CS144 Lab 1
Simple Router

Antonin

September 23rd 2013

Assignment supported by
AWS in Education Grant Award

With some slides from TY, Fall 2012

Overview

- You are going to write a “simplified” router
  - Given a static network topology
  - Given a static routing table
  - You are responsible for writing the logic to handle incoming Ethernet frames:
    - Forward it
    - Generate ICMP messages
    - Drop it
    - And more...

But how to do it???

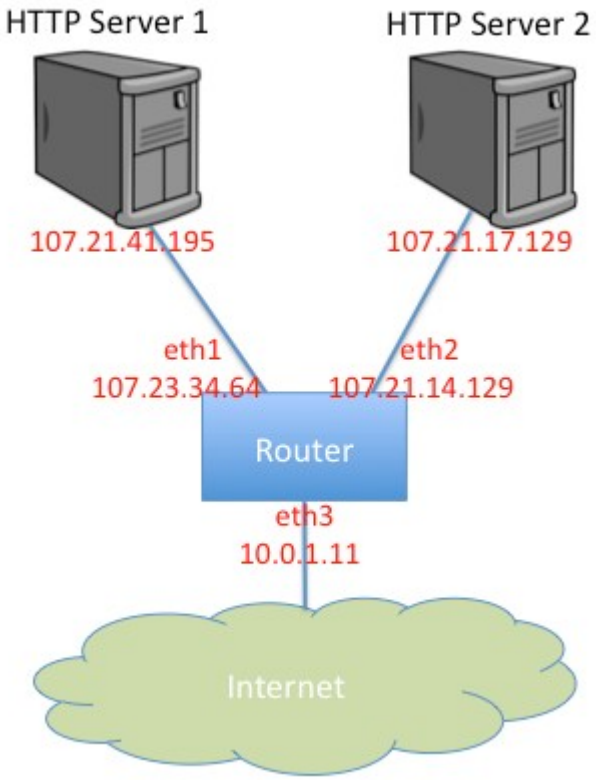
- Where will my routing logic run?
- Where will the traffic come from?
- How will I test my code?

- No hardware router :)
- Network topology emulated with Mininet: your router connects 2 servers to the Internet
- Your router will handle real traffic
- The topology is emulated on an EC2 instance, thanks to the $50 AWS credit from Amazon!

Outline

- The EC2 environment
  - Emulated topology
  - What's going on behind the scene?
  - Detailed instructions to be found on the website
- The required functionality
  - Rough decision flow chart

