
- Implementation on Berkeley nodes
  - Network API, Filter API

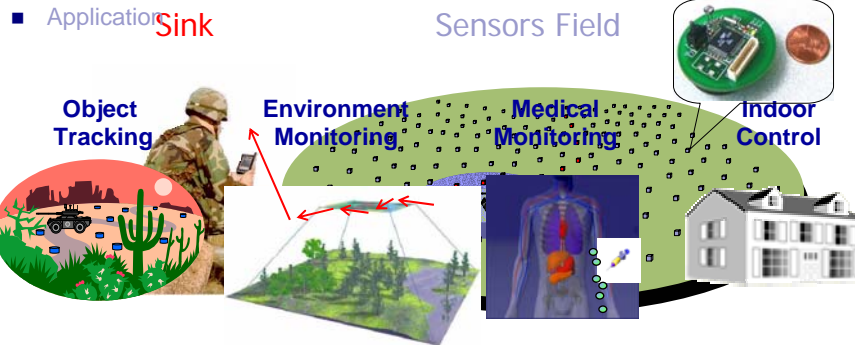
89

## Energy Data Aggregations over Sensor Networks

90

# Wireless Sensor Network

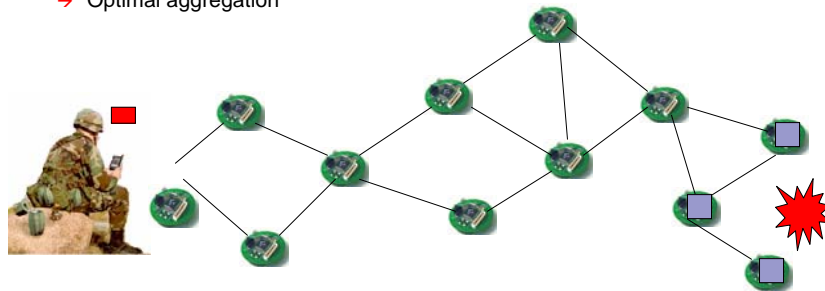
- Elements
  - Sink : sends queries to collect data from sensors
  - Sensor : monitors phenomenon and reports to sink



91

# Energy Efficient Protocols over WSN

- Topology control
  - Transmission power
  - Adaptive power control
- Downstream data delivery
  - loss sensitive query
  - Reliable data delivery
- Upstream data aggregation
  - Redundancy among data
  - Optimal aggregation



92



# EnCAS: An Energy Efficient Correlated Data Aggregation for Wireless Sensor Networks

Seung-Jong Park\*  
Computer Science and CCT  
Louisiana State University

Raghupathy Sivakumar  
Electrical and Computer Engineering  
Georgia Institute of Technology

93

CSC7602 – S.J. Park

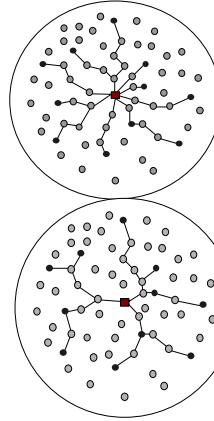
## Problem

- Target Problem
  - Energy-efficient upstream data aggregation from sensors to a sink
- Challenge
  - Correlated data -> In-network processing
- Types of correlations
  - Spatial correlation
  - Semantic correlation
- Degree of correlation
  - Correlation factor  $\rho$  ( $0 \leq \rho \leq 1$ )
  - Size of aggregated data  $m_1$  and  $m_2 = m_1 + (1 - \rho) m_2$

94

## Optimal Solutions

- For different correlation factor  $\rho$ 
  - Zero correlation:  $\rho = 0$ 
    - Shortest Path Tree
  - Perfect correlation:  $\rho = 1$ 
    - Steiner Minimum Tree
  - General correlation:  $0 < \rho < 1$ 
    - Open Problem ?



95

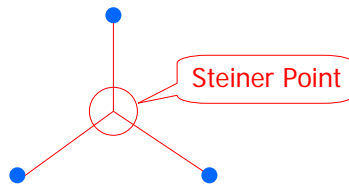
## Problem Scope & Goal

- Problem scope
  - Perfect correlation:  $\rho = 1$
  - Single sink
  - Approximation of the optimal solution (the Steiner minimum tree)
- Design Goals of EnCAS
  - Scalability to large number of sensors
  - Decentralized operation
  - Loose synchronization

96

## EnCAS Key Ideas

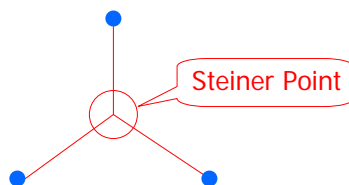
- General heuristics for Steiner Minimum Tree
  - Shortest path tree heuristics
  - Minimum spanning tree heuristics
  - Find “Steiner Points” to aggregate data from different sources
- Heuristic in EnCAS
  - Use the minimum dominating set as Steiner Points irrespective of locations of sources
  - Aggregate data as much and early as possible at minimum dominating set
  - Utilize the “Core” structure which was constructed by GARUDA



