Assignment 3: Network

Sanzheng Qiao
Department of Computing and Software

March 28, 2013

Due: April 8, Monday, 23:59:59

In the final assignment, we provide you with some low level network communications facilities; you will build a nicer abstraction on top of those.

Each separate Nachos (emulated as a UNIX process) is a node in the network; a network (emulated via UNIX sockets) provides the communication medium between these nodes. You can test out the basic network functionality by running ‘nachos -N -m 0’ and ‘nachos -N -m 1’ simultaneously (preferably, on different terminals or in different windows). [Note: this test case requires a working implementation of locks and condition variables, but nothing else.]

The reading for this assignment is Roadmap 8 and the Nachos files listed there.

The post office provides a more convenient abstraction than the raw network; you are to continue this layering process – at each level, the software removes one physical constraint and replaces it with an abstraction. Thus, reliable messages can be built on top of an unreliable service, large messages can be built on top of fixed-length messages, etc.

This assignment has the following item:

1. Implement reliable messages with limited size packets. Currently the communications facility provides unreliable transmission of limited size packets. You must implement a protocol to fix this. Specifically, you modify PostOffice::Send to provide reliability.

   For implementation, you can either modify network/post.* or introduce a new layer and modify network/post.*. Also, you can either modify `MailHeader` structure defined in network/post.h or introduce a new header structure. Do not change machine/network.*.

Warning: you are strongly advised to do a very careful paper design of your protocol before starting to implement them. It is very easy to design protocols (particularly for reliability) that do not work, and it is typically very difficult to find and fix protocol bugs after the protocols have been implemented.

In this assignment, you may assume that a message can fit in one packet.

The Nachos network emulation can be made to randomly drop packets by using the command line argument ‘-n #’; the number, between 0 and 1, reflects the likelihood that a packet will be successfully delivered. To simplify matters, you may assume that packet delivery is ”fail-safe”; packets may be dropped, but if a packet is delivered, its contents have not been corrupted. (In practice, a hardware or software checksum would be needed to detect this kind of error.)