Computer Networks

Wireshark and HW-2
Autumn 2023

Administrivia

- Project-1 is due October 25th
- HW1 is due October 18th

Wireshark

- Download : https://www.wireshark.org/download.html
- User's guide: https://www.wireshark.org/docs/wsug_html_chunked/

What is Wireshark

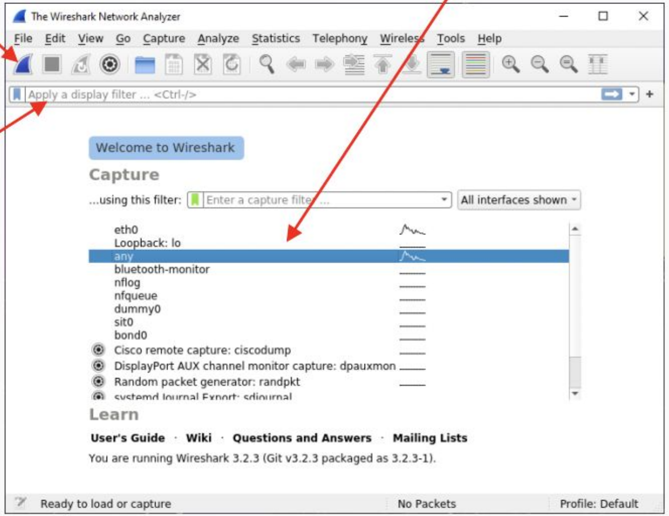
It’s a tool that captures and analyzes packets sent over the network

- Very commonly used in Network Forensics
- Captures all packets through a network interface (ethernet, WiFi)
- Analyzes packets and decodes raw data if the protocol is recognized
- Filters packets based on user’s input

