DATA LINK LAYER

September 26, 2020

Outline

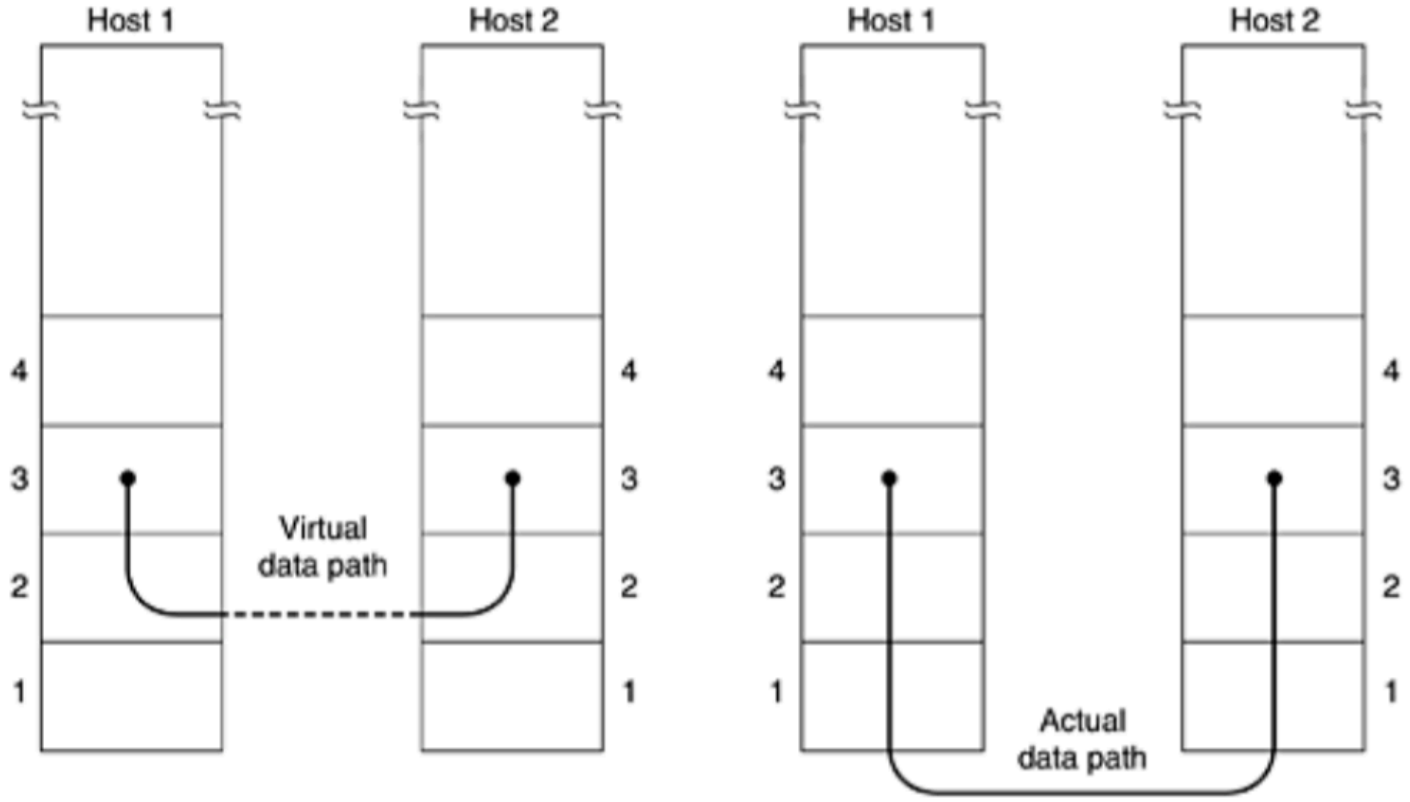
1. Frames
2. Framing Methods

Data Link Layer/Control

- This layer has a number of specific functions that include:
  1. Well-defined service interface to the network layer.
  2. Determining how bits from the physical layer are grouped into frames.
  3. Error correction and detection.
  4. Regulating the flow of frames through retransmission strategies.
- The main abbreviations are:
  - Data Link Layer (DLL)
  - Data Link Control (DLC)

