

Mobility & Wireless

Mobile IP

- You want your laptop to keep its IP address.
(Why?)
- You go from home, to work, to Starbucks, *etc.*
- IP Addresses reflect the structure of the network
(Why?)
- How can the Internet cope with mobile IP addresses?

Mobile IP

- Mobile IP uses *indirect routing*
- Nodes have a “home agent” that stays behind.
- Home agent tunnels packets to the mobile node.
- Route optimization for participating end-hosts.

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)

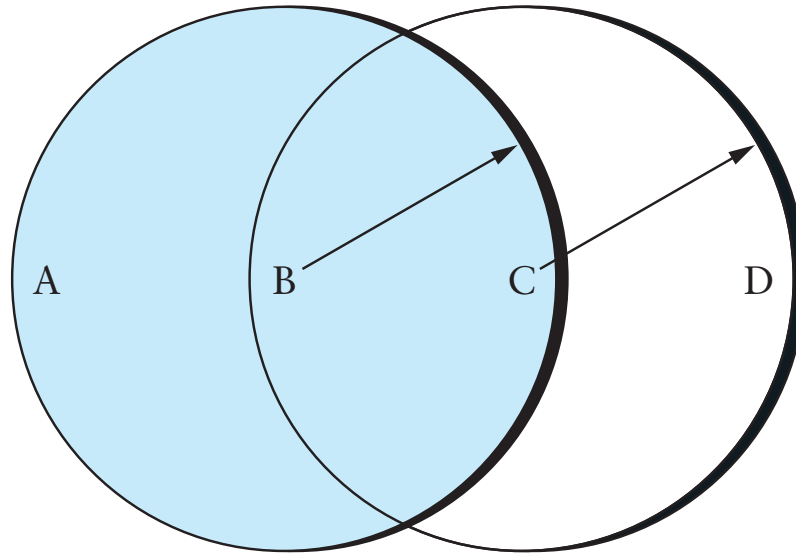
Throughput

- **Initial ALOHNET had ~18% throughput**
 - ~81% 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 (*not* TDMA!)
 - 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 Brown, 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)

Collision avoidance

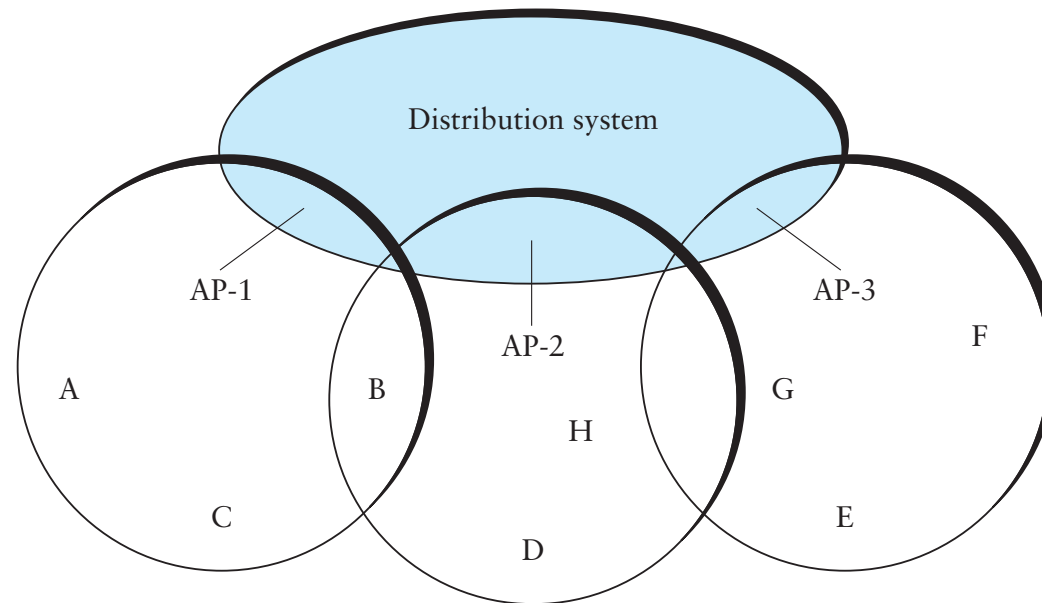


- **Not all nodes are in range of each other**
 - Makes situation much more complicated than Ethernet
- ***Hidden nodes***
 - Say A & C send to B, causes undetected collision
- ***Exposed nodes***
 - Say B sends to A, might cause C to delay sending to D

MACA(W)

- **Multiple Access with Collision Avoidance**
 - Sender sends RTS (request to send), with length
 - Receiver replies with CTS (clear to send), 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

