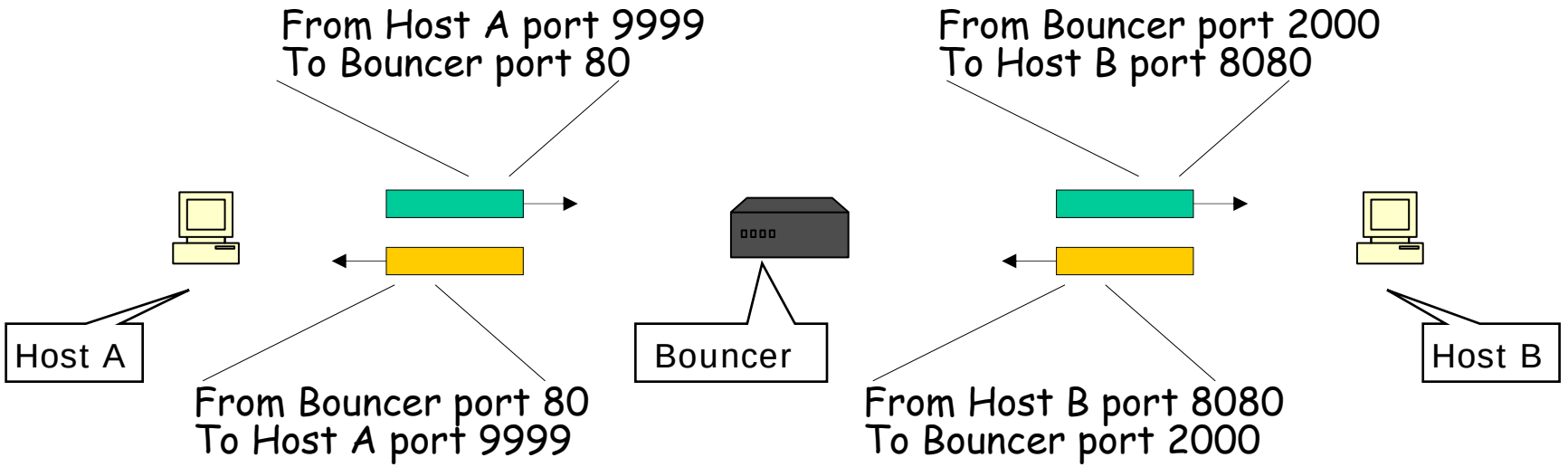
Assignment 4: Bouncer

Peter Sjödin

Goal

- Design and implement an IP-level packet relay
- Learn about advanced socket programming
- Learn about IP/TCP/UDP packet processing
- Preferred language is C
  - OK in Java as well

What is a Bouncer?

