ALOHA

• Possibly first modern data network
  - Deployed in 1970 (before Ethernet)
  - 9,600 Baud

• Designed to communicate across Hawaiian islands
  - Hub at ALOHANET headquarters
  - Many other sites on islands

• Used two radio frequencies
  - One frequency for hub to broadcast messages
  - Another (shared) frequency for other sites to transmit
ALOHA transmission

- Send all packets to hub
  - Hub re-broadcasts all packets it receives

- Listen after sending packet
  - If you hear your packet broadcast, transmission OK
  - Otherwise, assume a collision

- If collision, retransmit after random delay
Multiple access

• Multiple access was revolutionary at the time
  - Now seems like common wisdom

• Previous systems partitioned channels
  - TDMA, FDMA, etc.

• ALOHA optimized case of few senders
  - Can potentially transmit immediately—much lower delay
  - With TDMA, wait your turn while channel unused
  - But have to deal with collisions (which lower throughput)
Throughput

• Initial ALOHANET had $\sim18\%$ throughput
  - $\sim81\%$ of bandwidth wasted by collisions
  - Unlike Ethernet, large distance & no collision detection

• Improved with slotted ALOHA
  - Divide time into equal sized slots
  - Only transmit at beginning of a slot
  - If collision, transmit on future slots with probability $p$

• Slotted ALOHA raised utilization by factor of two
  - Avoided many partial collisions
802.11

- Modern standard for wireless networks
  - 11 Mbps (newer standards allow 54 Mbps)
  - Widely deployed (around NYU, in people’s homes)
  - Can often just pick up someone’s signal to get on net
    (Not recommended because it’s possibly illegal)

- Designed for smaller spaces (250m range)
  - Spread spectrum with frequency hopping
  - Operates over 79 1 Mhz-wide slots around 2.4 GHz
  - Makes it harder to jam the signal (inadvertently)

- Can also use direct sequence, or IR (10m)
Collision avoidance

- Not all nodes are in range of each other
  - Makes situation much more complicated than Ethernet

- Hidden nodes
  - Say A & C send to B, causes undetected collision

- Exposed nodes
  - Say B sends to A, might cause C to delay sending to D
MACA(W)

- **Multiple Access with Collision Avoidance**
  - Sender sends RTS (request to send) pkt, w. length
  - Receiver replies w. CTS (clear to send) pkt, echoing length
  - If you see CTS packet, don’t transmit (you are near receiver)
  - If you see RTS but not CTS, okay to transmit

- **MACA for Wireless LANS (MACAW) adds ACK**
  - Receiver sends ACK for packet
  - All nodes must wait for ACK before sending a packet

- **Two RTS packets may collide**
  - Detect by timing out CTS
  - Backoff similarly to Ethernet
Distribution system

- **802.11** usually operates in *infrastructure mode*
  - *Access points*—non-mobile nodes on wired network

- **Distribution System** connects Access Points
  - Example: A sends to E, AP-1 gets pkt, sends it to AP-3
  - Typically hook to Ethernet & use learning bridge technique
Selecting an AP

- **To select an AP, use scanning**
  - Node sends **probe** frame
  - All APs reply with **probe response** frames
  - Select an AP, send it an **association request** frame
  - AP replies with **association response** frame

- **Scan when joining net, or if unhappy with AP**
  - When node switches from AP-1 to AP-2, AP-2 notifies AP-1 over distribution network

- **Can also use passive scanning**
  - APs periodically broadcast **beacon** frames
  - Nodes may pick up on this and switch APs
802.11 frame format

- **Control field contains 6-bit type field**
  - Types include data, RTS, CTS, scanning

- **Two other bits in control: ToDS & FromDS**
  - DS = Distribution System – when packet goes via DS

- **Notice 4 address fields**
  - When FromDS/ToDS both set, specifies intermediary hops
  - Addr1 – ultimate destination, Addr4 – original source
  - Addr2 – immediate sender, Addr3 – intermediate destination
Ad hoc networks

- Sometimes want to deploy wireless network w/o infrastructure
  - Use nodes to forward data to each other
  - But don’t know anything about node locations a priori

- These are called *ad hoc* networks

- Examples:
  - Emergency workers deployed to disaster area
  - Construction sites, mining equipment
  - Sensor networks, Rooftop networks
The routing problem

- Find end-to-end path (route) from $S$ to $D$
  - Each router must find an adequate next hop
- Topology may be very dynamic in ad hoc networks
Existing routing algorithms

- Can we just use DV or LS routing protocols?

- Problems:
  - DV & LS both send maintenance packets – uses bandwidth and also maybe battery power (important for ad hoc nets)
  - Wireless networks have many more redundant links (increases size of routing updates, CPU utilization)
  - Asymmetry – host $A$ may hear broadcast from $B$, but not be able to send to $B$
  - Loops – simple DV/LS protocols can cause loops given incorrect information. More likely in dynamic environment.
DSR

• Dynamic Source Routing
  - Each packet contains source route—hops to go through
  - Routes determined on-the-fly, heavily cached

• Route discovery – to find a route to target $D$
  - Packet contains a route record with path from sender
  - If a node has the route, prepends route record and replies with route reply
  - As special case, if node is target this works, too
  - Otherwise, nodes re-broadcast the route request
  - Append own address to route record before re-broadcasting
DSR continued

- **Avoid re-broadcasting same packets**
  - Each route request contains a *request id*
  - Cache recently seen \( \langle \text{initiator-id}, \text{request-id} \rangle \) pairs
  - Discard already seen \( \langle \text{initiator-id}, \text{request-id} \rangle \) pairs
  - Also discard if your route already appears in route record

