COMP2001J Computer Networks

Lecture 3 – Data Link Layer

Dr. Shen WANG (王燊)

shen.wang@ucd.ie

Future labs

- You are highly encouraged to attend **all** lab sessions.
  - So you can get your questions answered immediately.
  - But you **don't have to**, so you can be TAs for labs upstairs ☺
  - The lab notes are designed in the way that you can practice yourself at home.
- The followings are lab sessions (no clash with your other labs as a TA) you HAVE TO attend:
  - Quiz 2: Week 6 (**26th March**)
  - Lab exam (Packet Tracer): Week 9 (**16th April**)
  - Mid-term exam: Week 10 (**23rd April**)
  - Lab exam (Wireshark): Week 15 (**28th May**)

Ready for the next lab

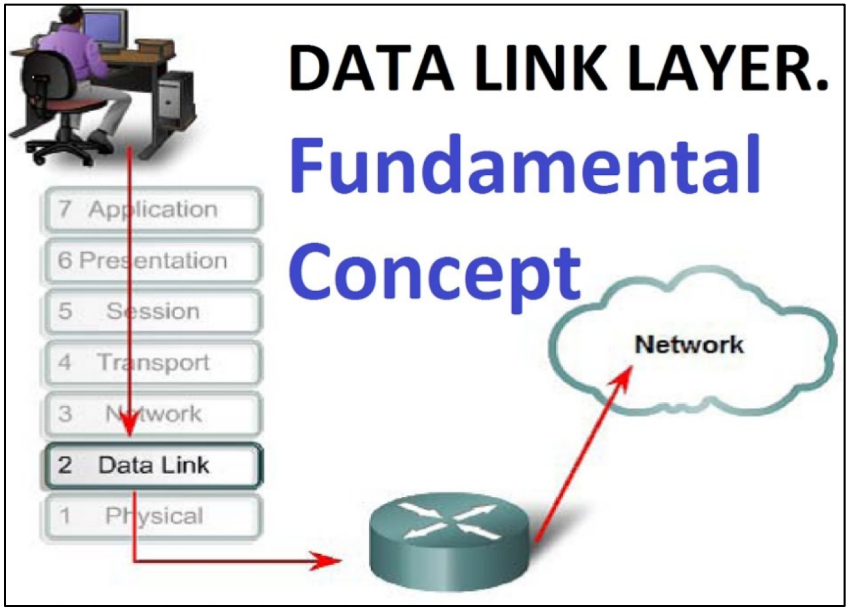
- Download PacketTracer 7.2.1
  - From official website!
  - https://www.netacad.com/courses/packet-tracer
- Register your own account! As you need to login every time when you launch PacketTracer.
- PacketTracer does not have Mac version!
  - contact me for Linux VM version

Outline

- Responsibilities
- Framing
- Error Control
- Network Devices

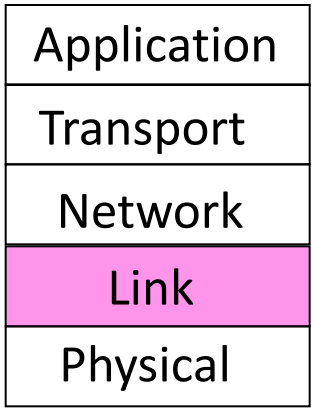
Point-to-Point

Not end-to-end!



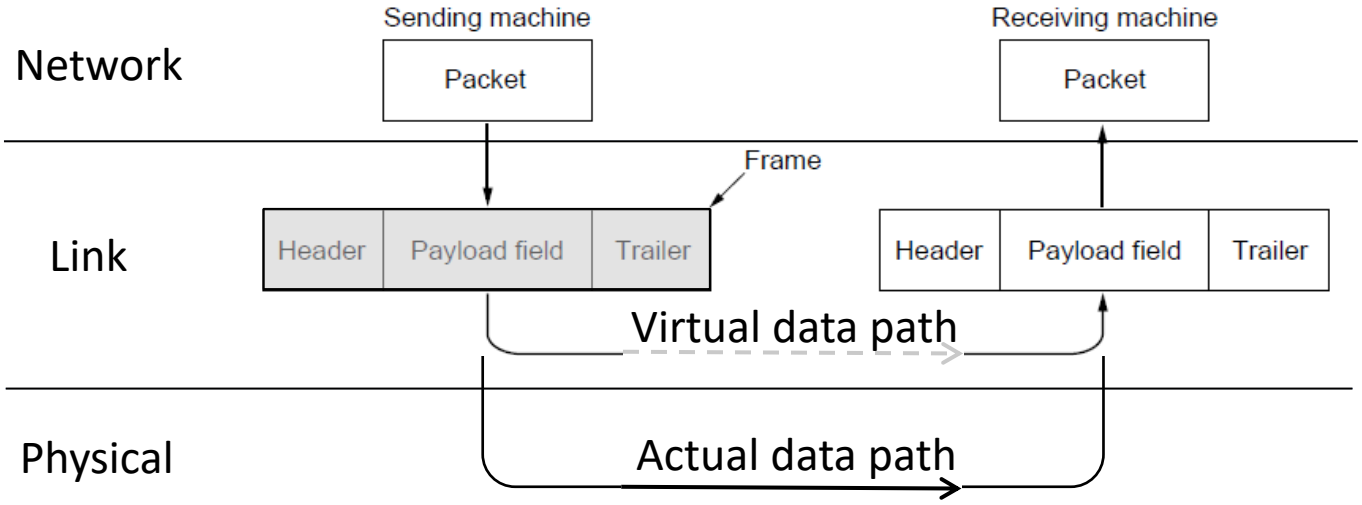
Responsibilities

- The data link layer is responsible for delivering **frames** of information over **a single link.**
  - Framing bit data streams
  - Handles transmission errors
  - Regulates the flow of data (week 4)
  - Medium access control (MAC layer, a sub-layer of the data link layer; week 5)



Frames

- Link layer accepts packets from the network layer, and encapsulates them into frames that it sends using the physical layer; reception is the opposite process
  - Frames have additional information added in headers (before the message) and trailers (after the message)



Possible Services

- Unacknowledged connectionless service
  - Frame is sent with no connection / error recovery
  - Ethernet is example
- Acknowledged connectionless service
  - Frame is sent with retransmissions if needed
  - Example is 802.11 (wireless channel is not reliable)
- Acknowledged connection-oriented service
  - Connection is set up; rare
- Note: The unacknowledged connection-oriented service does NOT exist!

Framing

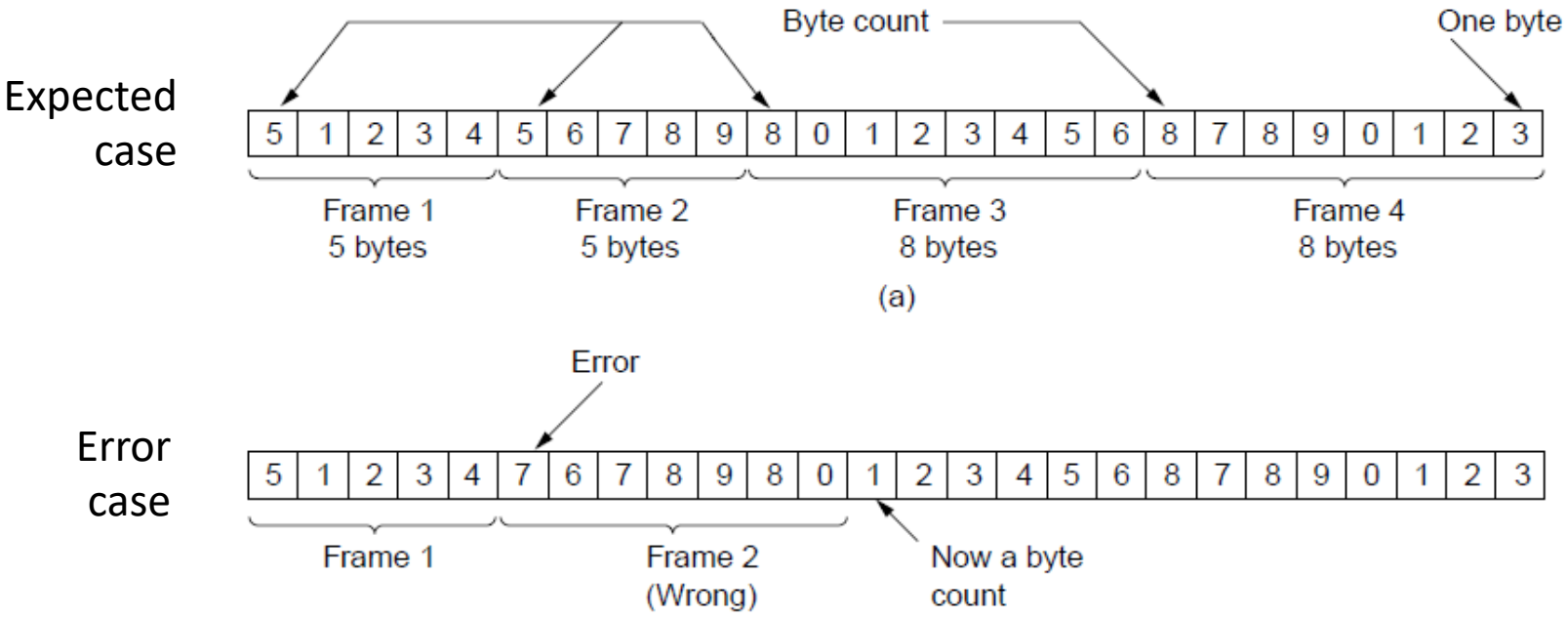
- The main reason for framing bit streams is when the transmission errors occur, instead of re-transmitting **all bit streams** again, one only needs to re-send the **frame** that is incorrectly transmitted.
- A good design must make it easy for a receiver to find the frame boundaries while introducing little overhead.
- “Frame” is a message unit used at the lowest network layer, so it must have both “header” and “trailer”.

Framing Methods

- Byte count
- Flag bytes with byte stuffing
- Flag bits with bit stuffing
- Physical layer coding violations

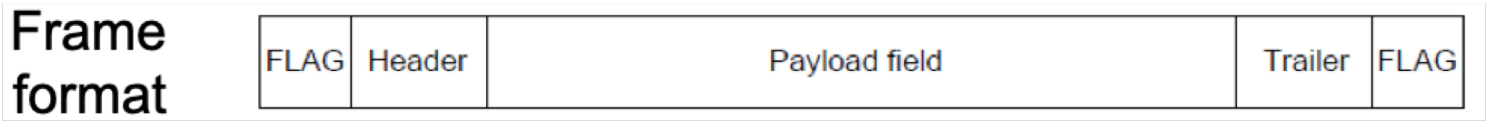
Byte Count

- Frame begins with a count of the number of **bytes** in it
  - Simple, but difficult to resynchronize after an error (one error, all wrong!)



