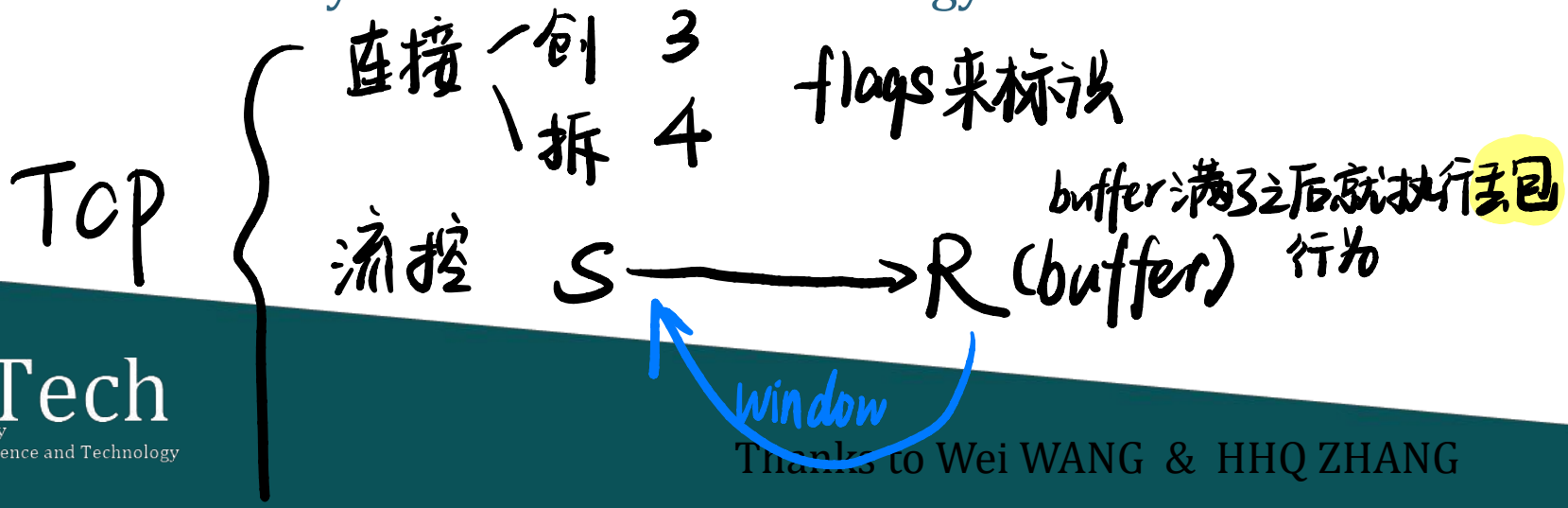
CS 305 Lab Tutorial
Lab 7 UDP TCP

MSS 最大报文长度，用于将应用层的包分片 Sequece Number

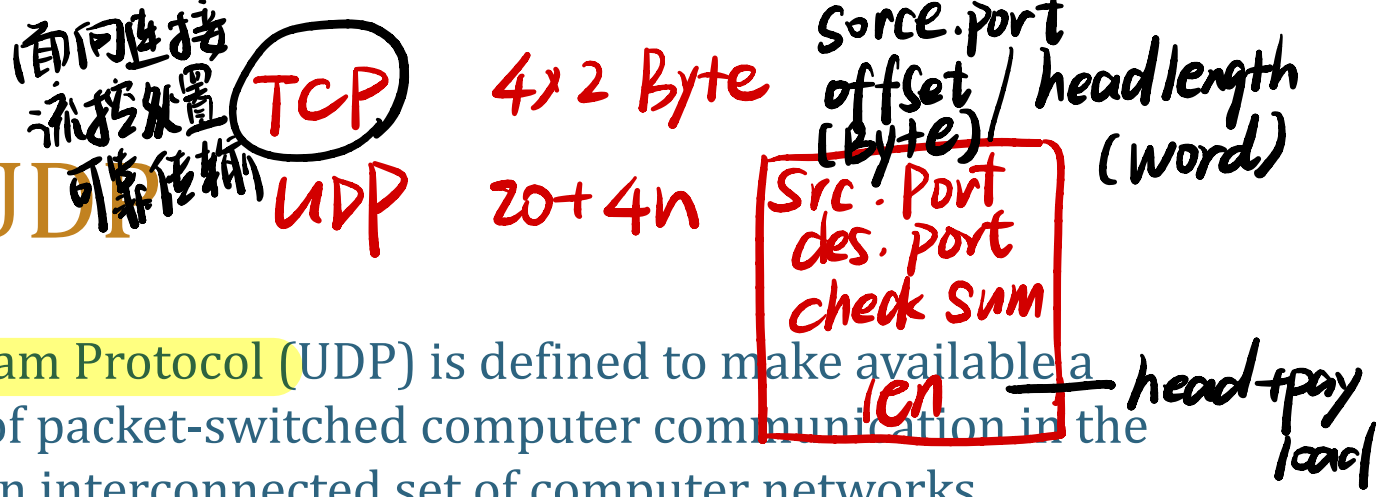
Dept. Computer Science and Engineering
Southern University of Science and Technology



SUSTech
Southern University of Science and Technology

Thanks to Wei WANG & HHQ ZHANG

Part A. UDP



This User Datagram Protocol (UDP) is defined to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks.

UDP assumes that the Internet Protocol (IP) is used as the underlying protocol.
UDP is transaction oriented, and **delivery and duplicate protection are NOT guaranteed.**

https://tools.ietf.org/html/rfc768

a UDP segment(1)

No.	Time	Source	Destination	Protocol	Length	Info
2732	17.881663	10.21.3.80	172.18.1.92	DNS	83	Standard query 0x0004 A www.sustc.edu.cn.edu.cn
2733	17.924398	172.18.1.92	10.21.3.80	DNS	83	Standard query response 0x0004 Server failure A

```
> Frame 2732: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface 0
> Ethernet II, Src: IntelCor_5c:69:58 (90:61:ae:5c:69:58), Dst: JuniperN_aa:6d:c3 (2c:21:31:aa:6d:c3)
v Internet Protocol Version 4, Src: 10.21.3.80, Dst: 172.18.1.92
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 69
    Identification: 0x4ca7 (19623)
  > Flags: 0x0000
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x732e [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.21.3.80
    Destination: 172.18.1.92
v User Datagram Protocol, Src Port: 64176, Dst Port: 53
    Source Port: 64176
    Destination Port: 53
    Length: 49
    Checksum: 0xc67f [unverified]
    [Checksum Status: Unverified]
    [Stream index: 468]
> Domain Name System (query)
```

UDP segment is the payload of IP package

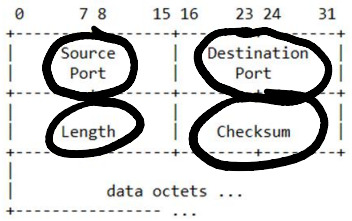
UDP is identified by 17 in protocol field of IP package

header 8+41 – payload

2^{16}

> wireshark 分析结果

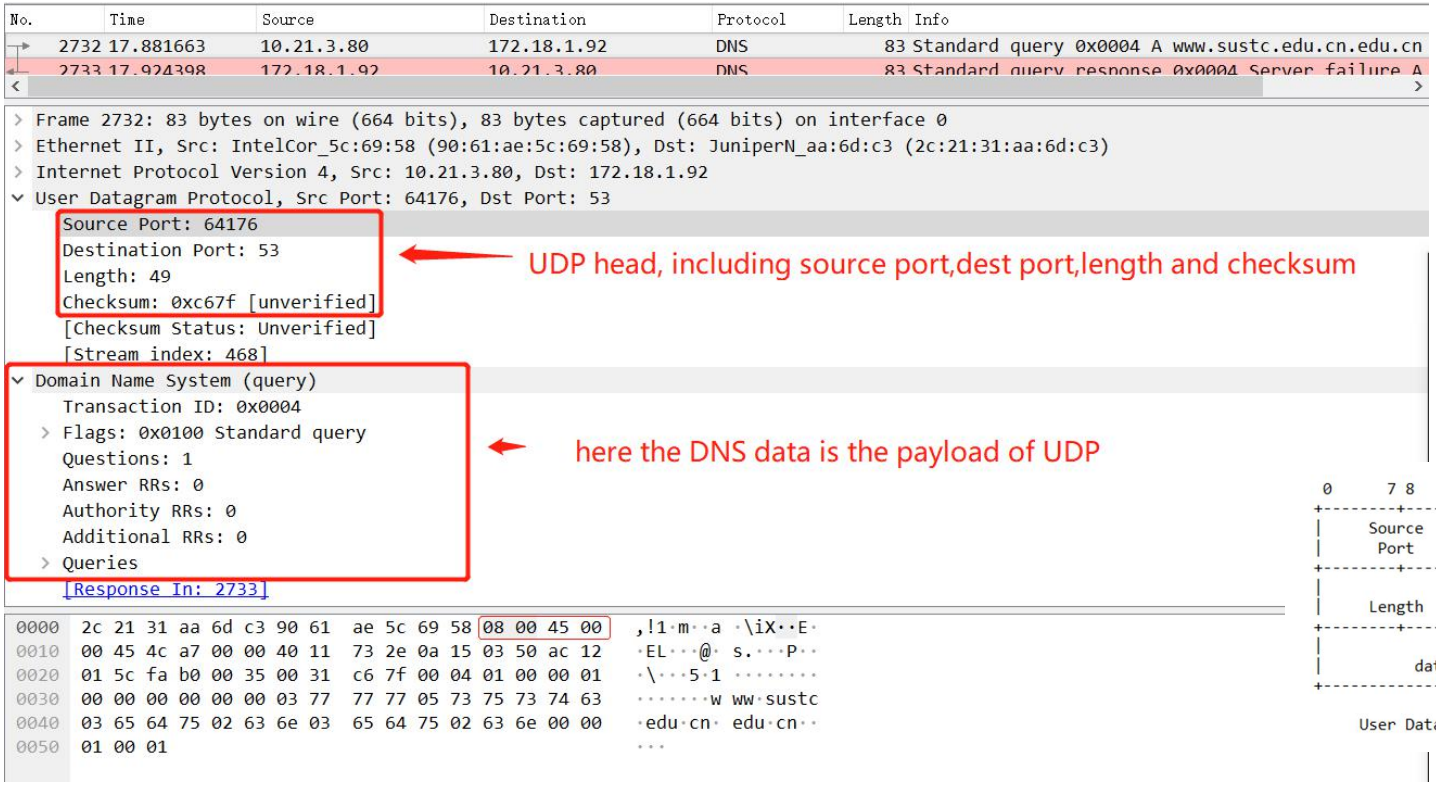
2 Bytes x4



User Datagram Header Format

SUSTech
Southern University of Science and Technology

a UDP segment(2)



While invoke an DNS query, this session is using UDP as transport protocol

You can use ‘nslookup’ or ‘dig’ to invoke an DNS query

Part B. TCP

TCP a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and in interconnected systems of such networks.