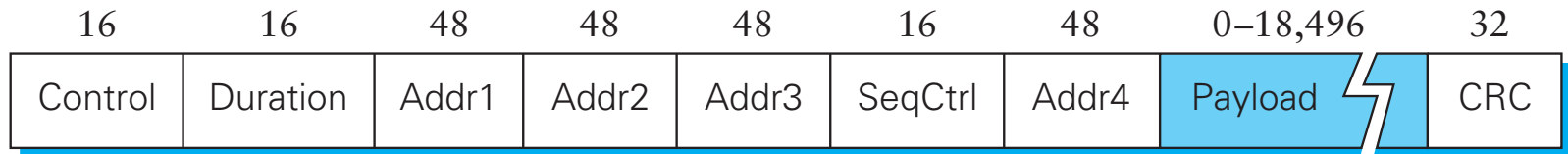


- **802.11 usually operates in *infrastructure mode***
 - *Access points*—non-mobile nodes on wired network
- ***Distribution System* connects Access Points**
 - Example: A sends to E, AP-1 gets frame, sends it to AP-3
 - Typically hook to Ethernet & use learning bridge.

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

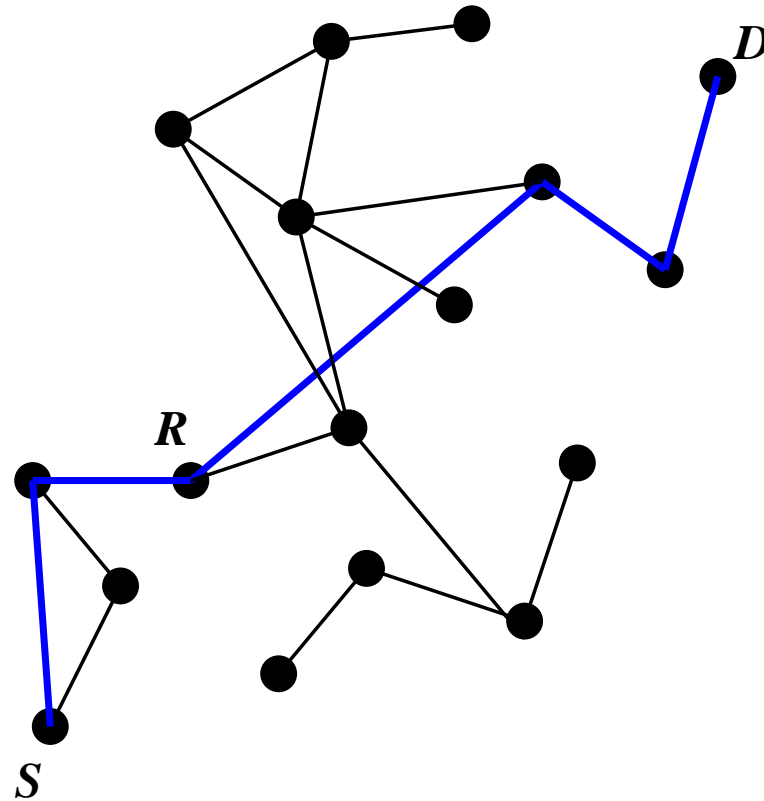


- **Control field contains 6-bit type field**
 - Types include data, RTS, CTS, scanning
- **Two other bits in control: ToDS & FromDS**
 - DS = Distribution System – when packet goes via DS
- **Notice 4 address fields**
 - When FromDS/ToDS both set, specifies intermediary hops
 - Addr1 – ultimate destination, 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 temporary loops given incorrect information. More likely in dynamic environment.

DSDV

- **Destination-Sequenced Distance-Vector**
- **Specializes DV for ad-hoc networks.**
- **Sequence numbers are used to avoid loops and counting to ∞ .**

DSDV Steps

- **Nodes still periodically broadcast route table**
 - Sequence of the route to itself incremented.
- **If “directly” connected link is down:**
 - Set costs of entries that use the link to ∞
 - Increment their sequence numbers.
 - Trigger an update.
- **Accept entries that:**
 - Have a higher sequence number than your copy.
 - Have an equal sequence number, but lower cost.

DSDV Problems

- Control message traffic grows as $O(n^2)$.
- In an ad-hoc environment, changes are frequent.
- Each node maintains complete table.

