Computer Networks ICS 651

- TCP delayed acks
- Bandwidth-Delay product
- TCP streams and push
- TCP header
- TCP options
- tcpdump and wireshark

TCP delayed acks

- however, in a two-way stream, an ack is piggybacked on every data segment going in the other direction
  - in such a situation, receivers don't need to send acks
  - and for fastest performance, should not send acks
- however, the network code doesn't know when the application will send more data
  - in a one-way stream of data, for fastest performance, a receiver should ack every segment immediately
- To optimize this tradeoff, a receiver should ack:
  - after receiving 2 MSS worth of data, or
  - 20ms after receiving the first data, if totaling to less than 2 MSS

Bandwidth-Delay product

- How big do I make the window?
- Suppose the receiver has a small buffer of size s = 512 bytes
  - it will set the window to size s
  - the sender is limited to s outstanding (unacked) bytes
- if the sender can send 100,000 bytes/second (800Kbps), and the round-trip delay is 10ms, the sender can only send 512 bytes 100 times per second, or at most 51,200 bytes/second
- if the sender can use a window equal to the bandwidth-delay product, s = 100KB/s x 0.01s = 1,000 bytes, it can send at full speed
- a window >= to the bandwidth-delay product allows sending at full speed, whereas
- a window < the bandwidth-delay product unnecessarily slows down the sender
- sending at full speed requires a large window on networks with large bandwidth-delay products, such as:
  - high-bandwidth networks such as gigabit and 10-gig ethernet, microwave and light links including fiber optic links, and
  - high-latency networks, especially networks with links over satellites

TCP window scaling

- the window field in the TCP header is 16 bits, so the largest window is 65,535 bytes
- this is not enough for full bandwidth on a 1ms (RTT) gigabit ethernet, with a bandwidth-delay product of 1Mb = 125KB
  - let alone on networks with higher bandwidth-delay products
- so TCP provides a window scaling option, sent with the SYN packet
- the option only takes effect if both sides send a window scaling option with their SYN packet
- if I send a window scale with a value of n, where 0 <= n <= 14, then I must divide the windows that I send by 2^n
- correspondingly, if I receive a window scale with a value of n, then I must multiply the windows that I receive by 2^n
- since n <= 14, the maximum window size is < 2^{30} bytes
- window scaling is defined in RFC 1323, "TCP Extensions for High Performance", which also provides protection against wrapped sequence numbers
  - on networks with high bandwidth-delay product there could relatively easily be over 4GB of data in transit at any given time, and without this extension, there could be multiple packets in transit at the same time with the same 32-bit TCP sequence number

TCP streams and push

- TCP actually has a segmentation bit: PSH, or push
- when the application "pushes" the data, that information could be conveyed all the way to the application at the other end
- if TCP can coalesce several user segments (each with PSH) into one TCP segment, that TCP segment can only carry one PSH bit
- so passing PSH to the application is optional, and TCP has no record boundaries
- so push is an advisory bit only: it encourages TCP (and the application) to send the data as promptly as possible