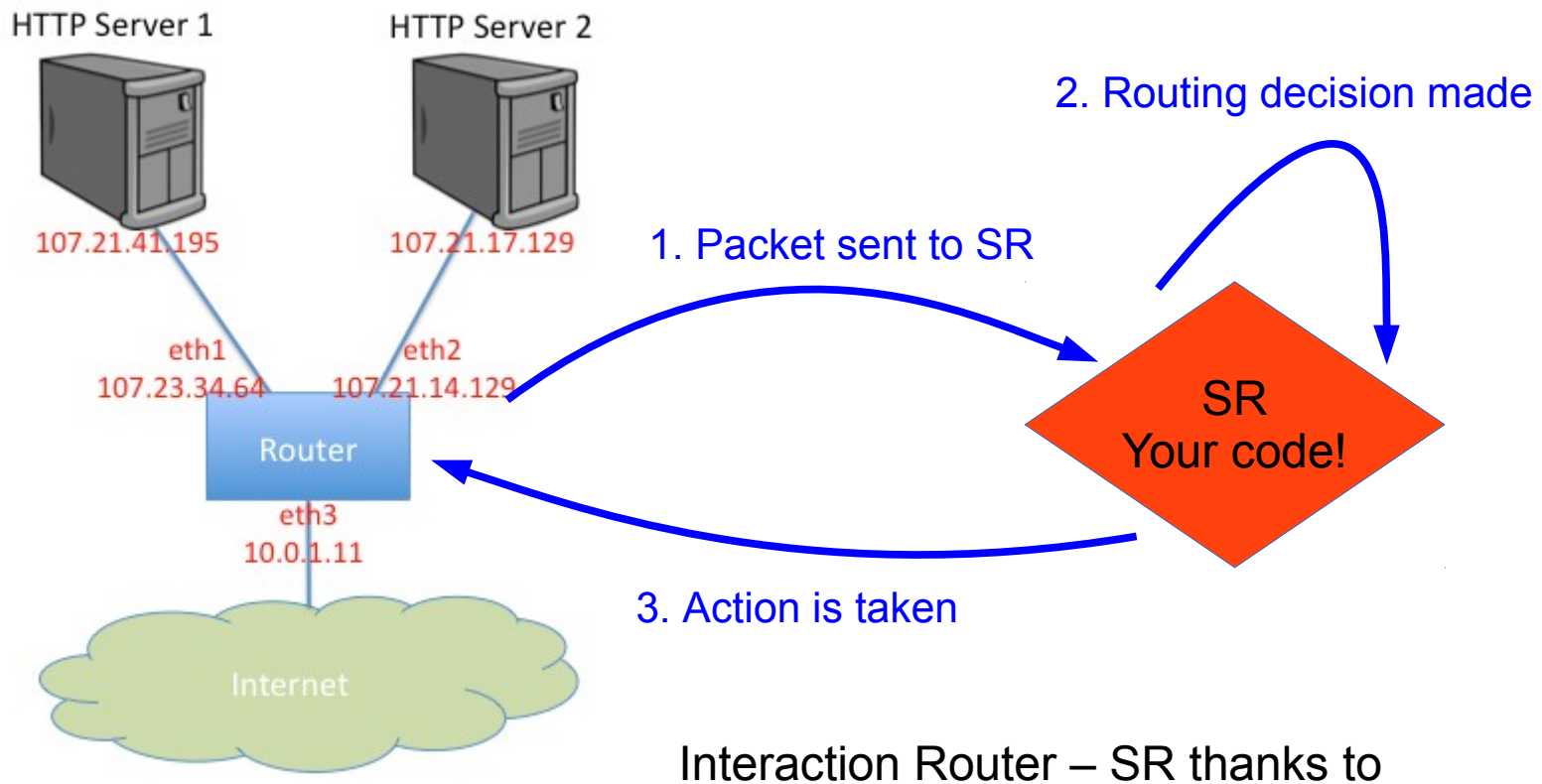
The EC2 environment



Destination Network	Gateway	Network mask	Intf
0.0.0.0	10.0.1.1	0.0.0.0	eth3
107.21.41.195	107.21.41.195	255.255.255.255	eth1
107.21.17.129	107.21.17.129	255.255.255.255	eth2

