Lecture 26: Networks

Brennon Brimhall

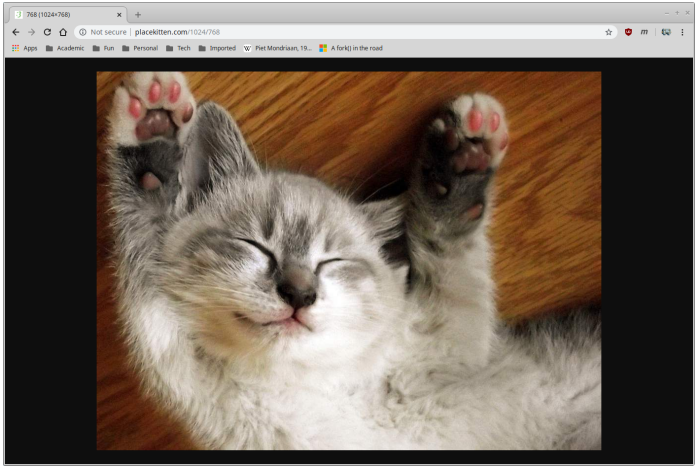
9 July 2025

Using a web browser

Type a URL into a web browser:

`http://placekitten.com/1024/768`

The internet of cats



Nice! (But how did that actually work?)

Networks

Networks

Network: allow communication between computers

Access remote data

Share information

Hard to overstate importance of networking: *everything* can communicate over the Internet now (laptops, phones, cars, refrigerators, ...)

Network interface

To connect to a network, a computing device needs a *network interface*

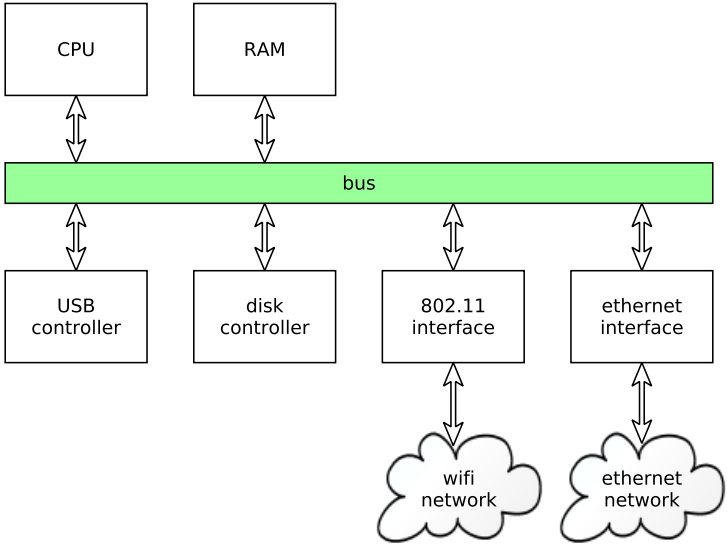
- Wired: ethernet, Infiniband (for high-performance applications)
- Wireless: 802.11 (wifi), cellular modem

To the computing device (the "host"), the network interface is just a peripheral device

- Much like a disk controller, USB controller, etc.

OS can request to send data out to the network
Network interface device notifies host CPU when data arrives from the network (possibly by raising a hardware interrupt)

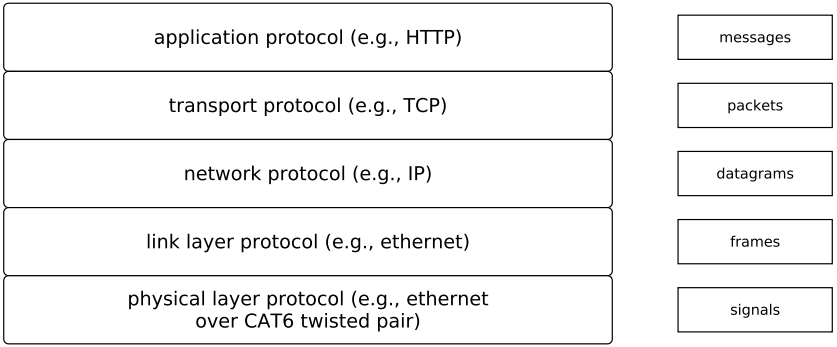
Network interface example



Protocol stack

In addition to network interface hardware, a *protocol stack* is needed to allow network applications to communicate over the attached network interface(s)

"Protocol stack": so called because network protocols are layered



Issues

Some important issues to consider:

- How are differing network technologies interfaced to each other?
- How are devices and systems identified on the network?
- How is data routed to the correct destination?
- What APIs do network applications use to communicate?

