15-441: Computer Networking

Lecture 22: Wireless, Ad-Hoc Networks, Sensor Networks, and Delay Tolerant Networks

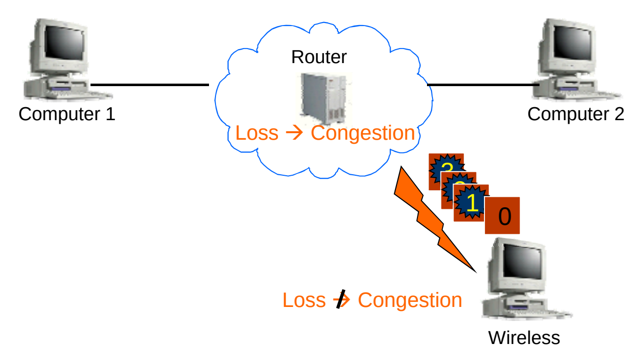
Scenarios and Roadmap

- Point to point wireless networks
  - Review of important concepts
- Ad hoc networks (wireless++)
  - Rooftop networks (multi-hop, fixed position)
  - Mobile ad hoc networks
  - Adds challenges: routing, mobility
  - Some deployment + some research
- Sensor networks (ad hoc++)
  - Scatter 100s of nodes in a field / bridge / etc.
  - Adds challenge: Serious resource constraints
  - Current, popular, research.
- Delay Tolerant Networks (DTNs)
  - When All this fails…what do we do?

Wireless Challenges (review)

- Need to share airwaves rather than wire
  - Don't know what hosts are involved
  - Host may not be using same link technology
  - No fixed topology of interconnection
  - Interference
    - Other hosts: collisions, capture, interference
    - The environment (e.g., microwaves + 802.11)
- Mobility -> Things change often
  - Environmental changes do too
- Other characteristics of wireless
  - Noisy → lots of losses
  - Slow
  - Multipath interference

Wireless Bit-Errors



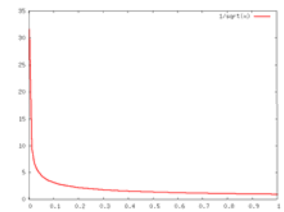
TCP Problems Over Noisy Links

- Wireless links are inherently error-prone
  - Fading, interference, attenuation -> Loss & errors
  - Errors often happen in bursts
- TCP cannot distinguish between corruption and congestion
  - TCP unnecessarily reduces window, resulting in low throughput and high latency
- Burst losses often result in timeouts
  - What does fast retransmit need?
- Sender retransmission is the only option
  - Inefficient use of bandwidth

Performance Degradation

- Recall TCP throughput / loss / RTT rel:
  - BW = MSS / (rtt * sqrt(2p/3))
  - = proportional to 1 / rtt * sqrt(p)
  - == ouch!
  - Normal TCP operating range: < 2% loss

Internet loss usually < 1%

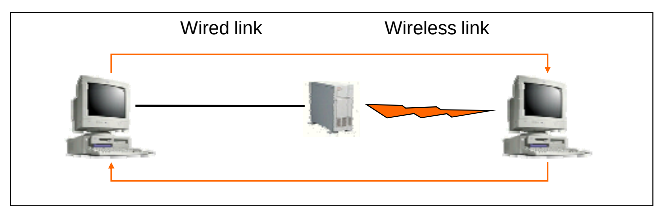


Proposed Solutions

- Incremental deployment
  - Solution should not require modifications to fixed hosts
  - If possible, avoid modifying mobile hosts
- Reliable link-layer protocols
  - Error-correcting codes (or just send data twice)
  - Local retransmission
- End-to-end protocols
  - Selective ACKs, Explicit loss notification
- Split-connection protocols
  - Separate connections for wired path and wireless hop

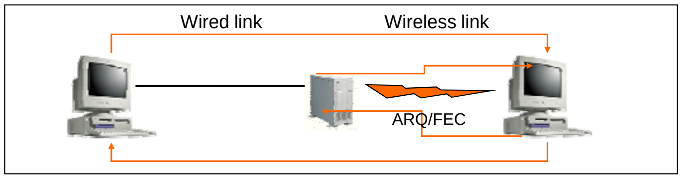
Approach Styles (End-to-End)