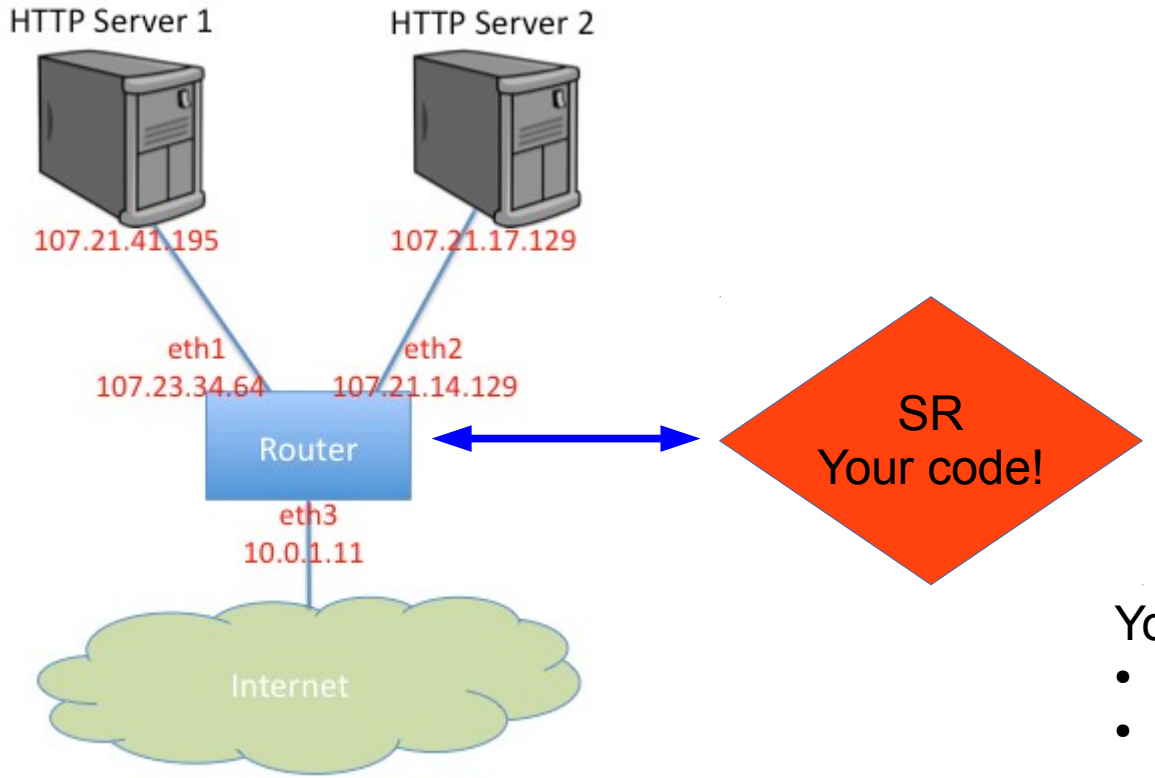
Network topology emulated with Mininet

The EC2 environment



Interaction Router – SR thanks to POX and Openflow

The EC2 environment



Ping 107.21.17.129

Your routing decision:

- Look at the routing table
- Figure out on which interface to forward the packet
- Make necessary changes to the packet

