Data Link Layer

Kevin Scrivnor

COMP 429

Spring 2023

The data link layer provides

- a service just like every layer

- *“efficient communication of frames between adjacent machines”*

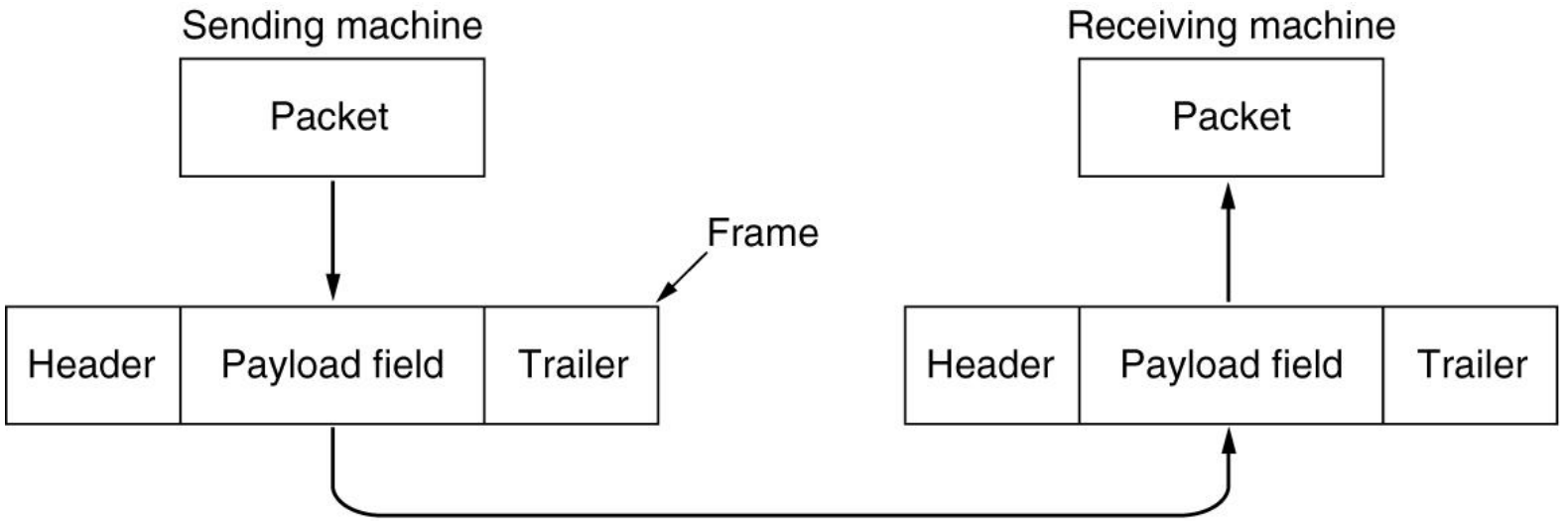
- For instance, your cable modem and the provider or,
- Between routers within an ISP
- Between NASA and Voyager I
- And so on

It might seem simple, but there is plenty to go wrong

- The data link layer will use services provided by the physical layer in order to reach its goals
  - Higher layers always take advantage of what is done below them

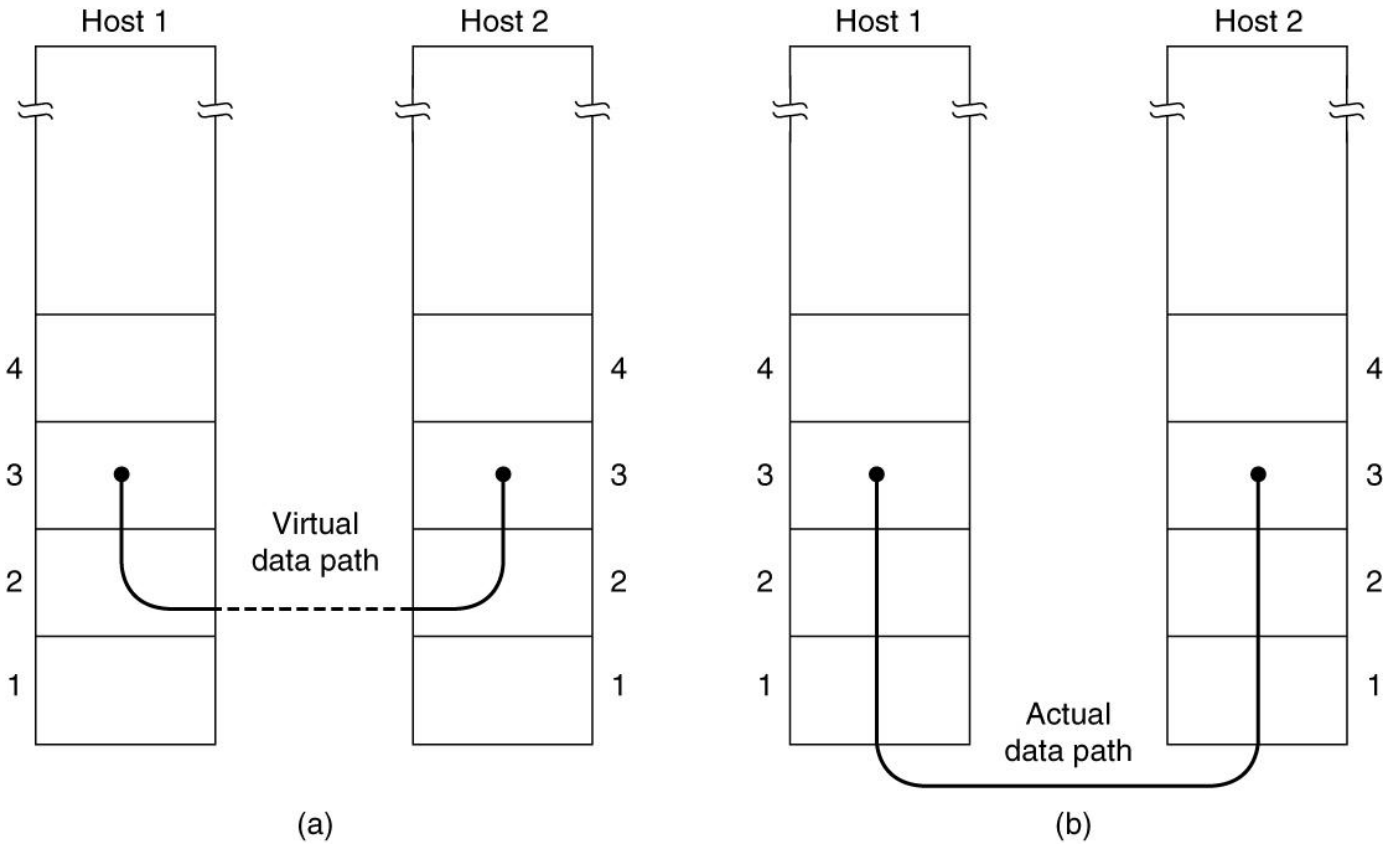
- Goals of the data link layer:
  - Provide a **service** interface to IP (networking layer)
  - **Framing** sequences of bits/bytes as self contained segments
  - Detect and/or correct **errors**
  - Regular the **flow** of data
    - (we do not discuss flow control just yet)

The network layer sends packets
The data link layer turns them into frames



When working with protocols, we tend to think about one layer talking directly to another

- We call this the *virtual path* rather than the *actual path*
- It's easier to think about two processes (programs) directly talking to each other



Services vary from protocol to protocol

- There are typically three types of possible *services*
  - Unacknowledged connectionless service
    - Unack'd connless
  - Acknowledged connectionless service
    - Ack'd connless
  - Acknowledged connection oriented service
    - Ack'd conn
- Our choice depends on the physical layer below
  - Wireless
    - Wants to know if the other side got the info
  - Wired
    - Low loss, recovery can be left to the higher layers
  - Satellite (super wireless)
    - Far away, high loss
  - *Which service suits which physical layer?*

Ack's exist at higher layers, why do it at a lower layer?