Virtual Connection

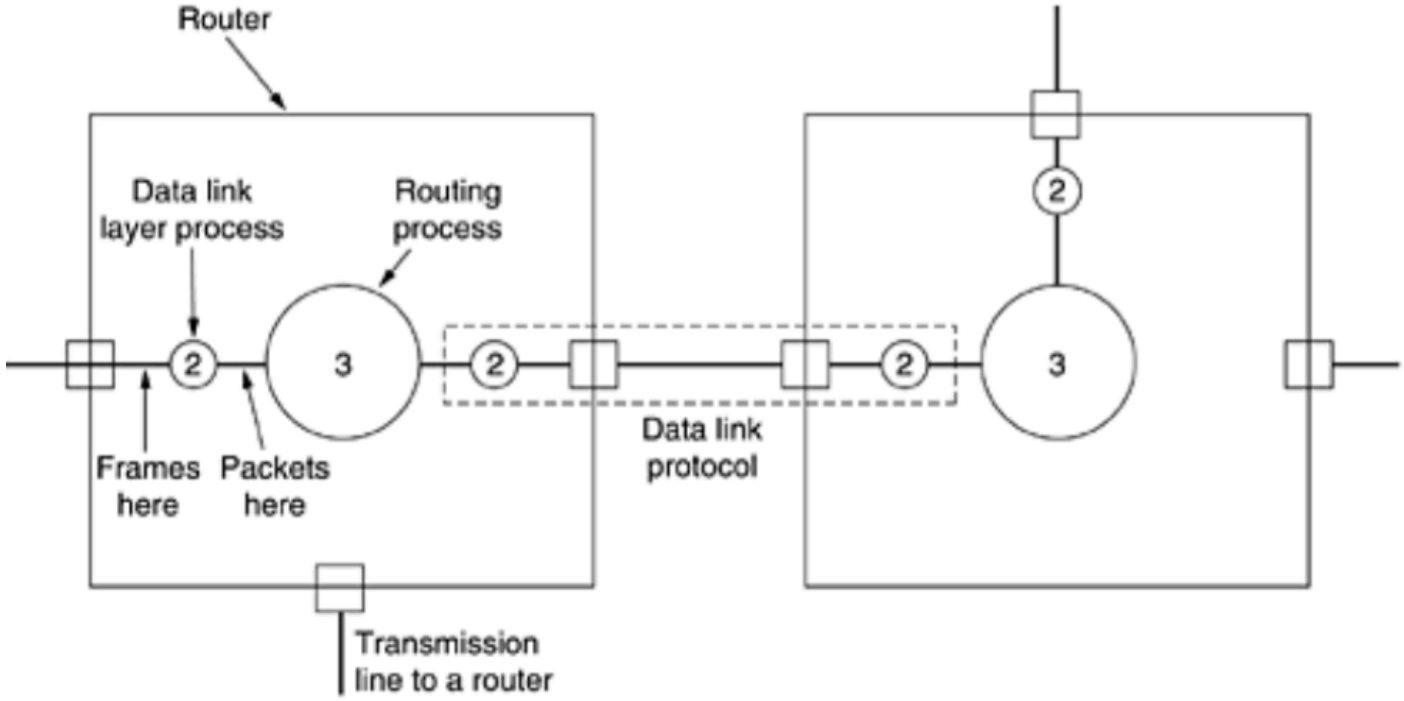
- Virtual communication in Data Link Layer



- versus actual communication.

Location with Respect to Router (1/2)

- When a frame arrives at a router, the hardware checks it for errors, then passes the frame to the data link layer software.



(which might be embedded in a chip on the network interface board)

Location with Respect to Router (2/2)

- The data link layer software checks to see if this is the frame expected, and if so, gives the packet contained in the payload field to the routing software.
- The routing software then chooses the appropriate outgoing line and passes the packet back down to the data link layer software, which then transmits it.

Frames

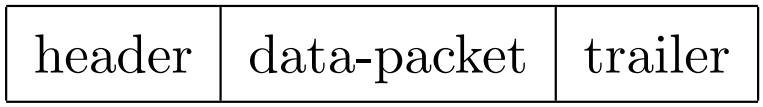
Frames and Pictures

- The idea of the terminology is as usual

- There is a picture (packet) and is placed inside a frame.
- The idea is that the picture is "most important" and must be ptotected by a frame!