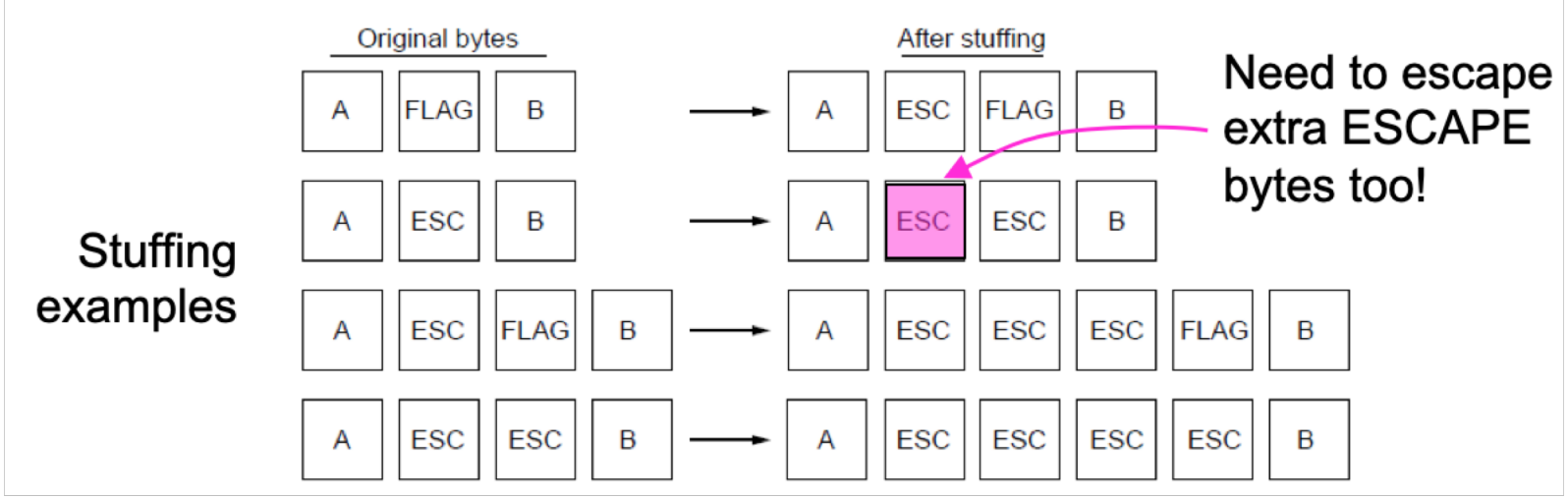
Flag Bytes with Byte Stuffing

- Having each frame start and end with the same special bytes, which is called **flag bytes**. Two consecutive flag bytes indicate the end of one frame and the start of the next.
- Longer overhead, but easy to resynchronize after error
  - if the receiver ever loses synchronization it can just search for two flag bytes to find the end of the current frame and the start of the next frame



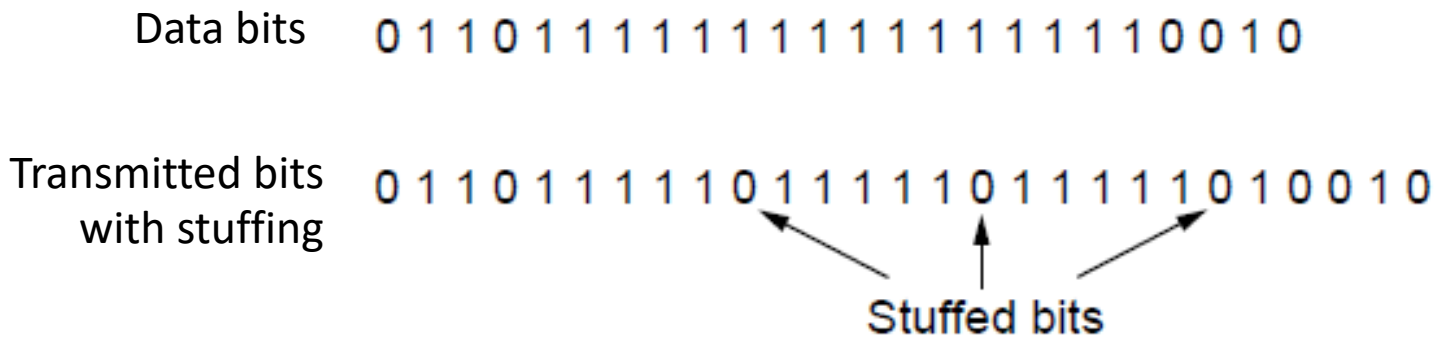
Flag Bytes with Byte Stuffing

- If the data we need to send contains flag bytes, or ESCAPE bytes, they must be stuffed (escaped, also called **byte stuffing**)



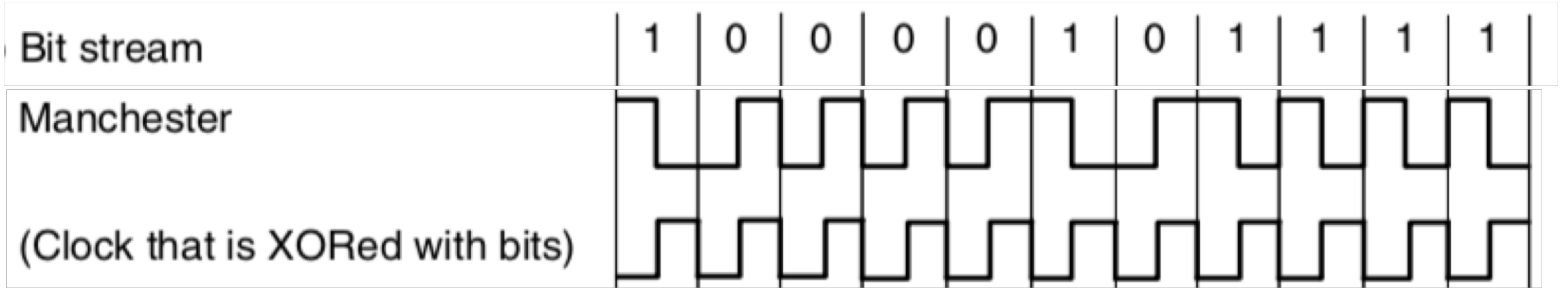
Flag Bits with Bit Stuffing

- Stuffing done at the bit level, less overhead:
  - Between each consecutive frames, add a special frame flag 01111110 (six consecutive 1s)
- To avoid flag bits appears in data bits, for data that within each frame:
  - On transmit, after five 1s in the data, a 0 is added
  - On receive, a 0 after five 1s is deleted



Physical Layer Coding Violations

- Use non-data symbol to indicate frame.
- For example, in IEEE 802 standard, Manchester Encoding considers “high-low” or “low-high”, as “1” or “0”, so “high-high” and “low-low” are violations (not used) that can be used for framing.



Physical Layer Coding Violations

- Coding violations do not need any “stuffing” technology.

- The commonly used framing methods are bit stuffing and physical layer coding violations.

Error Control

- Error Correcting Codes (ECC)
  - Hamming codes
  - Binary convolutional codes
  - Reed-Solomon codes
  - Low-Density Parity Check codes

- Error Detecting Codes (EDC)
  - Parity
  - Checksums
  - Cyclic redundancy codes

Error Control

- When a signal is transmitted a physical media, attenuation, distortion, and noise can cause the signal to be read incorrectly.
- To ensure a certain level of reliability for this data link layer:
  - If the transmission link is stable (e.g. optical fiber, LAN), EDC is applied. So when errors detected at the receiver side, just ask for retransmission
  - If the transmission link is error-prone (e.g. wireless), ECC is applied. As the retransmitted frame might be still incorrect.

Adding Redundancy

- If we add extra information to the frame we can use these to help detect or correct errors
  - Error codes add structured redundancy to data so errors can be either detected, or corrected.

- The problem is to structure the codes to detect as many errors with a **small amount** of extra bits and not too much calculation

Using Error Codes

- **Codeword** consists of D data bits plus R check bits
  - The **code rate**, or simply rate, is the fraction of the codeword that carries information that is not redundant, or *D/(D+R).*
- Called systematic block code
  - Number of check bits depends on the size of the data
  - Check bits are computed based on the data and then added to the end
- When the receiver gets the package it re-computes the check bits based on the data bits
  - If there are no errors the check bits should match

Some questions

- How many errors can it detect?
- How many errors can it correct?
- How many bits redundancy do we need to add?

Error Probability

- The probability of a bit error, p_b, at the Physical transmission layer leads on to a probability of a frame error, p.
- This depends on the number of bits of information in the frame, l, and the number of bits of header in the frame, l' .
- To get a good frame, which is a high (1-p), all the bits have to be good.
  - $(1 – p) = (1 – p_b)^{(l + l')}$
  - $p = 1 – (1 – p_b)^{(l +l')}$
  - Given p_b, greater chance of frame error p, as length (l +l') increases

Valid Codewords

- Within any encoding scheme there are a number of **valid** codewords

- This means that whatever check that is performed must be able to determine if an error has occurred by evaluating the codeword.

Hamming Distance

- The Hamming distance between two codewords W_1 and W_2 is the **number of bits** that must be flipped to change one to the other
  - We write this as $d(W_1, W_2)$

- Example
  - W_1 = 10001001
  - W_2 = 10110001
  - $d(W_1, W_2) = 3$

Hamming Distance

- For any error detection/correction scheme, we can define the minimum Hamming distance **D** (or "minimum distance") of the scheme as the ***smallest number*** of bit errors that changes one **valid** codeword into another.
- Each codeword is made up of **m** data bits and **r** check bits, therefore **n = m + r** total bits
- For data all possible $\mathbf{2^m}$ data strings are usually valid
- But because the check bits are calculated based on the data bits not all $\mathbf{2^n}$ possible codewords are valid

Hamming Distance

- If the method of computing the check bits is known, the list of **all valid codewords** can be calculated and stored at the receiver

- When a word **W** is received the receiver finds the closest valid codeword to **W** using the Hamming distance and takes this as the transmitted codeword

- The **minimum Hamming distance** of a **scheme** is the smallest Hamming distance between all possible pairs in a set of words in the scheme.

Single Parity Check

- Use a single check bit such that the number of 1s in every codeword is **even**
  - If the number of 1s in the data is even we add a 0
  - If the number of 1s in the data is odd we add a 1
- This is called **single parity check**
- If receiver gets a codeword with an odd number of 1's, it knows error(s) occurred
  - can **detect** any odd number of bit errors
  - but: can't tell how many errors, or which bits are in error
  - even worse: any even number of bit errors is *undetectable*

Single Parity Check - Example

- Data to be transmitted: 10110101
  - There are 5 1s in the data so the parity bit is 1
- We transmit: 101101011
  - If receiver gets 101101011 parity check is ok
  - If receiver gets 101100011 parity check fails
  - If receiver gets 101110011 parity check is ok but codeword is incorrect
  - If receiver gets 001100011 parity check is ok but codeword is incorrect
- Data to be transmitted: 10110001
  - There are 4 1s in the data so the parity bit is 0

General Parity Check

- If the minimum distance of an error-handling scheme is D, this scheme can **detect** any combination of **≤ D-1** bit errors and **correct** any combination of **< D/2** bit errors

- Alternatively:
  - If you want to **detect** B bit errors, use a scheme with minimum distance (D) at least B+1;
  - If you want to **correct** B bit errors, use a scheme with minimum distance (D) at least 2B+1

General Parity Check Examples

- Example - Single parity check
  - This scheme has a minimum Hamming distance of D = 2
  - We noted that it could **detect** only any single bit error but **cannot correct** any bit error
    - Detect: ≤ D – 1 = ≤ 1
    - Correct: < D / 2 = < 1

General Parity Check Example

- Suppose we have only 4 valid codewords
  - C_1 = 00000 00000
  - C_2 = 00000 11111
  - C_3 = 11111 00000
  - C_4 = 11111 11111

  - The minimum hamming distance is 5
    - Any combination of ≤ 4 bit errors can be detected
    - Any combination of < 2.5 bit errors can be corrected

General Parity Check Example

- If we transmit 00000 00000 but receive X = 00000 00011, we know it is not a valid codeword and compute the hamming distance between it and all valid code words

- d(W, C1)=2, d(W, C2)=3, d(W, C3)=7, and d(W, C4)=8

- The receiver takes C_1 = 00000 00000 as transmitted codeword ***(errors corrected)***

General Parity Check Example

- If we transmit 00000 00000 but receive X = 00000 00111, we know it is not a valid codeword and compute the hamming distance between it and all valid code words

- d(W, C1)=3, d(W, C2)=2, d(W, C3)=8, and d(W, C4)=7

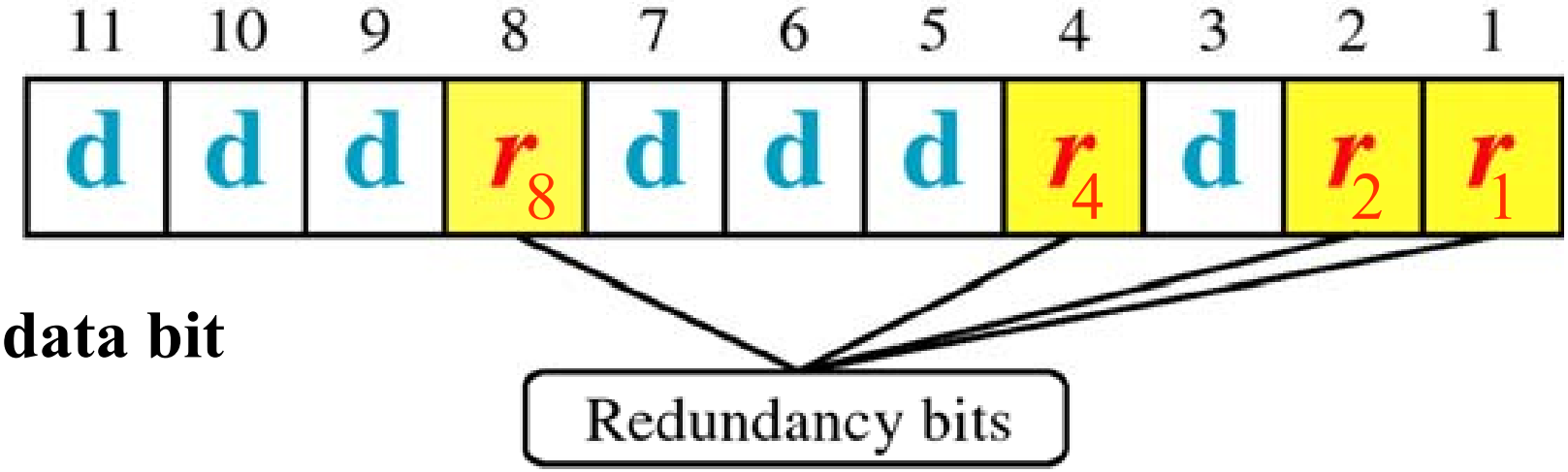
- The receiver takes C_2 = 00000 11111 as transmitted codeword ***(in this case, error correction fails)***

Creating Hamming Codes

- Hamming codes are based on the calculation of the **minimum number of check bits** needed to **correct** any **single bit error**
  - With 7 data bits we need at least 4 check bits
- In **Hamming codes** the bits of the codeword are numbered consecutively, starting with bit 1 at the left end, bit 2 to its immediate right, and so on.
- The bits that are powers of 2 (1, 2, 4, 8, 16, etc.) are **check bits**.

