3 OSI Layer 2 – The Data Link Layer

3.1 Goal of this section

The goal of this section is for readers to learn what the Data Link Layer of the OSI model is and what "interfaces" it uses to higher and lower layers in the OSI model. Further, readers will learn what matters about the Data Link Layer, understand some examples of Physical layer technologies, and tools used for monitoring and debugging Data Link Layer issues.

3.2 Purpose

The data link layer is about getting coherent clumps of bits (frames) delivered between known participants. Alongside this the Data Link layer provides additional services below.

3.2.1 Interfaces Up and Down

The interface to the physical layer is an ordered stream of bits; the data link layer provides a stream of bits to the physical layer for physical transport.

The interface to the layer above is a frame – a coherent set of bits of information to deliver that share fate, along with destination information. By sharing fate, we mean that when the Data Link Layer at A is given a frame of data and a destination B, either that frame is *all* delivered intact by the Data Link Layer to the Layer above at B, or it is not delivered at all – there is no partial delivery of information.

3.3 Services

Layer 2 typically provides the following services:

3.3.1 Frame Delivery

Layer 2 delivers coherent frames to destinations (using Layer 1 below).

3.3.2 Frame level error detection and reliability

Layer 2 provides some simple levels of error detection and reliability. For example, Ethernet includes in each frame of data a 32 bit hash, called the Ethernet Frame Check Sequence (FCS), to check the consistency of the frame . It discards any frame received with a bad FCS. (See the section below on Ethernet for more detail on how this is calculated).

3.3.3 Some form of addressing

Layer 2 does not (necessarily) consist of point-to-point links, and includes an addressing method by which higher layers can identify a particular destination to which a frame should be sent. For example, Ethernet uses MAC addresses (6 byte identifiers) to distinguish individual destinations.

3.3.4 Redundancy across Physical Layer links

Layer 2 provides abstraction across a network, providing the higher layers of the OSI stack with a single "connection" that could take multiple paths across different Physical Layer links. Examples of Layer 2 protocols that enable this include Link Aggregation Control Protocol (covered later in this section) and (Rapid) Spanning Tree Protocol (covered in the next chapter when we look at scaling up Layer 2 networks).

3.4 What does "Better" mean for Layer 2

The goal of the Data Link layer is to get frames delivered to the right place. "Better" covers all of

1. Higher Bandwidth – the more data you can get through your network, the better

2. Reliability – the less corruption, the less effort you need to spend on detecting that corruption and the less retransmission work there is.
3. Lower Latency and Less Jitter – Layer 2 is low level in the OSI stack, so low latency and jitter are important, as they impose a lower limit on what can be achieved at higher levels of the stack. (Other services such as guaranteed delivery etc. can be handled at higher levels, but those higher levels cannot reduce the latency or jitter of the underlying layers.)

3.5 Ethernet (Ethernet II)

Ethernet is the most common Layer 2 protocol, and we'll look into this in some detail as an example, as it demonstrates the important features of Layer 2.

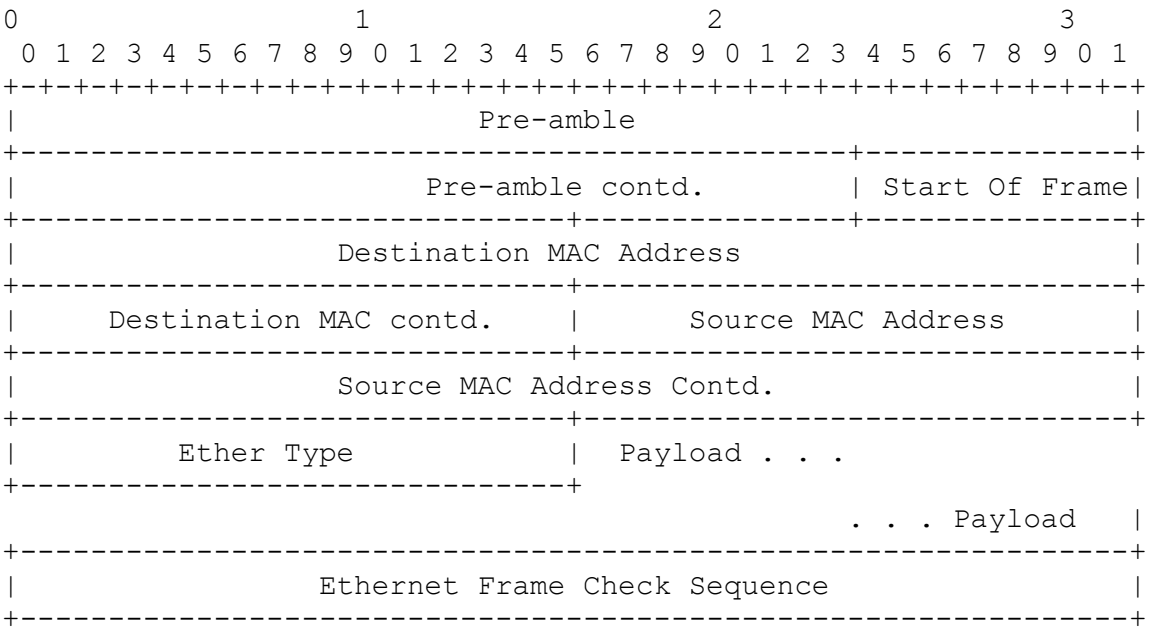
Ethernet runs point-to-point (two endpoints directly connected) or multi-access (multiple endpoints connected N-way through a network).