TCP must **recover from data that is damaged, lost, duplicated, or delivered out of order** by the Internet communication system.

- Ports
- connections
- Flow control
- Reliability

https://tools.ietf.org/html/rfc793

跟踪流：由 Ip & port 共同确认ID

Part B.1 A TCP connection

tcp.stream eq 0

No.	Time	Source	Destination	Protoc	Info
4	0.350305	192.168.88.149	14.215.177.39	TCP	60920 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	0.448978	14.215.177.39	192.168.88.149	TCP	80 → 60920 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1452 WS=32 SACK_PERM=1
6	0.449087	192.168.88.149	14.215.177.39	TCP	60920 → 80 [ACK] Seq=1 Ack=1 Win=66560 Len=0
7	0.449211	192.168.88.149	14.215.177.39	HTTP	HEAD / HTTP/1.1
8	0.487134	14.215.177.39	192.168.88.149	TCP	80 → 60920 [ACK] Seq=1 Ack=79 Win=24832 Len=0
9	0.493653	14.215.177.39	192.168.88.149	HTTP	HTTP/1.1 200 OK
10	0.497383	192.168.88.149	14.215.177.39	TCP	60920 → 80 [FIN, ACK] Seq=79 Ack=333 Win=66304 Len=0
12	0.563547	14.215.177.39	192.168.88.149	TCP	80 → 60920 [ACK] Seq=333 Ack=80 Win=24832 Len=0
13	0.566737	14.215.177.39	192.168.88.149	TCP	80 → 60920 [FIN, ACK] Seq=333 Ack=80 Win=24832 Len=0
14	0.566805	192.168.88.149	14.215.177.39	TCP	60920 → 80 [ACK] Seq=80 Ack=334 Win=66304 Len=0

互相交换参数信息

connection establish

仅为 tcp payload

http over tcp

忽略，因为seq主要

connection close

留给上层作解析

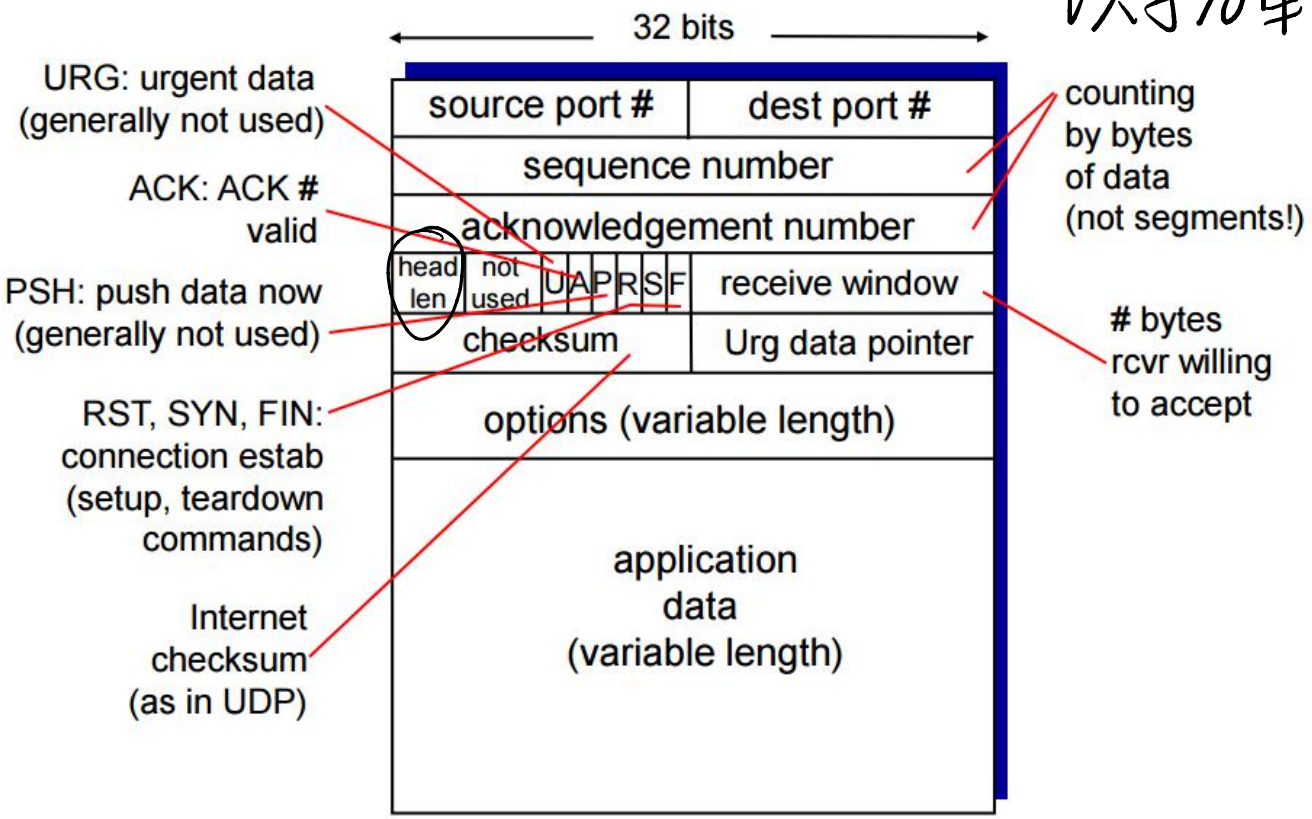
Source	Destination	Protoc	Info
192.168.88.149	14.215.177.39	TCP	60920 → 80

Source IP:192.168.88.149 port： 60920
Destination IP:14.215.177.39 port:80

Tips: Using command ‘curl’ to invoke a http request which using TCP for transport
For example: curl –I www.baidu.com

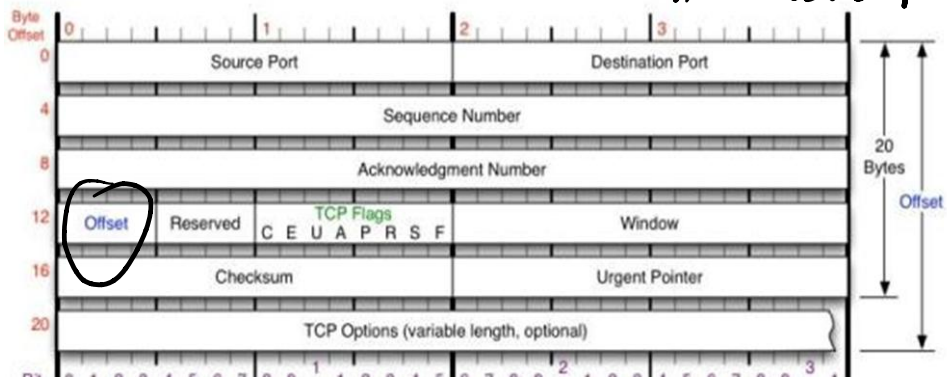
Part B.2 TCP segment structure

以字为单位



'Header len'/'offset' field in TCP header

以word为单位




```
Transmission Control Protocol, Src Port: 54861, Dst Port: 80, Seq: 1, Ack: 1, Len: 0
  Source Port: 54861
  Destination Port: 80
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 1    (relative sequence number)
  [Next sequence number: 1    (relative sequence number)]
  Acknowledgment number: 1    (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
> Flags: 0x010 (ACK)
  Window size value: 256
  [Calculated window size: 65536]
  [Window size scaling factor: 256]
  Checksum: 0x13ef [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
```

> 2 bytes each

head length is 20 byte
while there's no options

Data Offset: 4 bits

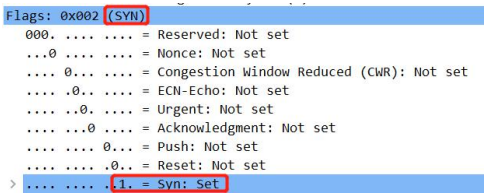
- The number of 32 bit words in the TCP Header. This indicates where the data begins.
- The TCP header (even one including options) is an integral number of 32 bits long.

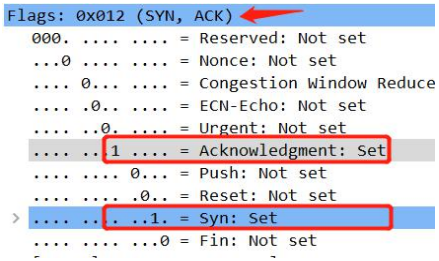
```
Transmission Control Protocol, Src Port: 54861, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 54861
  Destination Port: 80
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 0    (relative sequence number)
  [Next sequence number: 0    (relative sequence number)]
  Acknowledgment number: 0
  1000 .... = Header Length: 32 bytes (8)
> Flags: 0x002 (SYN)
  Window size value: 64240
  [Calculated window size: 64240]
  Checksum: 0x5335 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
> Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, N
```

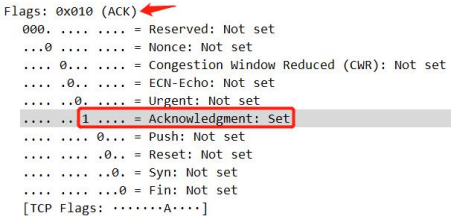
32 bytes= 8*4bytes

32bytes= 20(default length) +12 (options length)

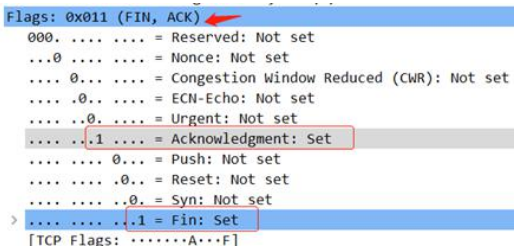
'Flags' in TCP header

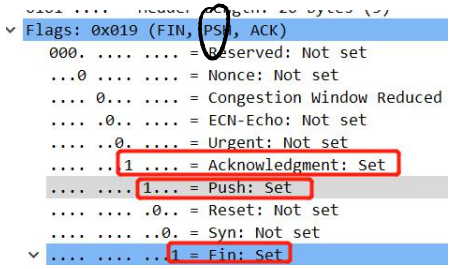






Control Bits:
URG: Urgent Pointer field significant
ACK: Acknowledgment field significant
PSH: Push Function 一不做缓存
RST: Reset the connection
SYN: Synchronize sequence numbers
FIN: No more data from sender





Tips in Wireshark: Using 'tcp.flags.xxx==1' as filter to view the corresponding package
While xxx is the name of the flag, such as tcp.flags.syn==1

'Sequence number' and 'Ack number'(1)

Transmission is made reliable via the use of **sequence numbers** and **acknowledgments**.

