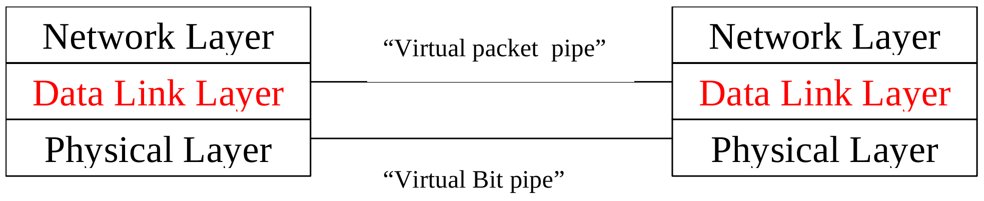
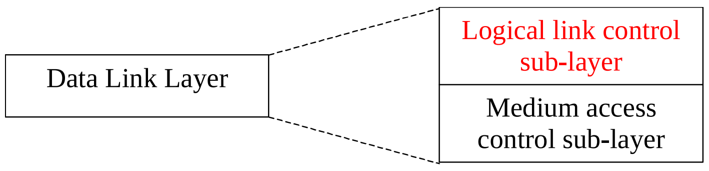
ECE 333: Introduction to Communication Networks
Fall 2001

Lecture 5: Data Link Layer I

- Introduction
- Framing

Data Link Layer




Data Link Layer
Logical link control sub-layer
Medium access control sub-layer

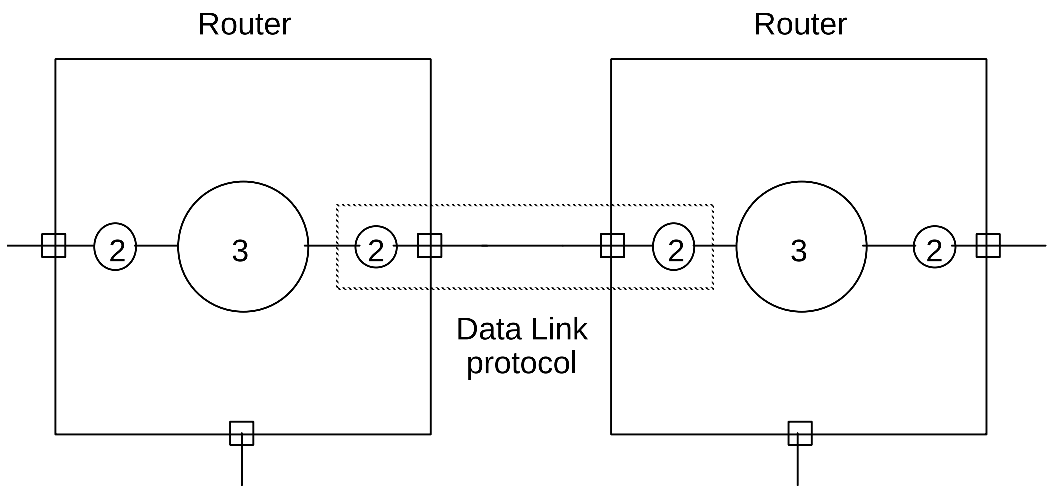
Notes:

In the OSI protocol stack, the layer directly above the physical layer is the ***Data Link Layer (DLL)***. The DLL is also refered to as ***layer 2.***

When the underlying link is a broadcast medium, the DLL is separated into ***MAC (Media Access Control)*** and ***LLC (Logical Link Control)*** sub-layers. The MAC sub-layer is responsible for determining which users get to use the broadcast medium at any time; we will study this sub-layer in several weeks. With point-to-point links, only the LLC sub-layer is needed. In the next few lectures, we will discuss issues that are addressed in the LLC sub-layer. As with the physical layer, the DLL still addressees "link-level issues" - that is the protocols at this layer are between two DLL peers - one at each end of a communication link.

DLL protocols can be implemented either in hardware or software. They can be implemented in either a computer's main CPU or in special purpose hardware, known as a ***network adapter*** or ***Network Interface Card (NIC)***. A common example of the later case is an Ethernet NIC.