- Pure optimization
- Frames are small and therefore problems can be found sooner

- Consider a higher layer sending 20 frames of information
  - The problem will not be discovered until all 20 frames have
  - Arrived, been reconstructed, and then checked

- The lower layer can handle this one frame at a time

Framing

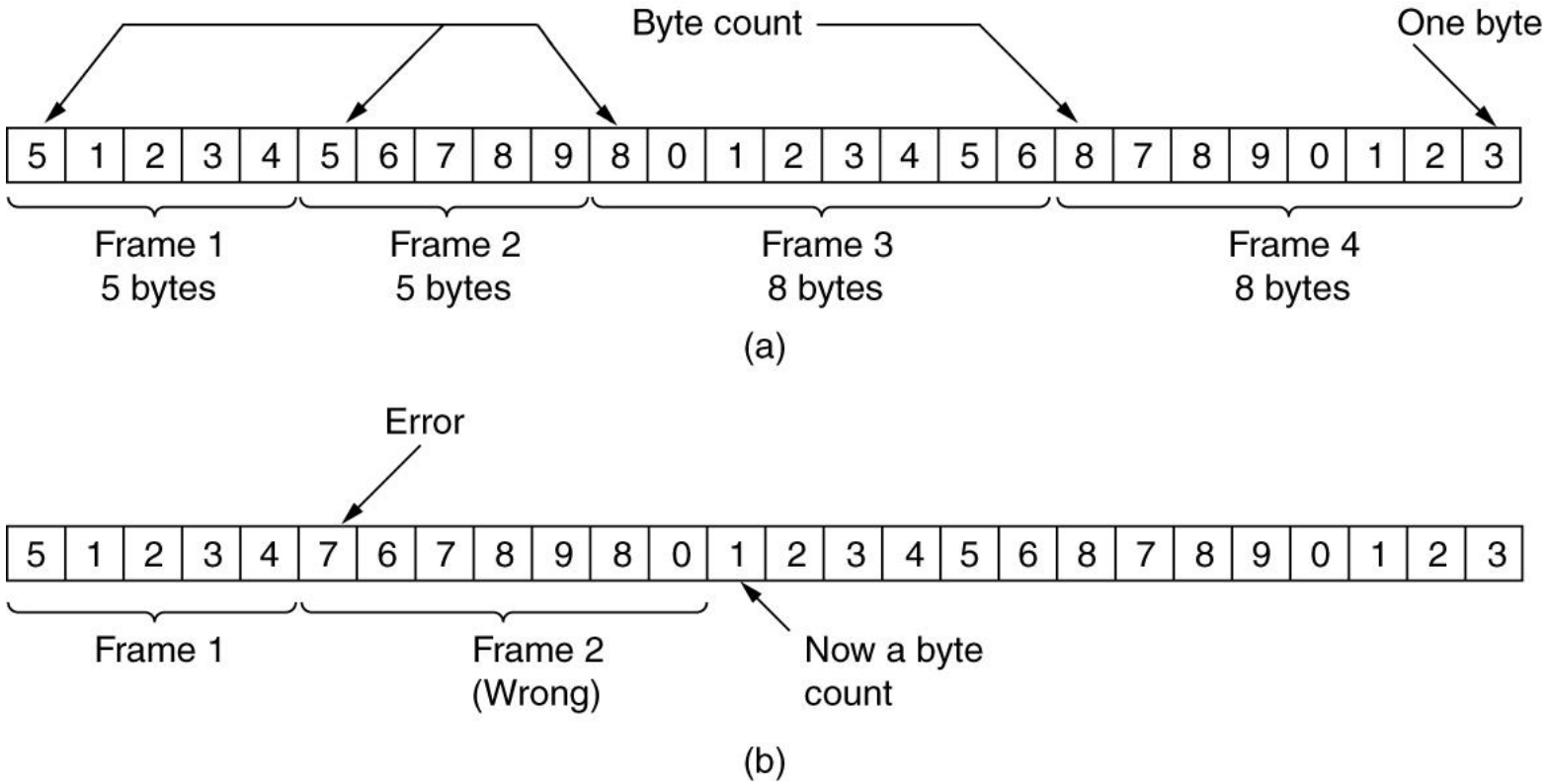
- Break up the bit stream into frames
- Compute a checksum
- Include checksum in the frame when transmitted

- Think a large file broken up into an n-byte length frame, transmitted one at a time
  - With each frame having a header containing information
  - Such as the checksum

- There are a few considerations here

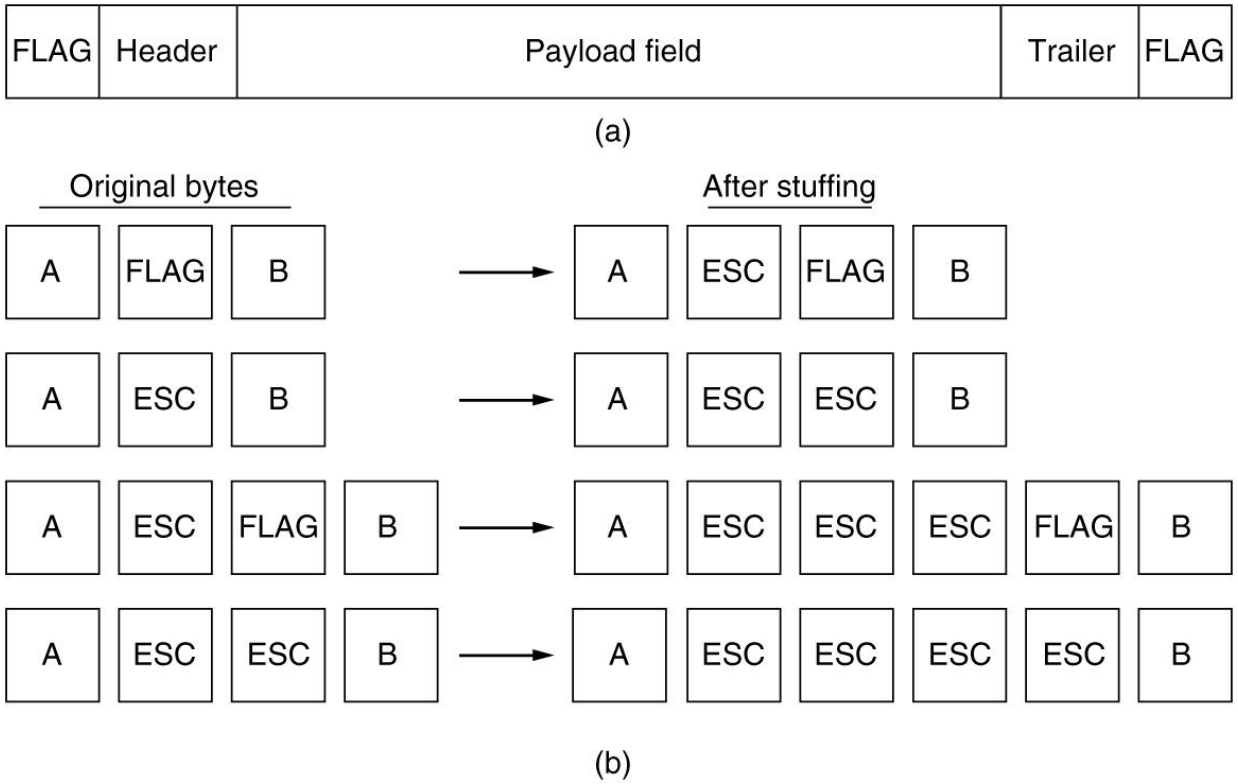
How many bytes?