- The sequence number **of the first octet of data in a segment** is transmitted with that segment and is called the segment **sequence number**.
- Segments also carry an **acknowledgment number** which is the sequence number of the next expected data octet of transmissions in the reverse direction.

When the TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer;

- when the acknowledgment for that data is received, the segment is deleted from the queue.
- If the acknowledgment is not received before the timer runs out, the segment is retransmitted.

An acknowledgment by TCP does not guarantee that the data has been delivered to the end user, but only that the receiving TCP has taken the responsibility to do so.

https://tools.ietf.org/html/rfc793

'Sequence number' and 'Ack number'(2)

```
Transmission Control Protocol, Src Port: 80, Dst Port: 54861, Seq: 81761, Ack: 333, Len: 1460
    Source Port: 80
    Destination Port: 54861
    [Stream index: 2]
    [TCP Segment Len: 1460]
    Sequence number: 81761    (relative sequence number)
    [Next sequence number: 83221    (relative sequence number)]
    Acknowledgment number: 333    (relative ack number)
```

No.	Time	Source	Destination	Protoc	Info
234	10.752731	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=81761 Win=55296 Len=0
235	11.462632	128.119.245.12	192.168.88.149	TCP	80 → 54861 [ACK] Seq=81761 Ack=333 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
236	11.463266	128.119.245.12	192.168.88.149	TCP	80 → 54861 [ACK] Seq=83221 Ack=333 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
237	11.463358	192.168.88.149	128.119.245.12	TCP	54861 → 80 [ACK] Seq=333 Ack=84681 Win=52480 Len=0

54861->80: seq = 333 len=0

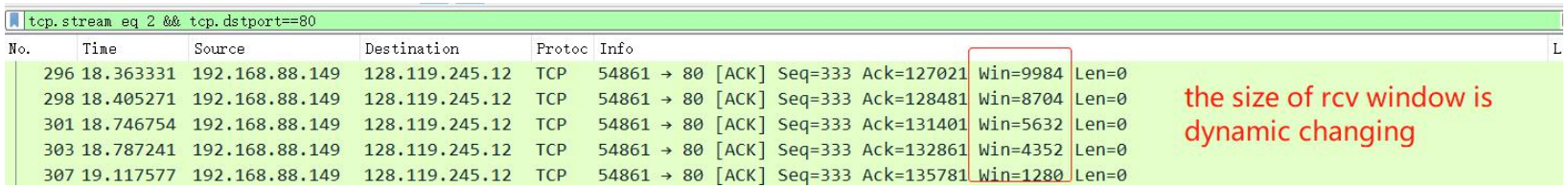
80->54861: ack=333+0 seq = 81761 len=1460

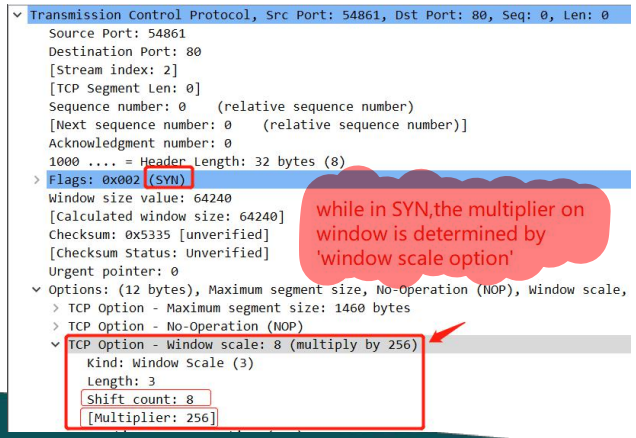
80->54861: ack=333+0 seq = 83221(81761+1460) len=1460

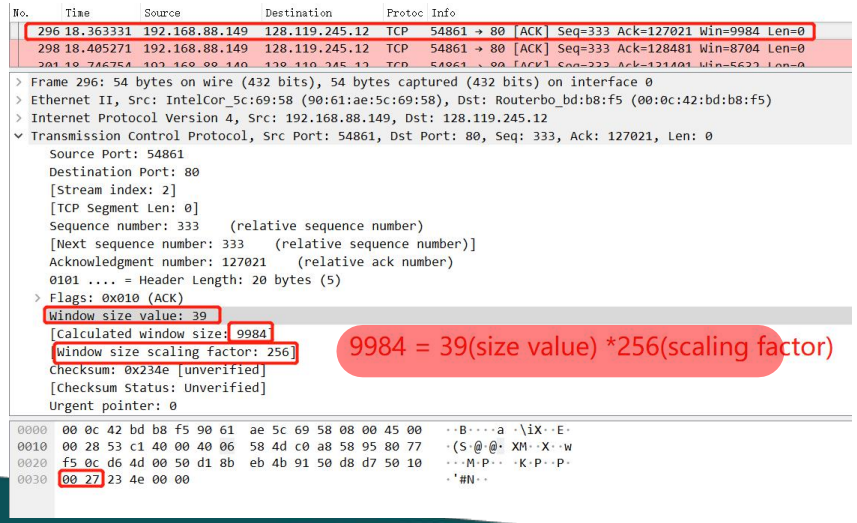
54861->80: Seq = 333(333+0) ack=84681(83221+1460) len=0

'Window' field in TCP header

- TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a "window" with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received.
- The window **indicates an allowed number of octets that the sender may transmit before receiving further permission**.







SUSTech
Southern University of Science and Technology

https://tools.ietf.org/html/rfc793

MSS，TCP playload的最大大小，是TCP用来限制application层最大的发送字节数。

'Options'(variable) in TCP header

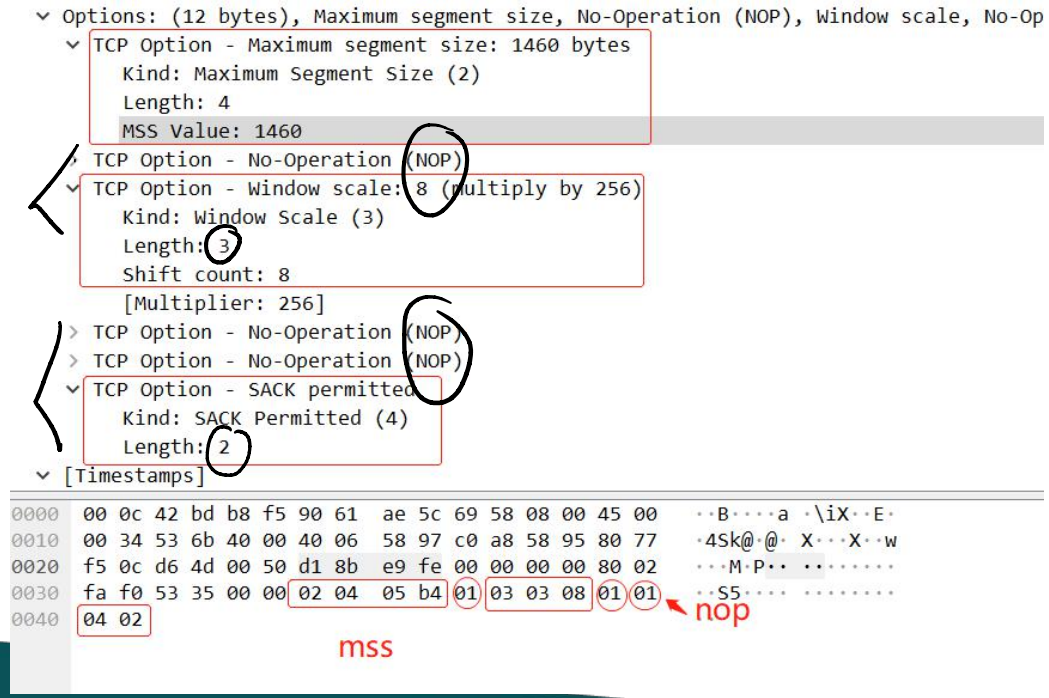
- May occupy **space at the end of the TCP header**
- **a multiple of 8 bits in length.**
- **No-operation** may be used between options, for example, to align the beginning of a subsequent option on a word boundary.

No-Operation

00000001
Kind=1

Maximum Segment Size

00000010	00000100	max seg size
Kind=2	Length=4	



SUSTech
Southern University of Science and Technology

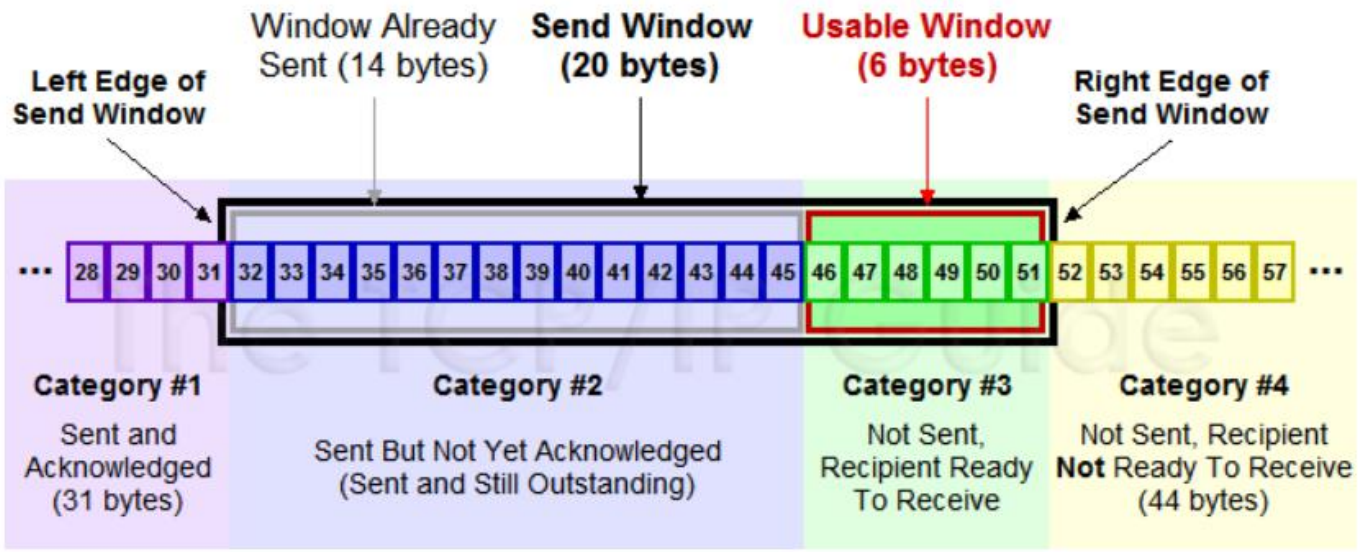
Practise 1

7.1 Select one UDP packet from your trace. From this packet,

- determine
  - 1) how many fields there are in the UDP header.
  - 2) the name and value of each fields in the UDP header.
  - 3) the length (in bytes) of each fields in the UDP header.
  - 4) What is the maximum number of bytes of a UDP packet ? (Hint: the answer to this question can be determined by your answer to 3) above)
  - 5) What is the largest possible destination port number? (Hint: same as the hint in 4) above.)
  - 6) What is the protocol ID for UDP in IP protocol?(Give your answer in both hexadecimal and decimal notation.)

