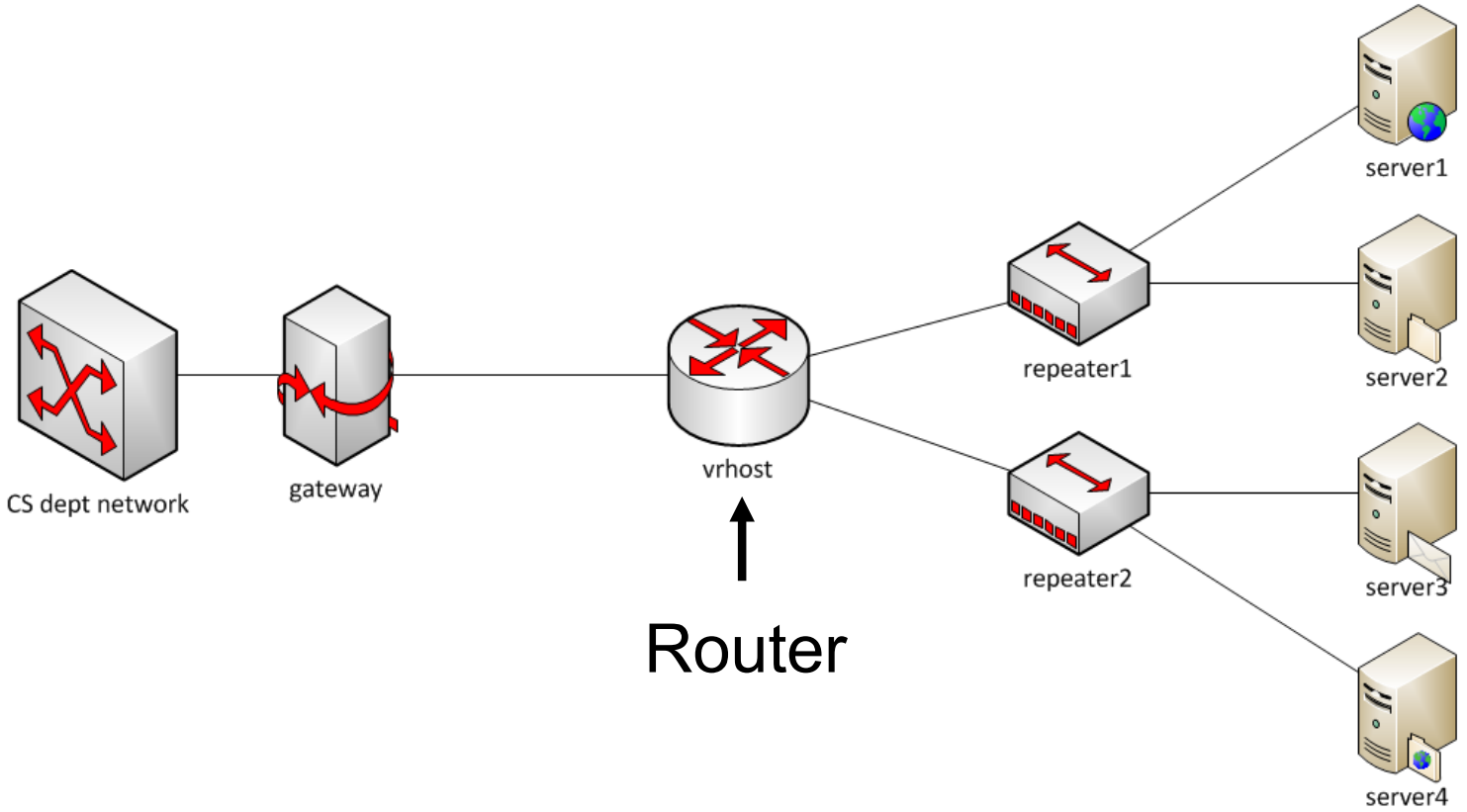
CSC 525:
Principles of Computer Networks

Build a simple router



- Implement ARP, basic IP forwarding, and ICMP at the router
- Will be tested with real traffic: http download and ping.