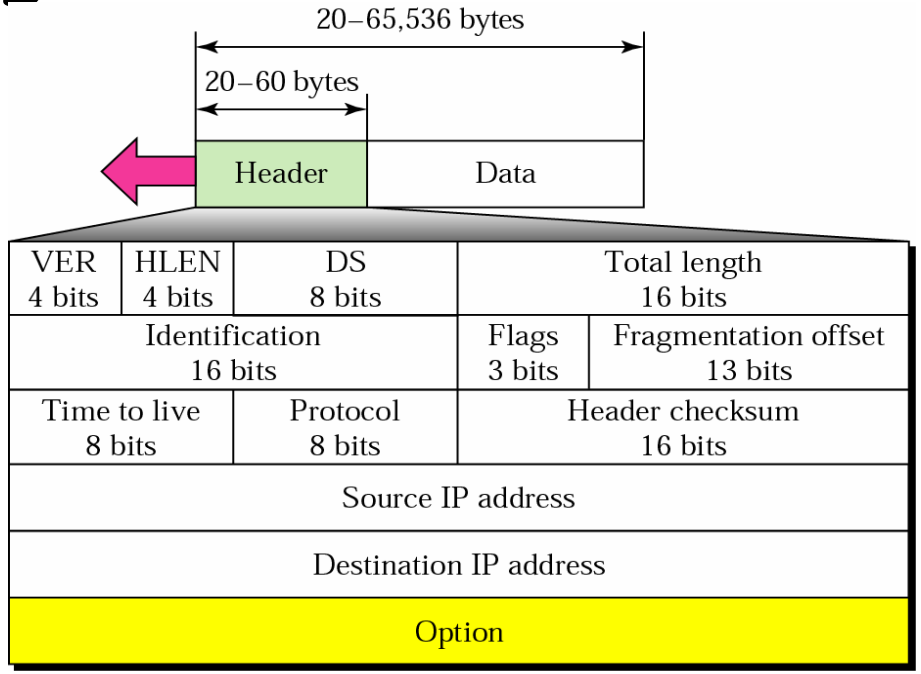


- Redirects and rewrites packets
- It appears to host A and B as if they are communicating with Bouncer
  - But they are communicating with each other
  - Bouncer is relaying traffic
- Examples
  - Traffic redirection behind firewall
  - Topology hiding
  - Load balancing
    - IP Sprayers

The Assignment

- Write a working bouncer that bounces packet for:
  - Ping (BASIC)
  - A TCP connection (MEDIUM)
  - An FTP file transfer (ADVANCED)
    - FTP proxy, or Application Layer Gateway
    - Active mode – PORT command

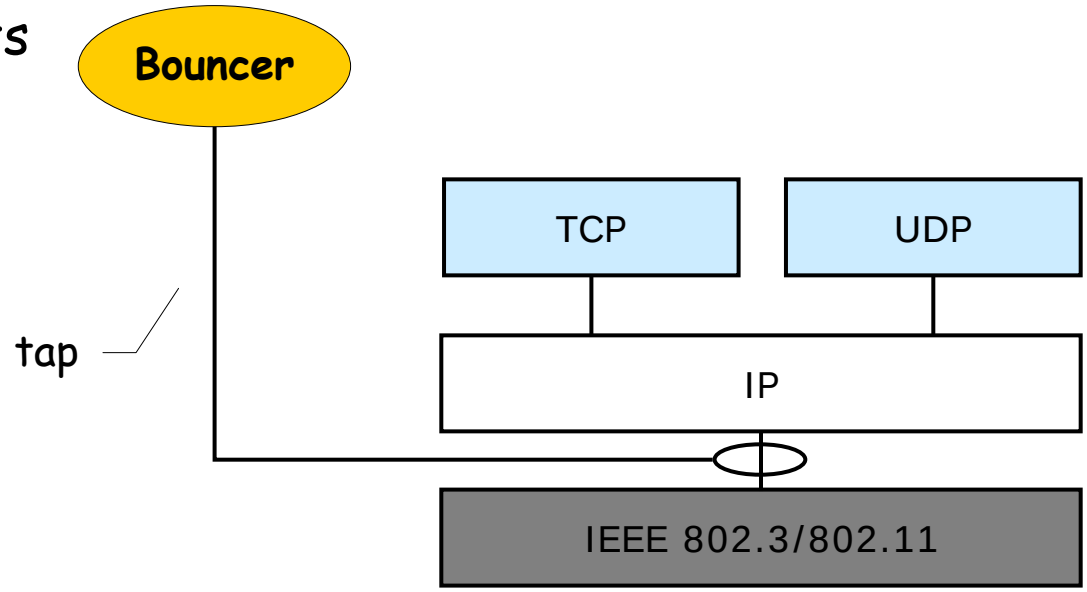
Things to Consider: Checksums



- The Bouncer modifies header fields
- This means that it needs to recalculate header checksums
- OK to use checksum function from someone else
  - Plenty of examples in textbooks and on the Web

Things to Consider: Raw Packets

- The bouncer is an IP stack parallel with the kernel's
- How do we get unprocessed "raw" packets to the bouncer application?
- Answer: use a "tap" (packet monitor interface)
  - libpcap
  - PF_PACKET sockets



pcap

- High level interface to packet capturing
- libpcap packet capture library
  - Available in most *nix distributions
  - "man 3 pcap"
- Does the filtering for you
- Preferred way of capturing in a portable way
- There is a Windows version
  - WinPcap
  - www.winpcap.org

PF_PACKET Sockets

```
int socket(PF_PACKET, SOCK_RAW, 0);
int socket(PF_PACKET, SOCK_DGRAM, 0);
```

- "man 7 packet"
- Linux specific
- Send and receive raw packets
  - Of any type
- You get everything from the interface
    - need to filter out relevant packets yourself
- SOCK_RAW with link level header (Ethernet header)
- SOCK_DGRAM without link level header