Frames and Packets

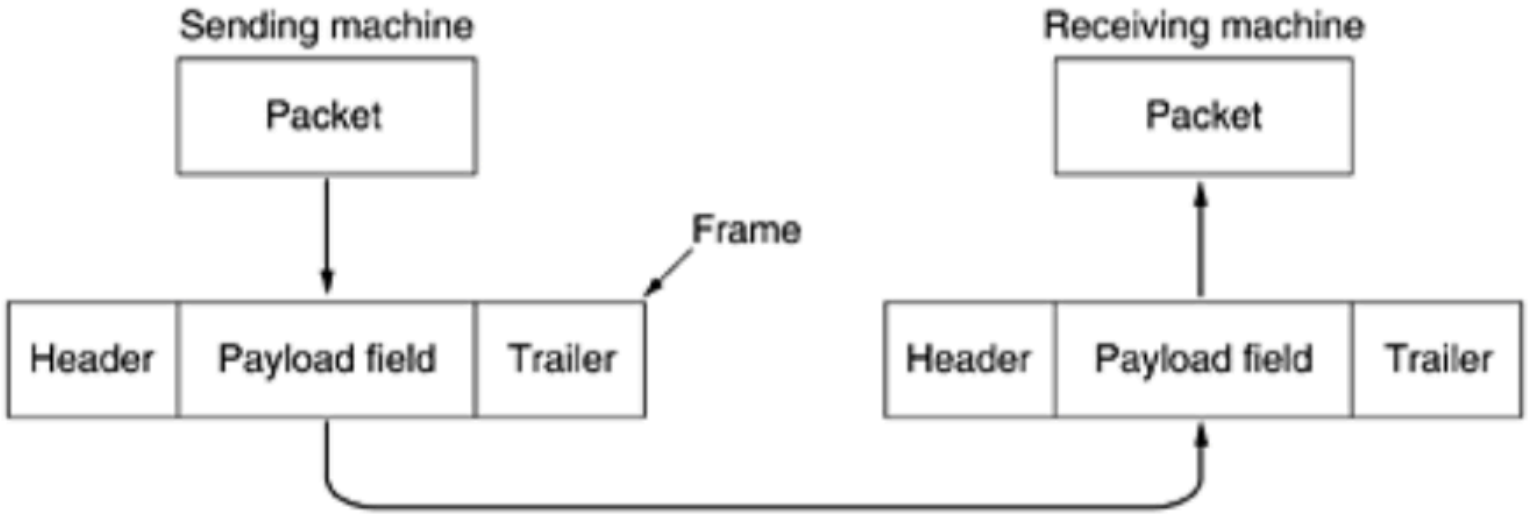
- Data is combined and organized in **frames** and **packets** for transmission purposes in the physical layer.
- **Problem 1:**
  Decide at the receiving DLC (Data Link Control) where successive packets start and end.
  - A typical frame structure contains the following



- **Problem 2:**
  Transmission errors may occur and issues of interest include error correction, error detection.

Framing

- Each frame contains a frame header, a payload field for holding the packet, and a frame trailer



- Frame is transmitted from Sender to Receiver.

Framing

- While "Physical" layer produces virtual bit pipe with no concept of packets, "Data Link" layer has input/output packets
- DLC (Data Link Control) places
  - **header** bits at the beginning, and
  - **trailer** bits at the end

  of packets to form **frames**
- Purpose of header/trailer:
  - Error control (e.g., CRC)
  - Distinguishing one frame from the next and from idle

How do you identify Frames?

- First task is to delineate frames
  - Receiver needs to know when a frame starts and ends
  - Otherwise, errors from misinterpretation of data stream
- Several different alternatives
  - Fixed length (bits) frames
  - Explicitly delimited frames
  - Length-based framing
  - Sentinel-based framing (use markers)
- Fixed duration (seconds) frames