3.6 Ethernet Frames

We've only reached OSI Layer 2 and already we encounter our first example of "Layer breaking".

Although Ethernet is a Layer 2 Protocol, Ethernet Frames are wrapped in some additional pieces of information designed to help the underlying Physical Layer.

An Ethernet Frame consists of a header, footer, and payload, and is preceded by pre-amble and followed by an pause. The Pre-amble and Frame are depicted here:



Side note: What is that wadge of text above?

It doesn't have a name that I know (or could find!), but it's a standard byte-diagram used for showing the contents of a Protocol message. The contents of the message as transmitted is read top left to bottom right, and the rows contain 32 bits (can be more), both because 32-bit words were the norm when networking really took off and these became used, and also because 4 bytes is a reasonable amount for a human to see and visualise on a single line. Each piece of information is labelled, and the labels explained elsewhere. The bits in each line are numbered across the top, 00-31.

Where a piece of data crosses a 4 byte boundary, it is labelled on both lines. For example, you can see 7 bytes of Pre-amble, the 4 bytes in the first line and the 3 bytes of "Pre-amble contd." in the second line.

Where a piece of data is variable and could cross a large number of lines, that's typically shown by elipses. For example, the Payload in the diagram above can be hundreds of bytes.).

I'll now describe the Ethernet Frame shown in the picture above…

3.6.1 Pre-amble (Layer 1 help)

The Ethernet Header starts with 7 bytes of **Pre-amble**. The Pre-amble consists of alternating 1s and 0s. The receiver at the other end of the link, will be looking for 1s and 0s, but the line might have been empty with nothing travelling on it for some time, and so the alternating 1s and 0s give the receiver a chance to get its clock synched up ("When am I measuring if something is a 1 or 0?").

After 7 bytes of Pre-amble, is the **Start of Frame** Byte, which is always 0xAB (which in binary is 10101011). The receiver might not have locked its clock onto the Pre-amble exactly at the start of the alternating 1s and 0s, and so the 0xAB tells it where the byte boundaries are and that it's time to start receiving the Frame itself…

3.6.2 Ethernet Header

The header consists of three key pieces of information

1. The Destination MAC (Media Access Control) address is a 6 byte unique identifier, indicating the destination
2. The Source MAC address similarly identifies the source of the frame
3. The "Length" field

MAC addresses

MAC addresses are 6-byte values that are unique within a particular network. Historically MAC addresses were not configurable and were baked into the hardware of each Ethernet network card. To enable plug-and-play uniqueness, that required those MAC addresses to be globally unique. There is no network hierarchical structure to MAC addresses, though to ensure global uniqueness, some hardware manufacturers "own" some MAC address prefixes.

Humans write MAC addresses as 6 pairs of Hex characters separated by hyphens. As an example: 01-a3-0b-de-2f-00.

Ether Type / Length

The Type field contains an indication as to what the payload is, including an indication for what the structure is. This could indicate to the receiving Layer 2 stack that it is some kind of Layer 2 protocol (and how to interpret the payload) or could be information to pass up to the higher layer of the stack to help it in parsing the payload. For example, 0x0800 indicates that the payload is an IPv4 packet (which is a Layer 3 protocol that we'll cover later in later chapters).

