Name: _________________________

If found, call/email:_______________________
# CS445 Calendar Fall 2018

Calendar subject to change – see Moodle for latest updates. Complete assigned reading prior to lecture.

<table>
<thead>
<tr>
<th>Week</th>
<th>Monday</th>
<th>Wednesday</th>
<th>Friday</th>
</tr>
</thead>
</table>
| Aug 27 – 31 | Course Introduction Protocols and Layering  
Reading: 1.1 – 1.4 | Latency, bandwidth, performance  
Reading: 1.5 – 1.6 | Lab 1: Intro Wireshark  
DUE: Pre-lab 1 |
| Sept 3 - 7 | Encoding; Framing  
Reading: 2.1 – 2.3 | Error Detection and Correction  
Reading: 2.4 | Lab 2: Error detection  
DUE: Pre-lab 2; Lab 1 |
| Sept 10 - 14 | Ethernet  
Reading: 2.5 - 2.6 | Wireless | Lab 3: Sockets  
DUE: Pre-lab 3; Lab 2 |
| Sept 17 - 21 | Switching and Bridges Review  
Reading: 3.1; Perlman | EXAM 1  
Chapters 1 - 2 | Spanning Trees  
DUE: Lab 3 |
| Sept 24 - 28 | IP packets  
Reading: 3.2 | IP, ARP, DHCP  
Reading: 3.3 | Lab 4: IP addresses  
DUE: Pre-lab 4  
DUE: Project Proposal |
| Oct 1 – 5 | Distance Vector Routing  
Reading: 3.3 | Link State Routing  
Reading: 3.3 | Lab 5: Wireless, sniffing  
DUE: Pre-lab 5; Lab 4 |
| Oct 8 – 12 | BGP  
Reading: 4.1 – 4.1.2 | IPv6  
Multicast  
Reading: 4.1.3, 4.2 – 4.3 | Lab 6: Project background  
DUE: Pre-lab 6; Lab 5 |
| Oct 15 – 19 | FALL BREAK: NO CLASS (ENJOY!) | | |
| Oct 22 - 26 | Lab 7: Routing, ping  
DUE: Pre-lab 7 | UDP, TCP  
Review  
Reading: 5.1 – 5.2 | EXAM 2  
Chapters 3 – 4  
DUE: Lab 6 |
| Oct 29 – Nov 2 | TCP  
Reading: 6.1 – 6.2 | TCP congestion control  
DUE: Paper Analysis  
Reading: 6.3 – 6.4 | TCP congestion control |
| Nov 5 - 9 | Lab 8: TCP  
DUE: Pre-lab 8; Lab 7 | Congestion avoidance; Compression  
Reading: 6.5 – 6.6 | Compression  
Reading: 7.1 – 7.3 |
| Nov 12 - 16 | Lab 9: Compression, SMTP  
DUE: Pre-lab 9; Lab 8  
Reading: 9.1.1 | Project beta demos  
DUE: Project beta demos | P2P  
Reading: 9.4 – 9.5 |
| Nov 19 - 23 | Lab 10: DNS, HTTP  
DUE: Pre-lab 10; Lab 9  
Reading: 9.1.2, 9.3 | P2P | Thanksgiving Break: No Class |
| Nov 26 – 30 | Review  
DUE: Lab 10 | EXAM 3  
Chapters 5 – 7, 9 | Guest |
| Dec 3 - 7 | Guest | Course Evaluations  
Code Review (Set A)  
DUE: Code + Test Cases | Code Review (Set B) |
| Mon, Dec 10 8 – 10 am | | Project Demos  
DUE: Project Brochure/Flyer |
CS 445: Computer Networks and Internetworking
Fall 2018

Course Information
Instructor: Dr. Tammy VanDeGrift
   Email: vandegri@up.edu
   Office Phone: 503-943-7256
   Office: Shiley 223
   Website: Course information on Moodle, learning.up.edu

Meetings: MWF 1:35 – 2:30 pm
Classroom: Shiley 206
Office Hours: Tentative: M 2:30 – 4:30pm, T 9:30 – 11:30 am; W 8:30 – 9:00am; F 12:30-1:30pm

Bulletin Description: A broad first course in computer networks and internetworking. OSI and TCP/IP layered models, TCP/IP protocol suite, transmission media, local area networks, network and transport-layer protocols, internetworking, internet addressing and routing. (Prerequisites: CS 305 with a grade of C- or better.)

Student Outcomes
At the end of the course, students should be able to:
   • Analyze performance using computer network metrics such as latency and throughput
   • Describe the functions of the network layers: physical layer, link layer, network layer, transport layer, and application layer; protocols such as Ethernet, Wireless, IP, TCP, UDP, HTTP, DNS, SMTP
   • Apply algorithms for forwarding, queuing, routing, congestion control, data compression, and data integrity
   • Synthesize, create, and present technical information related to computer networks
   • Apply network metrics or network programming to a collaborative course project

These outcomes will be accomplished by:
   • Completing labs, a programming project, a paper, in-class activities, quizzes, and exams
   • Participating in discussions and activities through regular class attendance
   • Seeking help when necessary
   • Communicating ideas orally and in writing
   • Working collaboratively as a teammate with other students
   • Providing help instead of giving answers to classmates seeking help

This is an elective course, so it does not serve as a benchmark course for ABET student outcomes.

Course Philosophy
General: This course is designed to introduce concepts related to computer networking. Because this course covers a wide variety of topics, it is critical that you keep up with the material by completing labs, projects, and papers on time and reading assigned material before the lecture. I ask that you read certain chapters, sections, or papers before attending the accompanying class session (see the online course calendar for the latest updates to the readings). Lectures are intended to supplement the textbook. It is okay to struggle with the concepts, but it is your responsibility to seek help when you are confused.

Code of Academic Integrity: Academic integrity is openness and honesty in all scholarly endeavors. The University of Portland is a scholarly community dedicated to the discovery, investigation, and dissemination of truth, and to the development of the whole person. Membership in this community is a privilege, requiring each person to practice academic integrity at its highest level, while expecting and promoting the same in others. Breaches of academic integrity
will not be tolerated and will be addressed by the community with all due gravity. See University Bulletin for policy.
(from UP Bulletin)

Seeking Help: I expect you to have questions as you learn the course material. You may receive help from classmates (see below about Collaborative Learning) and seek help from the instructor. I encourage you to ask questions during lecture meetings and attend office hours.

Collaborative Learning: Your classmates are a huge resource available to you. Because we understand material in different ways, I encourage you to discuss concepts from the course, but any work that you turn in must be your own (or your team’s for team projects and labs). Unacknowledged copying or using parts of someone else’s work (from past semesters or current semester) or information you find in manuals/books/article/the web, even if it has been modified by you, is plagiarism and is not acceptable. When you work with others on labs and projects, you must acknowledge places where you received help in your submissions. When giving help to other classmates, do not give them the answer or show them a copy of your code or labs. Instead, ask questions to learn of their understanding and give conceptual explanations - this practice will help you master the material yourself. Remember: you must turn in work that is your own or authored by your team, you must acknowledge the people who helped you, and you are encouraged to seek help when you are confused. Collaborative learning and teamwork experiences are designed into this course through in-class activities, labs, and a project.

Instructor’s and Students’ Responsibilities: In this course, the instructor’s job is to guide you in learning about networks. In addition to some traditional lecturing, I will have discussions and group activities during lectures. I expect your full participation and readiness to learn at all class meetings. Every student learns in a different way; therefore, the instructor will include a variety of activities in the course.

Classroom Conduct: The Shiley School of Engineering is committed to developing and actively protecting a classroom environment in which respect must be shown to everyone in order to facilitate and encourage the expression, testing, understanding and creation of a variety of ideas and opinions. Failure to meet these standards will result in removal from the class session.

In order to maintain a positive learning environment, students should avoid disruptive behaviors such as: receiving cell phone calls during class, leaving class early or coming to class habitually late, talking out of turn, doing assignments for other classes, reading the newspaper, sleeping, and engaging in other activities that detract from the classroom learning experience.

Because this course takes place in a computer classroom, students are expected to use computers for activities related to the course when instructed by the professor. Computers are not to be used for email, web surfing, social media, video playing, and other personal entertainment unless such activity is related to course activities. Do not unplug computer cables. Respect the computer equipment (no food and drink spills).

The Learning Commons: The Learning Commons, located in Buckley Center 163, offers a variety of peer tutoring programs that facilitate your active learning and mastery of skills and knowledge. For questions about the Learning Commons, please send all correspondence to Jeffrey White, Administrator, at white@up.edu. The Learning Commons is a program of the Shepard Academic Resource Center.

Math Resource Center: Monday through Thursday, 6:00 p.m. through 9:00 p.m. during the first week of classes. Regular shifts begin the Sunday after the first week. For course-specific schedule; visit www.up.edu/learningcommons, or the reception desk in BC 163.

Writing Assistance: Start brainstorming ideas for your paper with a Writing Assistant. Visit www.up.edu/learningcommons to access our Writing Center schedule.

The Language Studio: Language assistance hotlines to schedule a time to meet throughout the semester at chinesetutor@up.edu, frenchtutor@up.edu, germantutor@up.edu, or spanishtutor@up.edu.
Natural Sciences Center: Please send a request to meet to biotutor@up.edu, chemtutor@up.edu, or physicstutor@up.edu.
Speech & Presentation Lab: Improve your presentations by requesting an appointment at speech@up.edu.
Group Work Lab: Make an appointment for your group project at groupwork@up.edu.
Nursing Tutoring: Our peer tutors for pathophysiology will begin providing peer support in BC 163 during the first week of classes to help you start the semester on the right path. Tutoring is available on a walk-in or appointment basis. Up-to-date schedule information is at www.up.edu/learningcommons/nursing.
Economics and Business Tutoring: For support in economics, OTM, finance, accounting, and business law courses, send requests for appointments to your discipline’s respective tutor email hotline: econtutor@up.edu, otmtutor@up.edu, financetutor@up.edu, accountingtutor@up.edu, or bizlaw@up.edu.

Learning Assistance Counselor: Learning assistance counseling is also available in BC 163. The counselor teaches learning strategies and skills that enable students to become more successful in their studies and future professions. The counselor provides strategies to assist students with reading and comprehension, note-taking and study, time management, test-taking, and learning and remembering. Appointments can be made in the on-line scheduler available to all students in Moodle or during posted drop-in hours.

Assessment of Learning
I will assess your learning and mastery based on your submitted work, including pre-labs, labs, a project, a paper, exams, and in-class activities/quizzes. Generally, the labs, project, reports, and in-class activities are your chance to learn, while the exams are the main way I will assess what you have learned. I expect you to submit your labs and projects by the due date and time. If circumstances arise (e.g. you are ill for an extended period of time, you are out of town for a university-related activity) that prevent you from submitting your work on time, please discuss the reason with me before the due date. Note that you have two free late days to use on prelabs and/or labs.

Grading Scheme
Course grades will be assigned based on the total points you earn during the semester, weighted accordingly to the categories shown below. The minimum cutoffs for grades will not change. I do reserve the right to raise your grade, but the following minimum percentages are guaranteed. (For example, if you earn 90% of the points, you will get an A-. If you earn 89% of the points, you earn a B+ but I reserve the right to raise your grade to an A-.)

- >= 93% A
- >= 90% A-
- >= 87% B+
- >= 83% B
- >= 80% B-
- >= 77% C+
- >= 73% C
- >= 70% C-
- >= 67% D+
- >= 63% D
- >= 60% D-
- < 60% F

Weights of Graded Work in the Course

<table>
<thead>
<tr>
<th>Percent</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>Exams</td>
<td>Three exams @12% each, completed individually.</td>
</tr>
<tr>
<td>10</td>
<td>Analysis Paper</td>
<td>A paper describing and analyzing a protocol or research paper related to networking. Paper should be completed individually.</td>
</tr>
<tr>
<td>10</td>
<td>Pre-labs</td>
<td>Pre-labs should be completed individually.</td>
</tr>
<tr>
<td>15</td>
<td>Labs</td>
<td>Some class sessions will include laboratory exercises usually completed in pairs; some labs have post-labs to be completed in the same pair. You should work with a different partner for each lab.</td>
</tr>
<tr>
<td>20</td>
<td>Project</td>
<td>A team programming project related to networking, including the proposal, beta test, code review, demo, and flyer</td>
</tr>
</tbody>
</table>
Late Assignments: You are granted two full days to use at your discretion for submitting late pre-labs or labs without penalty. For example, you may choose to turn in two different assignments 24 hours after their due dates and times. You could choose to use the two days (48 hours) on a single assignment. Weekends count as regular days. If you use a late day, indicate the use of late day(s) with a note on the assignment when you submit your work. For partner labs, both partners will be charged late days. If you need to submit an assignment late (due to illness, death in the family, out of town for a university event) and you have already used your two late days, please contact me before the due date and time and we can discuss your options. Late days may not be used for exams, in-class activities, quizzes, or project deliverables.

Logistics


Course Calendar and Website: The latest version of the calendar and course materials are posted to Moodle (learning.up.edu). The course calendar lists the lecture topics, assigned readings, exams, and due dates for assignments. The calendar is subject to change as the semester progresses, so check the on-line version frequently.

Accessibility Statement: The University of Portland endeavors to make its courses and services fully accessible to all students. Students are encouraged to discuss with their instructors what might be most helpful in enabling them to meet the learning goals of the course. Students who experience a disability are also encouraged to use the services of the Office for Accessible Education Services [AES], located in the Shepherd Academic Resource Center (503-943-8985). If you have an AES Accommodation Plan, you should make an appointment to meet with your faculty member to discuss how to implement your plan in this class. Requests for alternate location for exams and/or extended exam time should, where possible, be made two weeks in advance of an exam, and must be made at least one week in advance of an exam. Also, you should meet with your faculty member to discuss emergency medical information or how best to ensure your safe evacuation from the building in case of fire or other emergency.

Non-Violence Statement: The University of Portland is committed to fostering a community free from all forms of violence in which all members feel safe and respected. Violence of any kind, and in particular acts of power-based personal violence, are inconsistent with our mission. Together, we take a stand against violence. Join us in learning more about campus and community resources and reporting options, along with our prevention strategy - Green Dot. To get involved or request more information, please contact Tiger Simpson at simpsont@up.edu.

Assessment Disclosure Statement: Student work products for this course may be used by the University for educational quality assurance purposes.
**Academic Regulation Statement:** Policies governing your coursework at the University of Portland can be found in the University Bulletin.

**Mental Health Statement:** As a college student, you may sometimes experience problems with our mental health that interferes with academic experiences and negatively impact daily life. If you or someone you know experiences mental health challenges at UP, please contact the University of Portland Health and Counseling Center in Orrico Hall at www.up.edu/healthcenter/ or 503-943-7134. Their services are free and confidential, and if necessary they can provide same day appointments. Also know that the University of Portland Public Safety Department (503-943-4444) has personnel trained to respond sensitively to mental health emergencies at all hours. Remember that getting help is a smart and courageous thing to do – for yourself, for those you care about, and for those who care about you.

**Improving the Course:** I welcome your feedback about the course at any time. I may ask for your feedback periodically and you will have the opportunity to evaluate the course at the end of the semester.
CS 445: First Day Activity

Instructions: Divide into groups of 3 to 4 students. Introduce yourselves to one another.

Names: ________________________________________________________________

With the other members of your group, discuss the following questions and jot down some of your group’s ideas. You can put the ideas on one piece of paper.

1. What is a network? (Provide a general definition)

   ________________________________________________________________

2. List examples of networks (need not be computer-based) you see in the world. For example, Dish Network is a network that provides TV broadcasts to customers.

   ✔ ________________________________________________________________
   ✔ ________________________________________________________________
   ✔ ________________________________________________________________
   ✔ ________________________________________________________________

3. Suppose you are designing a computer network (two or more computers connected by some link). What goals (properties) do you want to achieve with your network?

   ✔ ________________________________________________________________
   ✔ ________________________________________________________________
   ✔ ________________________________________________________________
   ✔ ________________________________________________________________

   (Stop here for class discussion. If you have time, discuss your personal goals for this course.)
First challenge: How do we share a link?

Suppose we have computers A, B, and C connected via a switch and D, E, and F connected via a switch. A wants to send to F, B wants to send to D, and C wants to send to E (the sending may be done in spurts). How do we share the link between the switches?

Below are three different approaches to sharing a link. For each approach, discuss and write down the pros and cons. Think of the pros/cons from several perspectives: the nodes, the switches, fairness, and any other metrics you think are important for network design.

**Option 1**: Time-sharing (STDM: Synchronous Time Division Multiplexing)
The sending nodes (A, B, and C) share the link by taking turns via time. Time is broken into slices (let’s say .1 second each), so A gets to use the link between the switches for .1 second, then the others get it for .2 second, and then A gets to use it again.

**Option 2**: FDM: Frequency Division Multiplexing
The signal is split up into three different channels and each channel is sent at a different frequency over the switch-to-switch link. This is similar to how TV signals are transmitted.

