Network Layer: Mobile, Wireless Routing

Y. Richard Yang

11/29/2012
Outline

- Admin. and recap
- Network layer
  - Intro
  - Location/service discovery
  - Routing
Admin.

- Please use Sign Up on classesv2 for project meetings

- Remaining topics to cover
  - Non-traditional routing
  - Localization
  - Controlled mobility
  - Applications such as signalguru, foursquare, ...
Recap: Basic Network Layer

Each node is a network attachment point (e.g., router, base station), to which hosts/user device attaches

Key problems
- Location/service discovery
  - Find attached point for id-based naming
- Routing
  - Find path from src to dst attach point

User device identified by addressing scheme
- locator: identifies attachment point
- identifier: independent of location
Recap: Service/Location Discovery

- Basic DNS
- Problem: fixed/static DNS record
  - Potential solution: Linda tuple space like store
- Problem: Needs server infrastructure
  - Potential solution: mDNS/DNS-SD
- Problem: Mapping from ID to locator can change as a device moves
  - Potential solution primitives: device update vs sys paging
  - Example: location area (LA) as a hybrid approach in cellnet
    - Issue of LA based approach: Users roaming in LA borders may generate a lot of updates
Other Potential Hybrid Design

- **Timer based**
  - A MS sends an update after some given time $T$

- **Movement based**
  - A MS sends an update after it has visited $N$ different cells

- **Distance based**
  - A MS sends an update after it has moved away for $D$ distance (need ability to measure distance)

- **Profile based**
  - A MS predicts its mobility model and updates the network when necessary
Timer-based Location Management

- A MS sends an update after some given timer $T$
- The network pages the MS upon a call request at all cells which the MS can potentially arrive during $T$
  - cells reachable from last update cell, e.g., within distance $v_{\text{max}} \times T$, where $v_{\text{max}}$ is the maximum speed

Question: how to determine $T$?
Timer-based Location Management

- Assume time between call arrivals is $T_{\text{call}}$
- Cell radius is $d_{\text{cell}}$
- Total bandwidth cost:

$$
\frac{T_{\text{call}}}{T} b_{\text{update}} \left( \frac{v_{\text{max}} T}{d_{\text{cell}}} \right)^2 b_{\text{paging}}
$$

Take derivative and set it to 0 to derive the optimal value:

$$
T = 3 \sqrt{\frac{1}{2} \frac{d_{\text{cell}}^2}{v_{\text{max}}^2} T_{\text{call}} \frac{b_{\text{update}}}{b_{\text{paging}}}}
$$
Summary: Location Discovery

- Two primitives of location discovery in cellular networks
  - update (a proactive approach)
  - paging (a reactive approach)

- Hybrid update/paging tradeoff
  - location area (LA) based
  - timer based
  - movement based
  - distance based
  - profile based

\[
T = \frac{T_{\text{call}} d_{\text{cell}}^2}{2 v_{\text{max}}^2} \frac{b_{\text{update}}}{b_{\text{paging}}} \]

\[
\frac{b_{\text{call}}}{T} b_{\text{update}} = \left( \frac{v_{\text{max}} T}{d_{\text{cell}}} \right)^2 b_{\text{paging}}
\]
Remaining Issue: Handoff

- A MS may be in a call during mobility
  - the signal to/from the current serving base station (e.g., Node B in 3G) may gradually degrade as the MS moves away, and the signal to/from the next base station may become better

- Issue:
  - signal strength of next base station is hard to know, if the MS is not actively communicating w/ the next base station
WCDMA Soft Handoff

- An MS communicates with multiple Node Bs
  - Downlink: multiple base stations send to the MS and the MS combines the received signals
  - Uplink: multiple base stations receive data from the MS and forward to a RNC to combine
    - The combing RNC is called the serving RNC
Serving RNC and Drift RNC

RNC1: Serving RNC
RNC2: Drift RNC

RNC: Serving RNC handoff
UMTS Serving RNC Handoff

### Preparation phase at the core network
1. Relocation Required
2. Forward Relocation Request
3. Relocation Request & Ack
4. Forward Relocation Response
5. Relocation Command
6. Relocation Commit
   - Downstream Packet forwarding
7. Relocation Detect
8. RNTI Reallocation & Complete
9. Update PDP Context Request & Response
10. Relocation Complete
11. Forward Relocation Complete & Ack
12. Iu Release Command & Complete
13. RA Update Procedure