Note that there are some other IEEE Ethernet specs (such as Raw Ethernet 802.3) that use the Ether Type field to cover the length of the payload. In short, all Ether Types are >1500 (the maximum Payload length). If you see an Ethernet Frame with Ether Type <=1500, this should be interpreted as a Length, and they Payload is interpreted using a more complicated set of rules, not covered in this course.

3.6.3 Payload

The Payload of the Frame is then the data that is being transmitted (which has been handed down from a higher layer in the OSI model. That payload data is meaningful to the higher layer, but opaque to this layer – as far as we're concerned it's a bunch of 1s and 0s that we've been asked to transport. Ethernet Frames are 64-1500 bytes long, so the minimum payload length is 46 bytes (and it is padded out to that, and indeed padded to the nearest 4 byte boundary as needed to allow the Footer to start on a 4-byte boundary).

3.6.4 Footer

The footer simply consists of the Frame Check Sequence. This is a 4 byte Cyclic Redundancy Checksum over the whole frame, and provides checking for corruption. We cover how this works and how it is calculated later.

3.6.5 Post-frame Gap (Layer 1 help)

After completing transmission of a frame, the end of the frame is signalled by allowing the line going back into idle state. An Ethernet endpoint leaves a gap (12 octets worth of clock cycles to be sure) before transmitting another frame, to allow the line time to truly return to idle.

3.7 Local Area Networks (LANs)

The simple way to convert an ethernet network from point to point to multi-access is to use a Hub (covered in the next section). Hubs have multiple ports, to allow multiple point-to-point cables to be plugged into them, and copy each Frame that arrives over one Port and send it out every other Port. This Layer 2 network is called a **Local Area Network** or LAN. Every endpoint on the LAN receives every Ethernet Frame – even if it was not addressed to it. Ethernet endpoints drop any Frame that they receive that is not addressed to them (the Destination MAC address in the Frame header does not match their MAC address).

3.8 Frame Transmission

To transmit a frame, an endpoint simply attempts to transmit the frame, which is delivered across the network to everywhere else. If the endpoint detects another endpoint attempting to transmit at the same time, a **collision,** then it stops sending (because elsewhere in the network, a third endpoint might be receiving both frames at once down the same cable). It tries again after a randomised delay (and again and again, with an exponential backoff between delays).

The more endpoints there are on a network, the more traffic, and the more collisions, until the network becomes saturated.

A wireshark trace showing both Ethernet traffic and Token Ring traffic (another different Layer 2 protocol which is now defunct) is at 3_Ethernet_And_TokenRing.trc0 stored alongside these notes. Have a look at this and work through the Ethernet header for some Ethernet packets.

3.9 Calculating a Cyclic Redundancy Check

Ethernet uses a Cyclic Redundancy Check as the "hash" for detecting corrupted bits in the frame. This is calculated as follows:

1. Consider a bit string as the coefficients of a polynomial in $Z_2[x]$ (a polynomial where every coefficient is 0 or 1).
2. Divide the input into chunks of length m (consider the equivalent polynomial for each of those M(x)) and agree on a well-known generator polynomial G(x), which is of degree < m.
3. To send M(x), divide by G(x) and calculate the remainder R(x) = $(M(x) * x^r) \bmod G(x)$.
4. Instead of transmitting M(x), transmit T(x) = T(x) = $(M(x) * x^r) - R(x)$.
5. T(x) is divisible by G(x) exactly, so when the receiver divides G(x) by R(x), they should get 0.

Note that in $Z_2[x]$ addition and subtraction are both bitwise exclusive-or.

3.9.1 Cyclic Redundancy Check Example:

A user wants to send the message {01011} in a system where $G(x) = x^3 + x^2 + 1$. (By prior agreement).

1. To send {01011} with $G(x) = x^3 + x^2 + 1$, encode it as $0x^4 + x^3 + 0x^2 + x + 1$.
2. $G(x) = x^3 + x^2 + 1$
3. Multiplying {01011} by x^3 gives {01011000}. Dividing this by G(x) gives {01100}, remainder {100}.
4. Subtracting (== adding) {100} from {01011000} gives {01011100}. Transmit this.
5. On receipt, dividing {01011100} by G(x) gives remainder {000}. Hurrah!
    - Conversely if we had received {01010110}, dividing by G(x) gives remainder {111}, and we know there is a problem.