Framing Methods

September 26, 2020

Framing Methods

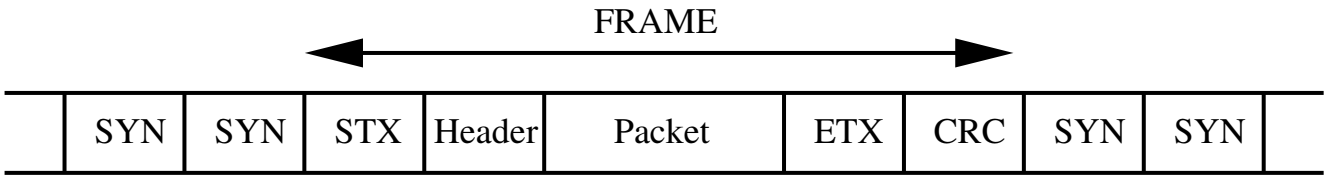
- Four methods used to distinguish beginning and end of frames:
  1. **Character-based framing**
     uses special communication control characters.
  2. **Bit-oriented framing with flags**
     uses flags (i.e., special strings of bits).
  3. **Clock-based framing**
  4. **Length fields**
     frame length given in the field.
- The idea is to "enforce" simple presentatiom techniques to mitigate inconsistencies and reduce errors.

Character (or Byte) Based Framing (1/4)

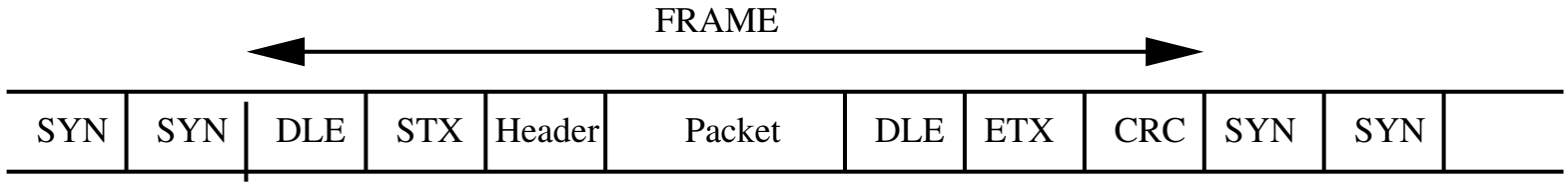
- Developed for and used in character-oriented communication, e.g., sending ASCII
- Special characters set aside for
  - Synchronous idle (SYN): provides idle fill between frames, used within frames for synchronization, bridges delays in supplying characters.
  - Start of text (STX)
  - End of text (ETX)
  - Data Link Escape (DLE): used only in special modes of transmission to provide more transparency (e.g., international uses of communication characters).

Character (Byte) Based Framing

- Uses characters as the basis of frames.



SIMPLIFIED FRAME STRUCTURE



CHARACTER BASED FRAMING AS USED IN ARPANET

SYN = Synchronous Idle
STX = Start of Text
DLE = Data Link Escape
ETX = End of Text

- Certain combinations of characters are given special meaning.

Character (Byte) Based Framing

- This framing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes.

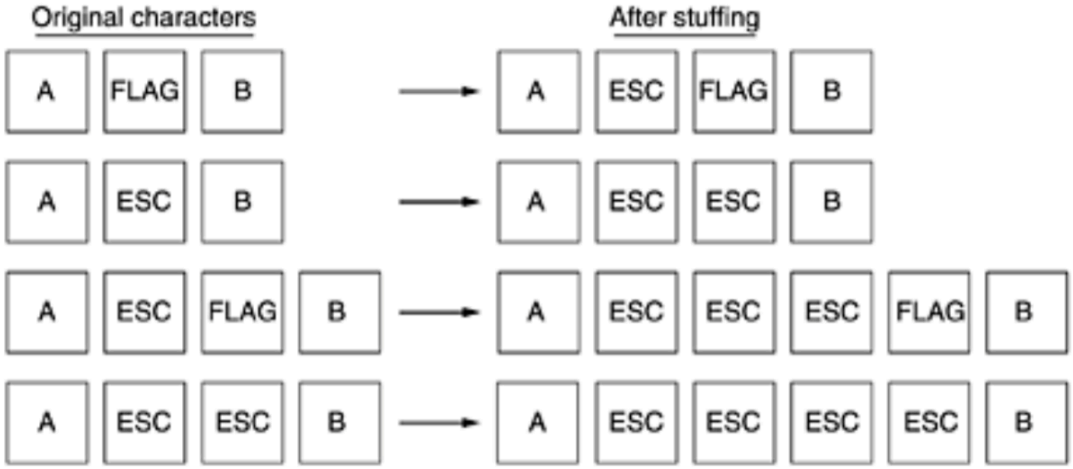
- In the past, the starting and ending bytes were different, but in recent years most protocols have used the same byte, called a flag byte, as both the starting and ending delimiter, known as FLAG.
- If the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame.
- Two consecutive flag bytes indicate the end of one frame and start of the next one.

