Project 5

Networking

Robert Escriva

Slide heritage: Bernard Wong

Cornell CS 4411, November 8, 2010

Announcements

- Project 4 due Wednesday at 11:59PM.
- Project 5 will be due November 21 at 11:59PM.
- Web page is being updated frequently; check for updates.

There are three kinds of death in this world. There's heart death, there's brain death, and there's being off the network.

Guy Almes

What is an “ad-hoc networking layer”?

What is an “ad-hoc networking layer”?

Ad-hoc networking enables wireless communication without the need for infrastructure

What is it useful for?

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

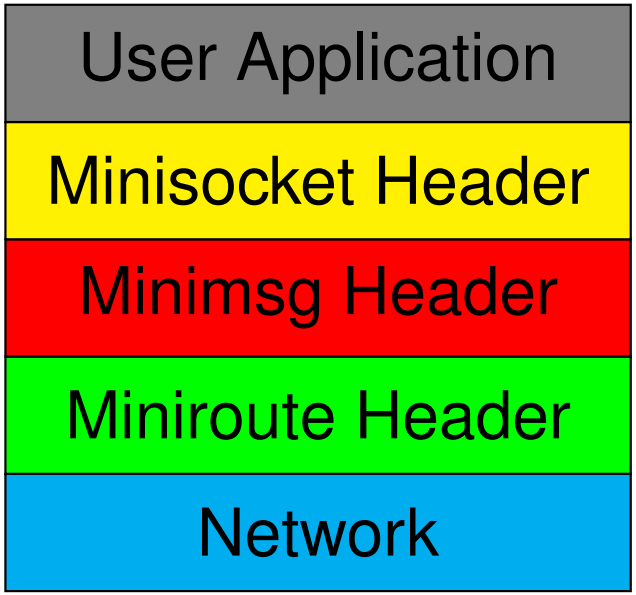
Based on Dynamic Source Routing.*

* 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 be meant for you.
- Packets not meant for you should be routed to their destination.
- Add a routing layer between the network and transport layer.
- Both minimsg and minisocket implementations should work on top of miniroute.

The new header in pictures



How does DSR work?

- DSR is a reactive protocol.
- When a host does not know the route a packet, it must discover the route.
  - It does so by sending a **route discover packet**.
- A route discover packet is broadcast to all hosts within proximity of a wireless signal.
- Hosts will re-broadcast discovery packets if they are not the destination.
  - The host will add itself to the constructed route.
- The destination will send a **route reply packet** along the reverse route.

Routing protocol (common case)

- If a route contains A $\rightarrow$ B $\rightarrow$ C where A is the source, C is the destination, C will flip the route to C $\rightarrow$ B $\rightarrow$ A to send a reply to host A.
- Upon receiving a reply, it will add this route into its route cache, and use it to send data.
- Route cache entries expire in 3 seconds to prevent stale cache entries.
- Route discovery must be performed again upon route expiration.
  - Is there a better way to do this?

Routing protocol (failure conditions)

- How does the protocol terminate?
  - By tracking a time-to-live (`ttl`) value.
- What should the `ttl` be initialized to?
- What happens when `ttl` = 0?
- How do we prevent redundant re-broadcasts?
  - We can re-use a trick you've seen before.
- How do we prevent loops?

What do you need to implement?

- Convert calls to `network_send_pkt` to `miniroute_send_pkt`.
- Update the network handler.
  - Interpret the miniroute header.
  - Handle routing control packets.
  - Deliver packet as usual if the destination has been reached.

Routing Cache

- The cache must be able to hold `SIZE_OF_ROUTE_CACHE` entries.
- Old items are invalidated after timeout.
  - Alarms may be used, but it can be done without.
- This should be somewhat efficient, as you may increase `SIZE_OF_ROUTE_CACHE` to be large.
  - Aim for $O(1)$ or $O(\log n)$ bound on operations.
  - Think in terms of hash tables, scatter tables, and trees.

Additional implementation requirements

- Need to track recently seen discovery packets.
  - Eliminate redundant broadcasts.
- Write an Instant Messenger application that runs on miniroute.
  - Read input from the user (look at `read*`).
  - Add `miniterm_initialize` to your system initialize functions.
  - `miniterm_read` will let you read from the keyboard.

Broadcast information

- Set `BCAST_ENABLED` to 1.
- Set `BCAST_ADDRESS`:
  - 192.168.1.255 for ad-hoc network (see instructions for setting up an ad-hoc network).
  - x.y.z.255 for CSUGLAB.
- When debugging:
  - Set `BCAST_TOPOLOGY_FILE`.
  - Provide a topology file (see project description).
    - Test without wireless.
  - Use only in CSUGLAB.

Additional requirements

- Only put one routing discover request per destination on the network at any one time.
  - Multiple threads should not trigger multiple routing discovery requests for the same destination.
- Use the latest route reply packet (use the `seq_no` for this).
- Use the structures and data-types provided in `miniroute.h`:
  - Allows all groups to route an arbitrary group's packets.
  - It's unlikely `minimsg` and `minisockets` will be groupwise-compatible.

Additional requirements

- Routing interoperability requires headers be in network byte order.
- Use the same endianness conversion you used in project 4.

Feeling ambitious?

Remove the routing cache timeout:

- Instead, detect broken links and perform re-perform discovery.
- Requires verifying that hops work.
- Hop-to-hop acknowledgements are very[†] inefficient.
- Take advantage of broadcasting to see when the next host forwards the packet.
  - Faster (less sends).
  - Requires more work.

[†] very

Feeling more ambitious?

Localized route patching.

- When hop-to-hop communication fails, have the hop that detects the failure perform a new route discovery.
- Patch the route on the failed packet to allow it to route successfully.
- Source and destination should be updated to reflect new route.

Feeling even more ambitious?

Cache aggressively

- There are lots of opportunities to cache more.
- Every packet presents the chance to update the cache.
  - Just because you can does not mean you should.
  - Some data is not worth caching.

Feeling *even more* ambitious?

Redundant routes

- Keep multiple routes to the same destination in cache.
- When the source receives an error, the backup route may be used.
- Consider how you could extend the header to encode multiple routes.

If you eat and breath this stuff

- Hybrid proactive/reactive routing protocols
- See Professor Sirer's SHARP[‡]

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

Concluding thoughts

- Have some fun with this project.
- It's much less work than P4, and much more fun too.
- Come see the TAs in office hours[§].

[§] some of the TAs get lonely