Digital Communications I

(Introduction to Digital Communications)

Lent Term — 2008

Andrew W Moore

andrew.moore@cl.cam.ac.uk

Topic 1: Introduction

Course aims:

The aims of this course are to develop an understanding of communications networks from a wide perspective, within a framework of principles rather than technologies or architectures.

Technologies and architectures will, however, be used as examples and motivation.

Feedback's useful at any point
e-mail *andrew.moore@cl.cam.ac.uk*

Mechanics — subject to revision

Subject Mechanics

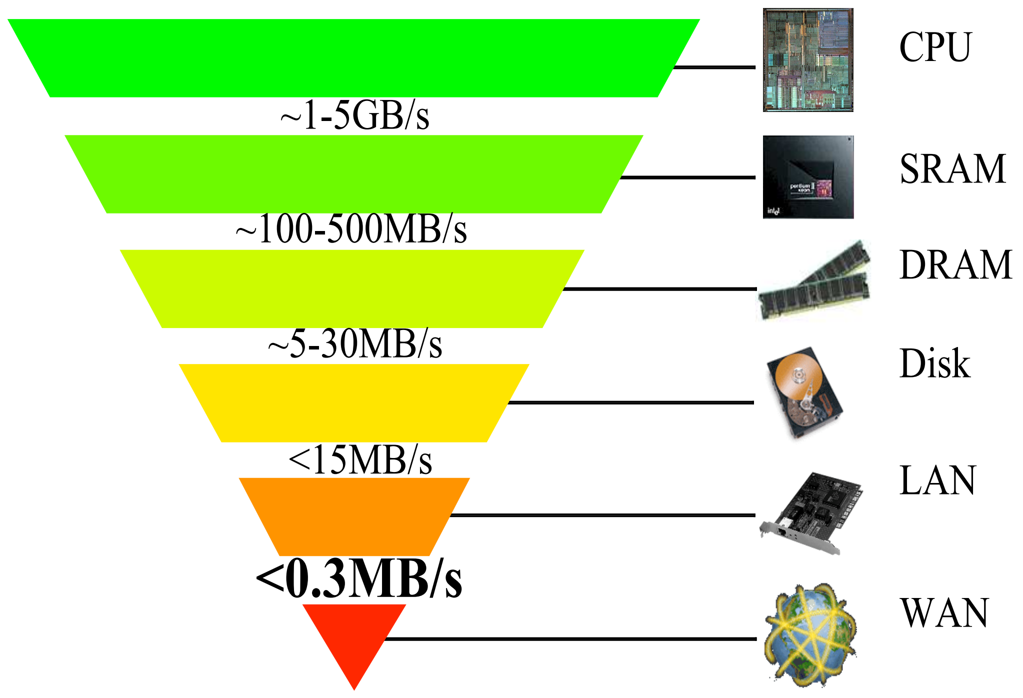
- All lectures are in William Gates Bulding, LT1
- 7 Weeks of Lectures (Tuesday and Thursday);
  12 $\times$ 50 minute lectures
- 2 lectures a week: Tuesday 1200-1300, Thursday 1200-1300

Lecturer Mechanics

- Australian (AQI)
- email me `andrew.moore@cl.cam.ac.uk`, BUT add the keyword **dc1** to the subject

Context 1 — Systems informs Networking

For a Systems-centric perspective...



- Why does this pyramid exist?
- What do we do to mitigate (reduce) its impact?
- Where is the important information?
- What is the *important* information?
- How can I *keep the beast fed*? (keep the CPU doing something useful?)
- How can I get the best efficiency from the WAN?

Context 2 – Multiplexing

Sharing makes things efficient.

- One airplane/train for 100 people
- One telephone for many calls
- One lecture theatre for many classes
- One computer for many tasks
- One network for many users

Disadvantages of Multiplexing?

- Might have to wait
- Might not know how long you have to wait
- Might **never** get served

Multiplexing is the action of sharing of common resources.

This links back to Concurrent Systems and Applications:
Concurrency is all about *how* common resources are shared.

Books

Most networking books actually tackle communication in a technology oriented way, so lectures will tend to *slice* across a text. There is no definitive book — sorry. However, the first on this list is a pretty good starting point.

- **Recommended** Peterson and Davie, *Computer Networks: A Systems Approach*, Morgan Kaufman
- Halsall, *Data Communications, Computer Networks, and OSI*, Addison-Wesley
- Tanenbaum, A.S. *Computer Networks*, Prentice-Hall International
- Stallings, *Data and Computer Communications*, Macmillan
- Douglas Comer, *Internetworking with TCP/IP* volume I, Prentice-Hall International. (definitive - high level)
- Douglas Comer, *Internetworking with TCP/IP* volume II, Prentice-Hall International. (definitive - the inner-workings)

Digital Communications I

- Layering
- Physical Transmission
- Modulation
- Coding
- Compression
- Multiplexing
- Protocol & State
- Names, Addresses, Routes
- The Internet
- Standards

Topic 2: Computer Networks

Overview of this section

- Digital Communications
- Digital Media
- Books
- Tale of Two Networks

What is Digital Communications?

Is it:

- how I get bits down a wire?
- how I join these “wires” together to build a network?
- what I can do given that I can get bits across a network?

Yes — a wide field, Digital Communications covers all of these fields.

Need to call on wide range of knowledge:

- mathematics: error detection, error correction, encryption
- physics: optics, propagation
- electrical engineering: modulation, noise
- computer science: algorithms, architecture, protocol specification...

An Aside: Why Digital for Video and Audio?

- maintaining fidelity using error correction (eg CDs)
- more flexible control (eg DVD vs. VHS cassette)
- large scale integration of switching equipment, ease of compacting in time and reusing components — eg the digital telephone network
- large scale integration of processing
  - digital signal processors
  - digital signal processing by general purpose processors
- the myth of “infinite precision” analog

Metacomments and Books

- principles not technologies (but technology as examples)
- communication changing rapidly; communications and computing becoming unified
- emphasis on "higher levels" not how to get bits down a wire

Books and References

Most books actually tackle communication in a technology oriented way, so lectures will tend to "cut across" a text.

A favourite is:

Computer Networks: A Systems Approach, Peterson and Davie, 3^{rd} Ed, Morgan Kaufman

Other good ones are:

Data Communications Computer Networks and OSI, Halsall, 2^{nd} Ed, Addison Wesley

Data and Computer Communications, Stallings, 2^{nd} Ed, MacMillan

Telecommunications: Protocols and Design, Spragins *et al* Addison Wesley

A very interesting paper is:

Ethernet: Distributed Packet Switching for Local Computer Networks, Metcalfe and Boggs, in *Communications of the ACM*, Vol 19 No 7.

Comparison of Two Networks

network	Shared Media (Old) Ethernet	Digital Telephone Network
total bandwidth	10 Mbps	vast (some trunks $>$ 10 Gbps)
max bandwidth user-user	10 Mbps	64 Kbps
interface cost	\$ 20	\$ 100
area	$<$ 3Km radius	global
traffic statistics	data packets bursty	circuit constant bandwidth
managed	locally	phone company
regulated	none	government
access control	distributed	network controls
connected to	computers	telephones
reliability	crashes	confinement of failures
traffic	data (voice)	voice (data)
availability	99.9%	99.9995%

What issues/concepts are common?

- transmission
- coding
- addressing
- multiplexing

Exercises

- [2-1] Consider new audio and video devices in your home and work-place;

  (i) Concentrating upon Digital devices (e.g. DVD, CD, MP3 player), what additional facilities do they have, in-contrast with their analog predecessor ?

  (ii) Are they networked? Could they be? What would you use such a network for?

- [2-2] Is your phone digital or analog or both? How can you tell?
- [2-3] Consider and compare the phone and postal system using the criteria of transmission, coding, addressing, and multiplexing.

Topic 3: Abstraction

Previous topic

- What is Digitial Communications?
- Why use Digital?
- Metacomments and Books

Overview of this topic

- Terminology
  - abstraction
  - layering
  - entities
  - peers
  - channels

Abstraction

Abstraction — a mechanism for breaking the problem down.

- *what* not *how*; don't need to know how a function is performed in order to understand how it fits into a larger system
- eg specification *versus* implementation
- eg Modules in programs
- tool for managing complexity
- allows replacement of implementation without adversely affecting system behaviour

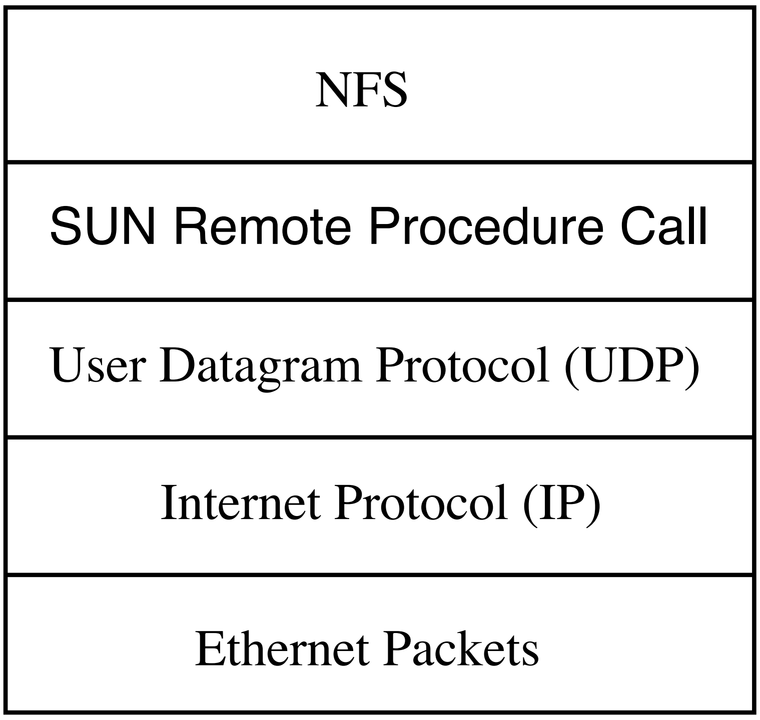
"Vertical" versus "Horizontal"

- "*Vertical*" One obvious way of splitting functionality up is to think about what happens in, say, physical box. "How does it attach to the next physical box" or "How do I connect my computer to the Internet?"
- "*Horizontal*" Perhaps less obvious at first, but think about communication paths running through the system rather than the boxes in the system. Key observation is that paths are built on top of ("layered over") other paths.

Layering

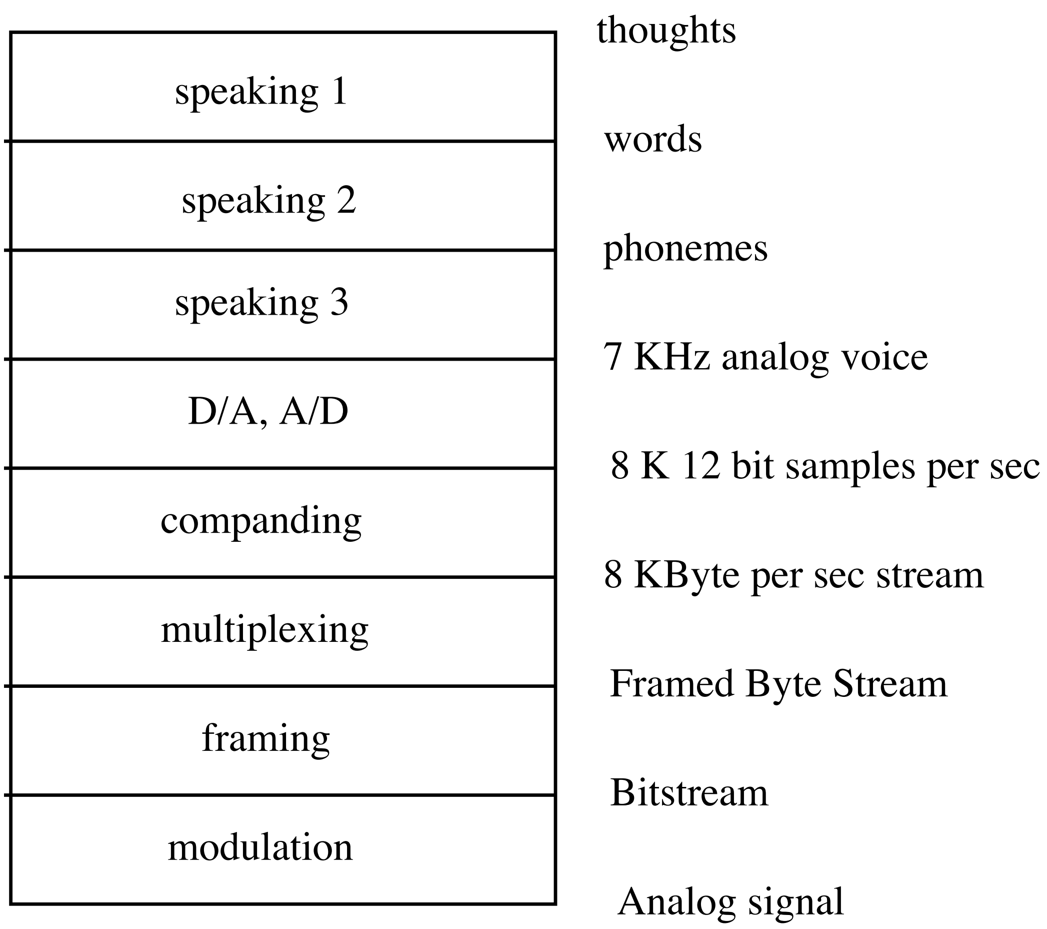
- restricted form of abstraction in which system function is divide into layers, one built on top of another
- sometimes layered structure referred to as a "stack", having nothing to do with the data structure.

Example: Networked File System

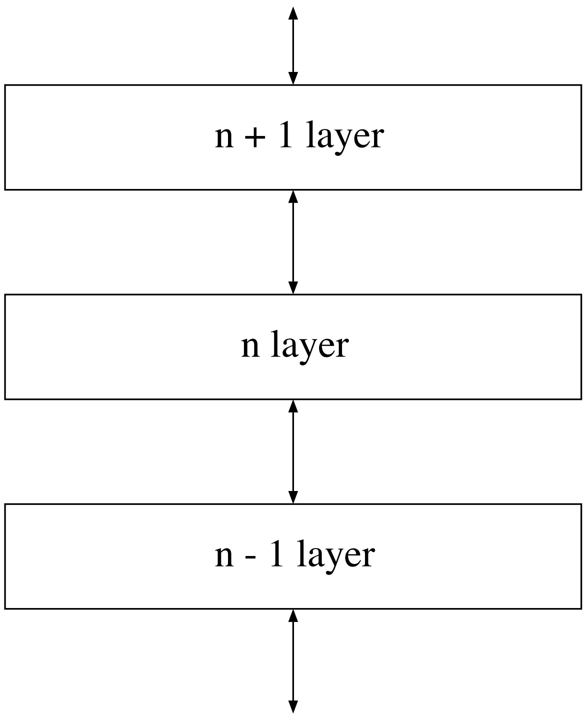


Layering examples - II

Example: Telephone Network

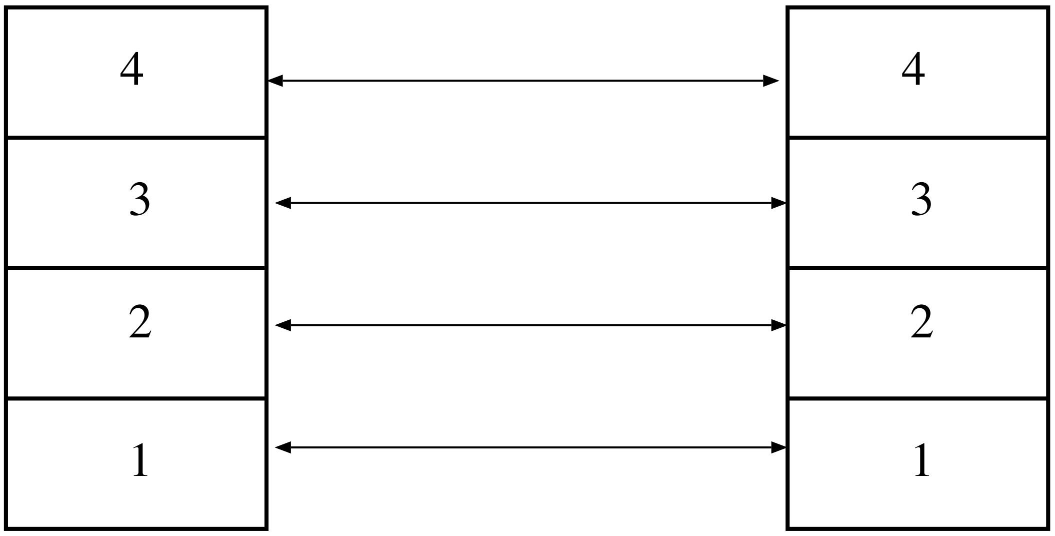


Layers and Communication

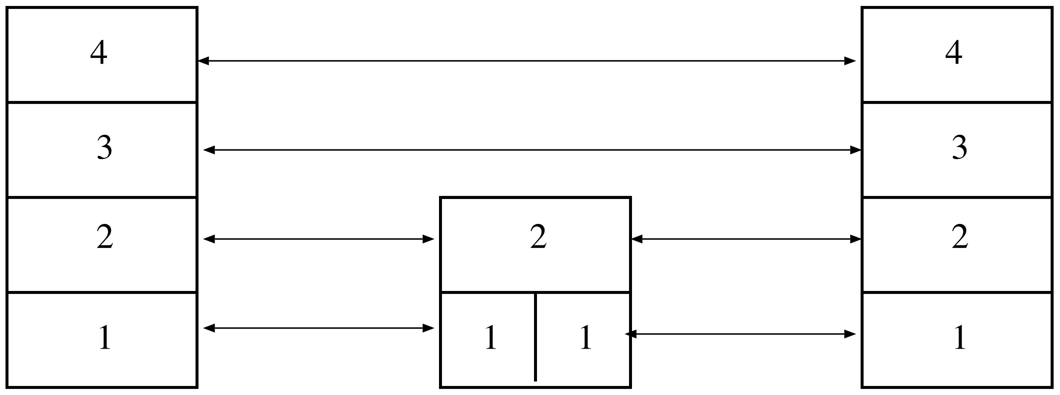


- interaction only between layers one level apart
- n layer makes use of services provided by $n - 1$ layer and provides service to $n + 1$ layer
- top layer is "application", something that does something useful (except some applications are built on top of other applications)
- bottom layer is physical media

Entities and Peers



- *entity* — that which has independent existence, in other words a thing
- entities are the things in layers which interact with the things in the layers above and below.
- entities communicate with *peer* entities which are at the same level in the layered stack but at a different place, eg a different machine, a different person.
- distinguish local interaction from peer to peer communication
- communication between peers supported by the entities at lower layers
- entities usually do something useful, like A/D D/A, encryption, error correction, but entities that do absolutely nothing are perfectly reasonable

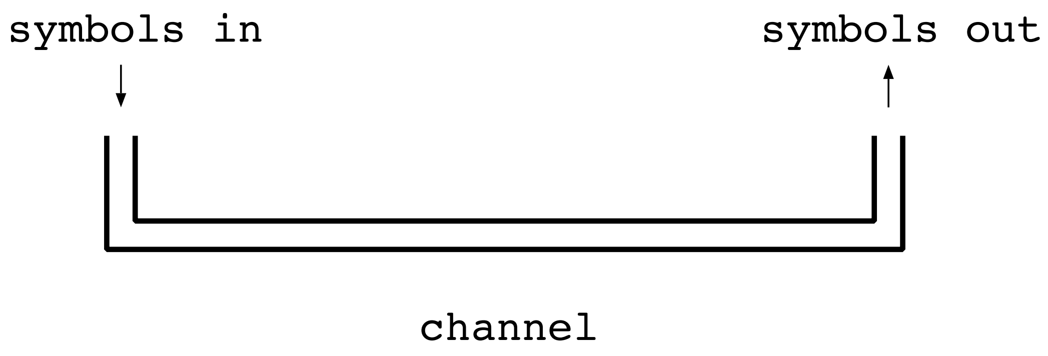


- not all communication is end to end!
- examples for thing in the middle:
  - IP router
  - microwave repeater
  - telephone switch
  - person in the middle relaying messages
  - person in the middle translating French to English

Channels

- peer entities communicate over *channels* and support communication for higher layer peers by providing them with higher layer channel
- channel in this sense is a very abstract term (refer to the voice on digital telephony stack).

A channel is that into which an entity puts symbols and which causes those symbols (or a reasonable approximation) to appear somewhere else at a later point in time.



Channel Characteristics

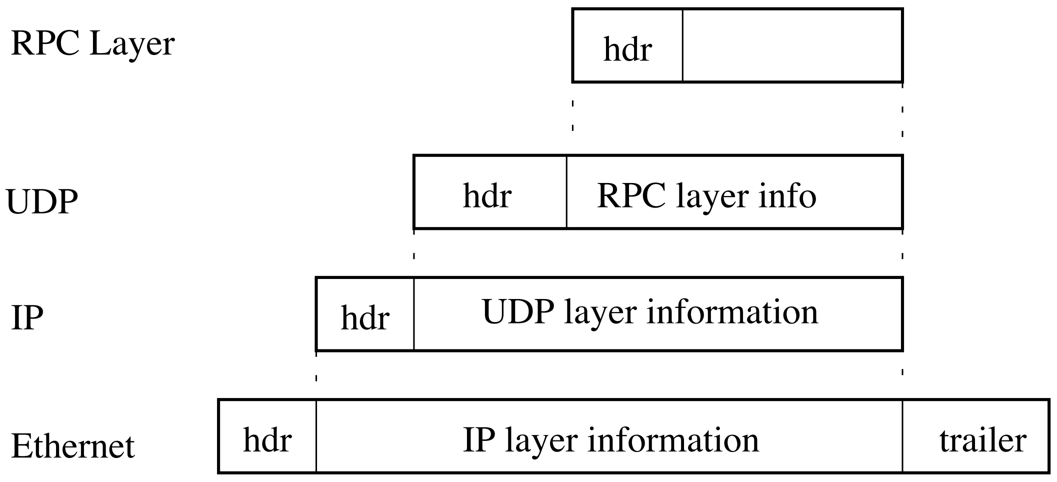
- symbol type: analog waveform, bits, packets
- capacity: bandwidth, data rate, packet rate
- delay: fixed or variable
- fidelity: signal to noise, bit error rate, packet error rate
- cost: per attachment, per call, for capacity consumed
- reliability
- security: privacy, unforgability
- order preserving: always, almost always, usually
- connectivity: point to point, 1 to many, many to many

Examples of channels

- a piece of coaxial cable
- a 64 Kbps channel in a telephone network
- the sequence of packets transmitted from one host to another on an Ethernet
- a telephone call (handset to handset)
- the audio channel in a room
- the sequence of words exchanged by two people in a room
- a diplomatic channel

Layering and Embedding

- often in data communications, we see higher layer information embedded in lower layer information.
- for example take our remote file server example

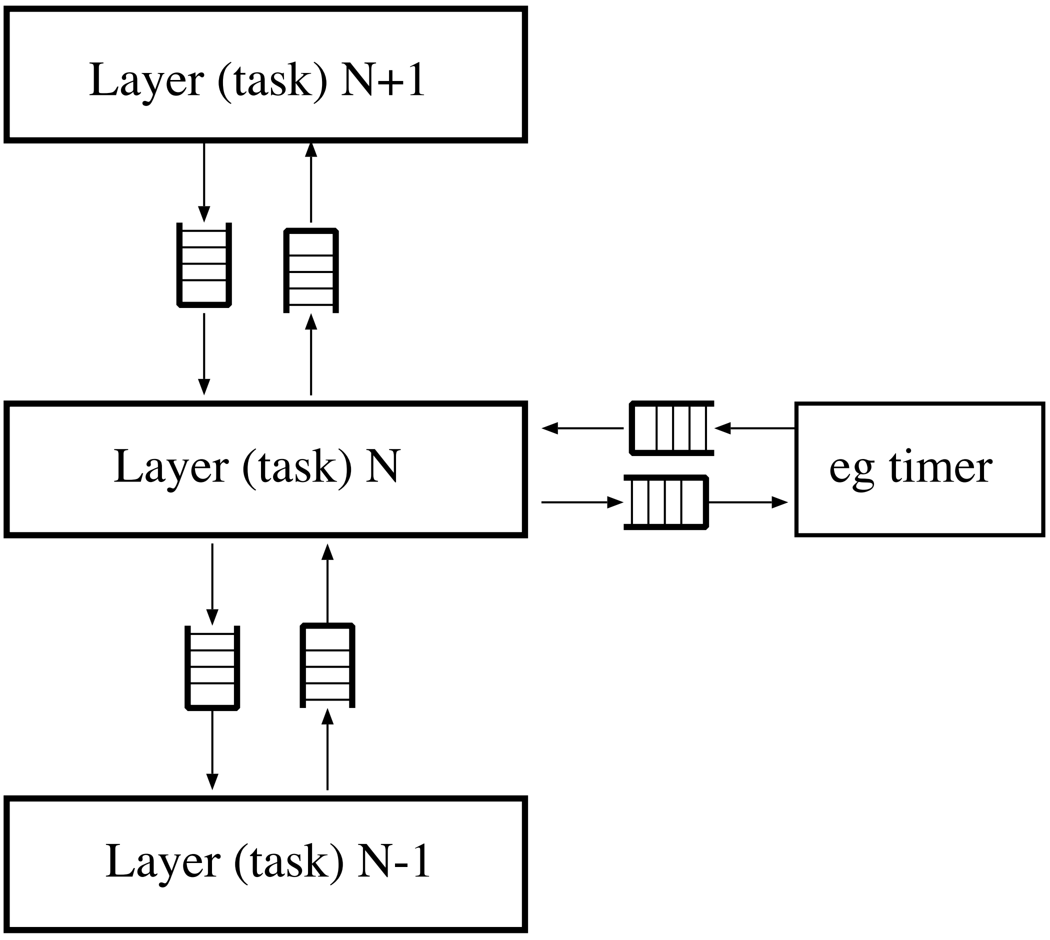


- this is embedding and can certainly be viewed as a form of layering
- information for higher layers is found by stripping off headers and trailers which are the only information processed at that layer.
- for example an IP entity only looks at IP headers in packets

BUT IT IS NOT THE ONLY FORM OF LAYERING

Layering and Implementation

Picture from Halsall's book:



This is one possible way of implementing a multilayer protocol stack in software. It is not the only or indeed in many circumstances the best. Layering is a tool to help understand how communication systems work, not to determine implementation strategy.

Peterson is slightly weird on this too...

Exercises

3-1 Abstraction is a form of breaking a problem down into constituent parts. Write down the modules involved when driving into a petrol station to fill the car, pay the cashier and depart. Now identify the changes if you were paying with a debit card or cash. Finally, identify the changes for you driving a car to the local supermarket and then buying an item from the local supermarket.

3-2 Abstraction is useful way to break-down a problem; layering is a simplified form of abstraction that is commonly used to describe modern networks;

 (i) In the style of the slides earlier in this topic, sketch the abstraction layers for a web-browser session.

 (ii) In slide 9, could the RPC of NFS be laid on top of TCP instead of UDP? If so, Why? If not, Why-not?

3-3 In slide 3-3, the Telephone Network as an example of layering, assuming a multiplex of 10 channels and no framing overhead, estimate what channel capacity is required at the lowest level?

Topic 4: Physical Transmission

Previous topic

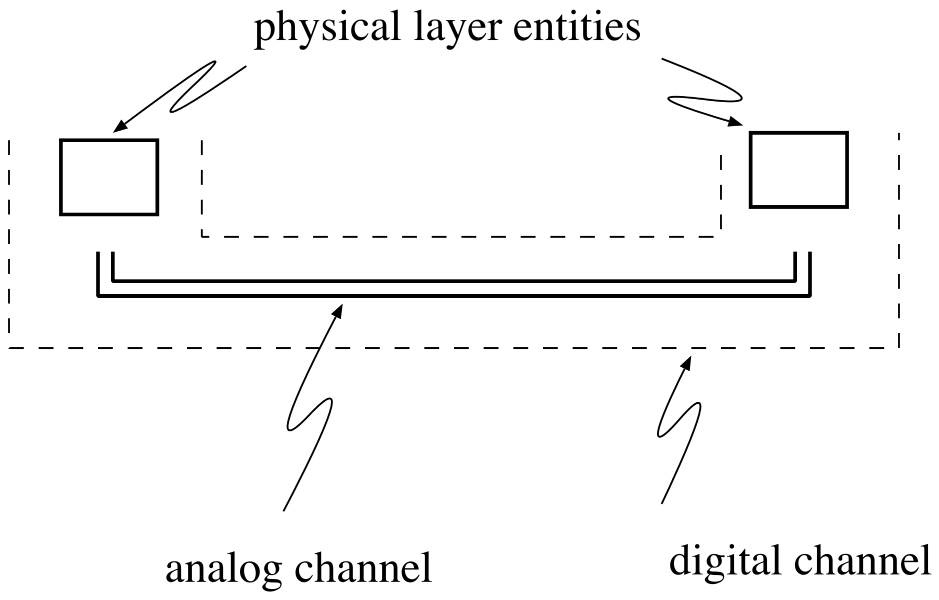
- Terminology
  - abstraction
  - layering
  - entities
  - peers
  - channels

Overview of this topic

- Transmission Media
- Characteristics (Fundamental Limits, Noise, Attenuation), Bandwidth vs Signal to Noise)
- Digital Channels
- Synchronization
- Modulation

Physical Transmission

- How do I get bits down a piece of wire?
- Physical transmission is the lowest layer function (except, perhaps, how I lay cable) we are usually concerned with
- We are concerned with turning an analog channel into a digital channel



Physical layer entities:

- make use of the "services" provided by the transmission medium
- usually model the transmission media as an analog channel
- translate bit stream into a representation suitable for transmission on the medium and retranslate back
- provide the service of a digital channel

Transmission Media

- acoustic
- electronic, guided and unguided
- optical, guided and unguided (eg smoke signals)

- what are the common considerations?
- what is this “service”
- not concerned with why

Characteristics: Fundamental Limits

- symbol type: generally an analog waveform — voltage, current, photo-intensity etc
- capacity: bandwidth
- delay: speed of light (or sound) in medium and distance
- fidelity: signal to noise ratio

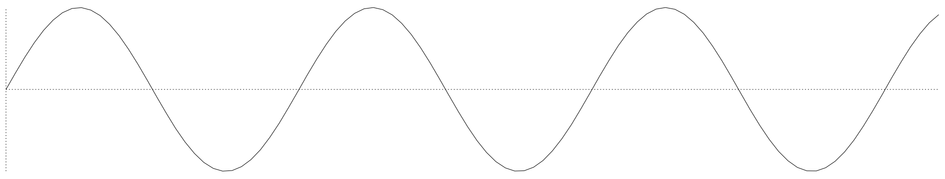
Bandwidth

- measure of the range of frequencies of sinusoidal signal that channel supports.
- eg a channel that supports sinusoids from 1 MHz to 1.1 MHz has a bandwidth of 100 KHz.
- "supports" in this context means "comes out the other end of the channel"
- some frequencies supported better than others
- analysing what happens to an arbitrary waveform is done by examining what happens to its component sinusoids $\longrightarrow$ Fourier analysis
- bandwidth is a resource

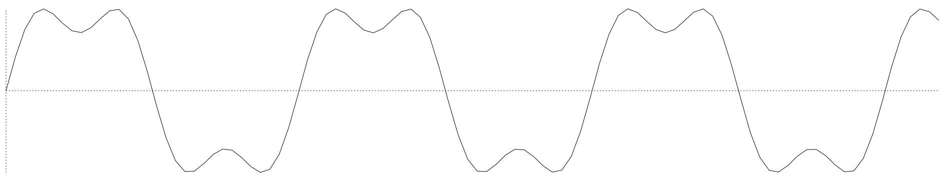
Brief Excursion into Fourier Analysis

square wave with period 2π “is”

$$sin(x) + \frac{1}{3}sin(3x) + \frac{1}{5}sin(5x) + ...$$

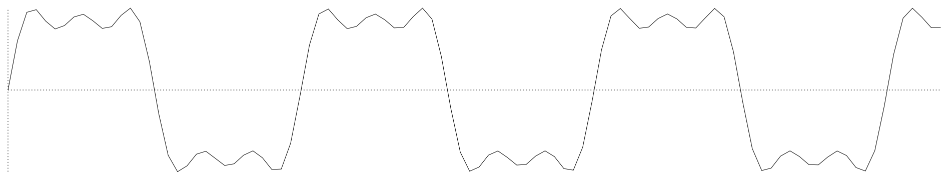


$$sin(x)$$

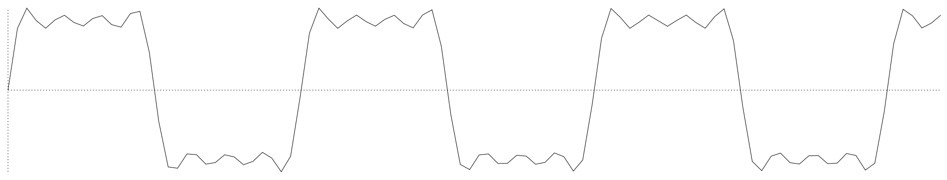


$$sin(x) + \frac{1}{3}sin(3x)$$

Excursion (cont'd)



$$sin(x) + \frac{1}{3}sin(3x) + \frac{1}{5}sin(5x)$$



$$sin(x) + \frac{1}{3}sin(3x) + \frac{1}{5}sin(5x) + \frac{1}{7}sin(7x)$$

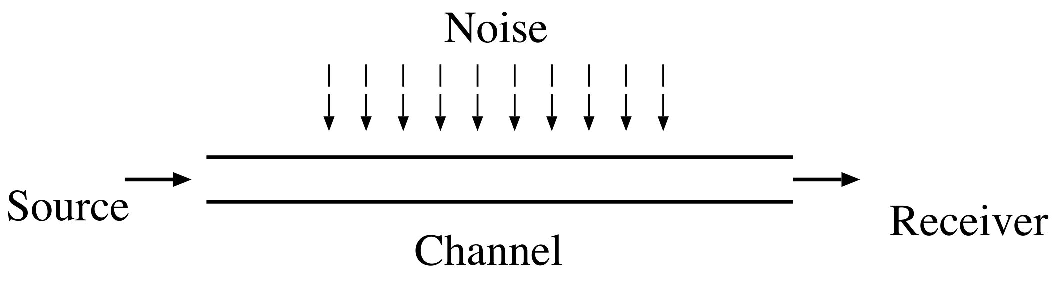
Excursion (conclusion)

- square waves have high frequency components in them

- channels which attenuate different frequencies differently will change the shape of the signal

- choose signals for transmission accordingly

Noise



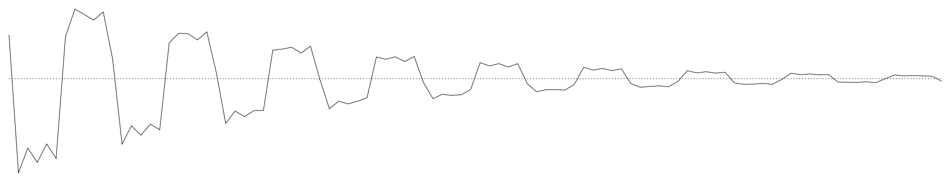
- receiver gets something related to the transmitted signal plus noise
- noise may be *systematic* or *random*

- systematic noise from interfering equipment
- can in principle be eliminated (but not always convenient)
- *Never buy a Monday car* joke

- random noise caused by thermal vibration (thermal noise)
- "white" power evenly distributed across frequencies

- signal to noise ratio $\frac{S}{N}$
- more distance more noise
- receiver (clearly) throws away noise not in the band of frequencies it is receiving

Attenuation



- signal looses energy as it travels along a channel
- radiation loss: in many media, higher the frequency, higher the radiation loss
- absorption by media — eg water molecules — then re-emitted as noise
- spatial dispersion

Attenuation is function of channel length, transmission medium, and signal frequencies

Bandwidth *versus* Signal to Noise

- what's better: high bandwidth or low signal to noise?

- for channels with white noise have *information capacity* C measured in bits per second, of a channel

$$C = Blog_2(1 + S/N)$$

  where B is the bandwidth of the channel and S/N is the ratio of received signal power to received noise power.

- (This is actually NOT the definition of information capacity; it is derived from the definition)
- channels with no noise have infinite information capacity
- channels with any signal have nonzero information capacity
- channels with signal to noise ratio of unity have an information capacity in bits per second equal to its bandwidth in hertz

Digital Channels

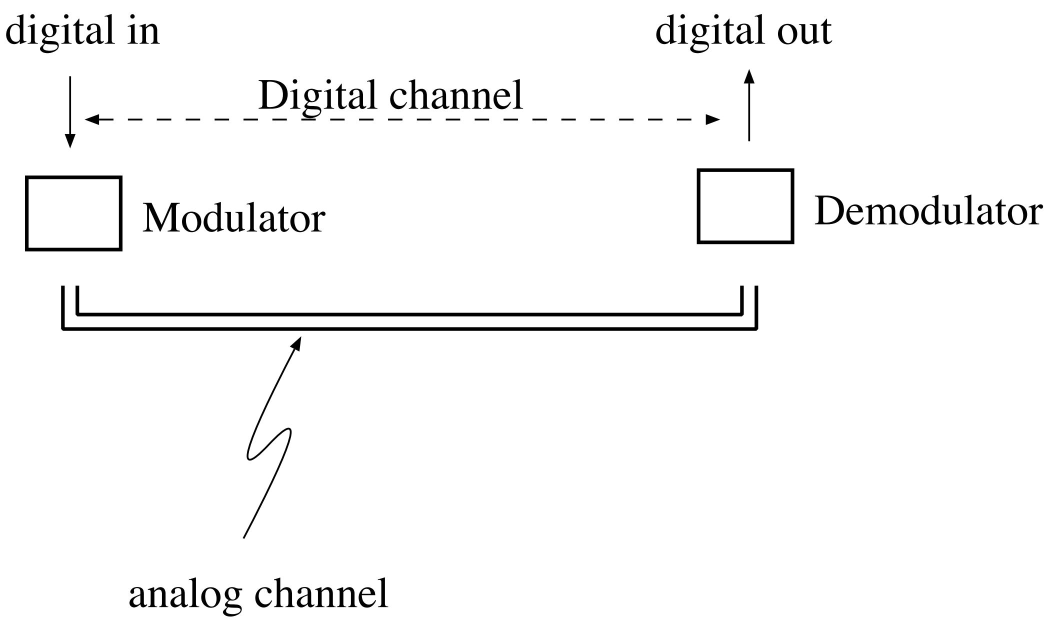
- service that physical layer provides
- consider them to be fixed rate for now
- symbols are discrete values which are sent on the channel at fixed rate
- symbols need not be binary
- *baud rate* is the rate at which symbols can be transmitted
- data rate (or bit rate) is the equivalent number of binary digits which can be sent
- for example if symbols are quaternary with rate R then the data rate is $2 \times R$.
- fidelity of the channel usually measured as a bit error rate — the probability that a bit sent as a 1 was interpreted as a 0 by the receiver or *vice versa*.

Modulation

Two definitions

- transform an information signal into a signal more appropriate for transmission on a physical medium
- the systematic alteration of a carrier waveform by an information signal

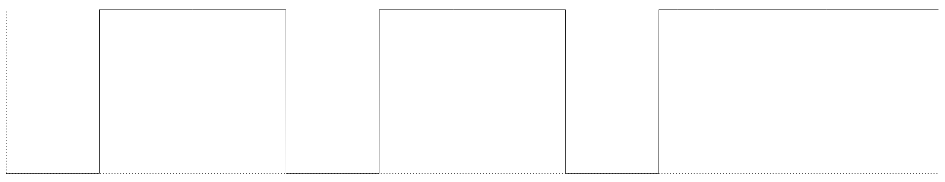
In general we mean the first here (which encompasses the second).



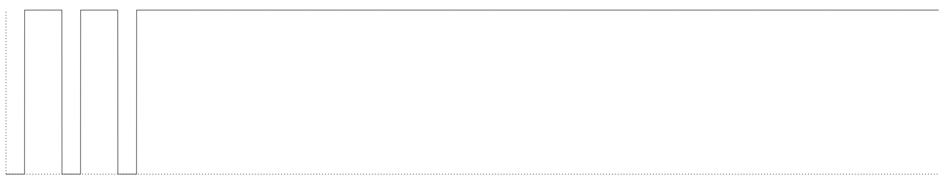
Baseband *versus* Broadband

- baseband is not modulated on a carrier
- broadband is, either for bandwidth sharing purposes or propagation purposes (radio, TV, satellite)

Baseband Modulation



A baseband signal

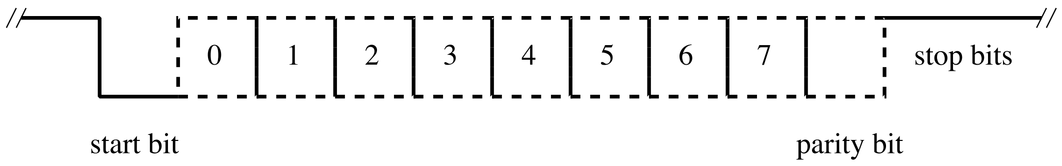


How many ones?

Synchronisation

- receiver needs to know where data bits boundaries are
- two ways
  - asynchronous
    - transmission is sporadic, divided into $frames$
    - receiver and transmitter have oscillators which are close in frequency producing tx clocks and rx clock.
    - receiver synchronises the $phase$ of the rx clock with the tx clock by looking at one or more bit transitions at the beginning of a frame
    - rx clock drifts with respect to the tx clock but stays within a fraction of a bit of tx clock throughout the duration of a frame
    - transmission time is limited by accuracy of oscillators
  - synchronous
    - transmission is continuous
    - receiver continually adjusts its frequency to track clock from incoming signal
    - requires bit transitions
    - phase locked loop: rx clock predicts when incoming clock will change and corrects slightly when wrong.

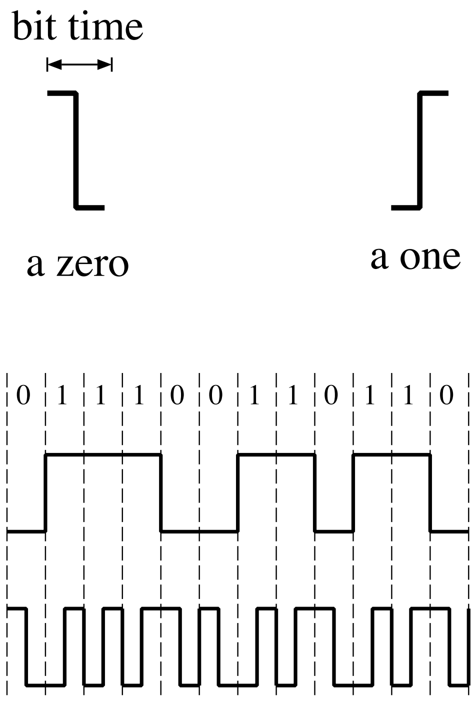
Asynchronous Transmission



- start bit tells receiver where first bit edge is
- receiver must stay synchronised for 9 bits
- stop bits allow receiver time to recover (ie get ready for next start bit)
- next start bit can be anytime later

Synchronous Transmission

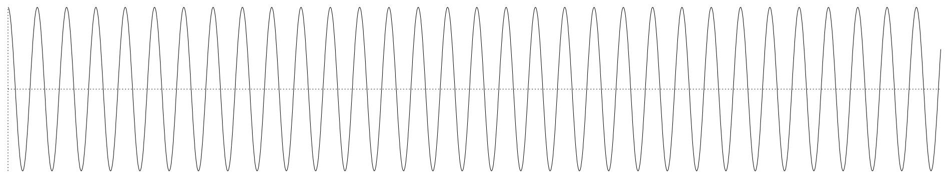
- want bit transitions
- Manchester coding, for example



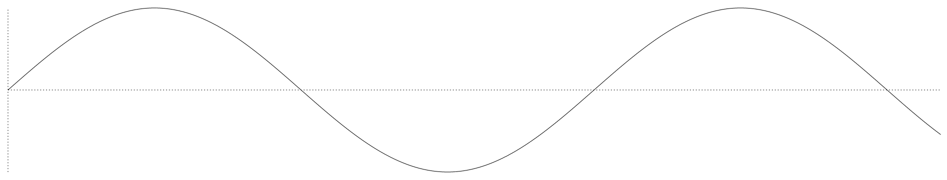
- how do you distinguish a series of ones from a series of zeroes?
- note that this signal has DC balance ...

Broadband Modulation

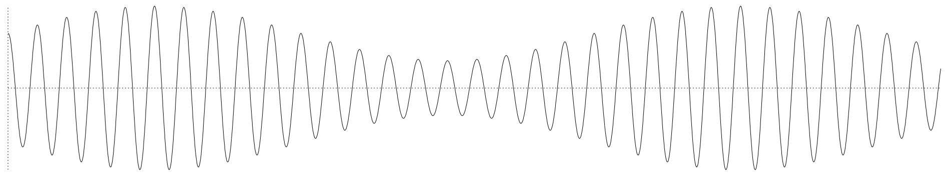
- carrier frequency which is modulated



$$\text{Carrier}: \quad Acos(2\pi f_c t)$$

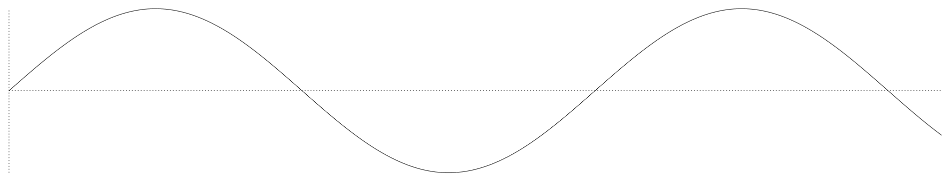


$$\text{Information}: \quad x(t)$$

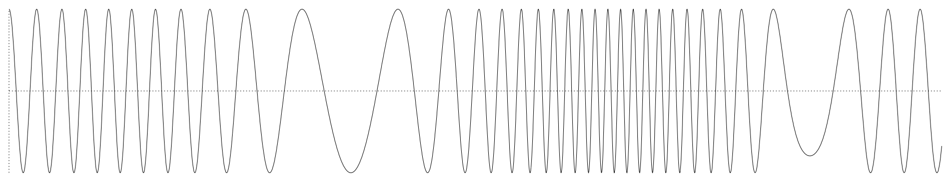


$$\text{Amplitude Modulation}: \quad A(mx(t)+1)cos(2\pi f_c t)$$

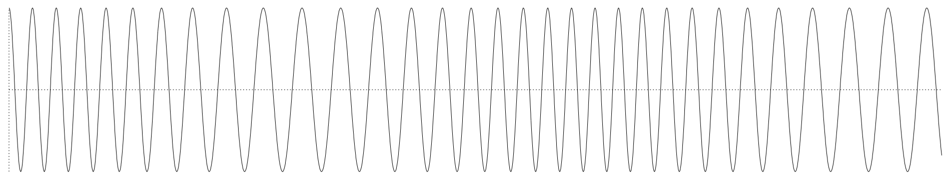
Frequency and Phase Modulation



$$\text{Information : } \quad x(t)$$

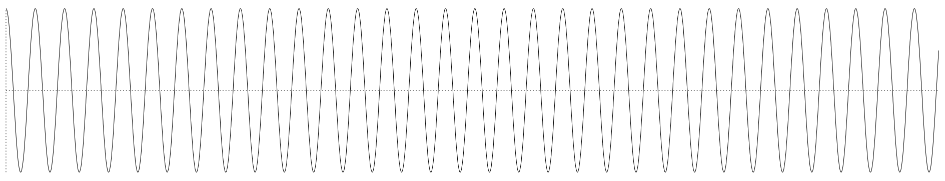


$$\text{Frequency Modulation : } \quad Acos(2\pi(f_c t + f_\Delta x(t)t)$$

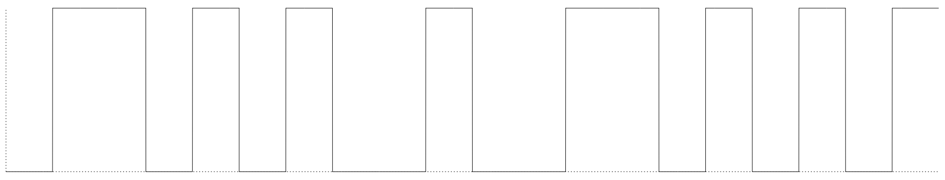


$$\text{Phase Modulation : } \quad Acos(2\pi(f_c t + \phi_\Delta x(t))$$

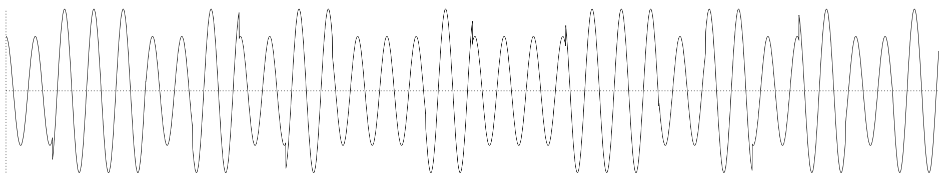
Broadband Modulation With Digital Informatio



Carrier : $Acos(2\pi f_c t)$

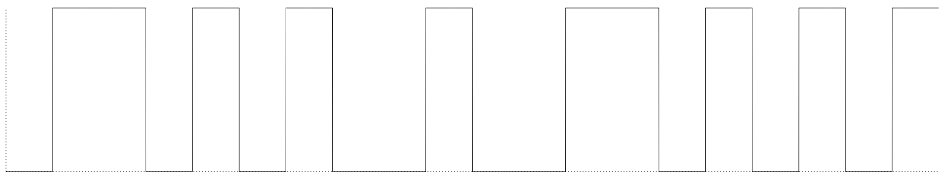


Information : $x(t)$

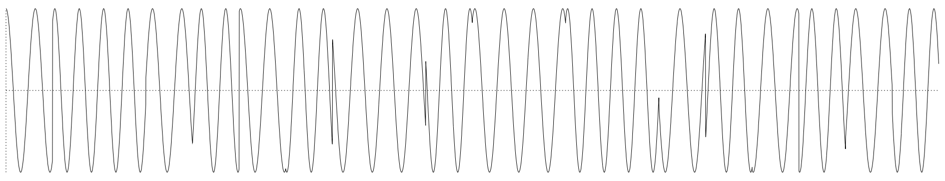


Amplitude Shift Keying : $A(mx(t)+1)cos(2\pi f_c t)$

Digital Frequency and Phase Modulation

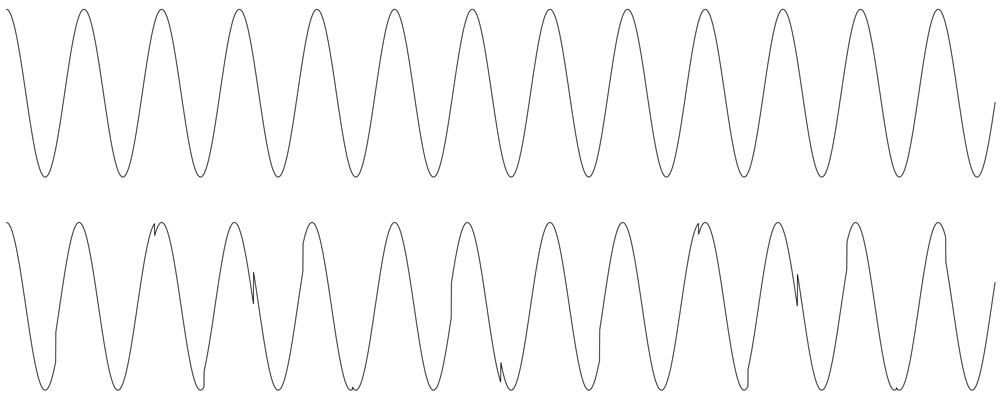


Information : $x(t)$



Frequency Shift Keying : $Acos(2\pi(f_c t + f_\Delta x(t)t)$

Carrier :



Phase Shift Keying : $Acos(2\pi(f_c t + \phi_\Delta x(t))$

Phase Shift Keying Finalé

- in practice one would filter to remove instantaneous changes in phase
- binary phase shift keying (BPSK) is shown but can have more than two levels, eg quaternary phase shift keying (QPSK)
- phase changes can be synchronised to the carrier frequency (coherence)
- possible to have more than one bit per hertz

Exercises

4-1 Compare the forms of modulation described in this Topic

4-2 Show a single modulation technique can achieve 2 bits/second for each change in the line

4-3 Read up on the definition of $Baud$, compare this with the definition of bits-per-second.
Can Baud ever be less than bits-per-second for the same communications channel? Why? Why-not?
Can Baud ever be greater than bits-per-second for the same communications channel? Why? Why-not?

4-4 Describe using a single physical transmission channel, examples of the following:
- (i) Attenuation
- (ii) Systematic Noise
- (iii) Random Noise
- (iv) Bandwidth
- (v) Signal to Noise (Ratio)

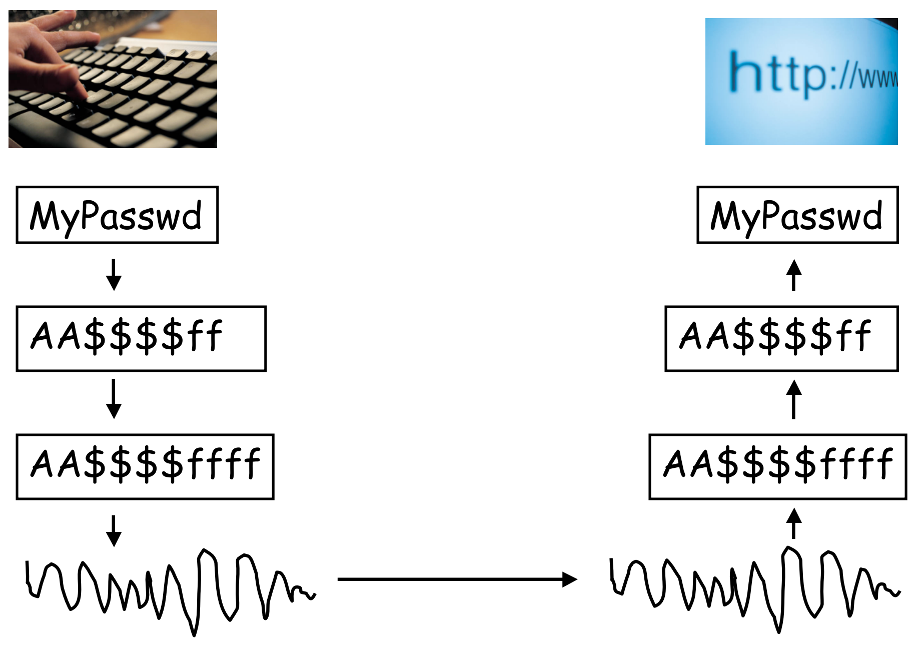
4-5 Compare two methods of synchronization, use at least four criteria

Digital Communications I

Topic 5-bis: Coding Data

Goal of Lecture

- Understand what coding is.
- Understand how error correction or detection codes work.



Coding

Change the representation of data.

$$\begin{tikzcd} \boxed{\text{Given Data}} \arrow[r, shift left, "\boxed{\text{Encoding}}"] & \boxed{\text{Changed Data}} \arrow[l, shift left, "\boxed{\text{Decoding}}"] \end{tikzcd}$$

Coding

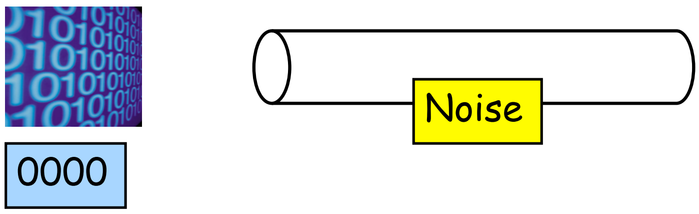
Change the representation of data.

$$\begin{tikzcd}\text{Given Data} \arrow[r, shift left, "\text{Encoding}"] & \text{Changed Data} \arrow[l, shift left, "\text{Decoding}"]\end{tikzcd}$$

1. Encryption: MyPasswd <-> AA$$$$ff
2. Error Detection: AA$$$$ff <-> AA$$$$ffff
3. Compression: AA$$$$ffff <-> A2$4f4
4. Analog: A2$4f4 <->

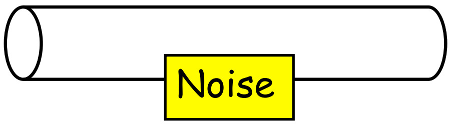
Error Detection and Correction

How to use coding to deal with errors in data communication?



Error Detection and Correction

How to use coding to deal with errors in data communication?



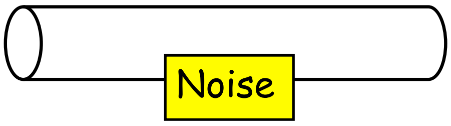
0000

Basic Idea :

1. Add additional information to a message.
2. Detect an error and re-send a message.
   Or, fix an error in the received message.

Error Detection and Correction

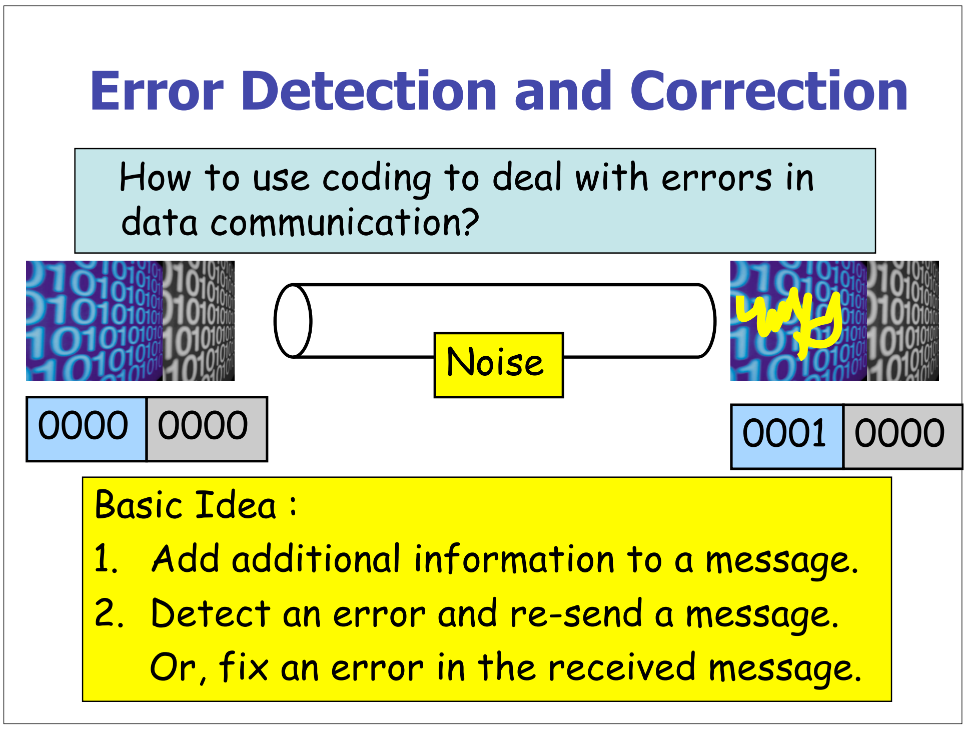
How to use coding to deal with errors in data communication?



0000	0000

Basic Idea :

1. Add additional information to a message.
2. Detect an error and re-send a message.
   Or, fix an error in the received message.



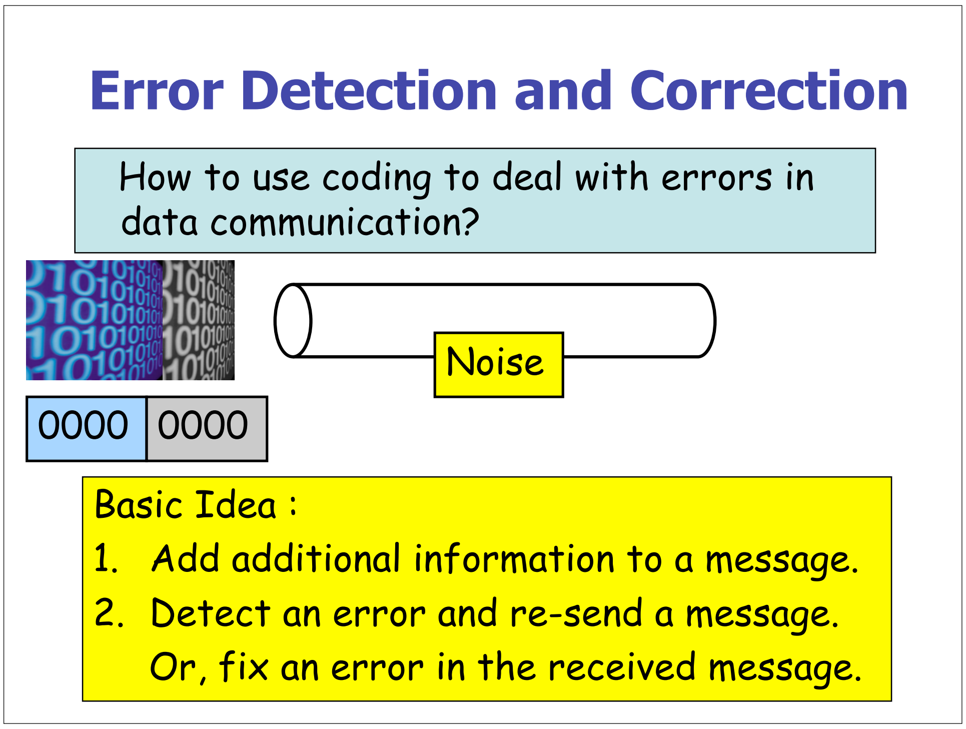
Error Detection and Correction

How to use coding to deal with errors in data communication?

Noise
0000 0000
0001 0000

Basic Idea :

1. Add additional information to a message.
2. Detect an error and re-send a message.
   Or, fix an error in the received message.



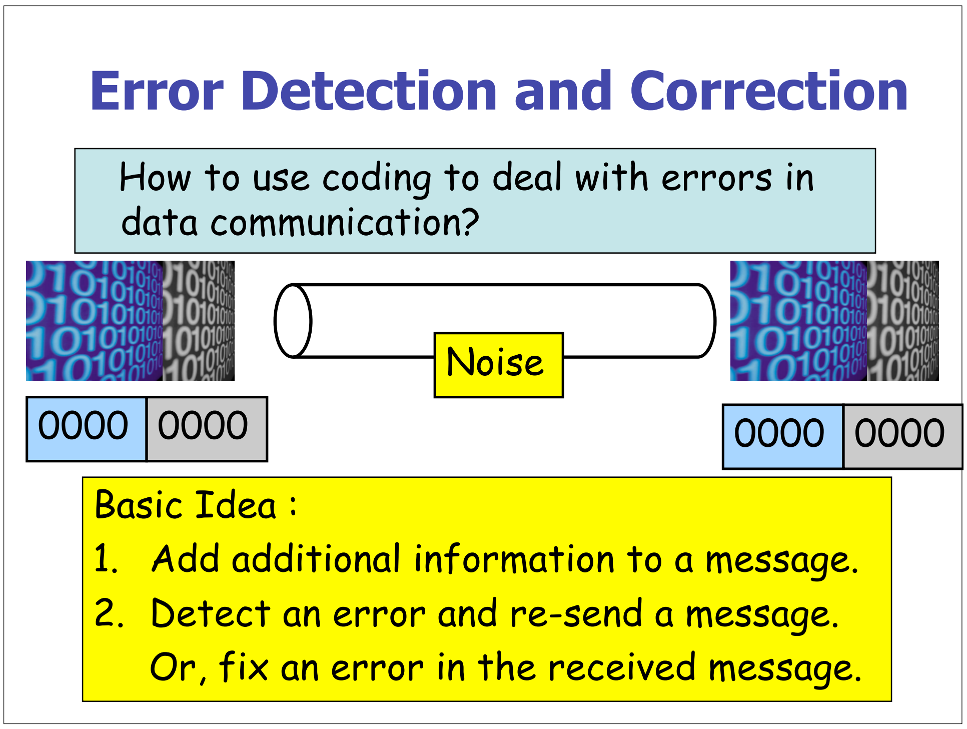
Error Detection and Correction

How to use coding to deal with errors in data communication?

Basic Idea :

1. Add additional information to a message.
2. Detect an error and re-send a message.
   Or, fix an error in the received message.

Noise
0000 0000



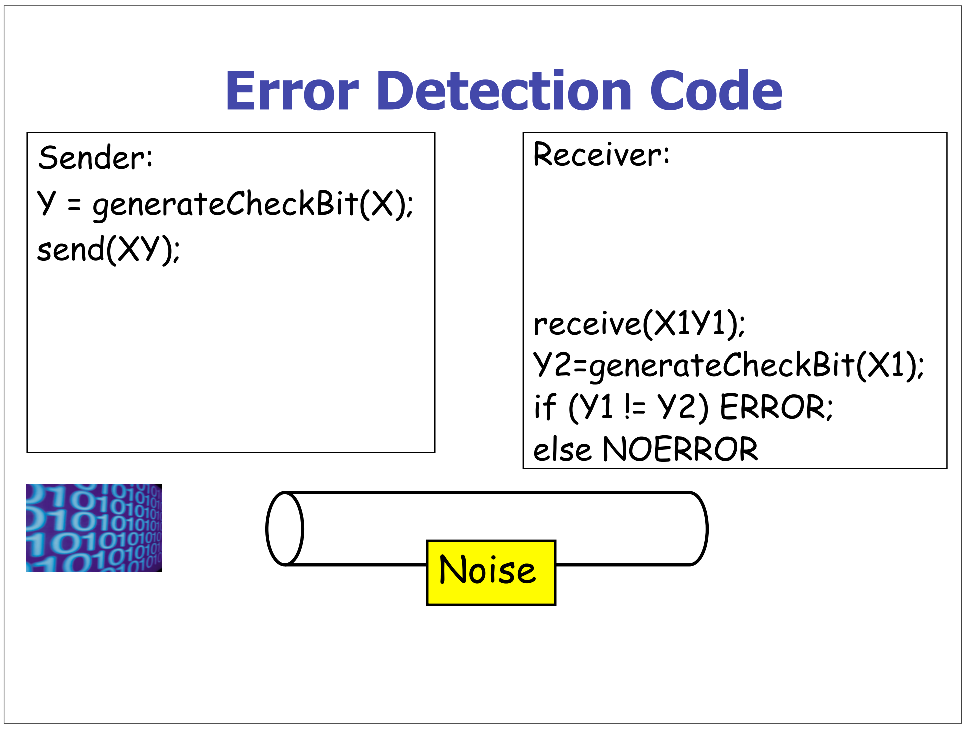
Error Detection and Correction

How to use coding to deal with errors in data communication?

0000 0000
Noise
0000 0000

Basic Idea :

1. Add additional information to a message.
2. Detect an error and re-send a message.
   Or, fix an error in the received message.



Error Detection Code

```
Sender:
Y = generateCheckBit(X);
send(XY);
```

```
Receiver:
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) ERROR;
else NOERROR
```

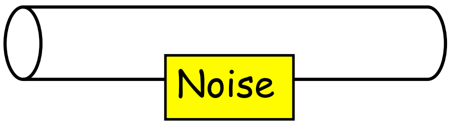
Error Detection Code

Sender:

```
Y = generateCheckBit(X);
send(XY);
```

Receiver:

```
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) ERROR;
else NOERROR
```



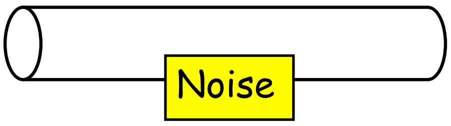
Error Detection Code

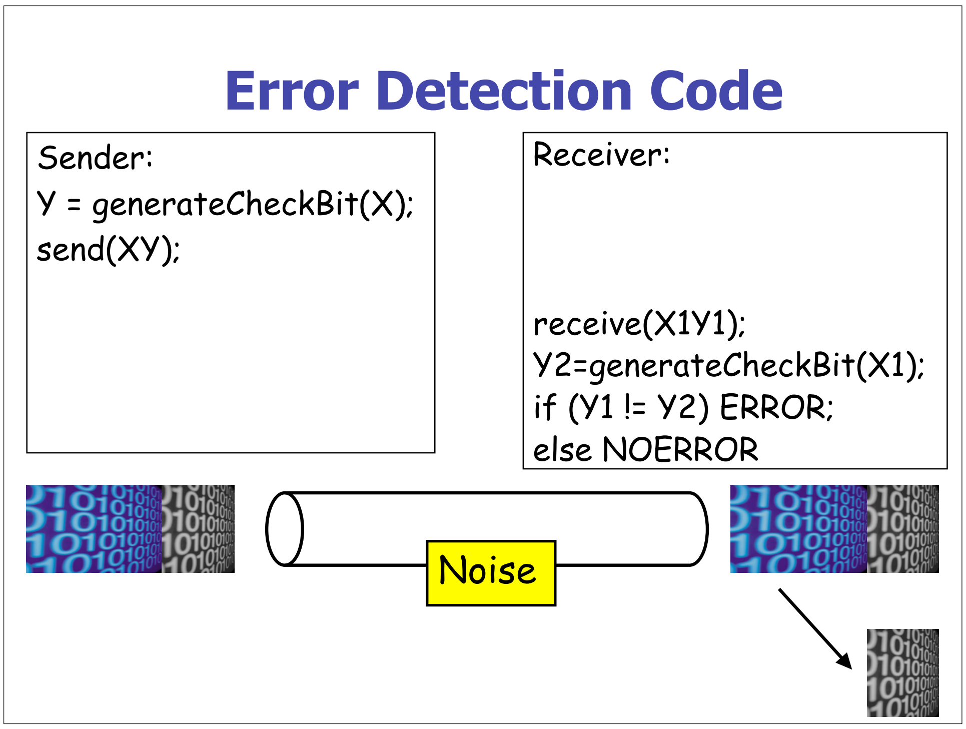
Sender:

```
Y = generateCheckBit(X);
send(XY);
```

Receiver:

```
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) ERROR;
else NOERROR
```

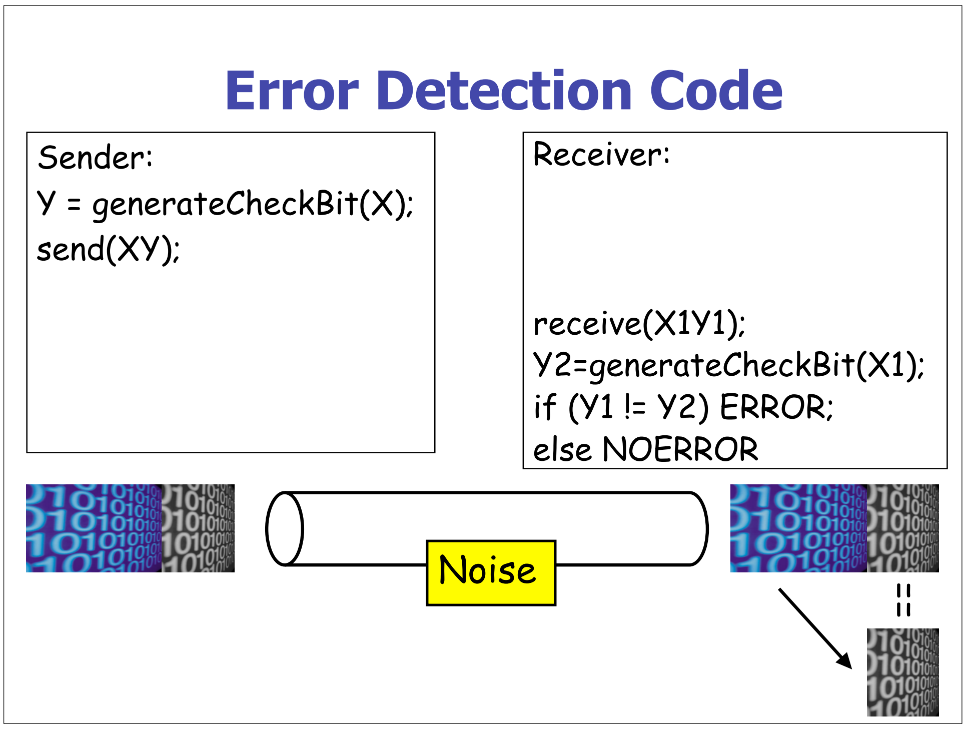




Error Detection Code

Sender:
```
Y = generateCheckBit(X);
send(XY);
```

Receiver:
```
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) ERROR;
else NOERROR
```



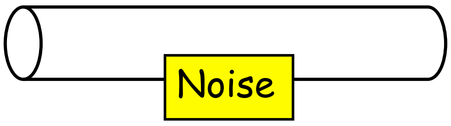
Error Detection Code

Sender:
```
Y = generateCheckBit(X);
send(XY);
```

Receiver:
```
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) ERROR;
else NOERROR
```

Error Detection Code: Parity

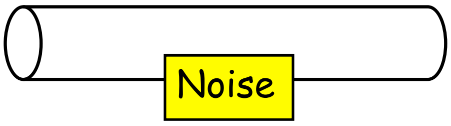
Add one bit, such that the number of 1's is even.



0000	0
0001	1
1001	0

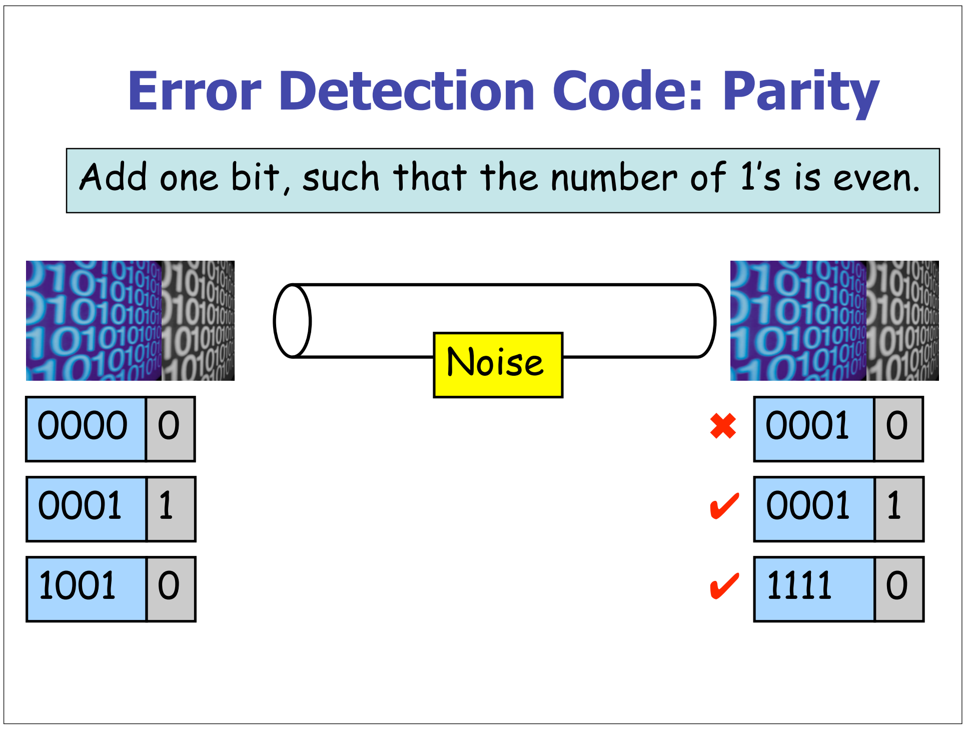
Error Detection Code: Parity

Add one bit, such that the number of 1's is even.



0000	0
0001	1
1001	0

0001	0
0001	1
1111	0

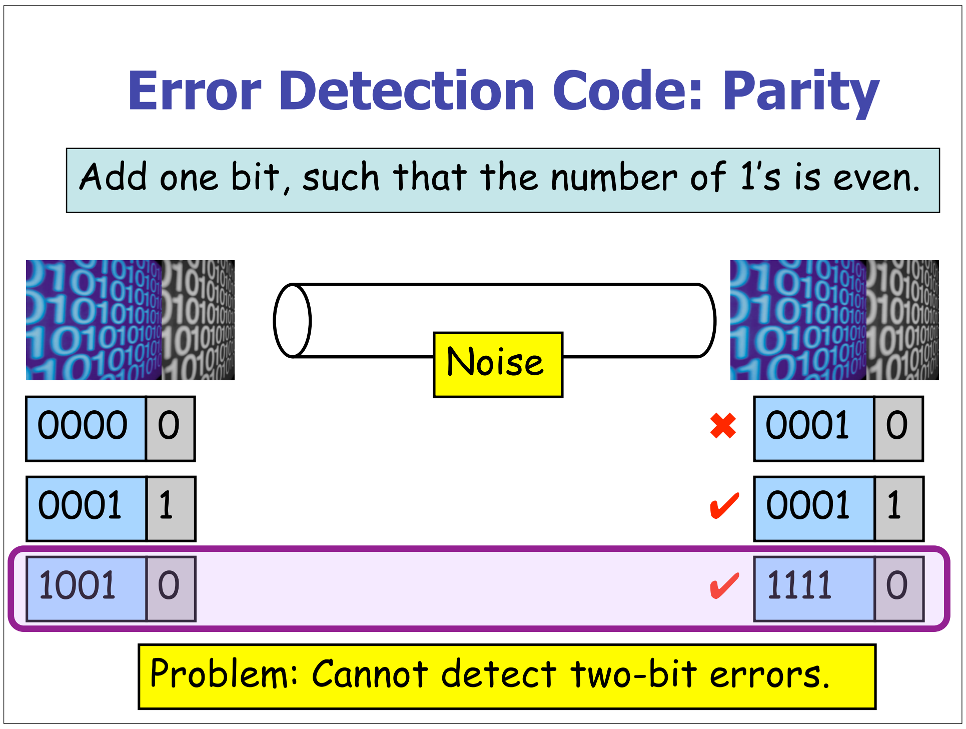


Error Detection Code: Parity

Add one bit, such that the number of 1's is even.

Noise

0000	0	✗	0001	0
0001	1	✓	0001	1
1001	0	✓	1111	0



Error Detection Code: Parity

Add one bit, such that the number of 1's is even.

Noise

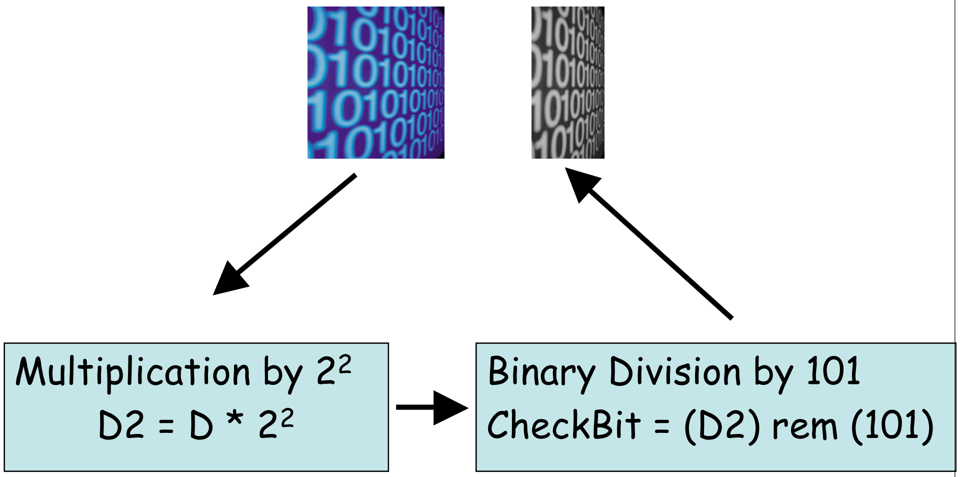
0000	0	✗	0001	0
0001	1	✓	0001	1
1001	0	✓	1111	0

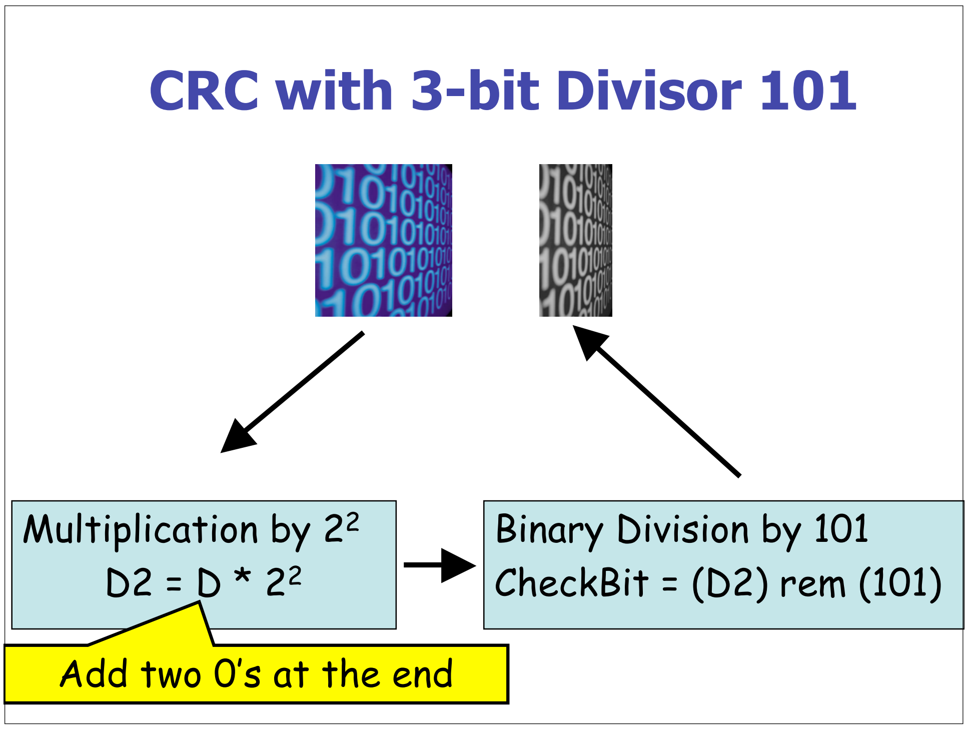
Problem: Cannot detect two-bit errors.

Error Detection Code: CRC

- CRC means “Cyclic Redundancy Check”.
- More powerful than parity.
  - It can detect various kinds of errors, including 2-bit errors.
- Uses more math: multiplication, binary division.
- Parameterized by n-bit divisor P.
  - Example: 3-bit divisor 101.
  - Choosing good P is crucial.

CRC with 3-bit Divisor 101





CRC with 3-bit Divisor 101

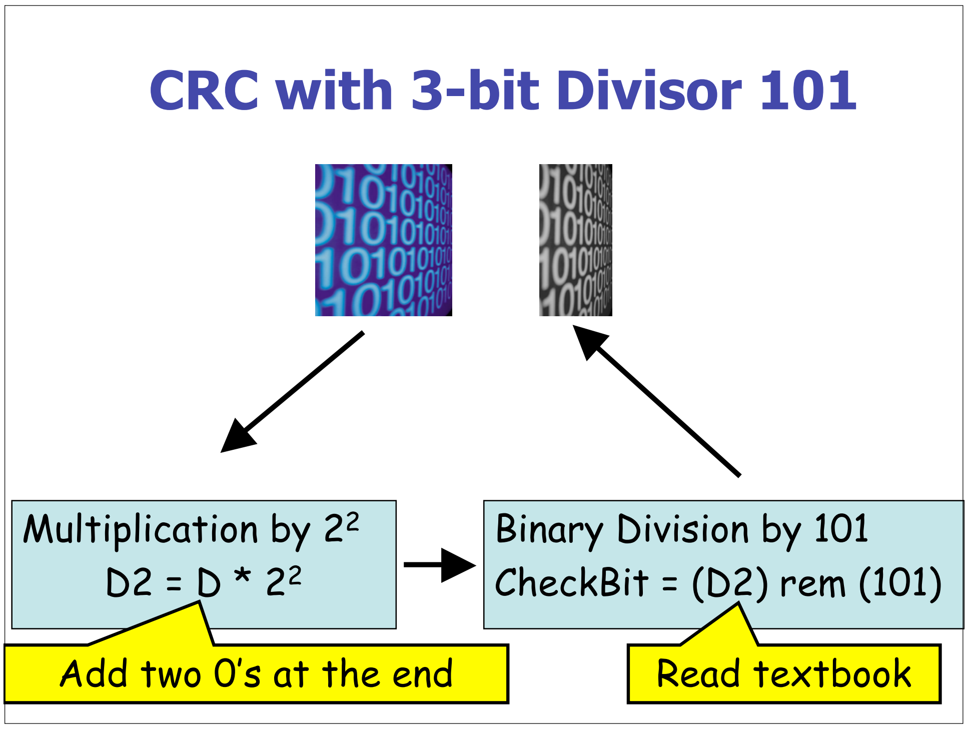
Multiplication by 2^2

$D2 = D * 2^2$

Binary Division by 101

CheckBit = (D2) rem (101)

Add two 0's at the end



CRC with 3-bit Divisor 101

Multiplication by 2^2

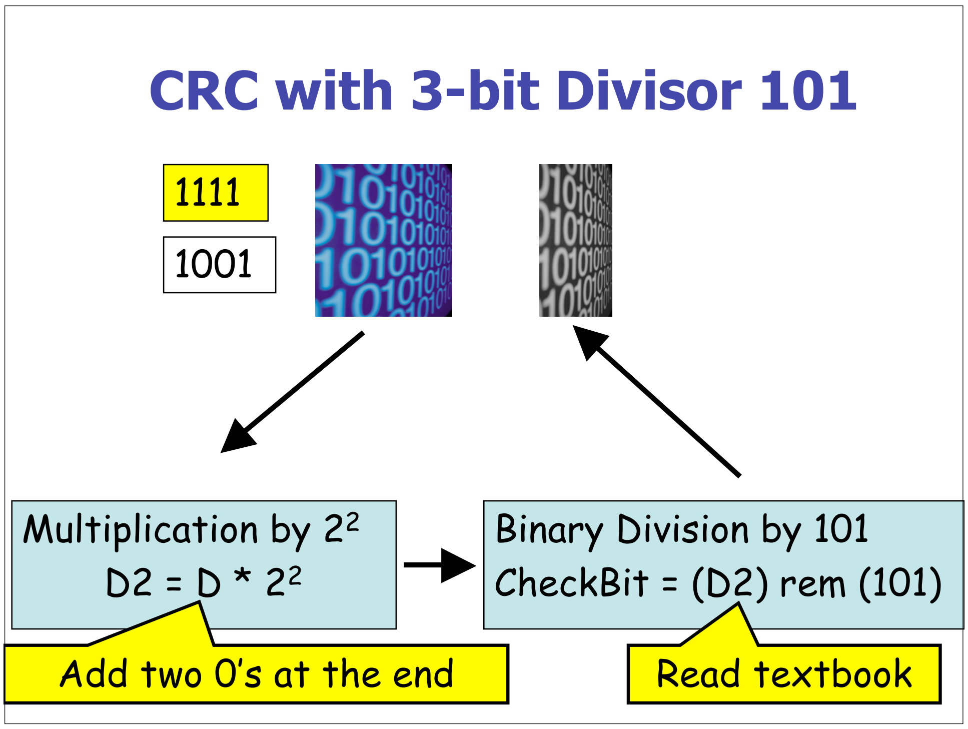
$D2 = D * 2^2$

Binary Division by 101

CheckBit = (D2) rem (101)

Add two 0's at the end

Read textbook



CRC with 3-bit Divisor 101

1111

1001

Multiplication by 2^2

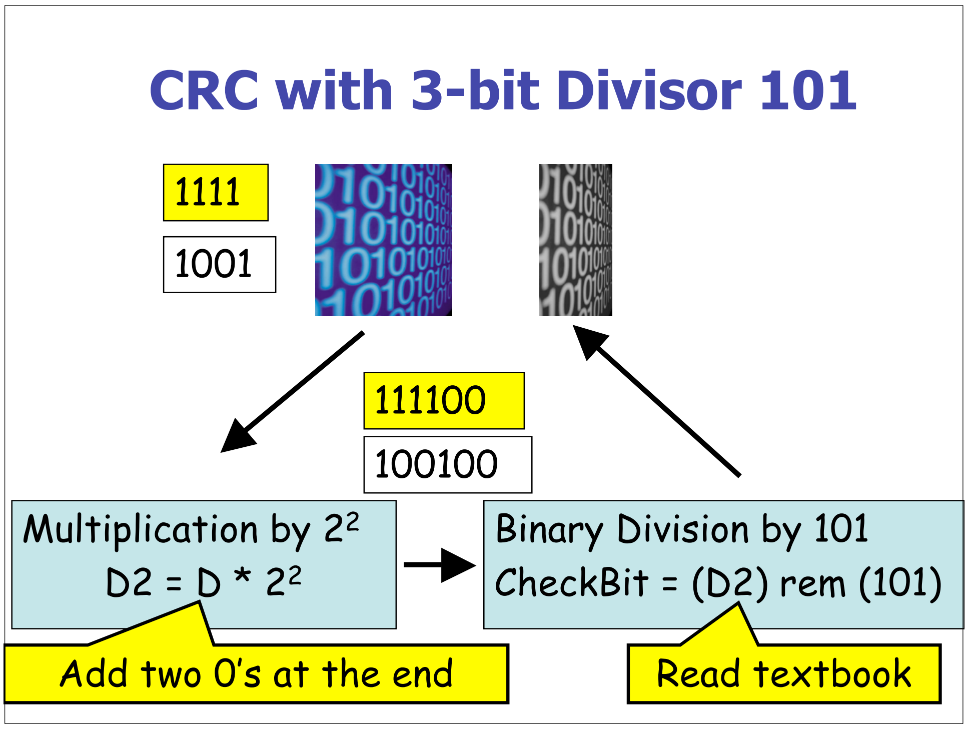
$D2 = D * 2^2$

Add two 0's at the end

Binary Division by 101

CheckBit = (D2) rem (101)

Read textbook



CRC with 3-bit Divisor 101

1111

1001

111100

100100

Multiplication by 2^2

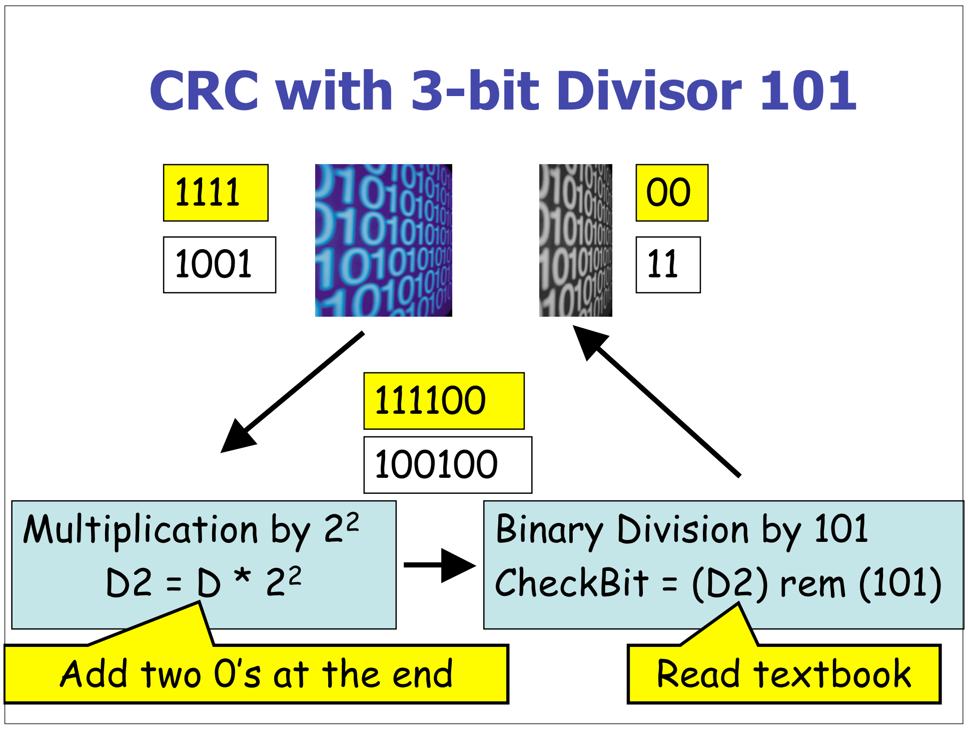
$D2 = D * 2^2$

Add two 0's at the end

Binary Division by 101

CheckBit = (D2) rem (101)

Read textbook



CRC with 3-bit Divisor 101

1111

1001

00

11

111100

100100

Multiplication by 2^2

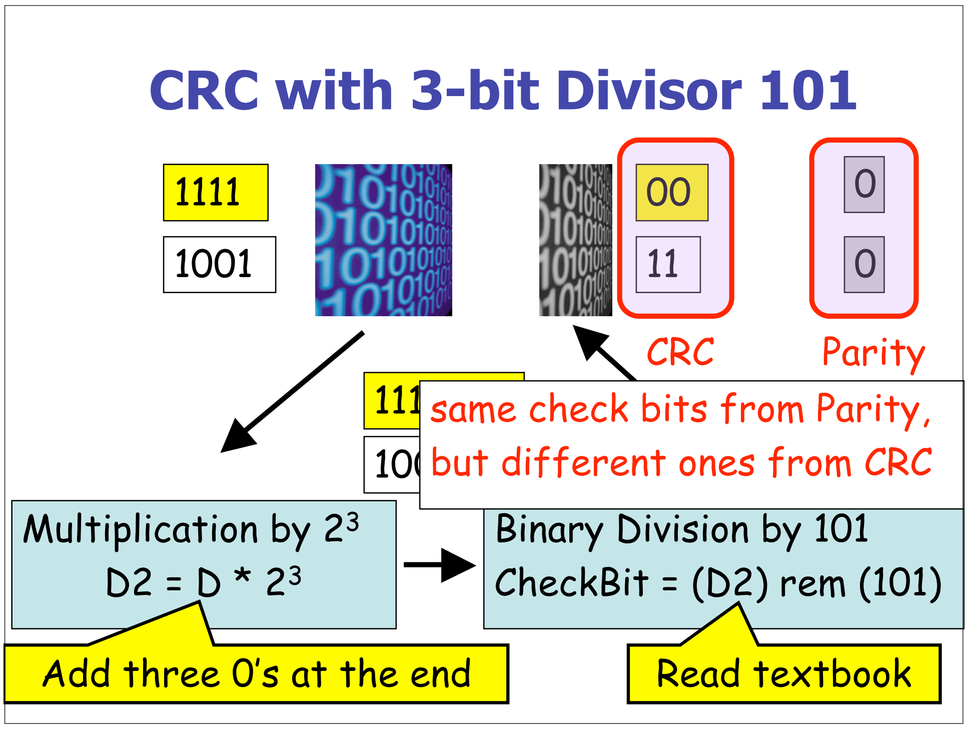
$D2 = D * 2^2$

Add two 0's at the end

Binary Division by 101

CheckBit = (D2) rem (101)

Read textbook



CRC with 3-bit Divisor 101

1111

1001

00

11

CRC

0

0

Parity

111

10(

same check bits from Parity, but different ones from CRC

Multiplication by 2^3

$D2 = D * 2^3$

Add three 0's at the end

Binary Division by 101

CheckBit = (D2) rem (101)

Read textbook

Divisor -- Secret of CRC

- If the divisor were 100, instead of 101, data 1111 and 1001 would give the same check bit 00.
- Mathematical analysis about the divisor:
  - Last bit should be 1.
  - Should contain at least two 1's.
  - Should be divisible by 11.
- ATM, HDLC, Ethernet each use a CRC with well-chosen fixed divisors.

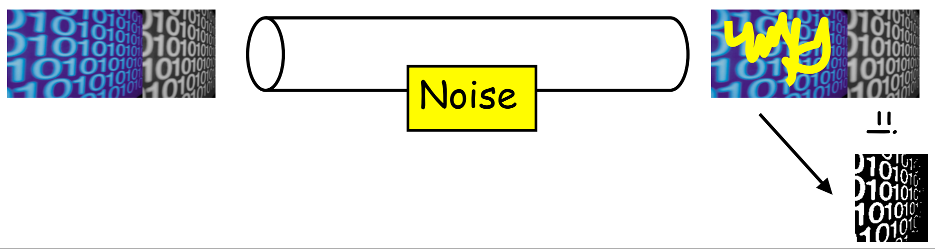
Forward Error Correction (FEC)

```
Sender:
Y = generateCheckBit(X);
send(XY);
```

```
Receiver:



receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) FIXERROR(X1Y1);
else NOERROR
```



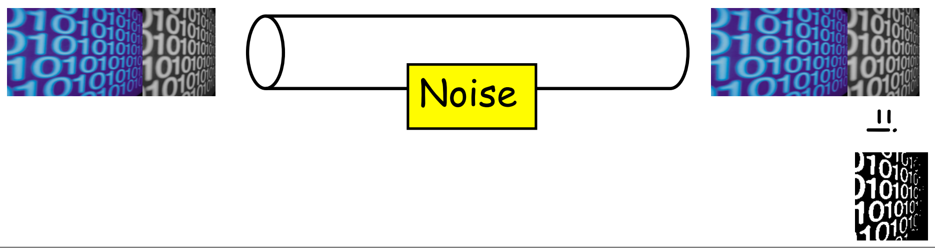
Forward Error Correction (FEC)

Sender:

```
Y = generateCheckBit(X);
send(XY);
```

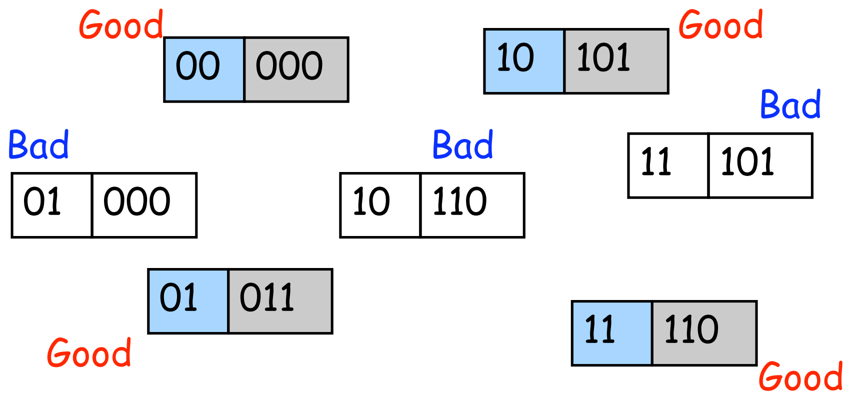
Receiver:

```
receive(X1Y1);
Y2=generateCheckBit(X1);
if (Y1 != Y2) FIXERROR(X1Y1);
else NOERROR
```



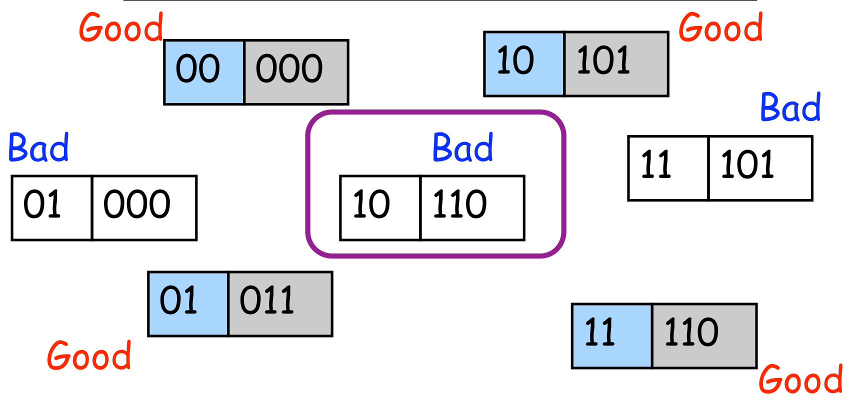
Basic Idea of Forward Error Correction

Replace erroneous data
by its "closest" error-free data.



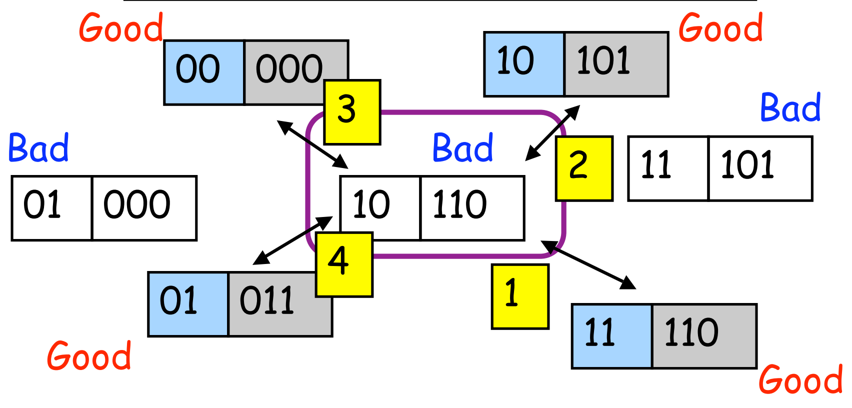
Basic Idea of Forward Error Correction

Replace erroneous data
by its "closest" error-free data.



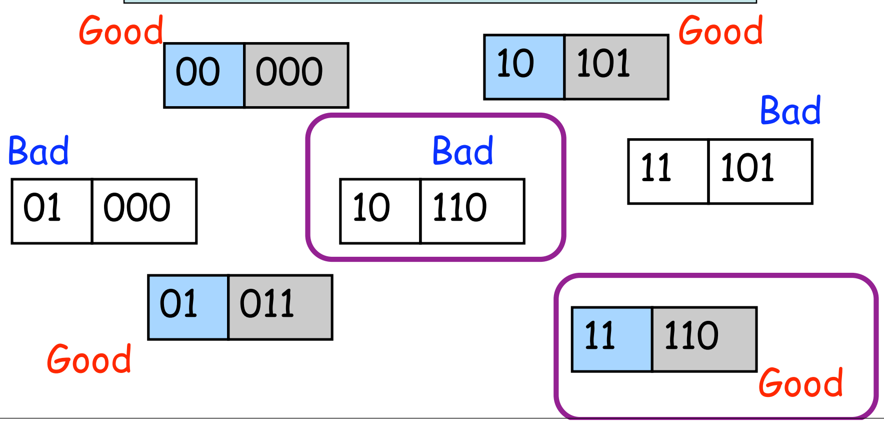
Basic Idea of Forward Error Correction

Replace erroneous data
by its "closest" error-free data.



Basic Idea of Forward Error Correction

Replace erroneous data
by its "closest" error-free data.



Error Detection vs Correction

Error Correction:

- Cons: More check bits. False recovery.
- Pros: No need to re-send.

Error Detection:

- Cons: Need to re-send.
- Pros: Less check bits.

Usage:

- Correction: A lot of noise. Expensive to re-send.
- Detection: Less noise. Easy to re-send.
- Can be used together.

Topic 5: Coding Data

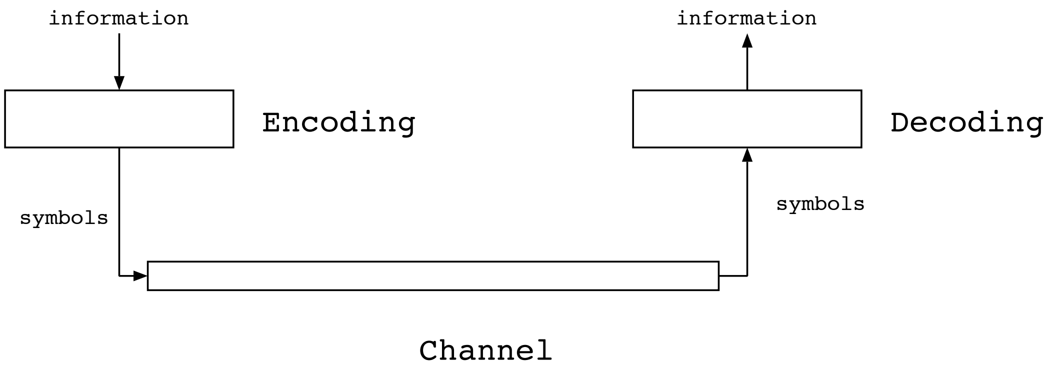
Previous topic

- Transmission Media
- Characteristics (Fundamental Limits, Noise, Attenuation), Bandwidth vs Signal to Noise)
- Digital Channels
- Synchronization
- Modulation

Overview of this topic

- Coding
- Digitization and Sampling
- Error Detection and Correction
    - Block Codes
    - Information Theory
    - Compression
    - Encryption

Coding



- *encoding* takes *information* and turns it into *symbols* which are to be transmitted on a channel

- *decoding* takes symbols from a channel and turns them into information for a higher layer

- symbols for one entity is information for another

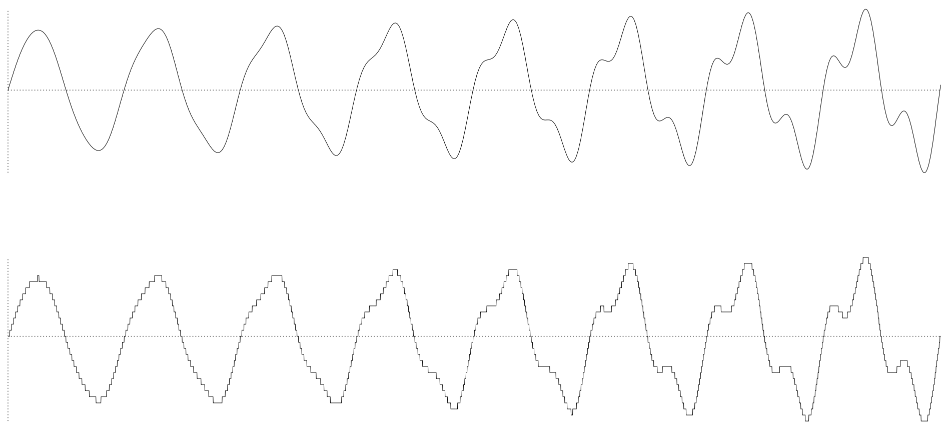
- using *coding* as a general concept; includes A/D, modulation, etc

Examples

- *modulation* used to represent digits on a physical medium;
- *digitization of analog signals*;
- *error detection and correction codes* used to determine if a transmission error has occurred, and perhaps to correct it;
- *compression coding* used to reduce the amount of data (and therefore usually the data rate) which needs to be sent;
- *cipher coding* used to protect the privacy of information as it travels through a channel, and
- *delineation coding (framing)* used to group symbols into logical units,
- *application level coding* which provides applications with an agreement as to how data items are to be represented.

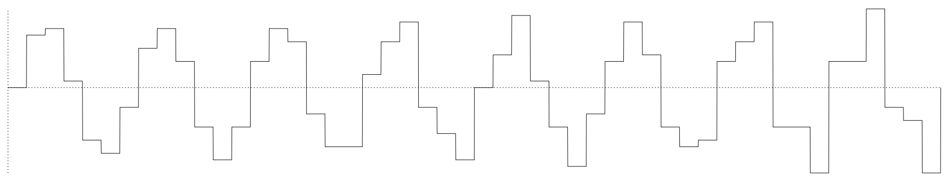
Digitization of Analog Signals

- D/A, A/D different from modulation, demodulation
- digital because:
  - ease of handling and storage
  - ease of time dilation
  - noise (bit error rate) is not linearly cumulative with processing and in principle (and practice) can be reduced to arbitrary levels
- introduces a one time quantisation noise



Sampling

- also have discrete time — *sampling*



- how often should we sample?
- if we have a signal which is frequency limited to B Hz then if we sample without error at a rate $2B$ Hz then we can reconstruct the signal

Error Detection and Correction

- noise in physical channels give rise to bit errors
- some errors are acceptable in some applications — in particular when it just adds noise to an analog signal which has been digitised.
- other applications are less tolerant
- either reduce error rate to acceptable level, or detect occurrence of an error and retransmit information found to be in error.
- reducing error rate might mean changing modulation technique or it may mean using error correction coding — sometimes called forward error correction (FEC)

Forward Error Correction

- relies on providing *redundant* information
- trivial example, send every bit of information three times and use majority voting. (improves error rate but burns up data rate)
- *block codes* are an important class of forward error correcting codes in which information is divided into chunks.
- *convolution codes* which operate on continuous streams are the other important class of FECs

Block Codes

- information is divided into fixed sized messages, say of length m
- messages are encoded into larger *codewords* of size k:

$$\mathcal{E} : \{0,1\}^m \longrightarrow \{0,1\}^k$$

  where$\{0,1\}^m$ denotes a bit string of length m.
- encoding mapping is one to one, but not onto; in other words for each message there is a unique codeword, but not all elements of $\{0,1\}^k$ are valid codewords.
- set of valid codewords is called the *code*.
- referred to as a (m,k) code.
- *distance* between two codewords is number of bits in which they differ
- *minimum distance*, d of a code is the minimum distance between any pair of codewords

- *decoding* is done by looking at the received codeword, which may be corrupted, and finding the closest valid codeword (and then finding the message that encodes to that valid codeword.
- can refuse to decode if sufficiently far away from a valid codeword, ie just report as an error

- given minimum distance d can either detect up to $d-1$ errors or correct $(d-1)/2$ errors or have an “inbetween” strategy.
- decoding is the hard part

A simple $(16, 25)$ Code

1	0	0	1	0
1	1	1	0	1
0	0	0	1	1
1	1	0	1	1
1	0	1	1	1

- upper left hand 16 bits are info
- bottom row, right hand column simple (even) parity
- minimum distance is 3
- can correct all single bit errors(but will incorrectly correct a two bit error)
- code rate is $\frac{m}{k}$, ie here $\frac{16}{25}$
- this code is $systematic$ since the information appears "in the clear" inside the code — simplifies decoding.

Amazing Result of Information Theory

- given a digital channel with a bit error probability of of P_e, and any desired probability of correct decoding there exists a code with rate R arbitrarily close to

$$R < C = 1 + P_e \, log_2 \, P_e + (1 - P_e) log_2 \, (1 - P_e)$$

  that will do it (note the log terms are negative).
- in fact, the definition of information for a binary channel with data rate D is

$$I = D(1 + P_e \, log_2 \, P_e + (1 - P_e) log_2 \, (1 - P_e))$$

- note that for $P_e = \frac{1}{2}$, I is 0 and that even for poor error rates of $P_e = 10^{-3}$, the capacity is greater than $.98 \times D$.

the bad news...

- this result doesn't tell us what the codes are
- it doesn't tell us how to build simple decoders
- codes which approach the limits get arbitrarily large, ie m and k are huge.

the good news ...

- keeps the mathematicians in business.

Error Detection

- eg simple parity on bytes an (8,9) code with a distance of 2
- non block codes which work over variable information size usually with check bits which are a function of the information bits
- transmitter generates and transmits check bits; receiver computes check bits and compares them with the ones the transmitter sent
- eg checksum — weak because dependency of some check bits on information bits is skewed $\longrightarrow$ low minimum distance (like 2!)
- cyclic redundancy check (CRC) spreads the dependency more uniformly across check digits $\longrightarrow$ better distances
- CRC based on polynomials with binary coefficients using arithmetic modulo 2

CRC — see also Stallings

message considered as a polynomial, eg $M = 01101$ becomes

$$M(x) = 0x^4 + 1x^3 + 1x^2 + 0x + 1 = x^3 + x^2 + 1$$

shifting is multiplying by x.

We have a special polynomial $P(x)$ with degree equal to the number of check bits. Call this number n

We wish to transmit a message $M(X)$ of size k bits. We find the remainder

$$R(x) = x^n M(x) \mathrm{rem}\ P(x)$$

and send

$$T(x) = x^n M(x) + R(x)$$

receiver sees a possibly corrupted version of $T(x)$, call it $T'(x)$. Note than $T(x)$ rem $P(x)$ is 0. So the receiver divides $T'(x)$ by $P(x)$ and is happy if the remainder is 0.

The difference in what was sent and received is

$$E(x) = T(x) - T'(x)$$

so we are interested in non zero values for $E(x)$ which are divisible by $P(x)$. These will fool the CRC algorithm.

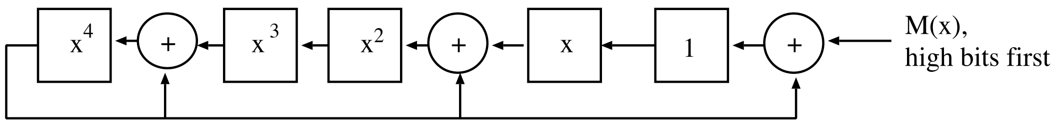
CRC Properties

- If $P(x)$ has both first and last terms nonzero it will not divide a poly of the form $E(x) = x^i$. Such a CRC will thus detect all single bit errors
- If $P(x)$ has a prime factor with three terms it will not divide a poly of the form $E(x) = x^i(1 + x^j)$. Such a CRC will thus detect all double bit errors
- If $P(x)$ also has a factor $(x + 1)$ it will not divide an $E(x)$ poly with an odd number of terms.
- won't divide a $E(x)$ of the form $x^i B(x)$ where $B(x)$ has its last term equal to one if $B(x)$ has order less than n. (burst errors)

CRC Realisation

$$P(x) = x^5 + x^4 + x^2 + 1$$

adding modulo 2 is exclusive or; multiplying by x is shifting.



- each shift is like a step in long division
- if the x^4 bit is set, then when it is shifted out it causes $x^4 + x^2 + 1$ to be subtracted as next bit of M(x) is shifted in
- note leading zeroes in M(x) are unhelpful!
- moderately painful in software

CRC Realisation Example

- divide $x^6 + x^3 + x^2 + x$ by $P(x)$
- initialise shift register to all zeros and shift 1001110 in
- shift register contents

  00000
  00001
  00010
  00100
  01001
  10011 note 1 now in high bit so will combine next shift
  10010 ditto
  10001
- so remainder is $x^4 + 1$

Compression

- perfect *versus* imperfect, does

$$\mathcal{D}(\mathcal{E}(x))) = x$$

- if imperfect, is it stable or unstable:

$$\mathcal{D}(\mathcal{E}(\mathcal{D}(\mathcal{E}(\mathcal{D}(\mathcal{E}(\ldots\mathcal{D}(\mathcal{E}(x))\ldots)))))) = \text{anything like x}$$

- imperfect coding must exploit knowledge of information: digital telephony 12 bit linear samples are mapped logarithmicly onto 8 bit samples $\longrightarrow$ more samples to represent values near zero. (constant rate code)
- also can eliminate silence periods (no longer constant rate)

perfect coding: run length, Huffman, compress(1), gzip(1), difference coding in video

Huffman identifies the more common symbols in a stream and finds a compact representation of those symbols.

Video Compression

- try to eliminate redundancy, code more efficiently
- eg if image doesn't change between frames, don't send
- transform image to find frequency content
- **gore:** *transform coding* works on 8 by 8 pixel tiles, requantizes at course resolution and then gets rid of frequencies with zero content
- eg tile all the same colour only has one frequency component
- eg JPEG (joint photographic experts group) uses this method to compress single images
- motion JPEG uses JPEG on successive frames, it only looks at one frame at a time (intraframe coding)
- MPEG does this and more
  - MPEG-1 1.5 Mbps (CD-ROM speeds)
  - MPEG-2 15 Mbps (compressed HDTV)
  - MPEG-4 ?? (very high level descriptions)

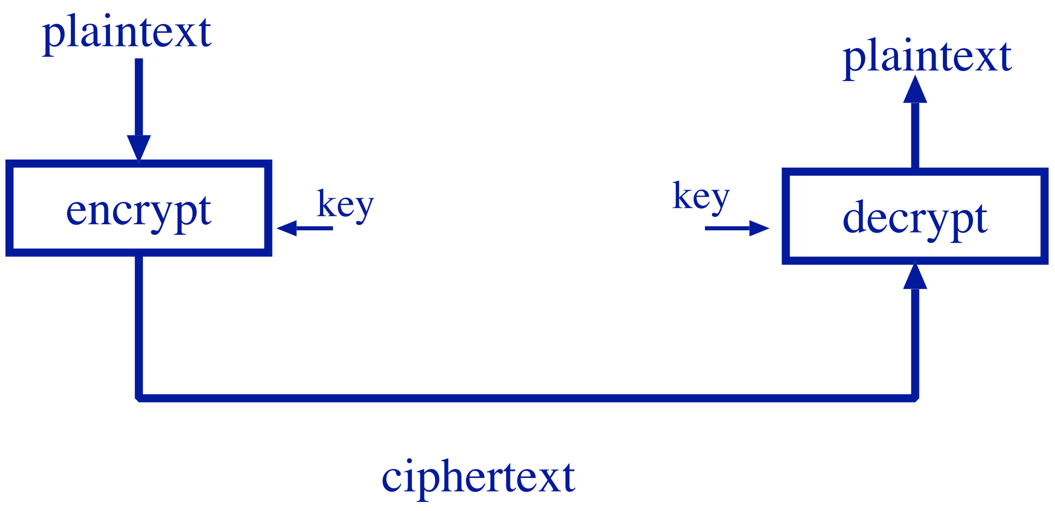
MPEG - basics

- take advantage of similarity between successive frames, ie interframe coding (sounds simple)
- including predicting bits of the picture moving about (sounds complicated)
- different frames:
  - I-frames (intraframe) which are just like JPEG (close enough)
  - P-frames (predictor) tiles are coded as being a tile from a previous frame but moved (and some difference information)
  - B-frames (bidirection interpolated) average of tiles (with motion) from previous and future frames
  - I-frame every 8 frames: I 2B P 2B P 2B I
- can vary quality and get same coded data rate
- can keep quality constant and get variable data rate
- nasty to encode, not too bad to decode

Encryption

- well known $\mathcal{E}$ and $\mathcal{D}$ functions which take an extra argument which is the key
- encoding is called $encryption$, decoding $decryption$.
- moving keys around is interesting (but nothing to do with coding)
- may be interested in only certain people being able to decrypt (in which case the decryption key should be secret) or certain people being able to encrypt (encryption key secret) or both

Symmetric Key Cryptosystems



$$ciphertext = Encrypt(plaintext, key)$$

$$plaintext = Decrypt(ciphertext, key)$$

symmetric — both sender and receiver know the same secret key

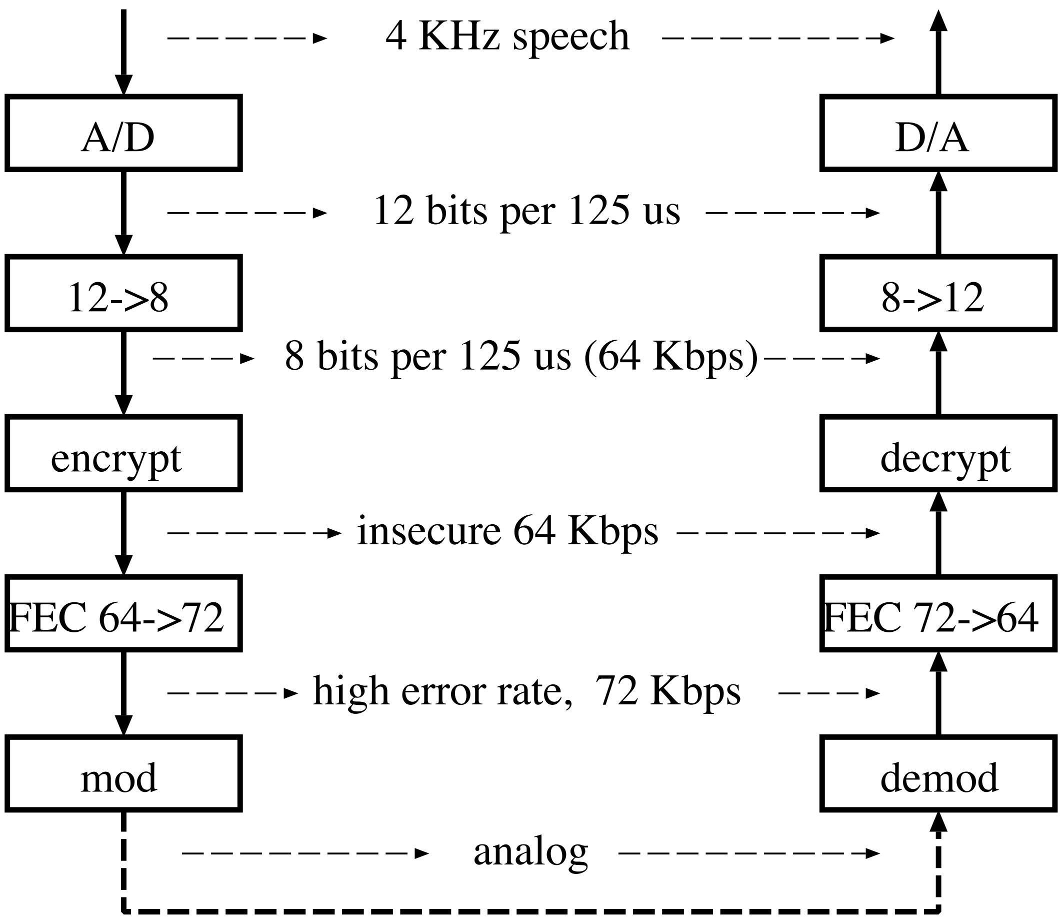
Using Symmetric Secret Keys

- **Authentication**
  - Ann $\longrightarrow$ Bob: Challenge
  - Bob $\longrightarrow$ Ann: Encrypt(Challenge, Key)
- **Integrity**
  - Ann $\longrightarrow$ Bob: Message, Encrypt(F(Message), Key)
- **Confidentiality**
  - Ann $\longrightarrow$ Bob: Encrypt(Message, Key)

Other forms of Coding

- coding to aid modulation eg 4B5B or scrambling
- application level coding — negotiation about coding
- framing, bit stuffing, “code violations”

Grand Example



Exercises

5-1 List 5 types of coding you encounter everyday, indicate the communications channel, the type of information being encoded, the type of coding taking place, and the type of (now coded) information carried by the communications channel

5-2 Why are CDs recorded at a sampling rate of 44.1kHz ?

5-3 Serial data is used to represent 8 data-bits, however only 7 bits are needed to represent the common ASCII data set (the data representing common punctuation, A-Z, a-z, 0-9 and so on). As a result serial data can use the eight bit to detect errors, this error detection is called parity. Parity can be described as $None$ which means it is unused, Odd which means that the parity data bit is changed so that the sum of all 8 data bits is an odd number. Eg. data: 0100101 would get a parity of 0, while data: 1100000 would get a parity of 1. The final type of parity is $Even$ which like Odd changes the value of the parity data bit, so that the sum of all 8 data bits is an even number. E.g. data: 1110001 will get parity 0, while data: 0000001 gets parity 1.

An example of an N,K block code (N=7 and K=8), show how a lookup table may be used to implement Odd parity.

5-4 Compression

(i) Compare Lossy and Lossless compression

(ii) Compare Temporal and Spatial Compression

(iii) Explain why lossy compression is commonly used for Audio and Video (e.g. mp3 audio, MPEG video, and used in Digital TV)

5-5 Encryption

(i) Describe how an encryption mechanism can be used to verify a stranger can be trusted.
(ii) Describe how an encryption mechanism can be used to verify an email from a friend.
(iii) Describe how an encryption mechanism can be used to allow two friends to safely share a secret.

Topic 6: Multiplexing

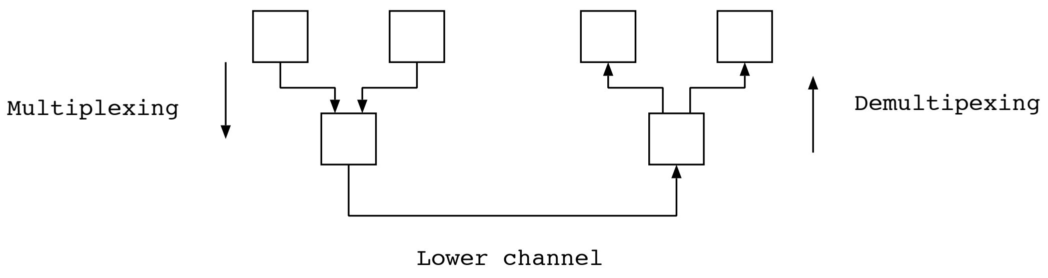
Previous topic

- Coding
- Digitization and Sampling
- Error Detection and Correction
    - Block Codes
    - Information Theory
    - Compression
    - Encryption

Overview of this topic

- Multiplexing
- Common-Place Example
- Time Division Multiplexing
- Local Area Network Example
- Shared vs. Non Shared Media

Multiplexing



- multiplexing is about how a channel is shared out
- can view it as producing a number of higher layer channels from a lower layer channel

- *coding* is required to distinguish the higher layer channels from one another

- *policy* is required to determine who uses what part (space, time, frequency, etc) of the lower layer channel

A Familiar Example

Broadcast Television and Radio

- lower layer channel is free space
- different upper layer channels use different frequencies
- (actually more than just television and radio ...)
- *coding* (ie choice of frequency) enables receivers to distinguish channels
- *policy* by the Home Office assigns transmitters to frequencies
- this is an example of Frequency Division Multiplexing (FDM)
- FDM can also be used in a guided fashion, eg cable TV and come to think of it, the cable attached to your antenna.
- frequencies are reused in geographically dispersed regions, we can look upon this as space division multiplexing (policy has to include power output).

FDM

(Frequency division multiplexing)

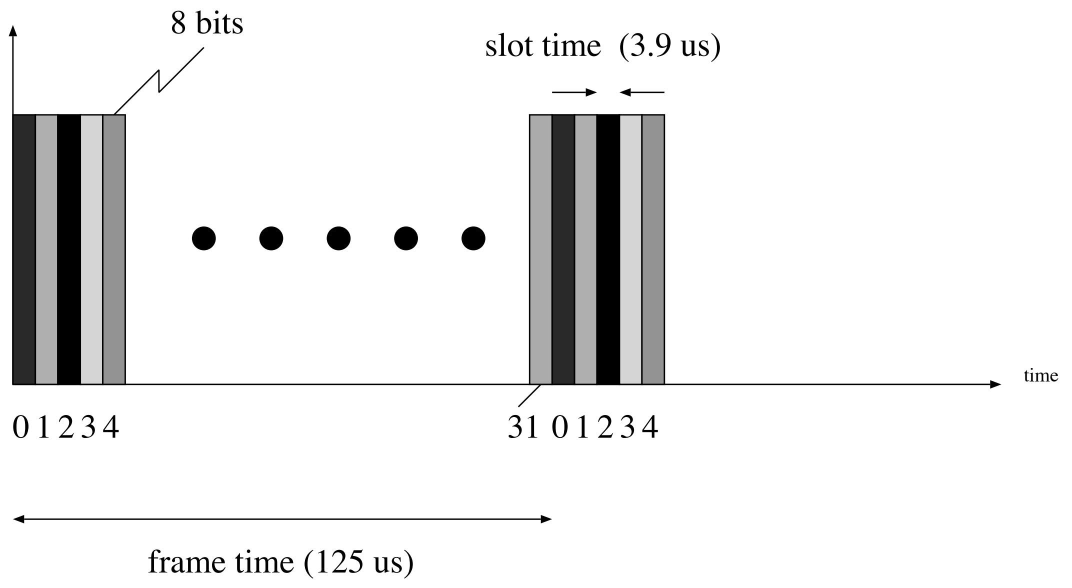
- based on *orthogonality* of sinusoids
- wave division multiplexing uses different wavelengths of light in fibre optics (different lasers for transmission, defraction gratings at receiver to separate out)
- tend to think of turning light on and off, but this is still modulation of the carrier
- WDM (wave division multiplexing), commonly used by telecommunications companies, is just FDM with higher frequencies

Time Division Multiplexing

- transmit different channels at different times
- think of television programmes on the same television channel
- television channel is lower layer channel
- series of episodes of Neighbours is a higher layer channel (for some value of higher)
- receivers discriminate by content or looking at the schedule; coding is by positioning in time.
- policy by programme controller.
- note orthogonality by non overlapping time
- $grain$ of multiplexing is length of continuous time for which higher layer channel occupies lower layer channel, ie on TV around 30 minutes, but variable.

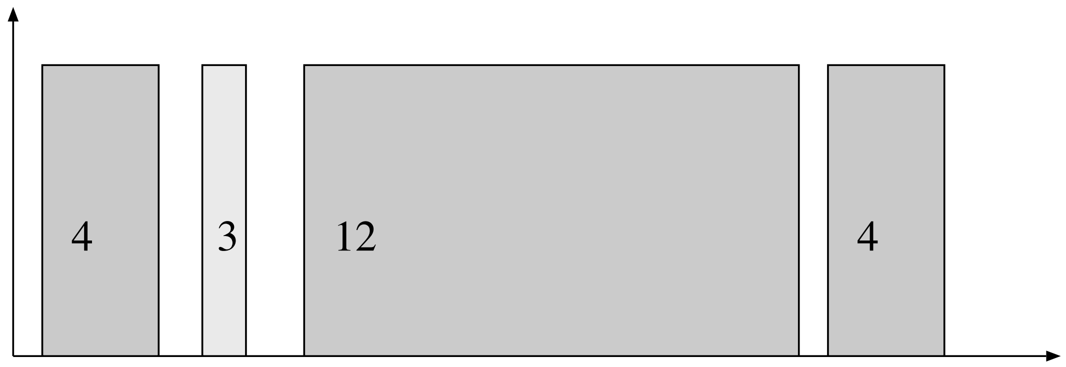
Synchronous Time Division Multiplexing

- in *synchronous time division multiplexing* the times at which a higher layer channel uses the lower channel are strictly periodic. receiver can tell where the higher layer channels are by looking in time; the symbols in the channel do not have to contain identification
- best example is digital telephony



- synchronous multiplexing provides for constant bandwidth, constant delay channels (just like FDM does, just like physical channel does)
- call a constant bandwidth, constant delay channel a *circuit*
- some people say "real" circuit to distinguish from virtual circuits
- circuits are great for carrying constant rate information

Asynchronous Time Division Multiplexing



- appropriate when demands from higher layer channels are varied
- if we have 100 workstations each of which occasionally requires 5 Mbps but average 100 Kbps we can give them each 5 Mbps circuits, or we can give them 10 Mbps to share.
- *statistical multiplexing* hoping that peak demands from different higher layer channels do not occur at the same time.
- three immediate consequences:
  - symbols in the channel must be grouped into *packets*
  - packets must contain labels identifying them
  - possibility of contention; some policy is required to resolve this
- labels are *addressing* information

Async (cont'd)

- policies for dealing with contention are varied
- on point to point links, or within an end system (host) these policies are implemented by the multiplexing entity.
- in the case of shared media access these policies are implemented in a distributed fashion (most of what is about to follow)
- contention means channels are not fixed delay and are not fixed capacity.
- packet switching
- rate adaption

Random Access

- consider conversation in a room
- an individual speaks by listening for a silent period and then talking
- if two people talk at the same time, then they *collide*. One or both stops; if both then they try again (and possibly recollide)
- time to realise collision has occurred is twice the delay of the channel
  - one delay for transmission to get from one end to the other (and thus stopping the other end from starting)
  - one delay to ensure that the other end didn't start just before the end of the first delay
- if collision time is less than the packet time then we detect collisions and stop transmitting, eg Ethernet, room conversations
- otherwise all we can do is listen for collision later and retry, eg Aloha (and satellite telephone conversations)

Random Access - continued

- some collisions are nondestructive, eg Hubnet, COPE wireless *XOR in the Air*

- advantages:
  - control is simple and distributed, very fault tolerant
  - access to channel is quick when lightly loaded
- disadvantages:
  - access to channel is not bounded under high loads
  - although potentially “fair” no easy way exists to skew fairness according to need

Token Passing

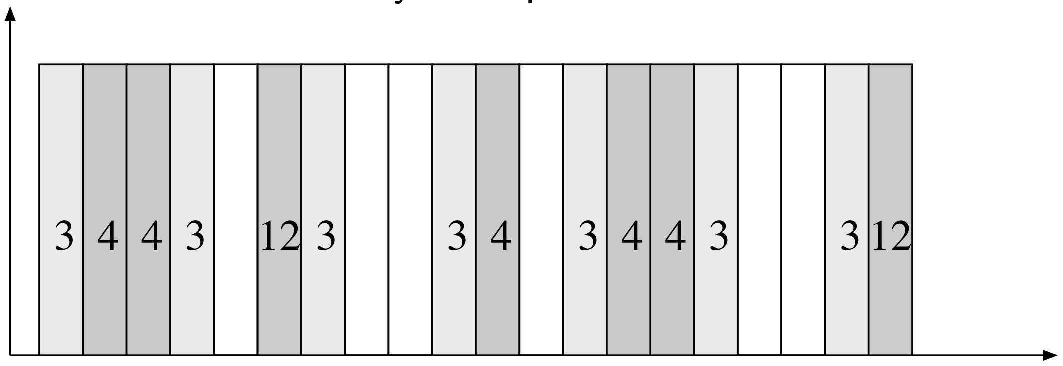
- transmitter must have $token$ in order to transmit
- when through, token is passed along
- eg in a two way radio conversation "over"
- usually in a ring (either physical or logical) to ease decision of to whom to pass token
- problems with
  - token disappearing
  - tokens getting duplicated
- examples: token ring, FDDI, token bus

Reservation Systems

- reserve time in advance
- synchronous TDM is an example (everyone knows which reservation belongs to who)
- in general, transmitters need only know their reservations
- useful when delays are large (eg satellites)
- still need a channel for reservation requests which has some other mechanism (such as random)

Slotted Systems

- channel is divided into fixed sized *slots* (just like in synchronous TDM)
- access to the slots is asynchronous ie transmitters contend for them in a statistically multiplexed fashion



- different schemes for access are possible:
  - slots are marked free or busy, a transmitter seeing an empty slot may fill it
  - may be a limit on how many slots a transmitter may use at one time
  - this limit may be adjustable (possibly by reservation)
  - possible to run a synchronous service by periodic allocation

Slotted Systems

- since access is asynchronous, requires addressing information inside
- fine, fixed grain of multiplexing, easier to make guarantees about delay in getting to channel
- examples: Cambridge rings, DQDB, ATM
- slots sometimes called *cells*
- inbetween circuits and variable sized packets

Shared Media Multiplexing

Shared Media Multiplexing in Local Area Networks

- distances are short
- transmission costs are low
- "customer" and provider are usually same organisation
- communications not tariffed by use, expense is primarily attachment cost and administration
- usually no roads to dig up
- "customers" have a co-operative relationship (aided by system administrators ...)

Shared versus Non Shared Media

- "shared media" implies that the capacity of the network is fixed and is shared (in some fashion) by the attached devices.
- as network grows, the capacity remains fixed (links may be added, but all traffic traverses all links)
- "shared media" may mean a shared physical channel (as in Ethernet) or a sharing just above the physical level (eg token ring)
- shared media networks have simple if not trivial routing strategies
- topologies like buses, rings, some tree networks
- requires an element of trust, devices can interfere with communication of others
- failure of a link or the single shared physical link can cause complete network failure, often difficult to add in redundancy
- **simple** — most local area networks prior to 1990 built this way

Shared versus Non Shared Media cont.

- non shared — links are added as network grows; network capacity grows (like the telephone network)
- ability to damage other communication is limited
- topologies are more general, ie graphs
- routing decisions are harder
- possible to include redundant links
- **not so simple** but now the way most local area networks are done

Shared Media LANs: Ethernet

- Carrier Sense Multiple Access / Collision Detect (CSMA/CD)
- carrier sense — listen before talking
- collision detect — listen after talking to see if you're the only one
- collision window is twice the cable length
- retransmission strategy tries to lighten up when the network is busy
- more collisions, longer delay between retransmissions:
  - when collision is detected, jam the media to ensure everyone realises there has been a collision
  - backoff a random time (uniformly distributed to some max) before retransmission attempt
  - keep doubling the max until reaching a limit of retries
- Less common in copper, but wireless (802.11 WiFi) has become the new copper.

Ethernet Continued

- minimum ethernet packet size is the collision window length
- cable interconnected by repeater is still a single collision domain
- bridges and routers isolate collisions
- as speed and distance goes up, minimum packet size goes up accordingly
- efficiency is also related to collision window: when collision occurs a collision window's time is consumed. If this is comparable to average packet size then there is impact; If packets are much larger then impact is smaller
- throughput depends upon distribution of offerred load: a single device trying to send at 10 Mbps on a 10 Mbps Ethernet can; 100 devices trying to send 100 Kbps each can't.

- Ethernet strengths are simplicity, passive cable
- relies on relatively low loading (eg 20% is shaky)
- 10Mbps is the common one (and the IEEE 802.3 standard)
- shared physical channel is slightly problematic — everyone is trying to send on the same cable they are listening to
- no guarantee of when a station gets access

Token Ring

- stations arranged in a physical ring of unidirectional links. Each link is a point to point link
- multiplexing is based on owning the token
- frames are delimited by *code violations*: patterns which violate the modulation scheme
- frame format is either

start delim	access control	end delim

  for a token or

start delim	access control	FC DA SA INFO FCS	end delim

  for a frame with some information in it.

  FC = frame control (boring)

  DA,SA addresses

  FCS covers from FC to INFO

  FS = frame status — receiver can mark as having recognised address
- access control contains token information
- a station wishing to transmit must wait for a free token and change it into a frame
- frame may be bigger than entire ring, snakes around past destination and returns to sender who takes it off the ring and releases a new free token
- tokens can have priority which can be modified (even busy tokens) by stations having higher priority information

Token Ring cont'd

- no collisions, system throughput very high almost irrespective of demand distribution (in fact better when everyone wants to send)
- latency guarantee (in the non priority case) is max packet size time number of stations (ie big)

Token Management

- monitor station required to ensure one and only one token exists
- each station is capable of being a monitor station
- monitor sends out special tokens occasionally saying "I am the monitor"
- if a station doesn't see one of these it enters a "claim monitor" state
- stations contend to be monitors, highest address wins

- first token rings developed in the 60's
- 802 token ring based on IBM design of 1980 (4 Mbps)
- FDDI token ring at 100 Mbps

Slotted Rings

- ring is divided into fixed sized slots
- each slot has a bit (like a token) which can be set to full or empty
- stations wishing to transmit wait for an empty slot and mark it full, placing addressing information and data into the slot
- a full slot rotates around the ring to destination where it is recognised and either marked empty or marked to indicate it has been received (or not)
- if not destination delete, then slot continues around to source which extracts it from ring

Slotted Rings cont.

- to prevent endlessly circulating full slots a monitor is used
  - when a slot is filled its $monitor$ bit is cleared
  - monitor sets monitor bits in all full slots
  - if it sees a full slot with a monitor bit set it sets the slot to empty
  - similar problems to token ring in terms of who is monitor
  - maintaining slot structure is arguably easier than token management

- latency guarantees are relatively good (ie if source delete and always pass empty slot on) because units are fixed size
- small items slip in the middle of big items (fine grain sharing)

- Cambridge ring
- Cambridge Fast Ring
- Cambridge Backbone ring
- Orwell ring (destination delete) (British Telecom)
- no huge commercial successes (just ahead of their time)
- also, like any new system not quite as easy to set up (as Ethernet)

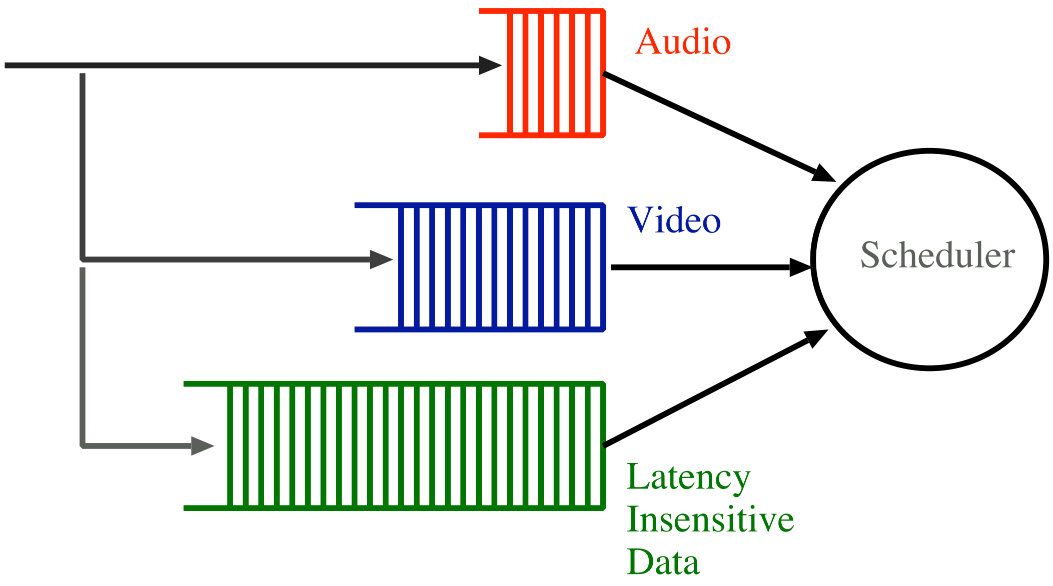
Shared Media Conclusions

- dominant technologies are ethernet (and variants)
- comparison of performance in Stallings, but misses the important point: these things rely on low utilisation
- token bus for the brave

- 100 Mbps, 1, 10 Gbps Ethernet (but not really used as shared media)
- non shared media local networks are now the norm
- Why did “Ethernet”, arguably the least appropriate shared media network for high speeds win?

Centralised Multiplexing

- When you have all the information in one place, should be able to do better
- In particular, no latency between transmissions.
- Priority versus share (general scheduling issue).



Scheduling Issues

- Priority is a good mechanism but often not a good policy
- "Fairness" is not a well defined concept.
- Allocating notional shares, eg weighted round robin is an alternative to strict priority
- So is *policing* within a priority class
- Care if layered multiplexing!

NonOrthogonal Multiplexing

- FDM, TDM are orthogonal in the sense that any given total channel signal can be divided into its basis function components uniquely.
- possible to find functions which are nearly orthogonal
- with too many get interference
- with a few get a little bit of interference, run error correcting code on top to compensate
- reduce need for co-ordination with other transmitters as long as transmitter and receiver know what is going on
- eg frequency hopping spread spectrum
- eg direct sequence spread spectrum (Code Division Multiplexing, Code Division Multiple Access)

CDMA

- Each channel is assigned a code which is a pseudo random sequence.
- The code is cycled through at a high rate, typically once through the entire code sequence for each data bit we wish to transfer. We sometimes call each bit of the code a *chip* to distinguish them from the data bits.
- transmitter exclusive ORs the chips with the databits and transmits the produced transmit chip sequence.
- receiver exclusive ORs same pseudo random sequence with the received chips and looks for big correlations to obtain chip sync.
- some chips get clobbered in the channel but statistically majority voting works

CDMA for mobile telephony

- Codes do not need to change during handoff. Could imagine very long lived codes.
- Silence suppressed voice transmission gives immediate benefits. (Your transmission is noise to everyone else.) Claim is a voice channel is only 35% active. (But need to be careful about code sync — don't want to chop-off the front of every word)
- Soft capacity, can increase capacity if willing to degrade the component channels.
- Single frequency channel (all processing is digital)
- Some security (need to know the code)

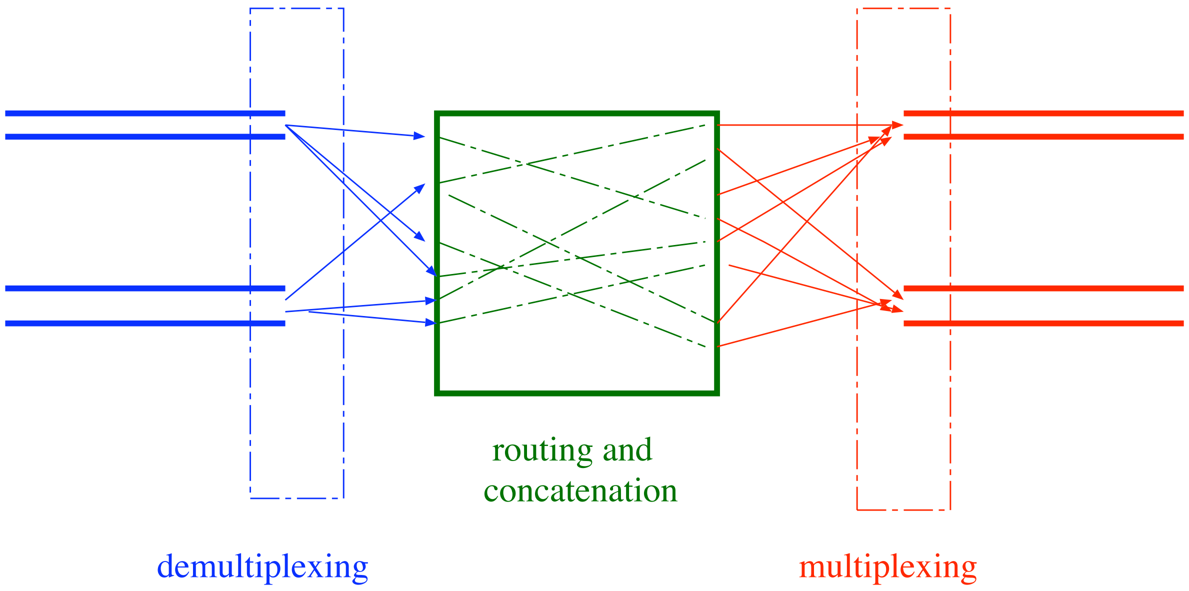
Multiplexing and Channel Characteristics

- synchronous multiplexing gives fixed delay, fixed capacity circuits
- asynchronous multiplexing doesn't
- the contention policy in asynchronous multiplexing directly impacts the channel characteristic: how much variability in capacity, delay
- relation to scheduling (packets run to completion, slots time sliced, synch strict periodic scheduling)
- think of requirements for traditional data, continuous media and interactive continuous media
- distinguish latency from variance in latency

Layered Multiplexing

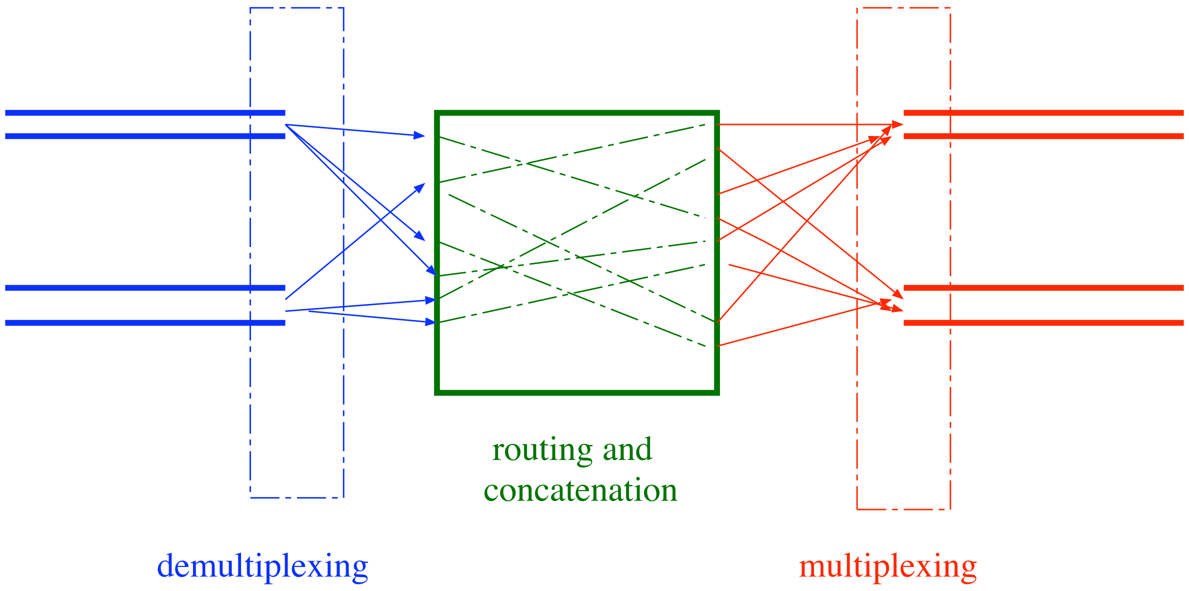
- asynch on top of sync
- sync on sync
- async on async
- eg an Ethernet interface may be only concerned with whether Ethernet packets are for it; not which process in the workstation they are for
- which process is done by a higher layer entity in the workstation
- each point of multiplexing is a point of contention, opportunity for more interference to creep in.

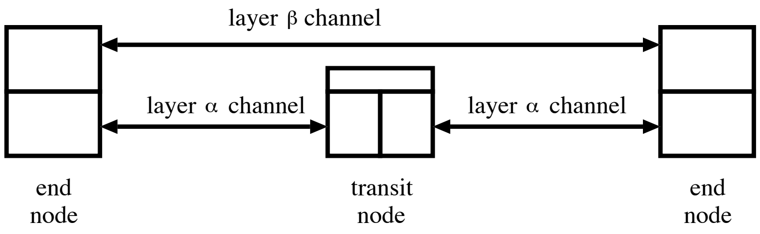
What is Switching?



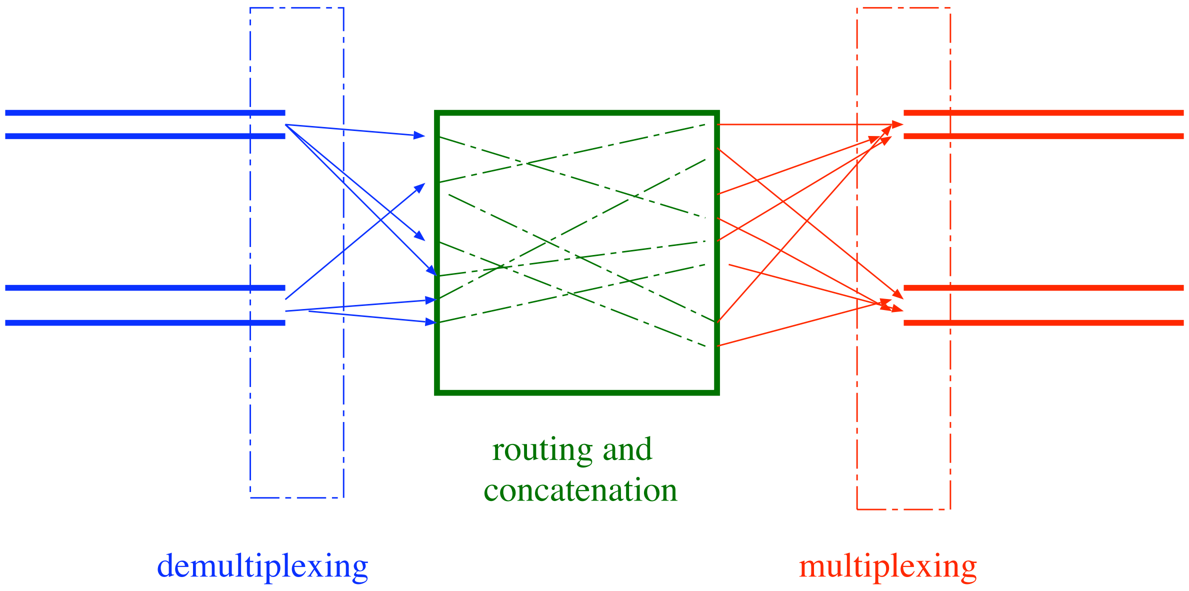
- need a way of recognizing the higher layer channels on the link
- need to figure out where they should go and get them there
- need to stuff them down links along with other higher layer channels.

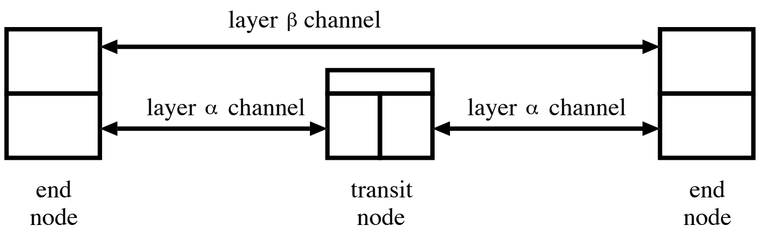
Circuit Switching



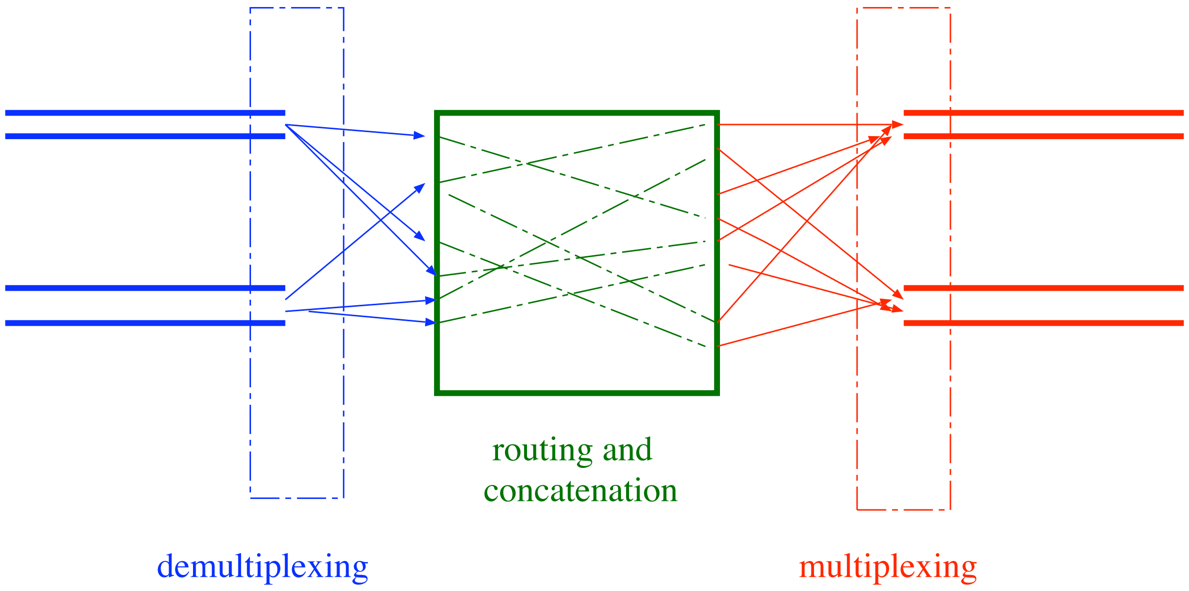


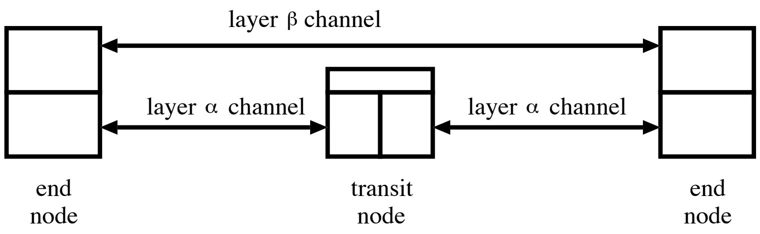
Packet Switching





WDM Switching



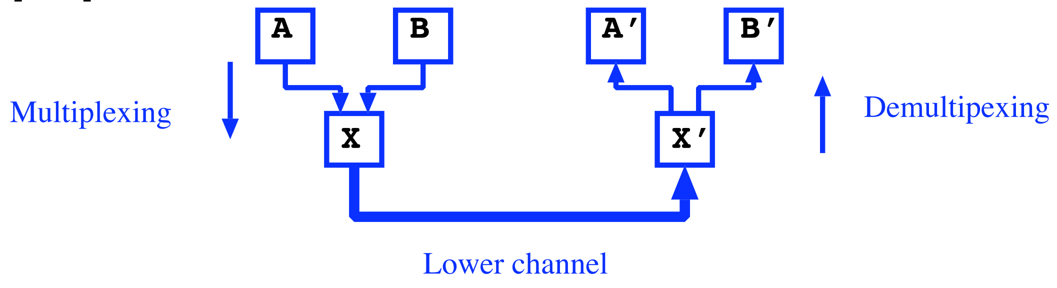


Notice a theme here?

These will last three pages will not make sense unless you came to lectures.

Exercises

- [6-1]



- [6-1] In the diagram above, how does the receiving entity **X'** distinguish data destined for **A'** from data destined for **B'**?
- [6-2] For the commercial television system we have examples of multiplexing in the spatial, temporal (time), and frequency domains. Briefly describe, using the television system as an example, what is meant by these three forms of multiplexing.
- [6-3] Why is synchronous time-division multiplexing commonly used in telephone systems?
- [6-4] Synchronous Time Division is popular for the deployment of fixed bandwidth services called circuits, sometimes also called virtual circuits. A virtual circuit provides an end user with the equivalent of a fixed dedicated link between two points. What properties make such a service possible?
- [6-5] Asynchronous Time Division requires the use of labels, why doesn't Synchronous Time Division?
- [6-6] Often Asynchronous Time Division limits the length

of packets — no one has an infinite amount of buffer space for the incoming packet — despite this, starvation is still possible. Explain why this may occur?

- [6-7] Random Access mechanisms have been common for controlling (or not controlling) access for Asynchronous Time Division systems. List properties that would make Random Access more-fair.
- [6-8] Describe one condition under which the performance of a well-behaved Random Access mechanism will degrade.
- [6-9] Describe what is meant by CSMA; how does the CD of CSMA/CD improve things?
- [6-10] Old-style (10Mbps) Ethernet is an example of a Shared-Media LAN, what is another example?
- [6-11] New-style (10Gbps) Ethernet is an example of a non-Shared Media LAN. Compare Shared and non-Shared LANs, noting what it is thats shared, drawing a typical topology for each and point out where a critical equipment failure might occur.
- [6-12] Centralizing Multiplexing is a great way of controlling/sharing access to a common resource — such as a shared media. Considering the physical properties of WiFi, why would this idea degrade rather than improve performance?

Topic 7: Protocol/State

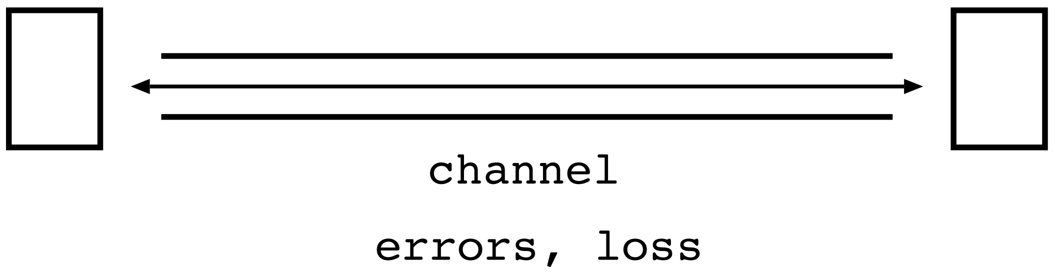
Previous topic

- Multiplexing
- Common-Place Example
- Time Division Multiplexing
- Local Area Network Example
- Shared vs. Non Shared Media

Overview of this topic

- Error Control Protocol
- Performance
- State Diagram
- Continuous ARQ
- Flow Control

Protocols and State



- a *protocol* is set of procedures and formats that entities use to communicate information.
- as always, different people mean different things when they use the term: is a coding scheme a protocol?
- here we are particularly concerned with protocols involving state changes within entities: indeed often implemented and or specified as finite state machines
- communication protocols of this type are difficult because one end only has an approximate model of the state of the other end (errors, loss from contention, delay variation)
- example of allies separated by mountains full of bandits wishing to co-ordinate attack; never sure if messenger gets through.
- important example is an error control protocol (we'll talk about flow control too)

Error Control Protocols

- strategy is to use an error detecting (rather than correcting) code and then retransmit when errors are detected or data never received
- it is the receiver that knows whether an error has been found; it must request retransmission of data found to be in error
- called Automatic Repeat reQuest (ARQ)
- means delay in getting information through is variable even if underlying channel is fixed delay
- see Halsall Chapter 4
- consider information transfer in one direction for clarity (although messages going in both directions)

How it works: an example

- information is divided into frames
- each frame also contains a sequence number which ranges from $0 \ldots N-1$, a field indicating that it is a DATA frame and a CRC:

DATA	seq no	information	CRC

- when receiver sees a data frame with a correct CRC and with a sequence number is was expecting, it sends an *acknowledgement frame* with the sequence number of the next frame it is expecting:

ACK	seq no	CRC

- this allows transmitter to send next frame and to forget the previous frame
- if it receives incorrect CRC or sequence number beyond what it was expecting it can send a NACK, saying which sequence number it is expecting:

NACK	seq no	CRC

- what should receiver do if it receives a sequence number for a frame it has already acknowledged?
- what should transmitter do if it never receives an ACK?
- note sequence number eventually wraps round

Performance of such a protocol

- frame size p bits, channel capacity b bits per second, channel delay d secs
- time to transmit frame is $\tau_{tx} = \frac{p}{b}$
- time to get frame down channel is d
- assuming acks and nacks are small and that no errors are found, time from beginning transmission of one data frame to beginning of next is $2 \times \tau_d + \frac{p}{b}$
- if τ_d is small, then throughput $\longrightarrow b$, if τ_d is large, throughput approaches $\frac{p}{2\tau_d}$

propagation 2×10^8 m/s, $p = 1000$ bits

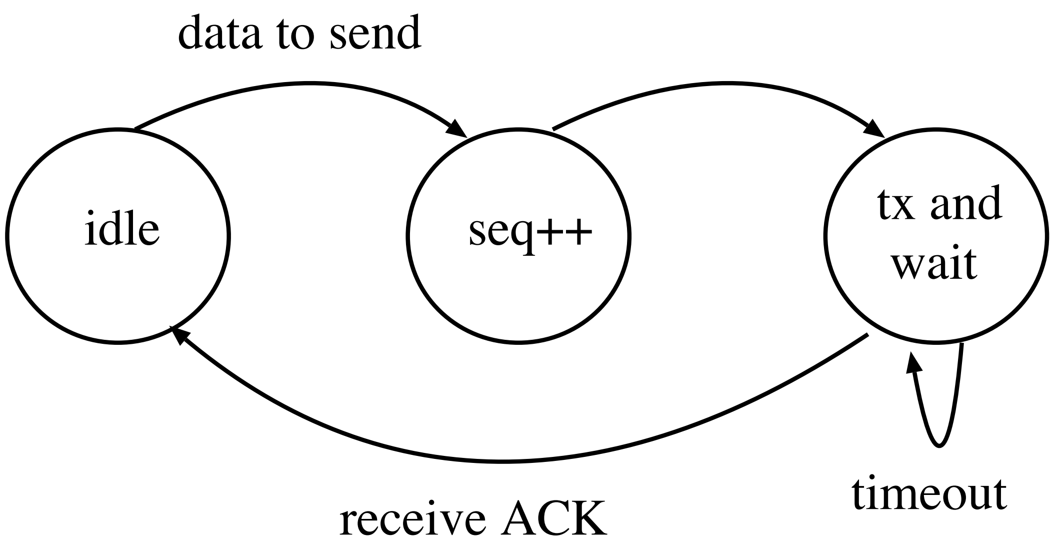
1 Kbps channel

distance	τ_d	tau_{tx}	throughput
1 km	50 μs	1 s	1 Kbps
200 km	1 ms	1 s	1 Kbps
5000 km	250 ms	1 s	670 bps

1 Mbps channel

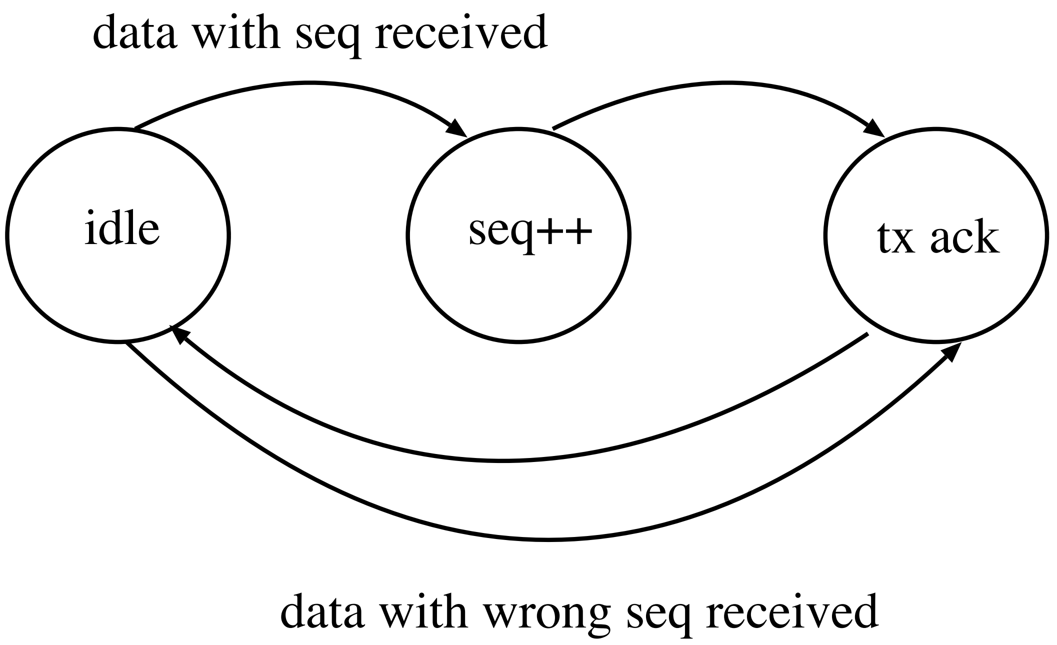
distance	τ_d	tau_{tx}	throughput
1 km	50 μs	1 ms	1 Mbps
200 km	1 ms	1 ms	333 Kbps
5000 km	250 ms	1 ms	2 Kbps

State Diagram



Transmitter

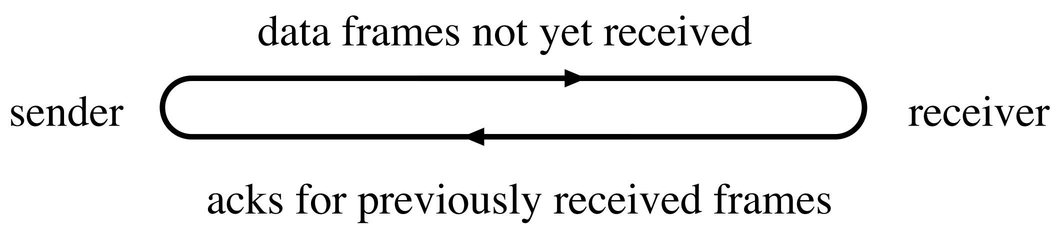
- transmitter must run time out and retransmit (why?)
- what makes a good time out period?
- ack must have correct sequence number
- note seq no is part of state


data with seq received
idle
seq++
tx ack
data with wrong seq received

Receiver

Continuous ARQ

- obvious improvement is to have multiple frames in transit at once with acknowledgements flowing back continuously
- think of a traction tread:


data frames not yet received
sender
receiver
acks for previously received frames

- when everything is going like clockwork a received ack causes a new data frame and a new received data frame causes a new ack
- the *window* is the number of frames (or bytes) that can be in transit unacknowledged (previous example was window of 1 frame)
- if window is big enough then can keep link full
- better than just big frames — why?
- different strategies when receiver detects a missing frame:
  1. *go back* N:transmitter goes back to missing frame and restarts from there possibly repeating correctly received frames
  2. *selective retransmission*: transmitter only repeats missing frame, receiver acknowledges all correctly received frames
- one simple, the other one more efficient, which is better rather depends on how often you expect to drop a frame

Flow Control

- flow control is a general problem
- want to ensure that receiver can take in information at rate transmitter is sending
- in circuit switching rate is the channel rate, everyone agrees
- packet switching allows rate *adaption*: transmitters don't have to be at the same bitrate ...
- ... but must be a mechanism for balancing long term information rates
- example look on person's face telling you they don't understand
- example, X-on, X-off flow controls:

  "go until I tell you to stop"

  receiver needs to be able to receive 2 channel delays worth of information after it issues the stop — why? $\longrightarrow$ what happens when channel delay is variable?
- window flow control — similar to our error control protocol, but performing a different function

Sliding Window

Sliding Window Error and Flow Control Protocol

- receiver tells transmitter what frames it has received and how far ahead it is willing to receive
- receiver effectively opens and closes the window
- receiver does this on the basis of changing availability of resources, in particular buffers

ACK	rx seq no	win size	CRC

- transmitter's model of receiver is now extended to resources (but is still potentially out of date)

Other Generic Sorts of Protocol

- media access
- remote procedure call
- commitment concurrency and recovery protocols (not just communications!)
- Byzantine generals

Conclusions

- entity state in communication protocols try to model other entity (and indeed the channel)
- this model is not always correct
- must cope with loss (and indeed disappearance of other entities)
- time outs to cope with loss; means of poking one's state machine revising the model of the channel / other end

Exercises

7-1 Identify three examples of ARQ (Automatic Repeat reQuest) you perform in everyday life (no computers)

7-2 Contrast Flow Control and end-to-end ARQ; consider particularly what are the objectives of each, what are the requirements on the channel and what are the limitations on distance between the ends of the control.
How much data is outstanding?
How quick is recovery?
The two approaches are interchangeable but each has disadvantages/advantages - give an example where each is appropriate.

7-3 Continuous ARQ can use a window: the number of frames waiting for acknowledgment. Why is it better to use many small packets and a large window size than just increase the size of the packet until it *fills* the link?

7-4 Consider packet loss in an ARQ link. Compare the number of packets retransmitted if we use a basic ARQ, continuous ARQ with *go-back* n, and continuous ARQ with *selective transmission*

7-5 Compare the end-system overheads (what resources are required) for each of the 3 ARQ schemes of the previous question.

Topic 8: Naming Addressing Routing

Previous topic

- Error Control Protocol
- Performance
- State Diagram
- Continuous ARQ
- Flow Control

Overview of this topic

- Hierarchical vs. Non-hierarchical
- Class-based/classless routing
- Routing
- ARP
- Interconnect/Layers/Routing
- When do we route?
- Routing examples

Naming, Addressing and Routing

One view:

- a name denotes something
- an address denotes where something is
- a route tells you how to get there

Example:

- name: Andrew Moore
- address: WGB FW16
- route: up one floor, go north then through the door turn right through next door...

Another view

These are all just names (or identifiers)

- “names” identify things
- “addresses” are identifiers of attachment points (to networks)
- “routes” are identifiers of paths

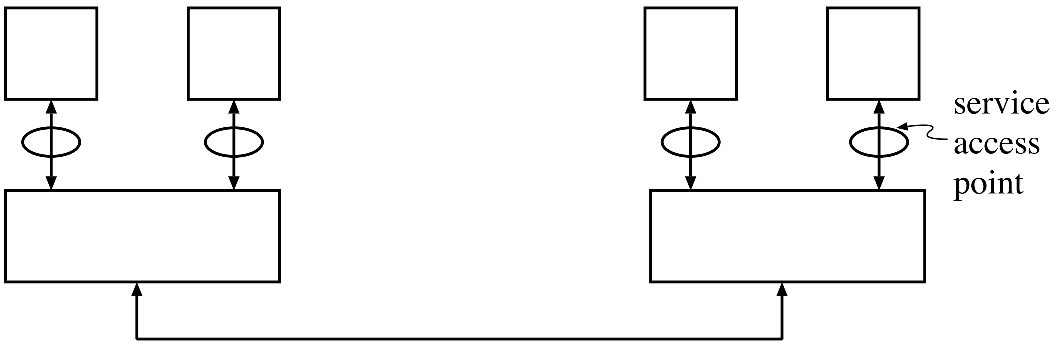
Issue is one of *binding*. So its just like operating systems...

Name resolution, name lookup: binding a name to an address

Routing: binding an address to a route

Addresses at multiple levels

- eg machine with name "frank.dcs.qmul.ac.uk" has
  138.37.88.242 for an IP address
  00:0E:0C:5A:E6:9C for an "ethernet" address



- *service access point* is the point of attachment between an upper layer and lower layer entity.
- can view addresses as denoting service access points
- service access points can be long lived or shortlived, eg a session or a transaction

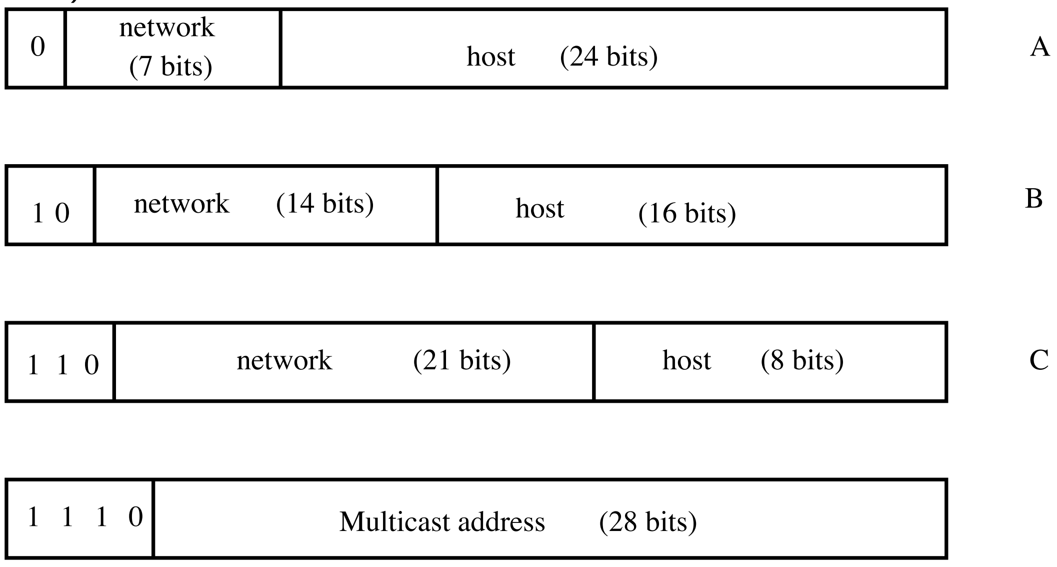
Hierarchical *versus* Flat Address Space

- flat address has no structure
- only function as unique identifiers
- for example Ethernet 48 bit addresses
- can be moved around without modification
- logically centrally allocated

- hierarchical address spaces are divided up, usually to aid the routing process
- our intuitive notion of address tends to be hierarchical

country	network	host

- IP (Internet Protocol) addresses are hierarchical four byte addresses with variable sized boundaries. Old style ("Class based"):



- IP addresses usually quoted as decimal-represented-bytes separated by "."
- eg we have a class B network which is 10.37 (with top bit set becomes 138.37)
- Class A networks $\longrightarrow$ who got them?
- Classless routing allows divisions on any boundary (how??)
- a network is free to put structure on the host field (and there are standard things people do)

Class Based Routing

Class-Based is sometimes referred to as classful.

```
multicast = false;
if ad[31] = 0
  then network = ad[30:24]
elseif ad[30] = 0
  then network = ad[29:16]
elseif ad[29] = 0
  then network = ad[28:8]
else multicast = true;
```

Class A networks are ridiculously large, class C ridiculously small, boundary inflexible, leads to unused address space.

Classless Routing

Classless is sometimes referred to as CIDR (pronounced "cider"): Classless Inter-Domain Routing.

- Store address space boundaries (network and mask) in all routers.
- Not a big deal because had to store next hop for all networks, just adding a mask.
- Allows opportunity for aggregation: compaction of routing tables.
- Incentives for logical network number assignment

Simple Example

Consider two networks 239.230.101.0 and 239.230.100.0.

Note that 239 is 11101111, so these would be class C old style.

New style, these would be 239.230.101.0/24 and 239.230.100.0/24. (24 describes the mask which is 24 ones followed by 8 zeros, sometimes written as 255.255.255.0)

These networks could be aggragated as 239.230.100/23 either because they were the same network or because the routes to them were in the same direction.

Even more flexibility by “longest matching prefix”.

Routing

- act of binding addresses to path
- can view as binding higher layer channels to lower layer channels (recall lower layer channel may not go all the way)

- routing may be
  - static (ie fixed tables)
  - dynamic with network topology
  - dynamic with network condition (adaptive)
- it may be performed
  - centrally
  - fully distributed
- simple example is binding an IP address to an Ethernet address using the Address Resolution Protocol (ARP)

Address Resolution Protocol

- purpose is to convert (protocol, address) into lower layer address
- for illustration, protocol = IP, lower layer is Ethernet
- A knows B's IP address but needs to know B's Ethernet address in order to communicate
- A sends out, using the Ethernet broadcast address an ARP Request packet containing:
  - tag saying this is an ARP Request
  - tag indicating IP is the protocol
  - A's IP address
  - A's Ethernet address
  - B's IP address

- If B receives this packet and recognises its IP address it updates its tables about A and transmits a reply directly to A, using A's Ethernet address rather than the Ethernet broadcast address:
  - tag saying this is an ARP Reply
  - tag indicating IP is the protocol
  - B's IP address
  - B's Ethernet address
  - A's IP address
  - A's Ethernet address

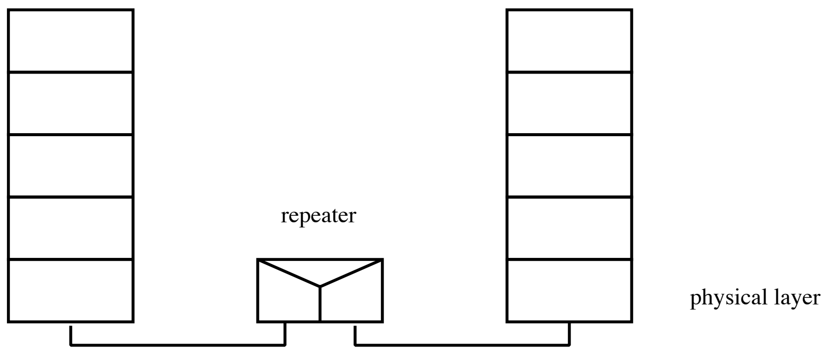
Address Resolution Protocol cont.

- other people seeing ARP Requests $update$ their tables (but generally don't create new entries)
- nice property that rebooting host sending out a single ARP request will update all tables knowing about that host

Interconnection Layers and Routing

Repeater, e.g., media converter

- repeater, just regenerates signal, “physical layer” interconnection, everything goes everywhere

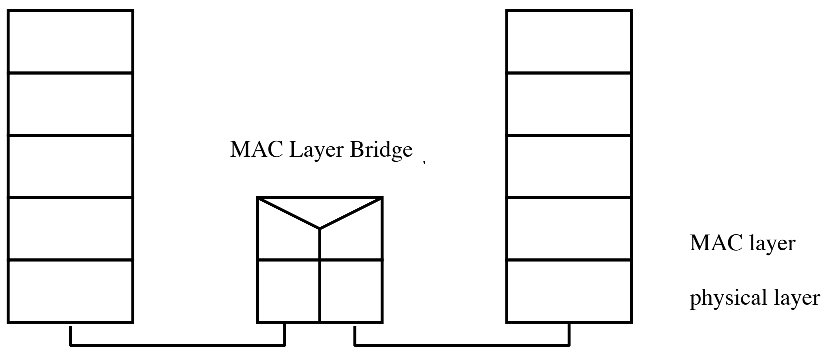


Interconnection Layers and Routing (cont'd)

Bridge, e.g., ethernet switch

- bridge, has a set of addresses (eg Ethernet 48 bit addresses) that it is willing to forward packets for "media access control" interconnection

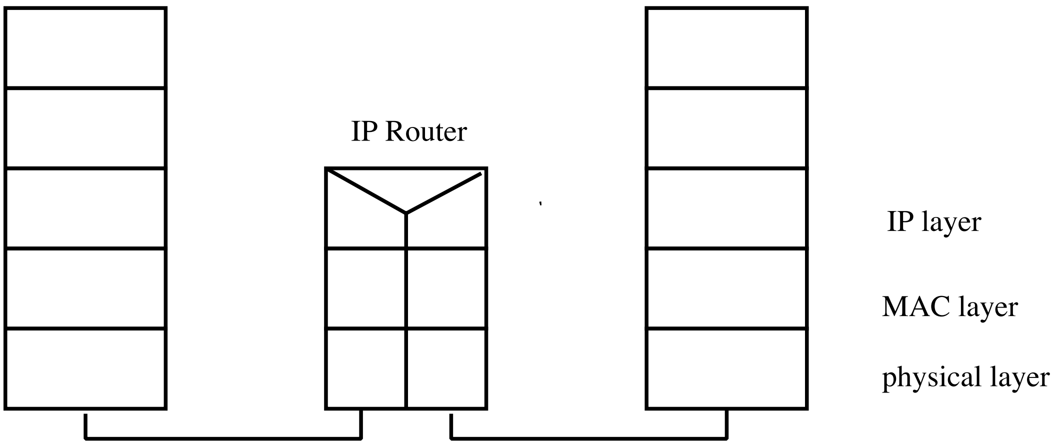
  don't have to be the same media access controls on both sides, just the similar concept of send this packet with this 48 bit address in it
- simple things with potentially big tables



Interconnection Layers and Routing (cont'd)

IP Router, e.g., your home firewall, more likely - big boxes you never see that make the Internet go

- router, knows about structured network addresses and exploit structure to perform routing. Eg IP routers know about IP addresses
- need to understand network layer protocol
- may be several network layer protocols being used $\longrightarrow$ multiprotocol routers
- may bridge when network layer protocol not understood



Routing in General Topology Networks

- network is represented by a graph with nodes and edges, edges have some cost associated with them. This cost may vary in time.

- *flood routing*: incoming packets are repeated on all links except the one on which the packet arrived. Packets contain a hop count which is decremented at each node. A packet with a hop count of zero is not repeated.
  - if initial hop count is high enough, all non looping paths will be tried (along with a few looping ones) $\longrightarrow$ robust
  - requires no knowledge of network topology
  - the shortest path will have been tried
  - burns network capacity

Routing (cont'd)

- *random routing*: an incoming packet is sent out on some randomly chosen link (unless it is addressed for that node).

- shortest path routing tries to find the path between two given nodes with the least cost, eg Dijkstra's algorithm
- however, each node must build up a picture of the network by exchanging information with other nodes, network topology may be changing as links or nodes fail $\longrightarrow$ inconsistent view of network
- inconsistencies may cause loops, as in A thinks the way to C is via B and B things the way to C is via A
- good analysis of distributed shortest path routing algorithms in Schwartz
- routes may also be constrained, eg don't go through the following nodes..

- alternative to having the network decide how to route packets is to have the sender decide (*source routing*):
  - route is embedded in packet, just a list of hops along the way
  - clearly can be made loop free
  - source needs to know network topology
  - source needs to know about changing conditions
  - loose source route is a list of some hops along the way

When do we route?

- on every packet
  - *datagrams* eg IP
  - robust to network failure
  - adapt to network loading
  - packets can arrive out of order
- when a channel is established (channel is established in case above, it just that the intervening nodes haven't been told about it)
  - routing is done once
  - further packets follow same route
  - packets must be tagged with something that indicates what route to follow (index into a table)
  - packets arrive in order
  - because network nodes know about channel, resources can be allocated to it, ie buffers
  - *virtual circuits*; tags in packets are *virtual channel identifiers* which conform to our notion of address and they clearly denote service access points
- datagram = connectionless
- virtual circuits = connection oriented
- *virtual* in the sense that not fixed delay or fixed capacity
- religious wars about state

Routing and resource

- adaptive routing takes into account link utilisation (on every packet or just when connection established, or something inbetween?)
- hot potato routing avoids contention on outgoing links (no need for buffering if all traffic is point to point and links all same capacity)

Exercises

- [8-1] Describe in terms of Names, Addresses and Routing, the *binding* processes as you look up a friends telephone number, select the part of the phone number that needs to be dialed, dial, and are connected to your friend.
- [8-2] Consider accessing a web page on `www.acm.org`. Describe the address used at each layer from ethernet up to the location by the web-server on the machine. Consider the IP layer, TCP, the possibility the web server is actually made up of several machines (www.acm.org is actually served by a cluster of machines) and so on.
- [8-3] Give two examples of hierarchical addresses (not from computing)
- [8-4] Give two examples of non-hierarchical addresses (not from computing)
- [8-5] Give two examples of partly-hierarchical addresses (not from computing)
- [8-6] Compute the number of A, B and C class addresses in the Internet; who have the (old) A-class address spaces??
- [8-7] Consider the design of a trivial routing algorithm that needs to share information among a small number of routers on the network. What is the quantity of data you will need to transfer between each router to keep tables up to date. Consider the network takes a given time t for each link to transfer data, estimate the propagation delay of routing updates across a network of routers. (The topology

is arbitrary, however if you are not sure how to begin, consider a line of routers and a ring of routers). Draw some conclusions on how intermediate summaries of change may affect things.
Now consider what did you assume about the routing functions?
Did all routers know about all destinations? (Is that necessary?)
How might class-based routing assist you?

Topic 9: The Internet

Previous topic

- Hierarchical vs. Non-hierarchical
- Class-based/classless routing
- Routing
- ARP
- Interconnect/Layers/Routing
- When do we route?
- Routing examples

Overview of this topic

- Internet Philosophy
- IP over Ethernet - an example
- Internetworking - the inter of internet
- IP Addresses and IP Routing
- Datagrams and Connections
- Next?

The Internet

Details in "Internetworking with TCP/IP Vol 1" Douglas Comer (Prentice Hall)

Gory details in "Internetworking with TCP/IP Vol 2"

> *"Any large functioning system should obey the following 3 principles:*
>
> - *it should be well specified*
> - *it should be well understood who is in charge of it*
> - *it should be well understood who pays for it*
>
> *The Internet violates all of these."*

Danny Cohen(?)

Philosophy I — The Problem

Context of the late 1970's:

- most data networks were wide area
- local area networks just emerging
- ARPANET
- manufacturer specific protocols
  - DECNET
  - SNA
- X.25 (public network) emerging

Conventional wisdom was one built *gateways* which translated protocols (formats and procedures) from one network to another. Eg file transfer gateways, terminal gateways, mail gateways ...

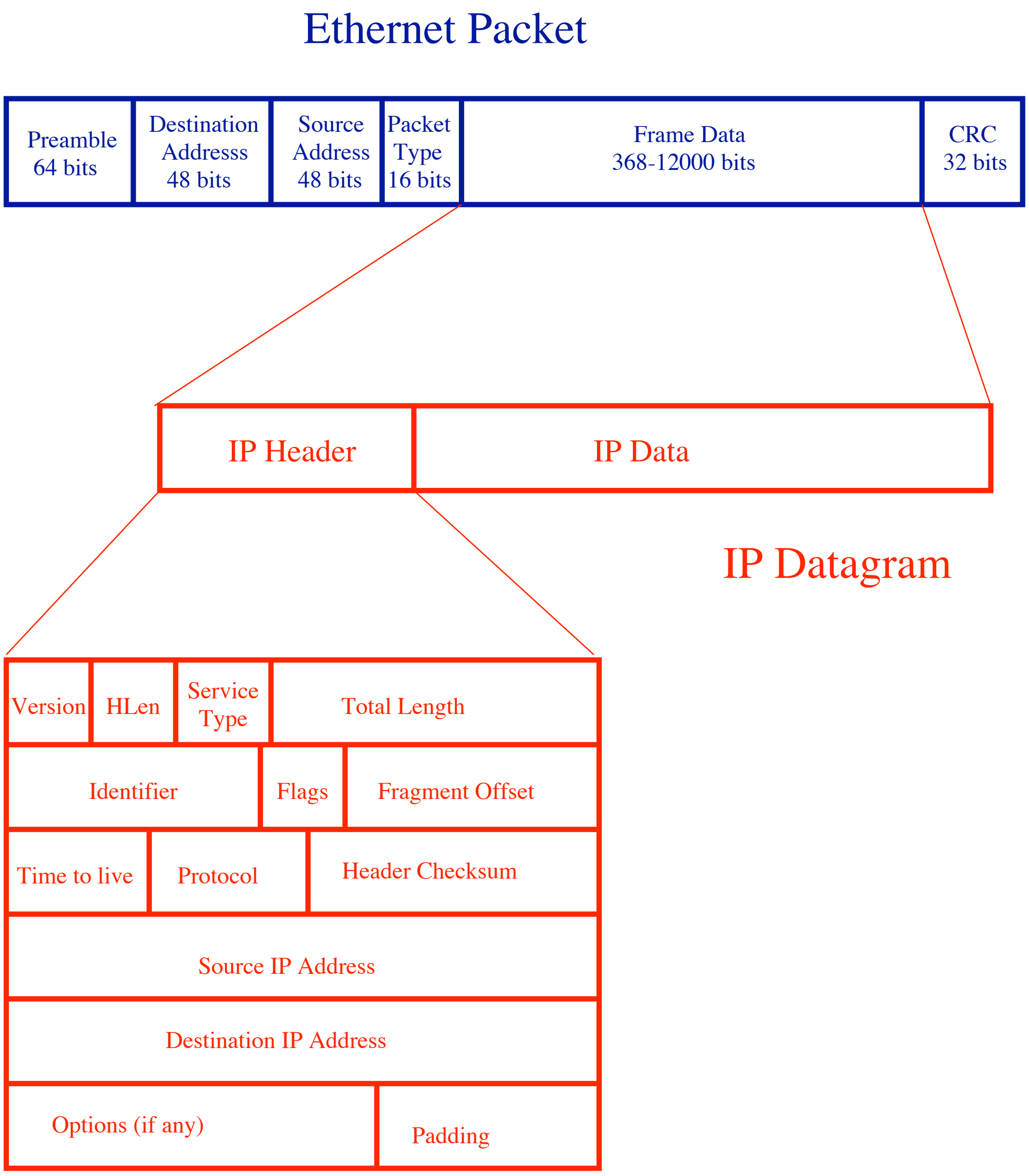
Unfortunately this limits application development. Why?

Philosophy II — The Solution

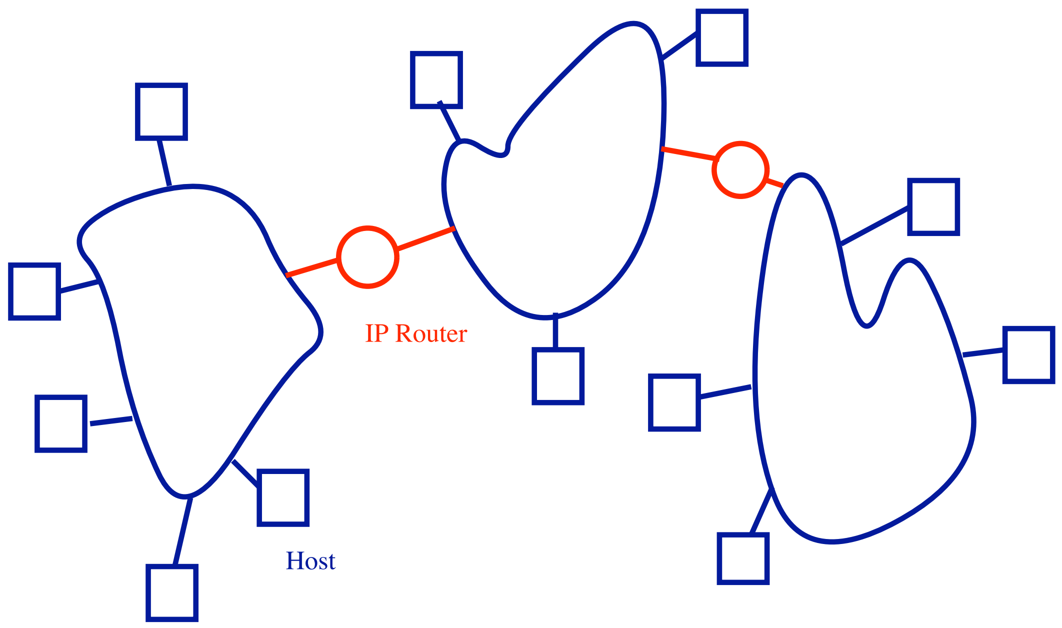
- use new protocols in all computers
  - slow process but investment not huge in 1978!
  - give source and implementations away
  - largely research based, Unix based

- but can't replace the networks
  - define a lowest common denominator communication service which can be made to work on any network
  - Internet Protocol (IP) service is the delivery of *Internet datagrams*
  - gateways now perform a simple task and are usually just called *routers*
  - give source and implementations away

IP Over Ethernet



Internetworking



IP Router's job is to forward IP packets towards the destination by sending to the next hop which is on one of the networks to which it is attached.

- select which network to use
- selects which lower level address (eg Ethernet) to send to
- next hop may be the end or another router
- uses the IP destination address to determine this

IP Addresses and IP Routing

- address has two parts: network address and host address.
- router examines network address to see if it is one of the networks to which it is attached. If so router sends to the host on that network using the protocol for that network
- if not, router must decide which router to send it to
- essentially it has a table, called a routing table, which contains entries of the form
  ($\mathit{IP\ network\ address,\ next\ router\ to\ use}$)
  where the next router to use is on a network attached to the current router
- can also have the notion of *default router* so that if you don't have a table entry for a particular IP network address you send to the default router which handles it.

Routing Tables

- originally, some routing tables were maintained “by hand”, ie they were in a file that was editted by the local system manager

- now more dynamic, with provision for core system routers (which know how to get everywhere) and autonomous subsystems which can control the routes into themselves

- adaption to link failure, and link load by routers exchanging information (Gateway-gateway protocols)

- arguably more work on routing done in the Internet than in any other network

Connectionless versus Connection-Oriented

(déjà vu again again)

Note:

$$\text{datagram} = \text{connectionless}$$

$$\text{virtual circuit} = \text{connection oriented}$$

Question:

What *state* is in an IP router?

Answer:

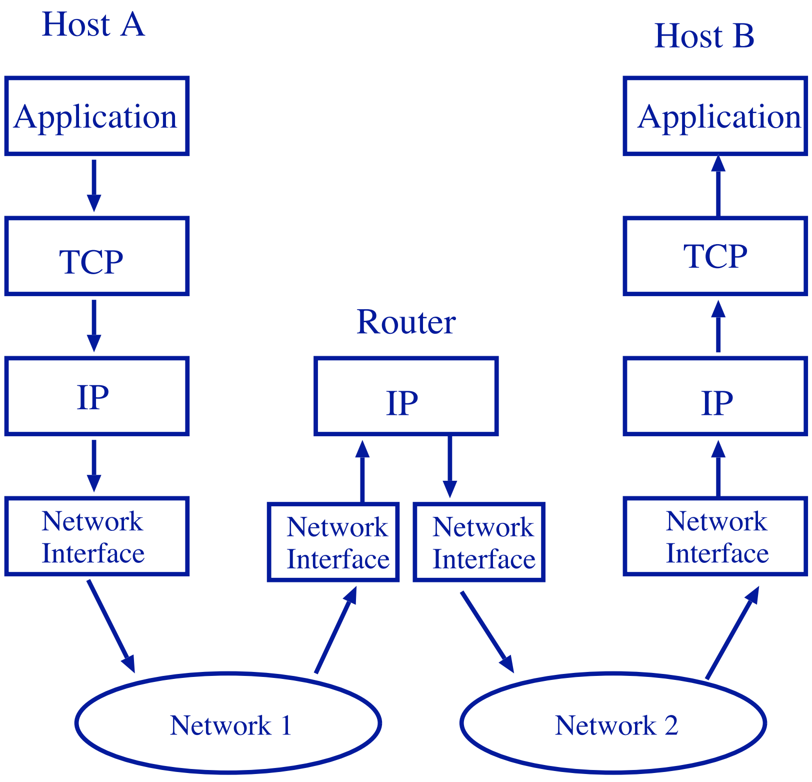
Routers know (something) about the state of the network.
They know nothing about who is communicating with whom.

A *datagram* is a packet that can be routed to its destination solely on the basis of its contents.

Transmission Control Protocol (TCP)

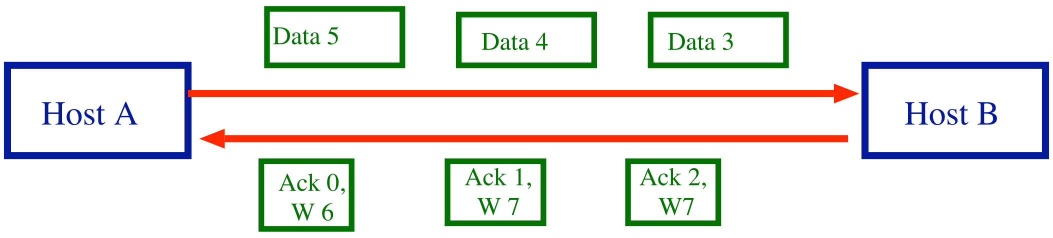
Context

- IP delivers packets between hosts across the globe
- routers know nothing about connections
- routers and links can become congested and lose packets
- bit errors can occur
- need something to provide *reliable* end to end delivery of packets



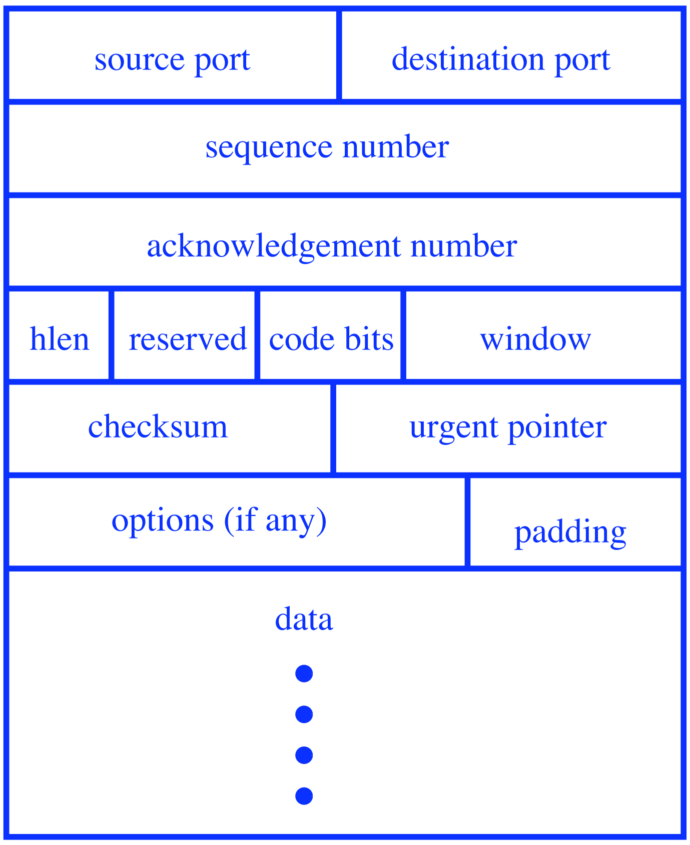
TCP Properties

1. *stream* oriented: a sequence of bytes
2. *connection* oriented: connection establishment (but the network doesn't know!)
3. *reliable*: recovers from errors or loss of packets by detecting loss / error (how?) and retransmitting
4. *full duplex*: can send information in both directions
5. *flow controlled*: one end can tell the other end how far ahead it can transmit (the *window*)



- control and data information
- received data packets are acknowledged
- acknowledgement also indicates window size
- large window keeps network full of packets, higher throughput

A bit of detail



- data, acks, window all *piggybacked*
- source and destination ports identify "application programs"
- sequence number, acknowledgement in bytes
- window is bytes from current ack

TCP: Back to modelling state....

- window size is a model of receiver resources

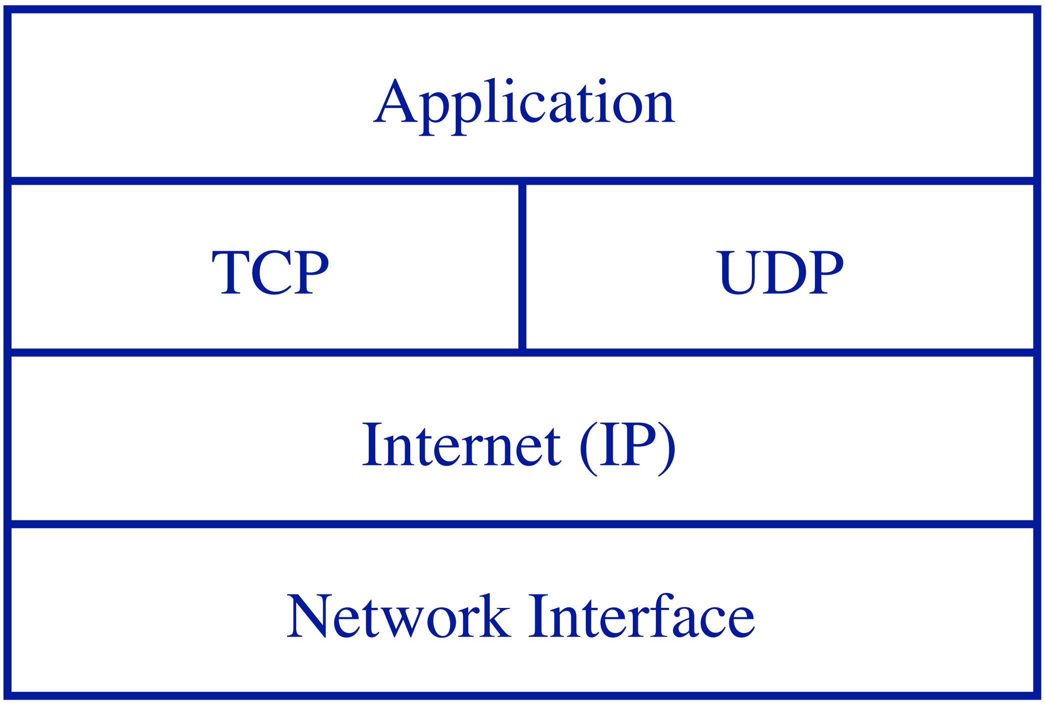
- acks provide a model of what receiver has received

- but need to model the channel as well:
  - round trip time (why?)
  - variance in round trip time (why?)
  - capacity of the channel (why?)

- takes time to build up this model and only way in the vast majority of implementations can determine capacity is through loss.

User Datagram Protocol (UDP)

At same layer as TCP but provides a datagram service only.



- UDP also has port numbers
- used for simple queries, status
- used for SUN Remote Procedure Call (RPC)

Application Protocols

Protocols using TCP:

- http
- telnet, rlogin, X
- file transfer protocol (ftp)
- network news transfer protocol (nntp)
- simple mail transfer protocol (smtp)

Protocols using UDP:

- echo
- time
- domain name server
- simple network management protocol (snmp)
- network file system (nfs)

Evolution and Conclusions

- now clear dominant applications
- firewalls to prevent(?) unauthorised use, applications
- firewalls are of course application level gateways....
- application innovation only over http???

Why Success?

- independence of vendors?
- designed and driven by the users?
- evolution based on experience?
- openness?
- lowest common denominator makes it hard to move away from?
- network had no view of $services$? (but now caching, Akamai...)

Exercises

9-1 Illustrate the following two examples of encapsulation as used in the communications stack for: web traffic on TCP/IP transported using Ethernet on wireless, and the NFS application over UDP/IP transported using Ethernet on unshielded twisted pair (UTP) cable

9-2 A router separates two Ethernet networks.
What two pieces of information are required by a computer to communicate with another connected on the same Ethernet network?
What additional information is required by a computer on one Ethernet network to communicate with a computer on the other side of the router?

9-3 It is said that the TCP protocol models the state of the communications path between the two ends of the TCP connection.
Why does it model the round trip time?
Why does it model the variance in round trip time?
Why does it model the capacity of the channel?

9-4 How does TCP build this model? Why does TCP need to build a new model every time?

9-5 Why would an Internet Router participate in the window-based flow control such as TCP?
When would such participation be a disadvantage?

9-5 Why has the Internet Protocol(s) succeeded?

Topic 10: Standards

Previous topic

- Internet Philosophy
- IP over Ethernet - an example
- Internetworking - the inter of internet
- IP Addresses and IP Routing
- Datagrams and Connections
- Next?

Overview of this topic

- ITU
- IEEE
- ISO
- IETF

Standards

- not like a single manufacturer building a nut and bolt
- devices have to communicate with potentially lots of devices made by different people.
- difficult to standardise in light of experience (unlike nuts and bolts)
- note distinction between conformance and interoperability

Standards Bodies - ITU

- worldwide telecommunications: International Telecommunications Union (ITU), formerly known as CCITT
- part of the UN
- national representatives, but mainly telecommunication operators (eg BT, AT&T, ...) and manufacturers (eg Lucent, Alcatel, Marconi ...)
- different series of “Recommendations”
- V series about digital communication over the public voice network – modems!
- X series about packet communication over the digital public network
- G series about low level transmission, modulation, framing
- Q series about signalling

Standards Bodies - IEEE

- Institute of Electrical and Electronic Engineers
- standards for local area networks, the “802” committee
- eg 802.3 Ethernet, 802.5 token ring, 802.11 wavelan
- not all used
- 802.2 says what a frame is, talks about 48 bit addresses

Standards Bodies - ISO

- International Standards Organisation (also UN)
- Open Systems Interconnection (OSI) programme
- layering model and 7 layered model
- a lot of standards which aren't used, but some are
- often held up as a way not to create standards

Standards Bodies IETF

- Internet Engineering Task Force
- interesting study in sociology
- standards are issued as RFCs (Request For Comment), but now proceed through a drafting process
- original RFCs were actually requests for comment...

Exercises

10-1 What is the vested interest of each of the ITU, IEEE, ISO and IETF?

10-2 Why did standards lead to an ATM cell size of 48-octets?

10-3 What makes a *defacto* standard? (e.g., Microsoft Word, Adobe Acrobat, Berkeley Unix, . . .)