Wireshark Interface

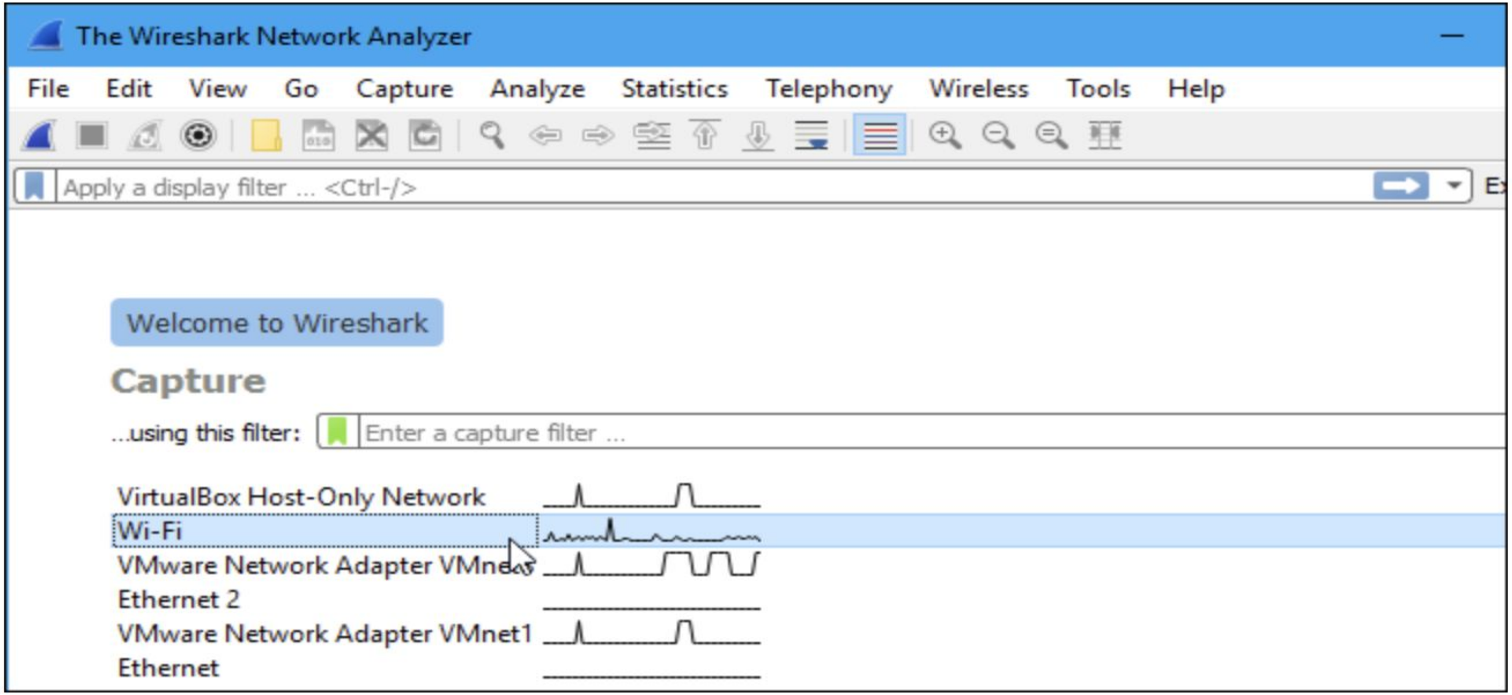
Start Packet Capture

Interface Selection

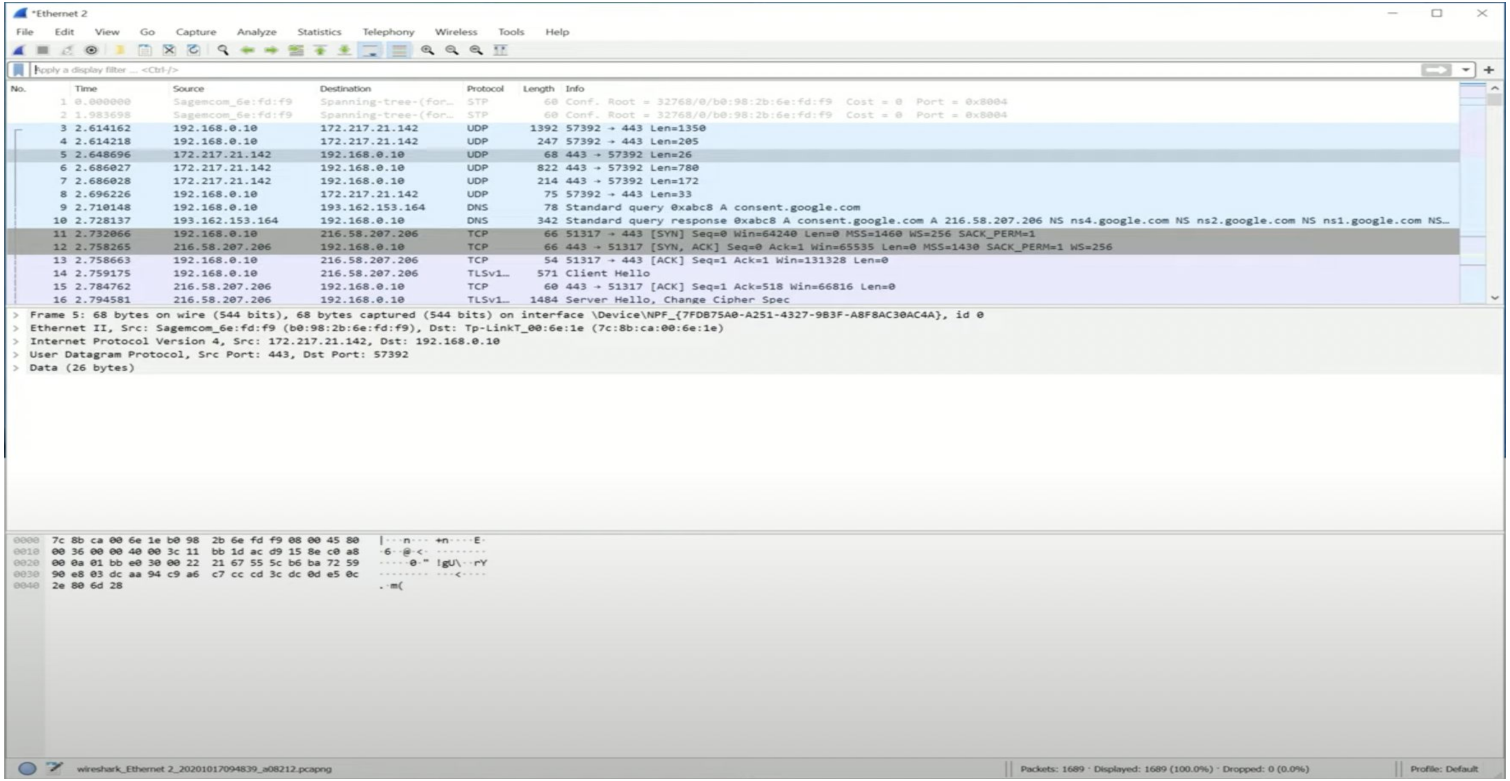


Display filter for captured packets

Wireshark Interface



Wireshark Interface



Wireshack Filtering

- If you want to capture all TCP packets, write TCP in the filter. Same for UDP
- You can also track the packets going to a particular host using tcp contains “host”
- You can track packets going and coming back to a particular IP address.

Wireshark filtering

- Let’s try to hack password of a not secure website.
  - http://vbsca.ca/login/login.asp

- This is very basic of Wireshark. It is capable of a lot more.
- Additional links:

https://www.wireshark.org/docs/

https://www.wireshark.org/docs/man-pages/wireshark-filter.html

Clock Recovery

