Project 5: Ad-Hoc Networking

Owen Arden

owen@cs.cornell.edu

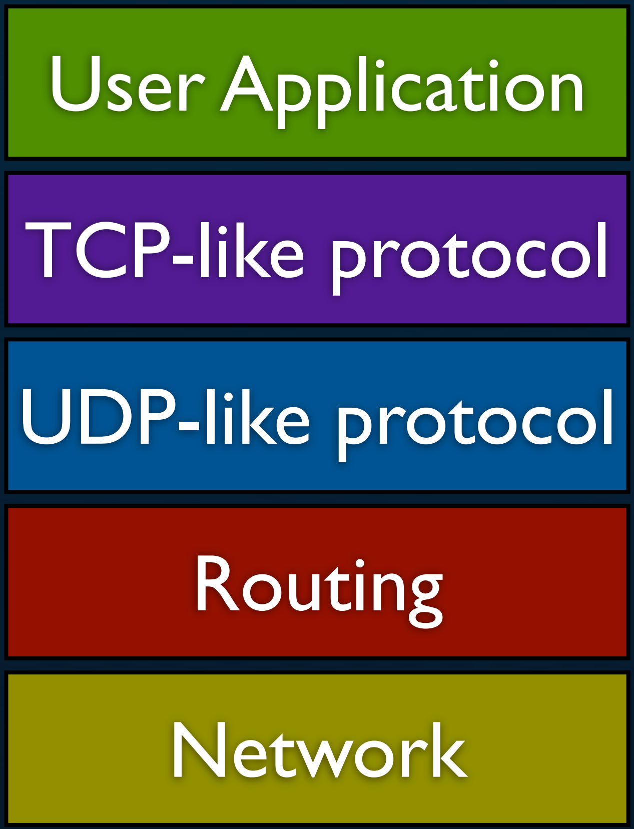
Modified from last year's slides

Miniroute

- Ad-hoc networking layer
  - Allows multi-hop wireless communication without the need for infrastructure
  - Low-cost
  - Quick deployment time
  - No single point of failure
- Based on Dynamic Source Routing (DSR)
  - http://www.cs.cornell.edu/People/egs/615/johnson-dsr.pdf

What is routing?

- Packets that arrive at your machine may not be for you
- Add a routing layer between network and transport layer
- minimsg/sockets works on top



User Application

TCP-like protocol

UDP-like protocol

Routing

Network

Dynamic Source Routing

- To deliver a packet when the route is unknown, broadcast a *route discovery* packet
- In-range hosts re-broadcast discovery packet, attaching themselves as part of the route
- When destination is reached, a reply is sent along the reversed path.

Dynamic Source Routing

- If the source receives a reply, add to the *route cache*, and use route to send data.
- For simplicity, route cache entries expire in 3 seconds to prevent stale routes
- Real protocols have error handling that allows routes to be re-discovered only when necessary

Unreachable hosts

- How does the protocol terminate if a host is unreachable?
  - A TTL (time to live) field initialized to MAX_ROUTE_LENGTH is decremented on each re-broadcast
  - If TTL is 0 and host is not the destination, do not rebroadcast
- A host should not re-broadcast a discovery request it has broadcasted before
  - Route discovery IDs are assigned per packet to prevent redundant re-broadcasts.

Implementation

- Replace `network_send_pkt` with `miniroute_send_pkt`
- Update network handler
  - Recognize miniroute header
  - Routing control packets must be passed to routing thread
  - Data packets delivered to ports/socket if arrived at destination, otherwise routed to next-hop

Implementation

- Routing thread
  - State machine for handling and routing packets
  - Use `network_bcast_pkt` for broadcasts
- Route cache
  - SIZE_OF_ROUTE_CACHE entries
  - Invalidate after timeout (with or without alarms)
  - Aim for average access time of O(1) or O(logN)
- Table for node discovery packet IDs
  - Can assume some max lifetime of an ID

Instant Ad-Hoc Messaging

- Write an IM application using miniroute
    - Requires reading input from user
    - Add read.c, read.h, read_private.h
    - Include “read_private.h” in minithread.c
    - Add miniterm_initialize()
    - Use miniterm_read() to read data from the keyboard

Additional Changes

- In network.h
    - Set BCAST_ENABLED to 1
    - Set BCAST_ADDRESS
        - X.Y.Z.255 for most networks, where X,Y,Z are the first three octets of your IP
        - Try setting up an ad-hoc network between laptops
    - Set BCAST_TOPOLOGY_FILE
        - see project description for format

Additional Requirements

- At any host, there must be at most a **single** routing discover request for any destination at any one time:
  - Multiple threads should not trigger multiple requests for the same destination
  - Only one cache entry per destination
- Use reply packets with the latest information
- Use structures and data-types provided in miniroute.h
  - Routing should work across groups, but other protocols don’t have to

Additional Requirements

- Routing interoperability requires routing header entries to be in network order
    - Every short, int, long, must be translated to network order before being send, and translated to host order after being received.
    - See functions in network.c

For the ambitious

- Routing cache does not need to have a timeout.
  - Hosts detect broken links, send back errors.
  - Source host can purge cache entry and discover new route
  - Requires integrity of each hop to be verified
  - Hop-to-hop ACKs : very very inefficient
  - Eavesdropping : each host waits for next hop to forward.
    - Replace unicast hop to hop sends with broadcasts
    - Additional filtering in network handler

Localized Route Patching

- Hop that discovered the broken route perform a new route discovery
  - Patch route and continue routing packet
  - Route cache on both source/destination should eventually be updated

Aggressive Caching

- Every reply/request/data packet routed is an opportunity
- BUT- only some of the data is worth caching, and is different depending on whether it is a reply/request/data pkt

Redundant Routes

- By keeping additional routes, packets can be quickly re-routed if a route breaks
- Can be re-routed on error at source, or embedded in header to allow localized re-routing

Hybrid Proactive/Reactive Routing

- See Prof. Sirer’s SHARP

  http://www.cs.cornell.edu/courses/cs414/2004SP/papers/sharp.pdf