Problems/Solutions: Byte Stuffing

- **Problem:** When sending arbitrary data (as opposed to characters like ASCII) control characters may appear in frame
  - **Solution:** introduce DLE (data link escape) character
  - Start of text indicated by DLE STX, end of text by DLE ETX
- **Problem:** What if DLE appears in packet as part of the data?
  - **Solution:** Use DLE DLE - the first DLE of every pair is stripped off at the receiving end
  - E.g., X DLE ETX means end of text but X DLE DLE ETX means data bits X DLE ETX

Problems/Solutions: Byte Stuffing

- The sender's data link layer inserts a special escape byte (ESC) just before each "accidental" flag byte in the data and the data link layer on the receiving end removes the escape byte before the data are given to the network layer.



- This technique is called byte stuffing or character stuffing.
- Thus, a framing flag byte can be distinguished from one in the data by the absence or presence of an escape byte before it.

Errors

- Errors in DLE STX or DLE ETX may cause a frame to be missed
- Errors inside the frame will (hopefully) be caught by CRC
- An error could cause DLE STX or DLE ETX to appear in the middle of the frame
- In either case a portion of the frame is dropped and remainder is unlikely to be accepted as correct due to CRC

Errors

- What happens if an escape byte occurs in the middle of the data?
- The answer is that it, too, is stuffed with an escape byte.
- Thus, any single escape byte is part of an escape sequence, whereas a doubled one indicates that a single escape occurred naturally in the data.

Extensions

- A major disadvantage of using this framing method is that it is closely tied to the use of 8-bit characters.
- Not all character codes use 8-bit characters.
- For example. UNICODE uses 16-bit characters,
- As networks developed, the disadvantages of embedding the character code length in the framing mechanism became more and more obvious, so a new technique had to be developed to allow arbitrary sized characters.

Bit-Oriented Framing (2/4)

- This technique allows the data frames to contain an arbitrary number of bits and allows character codes with arbitrary number of bits per character.
- Similar to above but a *flag* (fixed bit string) is used to delineate start and finish of frame
- Flags that could be used are strings of the form

$$01^j0 := 0\overbrace{11\cdots1}^{j}0$$

- E.g., ISO - High-level Data Link Control (HDLC) uses

$$01^60 := 0\overbrace{111111}^{6}0$$

as flag string.

Bit-Oriented Framing and Bit Stuffing

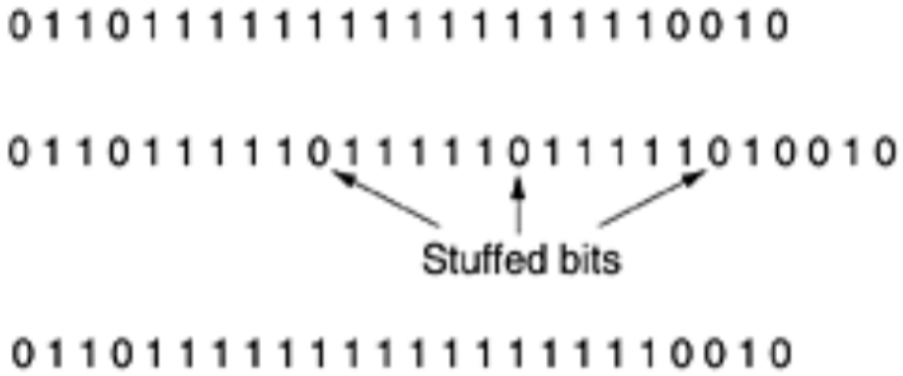
- To avoid interpreting data as flag *bit stuffing* is used.
  Sender:
  - With the exception of the flag 01^60, after every sequence of 5 1's a 0 is stuffed.