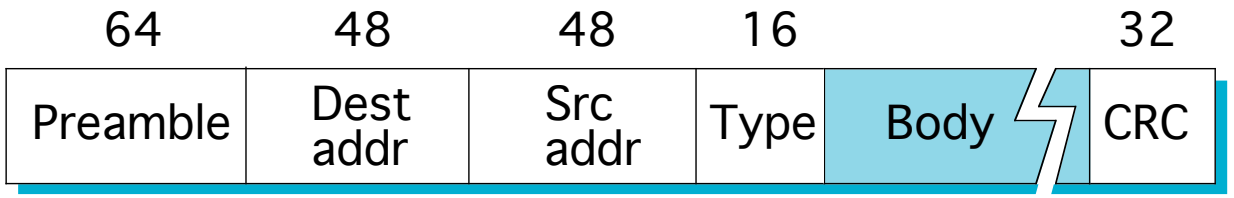
What you'll be given

- Network topology
  - One per student. Don't use others'.
  - Only accessible from within the CS department.
- Stub code
  - Receive packets: sr_handlepacket( )
  - Send packets: sr_send_packet( )
  - Basic data structures of protocol headers, interfaces, etc.
  - Your job is essentially to implement the packet processing logic.

What does a packet look like?

- Protocol stack:
  - Ethernet Header
  - IP or ARP header
  - TCP, UDP or ICMP header
  - Application payload
- How to tell what's next?
  - Each protocol header has a field to identify the type of next header.
- **How to parse the packet header?**
  - Cast to a data structure of the protocol header, use pointer to access specific header fields.
  - E.g., (struct sr_ethernet_hdr *) pkt; where pkt is (uint8_t *) that points to the beginning of a packet.
  - Header structures are in the header files.

Ethernet



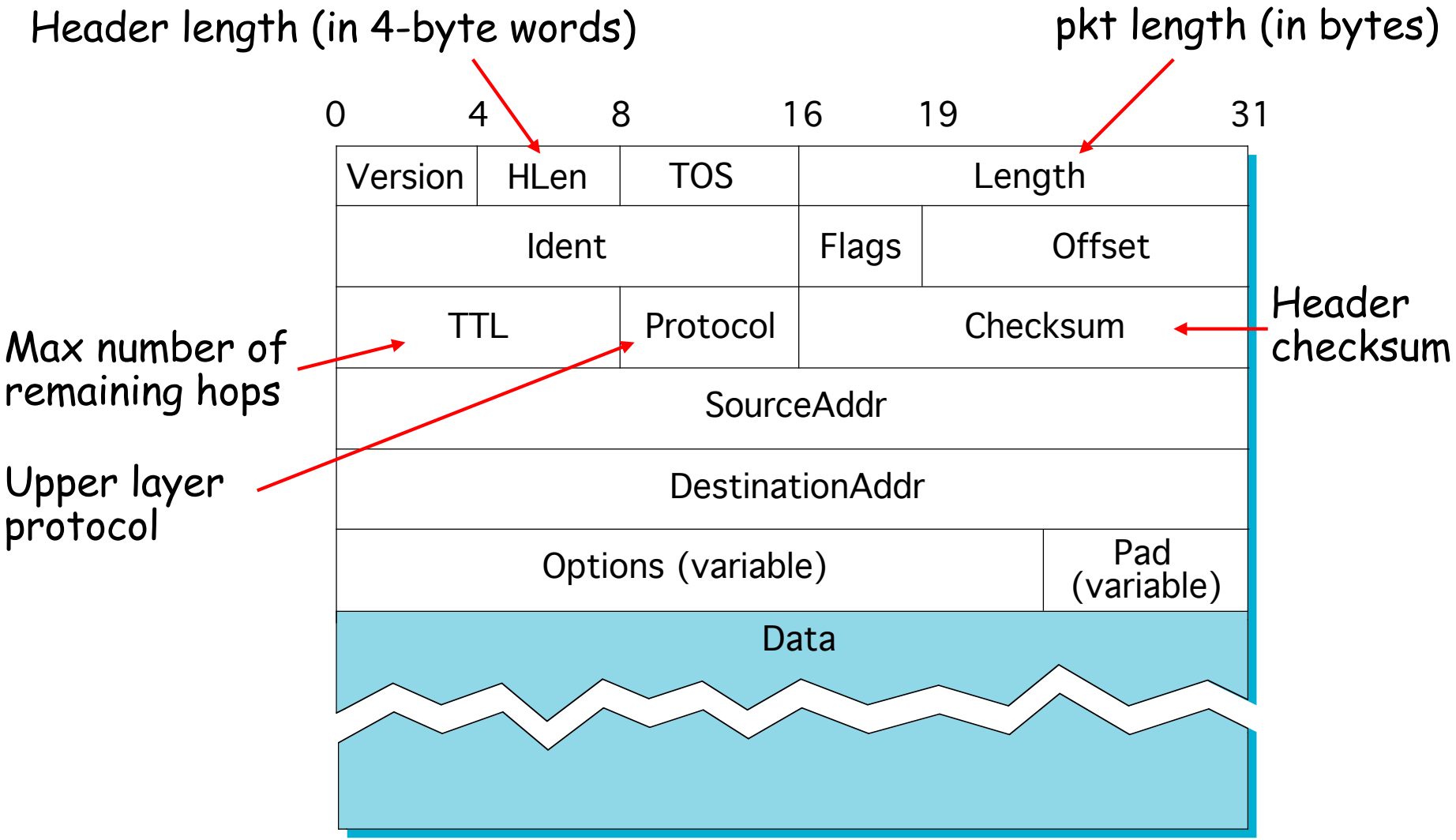
- Ignore preamble and CRC; they're handled by hardware, not passed to router software.
- Addresses
  - 48-bit unicast address assigned to each interface
  - example: `8:0:2b:e4:b1:2`
  - Broadcast address: all `1`s, i.e., ff:ff:ff:ff:ff:ff
- Type: what next protocol header is (e.g., IP or ARP)
- Body: min 46 bytes, max 1500 bytes