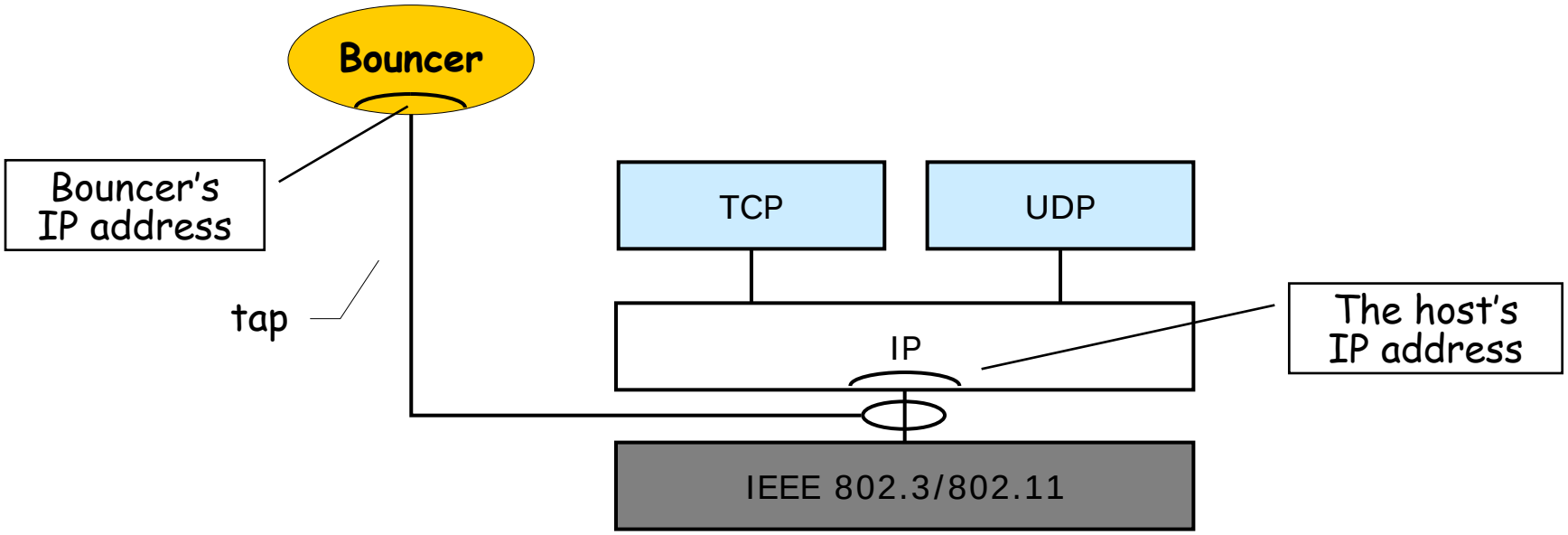
Raw PF_INET Sockets

```
int socket(PF_INET, SOCK_RAW, protocol);
```

- "man 7 raw"
- Receives all packets of a given IP protocol type
  - Hence, cannot be used to capture all kinds of packets
- Can be used for sending IP packets, though
- IP header checksum is filled in by kernel (but not TCP)
  - Linux

Things to Consider: IP Addresses

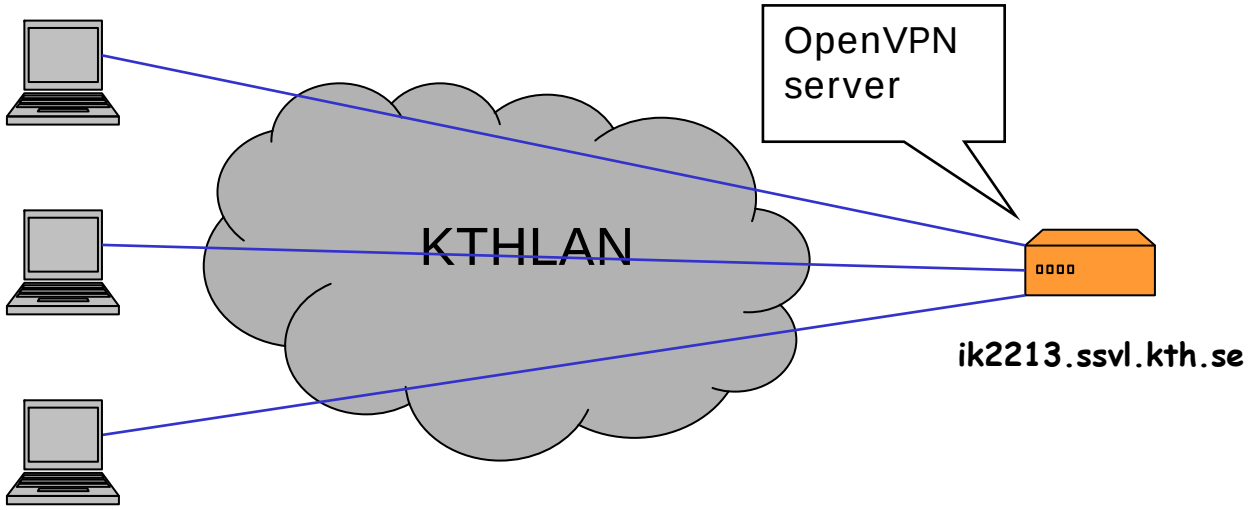
- The bouncer needs to execute in a host (obviously…)
  - Let's call it "the Bouncer host"…
- But how do we prevent the packets from being processed in the Bouncer host's IP stack?
  1) Use a separate IP address for the bouncer
  2) Make sure not to run the host with "IP forwarding" on (why?)