7.2 Finish the question 4, 6, 7, 9, 10, 12 of Wireshark_TCP_v7.0.pdf

Part B.3 Sliding window system(1)

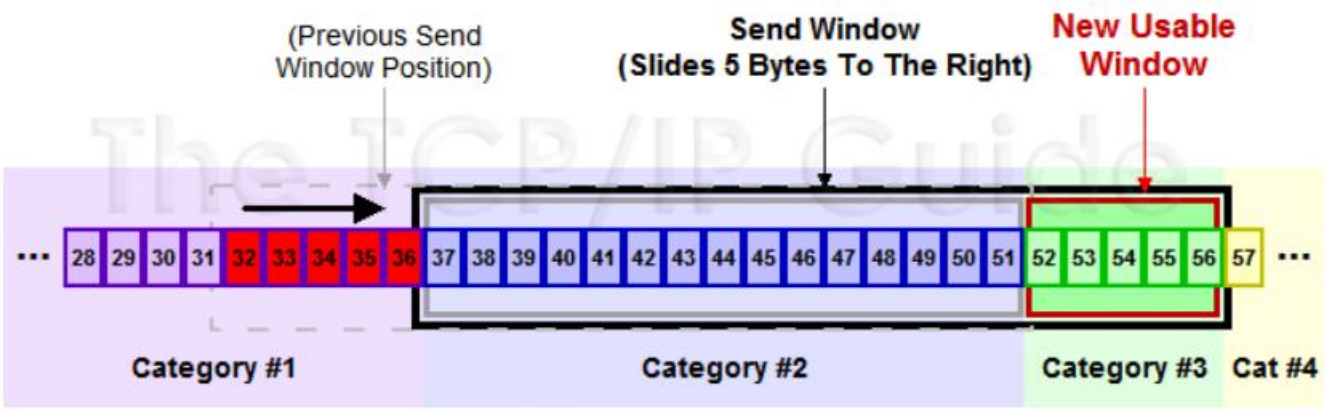


The ***send window*** is the key to the entire TCP sliding window system: it represents the maximum number of unacknowledged bytes a device is allowed to have outstanding at once.

The **usable window** is the number of bytes that the sender is still allowed to send at any point in time; it is equal to the size of the send window less the number of unacknowledged bytes already transmitted.

http://www.TCPipguide.com/free/t_TCPSlidingWindowAcknowledgmentSystemForDataTranspo-6.htm

Sliding window system(2)

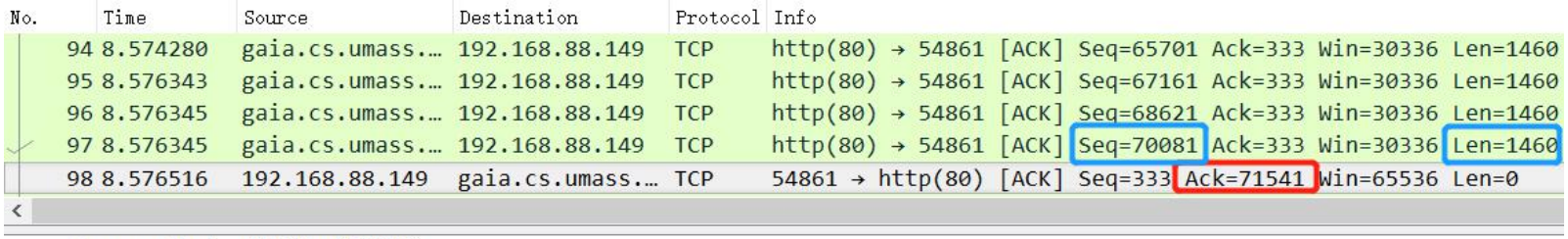


When the sending device **receives new acknowledgment**, it will be able to transfer some of the bytes from Category #2 to Category #1, since they have now been acknowledged. When it does so, something interesting will happen. Since five bytes have been acknowledged, and the window size didn't change, the sender is allowed to send five more bytes. In effect, the window shifts, or ***slides***, over to the right in the timeline.

At the same time five bytes move from Category #2 to Category #1, five bytes move from Category #4 to Category #3, **creating a new usable window for subsequent transmission**.

http://www.TCPipguide.com/free/t_TCPSlidingWindowAcknowledgmentSystemForDataTranspo-6.htm

ACK number, Sequence number, len



No.	Time	Source	Destination	Protocol	Info
94	8.574280	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=65701 Ack=333 Win=30336 Len=1460
95	8.576343	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=67161 Ack=333 Win=30336 Len=1460
96	8.576345	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=68621 Ack=333 Win=30336 Len=1460
97	8.576345	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=70081 Ack=333 Win=30336 Len=1460
98	8.576516	192.168.88.149	gaia.cs.umass.…	TCP	54861 → http(80) [ACK] Seq=333 Ack=71541 Win=65536 Len=0

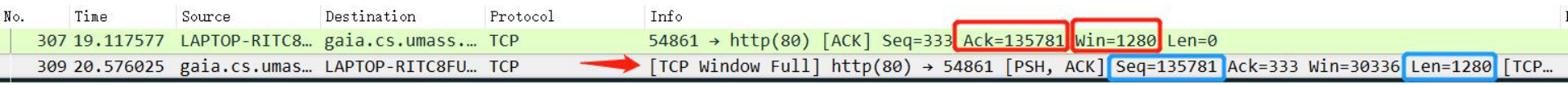
```
Source Port: 54861 (54861)
Destination Port: http (80)
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 333    (relative sequence number)
[Next sequence number: 333    (relative sequence number)]
Acknowledgment number: 71541    (relative ack number)
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
   .... .... ..0. = Syn: Not set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····A····]
```

ack_num (71541) =
seq (70081) +len (1460)

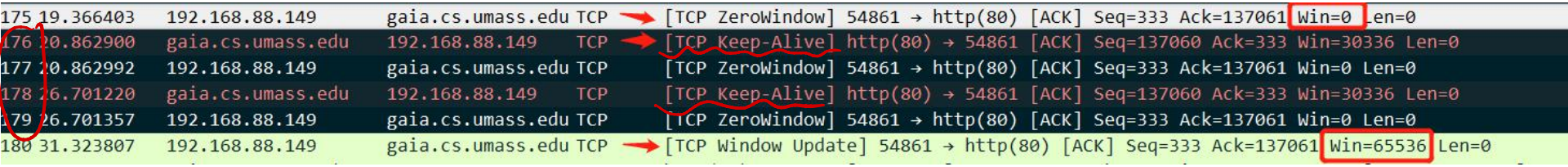
Changes of window

While the size of usable window turn to be 0, it means the sender will not send any segment at this moment. Wireshark mark the segment with “[Tcp Window Full]”

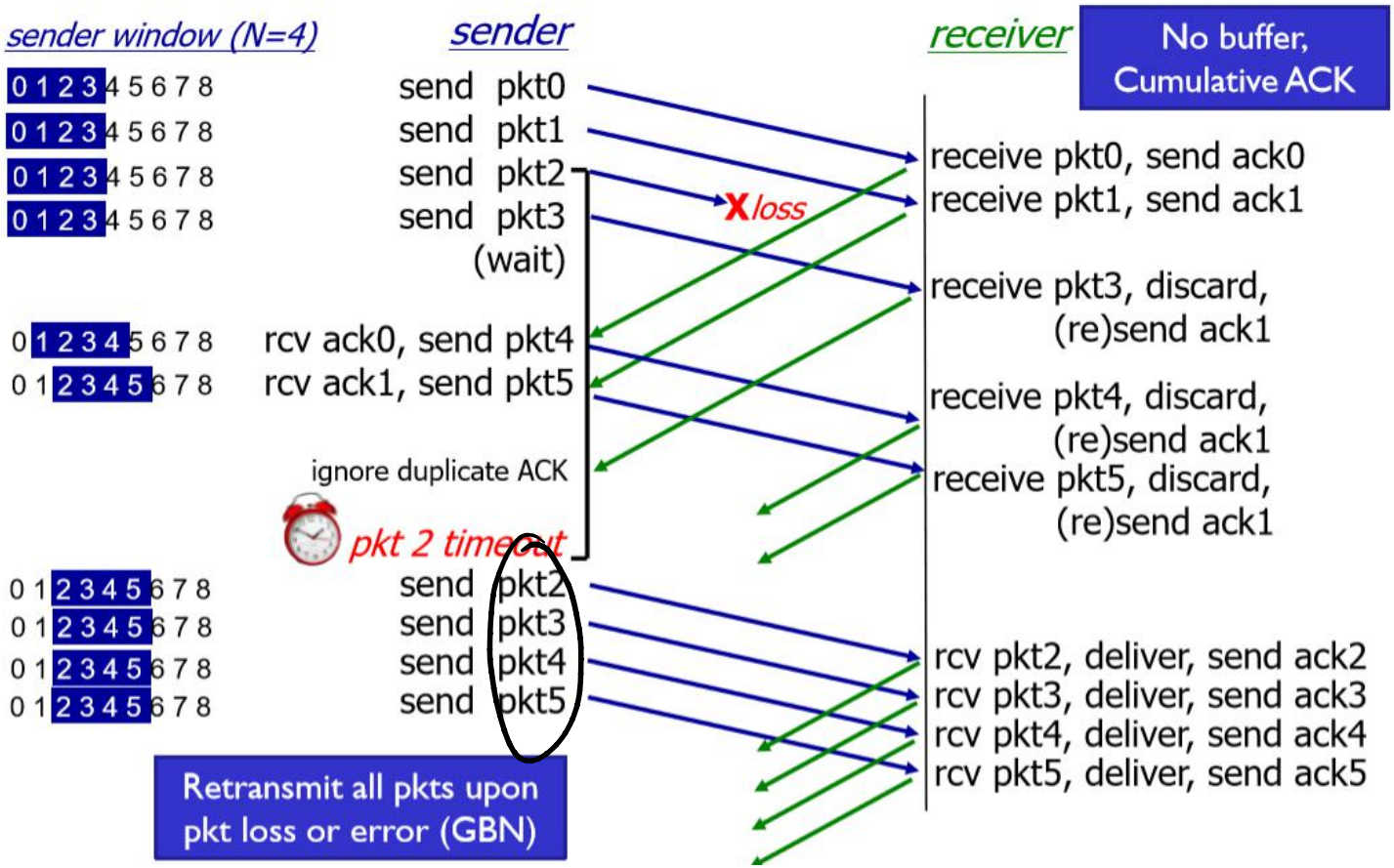
Seq（135781）+len（1280） – ack（135781） == win（1280）



