# Lecture 13: Wireless Routing

# Wireless is Different

- **Variable: signal attenuates over space**
- **Interference: other RF sources can interfere with signal**
- **Multipath: signal can self-interfere**
- **Distributed: nodes cannot detect collisions**

# Wired Routing Review

- **Goal: optimal routes**
- **Link state protocols**
- **Distance vector protocols**
- **Assumption: topology is stable**
- **Common metric: hop count**

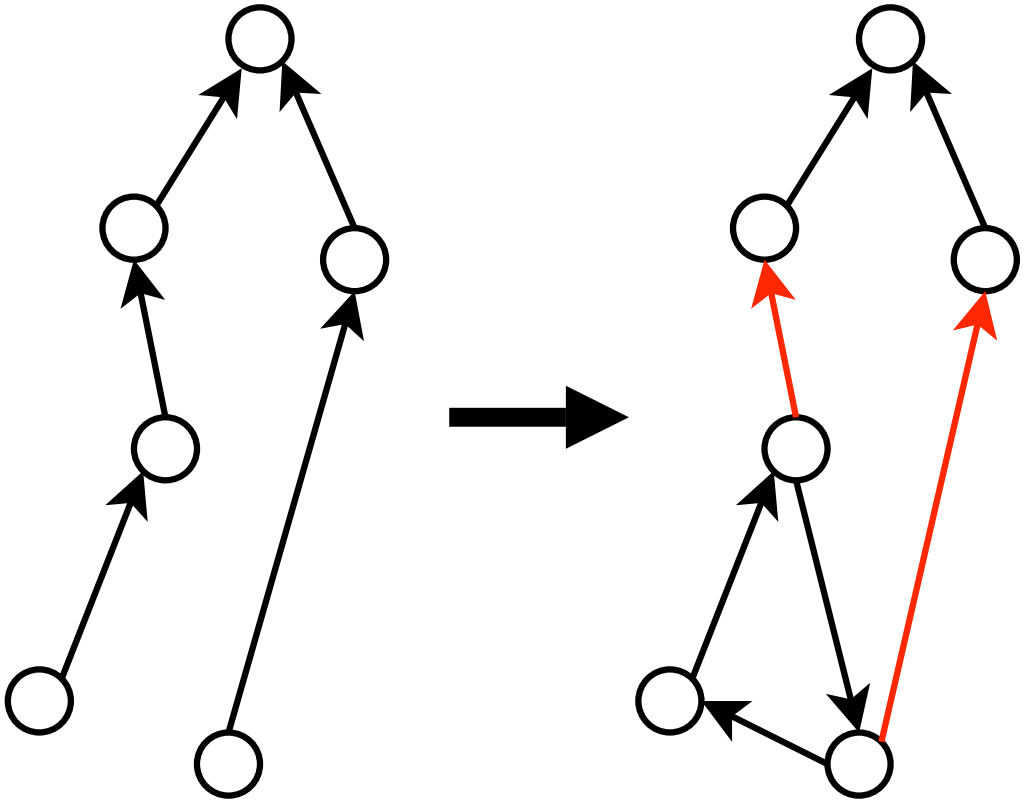
# Wireless Routing

- Network is much more dynamic
- Not constrained by physical topology
- Discovering and estimating links to neighbors
- Discovering and maintaining routes to nodes
- Four protocols: DSDV, AODV, OLSR, Srcr

# Destination Sequenced Distance Vector Routing (DSDV)

- Every node maintains routes to every other node
- Continuous maintenance, rather than on-demand
- Major challenge: loops, as nodes can have out-of-date state

# Loop Formation



# Preventing Loops

- **A route to a destination has a sequence number**
- **When a node advertises its distance vector, it reports sequence numbers of those routes**
- **If node hears a distance vector reporting a newer route to a destination, it immediately uses that route and the newer sequence number**
- **Two cases for incrementing sequence numbers**
  - When a link breaks: set cost to infinity
  - Destination can increment when it advertises

# Advertisement Damping

- **For a given sequence number, node route costs can change significantly**
  - Have to change to new route on larger sequence number
  - Can later hear better routes
- **Don't want to advertise on every route change**
- **Can compute *settling time*, expected time between a new version and the optimal route (EWMA of prior epoch settling times)**

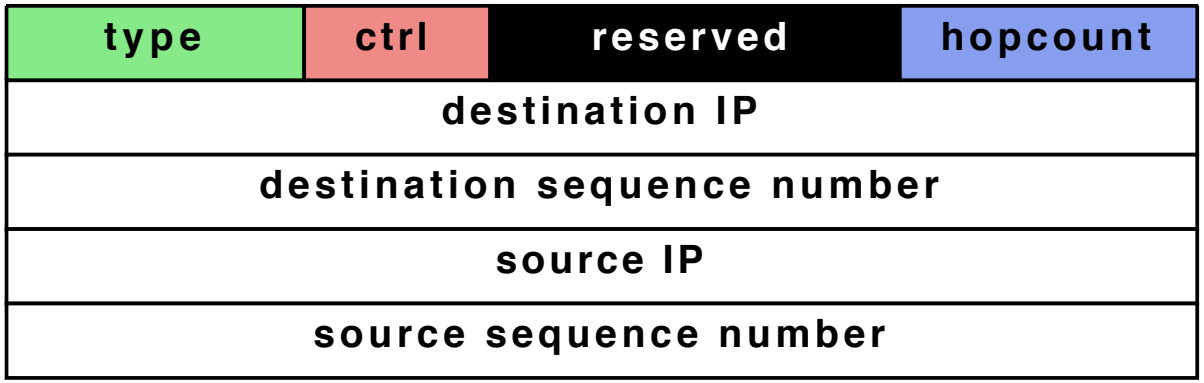
# DSDV Summary

- Distance vector protocol
- Continuously maintains routes to all nodes
- Destination-controlled sequence numbers prevent loops
- Never really tightly specified or seriously implemented