- Um, how many zeros was that?
  - Receiver needs frequent signal transitions to decode bits

- Several possible designs
  - E.g., Manchester coding and scrambling (§2.5.1)

Clock Recovery – 4B/5B

- Map every 4 data bits into 5 code bits without long runs of zeros
  - 0000 → 11110, 0001 → 01001, 1110 → 11100, … 1111 → 11101
  - Has at most 3 zeros in a row
  - Also invert signal level on a 1 to break up long runs of 1s (called NRZI)

Clock Recovery – 4B/5B (2)

- 4B/5B code for reference:
  - 0000→11110, 0001→01001, 1110→11100, …
    1111→11101
- Message bits: 1 1 1 1 0 0 0 0 0 0 0 1

Coded Bits:

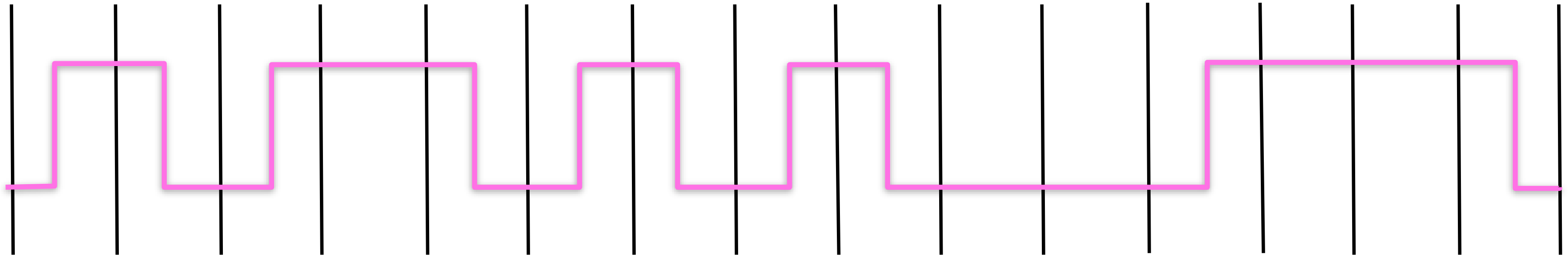
Signal:

Clock Recovery – 4B/5B (3)

- 4B/5B code for reference:
  - 0000→11110, 0001→01001, 1110→11100, …
    1111→11101
- Message bits: 1 1 1 1 0 0 0 0 0 0 0 1