TCP header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                        TCP Header Format
```

TCP header fields

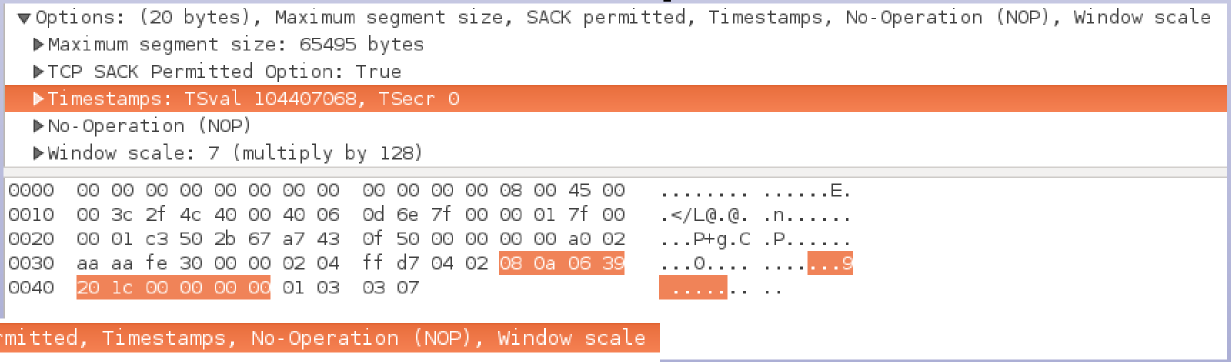
- Source and Destination port: demultiplexing
- Sequence and acknowledgement: reliable delivery
- Data Offset: header size, options
- Window: flow control
- Checksum: correctness
- Urgent Pointer: "special place" in the data stream

TCP header bits

- SYN: I want to establish a connection
- FIN: I will never again send data on this connection
- RST: kill this connection
- PSH: immediate delivery of this data is probably a good idea
- URG: the urgent pointer is valid
- ACK: the acknowledgement field is valid (set in all but the first SYN packet)
- ECE: this packet acknowledges a packet received with the IP "congestion experienced" bit set
- CWR: the sender of this packet has reduced its congestion window
- the last two bits will be discussed in the context of congestion control

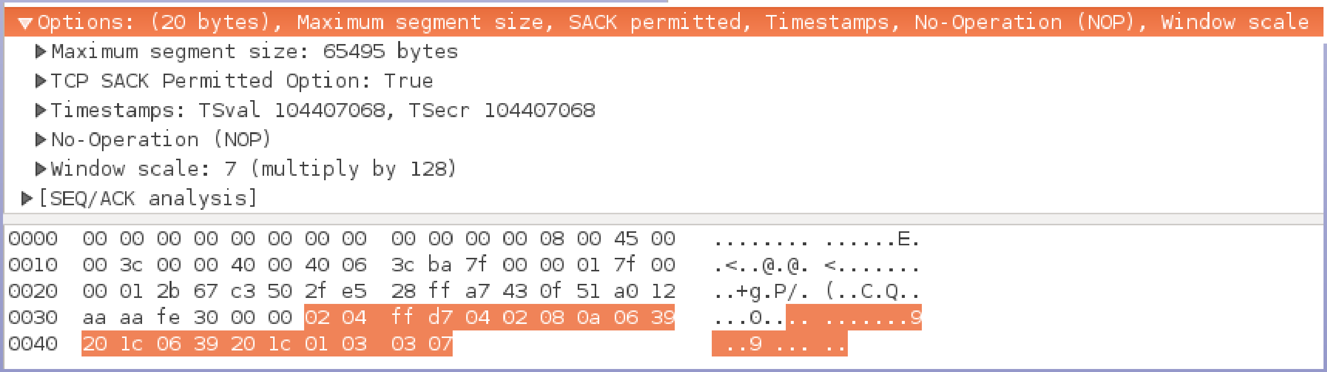
TCP options

- TCP options may follow the basic header
- most TCP options are only sent with the SYN and SYN+ACK packets
- can you guess the format of TCP options?



```
▼Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  ▶Maximum segment size: 65495 bytes
  ▶TCP SACK Permitted Option: True
  ▶Timestamps: TSval 104407068, TSecr 0
  ▶No-Operation (NOP)
  ▶Window scale: 7 (multiply by 128)