# Ad-hoc On-demand Distance Vector Routing (AODV)

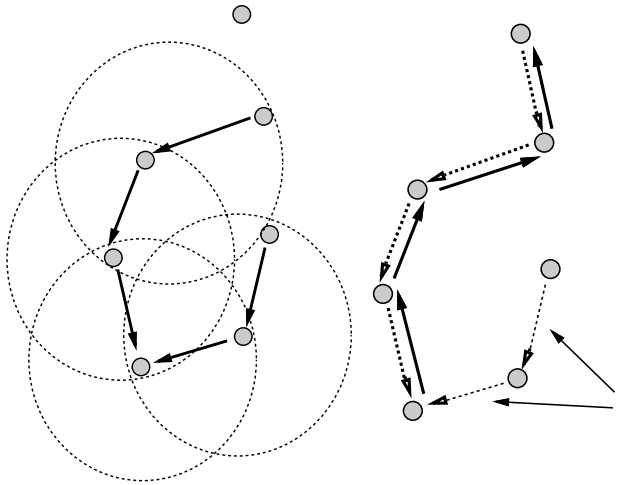
- **Nodes cache routes (next hop) to destinations**
- **If a node needs to send a packet to a destination for which it does not have a next hop, it floods a *route request* message**
- **Receiving a route request implicitly sets up a route**
- **If a node with a route receives a route request, it responds with a *route reply***
- **Base case: destination itself responds with route reply**

# Route Request Packet



# Flooding a Request

- **Each route request flood has a source sequence number to suppress duplicates**
  - A broadcasts, B hears and forwards, A won't reflood B's transmission
- **Hopcount field, incremented on each hop**



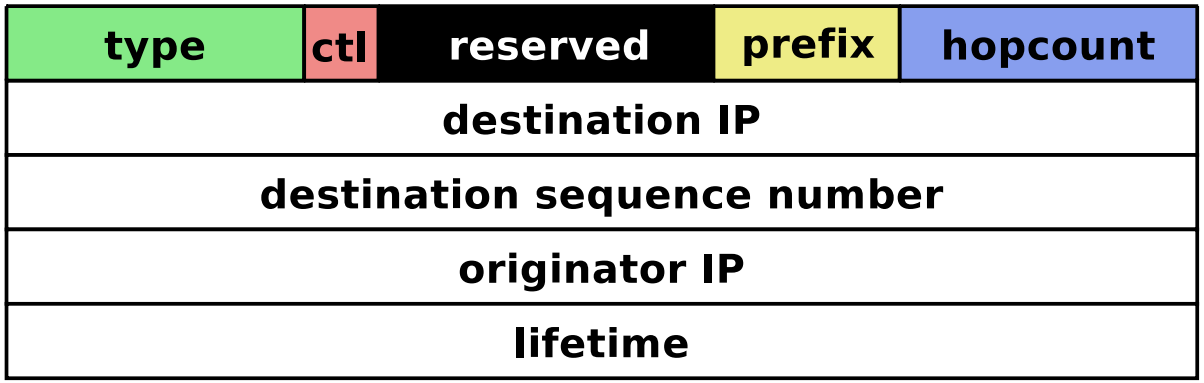
# Controlling the Flood

- **Don't want to flood the entire network for every route request**
- **AODV uses expanding ring search**
  - Send request with IP TTL of 1
  - Send request with IP TTL of 2
  - Send request with IP TTL of 3, etc.

# Request Procesing

- **Requests can set up forward and reverse routes**
- **Source sequence number prevents reflooding**
- **Receiver of request might already have a cached route, with which it can reply**
- **Destination sequence number used to determine if cached routes valid**
  - Whenever a node's link breaks for a route, it increments destination sequence number
  - When you receive a route reply, fill in sequence number
  - If you receive a request with a higher destination sequence number, do not reply from cache

# Route Reply Packet



# Reply Processing

- **Destination: increment sequence number if will be equal to request number**
- **Intermediate: fill in its own sequence number for the destination, increment hopcount**
- **Unicast replies along reverse route**
- **Only forward reply if:**
  - Destination sequence number is higher than earlier replies
  - Same sequence number with smaller hopcount
  - Suppresses redundancy when multiple nodes reply from cache

# AODV Summary

- **Distance vector protocol**
- **Generate routes on-demand by flooding**
- **Sequence numbers distinguish requests and route versions**
- **RFC 3561 (Experimental), superseded by DYMO**
- **Benefits**
  - Stable routes require little control traffic
  - No data traffic $\rightarrow$ no control traffic
- **Drawbacks**
  - Dynamics can lead to significant request/reply flooding