Hamming Code

- Each redundancy bit is the parity bit for a different combination of data bits
- Each data bit may be included in more than one parity check



d: data bit

Hamming Code Calculation

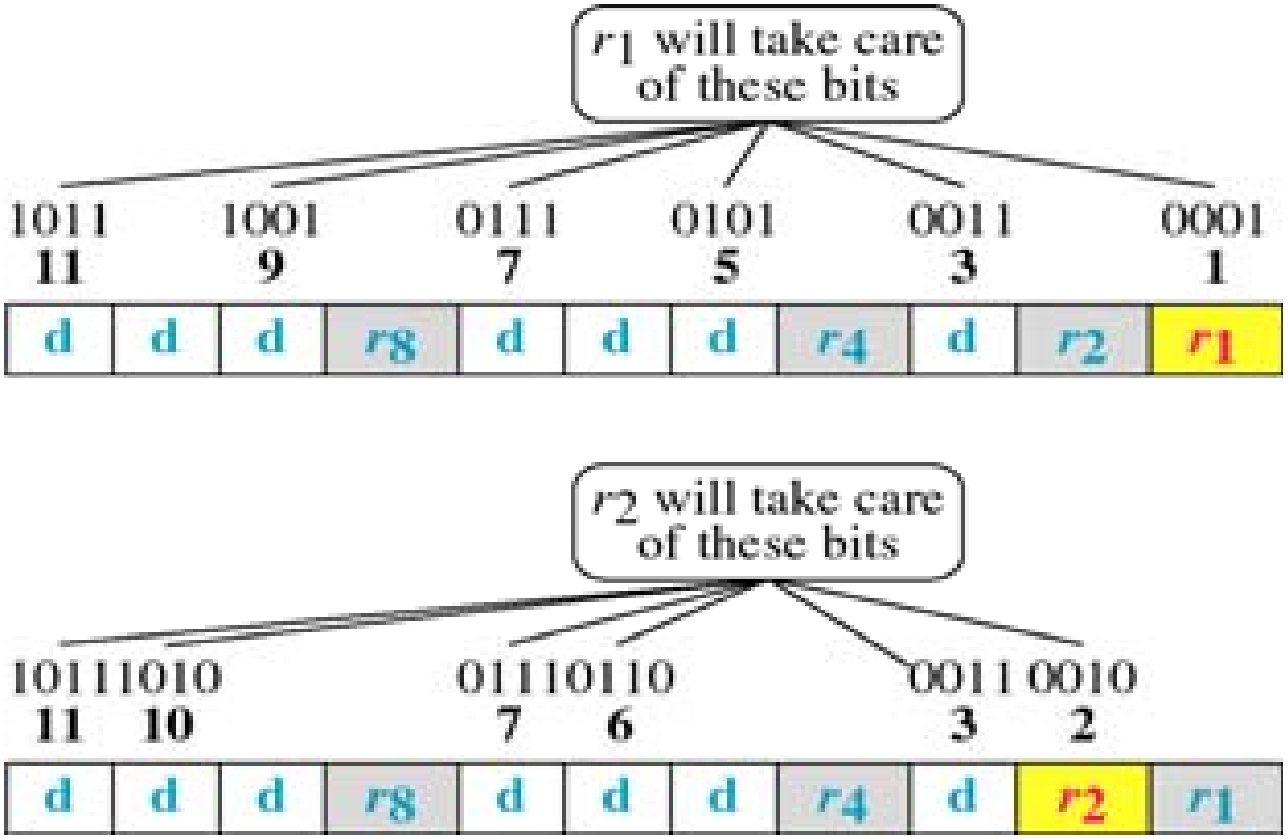
- Which data bits does each redundancy bit check?
  - r_1 checks data in **positions** whose **binary** representations have a: **1** in the **same binary position** as r_1
  - r_2 checks data in positions whose binary representations have a: **1** in the **same binary position** as r_2
  - r_4 checks data in positions whose binary representations have a: **1** in the **same binary position** as r_4
  - r_8 checks data in positions whose binary representations have a: **1** in the **same binary position** as r_8
- By writing a bit's position as a sum of powers of 2, can tell which redundancy bit(s) check this bit
  - bit in position **11** is checked by r_1 , r_2 , and r_8 because 11=1+2+8;
  - bit in position **6** is checked by r_2 and r_4 because 6=2+4; etc.

Hamming Code Calculation

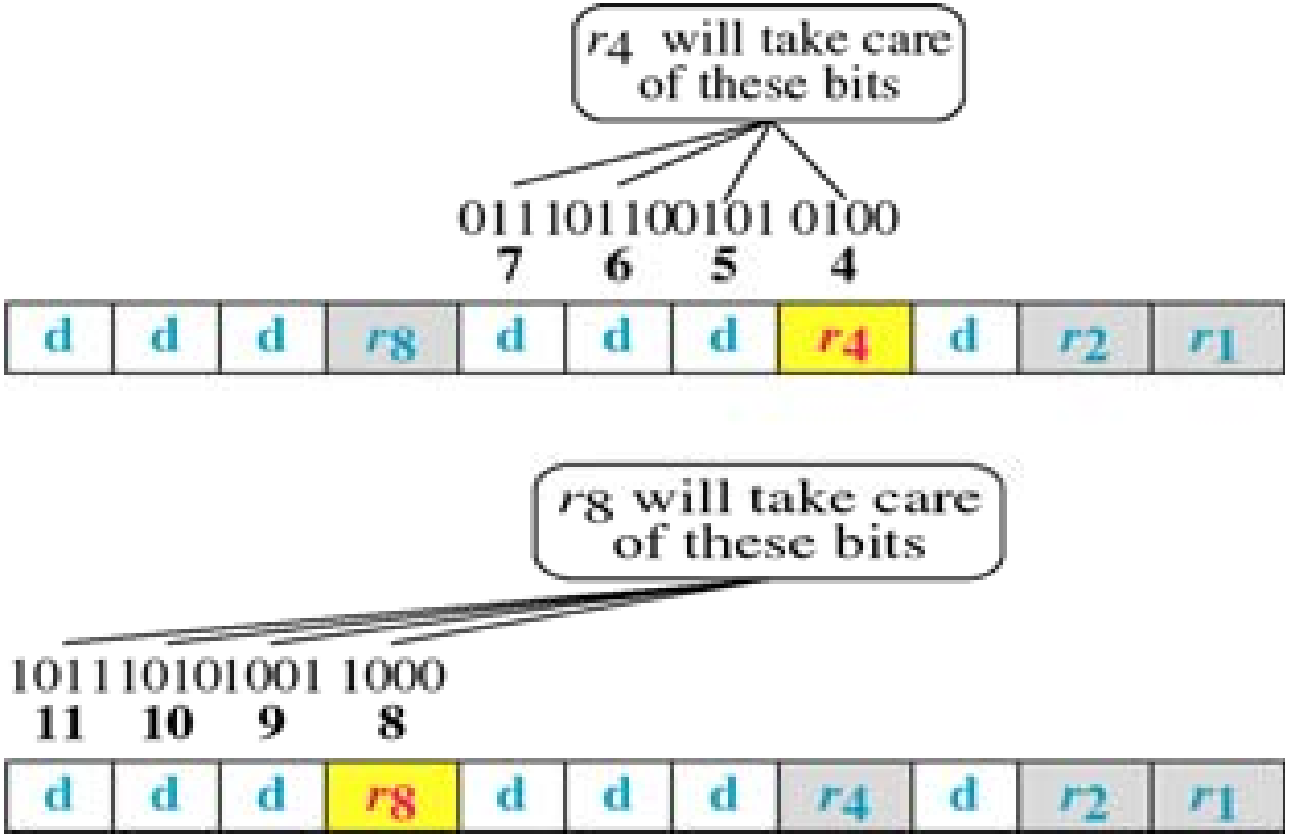
- Steps involved:

1. Number the bits in the codeword.
2. Turn numbers into binary representations.
3. Put in check bit placeholders.
4. Put data into the codeword.
5. Calculate the check bits individually.