We'll cover all of these (at least briefly)

Network security

Ideal of networking is to provide access to information and computing resources from anywhere

But...connecting a computing device to the network potentially exposes it to malicious actors

Issue: controlling access

- Permit only authorized agents access to data and services

When implementing and deploying networked systems and applications, we must think *very* carefully about

- what the security requirements are, and
- whether the system meets them

TCP/IP

TCP/IP

TCP/IP: a suite of *internetworking* protocols

- "internetworking" = connecting lots of physical networks together, including when they use different technologies or protocols

Two versions: IPv4 and IPv6

- IPv4: 32 bit addresses (not enough of these!), widely deployed
- IPv6: 128 bit addresses, not as widely deployed (but significant adoption in mobile networks)

Ubiquitous: if you're using a network, you're using TCP/IP

Scale of global TCP/IP internet is immense (*billions* of communicating devices)

IP

IP = Internet Protocol

This is the underlying *network protocol* in the TCP/IP protocol suite

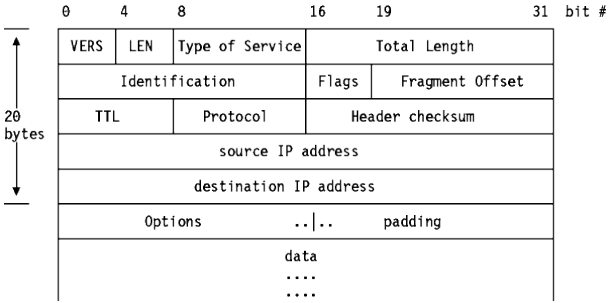
Ultimately, all data is sent and received using *IP datagrams*: fixed-size packets of data sent and received using IP addresses to indicate the source and destination

Transport protocols (such as TCP and UDP) are layered on top of IP

- E.g., a TCP connection consists of IP datagrams containing TCP data

IP is an *unreliable* protocol: when a datagram is sent, it might not reach the recipient (we'll see why in a bit)

An IP datagram



[Image source: http://www.danzig.us/tcp-ip-lab/ibm-tutorial/3376c23.html]

Details:

- Consists of *header* followed by *data*
- May be fragmented and reassembled
- Protocol field indicates which transport protocol is being used

TCP

TCP: Transmission Control Protocol

A *connection* protocol layed on IP (value in Protocol field is 6)

TCP allows the creation of virtual connections between peer systems on network

A connection is a bidirectional data stream (each peer can send data to the other)

Data is guaranteed to be delivered in the order sent

Connection can be closed (analogy: hanging up when phone call ends)

TCP is a *reliable* protocol: if any data is lost en route, it is automatically resent

- Much cleverness is required to make this work!

UDP

UDP: User Datagram Protocol

A *datagram* protocol layed on IP (value in Protocol field is 17)

Not connection-oriented: data could be received in any order, no fixed duration of conversation (more analogous to sending a letter than talking on the phone)

Unreliable: data sent might not be received

Used in applications where minimizing latency is important and data loss can be tolerated

Routing: idealized

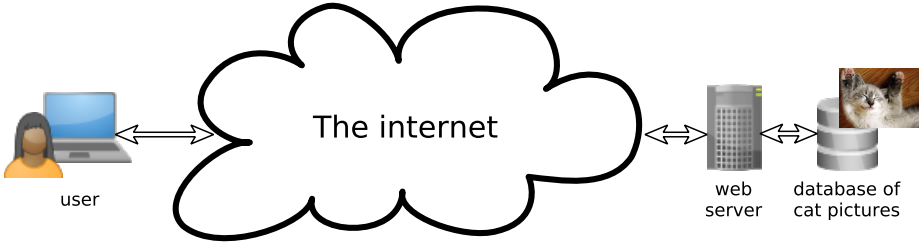
Routing: How does data get to its destination?

Idealized view:

Routing: idealized

Routing: How does data get to its destination?

Idealized view:



Routing: the reality

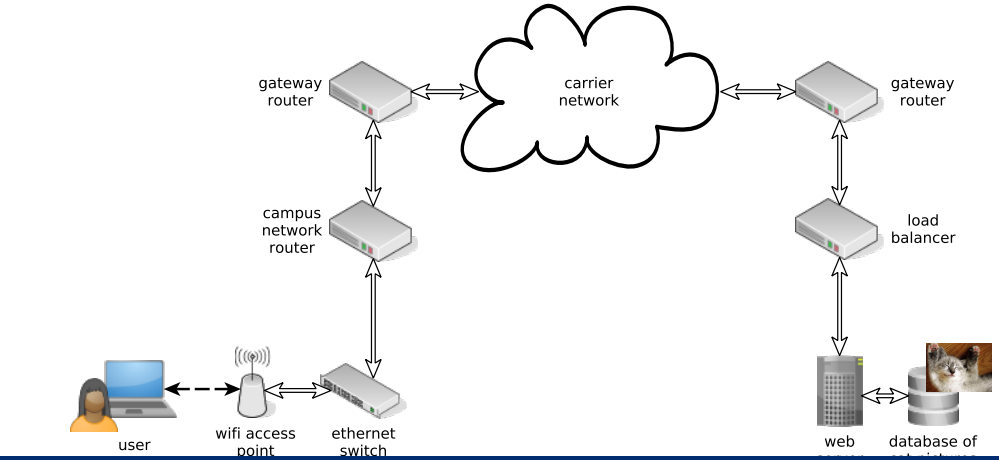
Routing: How does data get to its destination?