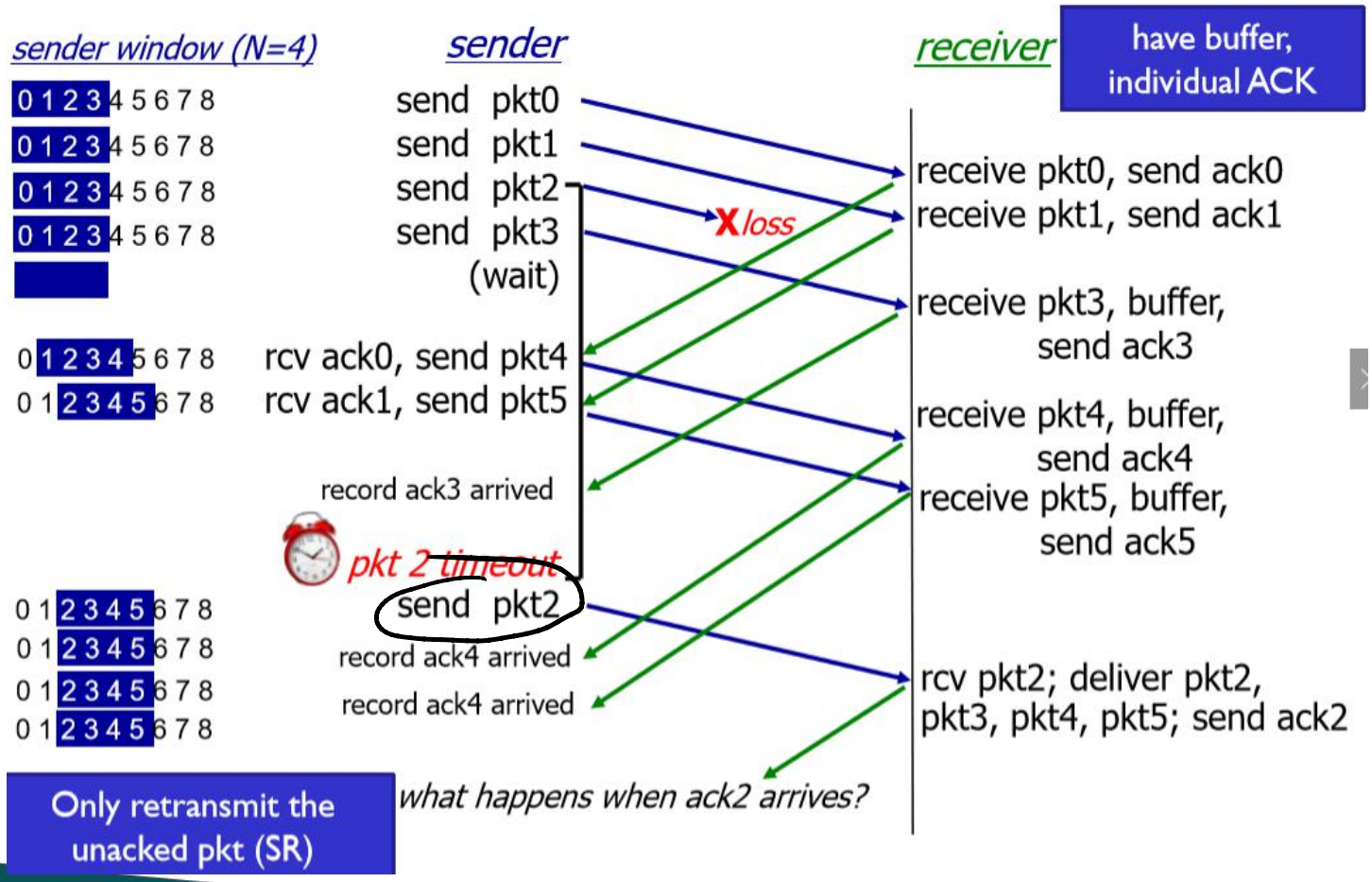
While the recv window turn to be zero, sender will stop to send packet, it will send “[TCP Keep-Alive]” to keep the TCP connection, waiting for the changing of recv window.



Part B.4 retransimission : GBN



SR


sender window (N=4)
sender
receiver
have buffer, individual ACK
0 1 2 3 4 5 6 7 8
send pkt0
send pkt1
send pkt2
send pkt3
(wait)
loss
receive pkt0, send ack0
receive pkt1, send ack1
receive pkt3, buffer, send ack3
rcv ack0, send pkt4
rcv ack1, send pkt5
receive pkt4, buffer, send ack4
receive pkt5, buffer, send ack5
record ack3 arrived
pkt 2 timeout
send pkt2
record ack4 arrived
record ack4 arrived
rcv pkt2; deliver pkt2, pkt3, pkt4, pkt5; send ack2
Only retransmit the unacked pkt (SR)
what happens when ack2 arrives?

TCP SACK

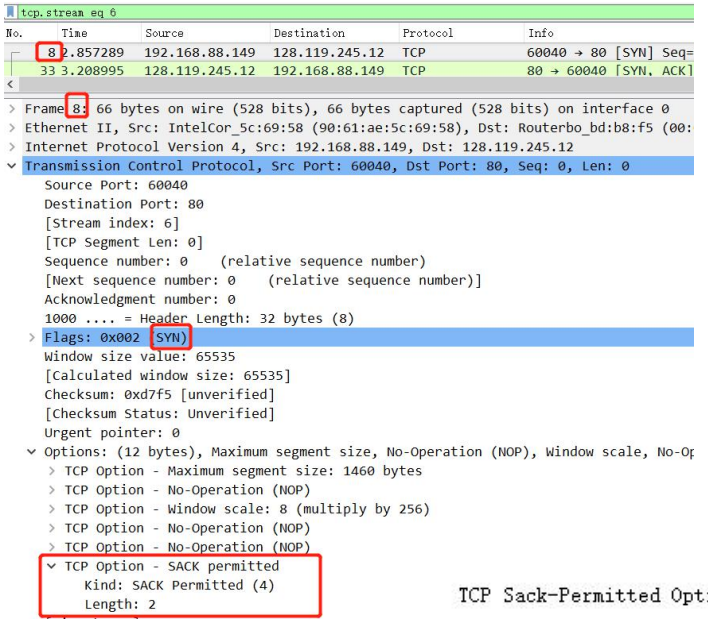
- A Selective Acknowledgment (**SACK**) mechanism, combined with a **selective repeat retransmission policy**, can help the sender retransmit only the missing data segments.
  - The receiving TCP **sends back SACK packets to the sender** informing the sender of data that has been received.
  - With selective acknowledgments, the data receiver can inform the sender about all segments that have arrived successfully, so the sender need **retransmit only the segments that have actually been lost.**

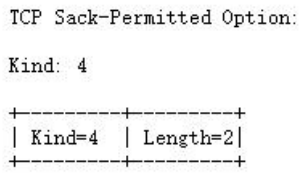
- The selective acknowledgment extension uses two TCP options:
  - **SACK-permitted** option
  - **SACK** option

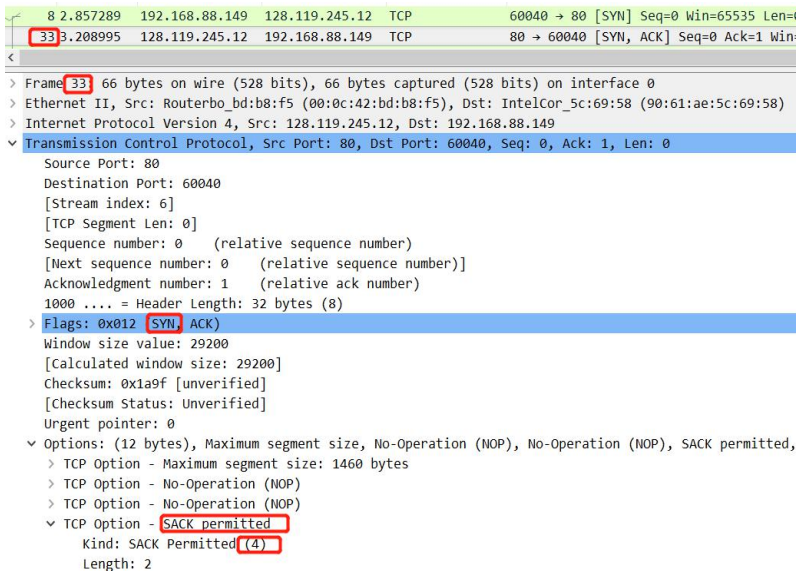
https://tools.ietf.org/html/rfc2018

SACK-permitted option

- "**SACK-permitted**", which may **be sent in a SYN segment** to indicate that the SACK option can be used once the connection is established.







Wireshark tips: TCP.option_kind==4

SACK option(1)

The **SACK option** is to be **sent by a data receiver to inform the data sender of non-contiguous blocks of data that have been received and queued**.