**Option 3**: Statistical Multiplexing
The data is broken into packets (with a fixed maximum size). Packets arrive at the switch and each is sent along the switch-to-switch link. The switch needs to determine how to prioritize the packets. This could be done in a first-in first-out (FIFO) manner, or a round robin fashion among A, B, and C.
<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>STDM</td>
<td></td>
</tr>
<tr>
<td>FDM</td>
<td></td>
</tr>
<tr>
<td>Statistical Multiplexing</td>
<td></td>
</tr>
</tbody>
</table>
Goal #1: You want to get the current news for the day.

Here’s you:

You open up a browser on your laptop and type in www.cnn.com in the address field.

What happens?

1. DNS – look to cache for a copy of cnn.com’s IP address.

   Address for cnn?

   157.166.226.25

   Hopefully, the local DNS nameserver has the IP address for cnn.com. If not, it can fetch it from another nameserver.

2. Now, HTTP messages are sent to the web server. [Application Layer]

   GET index.html  
   (page is returned)
   GET logo.gif  
   (file is returned)
   ...

   Each item displayed on the page is requested separately.
3. TCP makes sure each HTTP message is actually received. Each GET message is acknowledged, so an ACK is sent from the person back to CNN. If no ack is received, the page is sent again. [Transport Layer]

4. While this is happening, the CNN server is probing the network to see how fast the acks are being returned. This information is used to determine how congested the network is right now and then the server adjusts rate of how much data to send. [Transport Layer]

5. The information is broken into packets. Each packet is routed through the network. [Network Layer]

6. Each packet is encoded into frames and the bits are encoded into signals. [Link Layer]

Each packet contains the actual data and the address to which to deliver the packet.

Architecture: need abstractions to manage complexity (just like designing software). You would really like to just use the lower-level protocols rather than coding (or re-implementing) that entire process yourself. Can think of the lower-level protocols like APIs.

Protocol: provides communication service that higher-level objects use to exchange messages (OR) agreement dictating form and function of data exchanged

In computer networks, protocols are combined via layering, like an onion. (See figure 1.11 in the textbook.)
**Goal #2: You (host 1) want to send a file to another computer called host 2.**

You have a file that you want to send. You have some payload (which is the actual data in the file)

Host 1:

- **Apps:** video player, file app, email app
- **Transport Protocols:**
  - RRP (request/reply protocol);
  - MSP (message stream protocol)
- **Link Protocol:** HHP (host to host protocol)

What transport protocol would the video player use? ______________________

What transport protocol would the file app use? ______________________

What transport protocol would the email app use? ______________________

So, here is what host 1 will do to the file (payload):

```
Payload
```

Then, the RRP protocol adds a header to the payload:

```
RRP Payload
```

Then, the HHP protocol adds a header to the payload:

```
HHP RRP Payload
```

Then, this entire packet gets sent through the network and is received at host 2. Host 2 deals with the packet by looking at the first header: HHP. HHP deals with the packet by stripping the HHP header and looking at the next header: RRP. It hands it to the RRP protocol, which strips the header and hands the payload to the file application.
**Goal #3: Where do we assign functionality in a network?**

Generally, we want to push as much of the functionality to the endpoints (hosts), so the internal routers/switches can just focus on forwarding. This had aided the success of the Internet (new applications can be developed, since no changes to the infrastructure are necessary).

**Network Layers:**

**OSI (Open Systems Interconnection) architecture**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Up to app</td>
</tr>
<tr>
<td>Presentation</td>
<td>Encode/decode data, format data</td>
</tr>
<tr>
<td>Session</td>
<td>Manage connections and multiple streams</td>
</tr>
<tr>
<td>Transport</td>
<td>Reliability, congestion control</td>
</tr>
<tr>
<td></td>
<td>(layers above are done at the end hosts)</td>
</tr>
<tr>
<td>Network</td>
<td>routing packets</td>
</tr>
<tr>
<td>Link</td>
<td>Framing</td>
</tr>
<tr>
<td>Physical</td>
<td>Bit encoding</td>
</tr>
<tr>
<td></td>
<td>(internal to network)</td>
</tr>
</tbody>
</table>

**Internet Architecture**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Example protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>HTTP, SMTP, FTP</td>
</tr>
<tr>
<td>Transport</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Network</td>
<td>IP</td>
</tr>
<tr>
<td>Link</td>
<td>Ethernet, Wireless, Fiber</td>
</tr>
</tbody>
</table>

How does the Internet Architecture map to the OSI architecture?
The Internet architecture is like an hourglass. Everything must go through IP.

This actually makes the network scalable. It is similar to having a JVM for Java (Java code is run on a virtual machine that then translates instructions to the actual hardware). Because everything must use the Internet Protocol, new apps can be developed.

In this course, we’ll go through the material bottom-up:
- Link layer (Ethernet, wireless)
- Network layer (IP)
- Transport layer (TCP, UDP)
- Application layer (HTTP, SMTP, DNS, peer-to-peer systems)

*We will approach the course like a systems design course (discuss pros/cons of solutions) and become familiar with the network protocols.*
Network Performance Metrics

Bandwidth = \( \frac{\text{maximum number of bits transmitted}}{\text{time}} \)

Example: 10 Mbps (10 million bits per second)

Note: In EE, bandwidth refers to frequency band (3000 Hz), but here it is \# bits / time
Note: can look at bandwidth as how “wide” is a bit. If the bandwidth is 1 Mbps, then 1 bit is 1 microsecond “wide”.

Throughput = \( \frac{\text{actual number of bits transmitted}}{\text{time}} \)

Note: bandwidth and throughput are often used interchangeably, but bandwidth is the theoretical max and throughput is the actual data rate

Latency = Delay = total time to send message from sender (start of transmission) to receiver (end of transmission) = \( P + T + Q \)

Note: measured in time
Parts:
\( P \) = propagation time (time of travel in physical medium, nothing is faster than light)
Speeds:
2.3 x 10^8 meters/second in cable
2.0 x 10^8 meters/second in fiber
\( T \) = transmit (time to transfer data, based on size of data and bandwidth)
\( Q \) = queue (time data sits at switches waiting to be forwarded + processing time to check headers and forward)

\( P = \frac{\text{distance}}{\text{speed of medium}} \)

\( T = \frac{\text{size of data}}{\text{bandwidth of link}} \)

RTT = round trip time = 2 x propagation time
**Let’s think about latency versus bandwidth.**

Suppose I want to send a 1-byte message from Portland to New York and then receive a 1-byte message from New York.

Is this latency-bound or bandwidth bound? __________________________

Suppose I want to send a 50 Gigabyte video to New York.

Is this latency-bound or bandwidth bound? __________________________

**Another measurement: delay x bandwidth product**

Measures the number of bits that can be in flight in the pipe

Think of the cross-section of the pipe as the bandwidth. Think of the length of the pipe as the latency (delay).

Example:

Latency = 50 ms  
Bandwidth = 45 Mbps

Then $D \times B = (50 \times 10^{-3} \text{ seconds}) \times (45 \times 10^6 \text{ bps}) = 2.25 \times 10^6 \text{ bits}$

Practice: What is the $D \times B$ product for a latency of 10 ms and a bandwidth of 500 Mbps?

_________________

Why is this $D \times B$ measurement important?

Think of it is the number of bits sender can continuously send before the first bit arrives at the receiver. Want to fill the pipe. May need to send $2 \times (\text{delay} \times \text{bandwidth})$ bits before getting response from receiver (2 pipes for round trip).
Examples (from textbook – typical speeds with typical distances):

<table>
<thead>
<tr>
<th>Technology</th>
<th>d x bw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dial up</td>
<td>5 bits</td>
</tr>
<tr>
<td>Wireless</td>
<td>18 bits</td>
</tr>
<tr>
<td>Satellite</td>
<td>10 Mb</td>
</tr>
<tr>
<td>Trans-continental Fiber</td>
<td>400 Mb</td>
</tr>
</tbody>
</table>

Be careful with M and k!!

- b = bit
- B = byte
- $M = 10^6$ (when referred to Mbps, as in bandwidth)
- $M = 2^{20}$ (when referred to MB, as in data size)
- $k = 10^3$ (bandwidth)
- $k = 2^{10}$ (size)
Let's practice some sample calculations:

Example:
Suppose you want to send a 1.5 MB file.

- RTT = 80 ms.
- Packet size = 1 KB.
- There is 2 x RTT of handshaking preceding the transfer of the file (this is the case for TCP, which we will see later in the course).
- Assume no queuing or processing time (directly connected computers)

What is the total time to transfer the 1.5 MB file, to the nearest thousandth of a second, in the following cases?

a. bandwidth is 10 Mbps and data/packets are sent continuously

\[
\text{total time} = (2 \text{ RTT}) + \text{Latency} = (2 \text{ RTT}) + (P + T + Q) = (160 \text{ ms}) + (40 \text{ ms} + (1.5 \text{ MB}/10 \text{ Mbps}) + 0 \text{ seconds queuing})
\]

// note: 40 ms is one-half of the 80 ms of RTT

\[
= .160 \text{ sec} + .040 \text{ sec} + (1.5 \times 2^{20} \times 8 \text{ bits})/(10 \times 10^6 \text{ bits/sec})
\]

\[
= .2 \text{ sec} + (12582912 \text{ bits})/(10000000 \text{ bits/sec})
\]

\[
= .2 + 1.258 \text{ seconds}
\]

\[
= 1.458 \text{ seconds}
\]

b. bandwidth is 10 Mbps, but after sending each packet, must wait one RTT before sending the next packet

First, let's see how many packets we need to send:

\[
\# \text{ packets} = (1.5 \text{ MB} / 1 \text{ kB}) = (1.5 \times 2^{20}) / (1 \times 2^{10}) = 1572864 / 1024 = 1536
\]

\[
\text{total time} = (2 \text{ RTT}) + \text{Latency} = (2 \text{ RTT}) + (P + T + Q) = (1.458 \text{ sec}) + (1535 \times .08 \text{ seconds})
\]

//note: 1535 waits for 1536 packets

\[
= 1.458 + 122.8 \text{ sec}
\]

\[
= 124.258 \text{ sec}
\]
c. bandwidth is infinite, up to 20 packets can be sent each RTT

From part (b), we know the # of packets is 1536.

Thus, there are \( \frac{1536}{20} = 76.8 \) transfers. Need to round up to 77 batches. Each batch takes one RTT, but the first batch arrives in \( \frac{1}{2} \) RTT. Then 76 RTTs between the first batch and the 77th batch.

Group one takes \( \frac{1}{2} \) RTT to get to destination.

----------

Group two takes 1 RTT to get to destination
Group three takes 1 RTT ... 
Group 77 takes 1 RTT

Total time 
\[
= (2 \text{ RTT}) + \text{Latency} \\
= (2 \text{ RTT}) + (P + T + Q) \\
= (2 \text{ RTT}) + (0.0765 \times 0.08 \text{ sec } + 0 + 0) \\
= (78.5 \text{ RTT } \times 0.080 \text{ s }) \\
= 6.28 \text{ sec}
\]
**Example:** Suppose 128-kbps point to point link between Earth and Mars. Distance between Earth and Mars is 55 Gm and data travels at the speed of light: $3 \times 10^8$ meters per second.

a. What is the minimum RTT for the link?

$$\text{RTT} = 2 \times \text{Propagation}$$
$$\text{RTT} = 2 \times (55 \times 10^9 \text{ meters}) / (3 \times 10^8 \text{ m/sec})$$
$$\text{RTT} = 2 \times 184 \text{ sec}$$
$$\text{RTT} = 368 \text{ sec}$$

b. What is the delay x bandwidth product?

$$D \times B = \text{(one way time)} \times 128 \text{ kbps}$$
$$D \times B = (368 \text{ sec} / 2 \text{ sec}) \times (128 \text{ kbps})$$
$$D \times B = 184 \text{ sec} \times 128 \times 10^3 \text{ bits / sec}$$
$$D \times B = 2.81 \text{ MB}$$

c. A camera takes pictures and sends them back to Earth. How quickly after a picture is taken can it reach mission control on Earth? Assume the image is 5 MB.

$$\text{Latency} = P + T + Q$$
$$\text{Latency} = (184 \text{ sec}) + (41943040 \text{ bits})/(128 \times 10^3 \text{ b/s}) + 0$$
$$\text{Latency} = 184 + 328 \text{ sec}$$
$$\text{Latency} = 512 \text{ sec}$$

**Example:** Calculate the bandwidth necessary for transmitting the following data in real time.

a. HDTV (1920 x 1080 pixels, 24 bits/pixel, 30 frames/second)

$$\text{bandwidth} = (1920 \times 1080 \text{ pixels/frame} \times 24 \text{ bits/pixel} \times 30 \text{ frames/second})$$
$$\text{bandwidth} = 1.5 \text{ Gbps}$$

b. telephone service (8-bit samples at 8kHz)

$$\text{bandwidth} = (8 \text{ bits} \times 8 \times 10^3 \text{ Hz})$$
$$\text{bandwidth} = 64 \text{ kbps}$$
**Example:** Calculate the latency (first bit sent to last bit received) for the following cases:

a. 1-Gbps link with a single store and forward switch in the path. Packet size is 5000 bits. Each link introduces a propagation delay of 10 us and that the switch begins retransmitting immediately after it has finished receiving the packet.

   For one link:
   
   Latency = P + T + Q
   
   Latency = 10 us + (5 kb / 1 Gbps) + 0
   
   Latency = 10 us + 5 us
   
   Latency = 15 us

   For two links: latency is 30 us

b. Now suppose there are 3 switches between the sender and receiver (4 links).

   For one link:
   
   Latency = 15 us
   
   For four links:
   
   Latency = 60 us
CS 445: Performance Metrics Activity

Get into a group of 3 – 4 people to complete these exercises.

**Problem 1:** Consider a point-to-point link 50 km in length. At what bandwidth would propagation delay (at a speed of $2 \times 10^8$ m/s) equal transmit delay for a 100-byte packet?

Bandwidth = ____________________

**Problem 2:** Same situation as problem 1 except now the packet is 512-bytes.

Bandwidth = ____________________

**Problem 3:** Calculate the bandwidth necessary for transmitting in real time for the following data:

a. Mobile audio of 260-bit samples at 50 Hz

b. High-def audio of 24-bit samples at 88.2 kHz

c. Video at 768 x 432 pixels, 24 bits/pixel, and 30 frames per second
CS 445: Encoding (NRZ, NRZI, Manchester, 4B/5B)

<table>
<thead>
<tr>
<th>Encoding Name</th>
<th>Original Bit</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRZ</td>
<td>1</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>NRZI</td>
<td>1</td>
<td>Transition to low or high</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>No transition</td>
</tr>
<tr>
<td>Manchester</td>
<td>1</td>
<td>High to Low</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>Low to High</td>
</tr>
<tr>
<td>4B/5B</td>
<td>4 bit sequence</td>
<td>Use the 5-bit pattern in Table 2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Then use NRZI to encode those 5 bits</td>
</tr>
</tbody>
</table>

Below encode the signals for the following bit sequence. Assume the clock read/write is represented by the vertical lines. Assume the previous bit in the NRZI encoding was a 0 encoded as a low signal.

00110101

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRZI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manchester</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In 4B/5B, the bits 0011 translate to 10101. The bits 0101 translate to 01011. Then, the bits 1010101011 would be encoded using NRZI. Draw the encoding below (assume the previous bit before this sequence was a low signal.

<table>
<thead>
<tr>
<th>Encoding Name</th>
<th>Pros?</th>
<th>Cons?</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NRZI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manchester</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4B/5B</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CS 445: Framing

Now that we can get bits into a wire / signal (low/high voltages), how do we package bits together? This is the challenge that framing solves.

How would you package bits together?

_____________________________________________________________________

Option 1: Byte-oriented with sentinels
   Control code (sentinel) at beginning and end of frame
   Control code (sentinel) in data part needs to be escaped

   Example: PPP (Point to Point Protocol)
       Flag | Header | Payload | Checksum | Flag

       Flag is a special code to indicate start and stop of frame (used on dial up modems)

Option 2: Byte-oriented with header size
   Put # of bytes in the header
   Then read that many for the frame

Option 3: Use the clock
   Send a fixed number of bytes per frame

   Example: SONET (synchronous optical network)
   Send 9 rows of 90 bytes per frame

Option 4: Bit-oriented (have beginning and ending bit pattern)

   Example: HDLC (high-level data link control)
   Pattern is 01111110 for beginning and end

   If seen in the data, the sixth one is stuffed with a 0
CS 445: Error Detection

How could data get corrupted?

Why would we want to be able to detect data transmission errors?

Two-Dimensional Parity

Magic trick example

Assume this is the original data:
0100110
0000110
1011101
0010100

What data would be added for 2D parity error-detection?

Does this always detect 1-bit errors? Yes No
Does this always detect 2-bit errors? Yes No
Does this always detect 3-bit errors? Yes No
Does this always detect 4-bit errors? Yes No

Can this correct 1-bit errors? Yes No
Can this correct 2-bit errors? Yes No
Internet Checksum (done in IP)

Idea: Add up numbers and then transmit the sum.