### Resource release of the old connection
Outline

- Admin.
- Location management
  - cellular networks
  - IP networks
Mobile IP: Architecture

- The current architecture to handle mobility in Internet is Mobile IP
  - Assume the current Internet model
    - Mobile Node (MN): the node under consideration with a given IP address
    - Correspondent Node (CN): communication partner
    - Communication from CN to MN should not break as MN moves to a new attachment point
Mobile IP: Basic Idea

- Introduce Home Agent (HA) at home network to forward data to MN

- Introduce Foreign Agent (FA) in the foreign network
  - Use foreign agent’s address as Care-of Address (COA)

- MN always updates HA its current COA

- HA tunnels data to MN using COA
Illustration

HA

router

home network

(physical home network for the MN)

Internet

FA

foreign network

router

mobile node

MN

router

CN

end-system

(router)

(router)

(router)

(router)
Data Transfer from the Mobile Node

1. Sender sends to the IP address of the receiver as usual, FA works as default router.
Data Transfer to the Mobile Node

1. Sender sends to the IP address of MN, HA intercepts packet
2. HA tunnels packet to COA, here FA, by encapsulation
3. FA forwards the packet to the MN
Problem of Mobile IP

- Triangular Routing
  - CN sends all packets via HA to MN
  - higher latency and network load

- “Solution”
  - CN learns the current location of MN
  - direct tunneling to this location
  - HA or MN informs a CN about the location of MN

- Problem of the solution
  - big security problems!
Recap: Key Problems

✓ Location/service management

➢ Routing in wireless/mobile networks
Routing Overview

- The objective of routing is to find a good path for each source destination pair.

Issues

- How to define a path to be good?
- How do we compute the path?
Link Metric

- A typical measure for a good path is that it is the shortest path according to some metric.
- One possibility is to assign each link a metric of 1 (hop-count based routing)
Performance of Shortest Hop Count

x axis: throughput

y axis: fraction of pairs with less throughput

Run R1: 1 mW, 134-byte packets
Problems of Shortest Hop Count

- maximizes the distance traveled by each hop
  - low signal strength -> high loss ratio
  - uses a higher TxPower -> interference
Metric: ETX

- ETX of a link: The predicted number of data transmissions required to successfully transmit a packet over a link
- ETX of a path: the sum of the ETX values of the links over that path

Examples:
- ETX of a 3-hop route with perfect links: 3
- ETX of a 1-hop route with 50% loss: 2


http://meraki.com/about/
Acquiring ETX

- Measured by broadcasting dedicated link probe packets with an average period $\tau$ (jittered by $\pm 0.1\tau$)
- Delivery ratio:

$$r(t) = \frac{\text{count}(t-w,t)}{w/\tau}$$

- $\text{count}(t-w,t)$ is the # of probes received during window $w$
- $w/\tau$ is the # of probes that should have been received
ETX: Example
ETX Performance

Run R1: 1 mW, 134-byte packets

Cumulative fraction of node pairs

Max 4-hop throughput  3-hop  2-hop

Best static route  DSDV ETX  DSDV Hop-count

Packets per second delivered
ETX: Advantages and Problem

Advantage
- Tends to minimize spectrum use, which can maximize overall system capacity (reduce power too): node spends less time retransmitting data

Problem
- ETX does not handle multirate networks
Extending ETX for Multirate

- In a single rate environment, each transmission (of same pkt size) takes the same unit of time
- For multirate, computes:

\[
ETT = \left( \frac{S}{B} \right) \times ETX
\]

- packet size = S, Link bandwidth = B

- Interpretation: pick a path with the lowest total network occupation time

Outline

- Admin and recap
- Routing
  - Overview
  - Routing metric
  - Computing shortest path routing
Design Dimensions

- Proactive vs reactive (on-demand)
- Complete information (optimal) vs partial information
Outline