The data receiver awaits the receipt of data (perhaps by means of retransmissions) to fill the gaps in sequence space between received blocks.

When missing segments are received, **the data receiver acknowledges the data normally** by advancing the left window edge in the Acknowledgement Number Field of the TCP header.

The SACK option does not change the meaning of the Acknowledgement Number field.

This option contains a list of **some of the blocks of contiguous sequence space occupied by data that has been received and queued within the window.**

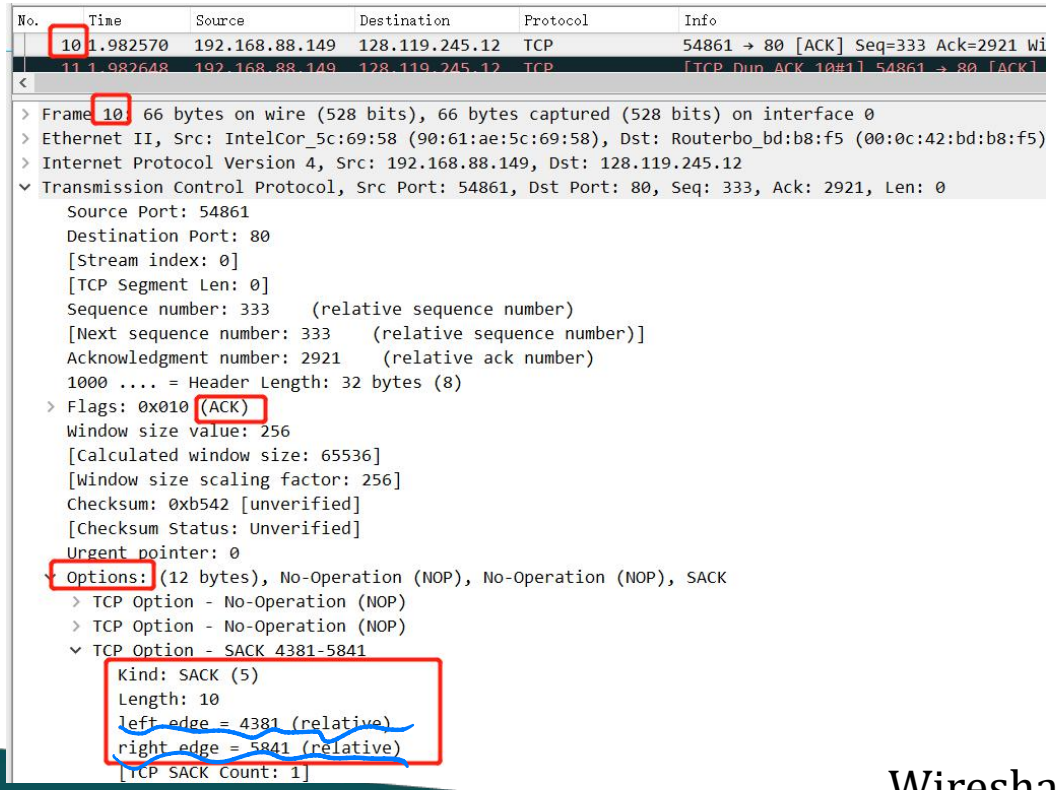
Each contiguous block of data queued at the data receiver is defined in the SACK option by **two 32-bit unsigned integers** in network byte order:

- **Left Edge** of Block This is the first sequence number of this block.
- **Right Edge** of Block This is the sequence number immediately following the last sequence number of this block.

Each block represents received bytes of data that are **contiguous and isolated**; that is, **the bytes just below the block, (Left Edge of Block - 1), and just above the block, (Right Edge of Block), have NOT been received**.

SACK option(2)

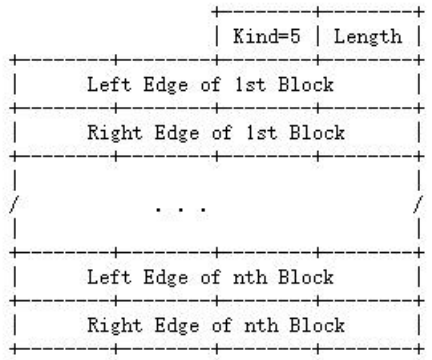
- SACK option , which may be sent over an established connection once permission has been given by SACK-permitted.



TCP SACK Option:

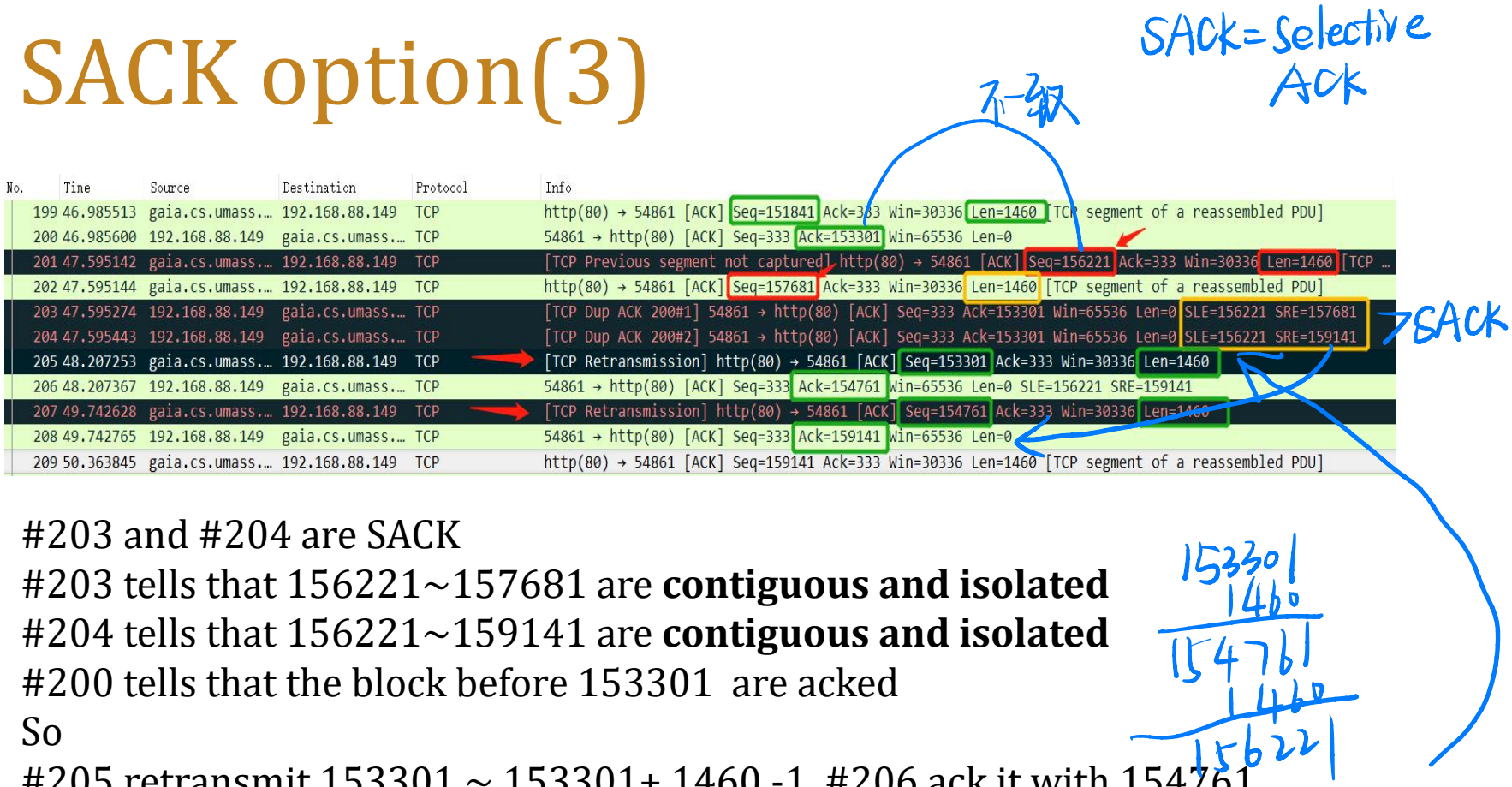
Kind: 5

Length: Variable



Wireshark tips: TCP.option_kind==5

SACK option(3)



No.	Time	Source	Destination	Protocol	Info
199	46.985513	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=151841 Ack=333 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
200	46.985600	192.168.88.149	gaia.cs.umass.…	TCP	54861 → http(80) [ACK] Seq=333 Ack=153301 Win=65536 Len=0
201	47.595142	gaia.cs.umass.…	192.168.88.149	TCP	[TCP Previous segment not captured] http(80) → 54861 [ACK] Seq=156221 Ack=333 Win=30336 Len=1460 [TCP …
202	47.595144	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=157681 Ack=333 Win=30336 Len=1460 [TCP segment of a reassembled PDU]
203	47.595274	192.168.88.149	gaia.cs.umass.…	TCP	[TCP Dup ACK 200#1] 54861 → http(80) [ACK] Seq=333 Ack=153301 Win=65536 Len=0 SLE=156221 SRE=157681
204	47.595443	192.168.88.149	gaia.cs.umass.…	TCP	[TCP Dup ACK 200#2] 54861 → http(80) [ACK] Seq=333 Ack=153301 Win=65536 Len=0 SLE=156221 SRE=159141
205	48.207253	gaia.cs.umass.…	192.168.88.149	TCP	[TCP Retransmission] http(80) → 54861 [ACK] Seq=153301 Ack=333 Win=30336 Len=1460
206	48.207367	192.168.88.149	gaia.cs.umass.…	TCP	54861 → http(80) [ACK] Seq=333 Ack=154761 Win=65536 Len=0 SLE=156221 SRE=159141
207	49.742628	gaia.cs.umass.…	192.168.88.149	TCP	[TCP Retransmission] http(80) → 54861 [ACK] Seq=154761 Ack=333 Win=30336 Len=1460
208	49.742765	192.168.88.149	gaia.cs.umass.…	TCP	54861 → http(80) [ACK] Seq=333 Ack=159141 Win=65536 Len=0
209	50.363845	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=159141 Ack=333 Win=30336 Len=1460 [TCP segment of a reassembled PDU]