- Improve TCP implementations
  - Not incrementally deployable
  - Improve loss recovery (SACK, NewReno)
  - Help it identify congestion
    - Explicit Loss/Congestion Notification (ELN, ECN),
    - ACKs include flag indicating wireless loss
  - Trick TCP into doing right thing → E.g. send extra dupacks if you know the network just burped (e.g., if you moved)



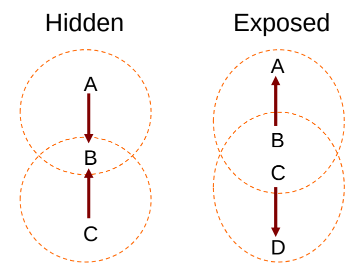
Approach Styles (Link Layer)

- More aggressive local rexmit than TCP
  - 802.11 protocols all do this. Receiver sends ACK after last bit of data.
  - Faster; Bandwidth not wasted on wired links. Recover in a few milliseconds.
- Possible adverse interactions with transport layer
  - Interactions with TCP retransmission
  - Large end-to-end round-trip time variation
    - Recall TCP RTO estimation. What does this do?
- FEC used in some networks (e.g., 802.11a)
  - But does not work well with burst losses



Next: CSMA/CD Does Not Work

- Recall Aloha from many lectures ago
  - Wireless precursor to Ethernet.
- Carrier sense problems
  - Relevant contention at the receiver, not sender
  - Hidden terminal
  - Exposed terminal
- Collision detection problems
  - Hard to build a radio that can transmit and receive at same time



RTS/CTS Approach

- Before sending data, send Ready-to-Send (RTS)
- Target responds with Clear-to-Send (CTS)
- Others who hear CTS defer transmission
  - Packet length in RTS and CTS messages
  - Why not defer on RTS alone?
- If CTS is not heard, or RTS collides
  - Retransmit RTS after binary exponential backoff
  - (There are lots of cool details embedded in this last part that went into the design of 802.11 - if you're curious, look up the "MACAW" protocol).

Ad Hoc Networks

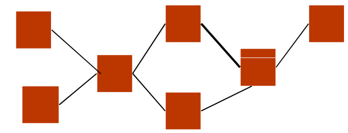
- All the challenges of wireless, plus some of:
  - No fixed infrastructure
  - Mobility (on short time scales)
  - Chaotically decentralized (:-)
  - Multi-hop!
- Nodes are both traffic sources/sinks and forwarders
- The big challenge: Routing

Ad Hoc Routing

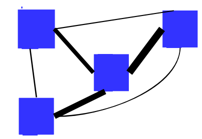
- Find multi-hop paths through network
  - Adapt to new routes and movement / environment changes
  - Deal with interference and power issues
  - Scale well with # of nodes
  - Localize effects of link changes

Traditional Routing vs Ad Hoc

- Traditional network:
  - Well-structured
  - ~O(N) nodes & links
  - All links work ~= well



- Ad Hoc network
  - N^2 links - but many stink!
  - Topology may be really weird
    - Reflections & multipath cause strange interference
  - Change is frequent



Problems using DV or LS

- DV loops are very expensive
  - Wireless bandwidth << fiber bandwidth…
- LS protocols have high overhead
- N^2 links cause very high cost
- Periodic updates waste power
- Need fast, frequent convergence

Proposed protocols

- Proactive
  - Modified/Optimized DV or LS
  - Each node maintains route to all other nodes
  - Periodic and/or event triggered routing
  - Ex1: Destination-Sequenced Distance Vector (DSDV)
  - Ex2: Optimized Link State Routing (OLSR)
- Reactive
  - Routes are built on-demand
  - Maintains only active routes
  - Dynamic Source Routing (DSR)
  - Ad Hoc On-Demand Distance Vector (AODV)
- Proactive vs Reactive
  - Proactive has more overhead and longer convergence.
  - Reactive causes more transmission latency
  - Choice would depend on target network
- Let's look at DSR

