README
CS143

Daniel Erenrich
Emil Ibrishimov
Nathan Watson
Robert Shimizu

Overview

Netproj is a project for CS 143, FA10-11 at Caltech by Daniel Erenrich, Emil Ibrishimov, Nathan Watson and Robert Shimizu. The goal is to create a realistic high-level network simulation that provides enough functionality to check the accuracy of theoretical modeling of networks.

Code and Javadoc

The code is in a public repository at https://github.com/emilibrishimov/CS-143-Netproj/

Code documentation (via Javadoc) is available at http://www.cs.caltech.edu/~emil/netproj/

You can check out the code using

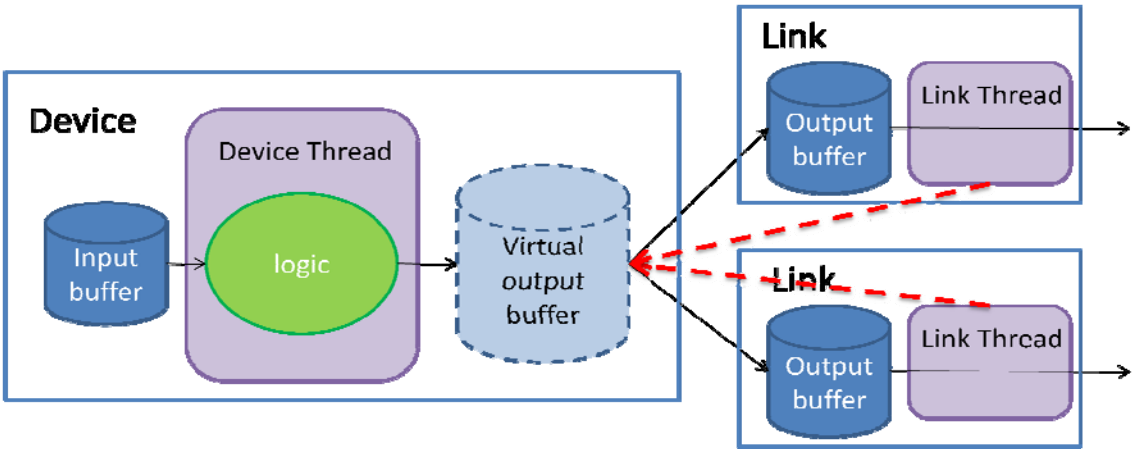
```
git clone git://github.com/emilibrishimov/CS-143-Netproj.git
```

Contributions

- Daniel Erenrich
  - TCP Tahoe
  - FTPHost
  - SimulatorSpecification (XML reader)
  - Terminal
  - ProbFlowSender (generates flows according to a probability distribution)
- Emil Ibrishimov
  - Skeleton (Device, Packet, Link)
  - FastTCP
  - Host
  - Pinger
- Nathan Watson
  - Router
  - Bellman-Ford Router
  - NetworkWatcher (Stats collector)
- Robert Shimizu
  - Pinger
  - UDPSender

Architecture outline

Our aim is to create the simplest possible design that supports the full set of features for specifying networks and devices. We tried to keep our code as modular and as flexible as possible.



The project is divided in four main modules:

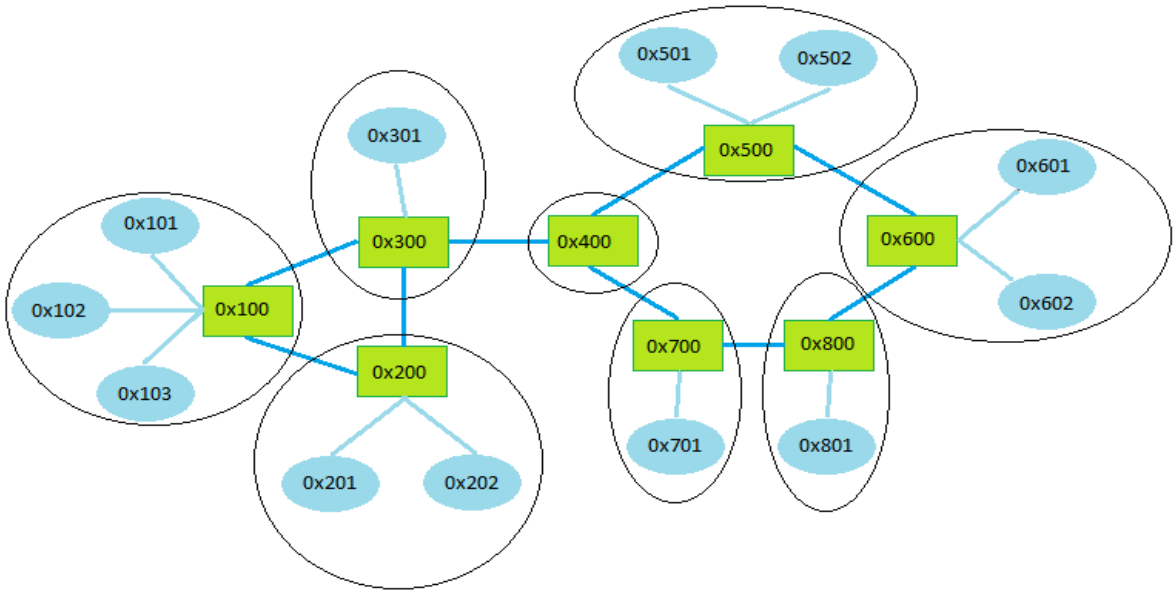
- Skeleton – base classes for the simulation.
  - Packet – base IPv4 packet with source, destination and TTL
  - Link – class that transfers information between devices (can be set up to behave as an Ethernet cable, Wi-Fi connection, etc.)
  - Device – base class for a device on the network
- Controller – user interface, network builder and stats collector
  - NetworkWatcher – collects stats from all around the network and logs them in GnuPlot compatible files
  - SimulatorSpecification – parses XML files with topology specification
  - Terminal – real time terminal for controlling the simulation
- Hosts – different host implementations
  - Host – base host class. Provides a nice terminal.
  - Pinger – a host that can ping and reply to pings
  - ProbFlowSender – a host that sends packets at probabilistic intervals
  - TcpHost – base TCP host. Handles sending and keeping track of timeout
  - FastTcpHost – a TcpHost that sends files using FastTcp congestion control
  - TcpTahoeHost – TcpHost that sends files using Tahoe congestion control
  - UDPSender – Sends information without ACKs
- Routers – routing algorithms implementation
  - Router – base class for routing. Stores the routing table in a O(1) Trie
  - BFRouter – uses Bellman-Ford for setting the entries in the routing table.

Network topology & Routing

Currently, the routers expect the following address spaces:

- Local subnet (last 8 bits of an address) – hosts attached to a router
- Network (bits 8-24) – a set of local subnets. Routing in a network is entirely handled by routers
- Big Network – a set of networks. Routing should be specified manually.

The following is a visualization of a network, where routers are in green, hosts are in blue and local subnets are represented as ellipses:



XML specification

You can specify network topology in XML files. Here is an example of a simple network consisting of 2 subnets of 2 hosts each:

```
<simulator>
  <bfrouter inputBuffSize="100000" outputBuffSize="100000" address="0x100" />
  <bfrouter inputBuffSize="100000" outputBuffSize="100000" address="0x200" />
  <FastTCP address="0x101" inputBuffSize="100000" outputBuffSize="100000" />
  <FastTCP address="0x102" inputBuffSize="100000" outputBuffSize="100000" />
  <FastTCP address="0x201" inputBuffSize="100000" outputBuffSize="100000" />
  <FastTCP address="0x202" inputBuffSize="100000" outputBuffSize="100000" />
  <hostlink host="0x101" router="0x100" bps="1000000" delayms="0" />
  <hostlink host="0x102" router="0x100" bps="1000000" delayms="0" />
  <hostlink host="0x201" router="0x200" bps="1000000" delayms="0" />
  <hostlink host="0x202" router="0x200" bps="1000000" delayms="0" />
  <link bps="1000000" delayms="5">
    <connect address="0x100" />
    <connect address="0x200" />
  </link>
</simulator>
```

Terminal

The terminal provides you with a list of commands. The most important (except start) is terminal. You can use terminal to enter any device and control it.

Here is an example terminal session:

```
Enter a command (or see help)
> terminal 0x101
101$ help
help - prints this message
ifconfig - prints network information about this device
ping <address> - pings <address>
101$ ping 0x301
Ping reply in 67ms.
101$ ifconfig
IPv4 address: 101
gate 0: -100-101-
101$ exit
Bye.
>
```