- We can append the number of bytes that follow to every first byte of the frame.
- What happens if the byte count gets corrupted?



A better approach is to mark the beginning and the end with a flag byte

- Say we use 0x7E (0111 1110) as a flag
- What happens if the flag appears in the data?
- Solution: same idea as printf or when escaping characters in a regex



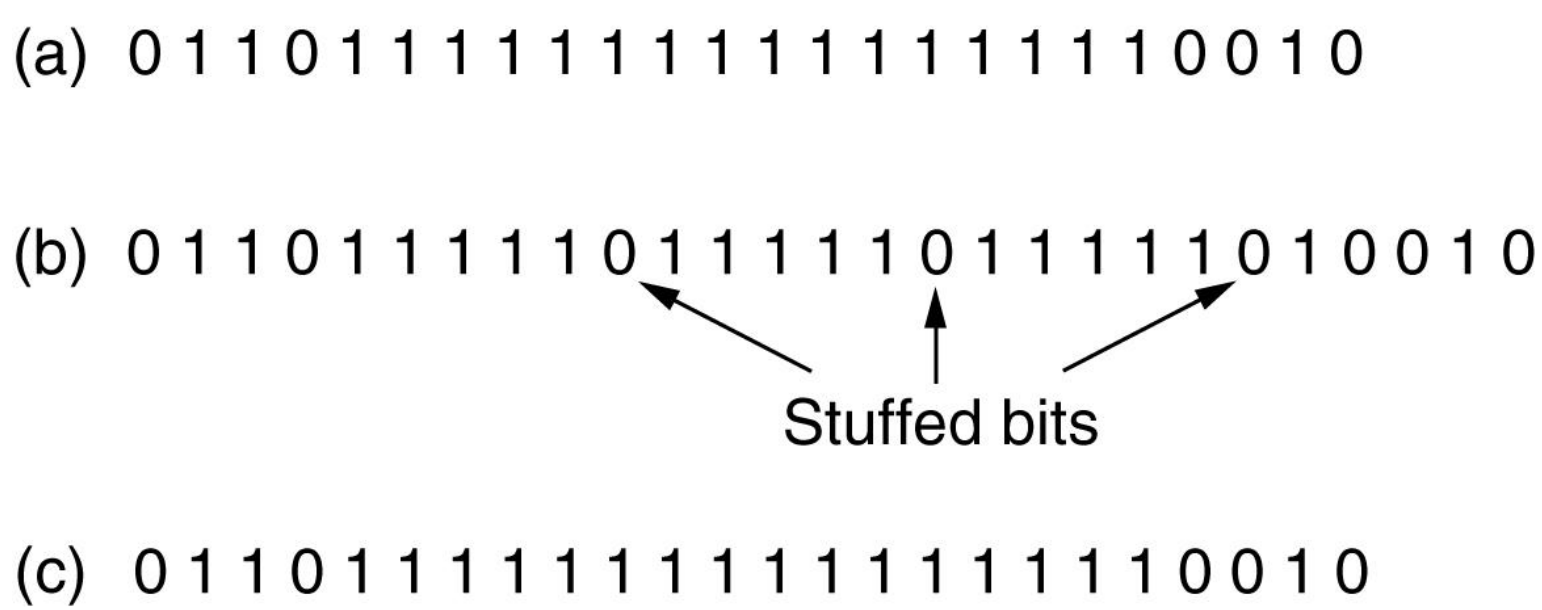
Last slide is a bit of an oversimplification

- Some protocols use a technique called bit stuffing
- For example, if your flag is 0x7E you have six 1's in a row
  - 0111 1110
- If the data ever contains five 1's in a row, a 0 is added immediately after
  - 11111 becomes 111110 (that way six 1's will never occur in the transaction UNLESS it is the flag)
- The other side simply deletes a 0 after five 1's to "unstuff" the data

- USB uses this technique

Bit stuffing with a picture

a) Original data
b) Stuffed data (add a 0 after five 1's)
c) Unstuffed data (remove 0 after five 1's)



We have frames, onto the next problem

- We want all packets to arrive and arrive in order
- In this case, we want the seder to be aware of what is happening on the other side

- Control frames are positive of negative acknowledgements
  - ack/nack

- Question: what happens when a control frame is lost?

We introduce a timer

- Rather than waiting forever for a control message that will never come, we start a timer
- If we get an ack or nack, we cancel the timer

- This can create problems
  - Usually, many frames are being sent at once (we do not stop and wait)
    - Send 1, wait for timer/ack/nack, send 2, wait… (this is bad!)
  - *As with everything in networking, a solution just creates a problem for someone else*

- Question: how can we differentiate timers/acks for the individual frames?

Each frame can have a sequence number

- So an ack/nack is identifying exactly which frame number it has received
- Additionally, *data can still be transmitted* if say, packet 2 is having trouble, we can still send 3-10 and so on.
- TCP and other layers apply this same logic to their flow control
- There's still another issue, if we transmit packet 2 and finally get an ack for packet 2, is it an ack for the original transmission or the retransmission? Can we know for sure?

We will also worry about the flow control

- in way more detail…

- Idea: what happens when a mobile phone requests a web page from a very powerful server?
  - Does the phone get blasted into oblivion with data?