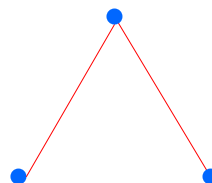
97

## Steiner Minimum Tree vs. Minimum Spanning Tree

### ■ SMT



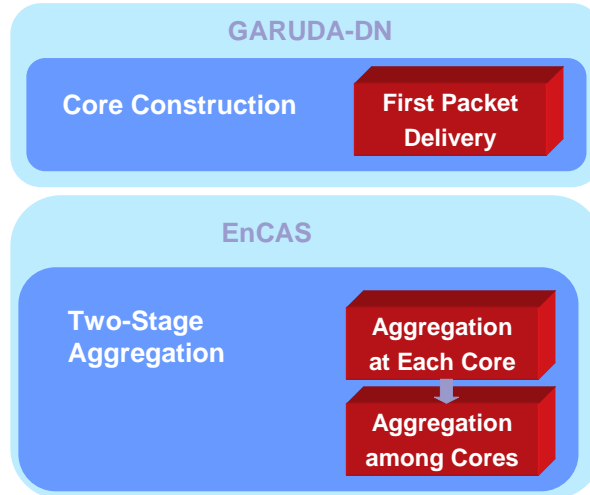
### ■ MST



- Minimum spanning tree is a tree interconnecting nodes using only edges with shortest length

98

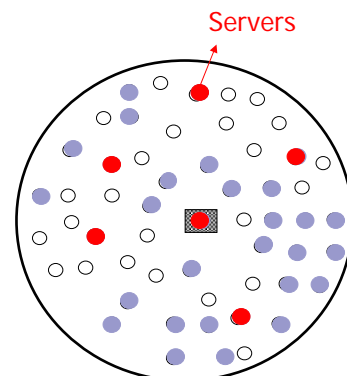
## EnCAS Overview



99

## Recovery Server Designation

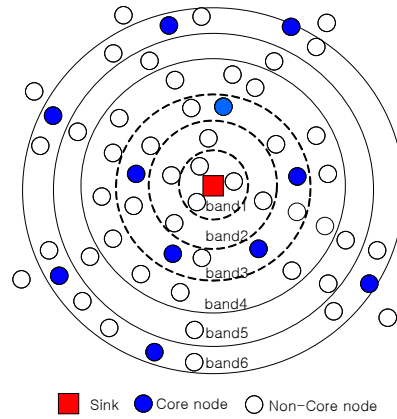
- Problem: Minimize the number of recovery servers
  - Minimize the number of blue nodes selected to cover all white nodes
- Ideal solution: **Minimum Set Cover (MSC)**
  - ↯ NP-hard problem
  - ↯ Infeasible because of per-packet basis
- GARUDA: Distributed **Minimum Dominating Set (MDS)**
  - Approximation of MSC
  - Independent of loss pattern
  - Per message basis



100

## Core Structure

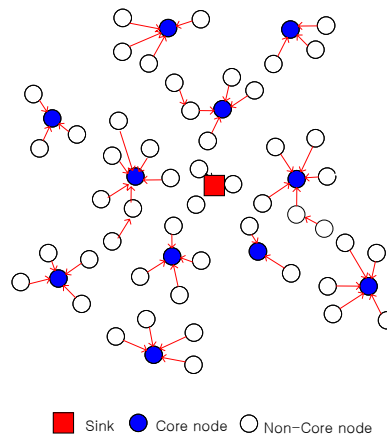
- Distributed MDS
  - Virtual bands constructed during the first packet flood
  - Nodes at every 3<sup>rd</sup> band choose themselves as core nodes if there is no neighboring cores
- Core Merits
  - ↳ Approximation of the ideal solution, MSC
  - ↳ Decentralized construction during the 1<sup>st</sup> packet delivery
  - ↳ Low maintenance overhead



101

## EnCAS: Aggregation at Each Core

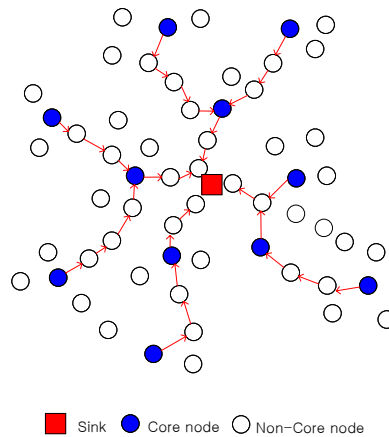
- Non-core nodes send original data to its core node



102

## EnCAS: Aggregation among Cores

- Core nodes send aggregated data toward a sink
  - Each node knows its parent in shortest path tree
  - Loose synchronization among core nodes



103

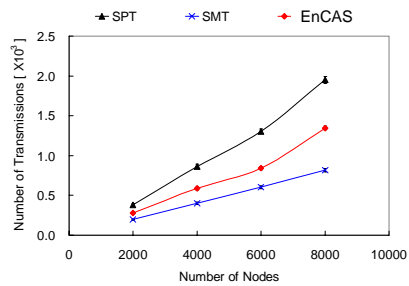
## Performance Evaluation

- Simulation environment
  - Discrete event simulation
- Metric
  - number of transmission
- Comparison EnCAS with
  - Distributed SPT (Shortest Path Tree)
  - Centralized & Approximated SMT (Steiner Minimum Tree)

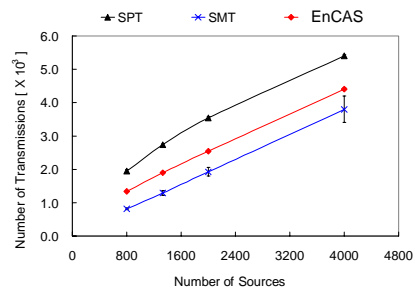
104

## Simulation Results

- Varying number of nodes, 10% sources



- Varying number of sources, 8000 nodes



105

## Conclusion

- Define the problem and ideal solutions for different correlation factors
- Propose EnCAS
  - approximating the optimal solution, the Steiner minimum tree
  - Utilizing the core structure of GARUDA
  - Operating in decentralized and simple fashion
- Demonstrate performance of EnCAS
  - Better than decentralized version of shortest path tree
  - Closely approaching the centralized and heuristics version of the Steiner minimum tree

106