- Admin and recap
- Routing
  - Overview
  - Routing metric
  - Computing shortest path routing
    - proactive/complete-info: link state
Link-State Routing Algorithms

- **Separation** of topology distribution from route computation

- Used in OSPF, the dominant intradomain routing protocol used in the Internet

- Net topology, link costs are distributed to all nodes

- Link state distribution accomplished via “link state broadcast”

- Each node (locally) computes its paths to all destinations
Link State Broadcast

- Represents a node that has received update
- Represents link
Link State Broadcast
To avoid forwarding the same update multiple times, each update has a sequence number. If an arrived update does not have a higher seq, discard!
- The packet received by E from C is discarded
- The packet received by C from E is discarded as well
- Node H receives packet from two neighbors, and will discard one of them
Summary of Link State Routing

- Separation of topology distribution from route computation
- Whenever a link metric changes, the node broadcasts new value
- Q: What is the scope of updates when a link changes status?
- Q: Does link state routing work well in a network with dynamic link status?
Outline

- Admin and recap

- Routing
  - Overview
  - Routing metric
  - Computing shortest path routing
    - proactive/complete-info: link state
    - proactive/partial-info: link reversal
Link Reversal: Basic Idea

- maintain a mesh
- (hopefully) local adaptation to dynamic links
Maintain a directed acyclic graph (DAG) for each destination, with the destination being the only sink.

Links are bi-directional, but the algorithm imposes logical directions on them.

This DAG is for destination node D.
Link Reversal Algorithm: Illustration of Idea

Any node, other than the destination, that has no outgoing links reverses some incoming links.

Node G has no outgoing links
Link Reversal Algorithm: Illustration

Now nodes E and F have no outgoing links, the process continues.
Link Reversal Algorithm: Illustration

Now nodes B and G have no outgoing links

Represents a link that was reversed recently
Now nodes A and F have no outgoing links

Represented a link that was reversed recently
Link Reversal Algorithm: Illustration

Now all nodes (other than destination D) have an outgoing link
Link Reversal Algorithm: Illustration

DAG has been restored with only the destination as a sink
Summary: Link Reversal

- **Motivations**
  - maintain a mesh
  - (hopefully) local adaptation

- **Remaining questions:**
  - how to implement it?
  - will reversal stop?

- Next we will look into the questions using **partial** reversal (not full reversal, as the preceding example)
A node $i$ contains a triple $(\alpha_i, \beta_i, i)$
- $\alpha_i$: an integer (the major integer)
- $\beta_i$: another integer (the minor integer)
- $i$: node index (to impose a total order)

The triple of a node is called the **height** of the node.

Suppose there is a link from node $i$ to node $j$, the direction is determined by their heights
- $i \rightarrow j$: if $(\alpha_i, \beta_i, i) > (\alpha_j, \beta_j, j)$

For destination $D$, the height is $(0, 0, D)$
Illustration of Heights
Partial Reversal Algorithm

- If the height of node $i$ is lower than all of its neighbors, i.e., $(\alpha_i, \beta_i, i) < (\alpha_j, \beta_j, j)$ for all $j$ in $N_i$, increases $\alpha_i$ to

$$\alpha_i^{new} = \min \{ \alpha_j \mid j \in N_i \} + 1$$

where $N_i$ is the set of neighbors of $i$.

- If there exists a neighbor $j$ with the same $\alpha$ value after $i$ has increased its $\alpha$, set $\beta_i$ to

$$\beta_i^{new} = \min \{ \beta_j \mid j \in N_i, \alpha_i^{new} = \alpha_j \} - 1$$

otherwise $\beta_i$ not changed
Illustration \( \beta_i^{new} = \min\{\beta_j \mid j \in N_i, \alpha_i^{new} = \alpha_j\} - 1 \)

\[ \alpha_i^{new} = \min\{\alpha_j \mid j \in N_i\} + 1 \]

\[ \alpha_i^{old} \]

\( \text{min } \beta \text{ of all neighbors with new } \alpha \)

\( \text{min } \alpha \text{ of all neighbors} \)
Example

(0,4,1) - (0,3,2) - (0,2,3)
(0,5,4) - (0,2,5) - (0,1,6)