This detects any error not divisible by G(x).

3.9.2 Ethernet Frame Check Sequence

Ethernet uses a 32bit remainder check called the Ethernet Frame Check Sequence, which uses the polynomial: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$. For reference, that's indices: {10000010011000001000111010110111}. Why? This detects the most likely errors that occur during bit transmission. In particular, it detects (a) Any single-bit error (and in fact any cases where there are an odd number of total errors) (b) All pairs of 2-consecutive-bit errors that are less than 2048 bits apart, (c) Any single burst of errors that spans <32 contiguous bits.

3.9.3 An IEEE 802.3 Spec side-note

There are actually a whole family of standards in this area that build additional features and abilities on Ethernet, all called 802.3xy (with some letters after the 802.3). For example, Gigabit Ethernet is 802.3ab. The standards are named with letters alphabetically in chronological order, starting with single a-z, and then starting again with aa-az, ba-bz etc. This is one of many examples in networking where what originally seemed to be a sensibly sized name-space or numbering system turns out not to work well at large scale.

3.10 Problems / Troubleshooting / Tools

Beyond your network saturating through trying to put too much data through it, almost all Data Link Layer problems are one of two:

1) Your network does not have a connection across it. In this case you'll see no traffic.
2) Your network has a loop in it. In this case you'll see all traffic forwarded endlessly, and your network will be instantly saturated.

Tools for debugging these problems include

- Connection and collision lights on switches/hubs. Most modern ethernet ports have two lights by them. The connection light indicates that there is connectivity to the other end of the link, and the collision light blinks every time there is a collision. No lights indicates you're in problem (1) above. Both lights indicates problem (2).

- A hub (with a port mirror) and tcpdump/wireshark tools. If you want to see what's going on in your network, attach a debugging endpoint to one of your hubs. It will get a copy of everything going on. You can then run a logging tool on that endpoint (tcpdump or wireshark are good choices) and have a look.

  Because of some network cleverness that I've glossed over, but we'll come on to next chapter, your debugging endpoint might not actually get a copy of everything – and you

might need to configure the hub that you've plugged your cable into to "port mirror" to your cable (explicitly copy everything going through the hub to the port that your cable is plugged into).

- Linkloop (Linux). Linkloop is a "ping" for a MAC address. If you don't know if you have a connection to a particular MAC address, Linkloop sends a Frame and elicits a responding frame from just that MAC address, and reports a successful (or not!) round trip back to the user.

3.11 Link Aggregation Control Protocol (IEEE 802.1 AX)

Link Aggregation Control Protocol allows us at Data Link layer to aggregate multiple Physical Layer links into a single Layer 2 link. This allows us to both present a larger bandwidth link than is possible with individual Physical Layer links, and also to provide a Data Link layer connection that is resilient to one or more of the underlying Physical Layer links failing.

In abstract, both ends of the LACP aggregated link are configured by a user. The physical ports to be aggregated are assigned a communal port channel. LACP runs at Layer 2 at both ends of those links, and handles sending frames down each of the underlying physical links.

LACP frames (called LACP Data Units or LACPDUs) are ethernet frames that are sent down each of the underlying links as a keepalive mechanism. These are sent to the well-known LACP multicast MAC address (01-80-c2-00-00-02) and are received by the LACP Protocol at the other endpoint. These are considered straightforward keepalives; if LACP does not use underlying physical links on which it is not receiving keepalives.

When the endpoint running LACP receives Ethernet Frames to send, it sends them down one of the underlying physical links which is live (according to the keepalives). If LACP assigned frames to links randomly, this could result in frames being delivered out of order at the far end of the link (which can be a problem for some higher level protocols). To maintain ordering, LACP can choose the physical link to use based on a simple hash of source/destination data. Typically, LACP can just use the source and destination MAC addresses, or it can use its knowledge of higher layer protocols to peek inside the payload at the addressing used there (another example of breaking strict layering being useful).

A wireshark trace for some LACP capture is stored alongside these notes, called **3_LACP.pcap** You are not expected to be able to decode the detailed fields of the LACP protocol, but do look and understand how the various Ethernet headers fit together.