Formula for checksum:
1. Add up the words transmitted using ones complement arithmetic.
2. Use the sum as the checksum (represented in ones complement)

Review of ones complement representation:
   Positive numbers: represented normal in binary
   Negative numbers: each bit is inverted

Examples of ones complement:
7 = 00000111
-7 = 11111000

Assume data is the following:
00000010 (2)
00000101 (5)
00000111 (3)
11111100 (-3)

Then, when we add the numbers, we get:
00000010 (2)
00000101 (5)
   = 00000111 (7)
+ 00000011 (3)
   = 00001010 (10)
+ 11111100 (-3)
   = 00000110 (plus a 1 in the carry bit, so this is added to the LSB)
+ 1
   = 00000111 (7)

So, 00000111 would be sent as the checksum for the data.
**CRC – Cyclic Redundancy Check (done in ethernet for 32-bit CRC)**

Idea: use polynomial algebra to find polynomial to pad original message

**Background:**
1. Represent message $M$ by a polynomial $M(x)$
   - If $M = 10010101$, then $M(x) = x^7 + x^4 + x^2 + x^0$
2. Sender calculates $P(x)$ from $M(x)$ that is exactly divisible by $C(x)$, where $C(x)$ is an established agreed upon polynomial with degree $k$.
3. Send $P(x)$.
4. Receiver divides $P(x)$ by $C(x)$. If remainder is not 0, an error occurred. $M(x)$ is all but the last $k$ bits and is used as the message.

To calculate $P(x)$:
1. Multiply $M(x)$ by $x^k$ where $C(x)$ has degree $k$. Let the result be $T(x)$.
2. Find remainder $R(x)$ of $T(x) / C(x)$
3. $P(x) = T(x) - R(x)$

**Example of finding $P(x)$:**
$M = 10101101$
$M(x) = x^7 + x^5 + x^3 + x^2 + x^0$
$C(x) = x^3 + x^2 + 1$

Since the degree of $C(x) = 3$, we multiply $M(x)$ by $x^3$.
$T(x) = x^{10} + x^8 + x^6 + x^5 + x^3$

Now, find remainder $R(x)$ when dividing $T(x)$ by $C(x)$ (division is done with subtraction as XOR):

$\begin{array}{c|ccccc}
1101 & 11011011 & \text{Remainder: 111} \\
10101101000 & 1101 & 111 & 1101 & 0101 & 0000 & 1010 & 1101 & 1111 & 1101 & 0100 & 0000 & 1000 & 1101 \\
\end{array}$
\[
\begin{array}{c}
1010 \\
1101 \\
111
\end{array}
\]

\[P(x) = T(x) - R(x)\]

\[
\begin{array}{c}
10101101000 \\
- 111
\end{array}
\]

\[\begin{array}{c}
10101101111
\end{array}\]

(subtraction done via XOR)

\[10101101111\] is sent to receiver

Receiver calculates message: \(P(x) / C(x)\)

\[
\begin{array}{c}
11011011 \\
10101101111
\end{array}
\]

Remainder: 0

Thus, the data is “good” and the message is 10101101. (remove last 3 bits from what was sent)
Practice with CRC:

M = 11100110
C(x) = x^3 + x^2 + 1

What bits get sent? ____________________________
CS445: Reliable Delivery and ARQ

What could fail when sending data from Host A to Host B? ____________________________

From Host A’s perspective: would like to know that Host B got the frame/packet (data)

ARQ: Automatic Repeat ReQuest

Rules of the game:
- Receiver automatically acknowledges correct frames
- Sender automatically resends after a timeout, until ack for that frame is received

Stop and Wait

Idea:
1. Send a frame
2. Wait for acknowledgment
3. If no ack received within a timeout, send frame again
4. If ack arrives, send next frame of data

Activity: What could other pictures look like?
- Frame lost, resend frame
- Ack lost, resend frame
- Timeout, send frame, receive ack from previous transmission of frame
Discussion questions:

1. How long should we set the timeout?

2. Suppose the sender has two frames of data to send. Draw a picture where the data for frame 1 is resubmitted but the receiver receives the resubmission as new data.

3. How do we solve the issue in the previous question? Sequence numbers (0 or 1) since two frames can be in transit at once.

Go through the scenarios from previous page again, but this time using sequence numbers.
Frame lost, resend frame

Ack lost, resend frame

Timeout, send frame, receive ack from previous transmission of frame

PRO of Stop-and-wait: simple
CON of Stop-and wait: not utilizing bandwidth (lots of waiting)
CS 445: Sliding Window (reliable transmission)

Idea: Keep multiple frames in flight at once (Keep pipe full)
Do not wait for ack from previous frame before sending next frame

On sender side:
Assign seqNum to each frame
Maintain 3 variables:
SWS – send window size (# of simultaneous frames in flight)
LAR – last ack received
LFS – last frame sent

Note: LFS – LAR <= SWS

Sender keeps timer for each frame sent, resends if necessary
Sends another frame when ack arrives

On receiver side:
Maintains 4 variables:
RWS – receive window size (# frames can store in buffer)
LAF – last acceptable frame
LFR – last frame received
SeqNumToAck – current frame number for acknowledgment

Note: LAF – LFR <= RWS

Upon receipt of frame F:
If F has seqNum <= LFR or seqNum > LAF, discard
Else, accept frame into buffer
Assigns SeqNumToAck to last consecutive frame received
If SeqNumToAck changed, send ack with the SeqNumToAck
LFR = SeqNumToAck
LAF = LFR + RWS
Example:

Sender  SWS = 4

S sends F1  
R gets F1 and sends Ack 1  
LFR = 1  
LAF = 5

S sends F2  
R gets F2 and sends Ack 2  
LFR = 2  
LAF = 6

...

S receives Ack 1
S sends F5  
R gets F5 and sends Ack 5  
LFR = 5  
LAF = 9

S receives Ack 2
S sends F6  
F6 dropped

S receives Ack 3
S sends F7  
R gets F7 (no ack sent)  
LFR = 5  
LAF = 9

S receives Ack 4
S sends F8  
R gets F8 (no ack sent)  
LFR = 5  
LAF = 9

X (timeout fires for F6)

S sends F6  
R gets F6 and sends Ack 8  
LFR = 8  
LAF = 12

Note: sender would send frames 1, 2, 3, and 4 immediately, since SWS is 4. The sender sends F5 once the ack for 1 is received.
Example: calculating the size of sequence numbers

- Link Bandwidth: 1 Mbps point-to-point link
- Distance: $3 \times 10^4$ km
- Each frame carries 1 KB of data
- Speed of light: $3 \times 10^8$ m/s
- RWS: 1

a. How many bits are necessary to store sequence numbers for sliding window?

Need to think about how much data can be in flight at once (RTT x bandwidth) since that is how much data could be in flight before hearing from the receiver.

One-way prop delay =
# frames per second =

Bandwidth x RTT =

Largest sequence number =
# of bits to store this number =

b. What if the RWS = SWS?

Largest sequence number =
# of bits to store this number =
CS 445: Sharing Links and Ethernet, Assume no one is in charge (this is a distributed system)

Suppose nodes share a single link or shared space. How should nodes share? Who sends when? No one is in charge of managing turn-taking.

What issues may arise?
Aloha – developed to support communication in a radio network in Hawaii in 1960s (developed by Norm Abramson)

Protocol: When you have data to send, send
When collisions occur, wait a random length of time and try again

Simple as that.

Is this a good idea? When would it work well? (Works at most 18% efficiency. Goes to 36% when time is divided into slots.)
Sharing (no switches)

**CSMA/CD** – Carrier Sense Multiple Access with Collision Detect

**Carrier Sense** = can distinguish between a busy and an idle link, so check it first (note: Aloha had no wires, much easier to do with wires)

**Multiple Access** = multiple nodes sharing a common resource (link)

**Collision Detect** = node listens as it transmits and can detect when an interference (collision) occurs

Protocol: If link is idle, send if you have data
If link is busy, wait until idle and may wait longer to send (see below)

Options:
1. 1-persistent: wait until link is idle and send (then all queued up senders will collide) [Ethernet]
2. p-persistent: send with probability p [Aloha]

Issue: What if 2 nodes start sending at the same time (both detect an idle link, but because of distance, they do not detect another transmission until after they have started sending)?

We get a collision, so nodes need to send jamming sequence to let other nodes know that the link is not idle. Then we need to determine when to send again.
**Classic Ethernet (802.3)**

Nodes share single link or are connected via hubs/repeaters. In any case, only one frame can be sent across the entire LAN at once.

**Background**

Frames (total size) are 64 bytes – 1500 bytes in size.
Length of LAN: coax (up to 500 m), can be up to 2500 m with twisted pair or fiber (can also use repeaters and hubs to gain distance) [limited due to jamming signal size]
Max # hosts: 1024
Original: used Manchester encoding, 4B/5B or 8B/10B used today on high-speed Ethernets

Each adapter has a unique address (6 bytes long, assigned by hardware manufacturer)
(MAC address, media access control)

**Frame format**

8 bytes preamble | 6 bytes destination | 6 bytes source | 2 bytes type (demux for higher-level protocols)
| payload | 4 bytes CRC

Preamble is alternating 0’s and 1’s to synchronize sender and receiver

**Receiving frames**

Every node sees every frame in the network.

*Promiscuous* mode: adapter receives all frames and delivers all to host
Most cases, adapter only delivers frames addressed to the host
- Broadcast address is all 1’s
- Multicast address: first bit is 1 and then the group address

**Sending frames**

If idle, send and hope for no collisions

**Dealing with collisions**

Each frame must be at least 64 bytes long, so it is on the wire long enough to detect collisions
Sends a 32-bit to 512-bit jamming sequence (plus 64-bit preamble), depending on how far away the hosts are from each other

Ethernet uses exponential backoff when collisions occur to get closer to probability 1/n for n concurrent senders.

1\textsuperscript{st} collision: wait 0 or 1 frame times (51.2 us) and retry
2\textsuperscript{nd} collision: wait 0, 1, 2, or 3 frame times and retry
Nth collision: wait 0, 1, ... $2^{N-1}$ frame times and retry
Cap N at 10

PROS: _______________________

CONS: _______________________

Mitigation: limit hosts to 200, limit spread so RTT delay is closer to 5 us rather than 51.2 us

Improvement we’ll see soon: switched Ethernet (fewer nodes on shared LAN)
CS 445: Wireless

Shared medium is space (signals transmitted over certain frequencies)

Different Forms:
1. Bluetooth – used for short distances, personal networks
2. Wi-Fi – used for local area networks, usually connect computer to a base station, 802.11
3. Cellular – tens of miles, cell phones to towers

Wi-Fi: 802.11 (data rate today: ______________)

Suppose we have 3 hosts (A, B, C):

Now, nodes have a limited range. Here, A can send to B, but A cannot send to C. A is a “hidden node” for C and C is a “hidden node” for A.

What if A and C both want to send to B simultaneously? __________________________

Will collision detection (used in Ethernet) work? __________________________

Idea: Listen before transmitting, reserve medium by sending a special signal, if no ack for reservation, assume there is a collision and use exponential backoff, otherwise – send data; the exponential backoff randomizes the length of time when a node will send again, so hopefully, there is less chance for nodes to send again at the same time.
Called CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)

Example: A and D are just out of range

A  B  C  D

B wants to send to C and A wants to send to B:
1. B sends C a RTS (Request to Send) frame that includes how long B wants to reserve the space.
2. A hears RTS and defers its transmission.
3. C replies to B with CTS (Clear to Send) frame
4. D hears CTS and defers to allow the data to be sent
5. B sends data to C
6. C sends ack to B

Medium is available again and now A can send a RTS to B

*Note: Any node that hears RTS but not CTS frame can send, too (the sending will be out of range). Any node that sees CTS must wait.

Commonly, nodes communicate directly with a base station (wireless access point). Use this CSMA/CA algorithm with nodes connected to same AP, then use whatever wired LAN that connects the access points. APs periodically send out beacons that advertise their capabilities.

To connect to an access point:
1. Node sends a Probe frame
2. All AP’s reply with a Probe Response frame
3. The node selects an AP and sends an Association Request frame
4. AP responds with Association Response frame.
802.11 data frame format:

<table>
<thead>
<tr>
<th>Control</th>
<th>Duration</th>
<th>Addr1</th>
<th>Addr2</th>
<th>Addr3</th>
<th>SeqCtrl</th>
<th>Addr4</th>
<th>Payload</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td>16</td>
<td>48</td>
<td>0-18496</td>
<td>32</td>
</tr>
</tbody>
</table>

Control – type of RTS, CTS, if using a distribution system, other bits for management
Duration – duration length (sender calculates how long to send based on size and data rate); others who read this frame can update their own counters for how long to wait before trying to send
Addr1 – target
Addr2 – source (or immediate sender if on a DS)
Addr3 – intermediate destination (AP)
SeqCtrl – frame #
Addr4 – original source
Payload (likely IP packet that has the length of the packet embedded)
CRC – cyclic redundancy check

Suppose node A is attached to AP1 and node B is attached to AP2. A sends to B. Then,
Addr1 = B
Addr2 = AP2
Addr3 = AP1
Addr4 = A

801.11 communicates in the unlicensed regions: 2.4 Ghz, 5 Ghz (several channels at these frequencies)

Also uses inter-frame spacing between frames to allow others to “get in” to send and to reduce the number of collisions. The spacing is different for acks, CTS frames (shorter SIFS) than for data frames (longer). There are two different spacers for data: time-bounded data versus asynchronous data.

Other ways to share:
**Bluetooth** – divide into numbered time slots for transmissions, master/slave model

Slaves only communicate with master, not other slaves.

Time-division multiplexing (time is broken into segments and each device gets a change to send at these time intervals.)

A frame takes up 1, 3, or 5 time slots
Up to seven slaves (more parked slaves can exist); slaves can only send during even-numbered time slots in response to the master. This gives the master control during the even slots and who has the next turn.

Because communication is at 2.45 Ghz (unlicensed), it uses a spread-spectrum technique to deal with interference. Have you ever heard your neighbor’s cordless phone calls?
To avoid this situation, Bluetooth uses frequency hopping over 79 channels, each for 625 us at a time, also is useful for setting its time slot duration. Use a pseudorandom # generator to keep the master and slaves using a consistent sequence of channels.

**CDMA (cellular networks)**

Have potentially many devices wanting to transmit data to the same base station. Yikes – how does the receiver handle all the signals together?

Think of our options and people talking in a room:
- Time –division (each person talks in turn)
- Frequency-division (each person talks at a different pitch)
- Code-division multiple access (each person talks in a different language)

3G mostly employs CDMA (some uses TDMA)

Each sender gets a unique code (chipping code) to use to encode its data. This chipping code also creates data redundancy (a nice benefit since there’s a lot more lost bits / transformed bits when the signal is carried through space rather than a wire).

Code is XORed with the sender’s data. If it is a 16-bit chip code, then each sender’s bit is XORed with 16 code bits to create the data that is sent. Then, the receiver uses each chipping code to decode the added signals from all senders to extract each sender’s unique sequence. It is a bit like decryption: other senders’ codes would make the data look like noise where the correct sending code makes the bits come out clean again.

**Review: dealing with interference:**

Frequency hopping: avoid interference by hoping that two communications are not sharing the same frequency at the same time (cordless phone problem)

Pro: _______________________________

Con: _______________________________

CDMA: let the interference happen, but assign codes so that each real signal can be gleaned from the noise

Pro: _______________________________
Con: _________________________________

Note: 802.11 some protocols used the chipping codes and some used frequency hopping and some used frequency multiplexing.
Each user gets a unique chipping code (aka chip, aka code). Here, to keep things simple, each chipping code is 6 bits long. Notice that each code is different. Mathematically, these codes are orthogonal to each other when they are thought of as vectors. Each user will be assigned a unique code.

To encode the data, each 1 bit is represented by sending a positive code $v$. Each 0 bit is represented by sending a negative code $-v$. For example, if the data is 1101, as above and the chipping code is $(1, 1, -1, 1, 1, -1)$ as user C, then the data would be encoded as the transmitted vector $(1, 1, -1, 1, 1, -1, 1, 1, -1, 1, 1, -1, 1, 1, -1, 1).$ The first six values are the chip code, the second six values are the chip code, the next six are the negative chip code, and the next six are the chip code. Notice that now it takes 6 values to represent a single bit (this is where the sharing comes in).

Activity 1: What would the signal from User B look like for the bit sequence 0011?
Suppose all three users send three bits of data (18 values).

User A sends 110.
User B sends 001.
User C sends 010.