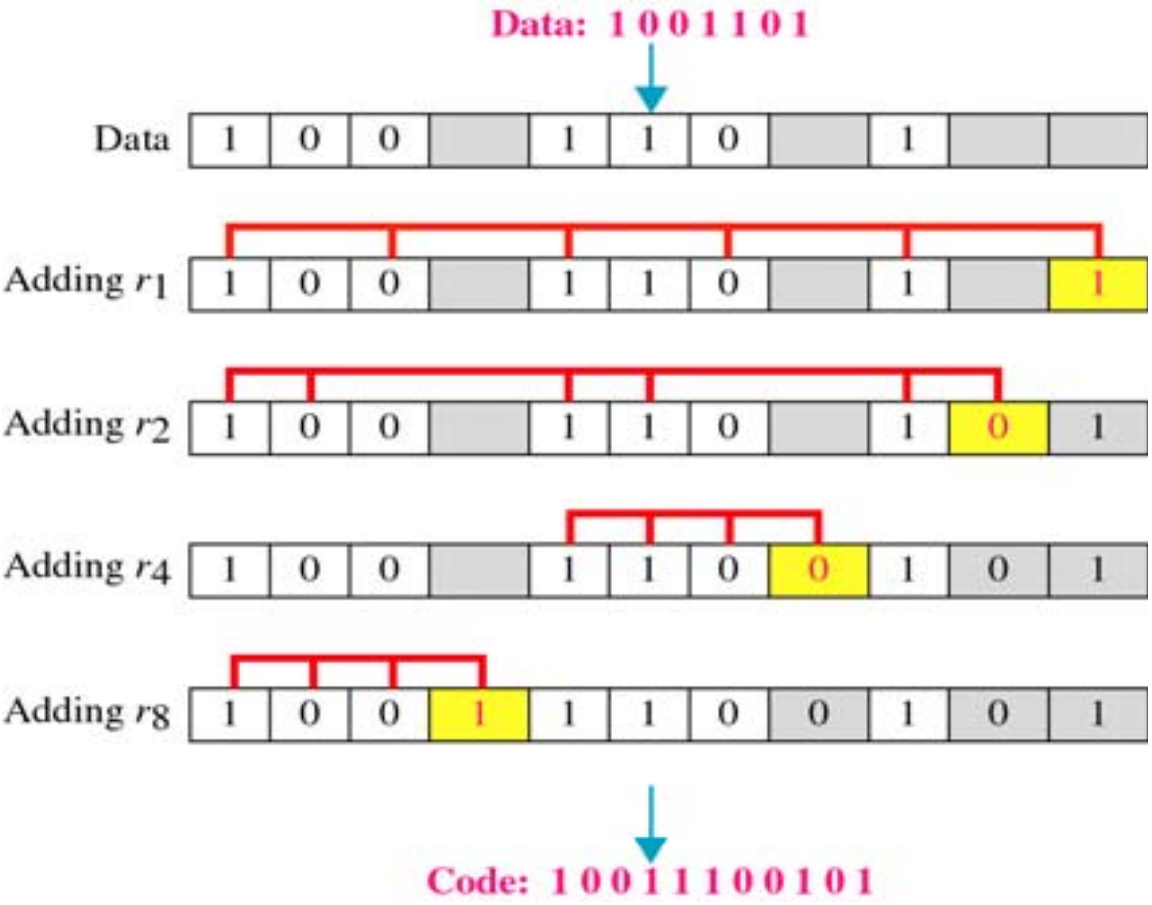
Hamming Code Calculation



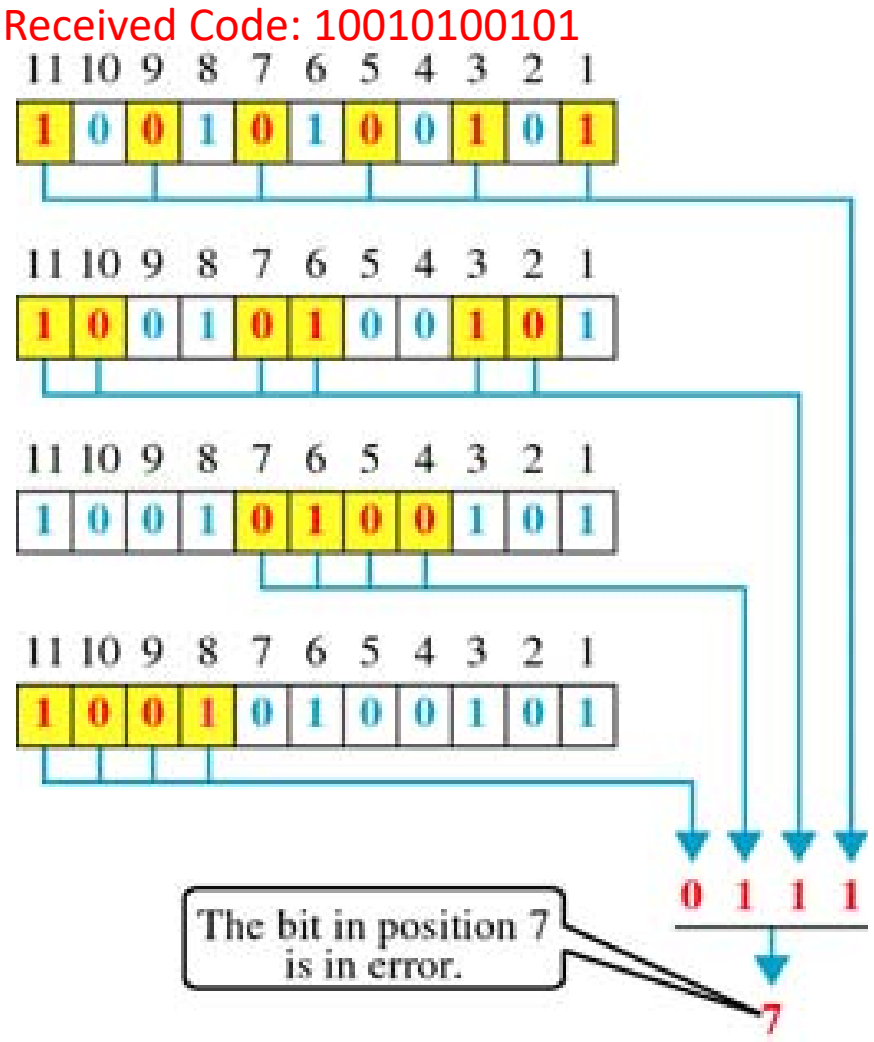
Hamming Code Calculation



Hamming Code Calculation

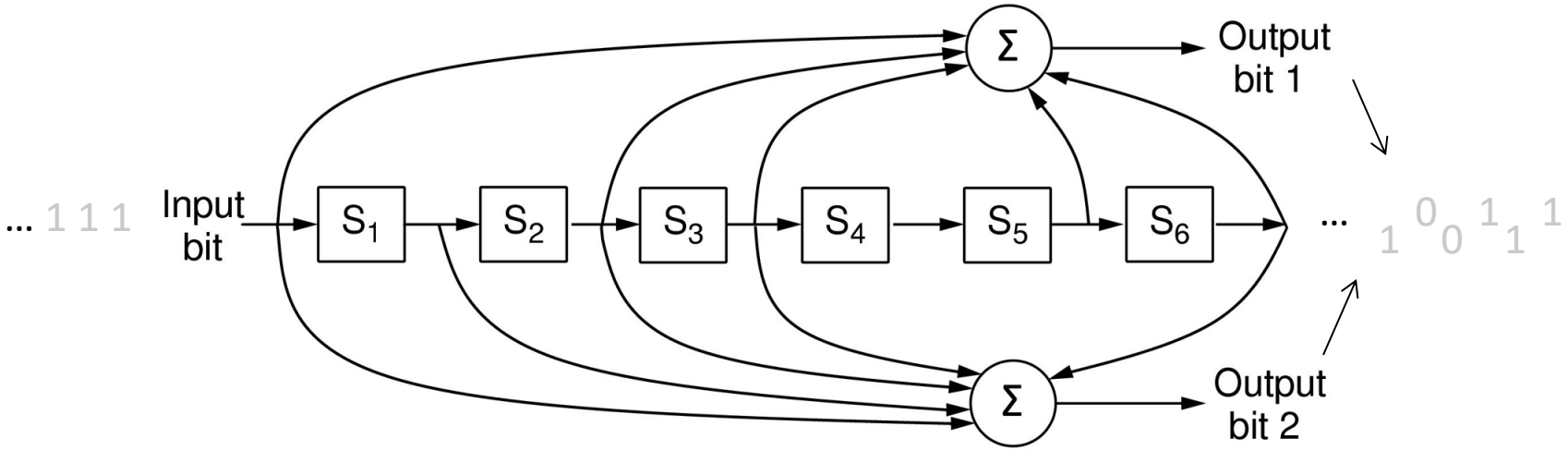


Hamming Code Checking



Binary Convolutional Codes

- Operates on a stream of bits, keeping internal state
  - Output stream is a function of all preceding input bits
  - Bits are decoded with the Viterbi algorithm



Popular NASA binary convolutional code (rate = ½) used in 802.11

Reed-Solomon codes

- Like Hamming codes, Reed-Solomon codes are linear block codes, and they are often systematic too.
- Unlike Hamming codes, which operate on individual bits, Reed-Solomon codes operate on ***m*** **bit symbols**.
- Reed-Solomon codes are widely used in practice because of their strong error-correction properties, particularly for burst errors.
- They are used for DSL, data over cable, satellite communications, and perhaps most ubiquitously on CDs, DVDs, and Blu-ray discs.

Reed-Solomon codes

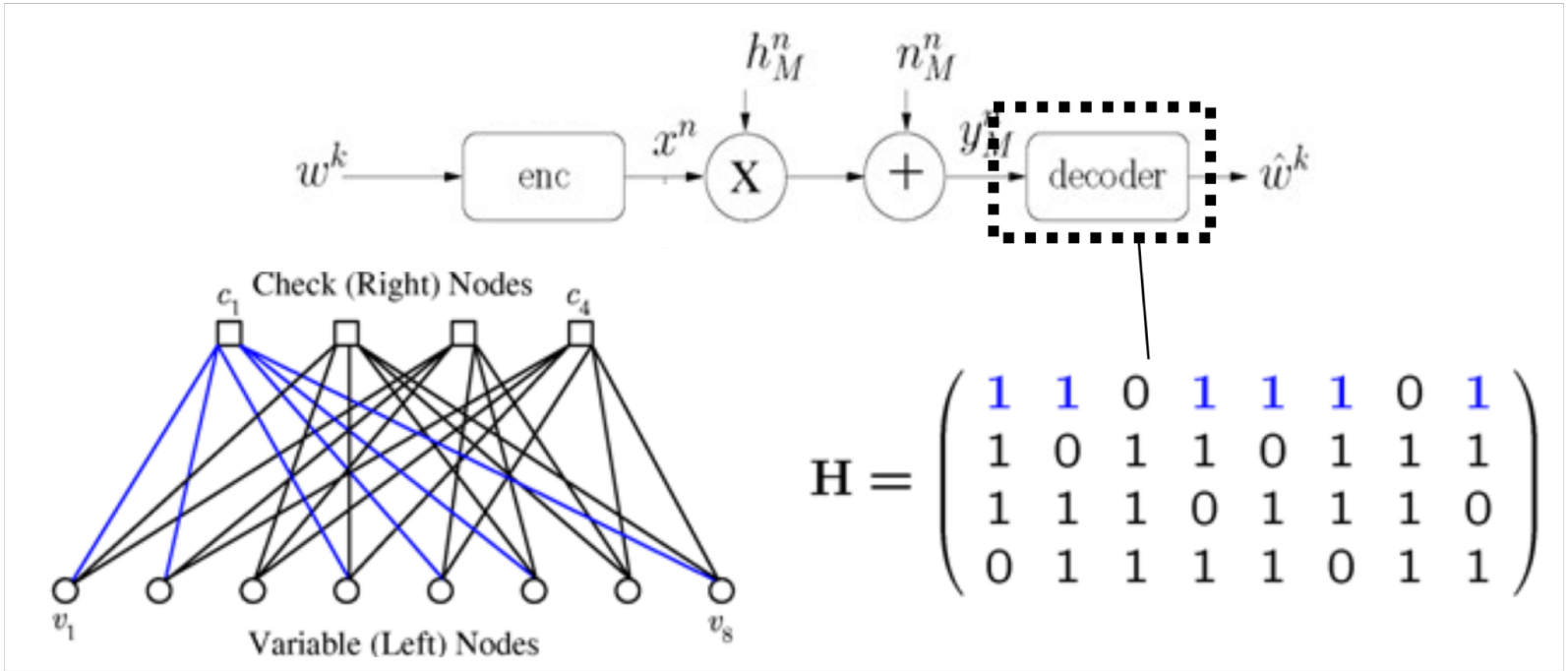
- Reed-Solomon codes are often used in combination with other codes such as a convolutional code.
  - Convolutional codes are effective at handling **isolated bit** errors, but they will fail, likely with **a burst of errors**, if there are too many errors in the received bit stream.
- By adding a Reed-Solomon code within the convolutional code, the Reed-Solomon decoding can mop up the error bursts, a task at which it is very good.
- The overall code then provides good protection against both single and burst errors.

Low-Density Parity Check codes

- LDPC codes are linear block codes
- Each output bit is formed from only a fraction of the input bits.
  - This leads to a matrix representation of the code that has a low density of 1s, hence the name for the code.
- The received codewords are decoded with an approximation algorithm that iteratively improves on a best fit of the received data to a legal codeword. This corrects errors.

Low-Density Parity Check codes

- LDPC codes are practical for large block sizes and have excellent error correction abilities that outperform many other codes in practice.
- They are part of the standard for digital video broadcasting, 10 Gbps Ethernet, power-line networks, and the latest version of 802.11.



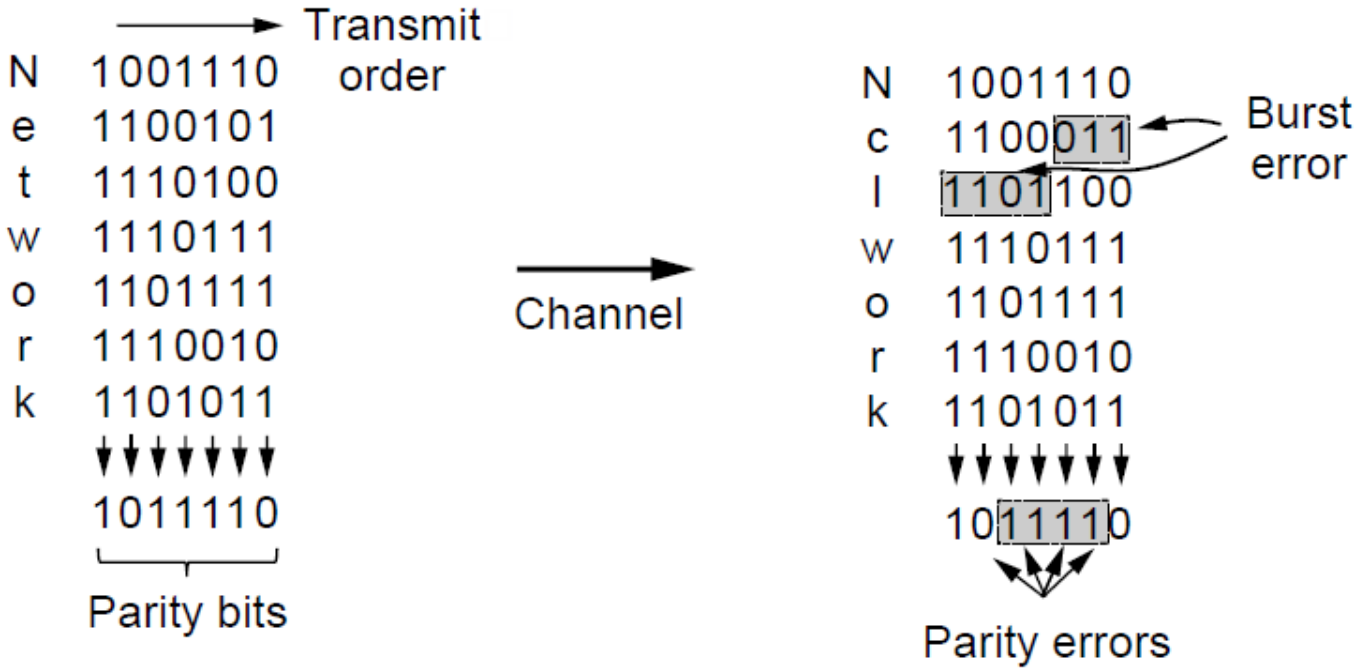
Parity

- Parity bit is added as the modulo 2 sum of data bits
  - Equivalent to XOR; this is even parity
  - Ex: 1110000 → 11100001
  - Detection checks if the sum is wrong (an error)

- Simple way to detect an *odd* number of errors
  - Ex: 1 error, 11100101; detected, sum is wrong
  - Ex: 3 errors, 11011001; detected sum is wrong
  - Ex: 2 errors, 11101101; *not detected*, sum is right!
  - Error can also be in the parity bit itself
  - Random errors are detected with probability ½

Parity

- Interleaving of N parity bits detects burst errors up to N
  - Each parity sum is made over non-adjacent bits
  - An even burst of up to N errors will not cause it to fail



Checksums

- Checksum treats data as N-bit words and adds N check bits that are the modulo 2^N **sum** of the words
  - Ex: Internet 16-bit 1s complement checksum
  - errors may be detected by summing the entire received codeword, both data bits and checksum.
  - If the result comes out to be zero, no error has been detected.