  Receiver:
  - The first 0 after every sequence of five consecutive 1s is deleted.
- **Idle:** 15 1's, i.e.. 1^{15}.
- **Abort:** is 7 1's in a row.

NB: *Read Ahead* is necessary to predict correctly whether it is idle or abort.

Stuffed Bits: Example

- Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream.



- When receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e., deletes) the 0 bit.
- If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110.

Receiver and Bit-stuffing

- If five consecutive 1s arrive, the receiver makes its decision based on next_bit.
- If next_bit = 0 it must have been stuffed and is removed.
- If next_bit = 1 then either this is end of frame or error has been introduced.
- To distinguish look at the next bit (after next_bit).

Example: Bit-Stuffing/Bit-Unstuffing

- **Sender**

$$00011111^{\downarrow}110011111^{\downarrow}01000$$

$$\downarrow \text{stuffing 2 bits}$$

$$00011111\mathbf{0}11001111\mathbf{0}01000$$

- **Receiver**

$$00011111\boxed{0}110011111\boxed{0}01000$$

$$\downarrow \text{bit unstuffing 2 bits}$$

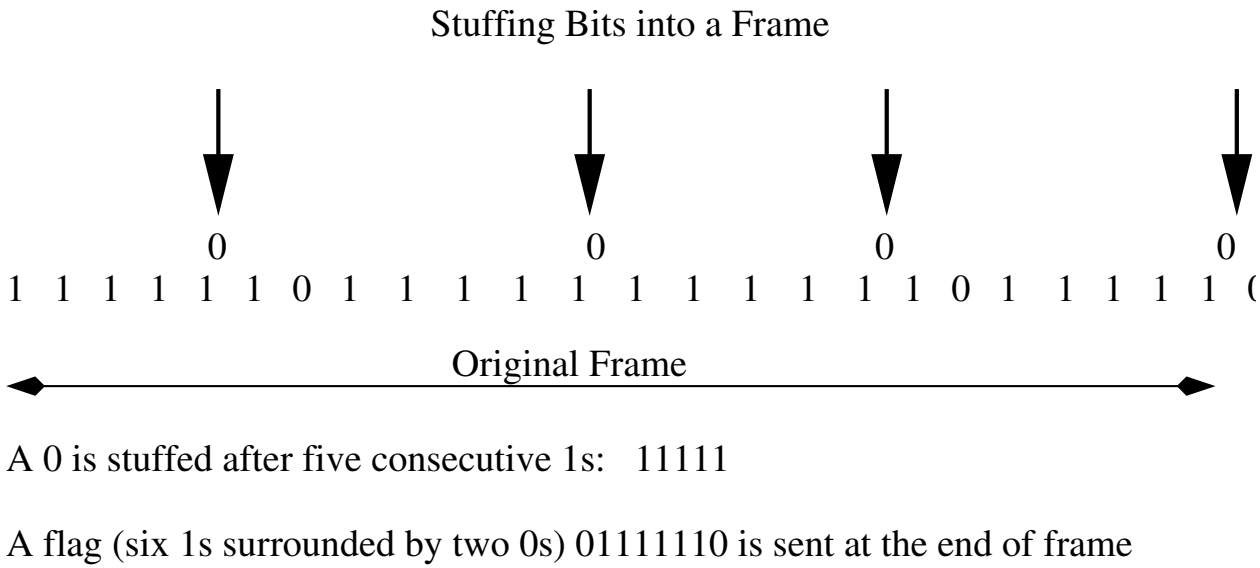
$$000111111100111101000$$

Uses of Bit-stuffing

- Thus, 01^6 when followed by a

$$\begin{cases} 0 & \text{indicates normal termination: it is the flag.} \\ 1 & \text{indicates abnormal termination: abort} \end{cases}$$

- Bit stuffing helps in synchronization because of short length sequences of 1s.
- **Example: Bit-Stuffing**