Let’s see how these combine:

<table>
<thead>
<tr>
<th>User</th>
<th>Chipping code</th>
<th>Bits</th>
<th>Transmitted vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 -1 -1 1 -1 1</td>
<td>110</td>
<td>1 -1 -1 1 -1 1 -1 1 -1 1 -1 1 -1</td>
</tr>
<tr>
<td>B</td>
<td>1 1 -1 -1 1 1</td>
<td>001</td>
<td>-1 -1 1 1 -1 -1 1 -1 1 -1 1 -1 1 -1 1 -1</td>
</tr>
<tr>
<td>C</td>
<td>1 1 -1 1 1 -1</td>
<td>010</td>
<td>-1 -1 1 -1 -1 1 1 1 -1 1 -1 -1 1 -1 1 -1</td>
</tr>
</tbody>
</table>

Thus, the receiver will get the sum of these as the transmitted vector:

| Total | -1 -3 1 1 -3 1 1 -1 -1 3 -1 -1 -1 1 1 -3 3 -1 |

How will the receiver determine the bits from each sender?

Take each chipping code and multiply it with every 6 bits of the composite signal:

Let’s look at the first set of the composite sequence:

Multiply each of these bits with A’s chipping code:
-1 3 -1 1 1 1

Then add these values together: 6
Because it is non-negative, we interpret it as bit 1.

Let’s look the first set of the composite sequence with B’s chipping code:
-1 -3 -1 -1 -3 1
Adding the values: -8
Because it is negative, we interpret it as bit 0.

Now, let’s look at the first set of the composite sequence with C’s chipping code:
-1 -3 -1 1 -3 -1
Adding the values: -8
Because it is negative, we interpret it as bit 0.
Activity 2: You complete the decoding of the second bit from each sender.

<table>
<thead>
<tr>
<th>Sender</th>
<th>Second bit composite sequence multiplied with chipping code</th>
<th>Total</th>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Assume we have the network topology above. Nodes are represented by letters and switches are represented by X’s and numbers. Assume ports are identified by the node/switch that is the outgoing neighbor.

Each switch has a forwarding table. For example, the forwarding table for X2 is:

<table>
<thead>
<tr>
<th>Dest</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>X1</td>
</tr>
<tr>
<td>B</td>
<td>X1</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>E</td>
<td>X3</td>
</tr>
<tr>
<td>F</td>
<td>X3</td>
</tr>
<tr>
<td>G</td>
<td>X3</td>
</tr>
</tbody>
</table>

What is the forwarding table for X3?

<table>
<thead>
<tr>
<th>Dest</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
</tbody>
</table>
Option 2: Virtual Circuit Switching

Each switch has a table with the following information: Incoming Port, VCI in (Virtual Circuit ID in), Outgoing Port, and VCI out

In port/VCI in is a unique pair
Out port/VCI out is a unique pair

Assume A is sending to F. What do the tables at the switches look like?

X1:

<table>
<thead>
<tr>
<th>Incoming Port</th>
<th>VCI in</th>
<th>Outgoing Port</th>
<th>VCI out</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>X2</td>
<td>8</td>
</tr>
</tbody>
</table>

X2:

<table>
<thead>
<tr>
<th>Incoming Port</th>
<th>VCI in</th>
<th>Outgoing Port</th>
<th>VCI out</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>8</td>
<td>X3</td>
<td>10</td>
</tr>
</tbody>
</table>

X3:

<table>
<thead>
<tr>
<th>Incoming Port</th>
<th>VCI in</th>
<th>Outgoing Port</th>
<th>VCI out</th>
</tr>
</thead>
<tbody>
<tr>
<td>X2</td>
<td>10</td>
<td>F</td>
<td>7</td>
</tr>
</tbody>
</table>

Note that the outgoing ID is the VC in for the next hop.

Sequence of Events for Sending:
When A wants to send to F:
1. Put the ID as 5 in header of packets destined to F

Then, S1 gets packet:
1. Looks up values 5 and A in table
2. Removes 5 as the VC ID in header
3. Puts 8 as the VC ID in header
4. Forwards packet along outgoing port X2.

How do VCIs get assigned?
Signal:
1. A sends a startup message to switch 1 with destination F
2. Switch 1 forwards message to switch 2 with an unused VCI, fill in incoming and outgoing port entries
3. Switch 2 forwards message to switch 3 with an unused VCI, fill in ports
4. Switch 3 forwards message to F. F chooses a VCI.
5. F sends ack to Switch 3 with VCI, so switch 3 uses it for its outgoing ID.
6. Switch 3 forwards ack with the VCI it used in table.
7. ....
8. Ack gets back to A and a virtual connection is created.

Option 3: Source Routing

If A sends to F:
1. In header, A puts the sequence F – X3 – X2
2. Then, each node looks at last address for forwarding and rotates the entry order (moves last to first). So, then X1 would see the header, rotate it to X2 – F – X3 and send it to X2.
**VCI Practice**

Assume hosts are connected in a switched network shown below. What are the virtual circuit tables for the switches after each connection is established. Assume the sequence of connections is cumulative (concurrent). Assume the VCI ID is chosen such that it is lowest unused VCI on each link, starting with the lowest VCI ID as 0. Assume that a VCI is consumed for BOTH directions of a virtual circuit.

<table>
<thead>
<tr>
<th>Connection</th>
<th>Switch</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Port</td>
<td>VCI</td>
<td>Port</td>
</tr>
<tr>
<td>D to H</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>B to G</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Host D connects to Host H
Host B connects to Host G
Host F connects to Host A
Host H connects to Host C
Host I connects to Host E
Host H connects to Host J
<table>
<thead>
<tr>
<th></th>
<th>4</th>
<th>3</th>
<th>1</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F to A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H to C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I to E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H to J</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Suppose B1 is a bridge between two LANs (one containing A, B, and C and the other containing D, E, and F).

How does a bridge know when to forward packets?

Let's look at B1's forwarding table:

<table>
<thead>
<tr>
<th>Host</th>
<th>Port</th>
<th>Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>---</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>F</td>
</tr>
</tbody>
</table>

At first, there are no entries.

1. A sends to D.
   B1 adds entry (A, 0, 10) to table
   Forwards packet to LAN2
2. F sends to E
   B1 adds entry (F, 1, 10) to table, previous entry timeout decrements
   Forwards packet to LAN1 (does not know where E is)
3. B sends to A  
   Adds (B, 0, 10) to table. Destination A is in table, so does not forward.

4. A sends to B  
   Hosts A and B are both in table, so does not forward to LAN2, resets timeout

5. E sends to F  
   Adds (E, 1, 10) to table, Destination F is in table, so does not forward.

...
Practice:

Give forwarding tables for each bridge after the following transmissions:
D to C
C to D
A to C
Let’s look at this example. What’s different?

Why might we want or have cycles in a LAN?

Consider a packet that is sent from LAN J to LAN J, but B4 does not know the destination is on LAN J. What happens?

Need to find a spanning tree in the network to know which bridges to use for forwarding.
Spanning Tree Algorithm (developed by Radia Perlman, full paper of this algorithm is on Moodle)

Assume we have switches arranged like this:

Are there cycles?

Draw dark edges on the figure above to create a spanning tree.

(Note: Prim’s or Kruskal’s from data structures course can work if we know the entire graph; unfortunately, in a distributed system, we do not know the entire graph).
Idea of algorithm:
1. Elect root node (use lowest address)
2. Grow tree as shortest distances from the root bridge
   - Break ties with lowest address
   - Bridges send config messages over ports for which they are the best path
   - Turn off ports that are not on best paths
3. LAN uses its designated bridge (one with port still active) and the designated bridges do the forwarding across LANs

Algorithm:
1. Each bridge believes it is the root
   - When learn not the root, stop sending config (hello) messages
   - Forward root’s config message with # hops incremented by 1
   - Records best config for each port
2. When not a designated bridge, stop forwarding config messages
3. Real root sends config messages periodically
4. If bridge does not receive a config message in a certain period of time, assume topology has changed and start sending config messages claiming to be the root

Assume Config (hello) message is formatted: (root ID, # hops, send ID)
[also has age and port info, but will keep it simple for this example]

Example:
Assume no bridge has any info about any other bridge in network. Let’s look at B3:

1. B3 sends (B3, 0, B3) to B5 and B2 [claiming to be root]
2. B3 receives (B2, 0, B2) and (B5, 0, B2) from B2 and B5, respectively. Since B2 is < B3, B3 accepts B2 as root
3. B3 sends (B2, 1, B3) to B5 to forward message [note that the #hops is incremented]
4. B3 receives (B1, 1, B2) from B2 and (B1, 1, B5) from B5. Since B1 < B2, B3 accepts B1 as root.
5. B3 could send (B1, 2, B3) to B2 or B5, but it does not since it is nowhere the “shortest path” from B1. So, B3 is not a designated bridge.
6. B3 receives (B1, 1, B2) from B2 and (B1, 1, B5) from B5 again, so network is stable. B3 turns off data forwarding to LANs A and C.
A. Does this algorithm form a tree? Give tree in figure above by highlighting the links that would be active.

B. How do the config messages clog the network?

C. What happens when a bridge fails?

D. How could a broadcast message be sent to all nodes in the network?

E. What are the limitations of using this spanning tree algorithm?

Algorhyme, by Radia Perlman, 1985

I think that I shall never see
A graph more lovely than a tree.
A tree whose crucial property
Is loop-free connectivity.
A tree that must be sure to span
So packets can reach every LAN.
First, the root must be selected.
By ID, it is elected.
Least-cost paths from root are traced.
In the tree, these paths are placed.
A mesh is made by folks like me,
Then bridges find a spanning tree.
CS 445: IP (Internet Protocol)

Up to now, looked at creating LANs and switches connecting LANs.

Goal: support scalability, heterogeneity, bandwidth control (routing)

Build internetworks, such as

![Network Diagram]

Want to connect networks (of varying types)

Routers: forward packets + build forwarding tables

IP Features:
1. Supports end-to-end delivery between hosts
2. Uses common addressing
Routers must implement all link level protocols to which they are connected, in addition to implementing IP.

Ex: R1 must implement Ethernet and 802.11 (wireless)
Ex: R2 must implement Ethernet and PPP
Ex: R3 must implement Ethernet and PPP

Let’s say H5 wants to send to H8 with reliable delivery:

On H5:
  App -> TCP -> IP -> 802.11

R1:
  802.11 -> IP -> 802.3

R2:
  802.3 -> IP -> PPP

R3:
  PPP -> IP -> Eth

Packet from H5 to R1:
Wireless Header | IP | TCP | Any other header by app | Payload for app | CRC (wireless)

Then R1 strips the wireless header and CRC, uses IP header, slaps on 802.3 header and Ethernet’s CRC [Notice that the TCP and app-level headers are untouched]

Let’s look at the services IP provides: (similar to postal service)

- Connection-less
- Best effort delivery (unreliable)
- Packets routed independently
- Address scheme is global (hierarchical, 32 bits long in IPv4, 128 bits long in IPv6)

In contrast to land-line telephone:

- Connection-oriented
- Signalling to establish connection
- Routed the same way for duration of connection
CS 445: Internet Protocol (Packet, Fragmentation and Reassembly)

4. Application

3. Transport

2. Internet

1. Link

IP is “lowest common denominator”
- Asks little of lower-layer protocols
- Gives little to higher layer services

IP packet format:
- Version (4) | HLen (4) | TOS (8) | Length (16)
- Ident (16) | Flags (3) | Offset (13)
- TTL (8) | Protocol (8) | Checksum (16)
- SourceAddr (32)
- DestAddr (32)
- Options | Pad
- Data

Most IP packets have a header of 20 bytes (5 4-byte words). Here are the reasons for the fields:

<table>
<thead>
<tr>
<th>FIELD</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>IP version #</td>
</tr>
<tr>
<td>HLen:</td>
<td>size of header in words (4-byte chunks); default is 5 words</td>
</tr>
<tr>
<td>TOS:</td>
<td>type of service (now is split into two fields: 6 bits for differentiated services and 2 bits for congestion notification)</td>
</tr>
<tr>
<td>Length:</td>
<td>packet (fragment) size in bytes</td>
</tr>
<tr>
<td>Ident:</td>
<td>ID of packet for reassembly</td>
</tr>
<tr>
<td>Flags:</td>
<td>0 bit (always 0); DF bit is set for do not fragment; M bit is set to 0 for last fragment; M</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Offset:</td>
<td>used for fragment reassembly, measured in 8-byte chunks from start of original packet</td>
</tr>
<tr>
<td>TTL:</td>
<td>time to live hop count; default is 64</td>
</tr>
<tr>
<td>Protocol:</td>
<td>TCP = 6, UDP = 17, specifies higher level protocol</td>
</tr>
<tr>
<td>Checksum:</td>
<td>error detection</td>
</tr>
<tr>
<td>SourceAddr:</td>
<td>IP address of sender</td>
</tr>
<tr>
<td>DestAddr:</td>
<td>IP address of destination</td>
</tr>
<tr>
<td>Options/Pad:</td>
<td>Can have optional extra header information</td>
</tr>
</tbody>
</table>

**Fragmentation and Reassembly Example:**
Suppose host A sends data to host B through router R. The network from A to R is FDDI (fiber) for the link layer and the network from R to B is Ethernet.

MTU on fiber: 4500 B  
MTU on Ethernet: 1500 B

Suppose the original packet sent by A has ident = 123 and contains 3000 bytes of data.  
M bit = 0; offset = 0

Now what happens when it gets to R?

**Let’s look at fragmentation and reassembly:**

Issue: As packets travel from one network to another, the maximum size frame might change. For example, the max size for FDDI is 4.5 KB and Eth is 1.5 KB.

Solution: Allow packets to be fragmented and reassembled at the destination.

IPv4: fragmented on demand  
IPv6: learn smallest frame size, source fragments into this size

MTU = maximum transmission unit (max size of frame for a network), note that IP packet header is part of the data size of the frame

In IP, if a fragment is lost, destination cannot reassemble. Asks for entire original packet again (not the fragment #), since source has no idea what networks have been crossed.
Example:
FDDI ------------ R --------------- Ethernet

MTU = 4500 KB  MTU = 1500 B

Let’s say a packet goes from FDDI to the Ethernet through R1 with size 3020 bytes (20 bytes header and 3000 bytes data)

On FDDI:

Ident = 123, M bit = 0, offset = 0
Data is 3000 bytes

Then R gets it/ Cannot forward as is, so it fragments it into 3 packets:

Ident = 123, M bit = 1, offset = 0
Data is 1480 bytes

Ident = 123, M bit = 1, offset = 185  [Note that 1480 / 8 = 185]
Data is 1480 bytes

Ident = 123, M bit = 0, offset = 370
Data is 40 bytes

What would happen to these packets if delivered to a network with MTU of 1000 B?

Observations of fragmentation:
- can fragment fragments (does not reassemble at intermediate routers)
- end host has pressure to do reassembly
- must timeout reassembly in case of missing fragments
- routers must use resources to fragment
- must re-send entire packet if one fragment is missing
IPv6:
Learn smallest MTU along path before sending
   -No more fragment burden on routers, but still reassembly at end host

Can also do this in IPv4 using ICMP (set DF bit to get feedback messages) to test for smallest MTU.

IP Addresses:
3 types (class A, B, C)
A:
   0 | network (7 bits) | host (24 bits)
B:
   10 | network (14 bits) | host (16 bits)
C:
   110 | network (21 bits) | host (8 bits)

Idea: All hosts on the same network would share the same network number as part of the address. Routers store network addresses to know which interface to use for forwarding.

Router’s two main jobs:
   Fragment packets
   Forward packets (let’s look at this now)

Forwarding Algorithm:
   If network address of destination is a network on one of my interfaces
      Deliver packet to that interface.
   Else if network address of destination is in my forwarding table,
      Deliver to next hop router
   Else
      Deliver packet to default router
**Fragmentation Practice:**

Assume A forwards IP packets to router R with a MTU of 3000 B. Then R sends to B with a MTU of 1500 B. The MTU includes the IP header of 20 B. The original data that A needs to forward is 4200 B. Show the packet fragments along the two links.

<table>
<thead>
<tr>
<th>Start of header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ident = 22</td>
</tr>
<tr>
<td>0 Offset = 0</td>
</tr>
<tr>
<td>Rest of header</td>
</tr>
<tr>
<td>4200 data bytes</td>
</tr>
</tbody>
</table>

Fragments from A to R:

Fragments from R to B:
CS 445: ICMP, DHCP, ARP

ICMP = Internet Control Message Protocol
Runs in addition to IP on routers/switches

Function: handles errors, queries for status of network

Host -> R1 -> R2 has an error
R2 sends ICMP packet to Host

Example Messages:
Destination unreachable
Packet needs fragmenting
TTL reached 0
IP header checksum failed
Redirect – tell host there is a better route

To keep ICMP messages from getting out of control, do not send messages in response to ICMP packets, broadcast packets, fragments other than first fragment

Achieving scalability through addressing

Issue: To scale, each host needs a unique address
  ➔ could lead to huge forwarding tables (addresses are hierarchical, so we can aggregate routes)
  ➔ IP addresses reflect location in topology (unlike MAC addresses), interfaces on same network have same prefix (network address)

Solution: IP addresses have network part and host part

Issue: How does a host get an IP address?

Option 1: System administrator does assignment (not very dynamic, a bit hard to maintain)

Option 2: DHCP = Dynamic Host Configuration Protocol
  Assign hosts IP addresses on demand
  Maintains table of IP address – hardware address pairs (if want to give same IP address to machines when connected)
  Can also dole out addresses for lease
DHCP server on a network and other hosts can send messages to it. The server maintains a pool of available addresses.

Example:
Host A is on Ethernet with DHCP server.

A sends a discover message to 255.255.255.255 with its hardware address.
The DHCP relay listens for “discover” packets and forwards them unicast to server.
Server looks for unassigned IP address and sends message to A with the IP address.

Host's responsibility to renew address (expires after so much time)

Issue: Now that a host can get an IP address, how do actual packets get to hosts?

Remember: MAC address used for destination on Ethernet, but packet has IP address as destination in header.

Solution: ARP (Address Resolution Protocol)

Each host builds up a table of mappings to map IP addresses to MAC addresses, so can use link level protocol to specify the MAC address in the link header.
Example: A and B are on same network.

A wants to send IP packet to B. A knows the IP address for B, but not its MAC address. A checks to see if B’s IP address and MAC address pair is in local table. If so, uses the MAC address. If not:

1. A sends broadcast query for IP address X.
2. Node B with assigned IP address X replies with MAC address M. That’s me.
3. A caches (X, M) pair
4. B caches A’s IP address and MAC address (probable future communication)
5. Other hosts seeing A’s message with an entry for A refresh caches (entries timeout after about 15 minutes)

Now, forwarding involves looking up MAC addresses, too.

If host and (dest IP address, MAC address) pair is in ARP table, then deliver locally. Otherwise, use MAC address of router to forward IP packet to destination.

Then, the router checks the IP address, its forwarding table, and forwards it to the next hop. Note that the next hop’s MAC address will also be stored in the router’s forwarding table, so the right link-layer header can be attached.
CS 445: Distance Vector Routing

What is routing? Figuring out to whom to forward packets and update forwarding table
What is forwarding? The process of selecting the next hop / interface as packets are processed

Note that routing is a process that must regularly take place in a network to maintain the accuracy of the forwarding tables. Generally, routing information is stored separately from the forwarding tables. Forwarding tables are often optimized for fast look-up.

Goals in routing:
1. Want to choose best path
2. Want to scale (not too many messages sent around network)
3. Want to adapt to changes/failures in network

Two General Options:

Link State Routing
Tell world about your neighbors

Distance Vector Routing
Tell your neighbors about the world

Option 1: Distance Vector Routing
Assume: routers know neighbors and costs to neighbors (same assumption as in link state)

Algorithm for router:
- Initialize table with neighbors and costs. If not a direct neighbor, distance is infinity.
- Periodically send copy of distance vector to neighbors

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>inf</td>
<td>1</td>
<td>1</td>
<td>inf</td>
<td></td>
</tr>
</tbody>
</table>

(if A is connected to B, C, E, and F)

When receive a distance vector packet, router looks up packet’s destination costs + my distance to this neighbor cost to see if value is smaller. If so, update table entry.
Example:

Suppose A is doing distance vector routing. Initially, its vector would be:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>inf</td>
<td>1</td>
<td>1</td>
<td>inf</td>
<td>inf</td>
</tr>
</tbody>
</table>

A would send this to B, C, E, and F.

Next iteration:
A would get DV from B:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
</tr>
</tbody>
</table>

No update to A’s DV, since costs to B + cost to others is bigger

A would get DV from C:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
</tr>
</tbody>
</table>

A would update it’s DV, since cost to C + C’s cost to D is 2

A’s DV:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>inf</td>
<td>inf</td>
</tr>
</tbody>
</table>

A would get DV from F:
A would update its DV, since cost to F + F’s cost to G is 2

A’s DV:

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>C</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>F</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>F</td>
</tr>
</tbody>
</table>

Note, A would get a vector from E, too (but those distances are all infinity).

As A calculates costs, it updates its forwarding table:
CS 445: Link State Routing (option #2)

Assumption: Each router knows its neighbors and costs to the neighbors

Idea: Tell all routers your neighbors and have each router build up network topology, calculate shortest paths, and create forwarding tables.

Phase 0: Detect neighbors and costs of links
Phase 1: Share Information and Build Topology (flood information through network)
Phase 2: Compute shortest paths at each router
Phase 3: Build forwarding tables

What information needs to be in a link state packet?
- Router ID (sender)
- Neighbors and costs to the neighbors
- Sequence Number (so other routers can tell if this is new news or old news)
- TTL (so eventually packet dies and does not clog network forever)

Phase 1:
What does router do upon receiving a LSP?
- Uses info to recalculate shortest paths (if info changed)
- If this LSP has not already been flooded by me, flood the packet to all my neighbors

Assume no router has sent or received LSPs.

A sends LSP (advertises that it is 1 hop from B and 1 hop from C)
B and C receive the LSP and each forwards it along all links, so D gets it.*
D receives the LSP and forwards it to B, C, and E.*

*can optimize by not forwarding packet directly back to node from which it was received

The SeqNum tells us if the LSP is new info. SeqNum resets to 0 after a node goes down. When TTL is 0, stop forwarding the packet.
Phase 2: Compute shortest paths

Use Dijkstra’s Algorithm to compute shortest paths from a node. (Review from data structures)

Notation:

- \( N \) set of all nodes
- \( M \) set of nodes for which we think we have the shortest path
- \( s \) node executing algorithm
- \( L(i, j) \) Link cost between nodes i and j, cost is infinity if no edge
- \( C(i) \) Cost of path from \( s \) to i

// initialization
\( M = \{ s \} \)

// M is the set of nodes already considered, initialized to \( \{ s \} \)

For each node \( n \) in \( N - \{ s \} \):
\( C(n) = L(s, n) \)

// find shortest paths
while(true):
    Unconsidered = \( N - M \)
    If Unconsidered == {}, break
    \( M = M + \{ w \} \) where \( C(w) \) is smallest and \( w \) is in Unconsidered
    For each \( n \) in \( Unconsidered - \{ w \} \):
        \( C(n) = \min(C(n), C(w) + L(w, n)) \)

Example:
Assume the following topology is learned through the LSPs:

![Diagram of the network](image-url)
Note that the costs are labeled along the edges and the edges are directed.

// initialize
M = \{s\}
C(A) = 10
C(B) = inf
C(C) = 5
C(D) = inf

// shortest paths: first iteration
U = \{A, B, C, D\}
M = M + \{C\} since C(C) is lowest
// update costs
C(A) = 8 \[C(C) + L(C,A) = 8\] Forward A through C
C(B) = 14 \[C(C) + L(C,B) = 14\] Forward B through C
C(D) = 7 \[C(C) + L(C,D) = 7\] Forward D through C

// next iteration
U = \{A, B, D\}
M = M + \{D\}
// update costs
C(A) = 8 \[C(A) same as before\]
C(B) = 13 \[C(D) + C(D,B) = 13\] Use path through B

// next iteration
U = \{A,B\}
M = M + \{A\}
// update costs
C(B) = 9 \[C(A) + C(A,B) = 9\] Use path through C and A

// next iteration
U = \{B\}
M = M + \{B\}

// END //
Phase 3:
Now, we build up the forwarding table. In this example, all packets would be forwarded to C.

<table>
<thead>
<tr>
<th>Dest</th>
<th>Cost</th>
<th>NextHop</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0</td>
<td>--</td>
</tr>
<tr>
<td>A</td>
<td>8</td>
<td>C</td>
</tr>
<tr>
<td>B</td>
<td>9</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>7</td>
<td>C</td>
</tr>
</tbody>
</table>
CS 445: Border Gateway Protocol (BGP)

Goal: build large networks; forward packets across smaller networks; scale gracefully

Hmmm – think about link state routing. Do you want packets flooding the entire Internet?

Hmmm—think about distance vector routing. How big would those distance vector packets be for the entire Internet?

Yikes. We need something better to scale. Fortunately, we actually have some hierarchical structure in the way networks are interconnected. Let’s look at how the Internet might be connected:

![Diagram of network structure]

Each entity is an Autonomous System (AS). We need a way to communicate paths (routes) from one AS to another.

Corp A is a stub AS. Backbone is a transit AS. Corp B is a multihomed AS (connected to 2 ASes, but refuses to carry traffic between the ISPs).
A router that connects one AS to another is called a **border router or gateway**. Use intra-domain routing with AS1 to move packets within AS1 and use inter-domain routing between ASes.

**Border Gateway Protocol**

Responsibilities of border routers:
1. Summarize / advertise internal routes to external neighbors (Speakers)
2. Get internal network addresses from external neighbors
3. Use policies for determining best route (could be based on cost, contracts between ASes, geographically closest peering point)

BGP Features:
1. Path vector routing
2. Application of policy
3. Operates over reliable transport (TCP)

1. Path Vector Example:

   ![Path Vector Diagram]

   The speaker for AS60 would get the path vector (30-20-40) from AS30 and learn that AS40 has network addresses A, B, and C. Field width for AS is 32 bits.

   Why is announcing the entire path important?
A, B, and C are addresses
Remember: route announcements move in opposite direction to the traffic

2. Policies

Chosen by AS. Negotiated by ASes.

Routes may depend on owner, cost, contracts between ASes. The policy dictates routes to choose and which routes will be advertised to other speakers. Border routers select the best path of the ones they hear, but BEST may not always mean “shortest”. It may be BEST according to real money costs.

Example policies: AS X does not provide transit for AS A (A could only use X for final delivery). AS X prefers not to use AS A since A is unreliable. AS X and AS A carry traffic for each other.

ISP – Customer Relationship
Responsibility of ISP:
- Sell transit to customers
- Announces path to all other IP addresses to customer, so customer can reach others

Customer:
- Announces path to their IP address (prefix) to ISP, so ISP can advertise via BGP
ISP – ISP Relationship (Peer)
Peers for mutual benefit (use me and I will use you for free)
Announce paths to each other about own customers

(Tier 1 ISPs are considered backbone providers, more global reachability)

As a Customer, you might connect to two ISPs

ISP 1        ISP 2
    /\        /\  
   Customer

As the customer, you could control which ISP you use for outgoing paths (based on announcements from the ISPs).
You cannot control which ISP others use to connect to you. Both ISP 1 and ISP 2 will announce they can reach you and the path vectors and policies will dictate which ISP outside traffic will use you reach you.
You are router D. A, C and F are “speaking” with other gateways using BGP. Routers A, C, and F provide interior gateway messages (sometimes called interior BGP) that flood to the other routers.

Suppose you get the following interior gateway messages from the gateway routers:

From A: Use me to get to 20.0/16 and 192.48.8/24
From C: Use me to get to 12.8.20/24
From F: Use me to get to 128.55/16

Build your BGP table (contains network prefixes and border router destination):

<table>
<thead>
<tr>
<th>Prefix</th>
<th>BGP Next Hop</th>
</tr>
</thead>
</table>
Meanwhile, an intra-domain routing protocol is also happening, so you know how to forward packets to each router in the network. Complete the following forwarding table:

<table>
<thead>
<tr>
<th>Router</th>
<th>Internal Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>------</td>
</tr>
<tr>
<td>E</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

Now, with the two tables, you can build the actual forwarding table for the network addresses. Figure out the ultimate destination (border router) and then use the forwarding table immediately above to figure out the next hop:

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Internal Next Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.0.16</td>
<td></td>
</tr>
<tr>
<td>12.8.20/24</td>
<td></td>
</tr>
<tr>
<td>128.55/16</td>
<td></td>
</tr>
<tr>
<td>192.48.8/24</td>
<td></td>
</tr>
</tbody>
</table>
CS 445: IPv6

Main motivation to develop version 6: need more global unicast addresses (32 bits per address in IPv4 is too limiting)

IPv6 addresses are 128 bits long -> $3.4 \times 10^{38}$ unique addresses (with 100% efficiency)

Packet Format:

Version (4) | Traffic class (8) | Flow label (20)  
Payload length (16) | Next Header (8) | Hop Limit (8)  
Source Address (first 32 bits)  
Source Address (second 32 bits)  
Source Address (third 32 bits)  
Source Address (fourth 32 bits)  
Destination Address (first 32 bits)  
Destination Address (second 32 bits)  
Destination Address (third 32 bits)  
Destination Address (fourth 32 bits)  
Next header / data (rest)

Nice. The packet header is actually simpler than IPv4.

Version: same field (specifies version 4 or version 6, so switch/router knows how to interpret the rest of the packet)
Traffic class: help with quality of service (we will see this at the transport layer)
Flow label: help with quality of service (we will see this at the transport layer)
Payload length: length of packet in bytes (includes header)
Next header: value specifies if/which special headers are included after the destination address; if no special header, then it is the value of the IPv4 Protocol field (specifies TCP or UDP)
Hop Limit: time to live count by hops (just renamed to be clear that the TTL is based only on hop count)
Source Address: 128 bits for sender
Destination Address: 128 bits for receiver
Next Header: if there are special optional headers, they are here
Data: actual data carried in the packet

1. What is missing that we had in IPv4?
**IPv6 fragmentation header (ID 44 in the NextHeader field):**

Next header (8 bits) | 00000000 | offset (13) | 00 | M bit (1)
Ident (32 bits)

So, the Next header field would contain the ID for the next header in the packet. If no such header exists, the value is set to the transport layer protocol (such as 6 for TCP).

**IPv6 Addresses**

128 bits long now...whew, we have a lot of space with which to work now

Certain prefixes of IPv6 addresses are meaningful:

<table>
<thead>
<tr>
<th>PREFIX</th>
<th>SIGNIFICANCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>000...000 (128 bits)</td>
<td>Unspecified</td>
</tr>
<tr>
<td>000...001 (128 bits)</td>
<td>Loopback</td>
</tr>
<tr>
<td>11111111 (8 bits)</td>
<td>Multicast – address a group of hosts together; start with byte of all 1s</td>
</tr>
<tr>
<td>1111111010 (10 bits)</td>
<td>Link-local addresses: used by devices that need to networked within a private domain (for routing/forwarding internally within a network), but that do not need to communicate globally in the Internet</td>
</tr>
<tr>
<td>All else</td>
<td>Global unicast addresses</td>
</tr>
</tbody>
</table>

Notation for IPv6: hexadecimal representation (each digit represents one 4-bit chunk)

Example:

For long runs of zeros, abbreviate with ::

Original address:
35AB:0000:0000:0000:0000:0000:45CC:2211

Abbreviated address:
35AB::45CC:2211 //can only abbreviate one run of zeros

Can embed IPv4 addresses in IPv6 as follows:
::FFFF:128.46.3.25

The last 32 bits are written in decimal IPv4 notation and the double colon at the beginning means a run of zeros at the beginning.

2. How would you organize the 128 bits of address space?
Consider grouping IPv6 addresses based on AS/ISP:

Pros:

Cons:

Consider grouping IPv6 addresses based on geography:

Pros:

Cons:

So, IPv6 uses a combination of these practices:

Internet Assigned Numbers Authority (IANA) allocates address space to each regional registry:
- RIPE NCC (EMEA)
- APNIC (Asia Pacific)
- ARIN (North America)
- LACNIC (Latin America)
- AfriNIC (African Region)

www.iana.org

So, IPv6 addresses that can be assigned as global unicast by IANA:
2000::/3 (meaning the first 3 bits are 001)

Generally, the first part of the address indicates the regional registry.
The second part indicates the ISP.
The third part indicates the site prefix (actual customer/company).
The fourth part may be a subnet, if customer/company wants to sub-divide network.
The fifth part is the host part: it is 64 bits long, so this can support the full MAC address. [the 48-bit MAC address is zero-extended]
Other benefits of IPv6
1. Autoconfiguration

2. Routing paths

3. Security

Dealing with IPv4 and IPv6 simultaneously

Clearly, we have not made it to an IPv6-only world. Some routers can handle IPv6, while some can only handle IPv4.

Dual stack routers: Those that can process v6 and v4 read the version field and handle the arriving packet with the appropriate interpretation.

Suppose you are A and your network can handle IPv6. Suppose your destination E can also handle IPv6. Hmmm, but in the middle, some routers can only do IPv4.

![Diagram]

Need to figure out how to take IPv6 packet and get it across the network. Use tunneling (encapsulate IPv6 packet as payload for IPv4 packet). In figure above, the packet crosses the first link fine. On the second link, the router B must take the IPv6 packet and jam it into the data portion of an IPv4 packet. It must create the IPv4 header with the sender as B and receiver D (using IPv4 addresses). The packet is then sent to C. C views it strictly as an IPv4 packet and sees that the destination is D, so it forwards it to D. D is a router and not a final destination, so D knows that it needs to inspect the payload to determine
where to forward the packet. It strips off the IPv4 header and sends the payload of the packet to
destination E (which was the original IPv6 packet sent by A).

Today: about 22% of google Internet users use IPv6 packets. There is no “transition” day where all IPv4
routers/hosts turn off and IPv6 routers/hosts turn on. It will take some time to get all equipment to the
point where it can handle IPv6.


http://www.worldipv6launch.org/measurements/
CS 445: Network Address Translation

Common scenario: home computers use “private” IP addresses, NAT connects home to ISP using a single external IP address

Keep an internal/external table (IP address + TCP port)

What host thinks: 192.168.1.12 : 5523
What ISP thinks: 44.25.80.3 : 1500
   Need ports to make the mapping one-to-one since there are fewer external IPs

Caution: can only get incoming packets after an outgoing connection is set up; makes running servers with fixed IP addresses challenging; not so good with connection-less services; breaks applications that have dedicated IP addresses (example: FTP)

Benefits: relieves IP address pressure, many home devices behind NATs and only communicate within the home, easy to deploy
**CS 445: Multicast (if time)**

1. Say the top node wants to send the same data (usually things like streaming media) to five of the other nodes in the network.

   In the Unicast model, what would happen?

2. What networked applications have a one-to-many service model?

3. What networked applications have a many-to-many service model?

**IPv4 Multicast Addresses**

- Start with 1110 (address range 224.0.0.0 to 239.255.255.255)
- 28-bits of network address: usually determined by out-of-band registration (for example, a web app has an explicit join procedure to a specific multicast address; an online game lets players join specific groups)

**Multicast Routing**

- Distance Vector (early approach): flood and prune idea
  1) Flood network with multicast packets
  2) Prune back networks with no hosts who want to receive packets for that multicast group
Routers only forward MC packets on links that form part of a shortest path somewhere (at least packets are not looping back to the sender and back to routers). But, goodness, that is still a lot of traffic that is unnecessary.

Think about the number of typical hosts in a multicast group versus the size of the Internet.

PIM-SM (Protocol Independent Multicast – Sparse Mode)
Instead of creating a giant tree of all networks and then pruning, routers explicitly join the MC distribution tree through JOIN messages.

Each domain has a designated Rondezvous Point (RP).

A shared tree is created (one tree per group to start, instead of one tree per sender).

Suppose we have the following connection of routers:

Here’s the sequence of events that happen in PIM-SM:

1) R3 sends Unicast JOIN message for multicast group G to RP.
2) R1 sees this message and creates entry in forwarding table (*, G, R3), meaning * for anyone can be the sender, G is the multicast address, and R3 is the next hop.
3) RP sees this message and creates entry in forwarding table (*, G, R1), meaning * for anyone can be the sender, G is the multicast address, and R1 is the next hop.
4) R4 sends Unicast JOIN message for multicast group G to RP.
5) R1 sees this message and adds outgoing entry to R4 for group G: (*, G, {R3, R4}). So, now R1 would forward a multicast IP addressed packet to R3 and R4.
6) Suppose sender S is connected to R5 and wants to send a packet to multicast group G. The packet has in its header address G as the destination.
7) R5 does not know how to forward this (no entry in forwarding table for group G), so sends to the RP, via tunneling. It slaps on a new IP header with RP as the address (embedding the original IP packet as the payload).

8) RP receives the packet and sees that it is destined for the RP. Since it is a router and not a host, it extracts the payload and sees that the packet is destined to multicast group G.

9) Then RP forward the extracted packet to R1. R1 receive it and see that it is for group G and then forwards it to R3 and R4.

OK, that works. But, what if sender S is sending lots of data (what if this is streaming video?). Tunneling and extracting data is extra work for the routers, so there are some optimizations:

10) When the RP gets the tunneled packet, it sends a JOIN to R5. R5 used its IP address in the tunneled packet, so RP knows who created the tunnel. This JOIN message has (S, G) in the body, meaning that sender S to group G are the packets you should forward.

11) R2 receives it and puts (S, G, RP) in its forwarding table and forwards the JOIN message to R5.

12) R5 receives it and puts (S, G, R2) in its forwarding table.

Now, when S sends to G, R5 can forward directly to R2, who forwards directly to RP, who then carries out the forwarding as before.

Also, with specific (sender, MC group, next hop) entries, each router can create source-specific trees, so the data paths from each sender are minimized.

4. Benefits of PIM-SM:

5. Drawbacks of PIM-SM:
CS 445: UDP

So far, we have covered:
- Link Layer [done]
- Network Layer [done]
- Transport Layer (starting today)

Thus far, we have concerned ourselves with protocols running within a network (link-layer such as Ethernet and network layer such as IP). Now, we will turn our attention to end-to-end protocols (what happens at the end-hosts). This brings us to the transport layer where there are two protocols: UDP and TCP. We’ll look first at UDP.

Why do we need transport-layer protocols? _____________________________

Best-effort model
- It can drop packets, it can re-order packets, duplicate packets could be received, packets have a max size, packets may have long delay through network

Hmmm, well – would you want your apps to have to deal with all these network issues?

What might we want in a service model at the transport layer?

1) Deliver only one copy of each message (packet)
2) Support arbitrarily long messages
3) Support synchronization between sender/receiver
4) Allow receiver to apply flow control (don’t sent me too much at once!!)
5) Allow multiple end-to-end processes on a host (in other words, I would like to use my email client, my web browser, my media player all at once)

UDP – User Datagram Protocol
- Provides demux support for hosts running multiple applications

SrcPort (16 bits) | DstPort (16 bits)
Length (16 bits) | Checksum (16 bits)

- Length – total length of packet in bytes
- Checksum calculated over UDP header, the message, and part of IP header (optional in IPv4, mandatory in IPv6)
Ports allow multiple applications to run on the same host. For example, DNS messages come in on port 53 and mail comes in on port 25.

How to learn port numbers?
- use well-known numbers that are published as standards
- port-mapper that accepts general connection and then replies with the right port number to use

UDP gives us unreliable service – if queues at the end host get full, packets are dropped.

So, let’s see what the host needs to do now:

App1       App2       App3
Port idx    port idy   porti idz
<>          <>          <>
<>          <>          <>
<>          <>          <>
<>

UDP (figures out which queue to put the incoming packet into based on the port number)

If the queue is full, the message is dropped.

UDP is connection-less:
- Messages may be lost, reordered, duplicated
- Limited message size (datagram size)
- Can send regardless of receiver state

Sockets, remember, allow apps to attach to the network at different ports.
### CS 445: Well-Known Ports / System Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Protocol</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>20, 21</td>
<td>FTP</td>
<td>File transfer</td>
</tr>
<tr>
<td>22</td>
<td>SSH</td>
<td>Remote login</td>
</tr>
<tr>
<td>25</td>
<td>SMTP</td>
<td>Email</td>
</tr>
<tr>
<td>53</td>
<td>DNS</td>
<td>Domain name service</td>
</tr>
<tr>
<td>80</td>
<td>HTTP</td>
<td>Web</td>
</tr>
<tr>
<td>110</td>
<td>POP-3</td>
<td>Remote email access</td>
</tr>
<tr>
<td>143</td>
<td>IMAP</td>
<td>Remote email access</td>
</tr>
<tr>
<td>443</td>
<td>HTTPS</td>
<td>Secure web</td>
</tr>
</tbody>
</table>
CS 445: TCP

TCP Segment format:

SrcPort (16) | DstPort (16)
SequenceNum (32)
Acknowledgment (32)
HdrLength in words (4) | 000000 | Flags (6) | AdvertisedWindow (16)
Checksum (16) | UrgPtr (16)
Options (variable)
Data

Flags:
SYN open
FIN close
RESET reset (receiver is confused and wants to abort)
PUSH signal receiver to notify process
URG urgent data
ACK acknowledgment (pay attention to that field)

TCP Properties
1. connection-oriented
2. multiple processes
3. reliable delivery
4. supports flow control (no over-running receivers)
5. in-order delivery

1. Connection-oriented
a. 3-way handshake to establish connection from sender to receiver

Sender

Receiver

SYN = 1, SeqNum = x

SYN = 1, ACK = 1, SeqNum = y, Acknowledgment = x + 1

ACK = 1, Acknowledgment = y + 1
Acknowledgment field identifies the next sequence number it expects to get.

Timer is scheduled for the first two segments; if no ACK received, segments are re-sent.

Starting sequence numbers are chosen at random (x and y), to minimize risk of duplicating segment numbers from a previous TCP connection between the two parties.

b. When one side is finished sending data, it sends a segment with FIN = 1

c. See below for state diagram for TCP connections. Edges are labeled with event/action pairs. States are labeled as rectangles.
2. Support for multiple processes

SrcPort and DstPort provide for multiple simultaneous applications, such as UDP

3. Reliable delivery

Uses the sliding window algorithm we saw earlier in the semester.

Sender

 Sender

 LastByteSent

 LastByteAced

 LastByteWritten

LastByteAced <= LastByteSent
LastByteSent <= LastByteWritten

Receiver

 Receiver

 NextByteReceived

 LastByteRead

 LastByteReceived

LastByteRead <= NextByteExpected
NextByteExpected <= LastByteReceived + 1

Receiver sends Acknowledgment = x [sequence number it is waiting for] means it has all segments up to segment (x-1).
4. Supports Flow Control

Do not want sender way ahead of receiver (do not overrun receiver’s buffer).

Sender adjusts size of sliding window based on feedback in the AdvertisedWindow field.

Receiver

Goal: ensure $\text{LastByteReceived} - \text{LastByteRead} \leq \text{maxRcvBuffer}$

$\text{AdvertisedWindow} = \text{maxRcvBuffer} - ((\text{NextByteExpected} - 1) - \text{LastByteRead})$

“All available space minus what is in the buffer”

Then, on the sender side:

Goal: $\text{LastByteSent} - \text{LastByteAcked} \leq \text{AdvertisedWindow}$ [do not get too far ahead]

$\text{EffectiveWindow} = \text{AdvertisedWindow} - (\text{LastByteSend} - \text{LastByteAcked})$
When ack arrives back to sender, advance LastByteAcked, updated AdvertisedWindow, and calculate EffectiveWindow. If EffectiveWindow > 0, send more data.
Example: to keep things simple, the advertised window is based on # of segments and not bytes

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>SequenceNum = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ack = 2, AW = 3</td>
</tr>
<tr>
<td>SequenceNum = 2</td>
<td></td>
</tr>
<tr>
<td>SequenceNum = 3</td>
<td></td>
</tr>
<tr>
<td>SequenceNum = 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ack = 3, AW = 2</td>
</tr>
<tr>
<td></td>
<td>Ack = 4, AW = 1</td>
</tr>
<tr>
<td></td>
<td>Ack = 5, AW = 0</td>
</tr>
</tbody>
</table>

At this point, the sender would stop sending segments. Sender waits and periodically sends segments with 1 byte just to get an ack back with an updated window size.

5. In-order delivery

Applications buffer data to read it sequentially (to match the order it was sent).

Note that TCP also provides single packet delivery (only one copy of each segment is kept in buffer) and can support arbitrarily long data streams.
CS 445: More Notes on TCP, Sequence Numbers, Advertised Windows, Re-transmissions

Questions that we will discuss today are below. In a small group (2 – 3 people), discuss these questions and formulate initial answers:

1. What are the implications of a limited size sequence number?

2. What are the implications of a limited size advertised window?

3. When should TCP segment transmissions be triggered (how much data to collect on sending host before sending it in a segment)?

4. When should a TCP segment be re-transmitted (how long should you wait for an ack)?
**Sequence Numbers**

32 bits for the sequence number in TCP, max wraps back around to 0. Field supports ~4 billion unique sequence numbers.

**Question 1:** Suppose you sent 1 byte per segment on a link with speed 100 Mbps. How long will it take for the sequence number to wrap around?

**Question 2:** What about a link with speed 2.5 Gbps? Is this long enough?

**Solution to limited Sequence Numbers:** Timestamp field of 32 bits indicates clock time of sender. If receiver gets two segments with the same sequence number, then it can differentiate them with the timestamp.

------------------

**Advertised Window** (16 bits)

**Question 3:** What is the largest number of bytes that the AW can advertise?

We would like to “keep the pipe full”, so it may be that the network can handle more data than what you found in question 3. Let’s look at some delay x bandwidth products (this gives us the size of the pipe) for various speeds with RTT of 100-ms:

<table>
<thead>
<tr>
<th>Bandwidth</th>
<th>Delay x Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 Mbps</td>
<td>18 kB</td>
</tr>
<tr>
<td>10 Mbps</td>
<td>122 kB</td>
</tr>
<tr>
<td>100 Mbps</td>
<td>1.2 MB</td>
</tr>
</tbody>
</table>
**Question 4:** Could the AW field support keeping the pipe full at 1.5 Mbps? How about 10 Mbps?

**Solution for limited AW field:** Use scaling factor for AW as an optional field. For example, if the optional field has the value of 16, it really means the AW is 16 * the number given in the AW field.

------------------

**When to trigger TCP segment transmissions**

**Question 5:** What is the worst case payload for a TCP segment in terms of header/payload ratio?

The best case would be to fill the TCP segment so that the TCP + IP + frame layer headers and the payload fit into a segment that is no bigger that the MTU.

Max Segment Size = MTU of network – size of headers

**Possible segmentation algorithm:**

If AW >= MSS,
  
  Accumulate MSS bytes of data and send one TCP segment

If application pushes data (the PUSH bit set),
  
  Package up bytes left in buffer and send one TCP segment

Else,
  
  What to do???

**Question 6:** Suppose the receiver sends an ack with an AW of MSS / 2. Should the sender send MSS / 2 bytes or wait until a full set of MSS bytes are collected?

**Question 7:** Now suppose the receiver sends an ack with an AW of 2 bytes. Should the sender send the 2 bytes or wait until having a bigger set of bytes?

Silly window syndrome: ack with a small window size – keeps a small “container” in the system for sending data.
Improvement:

**Nagle’s Algorithm:**
If AW >= MSS and data >= MSS, send full segment.
If app pushes data, send segment.
Else if there is un-acked data in flight,
  Buffer data until ack arrives
Else
  Send buffered data now

What is happening? Treat ack as a trigger (self clocking) to send more data, so wait for ack before sending data. Can send a small amount of data if there are no segments in flight.

-------------

**When to re-transmit a segment? (How long to set the timeout)**

In a point to point link, we could set the timeout based on the calculated RTT.
Hmmm, but in general, the segments are going through a network that can experience delays.

**Original TCP:**
Idea: keep a running average of the RTT and compute timeout based on RTT

\[
\text{SAMPLE\_RTT} = (\text{time segment acked}) - (\text{time segment sent})
\]

\[
\text{EST\_RTT} = (\alpha) \ast \text{EST\_RTT} + (1 - \alpha) \ast \text{SAMPLE\_RTT}
\]

(alpha is a smoothing value of old estimate and new estimate. Usually, alpha is between .8 and .9)

Set the timeout to **twice** the EST\_RTT:

\[
\text{TIMEOUT} = 2 \ast \text{EST\_RTT}
\]

**Question 8:** What flaw(s) do you see in setting the TIMEOUT in this way?

Think about the acknowledgments – don’t know if they are from the first transmission or second transmission (in calculation of SAMPLE\_RTT). Could cause long EST\_RTTs.

**Karn/Partridge Algorithm:**
Idea: Don’t take sample RTT for retransmitted segments and use exponential backoff for timeout when segment is not acked in time.
Same as original algorithm except:

SAMPLE_RTT only calculated for non-retransmitted segments.

If no ack for a segment, TIMEOUT = 2*TIMEOUT

**Jacobson/Karels:**

Idea: Take into account the **variance** of the RTT. Why? If variance is small, put more trust into the SAMPLE_RTT.

\[ \text{DIFF} = \text{SAMPLE}_\text{RTT} - \text{EST}_\text{RTT} \]

\[ \text{EST}_\text{RTT} = \text{EST}_\text{RTT} + (\text{delta} \times \text{DIFF}) \quad \text{// update ESTRTT} \]

\[ \text{DEV} = \text{DEV} + \text{delta} \times (|\text{DIFF}| - \text{DEV}) \quad \text{// calculate deviation} \]

\[ 0 < \text{delta} < 1 \]

\[ \text{TIMEOUT} = u \times \text{EST}_\text{RTT} + p \times \text{DEV} \quad \text{// } u = 1 \text{ (normally), } p = 4 \text{ (normally)} \]

**Question 9:** If the variance is small, does this calculation for TIMEOUT come close to the EST_RTT?

**Question 10:** Assume EST_RTT = 90 and SAMPLE_RTTs are 200. What are the next 3 values of TIMEOUT? Assume DEV = 25 and delta = 1/8 and use Jacobson/Karels:

\[ \text{TIMEOUT} = u \times \text{EST}_\text{RTT} + p \times \text{DEV} \quad \text{// } u = 1 \text{ (normally), } p = 4 \text{ (normally)} \]

**Question 9:** If the variance is small, does this calculation for TIMEOUT come close to the EST_RTT?

**Question 10:** Assume EST_RTT = 90 and SAMPLE_RTTs are 200. What are the next 3 values of TIMEOUT? Assume DEV = 25 and delta = 1/8 and use Jacobson/Karels:

--------

**Measuring Time (how do we get sample_rtt?)**

The sender can put a timestamp in the header of the TCP segment and the receiver can use the sender’s timestamp in the ack (so the sender can just use its own clock to determine the RTT).

--------
Those other flags in TCP

The PUSH flag – application can push data (can be used for record boundary indicator if want data to be interpreted in special chunks rather than a byte stream)
The URG flag – application can send urgent or out of band data (can be used to send special signal to receiver)
CS 445: Congestion and Queuing

Congestion Control: keep senders from sending more data than network can handle (note: this is separate from flow control where sender does not over-run the receiver’s capacity)

Question 1: What happens at routers when the network is very busy? What does a sender using TCP do?

Flow: sequence of packets between source and destination pair (usually follows same path through network)

Routers can watch what is happening locally (view packets as flows just using IP addresses of sender and receiver)

Let’s look at different approaches:

1. First, who should manage the resources?
   
   HOSTS (end)  

   ROUTERS (inside)

2. How does the host determine capacity of the network?

   RESERVATIONS  

   FEEDBACK

3. How is network capacity measured?

   WINDOW-BASED  

   RATE-BASED
**Question 2:** What design choices do we make for computer networks using IP?

**Question 3:** Suppose you want to maintain a network that provides a minimum quality of service. What design choices would you make?

Ideally, we want high “power” in a computer network, where power = throughput / delay. Get lots of traffic through the network without waiting a long time.

Now, let’s turn our attention to fairness.

**Question 4:** How would you define fairness for flows?

We have a mathematical way to calculate “fairness”. If each flow gets throughput $X_i$, then we can use the following fairness function:

$$F(x_1, x_2, x_3, \ldots x_n) = \frac{\text{Sum}(x_i)^2}{N \times \text{Sum}(x_i^2)} \quad // \text{square of sums divided by (N times the sum of the squares)}$$

**Question 5:** Assume all flows get the same throughout T and there are N flows. What does the fairness function equal?

**Question 6:** Assume now that there are K flows that get throughput T and (N – K) flows get throughput 0. What does the fairness function equal?
Now, given that routers have finite resources, they can drop packets. Also, routers forward packets, so routers need to make a few decisions. The first decision is which packet to forward next. The second decision is which packet to discard should the buffer get full. These are known as the scheduling policy and the drop policy.

**Question 7:** Determine at least one scheduling policy for the router. What are the pros/cons of this policy? Is it “fair”? Is it simple?

**Question 8:** For your scheduling policy in question 7, what would you decide for the drop policy? Which packets get dropped from the queue if it becomes full?

**Pros/Cons of the Scheduling and Drop Policies**

**FIFO**

<table>
<thead>
<tr>
<th>Pros:</th>
<th>Cons:</th>
</tr>
</thead>
</table>

**Priority Queuing**

<table>
<thead>
<tr>
<th>Pros:</th>
<th>Cons:</th>
</tr>
</thead>
</table>
Fair Queuing

Pros: 

Cons: 

Fair queuing (one queue per flow):

Notation:

$P_i$ = length of packet $i$ in bits 
$S_i$ = start time for transmitting packet $i$ 
$F_i = S_i + P_i$ = finish time for transmitting 

If a packet is waiting in a queue, then its time value is:

$F_i = F_{i-1} + P_i$  
//time of previous packet finish plus size of packet 

If a packet has not arrived at the queue, then its time value is based on its arrival time:

$F_i = A_i + P_i$ 

So, $F_i = \max(F_{i-1}, A_i) + P_i$ 

Router calculates the $F$ values for each packet and chooses to transmit the lowest $F$ value from any of its queues.

Example: 2 flows

F1 queue: P3(size 6) | P2(size 5) | P1(size 3) | FRONT OF QUEUE
F2 queue: P5(size 3) | P4(size 10) | FRONT OF QUEUE

Calculate the finish times for each packet:

P1
P2
P3
P4
P5
What is the order of transmission with fair queueing?

Suppose the following packets arrive at the router and it uses weighted fair queuing.

<table>
<thead>
<tr>
<th>Packet</th>
<th>Size</th>
<th>Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>160</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>120</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>160</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>210</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>150</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>90</td>
<td>3</td>
</tr>
</tbody>
</table>

What are the finish times of each packet?

What is the order of packet forwarding?

Now, we could assign weights to the flows, so each flow has a potentially different level of priority. Note that this is no longer fair, but it is called weighted fair queuing. Assume flow 2 has twice the weight as flow 1 and flow 3 has weight 1.5 times flow 1. Thus, $W(1) = 2$, $W(2) = 4$, $W(3) = 3$. The same packets above would now have weights of:
<table>
<thead>
<tr>
<th>Packet</th>
<th>F_i</th>
<th>W_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>400</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>160</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>280</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>440</td>
<td>110</td>
</tr>
<tr>
<td>6</td>
<td>210</td>
<td>70</td>
</tr>
<tr>
<td>7</td>
<td>360</td>
<td>120</td>
</tr>
<tr>
<td>8</td>
<td>450</td>
<td>150</td>
</tr>
</tbody>
</table>

(divide the finish time by the weight to get W_i)

What is the order of packet forwarding?
CS 445: Congestion Control (TCP)

Issue: Network can get congested with traffic
What can we do about it? (react to network conditions with sending rate)

Idea: Send packets into network, react to observable events
Use acks to pace transmission (self-clocking, acks determine capacity of network)

Issue: Network conditions change over time, cannot poll routers directly, a bit more difficult to get feedback from network than simply from the receiver (like in flow control). So, we need to get feedback via the acks and timeouts.
Overview of four TCP congestion control mechanisms:
1. Slow Start – used to get to an equilibrium sending rate
2. Additive increase / multiplicative decrease – used to react to network by slowly increasing congestion window and dramatically decreasing congestion window when network is busy
3. Fast retransmit – start retransmissions before timer fires
4. Fast recovery – less abrupt reduction of congestion window size

TCP Tahoe – uses mechanisms 1/2/3
TCP Reno – uses mechanisms 1/2/3/4

**Slow Start**
Goal: quickly determine appropriate send window size for network

Idea: Have a congestion window (CW)
Double CW for each ack received (grow window slowly, even though this is exponential growth)
Send \( \text{min} (AW, CW) \) – whichever is smaller (receiver or network is bottleneck)

CW doubles every round trip:

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
</table>

**Question 1:** Why should we use slow start?
Additive Increase / Multiplicative Decrease
Goal: Alter CW size after slow start phase to react to changes in network conditions during connection

Idea: Ack means network not congested, Lost segment means network is congested

Strategy:
On each RTT with acks, \( CW = CW + 1 \)
On each timeout (lost segment), \( CW = CW / 2 \)

Note: we still use \( \text{MaxWin} = \min(AW, CW) \), so we guarantee not overrunning the network and not overrunning the receiver.

EffectiveWindow = \( \text{MaxWin} - (\text{LastByteSent} - \text{LastByteAcked}) \)

Also note: CW here is in # segments, but in reality the CW is measured in bytes. So, instead of increasing by 1, we would increase by MSS.

Combination of SS and AIMD:
Assume connection already established.
If connection is open and no packets in flight, SS is used (ramp up faster)

If a packet ack times out.
\( \text{SSThreshold} = CW / 2 \) where CW is the size prior to the loss.
\( CW = 1; \)
SS is used when \( CW \leq \text{SSThreshold} \)
AI is used when \( CW > \text{SSThreshold} \)

Question 2: See TCP trace on page 129. Explain what is happening from 0 to 5.6 seconds.
Fast Retransmit

**Question 3:** Look at the second TCP trace below. What is different?

Goal in fast retransmit: learn of lost segments before timeout triggers

Ah ha, need help from receiver now.

Situation: most packets are getting through, but a few are lost. It would be nice to know about the lost packets, so sender can retransmit them.

Original TCP: receiver only acks for next in-order segment, never acks for out-of-order segment

Now: ask receiver to send the same ack sent last time if it receives out of order packet (duplicate acks)

Sender can now use duplicate acks as more information about the network. A dack means that the receiver got something, but it still waiting for the sequence # in the ack. If we get another dack, the receiver got something, but not the segment with the sequence # in the ack.... So, the sender can assume that the segment is lost before a timer fires for retransmission.

Use 3 dacks to indicate a lost segment.

Let’s look at the traces again .... The long lines of no data being sent are eliminated, thus improving throughput.
**Fast Recovery**

Idea: Well, if the sender is getting dacks, something is getting through, so instead of dropping CW to 1 when packet is lost, drop CW by half instead (eliminate slow start phase).

1. When receive 3 dacks, retransmit lost segment.
2. Set \( CW = \frac{CW}{2} + 3 \) \[\text{[where 3 accounts for 3 segments of data being acked]}\]
3. Send data if EffectiveWindow > 0
4. When ack arrives for lost segment, use AIMD

Result: only have SS phases on startup and on real timeouts

**Question 4:** Look at the second graph. What would be different if TCP used fast recovery? Would there be slow start at 3.8 seconds or 5.5 seconds?

**Summary**

4 mechanisms: SS, AI/MD, FastRetransmit, FastRecovery (TCP Tahoe – first three, TCP Reno – all)
Dashed lines at top of figure indicate when packets are in transit.
Dots indicate timeouts.
Full vertical lines indicate packets that are lost.

Timeline of TCP with Fast Retransmit
What would the TCP trace with fast recovery look like?
Congestion Avoidance

TCP Tahoe and TCP Reno use congestion detection to alter sending rate. But what if try to avoid congestion by detecting when segments might get lost?

How could we do this? _______________________________

What other info have we not used yet? _____________________

Idea: Measure change in RTTs and use this info to avoid congestion
Goal: Match sending rate and available bandwidth

BaseRTT = RTT of packet when network not congested (usually running min or RTT of first packet sent)

ExpectedRate = CW (in bytes) / BaseRTT
(Throughput when not congested)

ActualRate = # bytes in transit / SampleRTT

Send a distinguished packet P at clock time T, count # bytes sent in all packets from time T on until ack for packet P arrives at time T’. Then, SampleRTT = T’ – T.

Diff = ExpectedRate – ActualRate
   Alpha is lower bound on Diff
   Beta is upper bound on Diff

If Diff < alpha, CW = CW + 1    // increase linearly
If Diff > beta, CW = CW – 1    // decrease linearly
Else, CW = CW                   // do nothing

What does this do? Keeps sending rate between alpha and beta
But how do we determine alpha and beta?

Experimentally:
Alpha = segment size / BaseRTT
Beta = 3*segment size / BaseRTT    // 3*alpha

Can think of alpha as the minimum buffers in network and beta as the maximum number of buffers in network at current time.
If segment is lost, use multiplicative decrease.

(See handout below for congestion figure)

This version with *congestion avoidance* is called TCP Vegas.

**Random Early Detection**
What if routers help senders?

Idea: Routers drop packets prematurely to alert sender know sooner that router is busy.

At router, calculate:
AvgLen = (1-w)*AvgLen + w*(SampleLen) where 0 < w < 1
(weighted average of queue length)

Traffic is bursty, could be full and the empty then full then empty ... the running average gives sense of longer lasting congestion.

Now, routers will drop packets before the queue(s) get(s) full.

| MaxThreshold | MinThreshold |

If AvgLen <= MinThreshold, queue packet
Else if MinThreshold < AvgLen < MaxThreshold, calculate probability p for packet (based on fullness of queue) and drop with probability p
Else drop packet

Probability of packet getting queued:

```
1
/
/
/
1 0
```

Calculation for probability p a little more complicated (includes time since last packet was dropped). See textbook for more details.
**Quality of Service**

Some apps require guarantees about the network

- Example: videoconferencing, voice over IP
- But, the Internet is a best effort model

Contrast to phone network: allocate resources to individual flows, guarantee on latency and guaranteed delivery

Voice over IP – need guarantee on latency (too hard to have conversation when receiver gets data too late)

- One solution: remove pauses at playback point (distorting signal at playback)
- Delay-adaptive

Videoconferencing – can drop frames or reduce quality

- Rate-adaptive

**Approach 1: Differentiated Services** – allocate resources to certain classes of traffic

- Put info in header about priorities of data (some get expedited forwarding)
- Routers prioritize forwarding based on this info (priority queuing)

But, then who gets to assign this info in the header?

- Sys admins
- Payment by customer
- Routers at edges of network

...can lead to misuse (so, in general, there is no minimum quality of service)

**Approach 2: Reservations** – then routers need to maintain lots of state about individual flows

- Not widely deployed, conflicts with best-effort model
- Still under discussion/debate (for over 10 years)... could see a resurgence depending on new needs of applications

Return to the discussion of TCP...

- Streaming generally use UDP (no reliability and no congestion control)
- Means that TCP connections get smaller and smaller throughput in the network ... some research into protocols for doing fairness per flow
This leads to the development of TCP Vegas, which tries to control for congestion before a packet is lost.
The congestion window now does not fluctuate as much (top figure). The graph shows the additive increase and additive decrease used to avoid congestion.

The bottom figure shows the expected throughput (light line), the actual throughput (dark line), and the thick region indicates the rates bounded by alpha and beta. The difference between the actual rate and the expected rate is used to determine how to change the congestion window.

Notice the correspondence between the graphs. At 1.5 seconds, the actual rate is in the shaded region, so the CW does not change. When the actual rate drops below the shaded region at 2 seconds, the CW is decreased (as a drop in actual rate indicates network congestion).
CS 445: Compression Techniques

Today: Networks are limited resources. Perhaps the applications can try to send as little data as possible.

App -> Compressed Data -> Network -> Decompression -> App

Question 1: Why is compression useful in computer networks?

Question 2: Provide examples of data that can handle “lossy” compression:

Provide examples of data that can only handle “lossless” compression:

Lossless #1: Run-Length Encoding

Idea: Encode strings of same symbol with number of that symbol followed by the symbol

Example:

When does it work well?
Lossless #2: Differential Pulse Code Modulation (DPCM)

Idea: Define a reference symbol and encode the differences to the reference symbol

Example:

When does it work well?

Lossless #3: Delta Encoding

Idea: Encode each symbol based on difference from previous symbol

Example:

When does it work well?

Lossless #4: Dictionary

Idea: Create a dictionary of all symbols, mapping of index to symbol and use indices to encode

Example:

When does it work well?
Lossless #5: Huffman Coding

Idea: Create binary tree based on symbol frequencies and use tree paths to encode symbols. The higher the frequency, the shorter the encoding.

Example:
Symbols are A, B, C, D with frequencies of A = .25, B = .15, C = .20, D = .40
At each step, use two smallest value nodes to combine them via a parent node.

\[
\begin{align*}
A \ (0.25) & \quad B \ (0.15) & \quad C \ (0.20) & \quad D \ (0.40) \\
\end{align*}
\]

The two smallest are B and C, so we combine them

\[
\begin{align*}
B/C \ (0.35) & \\
B \ (0.15) & \quad C \ (0.20) \\
\end{align*}
\]

So, now we have A (0.25), B/C (0.35), and D (0.40). The smallest two are A and B/C, so they get combined.

\[
\begin{align*}
A/B/C \ (0.60) & \\
B/C \ (0.35) & \quad A \ (0.25) \\
B \ (0.15) & \quad C \ (0.20) \\
\end{align*}
\]

Now, we have A/B/C and D, so they get combined:

\[
\begin{align*}
A/B/C/D \ (1) & \\
B/C & \quad A/B/C \ (0.60) & \quad D \ (0.40) \\
B \ (0.15) & \quad C \ (0.20) & \quad A \ (0.25) \\
\end{align*}
\]

Now, we have a tree, so we label the left branches with 0 and the right branches with 1. The path indicates the code for each symbol:

A: 01
B: 000
C: 001
D: 1

Example data: AABCDAA
What bits are sent?
What bits are decoded?
When does it work well?
CS 445 Activity: Huffman Coding

Suppose a text file is compressed using dynamic Huffman coding (the tree is created based on the data). The contents of the file include the letters \{a, b, c, d, e, f\}. Suppose the frequencies are as follows:

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>.10</td>
</tr>
<tr>
<td>b</td>
<td>.20</td>
</tr>
<tr>
<td>c</td>
<td>.32</td>
</tr>
<tr>
<td>d</td>
<td>.25</td>
</tr>
<tr>
<td>e</td>
<td>.08</td>
</tr>
<tr>
<td>f</td>
<td>.05</td>
</tr>
</tbody>
</table>

1. Create the Huffman tree. Put labels of 0 or 1 along each edge.

2. What is the encoding of each letter to bit sequence?

<table>
<thead>
<tr>
<th>Letter</th>
<th>Bit sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td></td>
</tr>
</tbody>
</table>

3. Assume the first set of characters in the file is as follows. Produce the encoded bit string for this sequence.

abacfeddcb
**Lossy #1: JPEG (image compression)**

*Question:* When you take digital pictures, do you save the file in raw format or jpeg format? How much smaller are the jpeg images? Try this out sometime to see the differences in sizes of a raw photo and a photo that uses jpeg encoding/compression.

**OVERVIEW OF JPEG:**

1. DCT
2. Quantization
3. Encoding

Original Image -> JPEG formatted picture

For now, let’s focus on grayscale images. Each pixel in an image is a value between 0 and 255. (Color pictures are just 3 separate arrays, each passed through the JPEG compression algorithm)

The picture is broken into blocks of size 8 x 8 pixels (little squares).

**Step 1: DCT = Discrete Cosine Transformation**

Input: pixel values
Output: coefficients of spatial frequencies

Think of image as being a vertical and horizontal signal.
Imagine you are an ant traveling from left to right on the 8 x 8 block.
If the values do not change very much, there are no high frequency components to the signal.

Actual math (more info in book if you care):

\[
P(x, y) \text{ is the pixel in the } x\text{th row and } y\text{th column}
\]

\[
\text{DCT}(i, j) = \frac{1}{\sqrt{2N}} C(i) C(j) \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} P(x, y) \cos\left(\frac{(2x + 1)i\pi}{2N}\right) \cos\left(\frac{(2y + 1)j\pi}{2N}\right)
\]

\[
C(i) = \frac{1}{\sqrt{2}} \text{ if } i = 0 \text{ and } 1 \text{ otherwise}
\]
So DCT(0,0) is the average of all the 64 pixels in the block (called DC coefficient). The other DCT values (called AC coefficients) are the spatial frequencies.

Low freq components  High freq components

It turns out that the high frequency components are less important to the image. So, we can keep the lower coefficients more precise and drop the precision of the higher coefficients.

Step 2: Quantization

Quantum for 8 x 8 block:

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>17</td>
<td>19</td>
<td>21</td>
</tr>
</tbody>
</table>

…. (pattern continues)

... 17 19 21 23 25 27 29 31

Compression step (this is the lossy part):

QuantizedValue(i, j) = round(DCT(i, j) / Quantum(i, j))

Then, we store the quantized value. So, even the DC coefficient will be divided by 3. When decompressing, we simply multiply the QuantizedValue by the Quantum.

Step 3: Encoding

This last step encodes the 8 x 8 DCT quantized coefficients in a more compressed (but lossless) manner.
The idea is that similar quantized values will be close together in this pattern. Uses run length encoding on these, since many of the later coefficients in the zig zag pattern have the value 0.

Then, the actual coefficient values (#s) are encoded using Huffman coding (another lossless compression).

The DC values are encoded using delta encoding (another lossless compression).

Back to color images: a standard image has pixels with red, green, and blue values. Another 3-dimensional set is called Y (luminance), U (chrominance), and V (chrominance). It turns out that humans can perceive light/dark better than color, so one of the arrays just contains brightness info (the Y component). The U and V components create colors.

Each Y, U, and V image is compressed using JPEG independently.

In practice, JPEG can compress with a ratio of 30:1 (but it totally depends on the image).

**Question:** What images would achieve a higher compression ratio using JPEG?

**Question:** What images would not achieve such a high compression ratio using JPEG?
**Lossy #2: MPEG**

Now we focus on a series of images that comprise a video. Usually, 30 frames/second is visually smooth for humans.

**Question:** What data in a video can you take advantage of for compression?

Frame 1 – Frame 2 – Frame 3 .... -> MPEG -> I, B, P Frames

I frame = reference frame (just a JPEG compressed image)  
P frame = difference to I frame  
B frame = interpolation between previous I/P and next I/P frame

Compressed video might have this order:  
I  B  B  P  B  B  I

I is independent – decompressed is regular image  
The first B frame uses the first I frame and P frame as reference and encodes difference  
The second B frame uses the first I frame and P frame as reference and encodes difference  
The P frame uses the previous I frame

So, the actual order in which the frames are encoded is:  
I  P  B  B  I  B  B

So, the frames that you need for reconstruction come before the frame to be reconstructed.

Encoding of MPEG usually done ahead of time (since the frames need to be reordered). Note: videoconferencing will have very few B frames, since you need “after” data for this to work.

Macroblock in MPEG: 16 x 16 pixels (each encoded independently)

**Let’s look at decompression:**  
If I frame, decode like JPEG  
If P frame,  
   Let I be previous I frame  
   P frame contains differences to I frame, but these differences are to a reference location
(Might have object move between the P frame and I frame)

Each macroblock has a motion vector and encodes difference to motion vector
(Example: motion vector might be (1, 3) which means use as reference 1 macroblock to the left and 3 macroblocks up). Then the P frame’s macroblock encodes difference to I’s macroblock 1 left and 3 up.

If B frame,

- B frames have for each macroblock:
  - Motion vector relative to previous reference frame
  - Motion vector relative to next reference frame
  - Delta for interpolation

To decode: \( P(x,y) = P(x,y) \) in reference for previous + \( P(x,y) \) in next reference

\[
\frac{P(x,y) \text{ in previous reference} + P(x,y) \text{ in next reference}}{2} + \Delta(x,y)
\]

(average of reference frames plus difference)
(B frames can also just use intra-picture encoding without reference frames, like I frames)

Each frame is independently encoded in three spectra: \( Y \) (16 x 16), \( U \) (8 x 8), and \( V \) (8 x 8)

When sending video data over a network
Can change quantization matrix (for I frames) dynamically to send more or less info (rate adaptive)

Can split video into layers: layer 1, layer 2, ... with each layer having adding more detail

**Question:** What type of MPEG frames could you tolerate losing across a network?

**Question:** What type of frames is most critical for video reconstruction?
CS 445: DNS, HTTP, SMTP covered in labs

See lab handouts
Question 1: How would you define a peer-to-peer system?

Question 2: What are advantages of a P2P system versus a client-server model?

Question 3: What are the challenges of implementing a P2P system?

Example: Structured Overlays

Unstructured overlays require lots of flooded messages traversing the “overlay” topology and the real network topology.

Distributed hash tables provide a nice mechanism for distributing file content and for routing. Also, the hashing determines which hosts should store certain files.

\[ H(X) \rightarrow n \]

\( X \) is an object, and \( n \) is the node in which the object resides.

Idea: If there are 100 nodes in a P2P system, then hash the objects into 100 buckets. Each node will then be responsible for storing all objects that are hashed to its node’s value up to the objects for the next node’s value.

So, what about routing?
Let’s say the object hash values are written in hex.

You are node 891a0. You want to locate object d1259a.
Then, you look up in your own table for a node with value starting with d. You fire off a query to that node. Then, that node looks up d1 in its table and forwards the message. Then that node looks up d12, etc. Then, let’s say node d12510 gets the message and it retrieves the file with value d1259a.

Note – again, neighbors in this structured P2P system may be on different continents, so messages might take a while to get through the network. Could try to keep neighbors geographically close.
Suppose a structured overlay is created among 6 nodes and 10 files are stored among the nodes.

A hash function has mapped the node addresses and files to 16-bit values (normally, this would be a larger space but for the purpose of this activity, we'll use a smaller range), shown as hex values in the table below. Remember that in hex, each digit represents one of 16 values (0 through f). Put the node values with an 'X' along the circle in the correct place. Put the object values with an ‘O’ along the circle in the correct place.

<table>
<thead>
<tr>
<th>Node Values (hex, decimal):</th>
<th>Object Values (hex, decimal):</th>
</tr>
</thead>
<tbody>
<tr>
<td>DADA 56026</td>
<td>03E8 1000</td>
</tr>
<tr>
<td>10E0 4320</td>
<td>6223 25123</td>
</tr>
<tr>
<td>AAAA 43690</td>
<td>2AF8 11000</td>
</tr>
<tr>
<td>8111 33041</td>
<td>8D67 36199</td>
</tr>
<tr>
<td>1B58 7000</td>
<td>1595 5525</td>
</tr>
<tr>
<td>5283 21123</td>
<td>5307 21255</td>
</tr>
<tr>
<td></td>
<td>E2FF 58111</td>
</tr>
<tr>
<td></td>
<td>B037 45111</td>
</tr>
<tr>
<td></td>
<td>56CE 22222</td>
</tr>
<tr>
<td></td>
<td>0079 121</td>
</tr>
</tbody>
</table>
Answer these questions:

1. How many files/objects does host DADA store? ______________
2. How many files/objects does host 1B58 store? ______________
3. How many files/objects does host AAAA store? ______________
4. Suppose host 8111 leaves the structured overlay. Which host assumes which files?
   __________________________________________________________________
5. Suppose a new host ABE0 (hex), 44000 (dec) joins the structured overlay. Which files, if any, get
   moved to this new host? ____________________________________________
6. Suppose host DADA is looking for file/object 1B60 (hex), 7008 (dec). Host DADA only has entry
   10E0 in its routing table, so it sends the query to it. To whom does 10E0 query? _____________
   Show the arrows on the circle to represent these queries.
CS 445: BitTorrent

Have you used a BitTorrent app? What about this protocol makes file download fast?

How does it work with a tracker?
1. You want the file humangenome.fasta (containing the entire dna sequence of the human genome).
   This is a huge file.
2. You search for a “torrent” of this file.
3. You get the humangenome.torrent file, which has the URL of the tracker for this file, the number of pieces the file is split into, error detection codes for each piece, and the names of the pieces.
4. Your BitTorrent client then connects to the tracker for this file.
5. The tracker returns a list of peers who make up the swarm for this file.
6. Your client connects to some of those peers (via TCP).
7. Your client connects to a peer with a swarm ID (given by the tracker) to ensure both parties have/want the same file.
8. Your client receives bitmaps (showing which pieces each peer has) and your client sends a bitmap of 0000000...000 to the other peers.
9. Your client chooses random pieces from random peers to download.
10. As your client receives pieces, you send a new bitmap of your pieces and now you can be a supplier of pieces to other peers.
11. Eventually, you will get all the pieces of the file.

That’s how it works with a tracker. You can also get files without a central tracker. These use distributed hash tables.

1. Your client has a peer-finder process. Upon start-up, a few finder addresses are installed.
2. Once you have found peers, you can search for swarms (based on ID).
3. You can send a message to the peers asking if they know any peers for that swarm.
4. Peers respond with peer IDs in that swarm or peers who are closer to the swarm ID.
5. Then your client can contact those peers.
6. Getting the file happens in the same way (as above).

What are the issues/challenges with BitTorrent?

How does using BitTorrent impact the network differently than a traditional client/server model?
Note: Exam topics may differ as the course progresses. See Moodle for most up-to-date exam review sheets. See Moodle for sample exam questions.
CS 445 Exam 1 Study Guide

Content: Exam 1 will cover chapters 1.1 through 2.8 of Computer Networks: A Systems Approach by Peterson and Davie. Material will be drawn from readings, the textbook, lectures, and labs.

Procedure: The exam will be conducted in class, starting promptly at the beginning of the hour. You may use 1 sheet of 8.5” x 11” paper (both sides) of notes during the exam. The exam is closed-book, closed-notes (other than your 1 sheet), closed-computer, and closed-calculator. If you need to perform computations, you do not need to put the result in final form. For example, you need not multiply out $2^{20}$. You can leave it as $2^{20}$ in your solution. Please come to the exam on time.

Helpful Reminders:
1. Label all calculations with units (if appropriate).
2. A bit has the unit of b and a byte has the unit of B. Be careful to use the correct units in your computations.

Topics: This study guide is not a contract – in other words, the exam may include topics not listed below and some topics listed below may not be covered on the exam. The following list contains the material covered so far in the course.

- Network requirements
- Network architecture & layers
  - OSI (7 layers)
  - Internet (4 layers)
- Network Components
  - nodes, links, hosts, switches (bridges), routers (gateways)
- Network Data
  - frames, packets, messages
- FDM, STDM, statistical multiplexing (protocols for multiplexing flows)
- Network Performance
  - latency, bandwidth, throughput, RTT, delay x bandwidth
- Encoding Schemes
  - NRZ, NRZI, Manchester, differential Manchester, 4B/5B
- Framing Protocols
  - Byte-oriented, Bit-oriented, Clock-based,
    - Sentinels
- Error Detection
  - 2D parity, checksum, CRC
- Reliable Transmission: ARQ
  - Stop & Wait
  - Sliding Window
- LANs
  - MAC addresses (hardware addresses)
  - Aloha – early version of using a shared resource
- Ethernet (802.3)
  - CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
- Wireless (Wi-Fi 802.11)
  - CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance)
- Labs
CS 445 Exam 2 Study Guide

Content: Exam 2 will cover chapters/sections 3.1 through 4.3 of *Computer Networks: A Systems Approach* by Peterson and Davie. Material will be drawn from the readings, the textbook, lectures, and labs. Note that chapters 3 - 4 build on chapters 1 – 2, so earlier material that is necessary for the concepts in chapters 3 - 4 may be covered on the exam. However, the exam will focus on material since exam 1.

Procedure: The exam will be conducted in class, starting promptly at the beginning of the hour. You may use 1 sheet of 8.5” x 11” paper (both sides) of notes during the exam. The exam is closed-book, closed-notes (other than your 1 sheet), closed-computer, and closed-calculator. If you need to perform computations, you do not need to put the result in final form. For example, you need not multiply out $2^{20}$. You can leave it as $2^{20}$ in your solution. Please come to the exam on time.

Helpful Reminders:
1. Label all calculations with units (if appropriate).
2. A bit has the unit of b and a byte has the unit of B. Be careful to use the correct units in your computations. M and k represent different numbers in the context of bandwidth versus data size.

Topics: This study guide is not a contract – in other words, the exam may include topics not listed below and some topics listed below may not be covered on the exam. The following list contains the material covered since exam 1.

Network Layer
- Switches (Bridges) and Forwarding
  - Forwarding Packets
    - Datagram
    - Virtual Circuit Switching
    - Source Routing
  - Learning sources / ports
  - Spanning Tree Algorithm (Perlman paper) [if we get to this before exam]
- Internet Protocol (IPv4)
  - IP Addresses (Class A, B, C)
  - Header Format
  - Fragmentation/Reassembly
  - Subnets – sharing a class B address among several networks
  - CIDR – making use of prefixes as network addresses
  - Forwarding tables and forwarding of packets
- DHCP: assigning IP addresses to hosts
- ARP: determining IP address/ MAC address mappings
- ICMP: control messages
- Routing (creating the forwarding tables)
  - Intradomain routing
- Distance Vector
  - RIP
- Link State
  - OSPF
  - Dijkstra’s Algorithm
  - Interdomain routing
    - Border Gateway Protocol (BGP)
    - Autonomous Systems
    - Hierarchy, relationships/policies among ASes
- IPv6
  - Addresses (128 bits)
  - Packet header format
  - Features
- Multicast routing (if time)
- Labs
CS 445 Exam 3 Study Guide

Content: Exam 3 will cover chapters 5, 6, 7, and 9 of *Computer Networks: A Systems Approach* by Peterson and Davie. Material will be drawn from the readings, the textbook, lectures, and labs. Note that chapters 6 - 9 build on chapters 1 – 5, so earlier material that is necessary for the concepts in chapters 5 - 9 may be covered on the exam. However, the exam will focus on material since exam 2.

Procedure: The exam will be conducted in class, starting promptly at the beginning of the hour. You may use 1 sheet of 8.5” x 11” paper (both sides) of notes during the exam. The exam is closed-book, closed-notes (other than your 1 sheet), closed-computer, and closed-calculator. If you need to perform computations, you do not need to put the result in final form. For example, you need not multiply out $2^{20}$. You can leave it as $2^{20}$ in your solution. Please come to the exam on time.

Helpful Reminders:
1. Label all calculations with units (if appropriate).
2. A bit has the unit of b and a byte has the unit of B. Be careful to use the correct units in your computations. M and k represent different numbers in the context of bandwidth versus data size.

Topics: This study guide is not a contract – in other words, the exam may include topics not listed below and some topics listed below may not be covered on the exam. The following list contains the material covered since exam 2.

**Transport Layer**
- Ports
- UDP
- TCP
  - Reliability
  - Connection state diagram (will be provided with exam if needed)
  - Segment format
  - Sliding window
    - Flow control
  - Triggering transmissions
    - Nagle’s algorithm
    - Silly window syndrome
  - Setting timeouts
    - Original TCP
    - Karn/Partridge
    - Jacobseon/Karels
  - Timestamps
    - Wrap-around of segment numbers
  - Scaling advertise window size
- Congestion Control
  - Flows
- Responsibilities (router versus host, reservations versus feedback, window versus rate based)
- Fairness
- Queuing
  - FIFO
  - Round Robin
  - Fair Queuing
  - Priority Queuing
- Drop policies
- TCP
  - Slow start
  - Additive increase/multiplicative decrease
  - Fast retransmit (use backoffs)
  - Fast recovery (remove slow start after timeout, so CW goes to half)
  - TCP Vegas (use RTTs to adjust CW)
- Random early detection (if time)

Compression
- Lossless
  - Run Length Encoding
  - Differential Pulse Code Modulation
  - Delta Encoding
  - Dictionary Encoding
  - Huffman Encoding
- Lossy
  - JPEG
  - MPEG

Applications
- SMTP
- HTTP
  - Request/reply messages
  - Caching
- DNS hierarchy
  - DNS name servers (lab 3)
  - Resolution process
- P2P Systems
  - Structured Overlay
  - BitTorrent
- Labs