Accessing header fields

```
struct sr_ethernet_hdr
{
#ifndef ETHER_ADDR_LEN
#define ETHER_ADDR_LEN 6
#endif
    uint8_t  ether_dhost[ETHER_ADDR_LEN];
    uint8_t  ether_shost[ETHER_ADDR_LEN];
    uint16_t ether_type;
} __attribute__ ((packed)) ;
```

```
uint8_t *p;
struct sr_ethernet_hdr *eth;
struct ip *iphdr
void sr_handlepacket(sr, p, len, iface)
{
   eth = (struct sr_ethernet_hdr *) p;
   eth->ether_type is IP;
   iphdr = (struct ip*)(p + sizeof(struct sr_ethernet_hdr));
   ...
}
```

IP Packet Header



The basic header is 20 bytes long

Basic IP packet processing

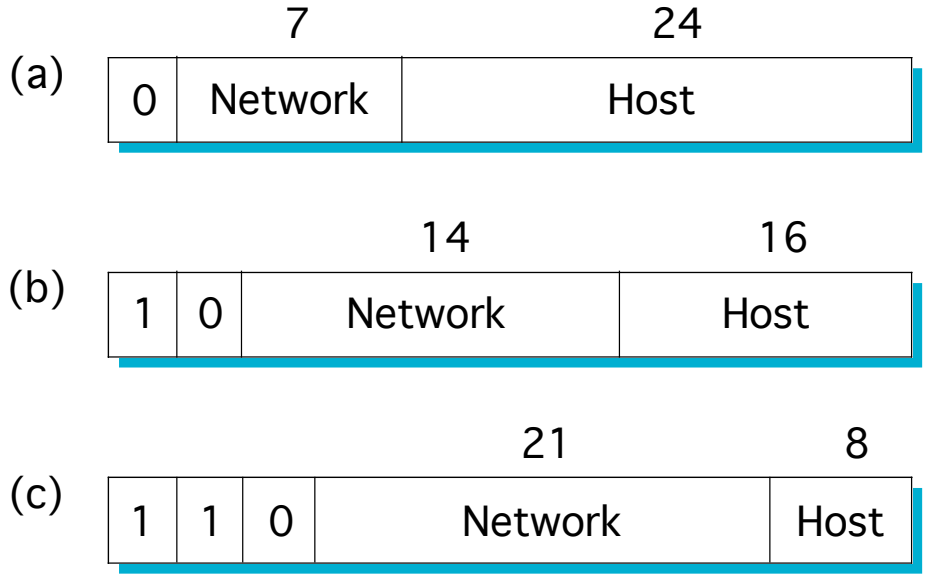
- Verify that version == 4.
- Verify the checksum:
  - The algorithm is on the page 95 of the Peterson & Davie book.
  - Feed the IP header to the algorithm, result should be 0. If not, discard the packet.
- Decrement TTL by 1,
  - if the result TTL == 0, discard the packet
  - Otherwise update the checksum
    - Set checksum field to be 0, recalculate the checksum over the header, and record the result in the field.