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 3c 2f 4c 40 00 40 06  0d 6e 7f 00 00 01 7f 00   .</L@.@. .n......
0020  00 01 c3 50 2b 67 a7 43  0f 50 00 00 00 00 a0 02   ...P+g.C .P......
0030  aa aa fe 30 00 00 02 04  ff d7 04 02 08 0a 06 39   ...0.... .......9
0040  20 1c 00 00 00 00 01 03  03 07                      ....... ..
```



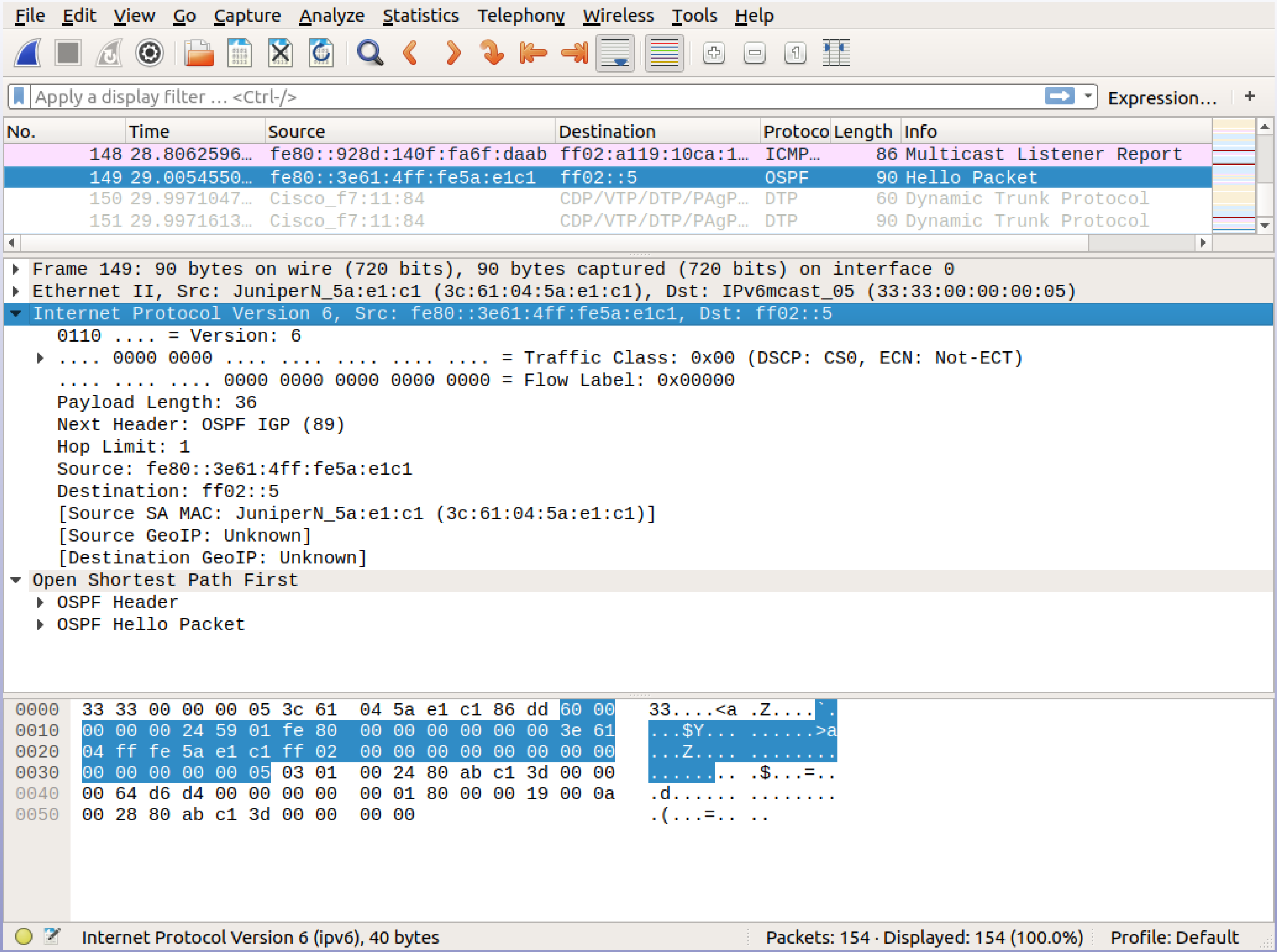
```
▼Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  ▶Maximum segment size: 65495 bytes
  ▶TCP SACK Permitted Option: True
  ▶Timestamps: TSval 104407068, TSecr 104407068
  ▶No-Operation (NOP)
  ▶Window scale: 7 (multiply by 128)
 ▶[SEQ/ACK analysis]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 3c 00 00 40 00 40 06  3c ba 7f 00 00 01 7f 00   .<..@.@. <.......
0020  00 01 2b 67 c3 50 2f e5  28 ff a7 43 0f 51 a0 12   ..+g.P/. (..C.Q..
0030  aa aa fe 30 00 00 02 04  ff d7 04 02 08 0a 06 39   ...0.... .......9
0040  20 1c 06 39 20 1c 01 03  03 07                      ..9 ... ..
```

tcpdump and wireshark

- tcpdump is a utility to look at all the packets on the network and print out the headers
-
```
16:41:58.905998 maru.ics.hawaii.edu.14407
>volcano.telnet: S 2671654129:2671654129(0)win
512  [tos 0x10]16:41:59.115893 volcano.telnet
>maru.ics.hawaii.edu.14407: R 0:0(0)ack
2671654130 win 0
```

- wireshark (formerly known as Ethereal) is similar but (a) window-based, (b) newer


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Expression…
+