Destination: (0,0,0)
Example: Link from 6 to 0 is down

Destination: (0,0,0)
Example: After Node 6 Reverses

1 = min{0,0} + 1

Destination: (0,0,0)
Example: After Nodes 3 and 5 Reverse

1 = \min\{0,1\} + 1; \ 0 = \min\{1\} - 1

Destination: (0,0,0)
Example: After Nodes 2 Reverses

(0,4,1) → (1,-1,2) → (1,0,3)

(0,5,4) → (1,0,5) → (1,1,6)

Destination: (0,0,0)
Example: After Nodes 1 Reverses

Destination: (0,0,0)
Analysis: Convergence

Q: Does the partial reversal algorithm converge?

What is convergence?
- a protocol converges if it stops after a finite number of steps after the last link Up/Down event
Analysis by Invariants

- Invariants are statements defined over the states of the distributed nodes.

- A very effective method in understanding distributed asynchronous systems is invariants.
Analysis

- The state of a node
  - its height as well as its neighbors’ (potentially obsolete) heights

- An invariant over the state of a node
  - $\alpha$ is non-decreasing
Convergence if Not Partitioned

- Partial reversal always converges after a finite number of steps, if not partitioned.
  - Proof by contradiction: assume not converge, i.e., the protocol runs forever after the last change. Since each node will increase its $\alpha$ value by at least one after each step, then there should exist a subset of nodes whose $\alpha$ values go to infinite.
Node A and node B will increase their heights repeatedly forever.
Analysis: Loop of Link Reversal

Does the partial reversal algorithm form loops?
- the algorithm does not have any loops after convergence because the heights form a total order
- before convergence, temporary loops may form due to inconsistent views
Temporally-Ordered Routing Algorithm (TORA)

- TORA modifies the partial link reversal algorithm to be able to detect partitions.
- When a partition is detected, all nodes in the partition are informed, and link reversals in that partition cease.
- For details, see the backup slides at the end.
Summary: Link Reversal Algorithms

Advantages

- the DAG provides many hosts the ability to send packets to a given destination
- beneficial when many hosts want to communicate with a single destination

Disadvantages

- paths may not be the best
- loops before convergence hurt performance
Design Dimensions

- What does each node know?
  - Whole network topology and per link cost (link state)
  - Set of neighbors that can reach dest
    - Link reversal
  - Neighbors’ costs to destination (distance vector)
Distance Vector Routing Algorithm

- Based on the Bellman-Ford algorithm
  - at node X, the distance (or any additive link quality metric) to Y is updated by
    \[
    d^X(Y) = \min_{Z \in N(X)} (d(X, Z) + d^Z(Y))
    \]

  where \(d^X(Y)\) is the current distance at node X from X to Y, \(N(X)\) is the set of the neighbors of X, and \(d(X, Z)\) is the distance of the direct link from X to Z

- Implemented in the RIP routing protocol and some wireless mesh networks
## Distance Table: Example

Below is just one step! The algorithm repeats for ever!

<table>
<thead>
<tr>
<th>destinations</th>
<th>distance tables from neighbors</th>
<th>computation</th>
<th>E’s distance, forwarding table</th>
<th>distance table E sends to its neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 7 ∞</td>
<td>1 15 ∞</td>
<td>1, A</td>
<td>A: 1</td>
</tr>
<tr>
<td>B</td>
<td>7 0 ∞</td>
<td>8 8 ∞</td>
<td>8, B</td>
<td>B: 8</td>
</tr>
<tr>
<td>C</td>
<td>∞ 1 2</td>
<td>∞ 9 4</td>
<td>4, D</td>
<td>C: 4</td>
</tr>
<tr>
<td>D</td>
<td>∞ ∞ 0</td>
<td>∞ ∞ 2</td>
<td>2, D</td>
<td>D: 2</td>
</tr>
<tr>
<td></td>
<td>1 8 2</td>
<td></td>
<td>E: 0</td>
<td></td>
</tr>
</tbody>
</table>

distance tables from neighbors:
- **A**: 0, 7, ∞
- **B**: 7, 0, ∞
- **C**: ∞, 1, 2
- **D**: ∞, ∞, 0

computation:
- **A**: 1, 15, ∞
- **B**: 8, 8, ∞
- **C**: ∞, 9, 4
- **D**: ∞, ∞, 2