Where to forward the packet

- Every packet has an **destination IP** address
- Every host has a routing table specifying the forwarding rules.
- Use the destination address to look up the routing table, find a matching entry in the table.
- The matching entry indicates the nexthop (IP and interface).
  - If more than one match, take the non-default one.
- Find out the MAC address of the nexthop
  - Either from local ARP cache or by sending an ARP request
- Send the packet to the nexthop.

How many bits for network part

- Original IP design: class-based address



1.0.0.0 to
127.255.255.255

128.0.0.0 to
191.255.255.255

192.0.0.0 to
223.255.255.255

The problem: class-B addresses are in great demand.

- Now it's called classless addressing.
  - Subnetting and classless addresses.
  - Subnet mask and prefix length, e.g., 255.255.0.0 = /16
  - Address AND Mask = Network

Take a look at the interface

```
lectura 136 > ifconfig
eth0      Link encap:Ethernet  HWaddr 00:E0:81:31:73:50
          inet addr:192.12.69.186  Bcast:192.12.69.255  Mask:255.255.255.0
```

- Mask indicates how long the network part is
  - addr & mask = network prefix
- Broadcast address has all ones in the host part.

Routing Table Lookup

network	nexthop	mask	interface
172.24.74.64	0.0.0.0	255.255.255.248	eth1
172.24.74.80	0.0.0.0	255.255.255.248	eth2
0.0.0.0	172.24.74.17	0.0.0.0	eth0

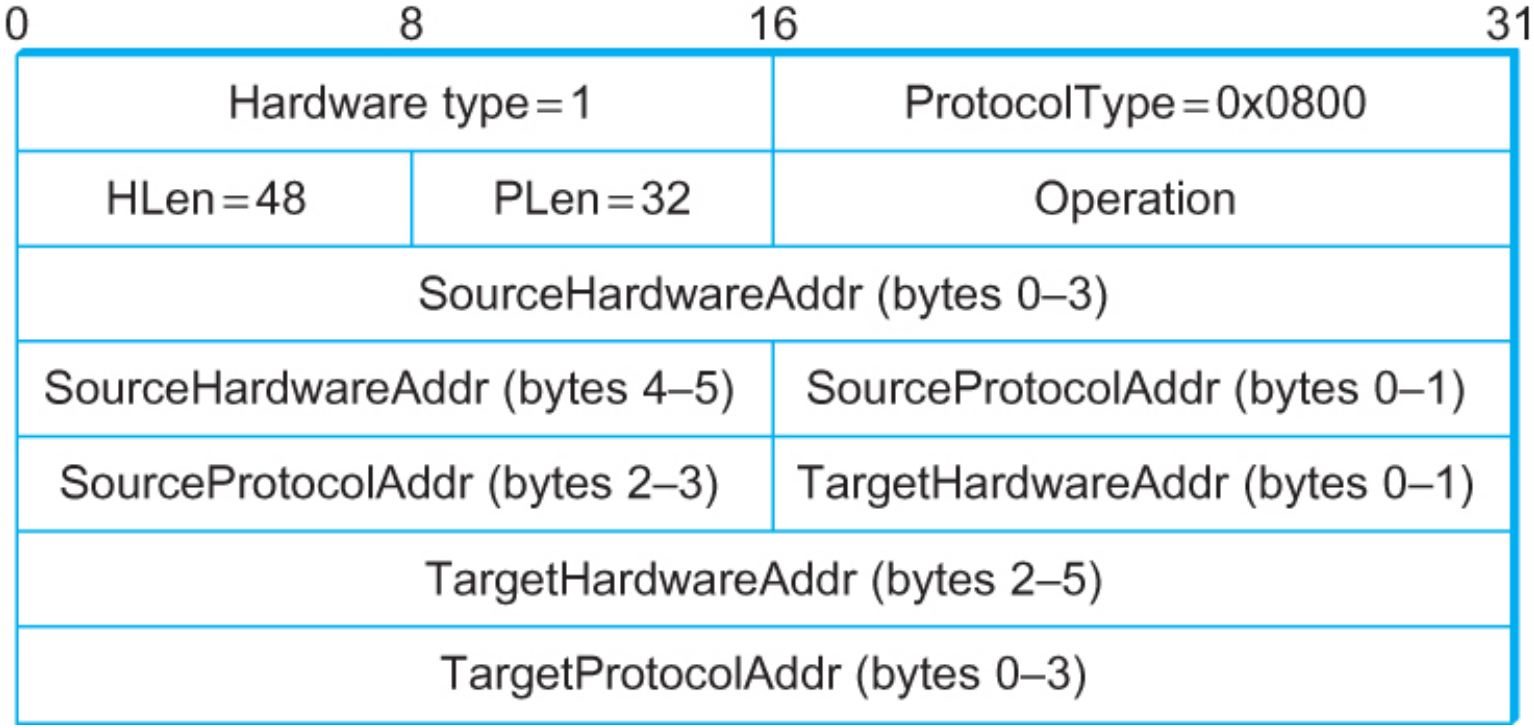
- Lookup match if (destination & mask) == (network && mask)
- 0.0.0.0 as the network means that this is the default route. It is used if no other entry matches.
- 0.0.0.0 as the nexthop means that the nexthop IP is the same as the destination IP of the incoming packet.
- In the first project the routing table is automatically loaded from a file.
- Information of your router's interface is stored in a linked list.