- Two approaches
  - Feedback-based flow control
    - Receiving end transmits information back to sender
    - Or at least provides an update on how it is doing
  - Rate-based flow control
    - Built in mechanism
    - Limits the rate without any feedback

Sometimes we want to know if an error exists
Sometimes we want to know and correct it

- In a wired setting, data travels fast enough that a retransmit will be optimal
  - In that case, we want **error detection**

- In wireless, it would be nice if we could fix a few errors with *math*
  - In this case, we want **error correction**

- This means we will add more information to our headers, either
  - Error correcting code (wireless, high error rate scenarios)
  - Error detecting code (fibre, ethernet, high quality copper, low error rate)

Techniques for Error-Correcting Codes

- All techniques add extra bits to the frame

- We define a frame as being composed of
  - *m* data bits
  - *r* check bits
  - where $n = m + r$
    - *n* is referred to as an n-bit codeword

- We call m/n the code rate
  - The fraction of the code word that is the data
  - This fraction is based on the quality of the channel
  - Might be ½ on an *unreliable channel*
  - Or very close to 1 on a *reliable channel*

There are four common techniques

- which will be covered in less and less detail as we go down the list

- Hamming Codes
  - COMP 162 anyone?
- Binary Convolutional Codes
  - Voyager I uses this technique
- Reed-Solomon Codes
- Low-Density Parity Check Codes

What is an error?

- Consider two codewords (that are supposed to be the same)

- 1000 1001
- 1011 0001 XOR
- =========
- 0011 1000

- If you XOR them, you get 0011 1000
- This means that the codewords differ by 3 bits
- In this case we would say the hamming distance, *d*, is 3

In general, if two codewords are d apart, then we require d single bit errors to correct

- What this means is mathematically we are limited by the formula

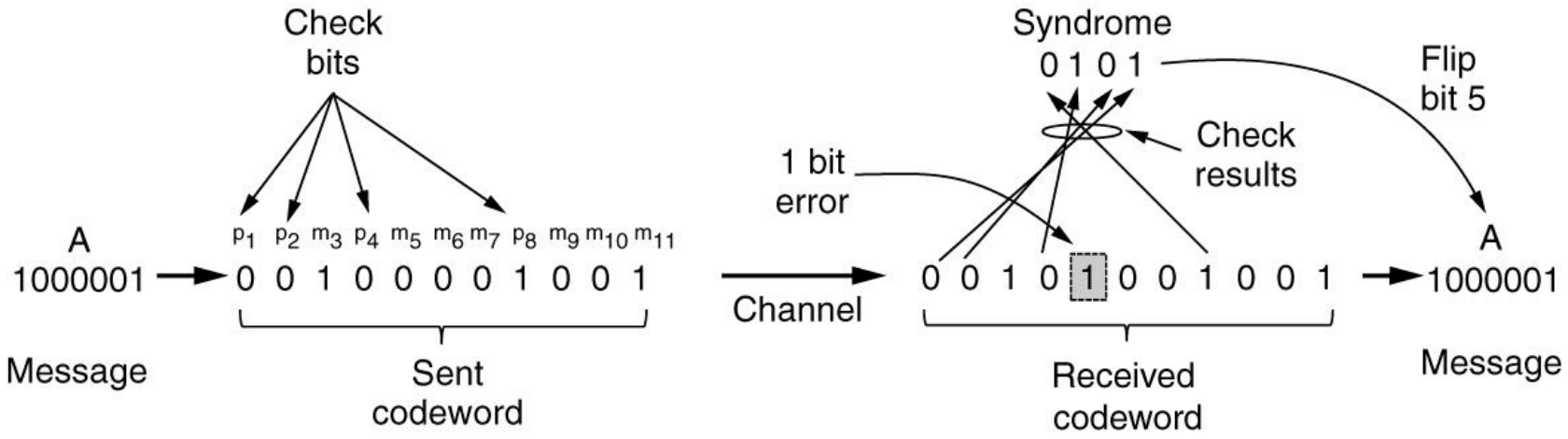
  - $(m + r + 1) <= 2^r$

  - For example, if we had 7 bits of data, then we need 4 check bits to correct single bit errors

  - $(7 + 4 + 1) = 12$
  - $2^4 = 16$
  - $12 <= 16$

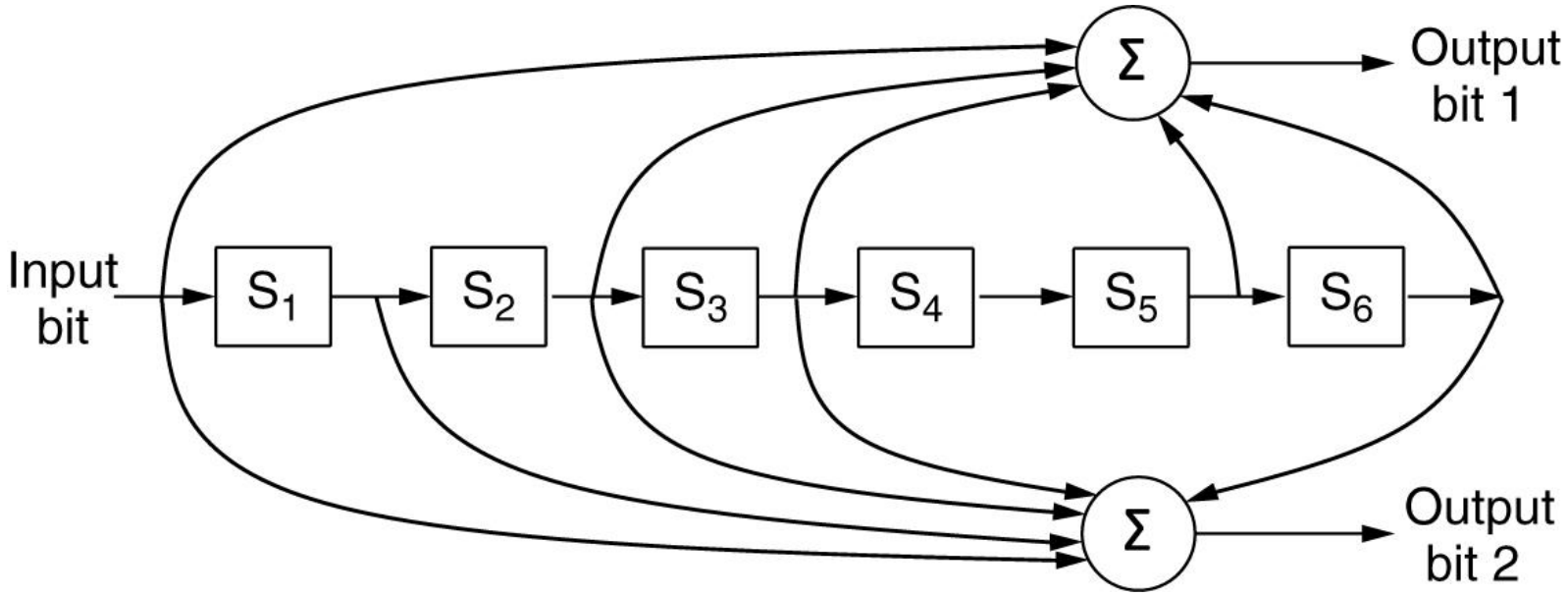
Board Demo of Hamming Codes

- Send the character ‘A’ with strategically placed check bits



Convolutional codes are not blocked based

- Where hamming codes are blocks of data (7 bits in our previous example)
- Convolutional codes process a sequence of input bits and generates a sequence of output bits
  - Used today in GSM, Satellite, 802.11
  - Question: what is the code rate here?