E’s distance, forwarding table:
- 1, A
- 8, B
- 4, D
- 2, D

distance table E sends to its neighbors:
- A: 1
- B: 8
- C: 4
- D: 2
- E: 0
Distance Vector in the Presence of Topology Dynamics

- Good news propagate fast

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Initially

- Link AB is up

1  2  3  4  5

After 1 exchange

1  2  3  4  5

After 2 exchanges

1  2  3  4  5

After 3 exchanges

1  2  3  4  5

After 4 exchanges
Distance Vector in the Presence of Topology Dynamics

- Bad news propagate slowly: referred to as the count-to-infinity problem

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>∞</td>
</tr>
</tbody>
</table>

- Link AB is down or cost increases substantially
Why Count-to-Infinity

- Routing loop: a loop is a global state (consisting of the nodes’ local states) at a global moment (observed by an oracle) such that there exist nodes $A_1, A_2, ..., A_n$ such that $A_1$ (locally) thinks $A_2$ as down stream, $A_2$ thinks $A_3$ as down stream, $A_n$ thinks $A_1$ as down stream
There are optimizations but we present the base protocol.

- Only handle the case when link is broken.

Let's assume the destination node is D.

Basic idea:

- DSDV tags each route with a sequence number.
- Each destination node D periodically advertises monotonically increasing even sequence numbers.
- When a node realizes that the link that it uses to reach destination D is broken, the node increases the sequence number for D to be one greater than the previous one (odd number).
DSDV: Details

- Periodical and triggered updates
  - Periodically D increases its seq# $S_D$ by 2 and broadcasts with $(S_D, 0)$
  - If A is using B as next hop to reach D and A discovers that the link AB is broken
    - A increases its sequence number $S^A$ by 1 (odd)
    - Sets $d^A$ to $\infty$, and
    - Sends $(S^A, d^A)$ to all neighbors
DSDV: Details

- Update after receiving a message
  - Assume B sends to A the information \((S^B, d^B)\), where \(S^B\) is the sequence number at B for destination D and \(d^B\) is the distance from B to D; when A receives \((S^B, d^B)\)
    - if \(S^B > S^A\) and A uses B as next hop then
      // higher seq#, always update
      - \(S^A = S^B\)
      - if \((d^B = \infty)\) \(d^A = \infty\); else \(d^A = d^B + d(A,B)\)
    - else if \(S^A = S^B\), then // conditional update
      - if \(d^A > d^B + d(A,B)\)
        \(d^A = d^B + d(A,B)\) and uses B as next hop
DSDV: Example
Question

Does DSDV solve the count-to-infinity problem (i.e., no routing loop is formed)?
Technique

- Again use the invariant method to understand the distributed asynchronous protocol
- Consider any node $A$
- What is the state of node $A$?
  - $(S^A, d^A)$
Consider a Single Node $A$

- What properties do you observe about the state of node $A$, i.e. $(S^A, d^A)$?
  - $S^A$ is non-decreasing
  - $d^A$ is non-increasing for the same sequence number
Invariants

- For any node
  - sequence number is non-decreasing
  - for the same sequence number, distance is non-increasing

- For a pair of nodes, if A (according to local state) considers B as next hop to destination D:
  - either $S_B > S_A$ (B updates seq# after sends update)
  - or $S_B = S_A$
    - implies $d_B < d_A$ if link cost is not zero
Claim: DSDV Does Not Form Loop

- Proof by induction and contradiction
  - Assume initially no loop (no one has next hop so no loop)
  - Derive contradiction by assuming that we have a loop when we add a new link, e.g., when A decides to use B as next hop
Loop Freedom of DSDV

- Consider a critical moment
  - A considers B as next hop and forms a loop

- If any link in the loop (X considers Y as next hop) satisfies $S^Y > S^X$
  - by transition along the loop $S^B > S^B$

- If all nodes along the loop have the same sequence number
  - by transition along the loop $d^B > d^B$
Discussion of DSDV