A 0 is stuffed after five consecutive 1s: 11111

A flag (six 1s surrounded by two 0s) 01111110 is sent at the end of frame

On Bit Stuffing

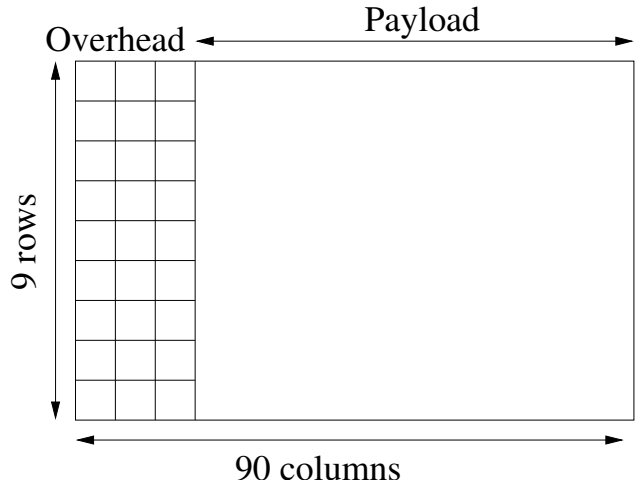
- With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern.
- If the receiver loses track of where it is, all it has to do is scan the input for flag sequences, since they can only occur at frame boundaries and never within the data.
- Bit-stuffing is only applicable to networks in which the encoding on the physical medium contains some redundancy.
  - Some LANs encode 1 bit of data by using 2 physical bits.
  - A 1 bit is a high-low pair and a 0 bit is a low-high pair.
  - This means that every data bit has a transition in the middle, making it easy for the receiver to locate the bit boundaries.
  - The combinations high-high and low-low are not used for data but are used for delimiting frames in some protocols.

Clock-Based Framing (3/4)

- A series of repetitive pulses are used to maintain a constant bit rate and keep the digital bits aligned in the data stream.
- Continuous stream of fixed-length frames
- Clocks must remain synchronized
- No bit or byte stuffing
- Exemplified by SONET (Synchronous Optical Network) Standard.

SONET

- Standard has specific frame sizes.



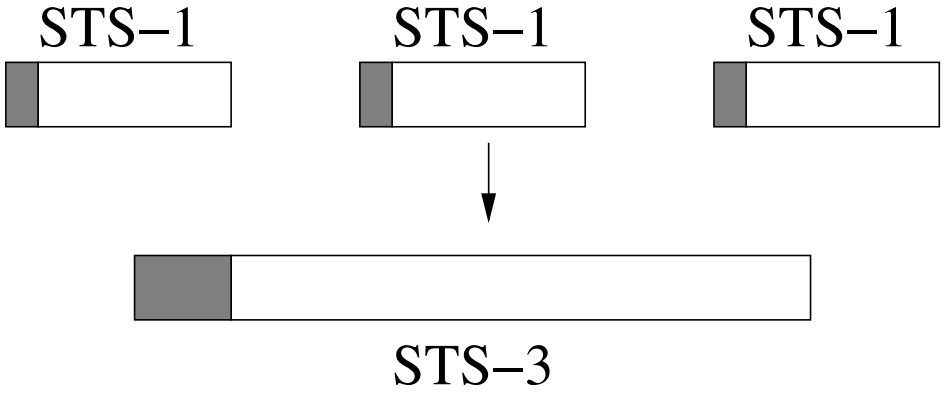
- Frame represented by nine rows, of ninety bytes each, with the first three bytes of each row overhead, i.e. total 810 bytes long.
- First 2 bytes of frame contain special bit pattern to help receiver determine start of frame.
  - Since this bit pattern may occur as part of payload receiver looks for special bit pattern every 810 bytes.

SONET Multiplexing

- SONET rates from STS-1 to STS-48: STS-n means n times STS-1.

SONET	STS-1	STS-3	STS-12	STS-24	STS-48
Speed Mbps	51.84	155.25	622.08	1,244.16	2,488.32

- In multiplexing frames are concatenated to form new frames at different rates.