Proactive vs. Reactive Routing

- **Proactive Routing**

- Strive to constantly maintain potential routes
- More: control plane overhead, router state
- Less: routing latency

- **Reactive Routing**

- Compute routes on demand.
- Less: routing latency
- More: control plane overhead, router state

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 initiator-id, request-id \rangle pairs
 - Discard already seen \langle initiator-id, 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 ∞

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 (surprise, given proactive)**
 - Every route the node employs
 - Possibly other routes overheard through eavesdropping
- **Source-route is overhead in data packets.**

Geographic Forwarding

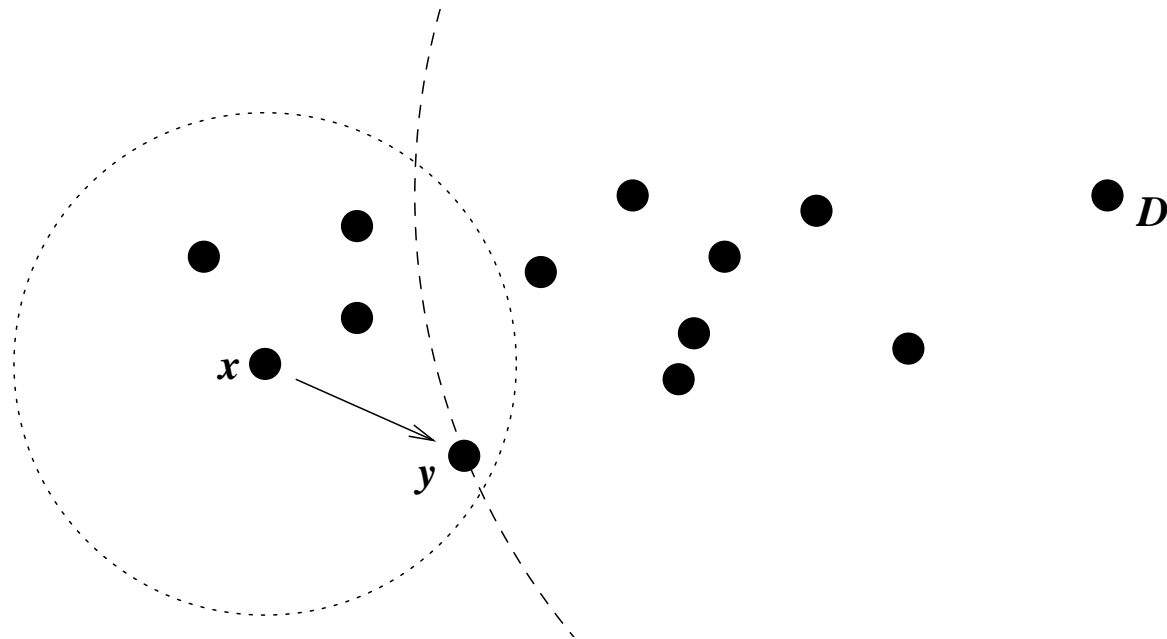
- **Use geography to forward packets**
 - Every node can know its exact location w. GPS receiver
 - *E.g.*, all new cell phones have GPS built in for E911
- **Assume some way to look up nodes' locations**
 - *E.g.*, There is some mapping from IP address to location
 - There are scalable ways of doing this ("Grid" for the curious)
- **Forward 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:

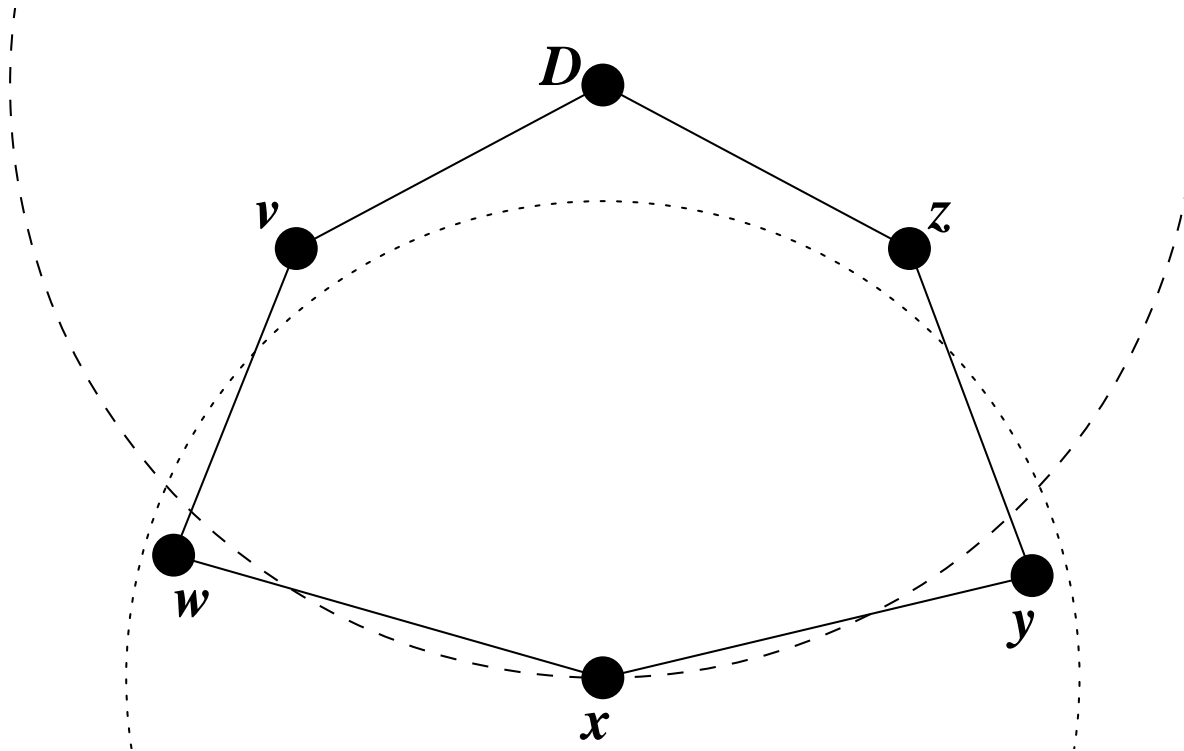
Forward to the neighbor geographically closest to the destination