Decoding requires special techniques

- Soft-decision decoding
  - Say -1V mapped to a 0
  - And +1V mapped to a 1
  - If you receive, 0.9V, it is likely a 1
  - If you receive, -0.1V it is likely a 0

- See *Viterbi Algorithm*

- Further decoding is not discussed

Reed-Solomon Coding

- Used very widely because of strong error correcting properties
  - DSL, data over cable, CD, DVD, and Blu-Ray uses this technology

- When *2t* symbols are added, it can correct up to *t* errors
  - In the case of m = 233 and r = 32
  - 32 = 2*16 symbols as checkbits
  - Therefore 16 errors can be fixed

- In reality, a symbol is typically 8 bits, therefore up to 128 bits can be corrected
- Which is how your scratched CD's still play

LDPC – Low Density Parity Check

- Developed in 1962 in a PhD thesis
- Ignored and then reinvented in 1995 once computers were powerful enough to perform the calculations

- Works with very large block sizes and out performs the previous three methods discussed

- Standard for 802.11, streaming, and others. Likely will be everywhere in the future

Onto error detecting codes

- When errors are rare, like on fibre or copper, it is simpler to detect that an error has occurred somewhere

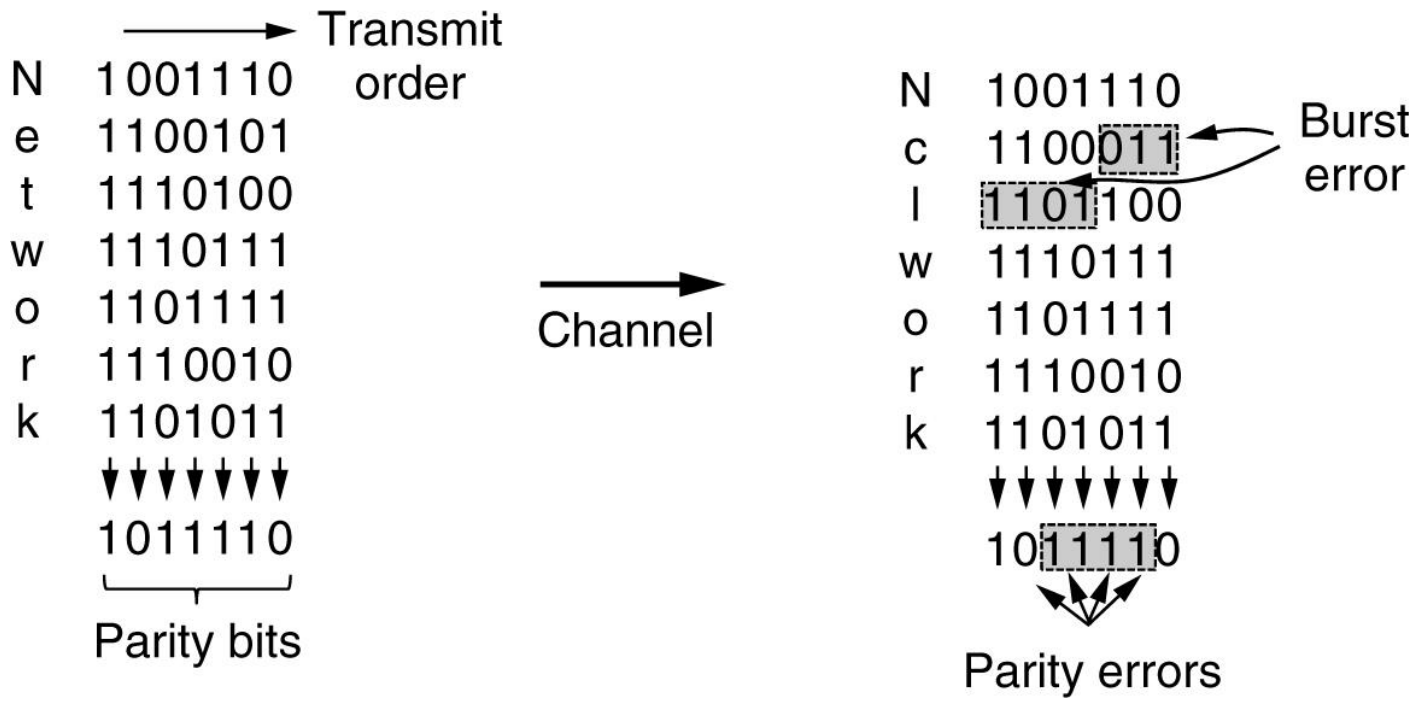
- Parity is the simplest technique
  - Add a bit to the end to make the number of 1's even
  - Data: 100111
  - Data + Parity bit: 100111**0**
  - Data: 110010
  - Data + Parity bit: 110010**1**

- This is very efficient in terms of bits required to determine if there was an error

Hamming vs Parity in our simple case

- If we have 1000 bit block sizes, then in order to check for errors we have: 1000 <= 2^10 or 1000 <= 1024
  - Therefore we need 10 check bits
  - That means that we need 10,000 extra bits total for a megabit of data
- With parity, we only need one check bit per block of 1000 bits
  - Occasionally we will have a retransmit, so an extra 1000 bits added
  - For a megabit, that translates to maybe 2001 extra bits
- Parity works well and but can only handle single bit errors
  - Long burst errors tend to cause major problems