- **When sending route reply packet**
  - If links bi-directional, can reverse route
  - Else, must discover own route to source (if none cached)
  - Can piggyback route reply onto route request packet
Route maintenance

- **May have many broken links when node move**
  - Expensive – takes time to timeout packet to bad destination
  - But proactive notification would be expensive, too

- **Report broken links with *route error* packets**
  - Must truncate all routes that use the broken link

- **Requires ACKs to know if next hop got packet**
  - Okay for MACAW
  - w/o ACKs, can use passive acknowledgement by listening for re-broadcast
Optimizations

- Intermediary hops augment their caches
  - If you reach $D$ via $A \rightarrow B \rightarrow C \rightarrow D$, have route for $C$
  - When $B$ forwards packet $A \rightarrow D$, learns routes to $C$ and $D$
  - Can also use promiscuous mode to learn routes from others

- Limiting collisions of route reply packets
  - Delay $d = H(h - 1 + r)$, where $h$ is number of hops in route, $r$ is random number $0 \leq r \leq 1$
  - Listen promiscuously for other route replies during delay
  - Favors shorter routes, avoids collisions, reduces traffic

- Route request has max # times to re-broadcast
  - Request 1 hop first, then $\infty$
More optimizations

• **Piggybacking on route discoveries**
  - Small packets (e.g., TCP SYN) can be piggybacked
  - But must handle cached routes differently—reply with cached route *and* forward packet to destination

• **Discovering short routes**
  - Suppose using routes $B \rightarrow C \rightarrow D$ and $D \rightarrow C \rightarrow B$
  - If $B$ moves close to $D$, would like to send directly
  - Discover this special case using promiscuous mode

• **Eavesdrop on route error packets**

• **Cache bad links**
  - Because of asymmetry, might receive cached routes containing a link just purged from your routing table
Limitations of DSR

- On-demand routing adds latency
- Each node must keep lots of state
  - Every route the node employs
  - Possibly other routes overheard through eavesdropping
- Extra overhead for routing packets
- What is solution proposed by GPSR?
GPSR

- Use geography to forward packets
  - Every node can know its exact location w. GPS receiver
  - E.g., all new cell phones have GPS built in for E911

- Assume some way to look up nodes’ locations
  - E.g., There is some mapping from IP address to location
  - There are P2P ways of doing this (perhaps discuss later)

- Forward pkts using location to minimize state

- Other assumptions
  - Bi-directional communication (as with 802.11 RTS/CTS)
  - All nodes roughly at same altitude
  - No weird obstacles to transmission
Greedy forwarding

Nodes learn immediate neighbors’ positions through beacons/piggybacking on data packets

Locally optimal, greedy forwarding choice at a node:
Forward to the neighbor geographically closest to the destination

Are we done?
Greedy forwarding failure

Greedy forwarding not always possible! Consider:
Voids

When the *intersection* of a node’s circular radio range and the circle about the destination on which the node sits is empty of nodes, greedy forwarding is impossible.

Such a region is a **void**:
Node Density and Voids

Probability of empty void region increases as nodes become sparser

Unacceptable not eventually to find valid routes
Void Traversal: The Right-Hand Rule

Well-known graph traversal: **right-hand rule**:

When arriving at \( x \) from \( y \), move to the next vertex counterclockwise about \( x \) from \( y \)

Traverses interior faces in clockwise edge order; exterior faces in counterclockwise edge order

So use RHR when greedy fails. Are we done?
Planar vs. Non-Planar Graphs

The right-hand rule may not tour enclosed faces on graphs with edges that cross (non-planar graphs).

Seek a distributed algorithm that removes crossing edges without partitioning the network, using only neighbors’ positions as input.
Planarized Graphs

Relative Neighborhood Graph (RNG) [Toussaint, ’80] and Gabriel Graph (GG) [Gabriel, ’69] are long-known planar graphs.

Assume an edge exists between any pair of nodes separated by less than a threshold distance (i.e., the nominal radio range).

RNG and GG can be constructed using only neighbors’ positions, and can be shown not to partition the network!
Planarized Graphs: Example

200 nodes, placed uniformly at random on a 2000-by-2000-meter region; radio range 250 meters
Full Greedy Perimeter Stateless Routing

All packets begin in greedy mode

Upon greedy failure, node marks its current location in packet, and marks packet in perimeter mode

Perimeter mode packets follow simple planar graph traversal:

Forward along successively closer faces by right-hand rule, until reaching destination, or node closer to it than perimeter mode entry point

Return packets to greedy mode when they reach a node closer than their perimeter mode entry point
Perimeter Mode Forwarding Example

Traverse face closer to $D$ along $xD$ by right-hand rule, until reaching the edge that crosses $xD$

Repeat with the next closer face along $xD$, etc.

Record first edge on face to detect disconnection
# GPSR packet header

<table>
<thead>
<tr>
<th>Field</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>Destination Location</td>
</tr>
<tr>
<td>$L_p$</td>
<td>Location Packet Entered Perimeter Mode</td>
</tr>
<tr>
<td>$L_f$</td>
<td>Point on $xV$ Packet Entered Current Face</td>
</tr>
<tr>
<td>$e_0$</td>
<td>First Edge Traversed on Current Face</td>
</tr>
<tr>
<td>$M$</td>
<td>Packet Mode: Greedy or Perimeter</td>
</tr>
</tbody>
</table>