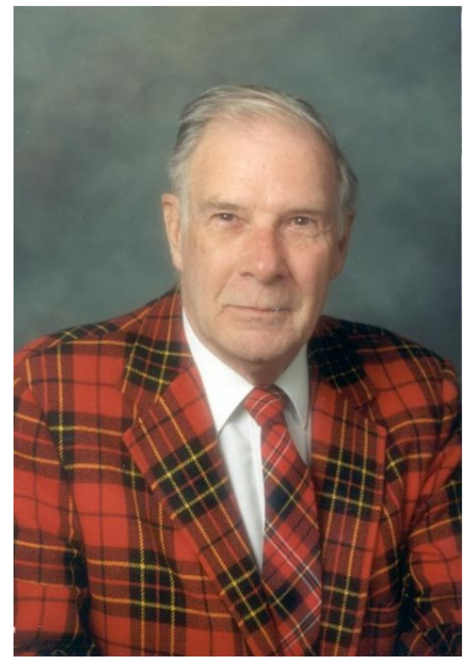
Coded Bits: 1 1 1 0 1 1 1 1 1 0 0 1 0 0 1

Signal:



R.W. Hamming (1915-1998)

- Much early work on codes:
  - "Error Detecting and Error Correcting Codes", BSTJ, 1950
- See also:
  - "You and Your Research", 1986



Source: IEEE GHN, © 2009 IEEE

Hamming Distance

- Distance is the number of bit flips needed to change D_1 to D_2

- Hamming distance of a coding is the minimum error distance between any pair of codewords (bit-strings) that cannot be detected

Hamming Distance (2)

- Error detection:
  - For a coding of distance d+1, up to d errors will always be detected
- Error correction:
  - For a coding of distance 2d+1, up to d errors can always be corrected by mapping to the closest valid codeword

Why Error Correction is Hard

- If we had reliable check bits we could use them to narrow down the position of the error
  - Then correction would be easy
- But error could be in the check bits as well as the data bits!
  - Data might even be correct

Hamming Code

- Gives a method for constructing a code with a distance of 3
  - Uses $n = 2^k - k - 1$, e.g., n=4, k=3
  - Put check bits in positions p that are powers of 2, starting with position 1
  - Check bit in position p is parity of positions whose p-th LSBit is same as p's
- Plus an easy way to correct [soon]

Hamming Code (2)

- Example: data=0101, 3 check bits
  - 7 bit code, check bit positions 1, 2, 4
  - Check 1 covers positions 1, 3, 5, 7 (LSB is 1)
  - Check 2 covers positions 2, 3, 6, 7 (2nd LSB is 1)
  - Check 4 covers positions 4, 5, 6, 7 (3rd LSB is 1)

_ _ _ _ _ _ _

1 2 3 4 5 6 7

Hamming Code (3)

- Example: data=0101, 3 check bits
  - 7 bit code, check bit positions 1, 2, 4
  - Check 1 covers positions 1, 3, 5, 7
  - Check 2 covers positions 2, 3, 6, 7
  - Check 4 covers positions 4, 5, 6, 7

$$\begin{array}{ccccccc} \underline{0} & \underline{1} & 0 & \underline{0} & 1 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \longrightarrow$$

$p_1 = 0+1+1 = 0$, $p_2 = 0+0+1 = 1$, $p_4 = 1+0+1 = 0$

Hamming Code (4)

- To decode:
  - Recompute check bits (with parity sum including the check bit)
  - Arrange as a binary number
  - Value (syndrome) tells error position
  - Value of zero means no error
  - Otherwise, flip bit to correct

Hamming Code (5)

- Example, continued

$\longrightarrow \underline{0}\ \underline{1}\ 0\ \underline{0}\ 1\ 0\ 1$

1 2 3 4 5 6 7

p_1= p_2=

p_4=

Syndrome =

Data =

Hamming Code (6)

- Example, continued

$$\longrightarrow \underline{0}\ \underline{1}\ 0\ \underline{0}\ 1\ 0\ 1$$

1 2 3 4 5 6 7

p_1= 0+0+1+1 = 0, p_2= 1+0+0+1 = 0,
p_4= 0+1+0+1 = 0

Syndrome = 000, no error
Data = 0 1 0 1

Hamming Code (7)