Mostly we just want to know what a parity is



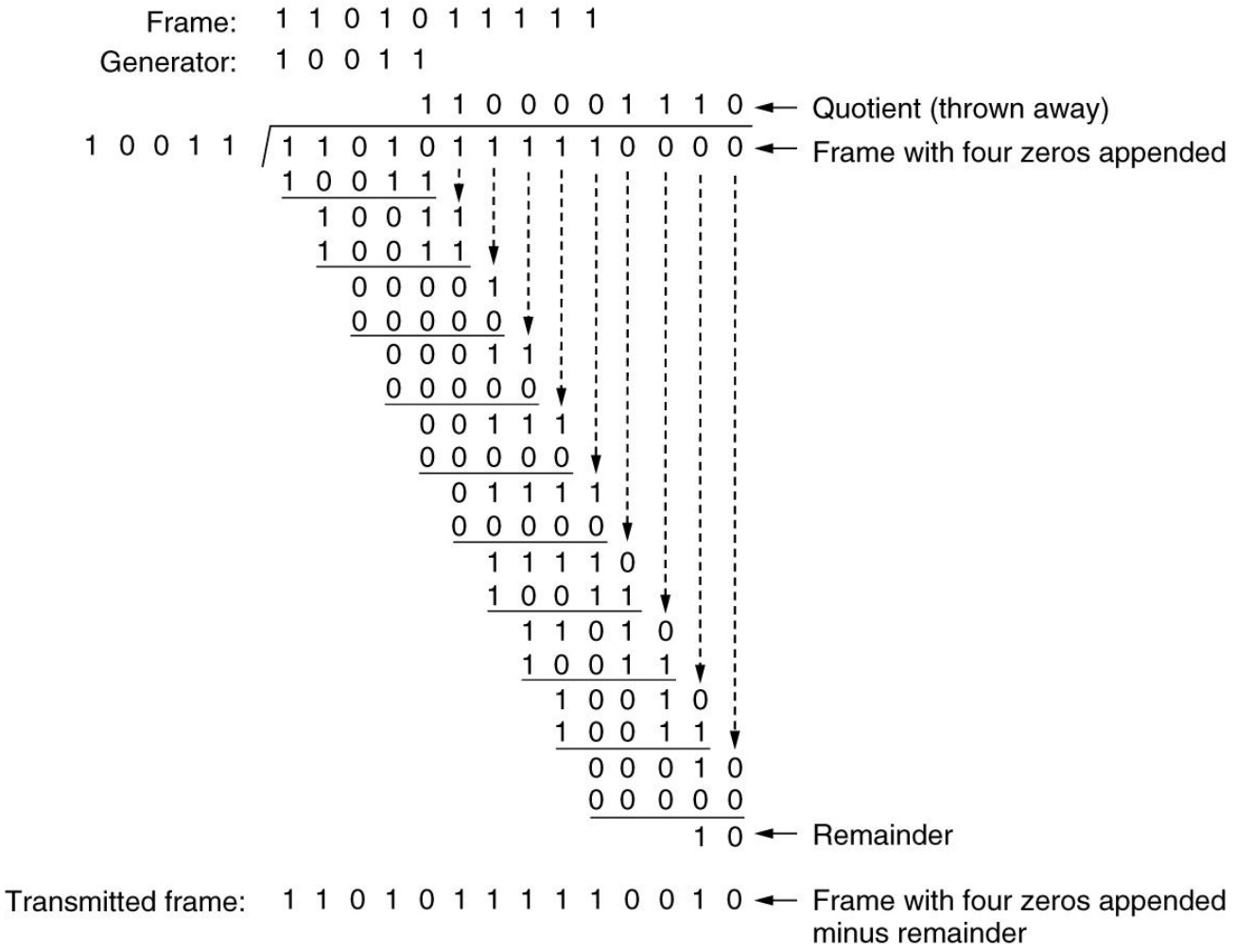
Checksum is a group of checkbits associated with a message

- Example: IPv4 Checksum (1988)
  - Sum up message parts in 16-bit words
  - Add all of them up
  - Add in any carry bits
  - Take the 1's complement (flip all the bits)

- Provides weak protection for these reasons:
  - Simple sum
    - Cannot detect reordering of data
    - Adding 0's doesn't change the sum
  - Assumed that all data was random
    - Data transmitted across a network isn't all that random after all

Lastly we have CRC (Cyclic Redundancy Check)

- The idea is if continuously divide, there should be no remainder
- The most widespread use at the link layer



Calculating an IPv4 Checksum

- Recall that IP calculates its checksum using 16-bit words (4 hex numbers) at a time from data within the packet

- Download the activity packet here: http://429.scrivnor.cikeys.com/captures/Lecture04-Activity.pcapng

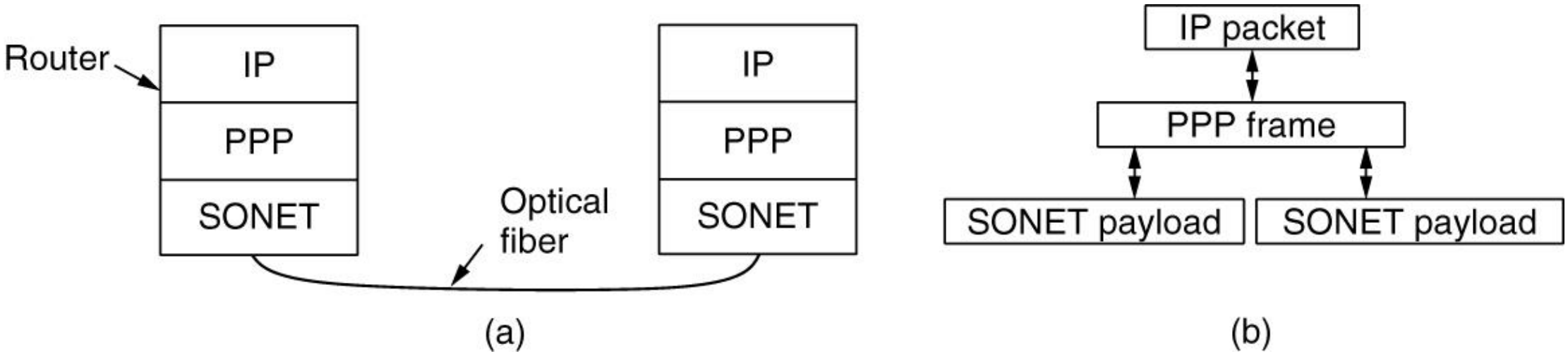
- Instructions
  - Add up all the header information in the IPv4 Packet (EXCEPT the checksum itself)
  - Add the carry bits back in (if you have 3 AF05) then (AF05 + 0003 = AF08)
  - Take the 1's complement
  - Does it match the IPv4 Checksum?

Data Link Protocols in Practice

- Point to Point lines
  - SONET optical fibre between WAN
    - Connects routers to locations within ISP network
  - ADSL links
    - Telephone network at the edge of the Internet (ISP to your home)
  - DOCSIS links
    - Local loop of a cable network

- All use the standard protocol PPP (Point to Point Protocol) framing mechanism

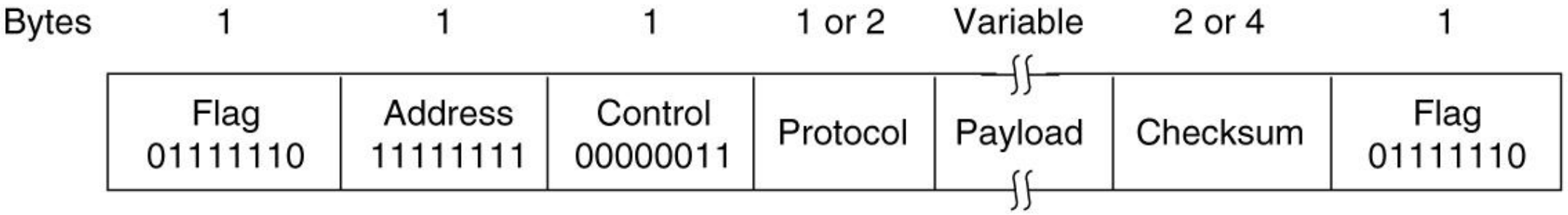
Packet over SONET



- Three main features provided
  - Framing method delimits the end of the frame/beginning of another
  - LCP Link Control Protocol
    - Communication for bringing lines up, testing, negotiating options, shutting down
  - Negotiate network layer options in some generic way
    - So that the layer is independent of layers above it