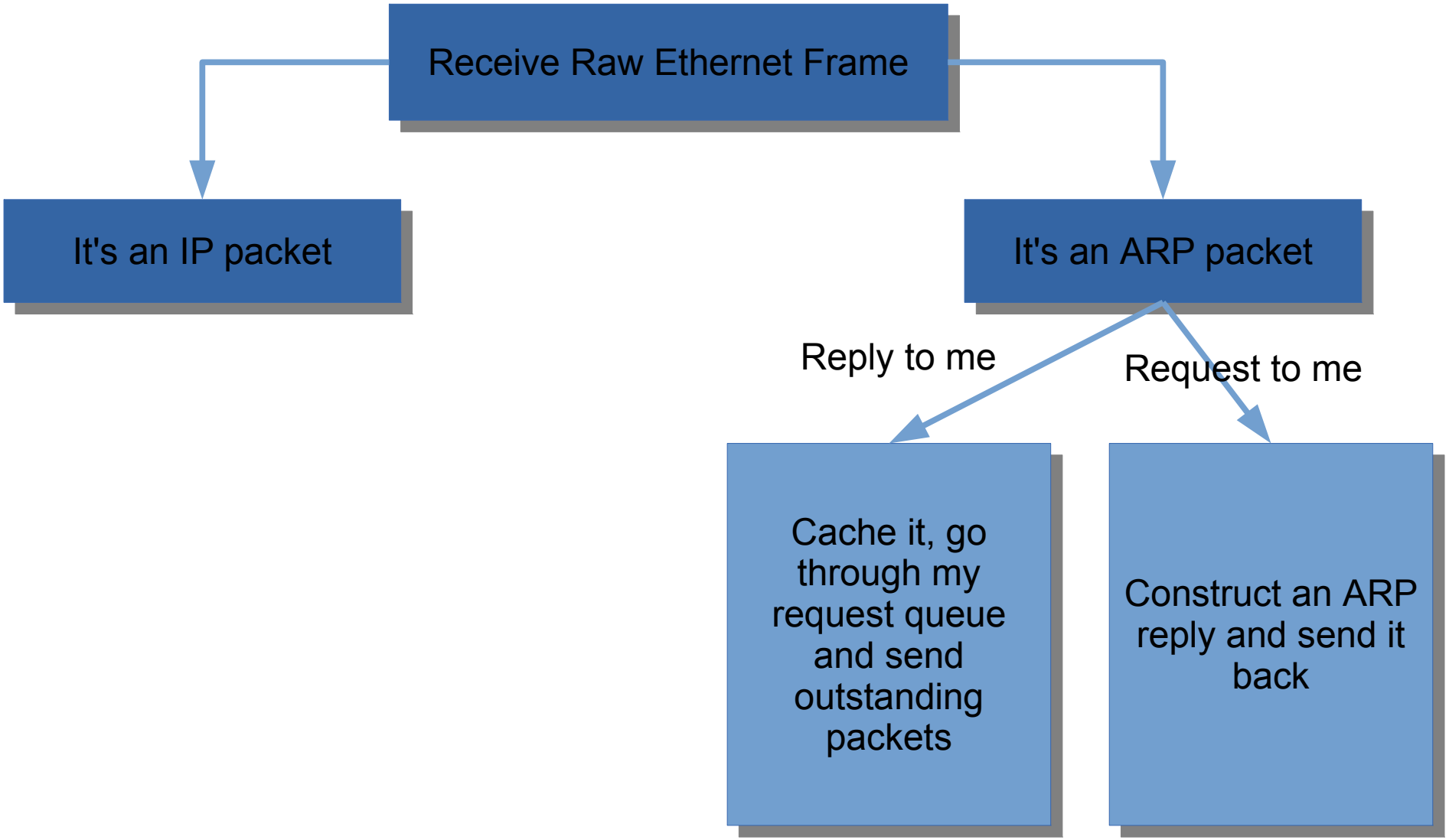
EC2 Summary

- $50 coupon will soon be available (wait for our email)
- Create an account on EC2
  - Need credit / debit card number + phone number
- EC2 setup script + start script + ssh script
- To run the lab, start (in this order):
  - Mininet script to emulate topology
  - POX script to setup link between the router node and your routing logic
  - Your own routing logic (SR)
- Use your credit wisely
  - $50 = ~250 hours of up time
  - Some will be needed for the last lab
  - Stop the instance when not using it (with stop.sh, it will not terminate the instance so your data will be preserved)