BON

- Bouncer Overlay Network
- This is a networking level exercise
  - Strange IP packets may appear…
- We do not want to interfere with the traffic on the KTH production network
- So we have set up a virtual private network on top of KTHLAN
  - OpenVPN

OpenVPN Network



- SSL VPN, www.openvpn.net
- Virtual layer 2 network (Ethernet)
- UDP tunnels over KTHLAN as "cables"
- OpenVPN server as Ethernet switch
  - Built-in DHCP server
- A local network interface (Ethernet) as tunnel endpoint
  - "tap0" or similar
- Instructions and config files on the Web

Things to Consider: ARP

- But how do we get the packets to the bouncer?
- Who should respond to ARP requests?
- Implementing ARP in Bouncer is one alternative – but outside the scope of this assignment

Things to Consider: ARP (2)

- Preferred approach (but non-transparent)
- What we want: for packets to the Bouncer, the IP destination and MAC destination should be different hosts
- This happens all the time on networks!
  - For packets going via a router
- Consider the Bouncer host as the gateway (next hop router) to reach the Bouncer
- Grab any IP address **not on the subnet** as the address of the Bouncer
  - An address outside the BON, that is
  - Use a 192.168.0.0/16 address, for example
- Configure the clients with static "host" routes for the Bouncer
  - Next hop is the Bouncer host's IP address

ARP (alternative approach)

- Transparent approach (but non-preferred)
- Assign an address from the subnet to the Bouncer
  - (We have lots of addresses)
- Configure the bouncer host reply to ARP requests for the Bouncer's address
  - Similar to "proxy ARP", a standard Linux functionality
  - But proxy ARP only works under certain conditions
  - Those conditions do not match well with the Bouncer
- Instead, you could use an additional fake ARP (farp) daemon on the Bouncer host
  - OK, since we have our own closed network
    - You can make great chaos with a farp daemon…

The Application

```
bouncer [-i <interface>] <listen_ip>:<listen_port> <server_ip>:<server_port>
bouncer [-i <interface>] <listen_ip> <server_ip>
```

- Example command syntax
  - You may chose your own syntax
- The Bouncer should allow the following to be given as arguments:
  - For Ping bouncing, local (listen) and remote (server) IP addresses
  - For TCP, local and remote port numbers
- Optionally, you may want to specify the network interface to tap
  - The "-i <interface>" argument
  - But don't bounce on any other interface than the BON tap!

Requirements

- BASIC: ICMP Echo bouncer
  - With correct validation of IP headers
- MEDIUM: As for BASIC, plus TCP bouncer
  - With correct validation!
- ADVANCED: As for MEDIUM, plus FTP bouncer
  - Support for FTP active mode
    - PORT command
  - With correct validation!

Notes and Hints

- This is high-density code
  - It won't be a lot of code
  - But you need to get it right
- Read the manual pages and documentation carefully
- Make sure you understand the difference between host and network byte order
- Make sure you understand how (and why) you need to set up your system
- Use Wireshark (or tcpdump)
- May we do this in Java or C# or…?
  - Yes, but then you're on your own

Deadline

- Sunday May 25
- Code package with instructions for compiling and running
  - Include scripts that perform any necessary configurations
- Short written report

Good Luck!