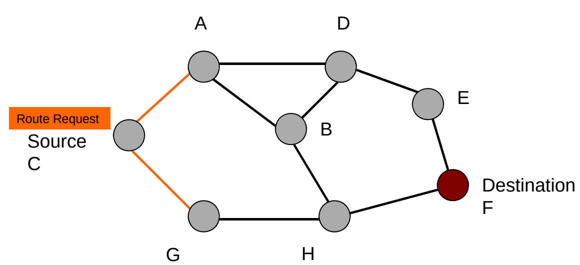
DSR Components

- Route discovery
  - The mechanism by which a sending node obtains a route to destination
- Route maintenance
  - The mechanism by which a sending node detects that the network topology has changed and its route to destination is no longer valid

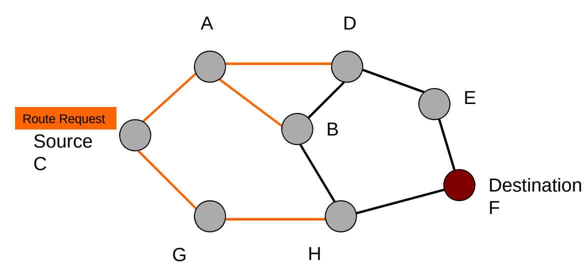
DSR Route Discovery

- Route discovery - basic idea
  - **Source** broadcasts route-request to **Destination**
  - Each node forwards request by adding own address and re-broadcasting
  - Requests propagate outward until:
    - Target is found, or
    - A node that has a route to Destination is found

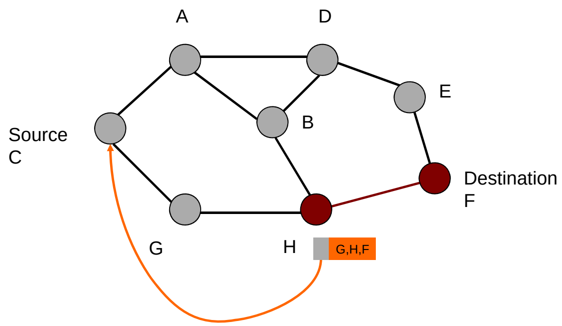
C Broadcasts Route Request to F



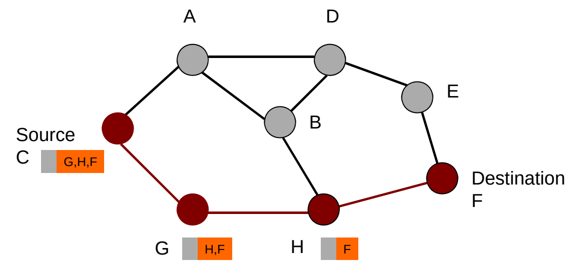
C Broadcasts Route Request to F



H Responds to Route Request



C Transmits a Packet to F



Forwarding Route Requests

- A request is forwarded if:
  - Node is not the destination
  - Node not already listed in recorded source route
  - Node has not seen request with same sequence number
  - IP TTL field may be used to limit scope
- Destination copies route into a Route-reply packet and sends it back to **Source**

Route Cache

- All source routes learned by a node are kept in Route Cache
  - Reduces cost of route discovery
- If intermediate node receives RR for destination and has entry for destination in route cache, it responds to RR and does not propagate RR further
- Nodes overhearing RR/RP may insert routes in cache

Sending Data

- Check cache for route to destination
- If route exists then
  - If reachable in one hop
    - Send packet
  - Else insert routing header to destination and send
- If route does not exist, buffer packet and initiate route discovery

Discussion

- Source routing is good for on demand routes instead of a priori distribution
- Route discovery protocol used to obtain routes on demand
  - Caching used to minimize use of discovery
- Periodic messages avoided
- But need to buffer packets