- Properties:
  - Improved error detection over parity bits
  - Detects bursts up to N errors
  - Detects random errors with probability $1/2^N$
  - Vulnerable to systematic errors, e.g., added zeros

Cyclic Redundancy Check (CRC)

- CRCs are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only
  - Ex: Ethernet 32-bit CRC is defined by:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$$

- A k-bit frame is regarded as the coefficient list for a polynomial with k terms, ranging from x^{k-1} to x^0

- Such a polynomial is said to be of degree $k - 1$. The high-order (leftmost) bit is the coefficient of x^{k-1}, the next bit is the coefficient of x^{k-2}, and so on

Cyclic Redundancy Check (CRC)

- For example, 110001 has 6 bits and thus represents a six-term polynomial with coefficients:
- 1,1,0,0,0,and1:
- $1x^5 + 1x^4 + 0x^3 + 0x^2 + 0x^1 + 1x^0$.

- When the polynomial code method is employed, the sender and receiver must agree upon a **generator polynomial**, $G(x)$, in advance

- Both the high-order and low-order bits of the generator **must be 1**

Cyclic Redundancy Check (CRC)

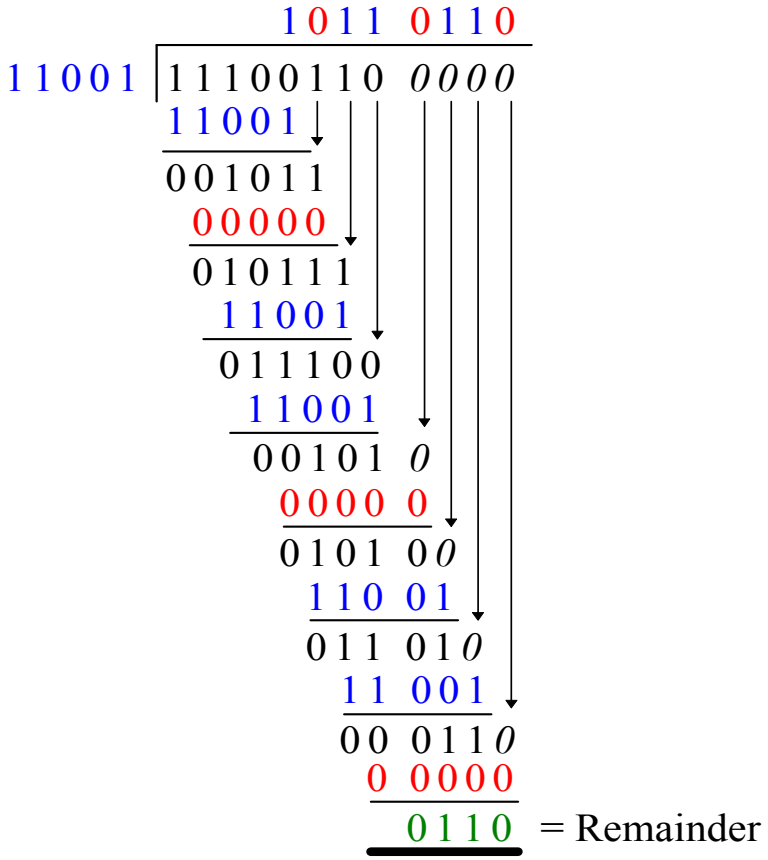
- Let M be the frame contents that are to be protected, usually from everything except the flags and the bit stuffing. Assume M to be k bits long.
- Let the CRC check bits be n bits long
- Let G, the Generator Polynomial, be $n + 1$ bits long
- Divide M with n bits zeros appended to it, by G and the remainder is the CRC check bits.
- This n bits long CRC check bits is sent along with the frame to the receiver

Cyclic Redundancy Check (CRC)

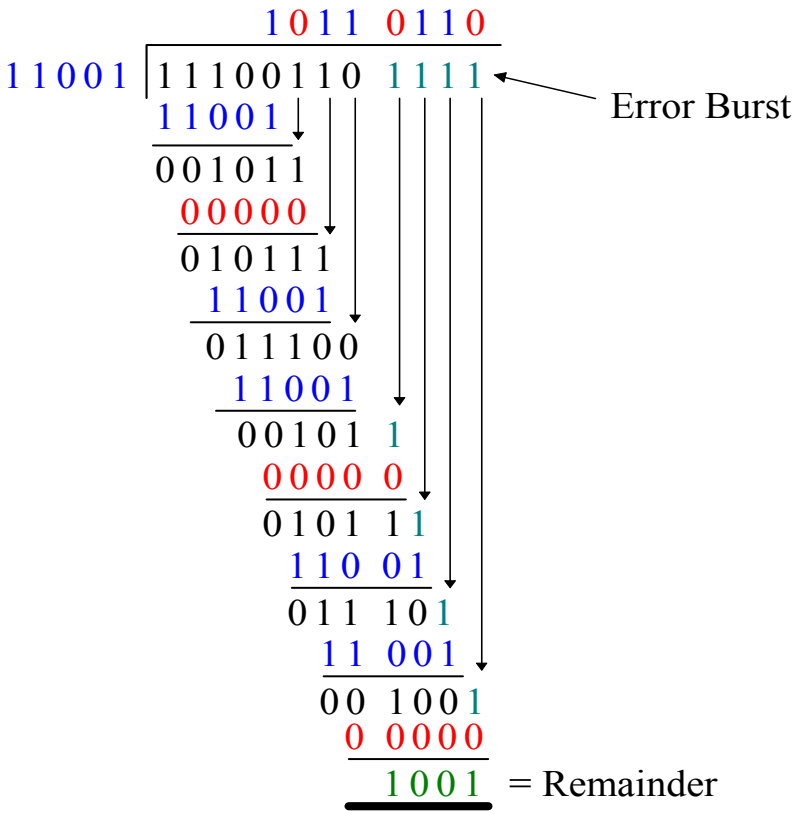
- You transmit the bit sequence, M, appended by the CRC, and any other unprotected bits like the flags.
- The receiver takes the unprotected bits off, and then divides the $k + n$ bits of the bit stream by the Generator G.
- If the remainder is zero then there are no detected errors. A non-zero remainder will indicate that errors have been detected.

Example 1

Note:
M: 11100110
(k=8)
M with zeros :
11100110 *0000*
(n=4)
G: 11001
CRC: 0110
Transmitted bits:
111001100110



Example 2 – At Receiver



Remainder ≠ 0, ⇒ Error Detected

CRCs

- Stronger detection than checksums:
  - E.g., can detect all double bit errors
  - Not vulnerable to systematic errors
- Although the calculation required to compute the CRC may seem complicated, it is easy to compute and verify CRCs in hardware with **simple shift register/XOR circuits**.
- Dozens of networking standards include various CRCs, including virtually all LANs (e.g., Ethernet, 802.11) and point-to-point links (e.g., packets over SONET).

Common Network Devices

- Physical layer
  - Network Interface Card (NIC)
  - Repeater
  - **Hub**
- Data link layer
  - Bridge
  - **Switch**
- Network layer
  - **Router**

The highlighted devices will be used in your “PacketTracer” labs.

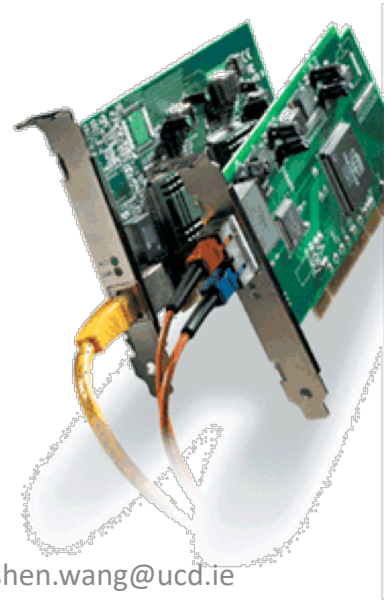
Devices in our lab

Why network devices?

- An efficient solution supports a growing network
  - The number of lines used to connect end devices reduced from $O(n^2)$ to $O(n)$
- Each network device works at different layers, thus manages network in difference scales and implements services for certain layers.
  - E.g.: Packet/Frame forwarding
- Connecting heterogenous networks that are in a similar size.
  - Different protocols might be used. Network device can do translation.

Network Interface Card (NIC)

- This the first point of contact to the network.
- Every connected device (e.g. phone, laptop) has at least one network interface card.
- It deals with the medium on which data will be sent (modulation/demodulation).



Repeater

- It is an electronic device that receives a signal and retransmits it.
- It is used to extend transmissions so that the signal can cover longer distances, as it **regenerates (not just amplify)** the signal by releasing its negative impact caused by attenuation and distortion.
- Amplifier can only improve the signal affected by attenuation.

Hub

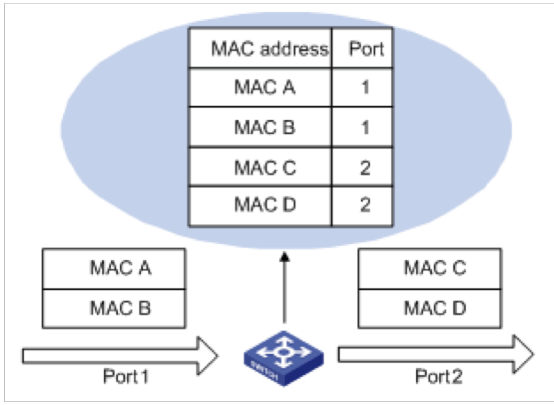
- It is a repeater that has multiple ports.
- It is the simplest and cheapest solution to connect multiple devices.
- Any signal that is inputted into an interface gets **broadcast** on all the other interfaces
- It can only work in half-duplex transmission mode. Therefore, machines connect to it are still in a **collision domain**.

Bridge

- It is a store-forward device. It deals with layer 2 frames, rather than layer 1 signals.
- It can inspect the frame and determine intended destination (**Next Hop** Address).
- It can connect LANs with different protocols.
- It can only forward one frame at the same time.
- It learns the forwarding table itself.
  - We will study this learning process later

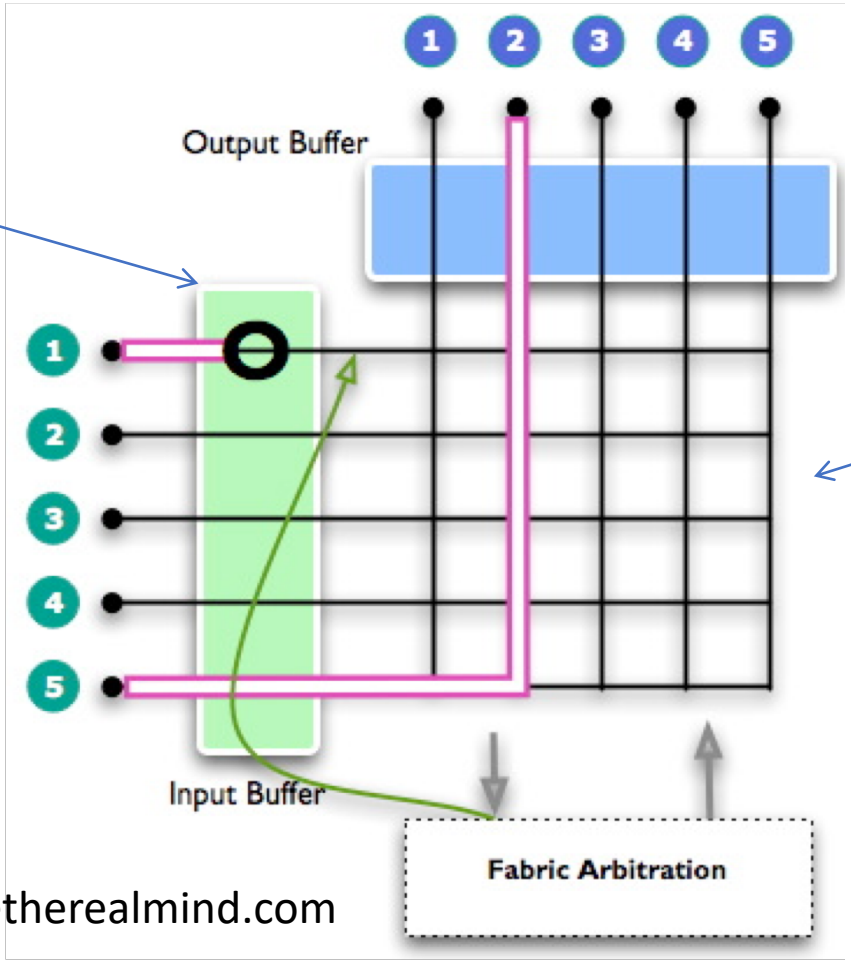
Switch

- It is a bridge that has multiple ports.
- It can forward multiple frames at the same time.
- It can connect end devices directly, while bridge can only connect to different LANs.