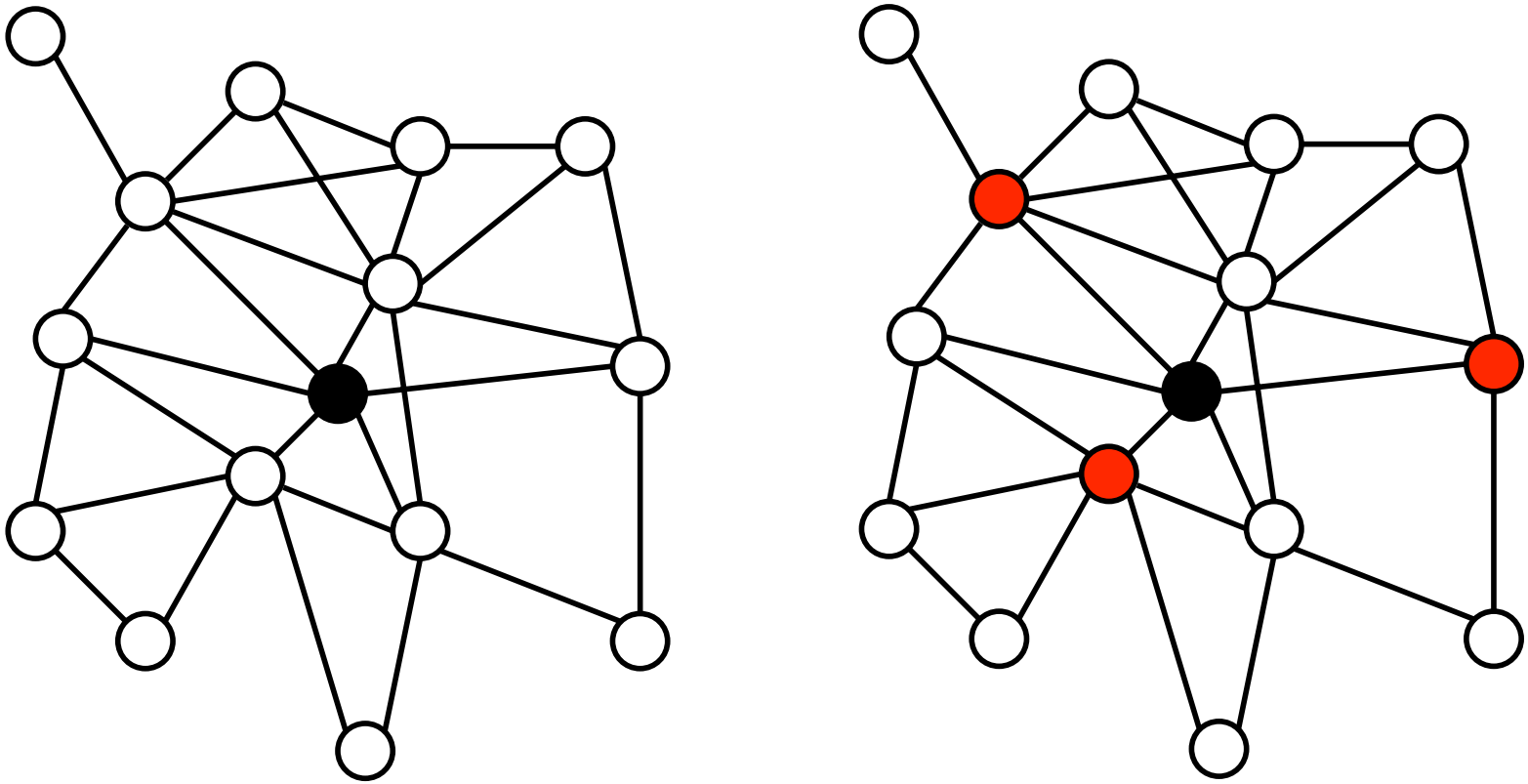
# Optimized Link State Routing Protocol (OLSR)

- **Each node collects local link state information**
- **Link state is flooded through entire network**
- **Multi-Point Relays (MPRs) responsible for flooding**
  - MPRs selected so only some nodes forward a flood
  - Reduces flooding cost and redundancy

# Multi-Point Relays (MPRs)

- Nodes advertise what nodes they have links to
- A node can discover its 2-hop neighborhood by taking the union of all advertisements it hears
- A node selects a subset of its neighbors to be its multi-point relays
- Rule: union of neighborhood sets of MPRs must be the complete 2-hop neighborhood