Forwarding Packets is expensive

- Throughput of 802.11b =~ 11Mbits/s
  - In reality, you can get about 5.
- What is throughput of a chain?
  - A -> B -> C ?
  - A -> B -> C -> D ?
  - Assume minimum power for radios.
- Routing metric should take this into account

ETX

- Measure each link's delivery probability with broadcast probes (& measure reverse)
- P(delivery) = 1 / (df * dr) (ACK must be delivered too)
- Link ETX = 1 / P(delivery)
- Route ETX = sum of link ETX
- (Assumes all hops interfere - not true, but seems to work okay so far)

Capacity of multi-hop network

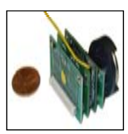
- Assume N nodes, each wants to talk to everyone else. What total throughput can we get?
  - We have N nodes, if perfect, we can get a total capacity of O(n). Great! But:
  - Each has length O(sqrt(n))
  - So each Tx requires up to sqrt(n) of the O(n) capacity.
  - Per-node capacity scales as 1/sqrt(n)
    - Yes - it goes down! More time spent Tx'ing other peoples packets…
- But: If communication is local, can do much better, and use cool tricks to optimize
  - Like multicast, or multicast in reverse (data fusion)
  - Hey, that sounds like … a sensor network!

Sensor Networks - smart devices

- First introduced in late 90's by groups at UCB/UCLA/USC
- Small, resource limited devices
  - CPU, disk, power, bandwidth, etc.
- Simple scalar sensors – temperature, motion
- Single domain of deployment
  - farm, battlefield, bridge, rain forest
- for a targeted task
  - find the tanks, count the birds, monitor the bridge
- Ad-hoc wireless network

Sensor System Types – Smart-Dust/Motes

- Hardware
  - UCB motes
  - 4 MHz CPU
  - 4 kB data RAM
  - 128 kB code
  - 50 kb/sec 917 Mhz radio
  - Sensors: light, temp.,
    - Sound, etc.,
  - And a battery.




Sensors and power and radios

- Limited battery life drives most goals
- Radio is most energy-expensive part.
- 800 instructions per bit. 200,000 instructions per packet. (!)
- That's about one message per second for ~2 months if no CPU.
- Listening is expensive too. :(

Sensor nets goals

- Replace communication with computation
- Turn off radio receiver as often as possible
- Keep little state (4 KB isn't your pentium 4 ten bazillion gigahertz with five ottabytes of DRAM).

Power

- Which uses less power?
  - Direct sensor -> base station Tx
    - Total Tx power: distance^2
  - Sensor -> sensor -> sensor -> base station?
    - Total Tx power: n * (distance/n) ^2 =~ d^2 / n
  - Why? Radios are omnidirectional, but only one direction matters. Multi-hop approximates directionality.
- Power savings often makes up for multi-hop capacity
  - These devices are *very* power constrained!
- Reality: Many systems don't use adaptive power control. This is active research, and fun stuff.

Example: Aggregation

- Find avg temp in 8th floor of Wean.
- Strawman:
  - Flood query, let a collection point compute avg.
    - Huge overload near the CP. Lots of loss, and local nodes use lots of energy!
- Better:
  - Take local avg. first, & forward that.
    - Send average temp + # of samples
  - Aggregation is the key to scaling these nets.
- The challenge: How to aggregate.
  - How long to wait?
  - How to aggregate complex queries?
  - How to program?

Delay Tolerant Networks (DTNs)

- Unstated Assumptions:
  - A path exists between endpoints
    - Routing protocols find the best path, or even "a path"
  - Small end-to-end RTT
    - Millisecond range
  - End to end reliability works well
    - Especially for low data loss rates
    - Loss = Congestion
  - Packet switching is the "right" abstraction
    - IP does best effort delivery for each packet separately

Applications

- InterPlanetary Networks
- Disconnected mobile/sensor networks
- Acoustic/underwater networks
- Military/tactical networks
- Disaster response

Core Ideas for DTN solutions

- Regional gateways for protocol translation
- Persistent storage
- Custody Transfer
- Scheduled vs Opportunistic routing
- Parallel Networks