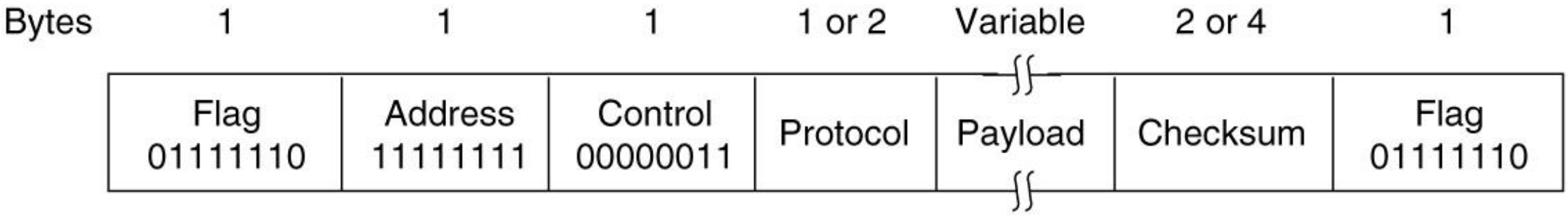
PPP is byte oriented

- Uses byte stuffing, the number of bytes is always a whole number
- Most of the time used in unnumbered mode (no sequence numbers)
  - ISP provides connectionless, unacknowledged data service
- The standard flag byte is 0xFE (0111 1110)
  - Escape byte is 0x7D followed by the data XOR'd with 0x20
  - This way 0xFE will never appear within the payload
  - Question: why is that useful?
- *NOTE: the flag only appears once between frames*



Looking at the frame format in detail

- Flag is 0111 1110 (0xFE)
- Address is always all 1’s
  - Idea is that all stations should accept the frame (which is generally the case)
- Control is 0x11
  - Indicates unnumbered (no sequence numbers)
- Protocol usually 1 byte
  - Allows for up to 256 protocols, can be negotiated to 2 bytes
  - (There are not that many protocols needed)
- Checksum is defaulted to 2 bytes, uses the CRC method



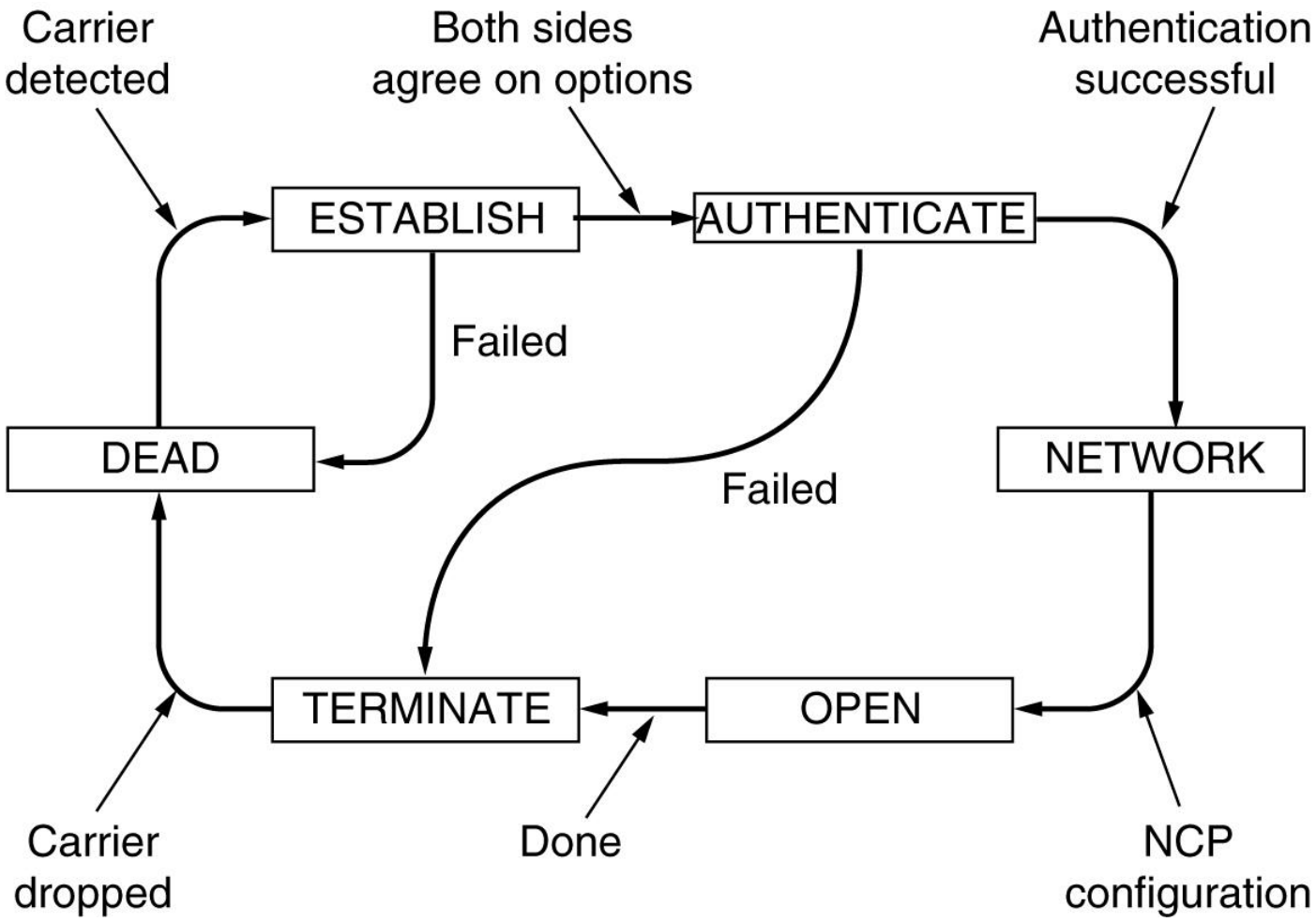
0’s are bad for checksums

- Recall with IP we added bits up
- All 0’s do not change the sum at all

- A malicious/annoying user could put all 0’s into the network

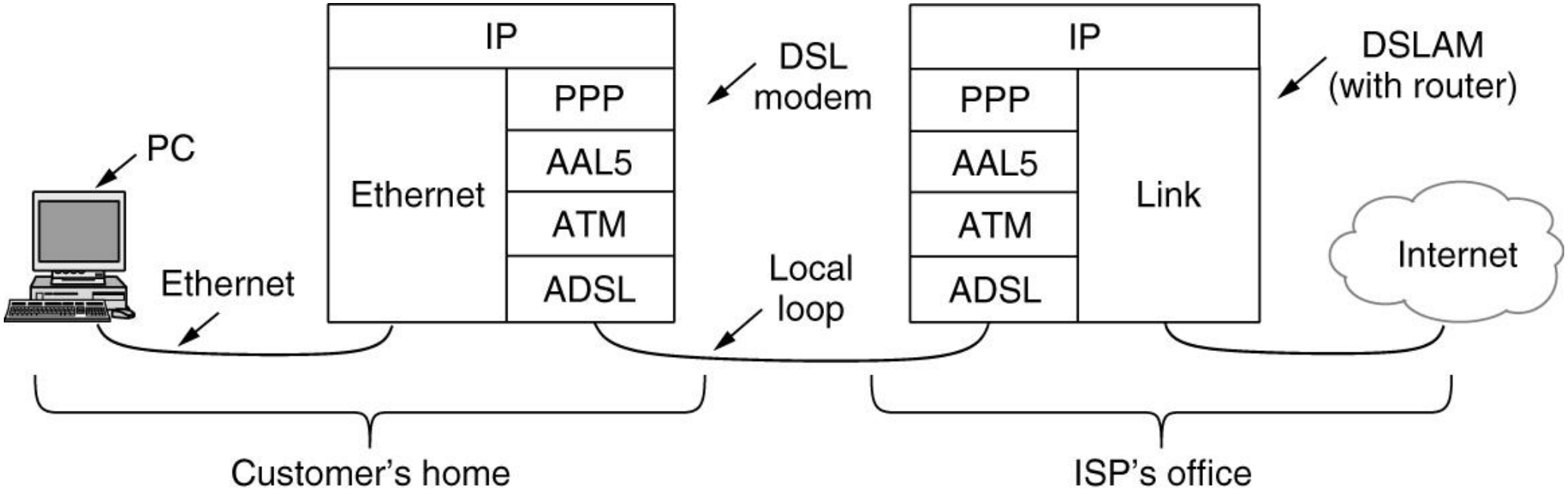
- SONET XOR’s data with a pseudorandom number so that there is guaranteed to be some 1’s in the payload