# MPR Example



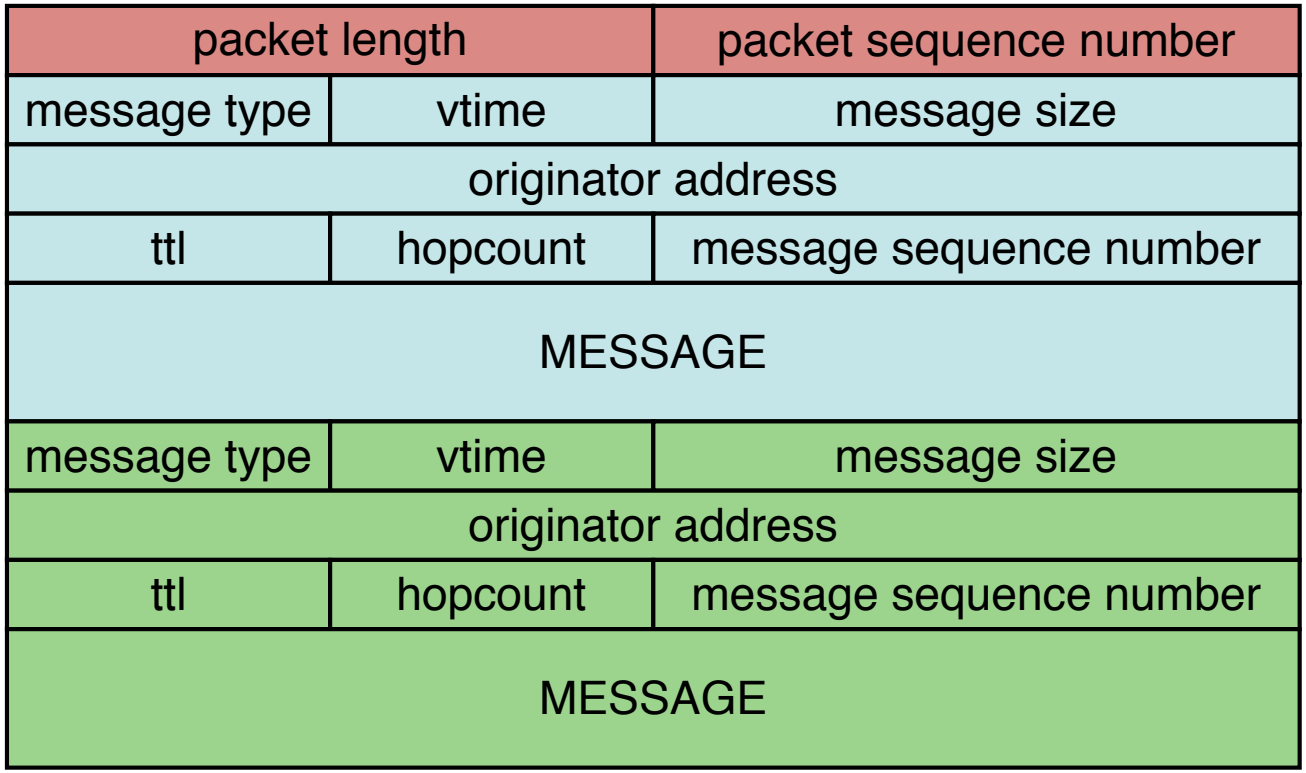
# MPRs, Continued

- **Nodes locally announce their MPR sets**
- **The *MPR selector set* of a node is the set of nodes that have selected it as an MPR**
- **When a node hears a message to flood, forwards it iff**
  - Single hop origin is in selector set
  - Message is not a duplicate
- **MPRs have their own MPRs**

# MPR Set Calculation

- RFC 3626 suggests an algorithm
- Add nodes which are the only node to provide access to a 2-hop neighbor
- From nodes that provide access to at least one new node, select nodes with highest “willingness”
- Among nodes with same “willingness,” pick those with highest degree