- Example, continued

$$\longrightarrow \underline{0}\ \ \underline{1}\ \ 0\ \ \underline{0}\ \ 1\ \ {\color{red}1}\ \ 1$$

$$\ \ \ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7$$

p_1= p_2=

p_4=

Syndrome =

Data =

Hamming Code (8)

- Example, continued

$\longrightarrow \underline{0}\ \ \underline{1}\ \ 0\ \ \underline{0}\ \ 1\ \ 1\ \ 1$

$\ \ \ \ \ \ \ \ \ \ 1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7$

p_1= 0+0+1+1 = 0, p_2= 1+0+1+1 = 1,
p_4= 0+1+1+1 = 1

Syndrome = 1 1 0, flip position 6
Data = 0 1 0 1 (correct after flip!)

Hamming Code (3)

- Example: bad message 0100111
  - 7 bit code, check bit positions 1, 2, 4
  - Check 1 covers positions 1, 3, 5, 7
  - Check 2 covers positions 2, 3, 6, 7
  - Check 4 covers positions 4, 5, 6, 7

$$\begin{array}{ccccccc} 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \longrightarrow$$

$p_1 = 0+0+1+1 = 0$, $p_2 = 1+0+1+1 = 1$, $p_4 = 0+1+1+1 = 1$

Hamming Code (3)

- Example: bad message 0100111
  - 7 bit code, check bit positions 1, 2, 4
  - Check 1 covers positions 1, 3, 5, 7
  - Check 2 covers positions 2, 3, 6, 7
  - Check 4 covers positions 4, 5, 6, 7

0	1	0	0	1	1	1	→
1	2	3	4	5	6	7	

$p_1 = 0+0+1+1 = 0$, $p_2 = 1+0+1+1 = 1$, $p_4 = 0+1+1+1 = 1$

HW-2

Suppose the following message has been sent:

466f726f757a616e

What is the internet checksum of this message? Please give your answer in hex and with lowercase letters and with no leading 0x

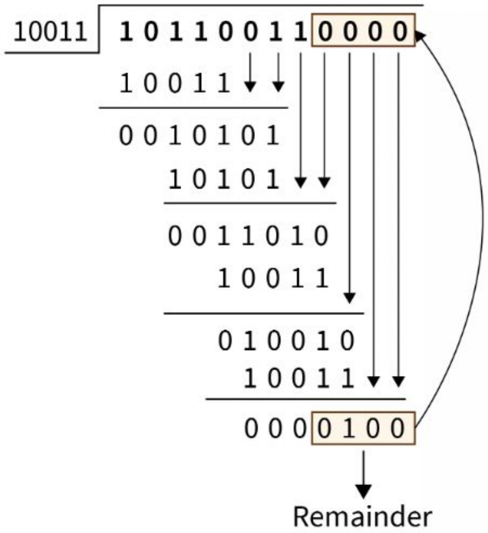
HW2

Let us see how to generate the CRC bits that are appended to the original data. Given that the data stream is 10110011 and the generator polynomial is

x^4+x+1

HW2

Step 1: Append the R number of 0 bits to the end of the data stream, where R is the highest degree of the polynomial. In our case, the value of R is 4 as the highest degree of generator polynomial function is 4 ($x^4 + x + 1$). So, our dividend data will be 10110011 + 0000 = 101100110000. Now, we will perform the division by dividing the input stream with the generator polynomial to generate CRC bits. The divisor in our case will be 10011 (i.e., $1.x^4 + 0.x^3 + 0.x^2 + 1.x + 1$).


10011
1 0 1 1 0 0 1 1 0 0 0 0
1 0 0 1 1
0 0 1 0 1 0 1
1 0 1 0 1
0 0 1 1 0 1 0
1 0 0 1 1
0 1 0 0 1 0
1 0 0 1 1
0 0 0 0 1 0 0
Remainder

Thank You