#203 and #204 are SACK

#203 tells that 156221~157681 are **contiguous and isolated**

#204 tells that 156221~159141 are **contiguous and isolated**

#200 tells that the block before 153301 are acked

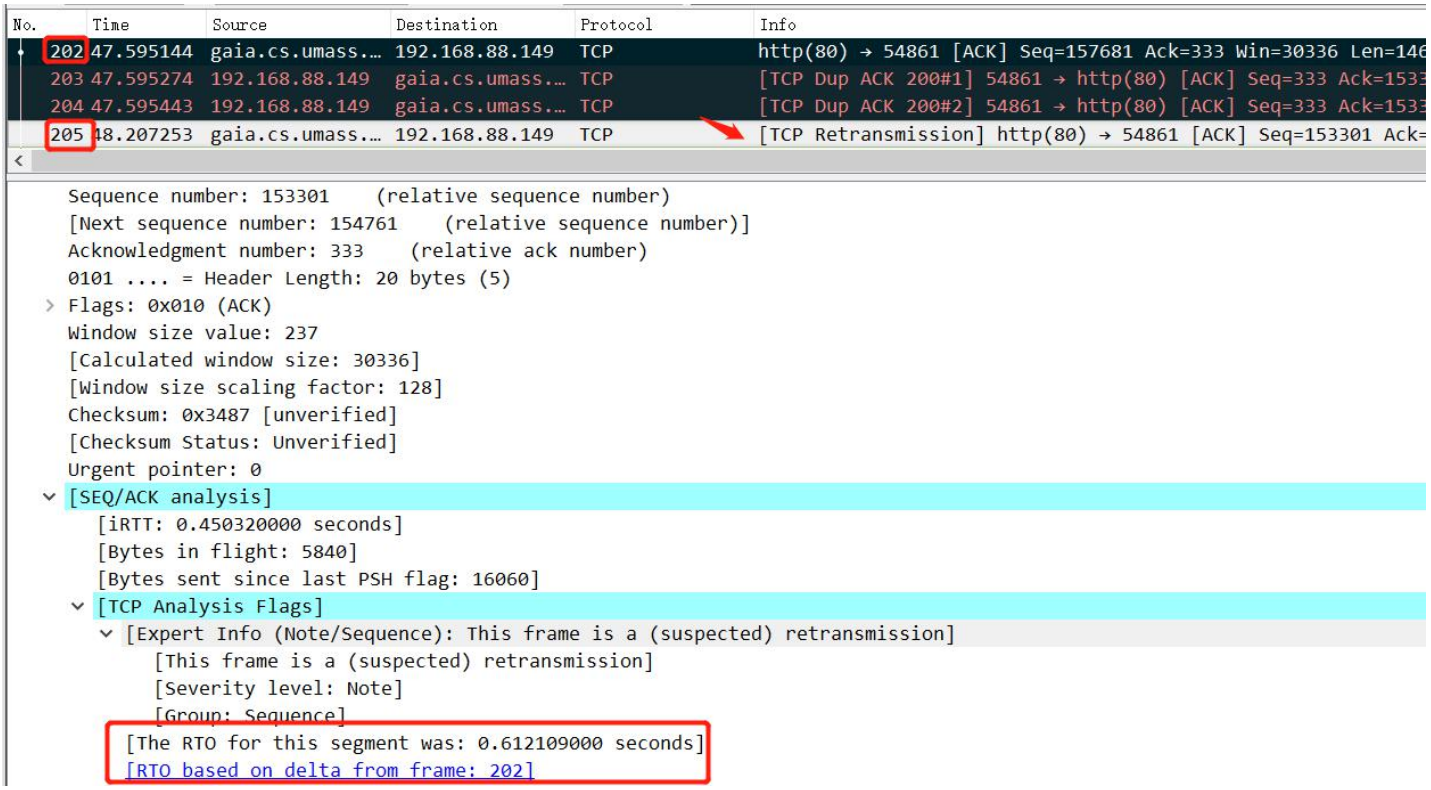
So

#205 retransmit 153301 ~ 153301+ 1460 -1 ,#206 ack it with 154761

#207 retransmit 154761 ~ 154761+1460 -1

#208 ack it with 159141 (for 157681~159140 are **contiguous but NOT isolated**)

Retransmission(1)

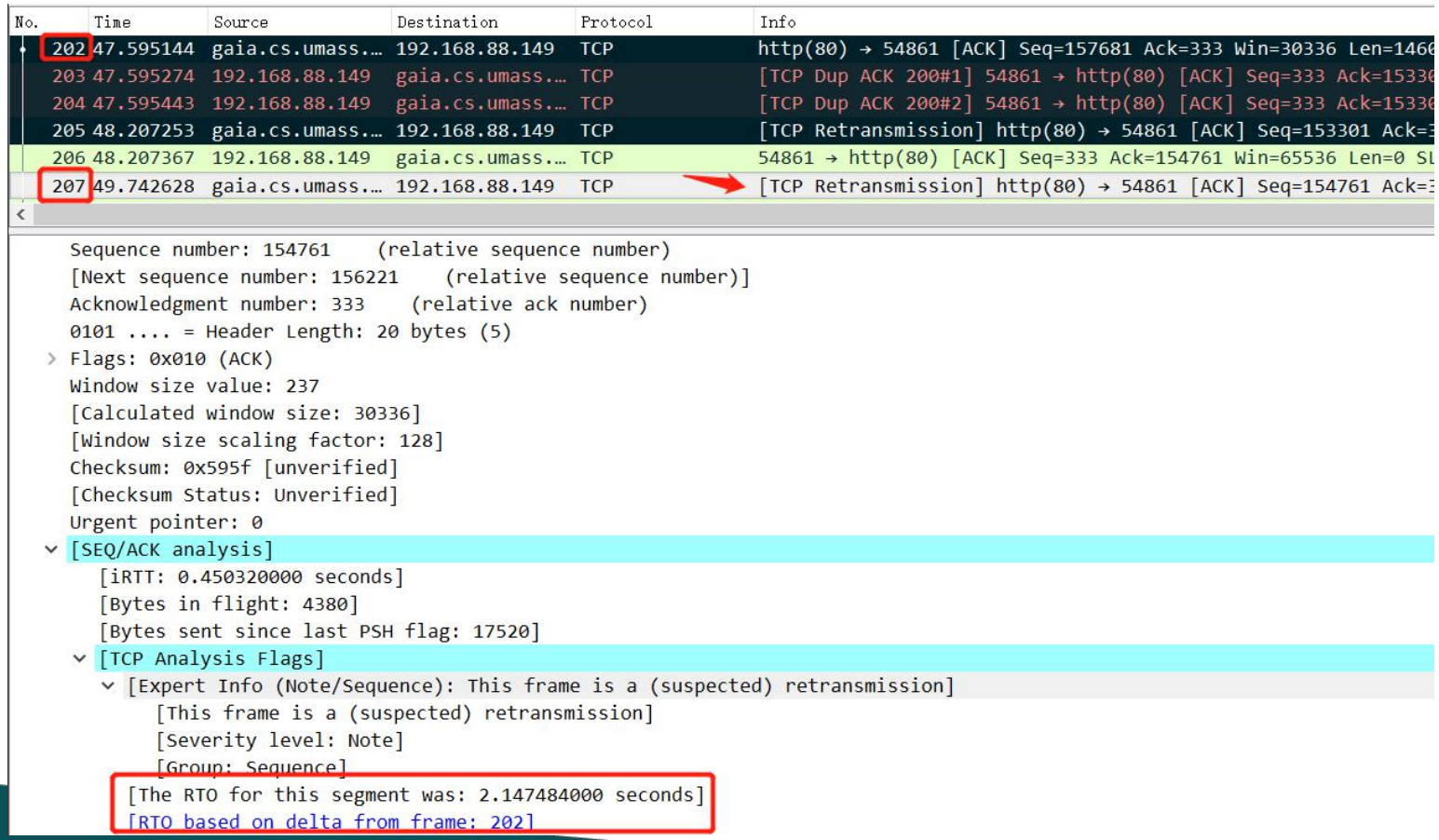


No.	Time	Source	Destination	Protocol	Info
202	47.595144	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=157681 Ack=333 Win=30336 Len=146
203	47.595274	192.168.88.149	gaia.cs.umass.…	TCP	[TCP Dup ACK 200#1] 54861 → http(80) [ACK] Seq=333 Ack=1533
204	47.595443	192.168.88.149	gaia.cs.umass.…	TCP	[TCP Dup ACK 200#2] 54861 → http(80) [ACK] Seq=333 Ack=1533
205	48.207253	gaia.cs.umass.…	192.168.88.149	TCP	[TCP Retransmission] http(80) → 54861 [ACK] Seq=153301 Ack=

```
Sequence number: 153301    (relative sequence number)
[Next sequence number: 154761    (relative sequence number)]
Acknowledgment number: 333    (relative ack number)
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window size value: 237
[Calculated window size: 30336]
[Window size scaling factor: 128]
Checksum: 0x3487 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
[SEQ/ACK analysis]
    [iRTT: 0.450320000 seconds]
    [Bytes in flight: 5840]
    [Bytes sent since last PSH flag: 16060]
    [TCP Analysis Flags]
        [Expert Info (Note/Sequence): This frame is a (suspected) retransmission]
            [This frame is a (suspected) retransmission]
            [Severity level: Note]
            [Group: Sequence]
        [The RTO for this segment was: 0.612109000 seconds]
        [RTO based on delta from frame: 202]
```

While RTO timeout , retransmission is triggered

Retransmission(2)



No.	Time	Source	Destination	Protocol	Info
202	47.595144	gaia.cs.umass.…	192.168.88.149	TCP	http(80) → 54861 [ACK] Seq=157681 Ack=333 Win=30336 Len=1460
203	47.595274	192.168.88.149	gaia.cs.umass.…	TCP	[TCP Dup ACK 200#1] 54861 → http(80) [ACK] Seq=333 Ack=15336
204	47.595443	192.168.88.149	gaia.cs.umass.…	TCP	[TCP Dup ACK 200#2] 54861 → http(80) [ACK] Seq=333 Ack=15336
205	48.207253	gaia.cs.umass.…	192.168.88.149	TCP	[TCP Retransmission] http(80) → 54861 [ACK] Seq=153301 Ack=
206	48.207367	192.168.88.149	gaia.cs.umass.…	TCP	54861 → http(80) [ACK] Seq=333 Ack=154761 Win=65536 Len=0 SL
207	49.742628	gaia.cs.umass.…	192.168.88.149	TCP	[TCP Retransmission] http(80) → 54861 [ACK] Seq=154761 Ack=

```
Sequence number: 154761    (relative sequence number)
[Next sequence number: 156221    (relative sequence number)]
Acknowledgment number: 333    (relative ack number)
0101 .... = Header Length: 20 bytes (5)
> Flags: 0x010 (ACK)
Window size value: 237
[Calculated window size: 30336]
[Window size scaling factor: 128]
Checksum: 0x595f [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
v [SEQ/ACK analysis]
    [iRTT: 0.450320000 seconds]
    [Bytes in flight: 4380]
    [Bytes sent since last PSH flag: 17520]
  v [TCP Analysis Flags]
    v [Expert Info (Note/Sequence): This frame is a (suspected) retransmission]
        [This frame is a (suspected) retransmission]
        [Severity level: Note]
        [Group: Sequence]
      [The RTO for this segment was: 2.147484000 seconds]
      [RTO based on delta from frame: 202]
```