State Diagram for a SONET PPP Link
Each point needs to be aware of the other



ADSL (Asymmetric Digital Subscriber Loop)

- Operates on the old telephone service
  - Most common scenario is a DSL Modem at home that talks to a DSLAM (dee-slam) at your ISP
  - DSLAM (DSL Access Multiplexer)
- DSL also uses PPP, but add some layers of its own (and removes some from PPP)



ATM had big promises and nothing happened

- ATM is Asynchronous Transfer Mode
  - Designed in the 90’s
  - Promised to connect voice, data, cable TV, telegraph, carrier pigeon, tincan + string, and so on
- Doomed by politics
- Think back to the picture about money and development

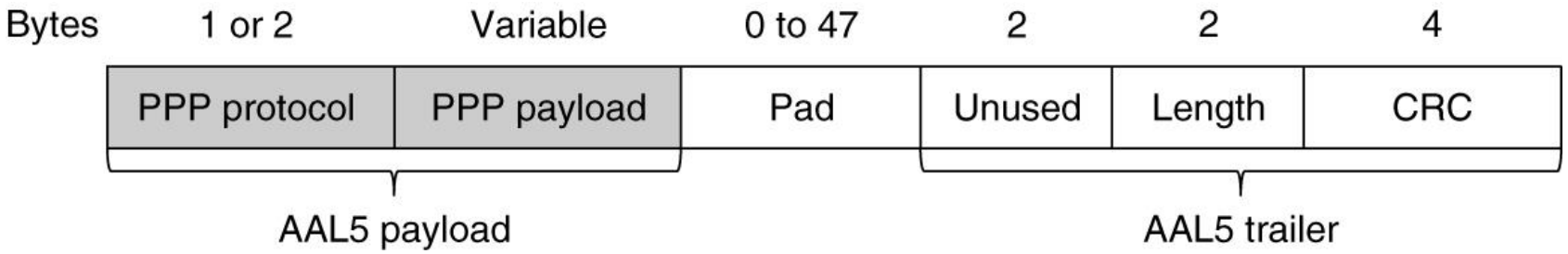
ATM transmits fixed length cells

- Cells only transmit when information is present
- A virtual circuit is established between two points
  - An identifier (address sort of) is used to forward cells along a path of connected devices
- The cells are 53 bytes long (what?)
  - 48 payload
  - 5 headers

- Why 48?
  - Europe wanted 32
  - America wanted bigger obviously at 64
  - 48 is right in between
  - Example of Politics actually ruining a protocol

The AAL5 ATM Adapter Layer 5

- Provides the segmentation and reassembly of cells on the link layer
- Note that most of PPP isn't included because ATM does
  - Framing
  - Error correction
  - For itself



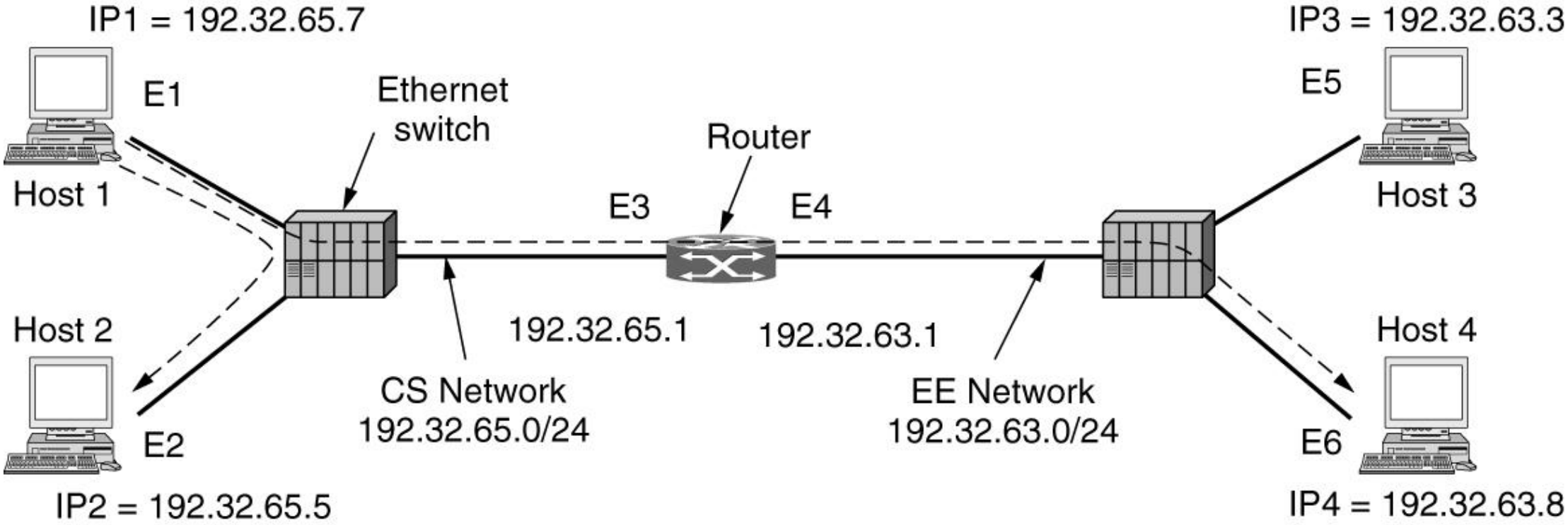
Address Resolution Protocol

- The Link layer is (layer 2) is below the Network layer (layer 3)
  - Which means at this level, there are no internet addresses (think IP addresses)

- Every network interface card (NIC) comes with a unique 48-bit address called an “Ethernet address”

- However, most communication is network layer based, so how do IP addresses get mapped to Ethernet addresses?

Two switched Ethernet LANs joined by a router



Frame	Source IP	Source Eth.	Destination IP	Destination Eth.
Host 1 to 2, on CS net	IP1	E1	IP2	E2
Host 1 to 4, on CS net	IP1	E1	IP4	E3
Host 1 to 4, on EE net	IP1	E4	IP4	E6