Q: what is the scope of updates when a link changes status?
Summary: Routing Algorithms

Complete information:
- Nodes maintain complete topology, link cost info
- “link state” algorithm

Partial connectivity information:
- Nodes maintain reachability topology (mesh) for each dest.
- “link reversal” algorithm

Distributed next hop distance:
- Nodes maintain distance to each dest.
- Iterative process of computation, exchange of info with direct neighbors
- “distance vector” algorithm, e.g., DSDV
Outline

- Admin and recap
- Routing
  - Overview
  - Routing metric
  - Computing shortest path routing
    - proactive/complete-info: link state
    - proactive/partial-info: link reversal
    - reactive/partial-info: dynamic source routing
Dynamic Source Routing

- On-demand route discovery
  - Link state, link reversal, DSDV protocols are proactive: they continuously maintain routes/topology
  - DSR is a reactive protocol, maintaining active routes only

- Source routing
  - No need to maintain information at intermediate nodes
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 its own identifier when forwarding RREQ.
Route Discovery: RREQ

Represents a node that has received RREQ for D from S
Route Discovery: RREQ

Broadcast transmission

[S] Represents transmission of RREQ

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

- A request is forwarded by a node if
  - the node is not destination
  - the node has not seen RREQ with the same sequence number (from the same source)

- When forwarding RREQ, use a random delay to avoid collision
• Node H receives RREQ from two neighbors B and C
• Node C and E send RREQ to each other
Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once.
Route Discovery: RREQ
Node D does not forward RREQ, because node D is the intended target of the route discovery.
Route Reply (RREP)

- Destination D on receiving the first RREQ, sends a Route Reply (RREP)
  - why first?

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

- Question: how to send RREP from D back to S?
Route Reply in DSR

- Route Reply is sent by **reversing** the route in Route Request (RREQ)
  - this requires bi-directional link
  - to ensure this, RREQ should be forwarded only if it received on a link that is known to be bi-directional
    - this also necessary for IEEE 802.11 MAC is used to send data, then links have to be bi-directional (since Ack is used)

- If unidirectional (asymmetric) links are allowed, then RREP may need a route discovery from D back to S
Route Reply

Represents RREP control message
When node S sends a data packet to D, the entire route is included in the packet header hence the name source routing.
DSR Optimization: Route Caching

- Each node caches a new route it learns by any means, e.g.,
  - 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 \([S,C,G]\), 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
  - A node may also learn a route when it overhears data packets
DSR: Summary

- Advantages
  - reactive: routes maintained only between nodes who need to communicate
  - route caching can further reduce route discovery overhead

- Disadvantages
  - packet header size grows with route length due to source routing
  - flood of route requests may potentially reach all nodes in the network
Ad Hoc On-Demand Distance Vector

- **Combination of ideas in DSR and DSDV**
  - on-demand mechanism of route discovery and route-maintenance from DSR
  - plus the hop-by-hop routing (not source routing), sequence numbers and periodic beacons from DSDV
Comparison of Routing Algorithms in the Presence of Topology Dynamics
Comparisons of The Protocols

- Some key parameters affecting the performance of a routing protocol
  - number of nodes
  - mobility model,
    - e.g., the way-point model, where each node picks a random target, moves to the target at a random speed picked uniformly in a range, pauses for some time after arrival
  - traffic model, e.g., Constant Bit Rate (CBR)

- Performance metrics
  - packet delivery ratio
  - routing overhead
  - path optimality
Path Optimality: Difference from Optimal

50 nodes; 20 sources; 20 m/s
Comparison: Packet Delivery Ratio

50 nodes; 20 sources; 20 m/s
Comparison: Packet Delivery Ratio

50 nodes; 20 sources; 1 m/s
Summary: Routing

- So far, all routing protocols are in the framework of traditional wireline routing
  - A graph representation of underlying network
    - Point-to-point graph, edges with costs
  - Select a lowest-cost route for a src-dest pair
  - Commit to a specific route before forwarding
  - Each node forwards a received packet as it is to next hop

- Problems: don’t fully exploit path (spatial) diversity and wireless broadcast opportunities