Fast retransmission

- TCP may generate an immediate acknowledgment (a duplicate ACK) when an out-of-order segment is received. This duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order, and to tell it what sequence number is expected.
- Since TCP does not know whether a duplicate ACK is caused by a lost segment or just a reordering of segments, it waits for a small number of duplicate ACKs to be received.
  - It is assumed that if there is just a reordering of the segments, there will be only one or two duplicate ACKs before the reordered segment is processed, which will then generate a new ACK.
  - **If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost.**

- TCP then **performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire**.

https://tools.ietf.org/html/rfc2001

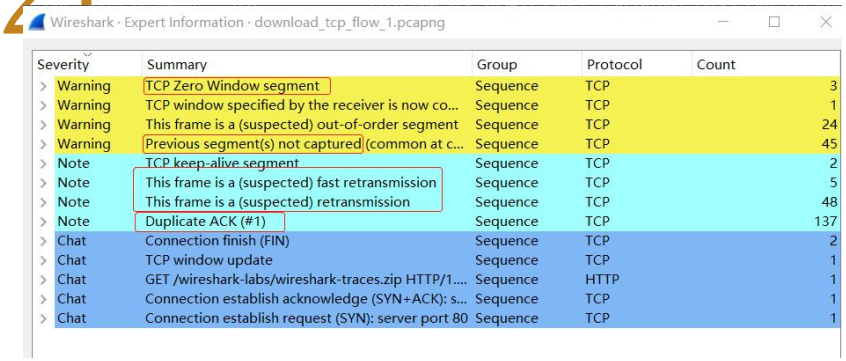
Tips on wireshark(1)

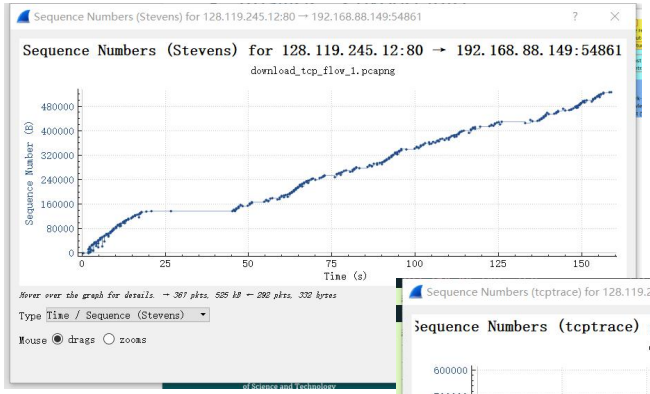
- How to get all the TCP segments of the TCP stream related to a http session in Wireshark:
  - 1st step: Find a HTTP packet in the HTTP session
  - 2nd step: right click the packet which will invoke a shortcut menu, then choose "follow ->TCP" in the shortcut menu
- How to find a special segment in a TCP stream
  - Using view filter in Wireshark
  - 1st step: if a TCP stream is filted, then a description such as "TCP.stream eq xx" (xx here is the id of this TCP stream) could be found in the view filter
  - 2nd step: in the same view filter, make a new filter rule description along with the original one,then press "Enter" key in your keyboard to make the new filter run
    - such as :

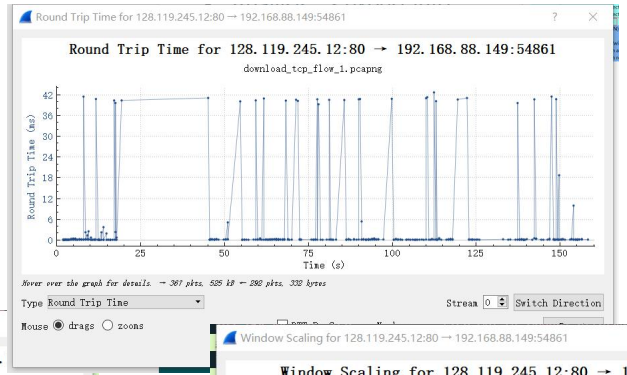
    TCP.stream eq 1 is the original one, to find a TCP zero window in this TCP stream,
    TCP.stream eq 1 && TCP.window_size_value==0 is the new view filter description

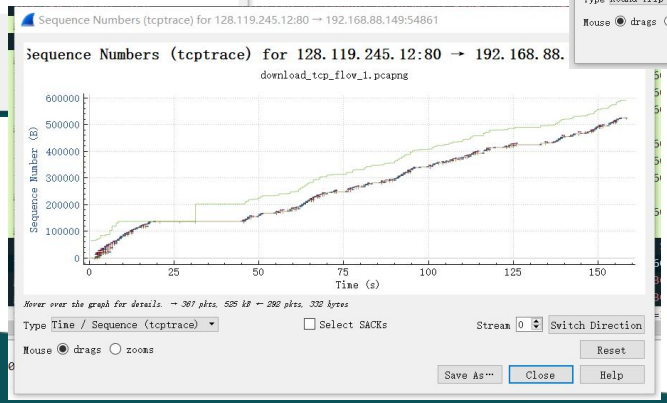
Tips on wireshark(2)

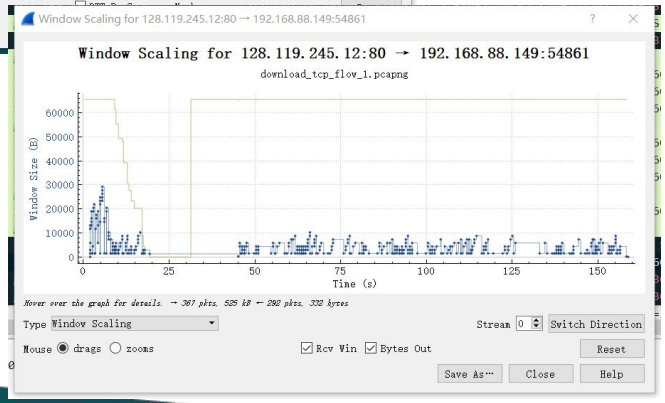
- Analysis -> expert info
- Statistic->TCP stream graphs











SUSTech
Southern University
of Science and Technology

Practise 2

7.3 Using Wireshark to capture and analysis the TCP stream

- Invoke a HTTP request to get http://gaia.cs.umass.edu/wiresharklabs/alice.txt
- Analysis the TCP stream
  - Any duplicate ack, what's the possible reason? last package
  - Any TCP segment with sack permit option and sack option
    - select a tcp package which contained a sack option, find the segment ranges which is acked in this sack option
  - Any TCP retransmission? Is it retransmission or fast retransmission?
  - Any window size 0 segment, what does it mean? If the window size is 0, what would happened next on this tcp connection? tcp.window_size==0 close and RST
  - Any TCP window full segment, what does it mean?

tcp.

window update

Tips while using Wireshark

1.if you want focus only on TCP while disable the HTTP analysis in wireshark

1) in menu: Analyze->Enabled Protocols.

2) Then uncheck the HTTP box and select OK

2. if you want to find the message include "POST" in TCP

in view filter using following rules：

tcp.segment_data contains "POST"

3. if you want to find the statistical information related to TCP

in menu: Statistics->TCP Stream Graph

4. Find if there is retransmit on TCP or not:

1) to check if there is 'Retransmission (suspected)' or 'tcp dup ack' or 'TCP Fast Retransmission' appears in the info items of packet list windows

2) expert info (analysis->expert info) may show you some hints

Practice：

1. find if there is a TCP segment whose window size is 0
2. Find the RTT value of a TCP segment

Tips while using Wireshark

1.if you want focus only on TCP while disable the HTTP analysis in wireshark

1) in menu: Analyze->Enabled Protocols.

2) Then uncheck the HTTP box and select OK

2. if you want to find the message include "POST" in TCP

in view filter using following rules：

tcp.segment_data contains "POST"

3. if you want to find the statistical information related to TCP

in menu: Statistics->TCP Stream Graph

4. Find if there is retransmit on TCP or not:

1) to check if there is 'Retransmission (suspected)' or 'tcp dup ack' or 'TCP Fast Retransmission' appears in the info items of packet list windows

2) expert info (analysis->expert info) may show you some hints

Practice：

1. find if there is a TCP segment whose window size is 0
2. Find the RTT value of a TCP segment

Tips while using Wireshark

1.if you want focus only on TCP while disable the HTTP analysis in wireshark

1) in menu: Analyze->Enabled Protocols.

2) Then uncheck the HTTP box and select OK

2. if you want to find the message include "POST" in TCP

in view filter using following rules:

tcp.segment_data contains "POST"

3. if you want to find the statistical information related to TCP

in menu: Statistics->TCP Stream Graph

4. Find if there is retransmit on TCP or not:

1) to check if there is 'Retransmission (suspected)' or 'tcp dup ack' or 'TCP Fast Retransmission' appears in the info items of packet list windows

2) expert info (analysis->expert info) may show you some hints

Practice:

zerowindow ≠ window=0

1. find if there is a TCP segment whose window size is 0
2. Find the RTT value of a TCP segment

Tips while using Wireshark

1.if you want focus only on TCP while disable the HTTP analysis in wireshark

1) in menu: Analyze->Enabled Protocols.

2) Then uncheck the HTTP box and select OK

2. if you want to find the message include "POST" in TCP

in view filter using following rules：

tcp.segment_data contains "POST"

3. if you want to find the statistical information related to TCP

in menu: Statistics->TCP Stream Graph

4. Find if there is retransmit on TCP or not:

1) to check if there is 'Retransmission (suspected)' or 'tcp dup ack' or 'TCP Fast Retransmission' appears in the info items of packet list windows

2) expert info (analysis->expert info) may show you some hints

Practice：

1. find if there is a TCP segment whose window size is 0
2. Find the RTT value of a TCP segment