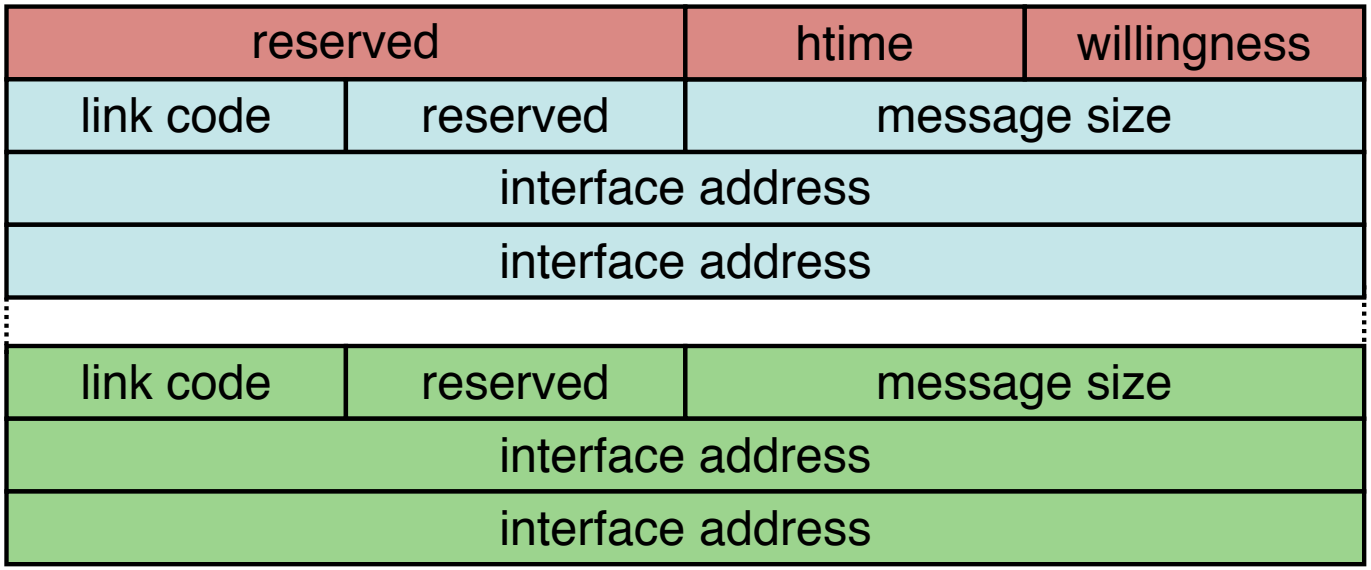
# OLSR Packet Format



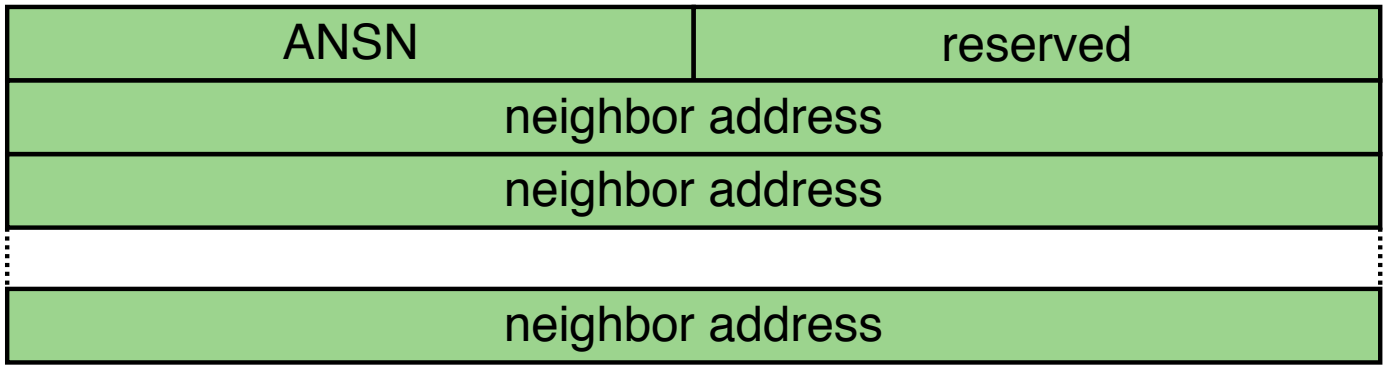
# OLSR Message Types

- **HELLO messages, for link maintenance, local discovery, and communication with MPRs (not forwarded)**
- **TC (topology control) messages, which advertise link state**
- **MID, for declaring multiple interfaces**
- **HNA, for advertising access to networks without OLSR (e.g., gateway)**