How to find out MAC address?

- *As a packet travels in the network, its IP src/dst addresses don't change, but its MAC src/dst addresses change at every hop.*

- ARP: learn the nexthop's MAC address
  - Map an IP to a MAC address
  - broadcast request, the target machine responds with its MAC address.
  - Cache the reply in a table, refresh if see more traffic, delete if stale.

ARP Packet Format



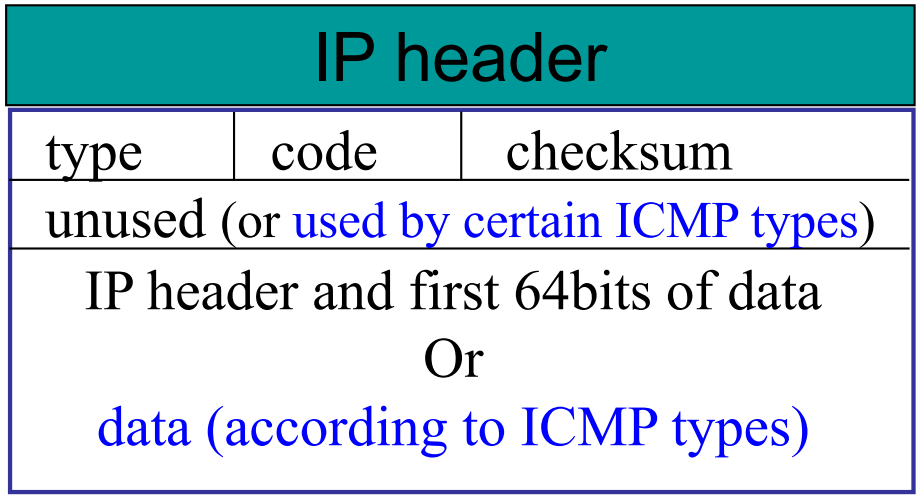
- Ethernet header : ARP header : Data.
- HardwareType: type of physical network (e.g., Ethernet)
- ProtocolType: type of higher layer protocol (e.g., IP)
- HLEN & PLEN: length of physical and protocol addresses
- Operation: request or response
- Source/Target Physical/Protocol addresses

ARP Details

- The request is broadcasted onto the local link only.
- The reply is usually unicast to the requester, but can also be broadcasted.
- A host can learn IP/MAC mapping from incoming packets without generating separate ARP requests.
- An entry in ARP table will time out if the entry is not refreshed by incoming traffic after a while.
- ARP is only for within the local network, i.e., not crossing any IP router.