DATA LINK LAYER



- 2 DLL peer's for each link.

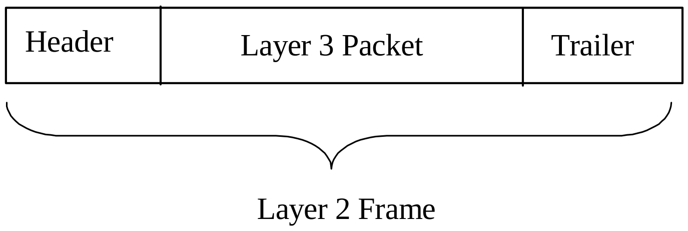
Notes:

Recall, the physical layer provides the DLL with a “virtual bit pipe,” which allows DLL peers to exchange a continuous stream of bits. Loosely, the goal of the data link layer is to convert this bit pipe into "virtual packet pipe" that will allow peers at the network layer to exchange ***packets*** (strings of bits). There are two main issues we will consider here: (1) ***Framing*** and (2) ***Error control***. Framing refers to the problem of deciding which bits belong to which packets of data at the receiving DLL peer. Error control arises because the bit pipe provided by the physical layer may introduce errors in the packets, it may also add additional bits or remove some bits in some cases. Both of these issues also arise at higher layers.

We discuss several possible characteristics of the service provided by the data link layer (these terms will also be used in discussing the service provided by other layers). First we consider the reliability of the service. The service may be ***confirmed*** or ***acknowledged*** meaning that the transmitting DLL informs the network layer whether or not a packet was correctly received. The service may also be ***unconfirmed***, in which case no confirmation is given if a packet arrives or not. Finally the service may be ***reliable***, meaning that the DLL guarantees that every packet is received correctly, in order and only once.

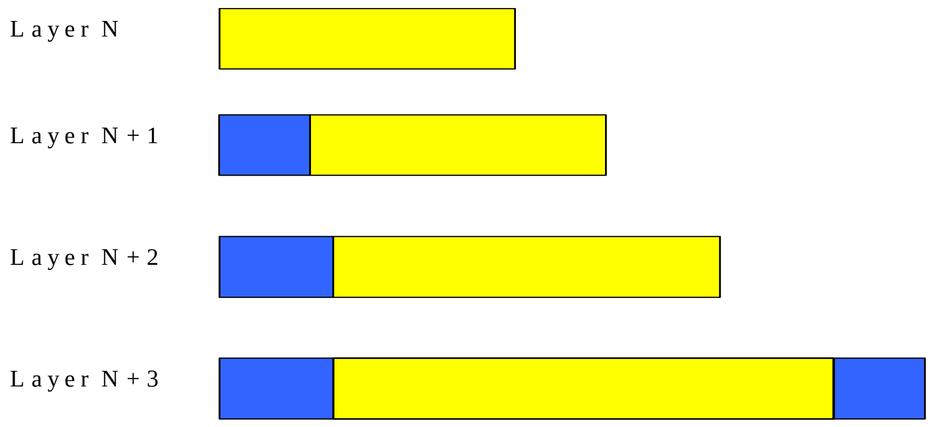
Another classification of the offered service is as either ***connection-oriented*** or ***connectionless***. Recall, we used this distinction to classify networks in Lecture 1 - here we are using it to classify a service at the DLL, but the meaning is the same. A connectionless DLL would not require any connection setup before packets could be sent, while a connection-oriented DLL would require such a setup.

Encapsulation



- Header and/or Trailer added at DLL.
- Repeats at higher layers.

Encapsulation



Notes:

To provide the required services, the sending DLL peer needs to add some additional information or ***overhead*** to each packet received from the network layer.
This overhead is typically in the form of either a ***header*** at the beginning of the packet and a ***trailer*** at the end of the packet. The combination of header, network layer packet, and trailer can be considered a new packet - to distinguish this from the network layer packets; the string of bits formed at the DLL is sometimes referred to as a ***frame.*** The process is typically repeated at higher layers; at each layer the data from the higher layer is ***encapsulated*** in a larger packet by adding a header and/or trailer. (Sometimes, however, higher layer packets are divided or grouped together at the next layer - we will discuss this when we consider the network layer.)