Switch

Queuing Mechanics can be complex



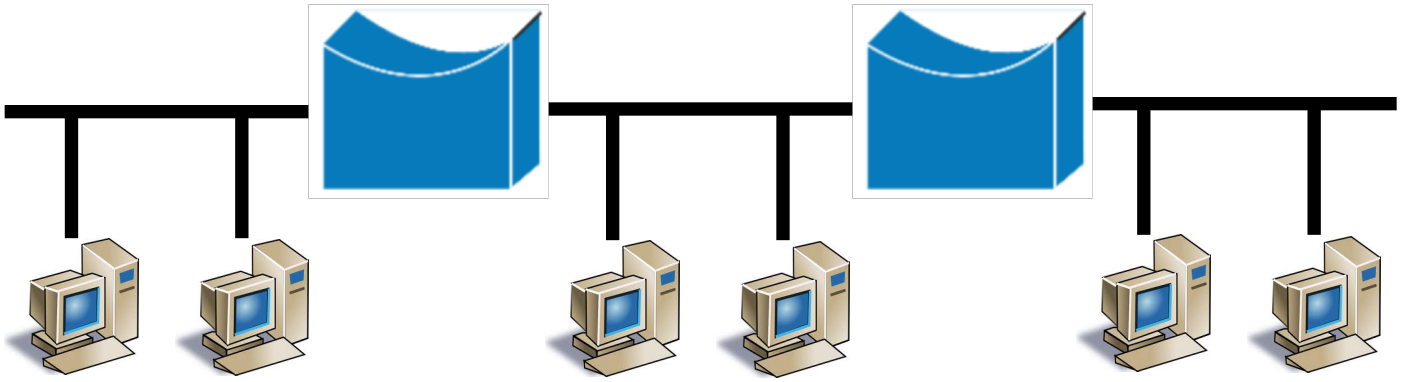
Switching Backplane

CPU Managed

Image: http://etherealmind.com

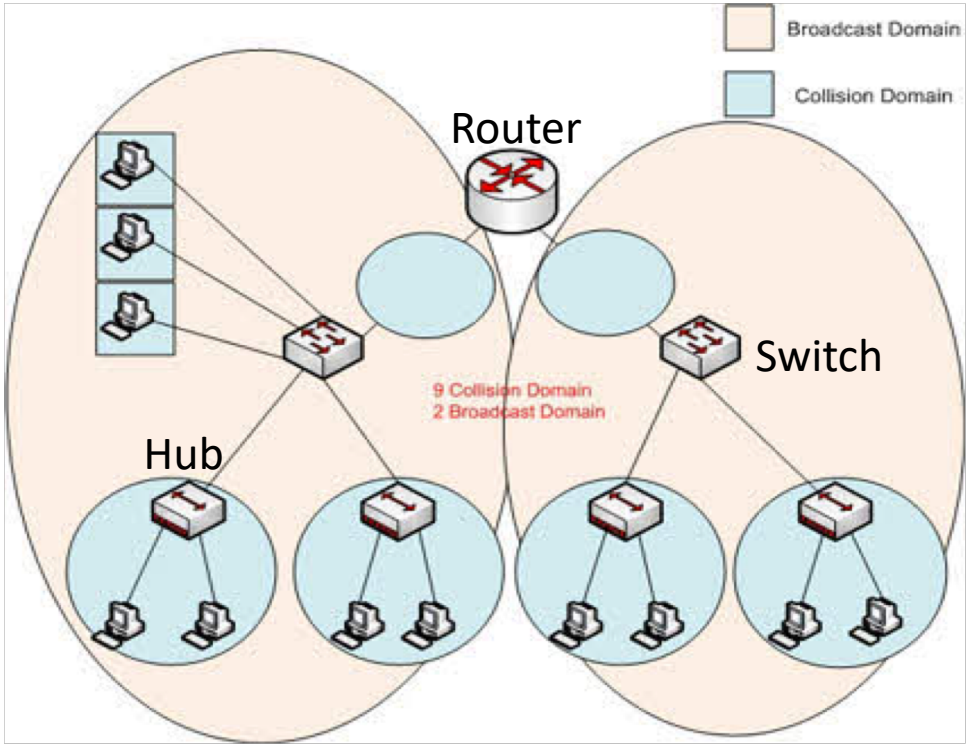
Question

- Suppose the data rate for each network segment is 10Mb/s, what is the total throughput for the following network?
  - 10*3 = 30Mb/s
- If replacing bridges with hubs, what is the total throughput again?
  - Still 10Mb/s



Collision & Broadcast Domain

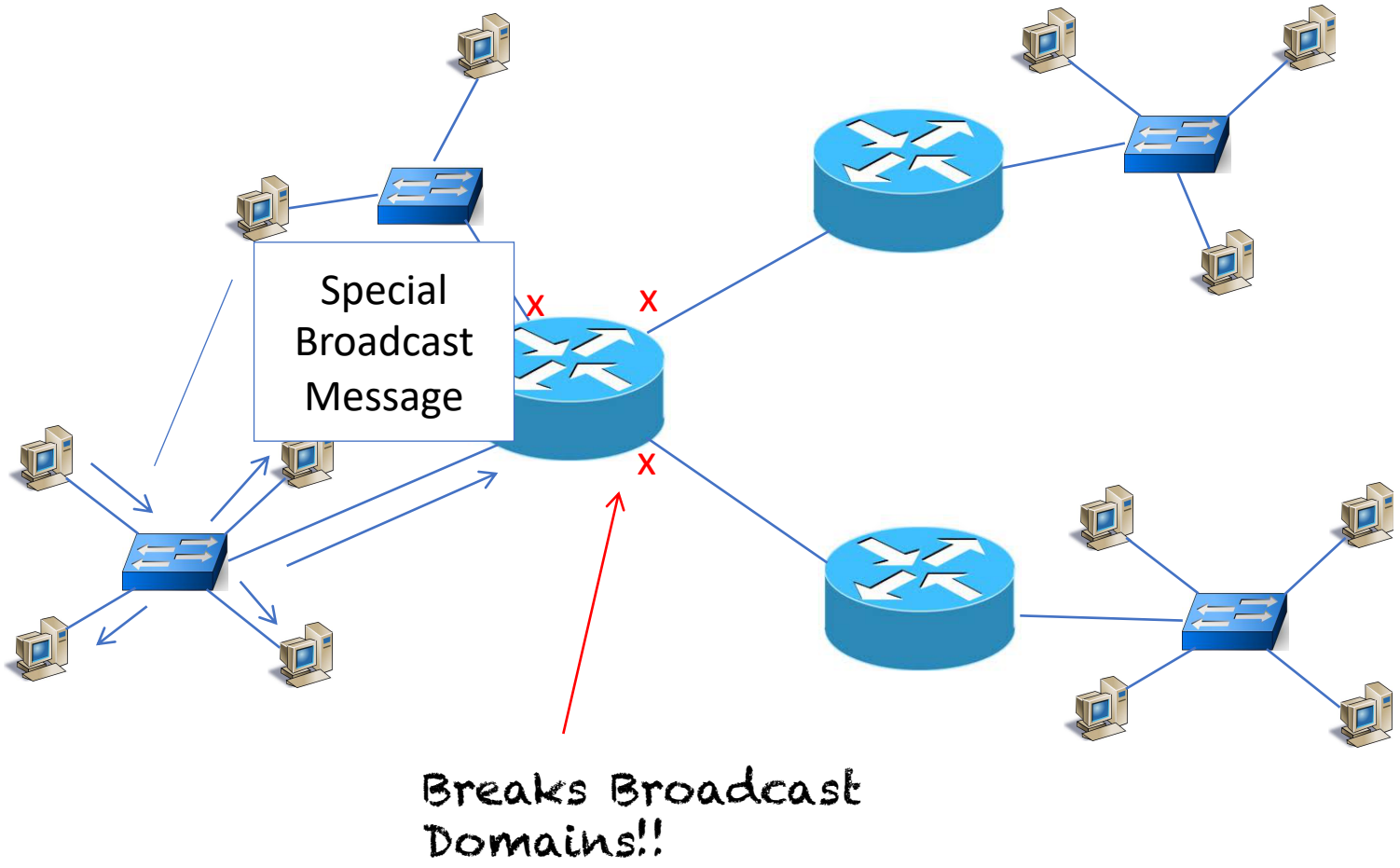
- Switch (or layer 2 devices) separates collision domain.
- Router (or layer 3 devices) separates broadcast domain.



Router

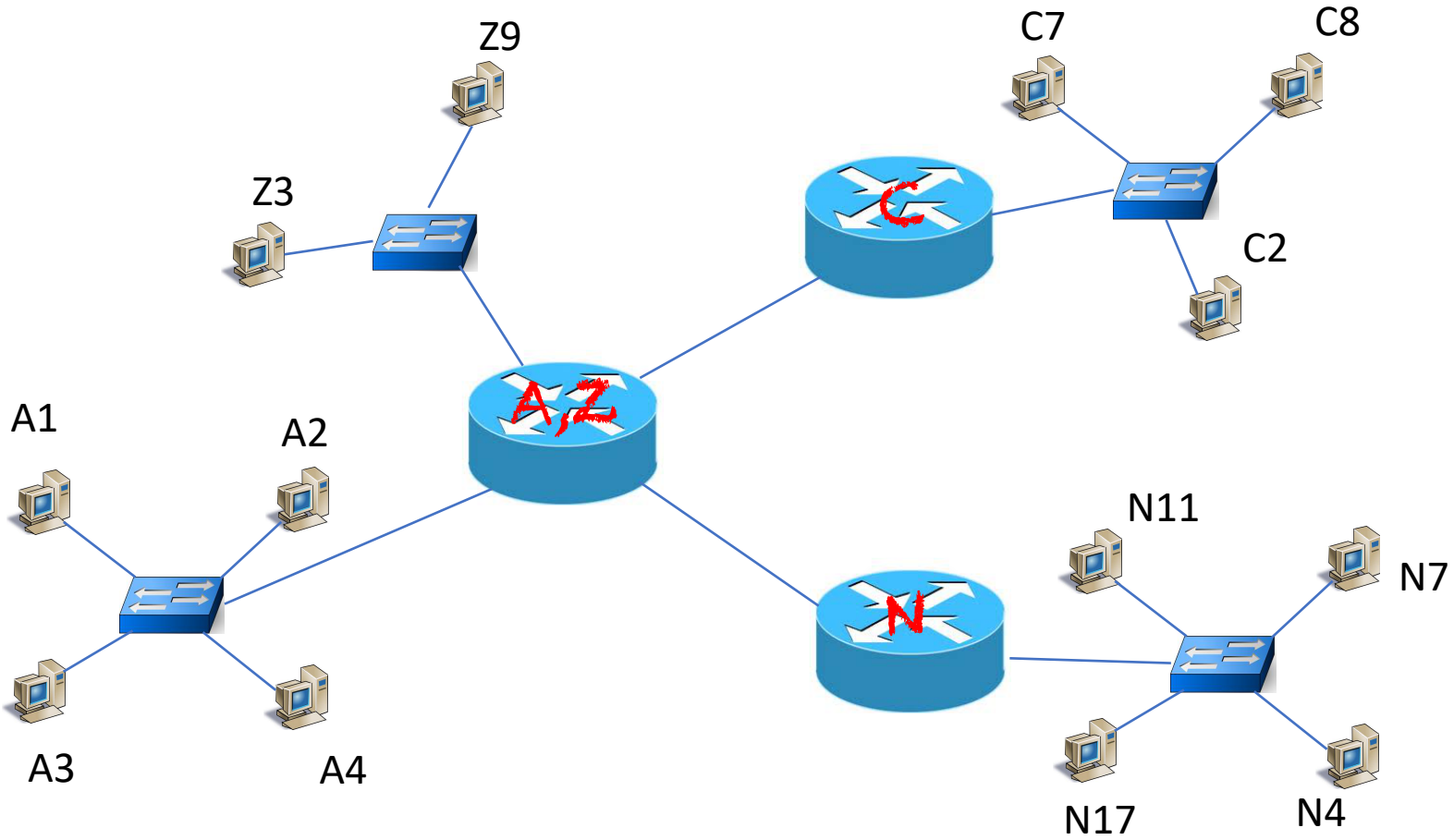
- It connects heterogenous networks, rather than LANs (much smaller scale).
- It builds network in a much larger scale as it avoids broadcast storm.
- It inspects the **IP packet** (End-to-End Address)
- It makes a decision on the best direction for the packet.