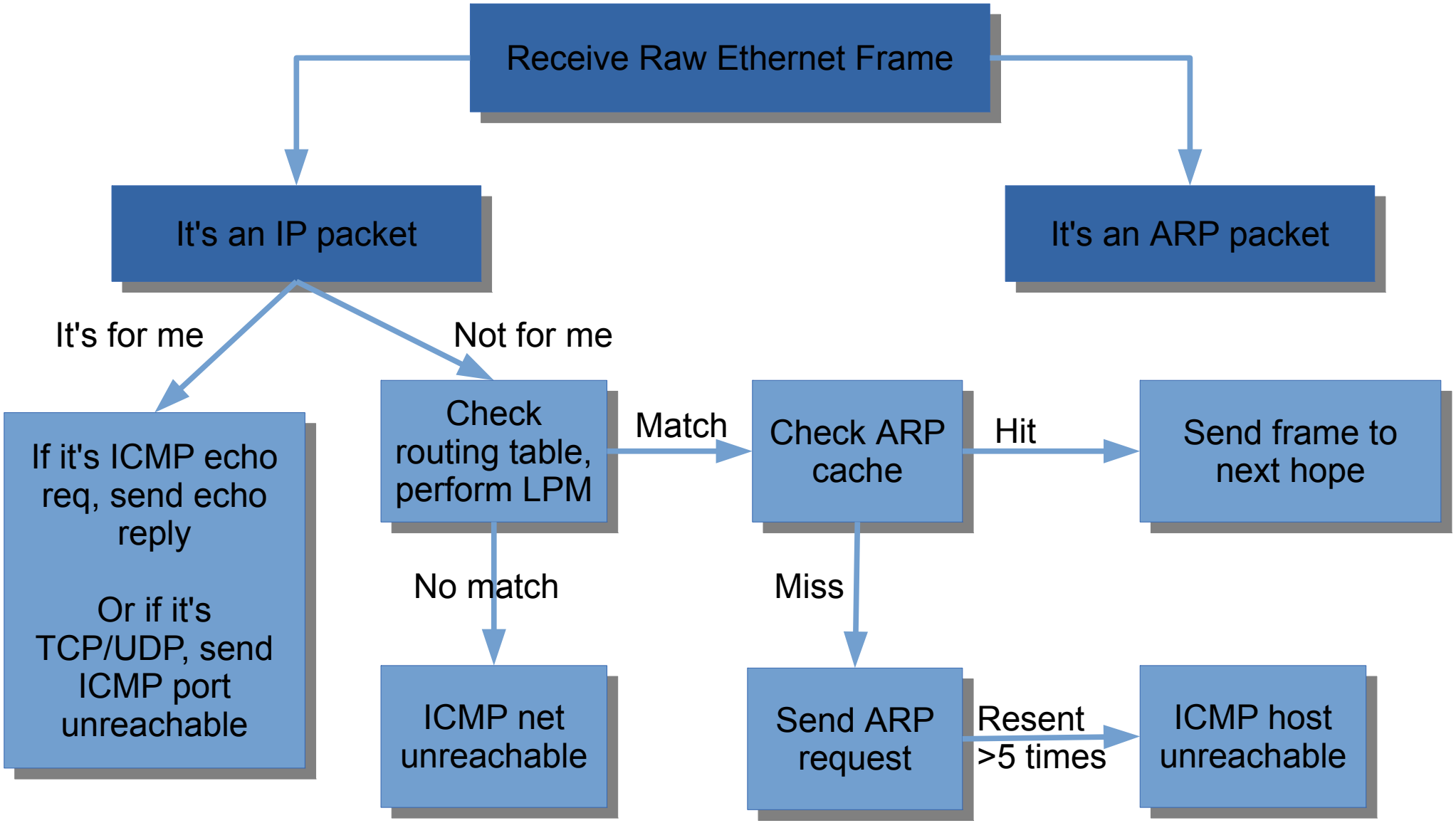
What your routing logic needs to do

- Route Ethernet frames between the Internet (the myth cluster) and the HTTP servers
- Handle ARP request and replies
- Handle traceroutes
  - Generate TTL Exceeds Message
- Handle TCP/UDP packets sent to one of the routers' interfaces
  - Generate ICMP Port Unreachable
- Respond to ICMP echo requests
- Maintain an ARP cache
- See webpage for full requirements

A rough flow chart



A rough flow chart



A rough flow chart

- Many things missing from this chart
  - Checksums, TTLs
- Read the instructions carefully
- 500+ lines of code, so start early
- First milestone: October 2nd at **noon**
  - Protocol parsing: read incoming frames, parse IP header, decide on the right course of action
  - No functionality testing, just a code review
- Final submission: October 14th at **noon**

How to test your code

- Test connectivity with ping from any machine on the Internet
- Traceroute will not work well outside of Mininet:
  - Use Mininet CLI
  - mininet> server1 traceroute -n server2
- HTTP requests with wget, curl
- Don't forget to test “error” cases

Some advice

- Be thorough in your testing
- Do not hesitate to change the routing table (what about an incorrect routing table?)
- Be careful when implementing Longest Prefix Match
- Don't get mixed up with endianness: Linux is little endian, network big endian
  - Try to put the calls to hton, ntoh in a single place
- Write good quality code
  - Do not hardcode constants, avoid code duplication...

Things that may be useful

- Mininet console, which supports tcpdump, ping, traceroute (apt-get install traceroute on instance)
- Debug functions in sr_utils.c
  - print_hdrs, print_addr_ip_int
- GDB/Valgrind

Start reading!

http://www.stanford.edu/class/cs144/labs-fall2013/lab1/lab1.html