# HELLO Message



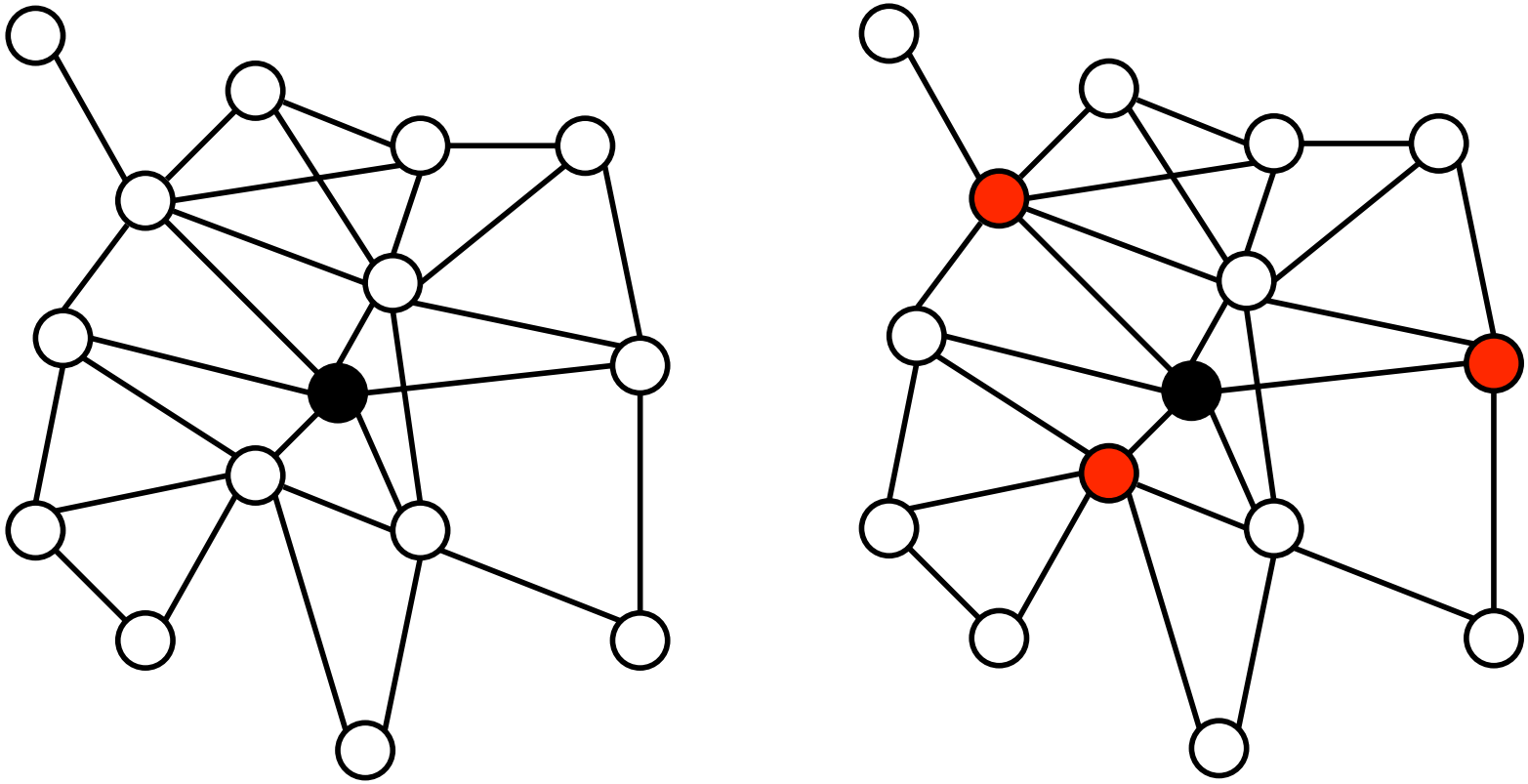
# TC Message


ANSN
reserved
neighbor address
neighbor address
neighbor address

# OLSR Routing

- **Use link state of network to iteratively compute routes**
  - Compute single hop routes from neighborhood
  - Compute 2-hop routes from 2-hop neighborhood
  - Compute $n + 1$ hop route from $n$ hop routes and link state
- **Link state can be limited to links between nodes and their MPRs**

# MPR Set, Revisited



# OLSR Summary

- **Periodic messages discover links**
- **Nodes select MPRs to cover their 2-hop neighborhood**
- **TC messages flooded through MPRs**
- **Routes computed based on link state**
- **Benefits**
  - Link state protocols easier to debug
  - Reference implementations exist
- **Drawbacks**
  - Link state can be expensive to maintain
  - More vulnerable to loops

From: owada@ie.niigata-u.ac.jp
Subject: [manet] Packet looping issue in MANET routing protocol
Date: November 25, 2007 9:04:22 PM PST
To: manet@ietf.org

Hello all,

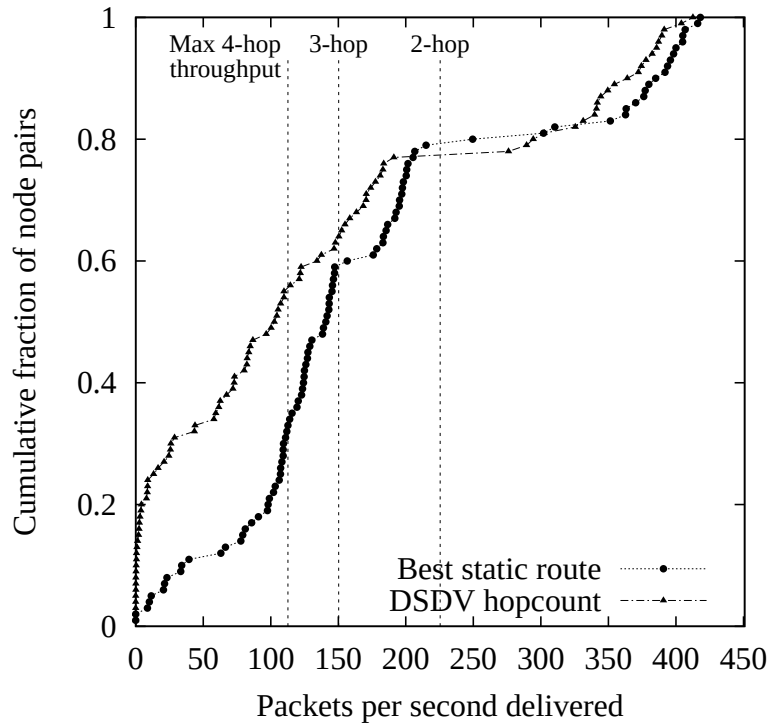
We have implemented OLSRv2 routing protocol that can run on Linux platform as daemon prosess and also run on QualNet simulator, and have been checking the protocol operation and performance. Through the real world experiments and simulation, we've found that MANET routing protocol cannot solve the looping path issue completely. The looping issue is acceralated when the Link Layer Notification is enabled, and much Looping occured with the proactive routing protocol such as OLSR and OLSRv2 compare to the reactive routing protocol such as AODV. This issue is caused form the node's network topology information inconsistency (usually more than 2 hop information inconsistency) for the proactive routing protocols. Once the looping occures, the traffic load jump up localy, and total network load incleases up to 20% in our simulation (50 nodes, random way-point movement, 1500m squared area).

# 2 minute break

# Hopcount Considered Harmful

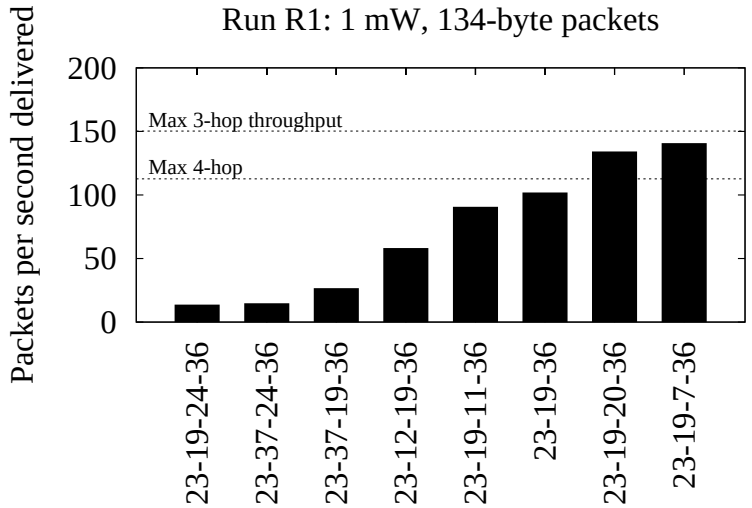
- **Minimizing hopcount causes protocol to choose long links**
- **Links are more likely to be on edge of SNR/PRR curve**
  - Less stable
  - Require more maintenance
- **One way wireless routing is different**
- **OLSRv2 adds the concept of link metrics**