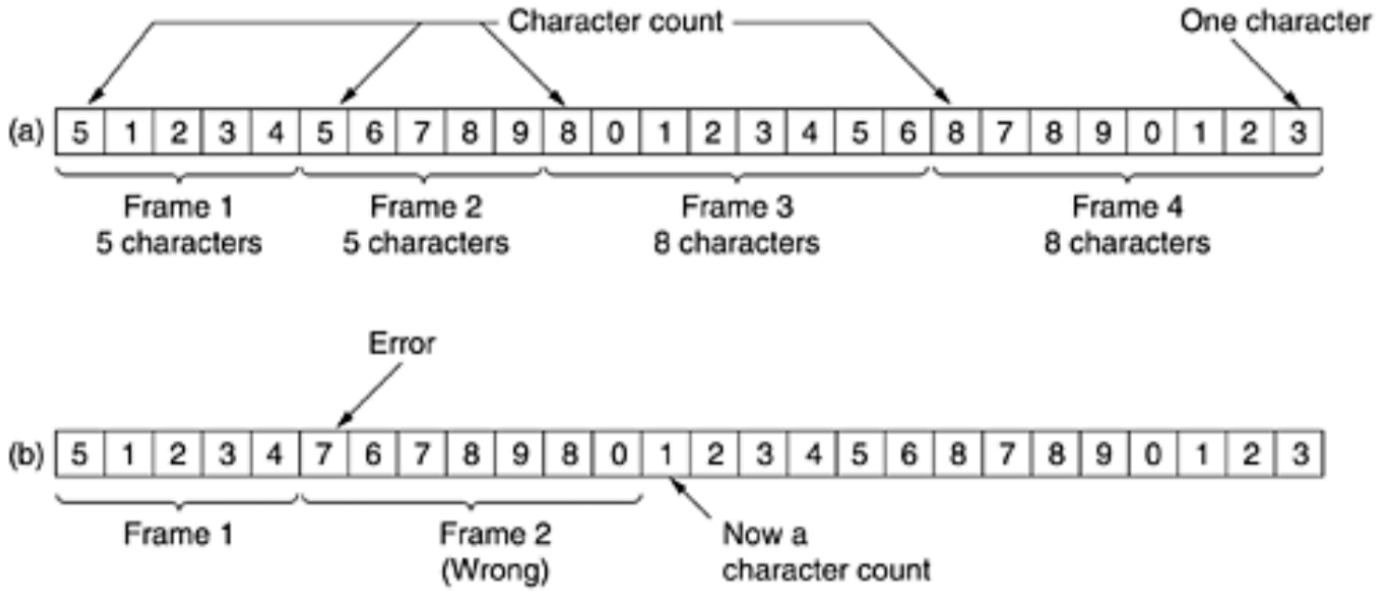


Length Fields (4/4)

- A special field is required.
- The field specifies the length (in bits) of data following the header.
- To specify precisely the length of n bits of data you must specify n!
- For that you need $\lceil \log n \rceil$ bits.

Length Fields: Example

- Uses a field in the header to specify the number of characters in the frame.



- When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is.

Length Fields: Example

- This technique is shown for four frames of sizes $5, 5, 8$, and 8 characters, respectively.
- The count can be garbled by a transmission error.
  - For example, if the character count of 5 in the second frame becomes a 7, the destination will get out of synchronization and will be unable to locate the start of the next frame.

Length Fields: Example

- Some kind of error control is required!
- Even if the checksum is incorrect so the destination knows that the frame is bad, it still has no way of telling where the next frame starts.
- Sending a frame back to the source asking for a retransmission does not help either, since the destination does not know how many characters to skip over to get to the start of the retransmission.
- For this reason, the character count method is rarely used anymore.

Exercises[a]

1. List a few specific functions that the data link layer has to carry out.
2. Could you base the start and end of a frame on timing? On what types of networks would this work best?
3. How do we make sure all frames are eventually delivered to the network layer at the destination and in the proper order?
4. The Hamming distance between two strings of the same length is the number of positions on which their bits differ. What is the Hamming distance of the following sequences: $0000000000, 0000011111, 1111100000, 1111111111$.
5. ($\star$) Interestingly enough bit stuffing has the same performance as length of fields in that in expectation at $\log n$ bits will be added.

[a] Not to submit!

Assume that a "random" source generates n bits at random, independently with the uniform distribution,

$$\Pr[\text{output bit} = b] = \begin{cases} 1/2 & \text{if } b = 0 \\ 1/2 & \text{if } b = 1 \end{cases}$$

Show that bit stuffing is optimized for uniformly random inputs when $k = \log n$. For simplicity, assume that you stuff a bit when you find k consecutive 0s. Then generalize it to k consecutive 0s or k consecutive 1s