Framing

Issues:

1. Separate packets from each other
2. Separate packets from idle.

Four approaches:

1. Character Count
2. Starting and ending characters
3. Starting and ending flags
4. Physical layer coding violations

Character Count:

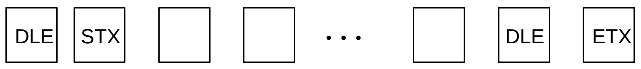
Place number of bits(or characters) in frame as first piece of information.

Problems ?

Framing Characters

- Based on particular character set, e.g. ASCII character code
- Begin each frame with DLE STX.
- End each frame with DLE ETX.

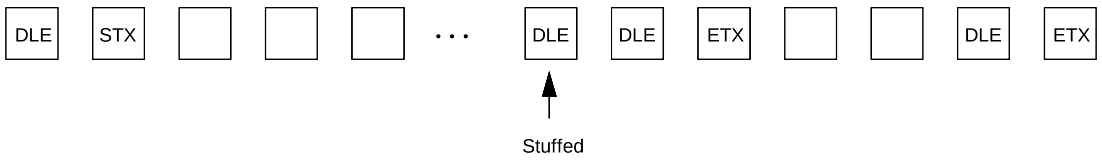
DLE	-	*Data Link Escape*
STX	-	*Start of Text*
ETX	-	*End of Text*



What happens if a DLE ETX occurs in the middle of the data?

Character Stuffing

Any DLE that occurs in the data is doubled.



Receiver interprets single DLE as escape indicating the next
character
is a control character.

If two DLE's in succession, the first is discarded and the second is regarded as data

Framing Bits

Bit pattern framing uses a particular sequence of bits called a ***flag*** for framing.

Uses ***bit stuffing*** to prevent the sequence in the data.

Example:

Flag = 0 1 1 1 1 1 1 0 (begins and ends frame.)

Transmitter automatically stuffs a 0 after 5 consecutive 1's in data.

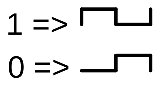
⇒ only time 6 1's are received is in flag.

Original Data	1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1
Sent Data	?

Framing by Illegal Code

Recall certain physical layers use a line code for timing reasons.

E.G.: Manchester Encoding



Codes of all low or all high aren't used for data and therefore can be used for framing.

Notes:

Framing refers to techniques that are used to tell when a frame begins and ends. In cases where the physical layer is a "synchronous bit pipe" (see lecture 3), it must be supplied with bits at a fixed rate, even when there is no data to send (one reason for this is that the receiver relies on the arriving bits to maintain timing information) When there is no data to transmit the DLL must insert “idle fill”, in such cases framing techniques are also used to differentiate idle fill from useful data.

The problem of framing is complicated by the fact that errors can occur in the received data. These errors can cause a loss of framing that must be recovered from.

Character or length counts work by simply placing the number of characters in the frame as the first piece of information. Assuming the beginning of some frame is known, the next frame boundary can be found by counting. There are length counts in the convergence sub-layers of ATM, in IP, and in many transport protocols. A disadvantage is that resynchronization is difficult after an error in the length count. Also length counts do not address the idle fill problem.

The use of framing characters have their roots in protocols developed in the 1960's to connect terminals to main-frames, such as DECNET and BISYNC. When character-based framing is used each frame is required to contain an integral number of characters per frame.

Using framing bits is just an implementational variation of character-based framing. With bit-based approach, the data in a frame is not tied to a particular character set.

Note the use illegal codes for framing breaks strict layer boundaries with physical layer. That is the framing service provided the network layer is tied to an internal detail of the physical layer implementation.

In practice, various framing methods are often combined. For example, bit stuffing may be used for framing and a character count may also be present, for extra protection. Another example is the framing used in Ethernet given below.

Example: Framing in IEEE 802.3 (Ethernet)

> An illegal Manchester coding is used for framing. A character count is also included in the frame header. Also, since all frames have an integral number of bytes, if the received frame does not end on a byte boundary, frame is considered to be received in error.