# DSDV and Hopcount on Roofnet




- **From DeCouto et al., "A High-Throughput Path Metric for Multi-Hop Wireless Routing."**

# Variations Across Hopcounts

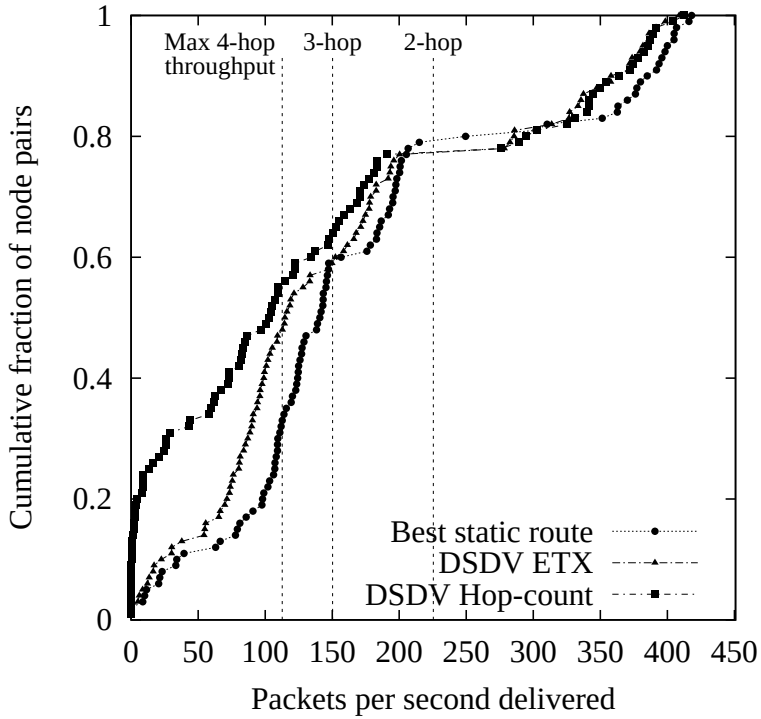



- **From DeCouto et al., "A High-Throughput Path Metric for Multi-Hop Wireless Routing."**

# Expected Transmissions (ETX)

- **Proposed by DeCouto et al.**
- **Alternative metric: ETX, number of transmissions until you receive an ACK**
- **Cost of link is** $\frac{1}{PRR_{AB} \cdot ARR_{BA}}$
  - $PRR_{AB} = 75\%, ARR_{BA} = 66\%, ETX_{AB} = 2.0$
  - $PRR_{AB} = 50\%, ARR_{BA} = 50\%, ETX_{AB} = 4.0$
- **Cost of route is sum of ETX values of links on route**

# ETX Benefits




- **From DeCouto et al., "A High-Throughput Path Metric for Multi-Hop Wireless Routing."**

# ETX Is Not Enough

- **802.11b supports four different bit rates**
- **ETX can select the route, but not the bitrate**
- **One packet at 11Mbps $\neq$ one packet at 1Mbps**
- **Solution: Estimated Time of Transmission (ETT)**
  - Probe at different bit rates
  - Choose link bit rate based on minimum cost

# Link Metrics Today

- **Rough consensus that ETX/ETT is the right metric**
  - Addresses intermediate links
  - Can be used across link layers
- **No consensus on how to estimate the value**
  - Several proposals
  - Still an active area of research
- **Issue: conflates hopcount and link quality, making loops very easy (100% -¿ 33% can look like 2 more hops)**
- **Issue: minimizes delay, does not maximize throughput**

# Srcr

- Link state protocol
- Link costs measured with ETT
- Source routing to prevent loops

# Source Routing

- **Packet source decides on route**
- **Two options in IP: strict and loose**
- **Strict source routing: specify each hop packet takes**
- **Loose source routing: specify some hops packet must take**
  - "Send from A to B via C"

# Source Routing on the Internet

- Loose source routing is a security issue
- Reach private machines (e.g., 10.0.0.1) through gateways
- LSRR packets dropped by most routers
- Srcr uses its own source routing