Slightly more realistic view:

Routing: the reality

Routing: How does data get to its destination?

Slightly more realistic view:



Addressing

Two kinds of address:

- Network address: address of a network interface within the overall internet (e.g.: IPv4 address)
- Hardware address: a hardware-level address of a network interface (e.g.: ethernet MAC address)

Addressing

Two kinds of address:

- Network address: address of a network interface within the overall internet (e.g.: IPv4 address)
- Hardware address: a hardware-level address of a network interface (e.g.: ethernet MAC address)

Network address is used to make routing decisions at the scale of the overall internet

- Network address conveys information about the network on which the interface can be found
- A *router* makes routing decisions based on a network address

Addressing

Two kinds of address:

- Network address: address of a network interface within the overall internet (e.g.: IPv4 address)
- Hardware address: a hardware-level address of a network interface (e.g.: ethernet MAC address)

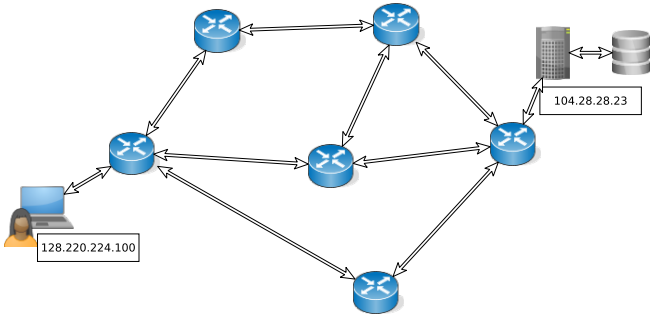
Network address is used to make routing decisions at the scale of the overall internet

- Network address conveys information about the network on which the interface can be found
- A *router* makes routing decisions based on a network address

Hardware address is used to deliver a data packet to a destination within the local network

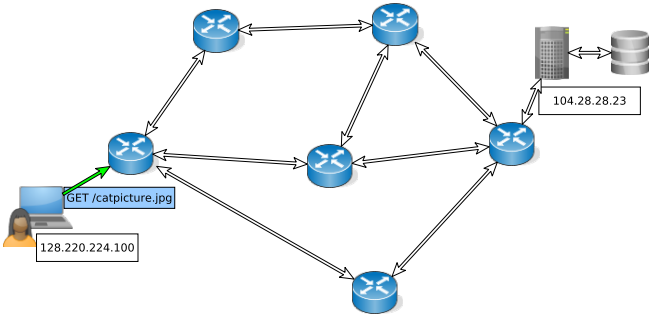
- A *switch* makes routing decisions based on a hardware address

Routing



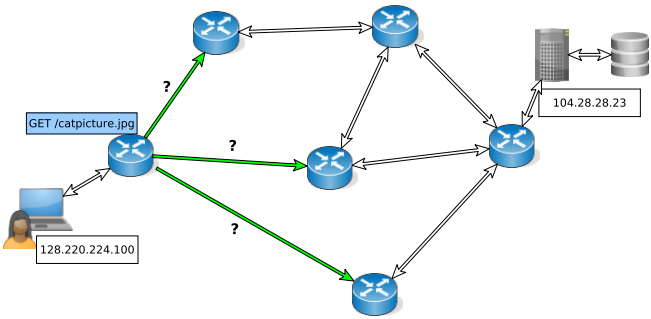
Network with client, server, and intermediate routers

Routing



Client sends request to server: packet sent on default route (user's computer has only one network interface)

Routing

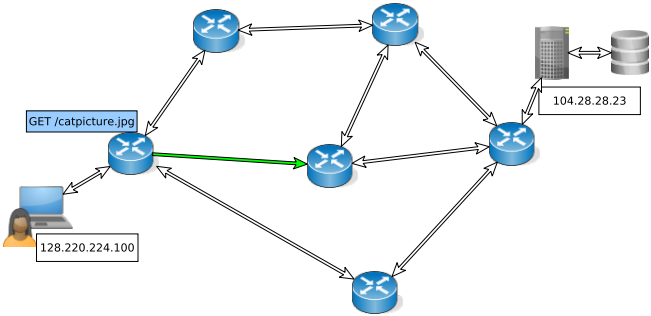


Router has a choice of outgoing links on which to send the packet

Each router has a *routing table* specifying which link to use based on matching the network part of the destination address

