Project 5: Miniroute

Bernard Wong
What is Miniroute?

It is an ad-hoc networking layer

What is ad-hoc networking?

Ad-hoc networking allows multi-hop wireless communication without the need for infrastructure

Why would I want this?

- Removes infrastructure costs
- Allows quick deployment
- Possibly more reliable (no single point of failure)

Based on Dynamic Source Routing (DSR)

http://www.cs.cornell.edu/People/egs/615/johnson-dsr.pdf
What do you mean by routing?

- Packets that arrive at your machine may not necessarily be meant for you.
  - Previously, these packets would be dropped, now they should be routed to the destination.

How do I do this?

- Add a routing layer in BETWEEN the network layer and the transport layer.
- This means your minimsg/minisockets works on top of it and for the most part do not need to be modified.
How does the routing protocol work?

DSR is a reactive protocol

- When a host wants to deliver a packet to a destination host where the route to the destination is unknown, it will send a route discovery packet.
- A route discovery packet is broadcasted to any hosts that can hear it (within proximity of wireless signal).
- These hosts in turn will re-broadcast the discovery packet if it is not the destination, while attaching itself as part of the route.
- When the destination is reached, the collected routing path is reversed, and a reply message is sent back along this reversed path.
How does the routing protocol work?

- e.g. a route (which is stored in the routing header) may contain A->B->C where C is the destination, at which point host C will flip the route to C->B->A and send a reply back to host A
- If the source receives a reply, it will add this route into its route cache (as route discovery is expensive), and use this route to send the data
- Route cache expires in 3 seconds, to prevent stale cache entries (due to host movement)
- Route discovery has to be performed again when route expires
  - Is there a better way of doing this other than timeouts? (Yes!)
How does the routing protocol work?

How does this protocol terminate if the destination host cannot be reached?

- A TTL (time to live) field is decremented on each re-broadcast (TTL initialized to MAX_ROUTE_LENGTH)
- A host receiving a packet with TTL of 0 and is not the destination host should not re-broadcast it
- To prevent redundant re-broadcasts, route discovery ids are assigned per route discovery packet
- A host should not re-broadcast a discovery request that it had broadcasted before
  - This means each host needs to somehow keep track of what discovery packets its seen in the past
What needs to be implemented?

- In minimsg/minisockets, replace network_send_pkt with miniroute_send_pkt
- Network handler needs to be updated
  - Must recognize the miniroute header
  - Routing control packets must be passed off to routing thread
  - For data packets, if destination reached, simply deliver packet to ports/socket
  - Otherwise, again must deliver to routing thread
What needs to be implemented?

- **Routing thread needs to be created**
  - Contains state machine to handle and route packets
  - network_bcast_packet() provided for broadcasting

- **Route cache table needed**
  - Must contain SIZE_OF_ROUTE_CACHE entries
  - Route cache needs to be invalidated after timeout
    - This can be done with or without alarms
  - Should be somewhat efficient, as SIZE_OF_ROUTE_CACHE can potentially be large
    - Aim for average access time of O(1) or O(logN)
    - Think hash table, scatter table, tree
What needs to be implemented?

- A table containing recent node discovery packet ids that the host has heard is needed
  - In order to eliminate redundant broadcasts
- Write an Instant Messenger application using miniroute
  - Requires reading keyboard input from user
    - Add read.c, read.h and read_private.h
    - Include "read_private.h" to minithread.c
    - Add miniterm_initialize to minithread_system_initialization
    - Use miniterm_read() to read data from the keyboard
Additional changes

In network.h

- Set BCAST_ENABLED to 1
- Set BCAST_ADDRESS
  - 192.168.1.255 for ad-hoc network (see instructions for setting an ad-hoc network)
  - x.y.z.255 for CSUGLAB

For debugging purposes

- Set BCAST_TOPOLOGY_FILE
- Provide a topology file (see project description)
  - Allows testing without wireless
  - Use only in CSUGLAB (not for Tablets)
Tables?

- Yup, you’ll finally use them
  - Only real way to test an ad-hoc routing is through wireless
- Can compile and run tests like CSUGLAB desktops
- Setup tablet to use the wireless card
  - Set to ad-hoc mode
  - Specify an IP address for your tablets based on your group
    - 192.168.1.$\{GROUPID\}$
  - Set Subnet Mask to 255.255.255.0
  - Set Gateway to 192.168.1.254
Additional Requirements

- At any host, there must be at most a single routing discovery request for any destination at any one time
  - Multiple threads should not trigger multiple routing discovery requests for the same destination
  - Only one cache entry for each destination (unless...)
- Use the route reply packets with the latest information (use seq_no for this)
- Use the structures and data-types provided in miniroute.h
  - Allows everyone to participate in the routing (i.e. routing should work across groups)
  - However, minimsg/minisockets do not have to interoperate across groups
Additional Requirements

Furthermore, routing interoperability requires the routing header entries to be in network order

- Every short, int, long must be translated to network order before being sent, and translated to host order after being received
- Translation functions provided in network.c
For the ambitious...

- Lots of optimization opportunities

1) Routing cache does not need to have a timeout
   - If a host detects a broken link in the route path, it can send back an error message to the source host and the source host can purge the cache entry and re-perform discovery
   - Requires the integrity of each hop to be verified
   - Can be done via hop to hop acknowledgements
     - Very very inefficient
   - Can have each routing host eavesdrop, waiting for the next hop to forward the packet
     - Replace unicast hop to hop sends with broadcasts
     - Requires additional filtering work in the network handler
For the ambitious...

2) Localized route patching
   - Instead of sending an error message back to the source host if a particular hop to hop communication fails, have the hop that identified the route breakage to perform a new route discovery.
   - It can then patch the route, thus allowing it to continue routing the packet to the destination host.
   - Route cache on both source/destination should also be eventually updated.

3) Aggressive caching
   - There are lots of unexploited opportunities for caching.
   - Every reply/request/data packet that is routed through a host is an opportunity.
     - Have to be careful, only some of the route data is worth caching, and is different depending on whether it is a reply/request/data packet.
For the ambitious...

4) Redundant routes
   - By keeping additional routes to a destination, packets can be quickly re-routed if the primary route breaks
   - Re-routing using the redundant routes can be done when the source receives an error
   - Redundant routes can be embedded into the header (in some tree format), allowing localized re-routing

5) Hybrid proactive/reactive routing protocol
   - See Professor Sirer’s SHARP: