Part IB Revision Notes
Digital Communication I

Michael Smith

May 9, 2004

1 Basic Principles

- **Layering** is an abstraction used to divide system functions, resulting in a 'stack' such that we only have interaction between layers that are one level apart.
- An **entity** is something with an independant existence, which communicates with **peer entities** at the same level on the stack, but at a different place. Communication is carried out using the lower levels in the stack.
- A **channel** is that into which an entity puts symbols and which causes these symbols (or a reasonable approximation) to appear somewhere else at a later point in time. Its characteristics include symbol type, capacity, bandwidth, delay, fidelity, cost, reliability, security, order preservation and connectivity.
- **Embedding** is a form of layering such that higher layer information is embedded into lower layer information (e.g. a UDP datagram contained within an IP packet, contained within and ethernet packet).

2 Physical Transmission

- To get bits down a wire, we need to change an analog wire into a digital channel. The transmission is typically as an analog waveform, with a delay given by the speed of light (or sound) and the distance it must travel (constant delay for constant medium and distance).
- The **bandwidth** is the range of frequencies of a sinusoidal signal that are supported (transmitted) by the channel. This is a resource - if one person is using it, nobody else can. We can't directly transmit square waves, as they have high-frequency components in their Fourier series, which can't be transmitted.
- **Noise** in a channel can be systematic (which can in principle be eliminated) or random. The **signal to noise ration**, $\frac{S}{N}$, is the ratio of received signal power to received noise power. If B is the bandwidth, then the channel has an **information capacity** of $C = B\log_2(1 + \frac{S}{N})$ (in bits per second).
- **Attenuation** is the loss of energy in a signal as it travels along a channel. This may be caused by radiation loss (e.g. high frequency signals), absorption by the media, or spatial dispertion (for aerial signals), and is a function of channel length, transmission medium and signal frequency.

- For a digital channel, the **baud rate** is the rate at which symbols can be transmitted, and the **data rate** is the rate at which bits can be transmitted. Channel fidelity is measured by the bit error rate (probability that a sent bit is incorrectly interpreted).
- **Modulation** is the systematic alteration of a carrier waveform by an information signal (transforming the signal into a more appropriate form to be transmitted). **Baseband** is not modulated on a carrier, whereas **broadband** is (hence the bandwidth can be shared).
- **Asynchronous transmission** divides the transmission into frames. The receiver and transmitter agree on the clock frequency, such that the two clocks do not become out of sync over a frame. In the case of RS232, a start bit is used to start the frame (by dropping a high carrier), and the frame lasts a constant number of bits (8 data and 1 parity). In the case of SLIP, a datagram starts and ends with the `END` character `0xc0`, and cannot be larger than the specified MTU (maximum transmission unit). An `ESC` character is used to send a byte equal to `END` (sending `0xdb 0xdc` and `0xdb 0xdd for END` and `ESC` respectively).
- **Synchronous transmission** is continuous, and requires the receiver to adjust its clock frequency in line with the incoming signal. This is typically done using a phase-locked loop (a feedback circuit is used to compare the expected frequency with the input signal, and adjust its clock accordingly). We need bit transitions to keep in sync – **Manchester coding** sends a '0' as a falling edge, and a '1' as a rising edge to ensure this (it is a '1B2B' code, and also has DC balance).
- Broadband modulation can take a number of forms. For a carrier $A\cos(2\pi f_c t)$ and information signal $x(t)$:
  - **Amplitude modulation** – $A(mx(t)+1)\cos(2\pi f_c t)$, where m determines the degree of modulation. This is particularly susceptible to noise (e.g. spikes), although it uses very little bandwidth.
  - **Frequency modulation** – $A\cos(2\pi(f_c t + f_\Delta x(t)t))$, where f_Δ determines the degree of modulation. This is less susceptible to noise than AM, but may use a lot of bandwidth.
  - **Phase modulation** – $A\cos(2\pi(f_c t + \phi_\Delta x(t)))$, where ϕ_Δ determines the degree of modulation. This uses much less bandwidth than FM for the same effect (note we can still get phase noise, caused by clipping at low values).
  - We use **shift keying** to represent bits by instantaneous changes in amplitude, frequency or phase. Instantaneous changes in phase can be filtered out and/or synchronised to the carrier frequency.
- If we **digitise** an analog signal, we can introduce two sorts of noise:
  - **Quantisation noise** – the continuous values of the signal must be mapped to a finite set of discrete values. This introduces an error, which is unavoidable, but only one-time (once the signal is quantised, we don't introduce any more errors).
  - **Sampling error** – we also need to quantise time, since we can only sample at discrete intervals. However, the **Nyquist criterion** states that if a signal has a maximum frequency of BW Hz, then so long as the sampling frequency $f_s > 2 \times BW$, we can reconstruct the signal with no sampling error (e.g. 4kHz telephony sampled at 8kHz). This is because we can reconstruct the highest frequency component so long as we have more than two sampled points in a period.

3 Error Correction and Detection

- **Forward error correction** (FEC) is used to reduce the error rate by sending redundant information (so that we can correct some errors at the other end). The most trivial example is sending every bit three times, using majority voting.
- **Block codes** are one form of FEC:
  - Divide the message into fixed size blocks of length m. Each is encoded (one-to-one) to a larger **codeword** of size k (this is an (m, k) code). The **code** is the set of valid codewords.
  - The **Hamming distance** is the minimum distance (in terms of differing bits) between two valid codewords. Errors can be corrected by mapping to the closest codeword, or simply detected if too far away – to correct c-bit errors, and detect e-bit errors, we require $2c < d$ and $2e \leq d$.
  - A simple **systematic** block code (the message appears 'in the clear' inside the code) writes the block as an n by n matrix, and calculates the parity (number of '1'-bits modulo 2) of each row and column. It also calculates the parities of the calculated row and column (should be the same). This can correct a single bit error, as the parities of the rows/columns will not match those sent in two instances. A 2-bit error will be incorrectly corrected.
- Information theory states that for a bit error probability P_e, there is a code with rate R arbitrarily close to the channel capacity $C = 1 + P_e \log_2 P_e + (1 - P_e) \log_2(1 - P_e)$. Note that as P_e tends to 0 or 1, we get a capacity of 1, and for $P_e = \frac{1}{2}$ the capacity is 0.
- **Error detection** generally works by the transmitter computing and sending check-bits, and the receiver comparing these with re-computed values. If an error is detected, we need to retransmit the data.
- A **checksum** works by dividing the message into words (e.g. 32-bit), and summing them all modulo the word length. This depends most on the lowest-order bits though, so we have a minimum distance of only 2.
- A **CRC** (cyclic redundancy check) works by considering a bit sequence to be a polynomial $M(x)$ (with the bits representing its coefficients). We use another predefined polynomial $P(x)$ of degree n (the number of check-bits):
  - Multiply $M(x)$ by x^n (right shift n places) to get $M'(x)$.
  - Find the remainder $R(x)$ of $M'(x)/P(x)$.
  - Transmit $M'(x) + R(x)$. This is divisible by $P(x)$, which is checked at the other end. $P(x)$ is chosen to reduce the probability of a wrong acceptance.
  - To perform the division in hardware, use a sequence of n shift registers, with an XOR gate before those with a non-zero coefficient in $P(x)$, with extra input being the last shift register. Shift in the value you want to divide (high-order bits first). When the most significant bit in the shift register is set and shifted out, it subtracts $P(x)$ from the value in the register (as addition and subtraction are equivalent modulo 2). This equates to subtracting some multiple of $P(x)$ from the input. Once the input has been completely entered, the remainder of dividing by $P(x)$ is contained in the shift register.

4 Compression and Encryption

- Under **perfect compression**, $\mathcal{D}(\mathcal{E}(x)) = x$, e.g. with Huffman coding, gzip etc (lossless compression).
- If the compression is imperfect, the **stability** determines how fast we diverge from x when applying compress/decode alternations. We need to know something about the data (e.g. mapping 12-bit samples to 8-bit logarithmic values using A-law or μ-law companding in digital telephony).
- Video compression tries to eliminate redundancy:
  - **JPEG** uses transform coding on 8 by 8 pixel tiles, by using a Fourier transform to determine which of the frequency components have zero content. **Motion JPEG** uses JPEG on successive frames.
  - **MPEG** codes the differences between frames, using a full I-frame (intraframe) every 8 frames, with P-frames (predictor) giving difference information from the previous frame (e.g. translations). The other B-frames (bidirection interpolated) are just interpolations between I- and P- frames. MPEG-1 works at up to 1.5Mbps (CD speed), and MPEG-2 up to 15Mbps (DVD).
- **Encryption** adds a **key** to the coding/decoding functions. For symmetric cryptosystems, the sender and receiver know the same secret key. This can be used for authentication (encrypt a challenge with the key), integrity (send the message with an encrypted hash of it) and confidentiality (encrypt the message).

5 Multiplexing

- **Multiplexing** produces a number of higher layer channels from a single lower layer channel, using a **coding** to distinguish them, and a **policy** to determine access to the base channel.
- **Frequency division multiplexing** (FDM) divides the bandwidth of the channel into different frequency components (since sinusoids are orthogonal). This is used in TV and radio, where the channel is free-space (Ofcom assign frequencies to transmitters). **Wave division multiplexing** (WDM) uses higher frequencies (different light wavelengths in fibre optics).
- **Time division multiplexing** (TDM) transmits different higher-layer channels at different times (e.g. scheduling of TV programmes):
  - **Synchronous TDM** – the channel is divided into periodic times, such that the higher level using it is determined by the slot number in the frame. Digital telephony uses a 125μs frame, with 32 8-bit slots (as we're sampling at 8kHz). A constant bandwidth, constant delay channel is a **circuit**.
  - **Asynchronous TDM** – the time division is aperiodic, so variable sized **packets** may be put out at different times. We hope that all the higher level channels don't try to use their peak bandwidth at the same time (statistical multiplexing). Each packet needs a label (identifier), and we have the possibility of contention – for point-to-point links, the multiplexer implements this. The channels are both variable delay and capacity.

- **Random access** allows anybody to access the shared channel at any time. If two packets collide, it takes twice the channel delay to realise this. If we are transmitting when we see a collision, stop transmitting. We need to back-off and retry later (various delay schemes for this) whenever we see a collision. While good under light load, we can get starvation under heavy loads and we have no 'fairness control'.
- **Token passing** requires the transmitter to hold a 'token'. This is often used in a ring (e.g. token ring), where the sequence of token-passing is fixed. The token could disappear or get duplicated though.
- **Reservation systems** require the transmitter to reserve a time to access the channel in advance (e.g. synchronous TDM). This is good for large channel delays, but we need a reservation request system.
- **Slotted systems** divide the channel into fixed timeslots, but have asynchronous access to the slots (statistically multiplexed). This can allow synchronous reservation as well, but we need addressing information in each slot.
- **Centralised mutiplexing** is when a station must decide which higher level channel gets scheduled to use the shared media first:
  - **Priority scheduling** can be used to give certain information (e.g. streaming media) a bigger (or more reliable) share of the network capacity.
  - A **weighted round robin** system may work better (avoiding starvation), and we can use **policing** to prevent over-use by a high priority service.
- **Non-orthogonal multiplexing** deals with 'nearly orthogonal' functions (unlike FDM and TDM), such that we may get a bit of interference but we use error correcting codes to compensate. This means that we rely less on other transmitters co-operating. A good example is **frequency hopping spread spectrum**.
- **Code Division Multiple Access** (CDMA) assigns a code (pseudo-random sequence) to each channel. The channel can use the entire frequency and time domains:
  - The code (each bit is called a **chip**) is cycled at a much higher rate than the data, so if you XOR the code with the data bits, it acts to alter the transmitted frequency, spreading it across the spectrum.
  - The receiver uses the same code to determine its frequency response, and looks for correlations (plus statistical majority voting) to reconstruct the data signal. As we need to know the code, this gives some (limited) security.
  - This is commonly used in mobile phones (2G and 3G), as opposed to GSM. **Silence suppressed voice transmission** is used such that silences aren't transmitted, so there is less noise for other channels.
- Multiplexing can be **layered**, e.g. multiplexing access of ports to an Ethernet interface, and of the interface to the network. Telephone networks have different levels of connection, under the **Plesiochronous Digital Hierarchy** (PDH), e.g. T0 to T4 in America/Japan.

6 Shared Media Systems

- A **shared media** system has a fixed network capacity that is shared by a number of devices. This can be on the physical channel level (Ethernet), or just above the physical layer (token ring). This is opposed to a non-shared system (e.g. the telephone network), where we add network capacity in adding new links. If the shared link fails, it can cause complete network failure.
- **Ethernet** uses a single shared wire:
  - **Carrier Sense Multiple Access / Collision Detect** (CSMA/CD) – listen to the wire before talking, and check that nobody else is talking after you've finished. The collision window is twice the cable length, and we enforce this as the minimum packet size.
  - The retransmission strategy uses a truncated exponential backoff (we wait a random time up to some exponentially increasing maximum). When a collision is detected, send a jam pattern (32 to 48 bits) of anything other than the CRC of the bits before the jam.
  - The **collision domain** is isolated by a router or bridge, but connected by a simple repeater.
  - IEEE 802.2 (link control) and 802.3 (CSMA/CD) give one definition of Ethernet encapsulation – the alternative is RFC 894 (the most commonly used, although any host connected to a 10Mbps Ethernet cable should support both).
- **Token ring** connects devices by a ring of unidirectional point-to-point links, such that only the current token holder may transmit:
  - A free token is sent as a frame with no content. 'Code violations' are used to delimit tokens.
  - To transmit, wait for a free token and change it into a busy token (by altering one bit). Append the data to the token header (with addressing information) to create a data frame, sending it around the ring. Data that wraps back around is removed by the transmitter. After the data has been sent, a free token is released to the next device on the ring.
  - To perform token management, one of the stations is a **monitor**. It sends out 'monitor indentification' tokens from time to time, and if a device doesn't see any of these it will clain monitor status (contention resolved by highest address). The monitor ensures that one, and only one, token exists.
  - The IBM token ring is based on a star topology, with devices connecting to a **multistation access unit** (MSAU), which may forms the actual ring (the ring can be maintained if one device dies).
- **Slotted rings** divide the ring into fixed size slots (each with a full/empty marker) that rotate around:
  - The transmitter waits for an empty slot, and sets it to full. When it arrives at the destination, the data can be read and the slot made empty (or at least marked as received). If a full slot comes back to its source, it gets removed.
  - To ensure the freeing of slots, a monitor sets a 'monitor bit' on all passing full slots. When a slot is filled, its monitor bit is cleared. Any full slots passing the monitor are removed if their monitor bit is already set.

7 Switching

- The purpose of a **switch** is to get the higher layer incoming channels to the correct outgoing channels. This requires demultiplexing of incoming traffic, and remultiplexing of outgoing. We have a layer α channel between each device and the switch, but each higher level channel sees only a layer β channel between the devices.
- **Circuit switching** operates on a network with synchronous multiplexing, such that each channel is a circuit. The most common example is synchronous TDM over a telephone network. The switch provides a set of mappings $\{L_n : s_m \mapsto L_{n'} : s_{m'}\}$, for physical links L and timeslots s. The switch must preserve the quality of the channel (not introducing variable delay).
- **Packet switching** operates on variable delay, variable capacity networks, where data is sent as packets. The switch must recognise each packet, and direct it to the correct output (this differs from a **hub**, which is just a simple repeater). We must buffer traffic, since we may get multiple packets sent to one output – if the buffer capacity is exceeded, packets get thrown away.
- **Wavelength switching** (used in optical networking) can be done simply if we maintain the wavelength of the signal as we redirect it to the output. Converting to a different wavelength, however, is expensive (plus we need knowledge of the lower level details as to how the light is modulated).

8 Error Control Protocols

- A **protocol** is a set of procedures and formats used by entities to exchange information. Higher level networking is concerned with state changes in the entities – communication is difficult as we have only an approximation to another entity's state.
- **Automatic Repeat reQuest** (ARQ) is an error control protocol – use an error detecting code, and retransmit when necessary:
  - We divide information into **frames**, each with a sequence number, a CRC and a type (DATA, ACK or NACK). DATA frames also contain information.
  - When a DATA frame with correct CRC and expected sequence is received, send an ACK frame with the next expected sequence number. If the CRC or sequence are incorrect, send a NACK frame with the sequence number it is still waiting for.
  - For a frame size p bits, channel capacity b bits/s and delay d s, it takes $\tau_{tx} = \frac{p}{b}$ s to transmit a frame, and $\tau_d = d$ s for it to travel down the channel. So the time between transmitting frames must be $2\tau_d + \tau_{tx}$, since we must allow time for an ACK (or a NACK) to be sent back. For a larger channel delay, the data throughput approaches $\frac{p}{\tau_d}$ as we can only send one frame per channel delay.
- **Continuous ARQ** has multiple frames in transit at once – we have a **window** of frames that may be in transit and be unacknowledged. A big enough window can keep the channel busy (and is better than just having big frames). If we have missing frame we can:
  - **Go back** N – restart transmission from the missing frame (repeating all subsequent frames).

  - **Selectively retransmit** – only retransmit the missing frame (others that are already acknowledged are not repeated).

- **Flow control** is the problem of sending data only at the rate the receiver can handle:
  - **X-on, X-off** – the receiver tells the transmitter when to start/stop. After issuing an X-off, it must be able to receive another two channel delays of information.
  - **Sliding window** – the receiver sets the window size of the transmitter when it sends ACKs, so it can force the transmitter to slow down the transmission rate.

9 Naming and Addressing

- There are different levels of naming – **names** identify entities, **addresses** identify network attachment points, and **routes** identify paths through the network.
- **Binding** provides a correspondance between identifiers (DNS binds names to addresses, routing binds addresses to routes).
- A **service access point** is the point at which an upper and lower layer entity attach, and is usually denoted by an address (e.g. ports between applications and IP entities, IP addresses between IP entities and Ethernet cards, Ethernet addresses between Ethernet cards and the network).
- A **flat address space** has no structure and is 'centrally' allocated (e.g. 48-bit Ethernet addresses).
- A **hierarchical address space** is divided up into successively more specific parts (e.g country, network, host). In the case of IPv4:
  - **Class A** network – address begins with '0' and has a 7-bit network identifier and 24-bit host.
  - **Class B** network – address begins with '10' and has a 14-bit network identifier and 16-bit host.
  - **Class C** network – address begins with '110' and has a 21-bit network identifier and 8-bit host.
  - **Multicast** – address begins with '1110' and has a 28-bit multicast address. A dynamic set of hosts (the **host group**) listen to each multicast address. This is used by IGMP.
- **Classless routing** can be used to aggregate two networks. If they only differ in the last bit of the network identifier, decrease each of the subnet masks to logically combine the networks (this is like the reverse of **subnetting**). This reduces the size of routing tables.

10 Routing

- **Routing** binds addresses to paths on the network, and may be either static or dynamic, central or distributed.
- The **Address Resolution Protocol** (ARP) translates an IP address to an Ethernet address (although it could potentially be used for different naming schemes):

  - A sends out an ARP request (Ethernet broadcast) containing its IP and Ethernet addresses and the IP address of B (whose Ethernet address is required).
  - When B sees this request, it updates its table with A's details, and sends its IP address in an ARP reply directly to A.
  - Other hosts that see the ARP request will update their tables about A (but not create a new entry).

- A **repeater** works at OSI layer 1 (the physical layer), and simply regenerates all signals it sees on the other side (e.g. collisions get propagated).

- A **MAC layer bridge** works at OSI layer 2 (the data link layer), and knows about Ethernet addresses. It has a set of addresses for which it will forward packets (**media access control**), and it only sends through packets that it needs to (these are quite simple, but potentially large tables, with fast lookup).

- An **IP router** works at OSI layer 3 (the network layer), and knows about IP addresses. If the router is on the required network, it sends the packet out to the host. Otherwise, it does a lookup in its **routing table**, which provides a list of network addresses and the next router to use for each (at the bottom of the table is the default entry, that specifies where to send any other packets). Routing is performed by searching linearly through the table for the first match. The router 'unwraps' the Ethernet packet, to read the IP header, then re-packages it as an Ethernet packet to the next router/host. Various dynamic mechanisms provide more sophistication than hand-maintained tables, such as **gateway-gateway protocols** (routers exchanging information) and **autonomous subsystems** (controlling the routes into themselves).

- There exist a number of different routing strategies:

  - **Flood routing** – repeat all incoming packets on all links other than the one they came in on (unless it is for you). Each packet has a **time to live** hop count that is decremented at each router (hop count of zero isn't repeated).
  - **Random routing** – send out an incoming packet on a randomly chosen link. This is a theoretical boundry case.
  - **Shortest path routing** (link state routing) – try to find the least cost path between nodes, by building up a picture of the network. Each node exchanges information with other nodes. As the network topology changes, the views become inconsistent (which could cause loops), so we have to be very careful.
  - **Source routing** – the sender decides the route, embedding it in the packet (every hop is specified). This requires that the source knows the network topology, and changing conditions. Loose source routing enables you to specify some of the hops.

- Routing may be done:

  - On every packet (**connectionless**) – each packet (**datagram**) is routed separately, so may arrive out of order (but can adapt to a changing network).
  - When a channel is established (**connection oriented**) – routing is done only once, and all packets follow the same route (so are never out of order). These are **virtual circuits**(not actually fixed delay, fixed capacity), such that each packet is tagged with a virtual channel identifier.

- **Adaptive routing** takes link utilisation into account, whereas **hot potato routing** passes traffic off to another as quickly as possible (to avoid contention on outgoing links).

11 TCP/IP

- In the late 1970's there were a large number of vendor-specific and independant networks (e.g. DECNET, SNA). The X.25 protocol (connection oriented) was just emerging as a standard for network intercommunication. **Gateways** were used to translate protocols from one network to another (working at a higher level). The problems with this were 'solved' by pushing the intercommunication down to the IP layer (a lowest common denominator) so that gateways effecively became routers (this was made open source).

- An **IP header** contains the source and destination IP addresses, the protocol of the data (e.g. TCP/UDP), its time-to-live (TTL), length and checksum information, a fragment offset (if the data was too big and had to be fragmented), and other flags/options. The header must be some multiple of 64-bits in length (it may be padded).

- The **Transmission Control Protocol** (TCP) provides a reliable end-to-end delivery of packets:

  - It is **stream oriented**(a sequence of bytes), and **connection oriented**(although the network doesn't know). Information flow is **full duplex**.
  - The service is made **reliable** using a form of ARQ, so that lost packets are retransmitted. **Flow control** is used, allowing ACKs to set the window size.
  - The **TCP header** contains the source and destination ports, sequence and acknowledgement numbers (ACKs are 'piggy-backed' onto data packets), window size, a checksum, and some other flags/options.
  - In order to work out how to set the timeouts, we need to model the round trip time of the channel (and variance), and its capacity – the only way to build up this model is through losses.
  - **Slow start** is used, in that we start with a small window size, and slowly increase it until losses occur (very good for large transfers, but poor for smaller ones).
  - Flow control may be **end-to-end** (in the case of TCP and the Internet), **hop-by-hop**, or **entry/exit** (e.g. X.25).

- The **User Datagram Protocol** (UDP) provides an unreliable datagram service (runs at the same layer as TCP).

- **Application protocolos** typically use either UDP (e.g. echo, traceroute, DNS, NFS, SNMP) or TCP (e.g. HTTP, telnet, FTP, NNTP, SMTP). HTTP version 1 sets up a new TCP connection for every object on a webpage (each with slow start), whereas version 2 allows persistent connections. **Firewalls** are application level gateways.