Are we done?

Greedy forwarding failure

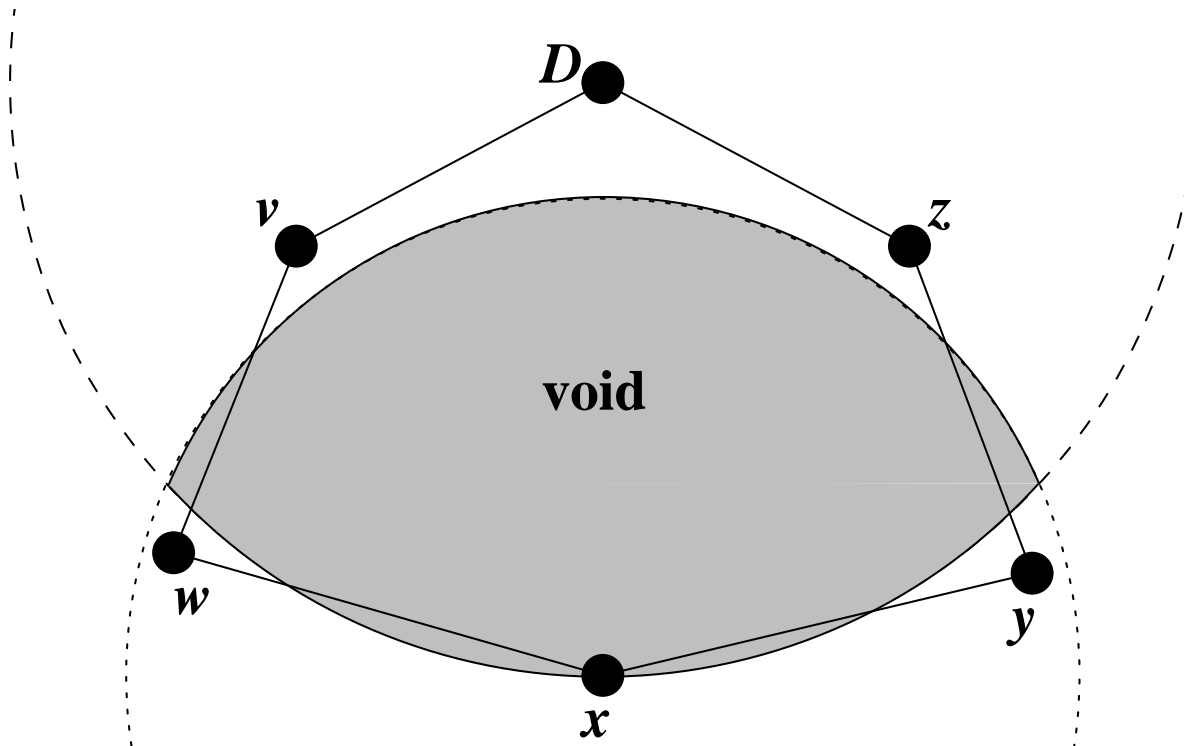
Greedy forwarding not always possible! Consider:



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**:



Coming up

- Today, HW2 out.
- Tomorrow, IP due.
- TCP (and “second half”) starts.
- Thu, 4:30pm: VMWare talk, recruiting [Lubrano]
- Tue, Oct 16th: **Midterm**
- Tue, Oct 17th 6p: Sun talk, recruiting [Lubrano]