Router

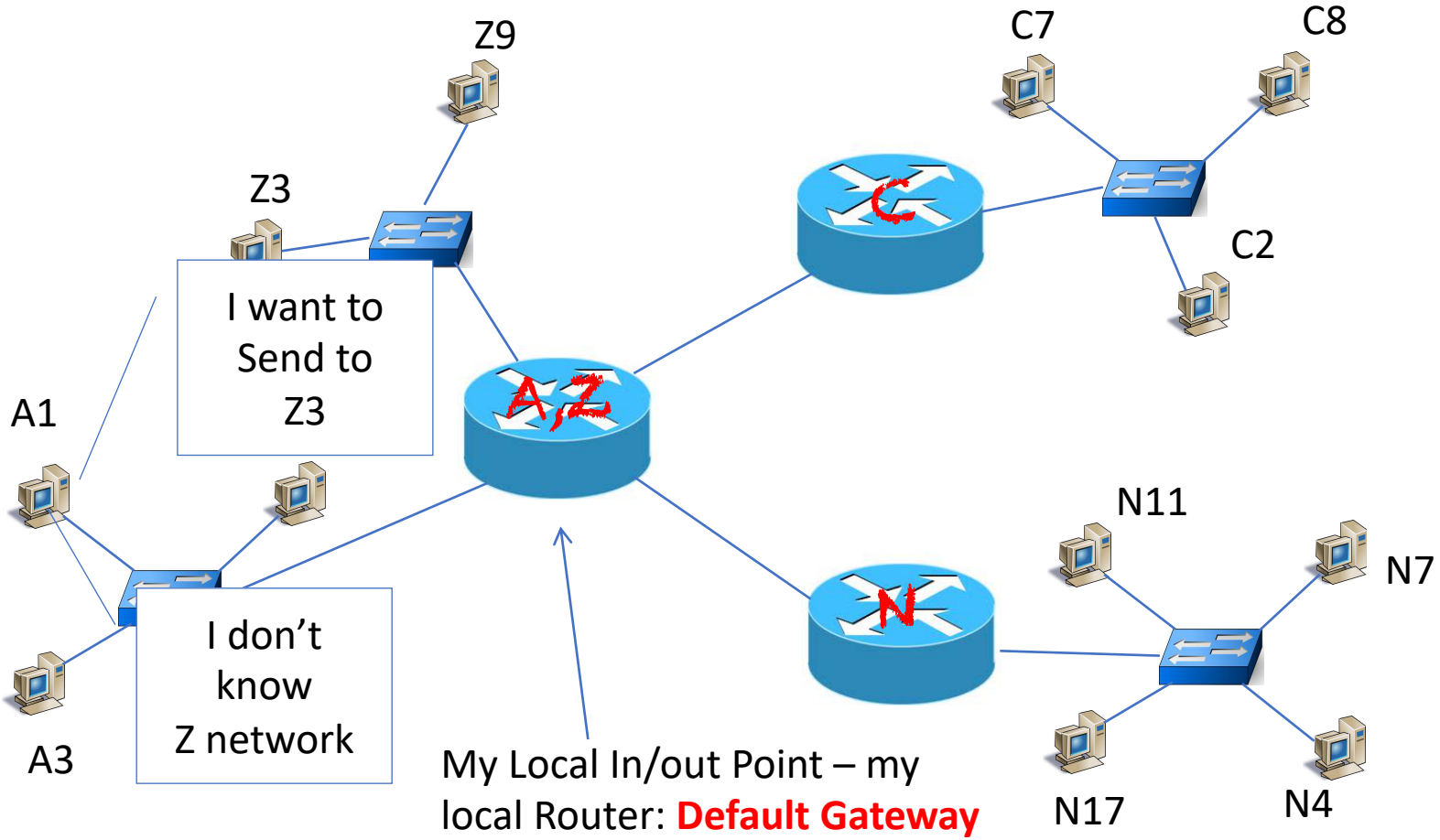


Router

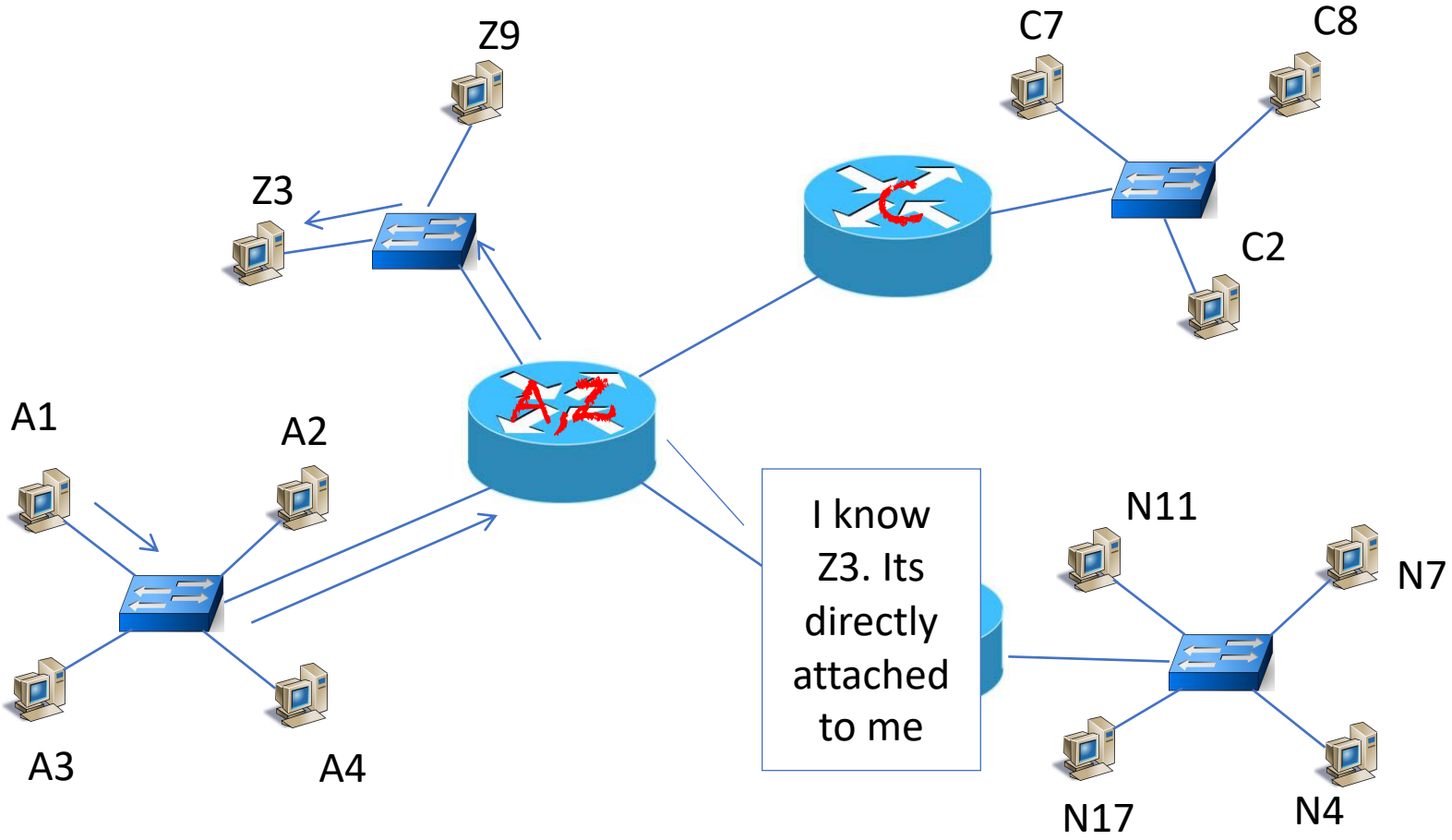
Add Global Addresses!
End-to-End



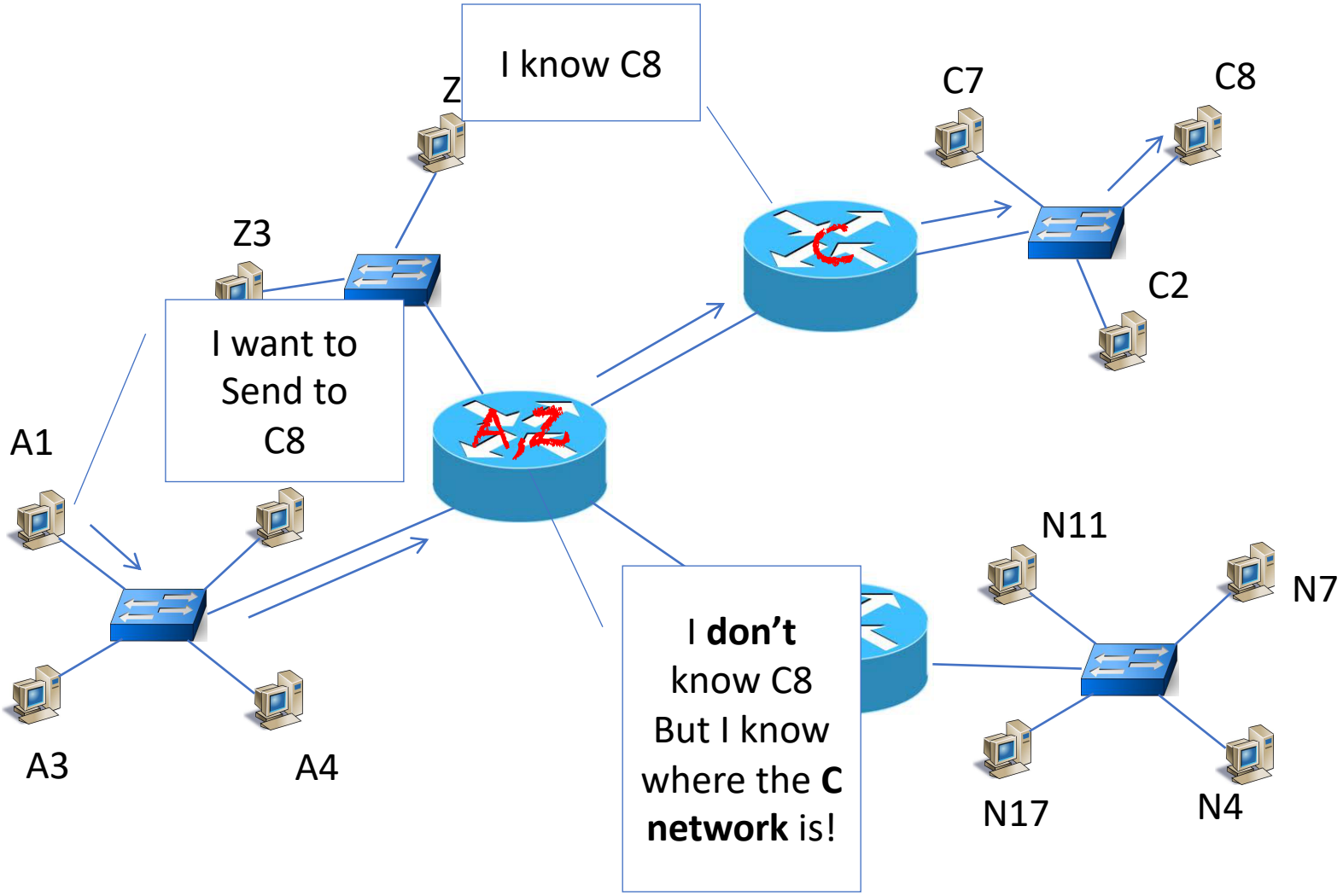
Router


Z9
C7
C8
Z3
C
C2
I want to
Send to
Z3
A1
A,Z
N11
N7
N
I don't
know
Z network
My Local In/out Point – my
local Router: Default Gateway
A3
N17
N4
Who can I send this message to?