Internet Control Message Protocol

- ICMP: used by hosts & routers for network-level error reporting and diagnosis
  - unreachable host, network, port, protocol, echo request/reply
- ICMP msgs are carried in IP packets
- ICMP message format



Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

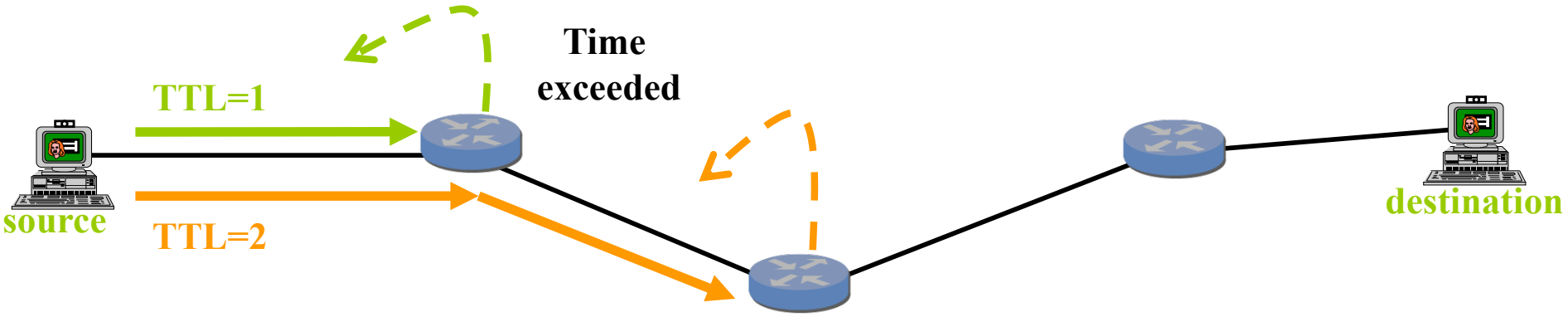
Ping

- ICMP Echo request and reply

```
bobcat2 177 >ping peterpan2
PING peterpan2 (10.0.0.104): 56 data bytes
64 bytes from 10.0.0.104: icmp_seq=0 ttl=64 time=6.789 ms
64 bytes from 10.0.0.104: icmp_seq=1 ttl=64 time=2.765 ms
64 bytes from 10.0.0.104: icmp_seq=2 ttl=64 time=4.368 ms
64 bytes from 10.0.0.104: icmp_seq=3 ttl=64 time=6.713 ms
64 bytes from 10.0.0.104: icmp_seq=4 ttl=64 time=3.716 ms
64 bytes from 10.0.0.104: icmp_seq=5 ttl=64 time=2.825 ms
^C
--- peterpan2 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.765/4.529/6.789/1.662 ms
```

Traceroute

- Measuring the forwarding path
- Time-To-Live field in IP packet header
  - Source sends a packet with a TTL of *n*
  - Each router along the path decrements the TTL
  - "TTL exceeded" sent when TTL reaches *0*
- Traceroute exploits this behavior by sending a sequence of packets with increasing initial TTL.



Send packets with TTL=1, 2, 3, … and record source of "TTL exceeded" mesg

Traceroute Example

```
bobcat2 117 > traceroute www.google.com
 1  * * *
 2  tcsn-dsl-gw04-196.tcsn.qwest.net (168.103.240.196) 52.629 ms
 3  * tcsn-agw1.inet.qwest.net (168.103.240.93)  52.172 ms
 4  tus-core-01.inet.qwest.net (205.171.149.33)  51.762 ms
 5  svl-core-01.inet.qwest.net (67.14.12.6)  74.450 ms
 6  pax-edge-01.inet.qwest.net (205.171.214.30)  76.415 ms
 7  * * *
 8  209.85.130.6 (209.85.130.6)  76.325 ms
 9  66.249.94.226 (66.249.94.226)  77.372 ms
10  216.239.49.66 (216.239.49.66)  76.611 ms
11  mc-in-f99.google.com (66.102.7.99)  76.144 ms
```

The Implementation

- ARP implementation
  - Be able to respond to ARP request, send ARP query when needed, and refresh ARP cache by incoming traffic.
- IP forwarding: given an incoming packet
  - Decrement TTL and update checksum.
  - Lookup the routing table to find the nexthop's IP.
  - Invoke ARP to find out the nexthop's MAC address.
  - Send the packet to the nexthop.
- ICMP
  - Ping
  - Traceroute (only for students who were in CS425 Fall'17)

Four steps

- Test the stub code
- Implement ARP
- Implement IP forwarding
- Implement ICMP ping and traceroute