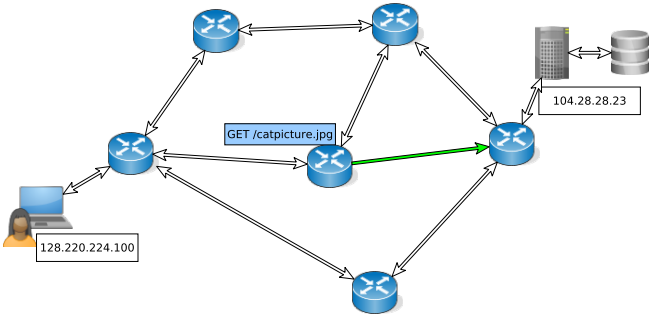
Routing algorithms: try to deliver packets efficiently, and avoid routing loops

Routing



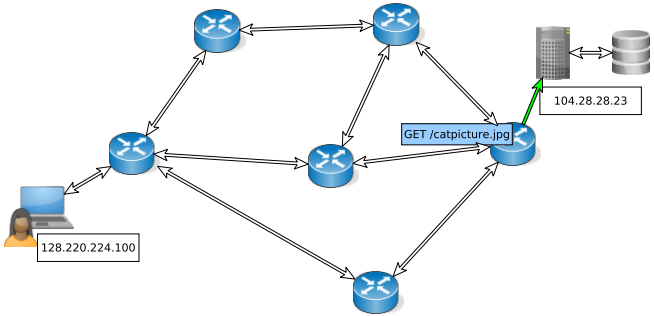
Choose outgoing link based on routing table

Routing



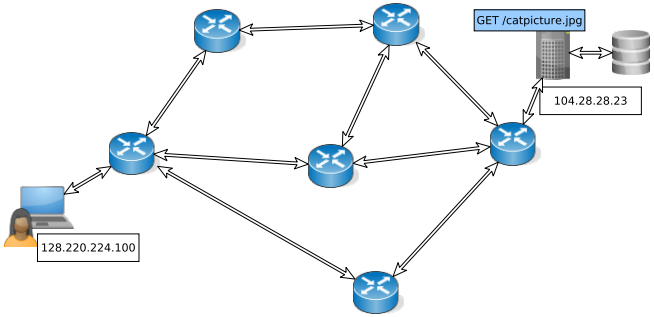
Next hop

Routing



Final hop

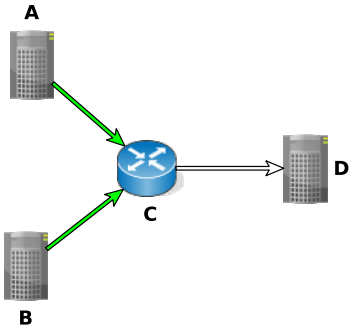
Routing



Packet delivered to server

Server's response will be delivered back to client in a similar manner

Why IP is unreliable

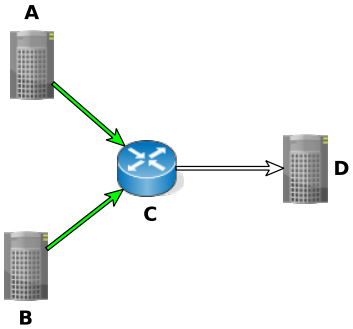


Scenario: A and B both try to send a packet to D at the same time

Outgoing link C→D can only carry one of the two packets

What to do?

Why IP is unreliable

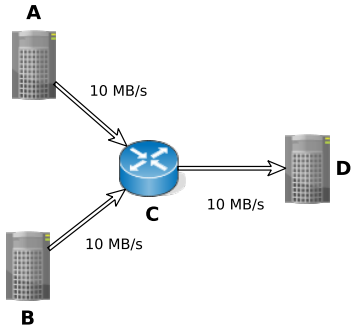


Solution: *queuing*

Router C has a *queue* of unsent packets to be forwarded to D

Either A's packet or B's packet will need to wait in the queue

Why IP is unreliable



Problem: outgoing link C→D cannot handle aggregate data rate of incoming data from A→C and B→C

But, C's queue of packets waiting to be sent to D is finite! (An unbounded queue would imply unbounded delay, not good)

Solution: C discards packets to D when its queue is full

Clicker quiz!

Clicker quiz omitted from public slides

Dropped packets

Dropped packets are a necessary consequence of finite capacity links and finite queues

Reliable protocols such as TCP require acknowledgment of data sent

No acknowledgment $\rightarrow$ assume packet dropped, retransmit

Acknowledgements

Slides adapted from materials provided by David Hovemeyer.