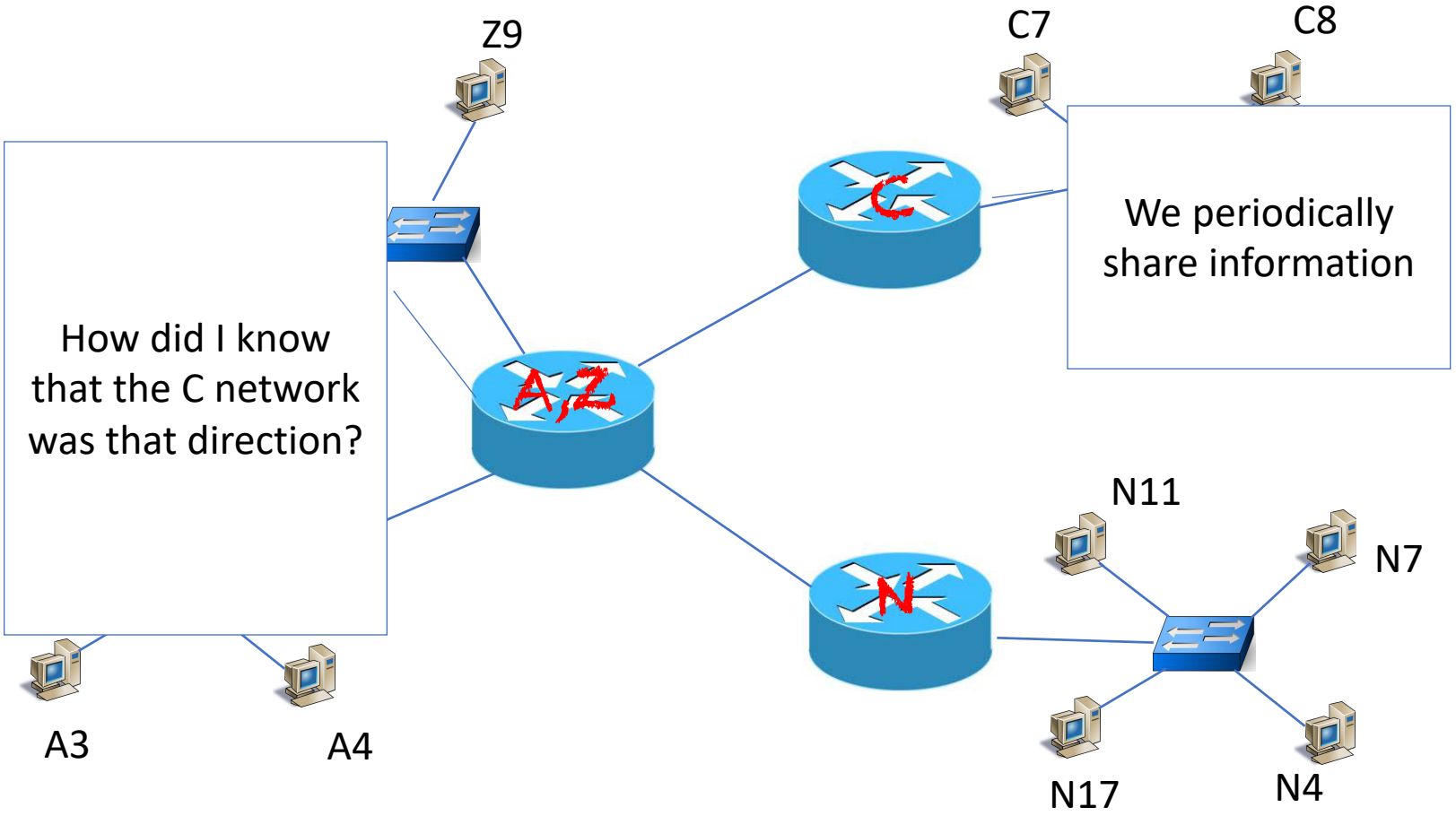
Router


Z9
C7
C8
Z3
C2
A1
A2
A3
A4
I know Z3. Its directly attached to me
N11
N7
N17
N4

Router

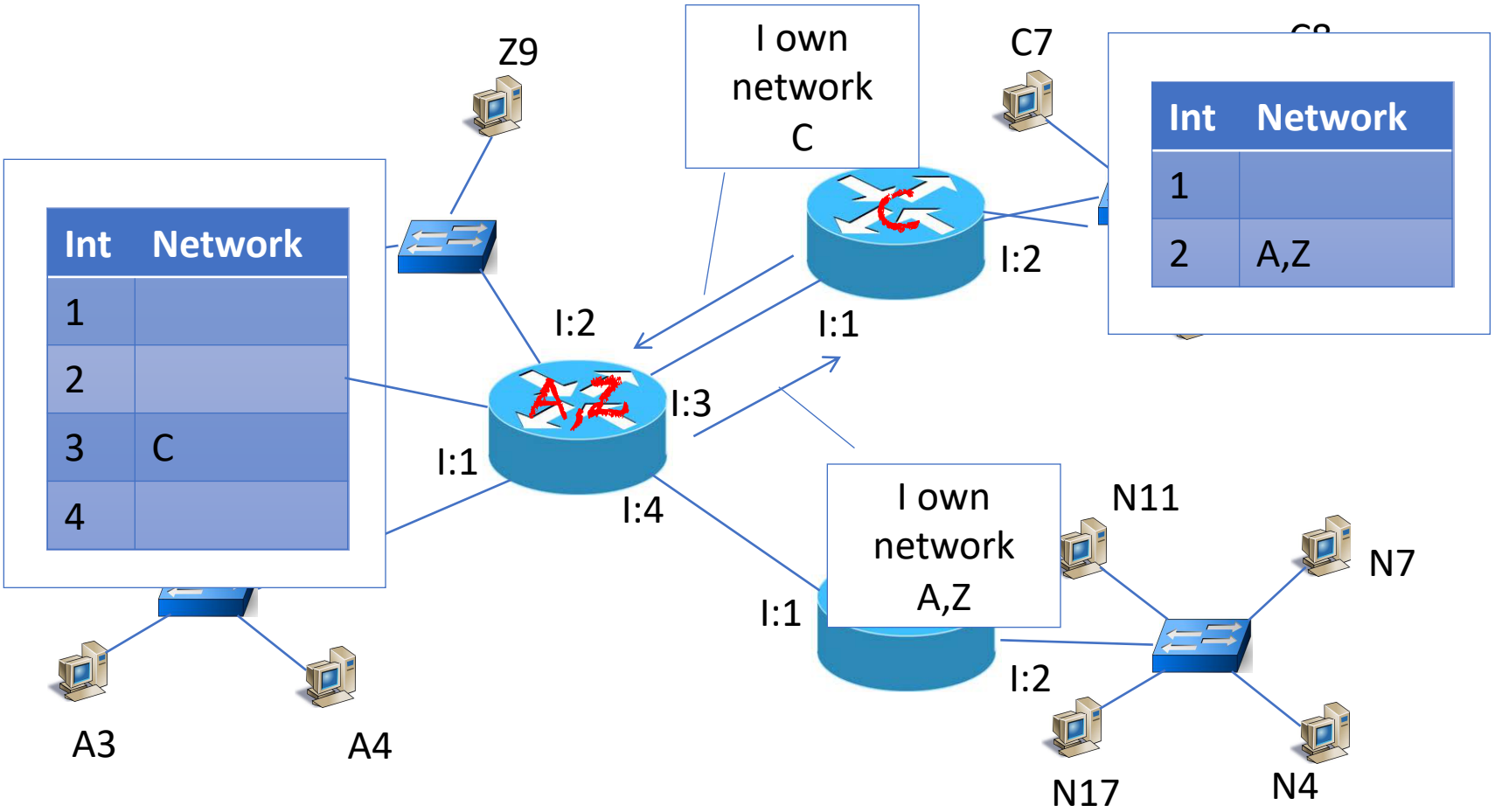

I know C8
Z
C7
C8
Z3
C
C2
I want to
Send to
C8
A1
A,Z
N11
N7
I **don't**
know C8
But I know
where the **C**
network is!
A3
A4
N17
N4

Router

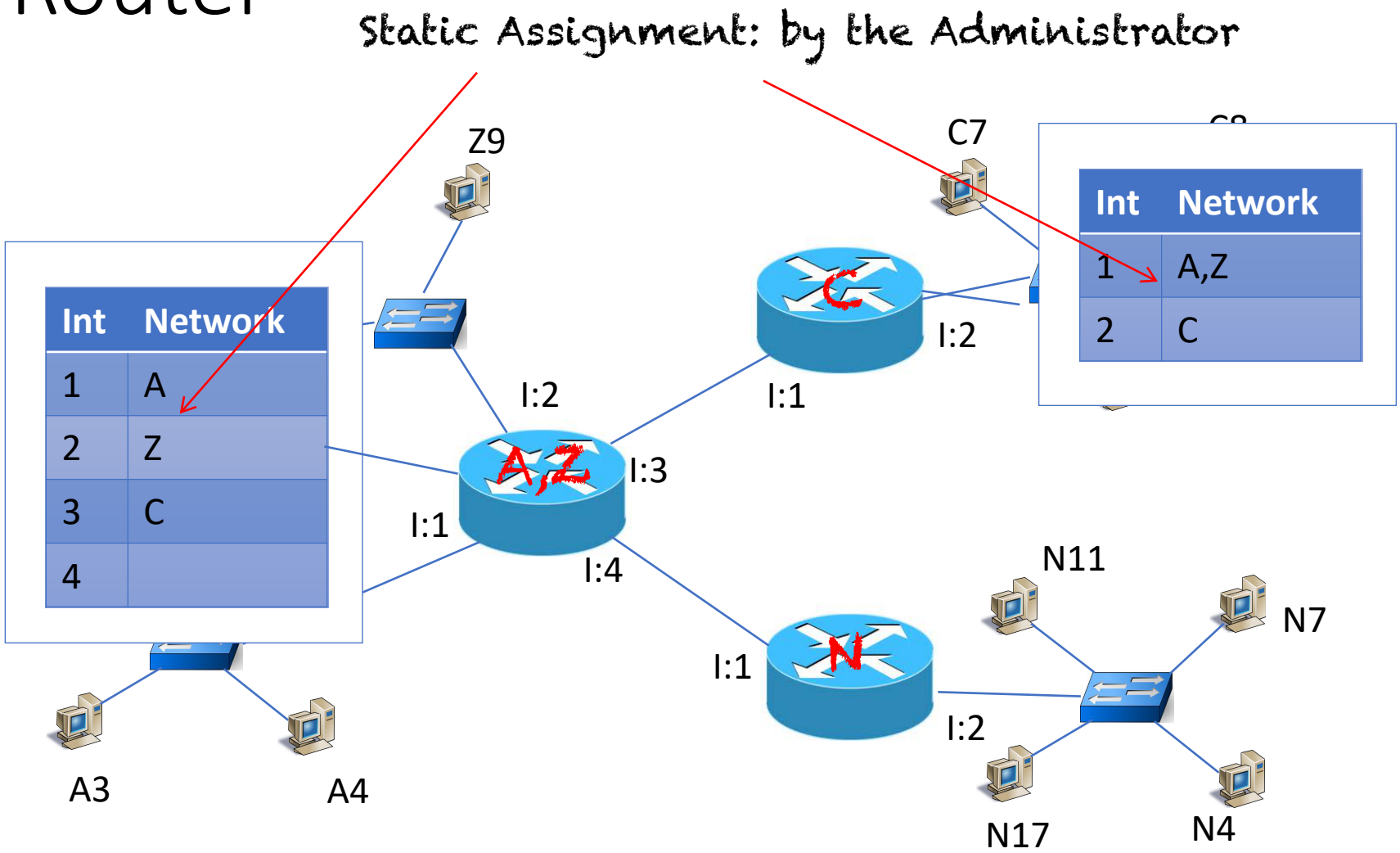

Z9
C7
C8
We periodically share information
How did I know that the C network was that direction?
C
A,Z
N
N11
N7
A3
A4
N17
N4

Router

A routers port is called an **Interface**



Router


Static Assignment: by the Administrator
Z9
C7
Int
Network
1
A
2
Z
3
C
4
Int
Network
1
A,Z
2
C
I:2
I:1
I:2
I:3
I:1
I:4
A,Z
C
N
I:1
I:2
N11
N7
N17
N4
A3
A4

Gateway

- Usually an interface of a router that is configured on a network ownership border.
- It is the first place (IP address) that each outgoing packet needs to go, if it intends to visit other networks.
- Administers traffic Policy!
- Can inspect the whole packet.

Summary

Application layer	Application gateway
Transport layer	Transport gateway
Network layer	Router
Data link layer	Bridge, switch
Physical layer	Repeater, hub