```
No.   Time            Source                      Destination            Protoco Length Info
  148 28.8062596…  fe80::928d:140f:fa6f:daab  ff02:a119:10ca:1…  ICMP…       86 Multicast Listener Report
  149 29.0054550…  fe80::3e61:4ff:fe5a:e1c1   ff02::5            OSPF        90 Hello Packet
  150 29.9971047…  Cisco_f7:11:84             CDP/VTP/DTP/PAgP…  DTP         60 Dynamic Trunk Protocol
  151 29.9971613…  Cisco_f7:11:84             CDP/VTP/DTP/PAgP…  DTP         90 Dynamic Trunk Protocol
```

```
▸ Frame 149: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface 0
▸ Ethernet II, Src: JuniperN_5a:e1:c1 (3c:61:04:5a:e1:c1), Dst: IPv6mcast_05 (33:33:00:00:00:05)
▾ Internet Protocol Version 6, Src: fe80::3e61:4ff:fe5a:e1c1, Dst: ff02::5
    0110 .... = Version: 6
  ▸ .... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... .... 0000 0000 0000 0000 0000 = Flow Label: 0x00000
    Payload Length: 36
    Next Header: OSPF IGP (89)
    Hop Limit: 1
    Source: fe80::3e61:4ff:fe5a:e1c1
    Destination: ff02::5
    [Source SA MAC: JuniperN_5a:e1:c1 (3c:61:04:5a:e1:c1)]
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▾ Open Shortest Path First
  ▸ OSPF Header
  ▸ OSPF Hello Packet
```

```
0000  33 33 00 00 00 05 3c 61  04 5a e1 c1 86 dd 60 00   33....<a .Z....`.
0010  00 00 00 24 59 01 fe 80  00 00 00 00 00 00 3e 61   ...$Y... ......>a
0020  04 ff fe 5a e1 c1 ff 02  00 00 00 00 00 00 00 00   ...Z.... ........
0030  00 00 00 00 00 05 03 01  00 24 80 ab c1 3d 00 00   ........ .$...=..
0040  00 64 d6 d4 00 00 00 00  00 01 80 00 00 19 00 0a   .d...... ........
0050  00 28 80 ab c1 3d 00 00  00 00                     .(...=.. ..
```

```
Internet Protocol Version 6 (ipv6), 40 bytes        Packets: 154 · Displayed: 154 (100.0%)    Profile: Default
```

tcpdump example

```
16:47:02.285753 maru.1022 > volcano.ssh:
   S 185741093:185741093(0) win 512
16:47:02.495648 volcano.ssh > maru.1022:
   S 3829593384:3829593384(0)
   ack 185741094 win 16352
16:47:02.495648 maru.1022 > volcano.ssh:
   . ack 1 win 32120 (DF)
16:47:07.183328 volcano.ssh > maru.1022:
   P 1:16(15) ack 1 win 16352 (DF)
16:47:07.183328 maru.1022 > volcano.ssh:
   P 1:16(15) ack 16 win 32120 (DF) [tos 0x10]
16:47:07.433203 volcano.ssh > maru.1022:
   P 16:292(276) ack 16 win 16352 (DF)
16:47:08.502673 volcano.ssh > maru.1022:
   P 16:292(276) ack 16 win 16352 (DF)
16:47:08.522663 maru.1022 > volcano.ssh:
   . ack 292 win 32120 (DF) [tos 0x10]
```

TCP summary so far

- reliable transmission:
  - sequence and acknowledgement numbers
    - counting bytes (not packets)
  - retransmission if no ack is received by timeout
  - checksum for correctness
  - flow control window to regulate sending speed
- ports and demultiplexing
- optimizations: Nagle algorithm and delayed acks
- TCP options
- push and urgent
- next topic: congestion control window