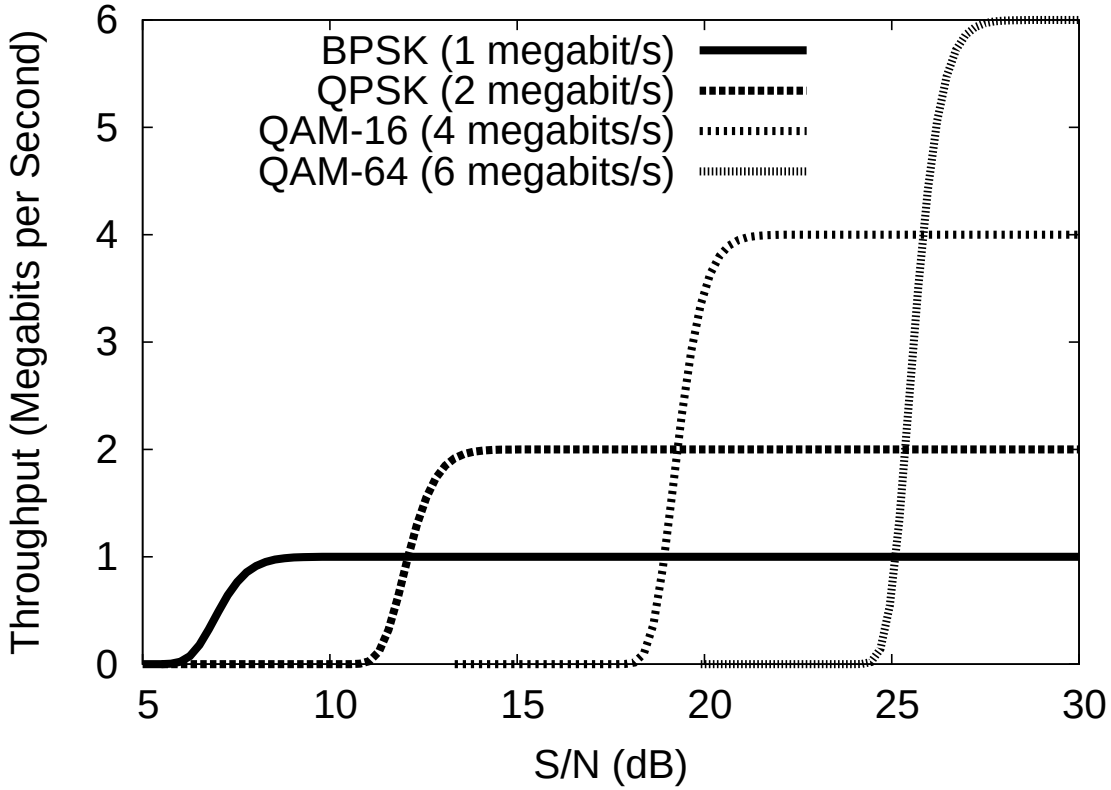
# Srcr ETT Link State

- Srcr measures ETT as time to transmit a 1500 byte packet at highest-throughput bit-rate
- Nodes broadcast periodic 1500-byte packets (all bit-rates) and 60-byte packets (1 Mbps)
- Nodes measure ETX/ETT using a window, detecting lost broadcasts via timing

# Distributing Link State

- Each data packet contains link metric of that link
- Nodes can query for link state
- Nodes can overhear queries and responses
- Querying is like AODV (flooding), replies contain link metrics

# Optimal Bit-rate




- **From John Bicket, "Bit-rate Selection in Wireless Networks."**

# Routing

- **ETT-based link state to pick route**
- **Dynamically adapts bit-rate to maximize throughput**

# Srcr Bit-Rate Algorithm

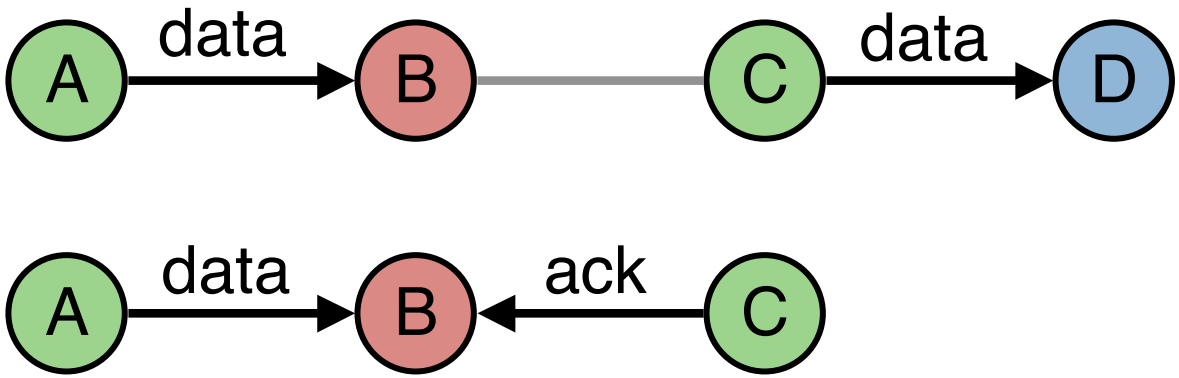
- If no successes, return highest bit rate without 4 successive failures (slower if many failures)
- Every tenth packet, select a random bit-rate that hasn't failed four times and has a minimum transmission time lower than current average transmission time (probing faster)
- Otherwise, transmit at bit-rate with lowest average transmission time (minimize ETT)

# Roofnet Throughput

| Hops | Number of nodes | Throughput (kbits/sec) | Latency (ms) |
|---|---|---|---|
| 1 | 12 | 2752 | 9 |
| 2 | 8 | 940 | 19 |
| 3 | 5 | 552 | 27 |
| 4 | 7 | 379 | 43 |
| 5 | 1 | 89 | 37 |
| Avg: 2.3 | Total: 33 | Avg: 1395 | Avg: 22 |

- **From John Bicket et al., "Architecture and Evaluation of an Unplanned 802.11b Mesh Network."**

# Throughput Dropoff



# Wireless Routing

- **Maintaining consistent, distributed state on a dynamic system**
- **Preventing loops via serialization or source routing**
- **On-demand versus continuous**
- **ETX/ETT better metric than hopcount**