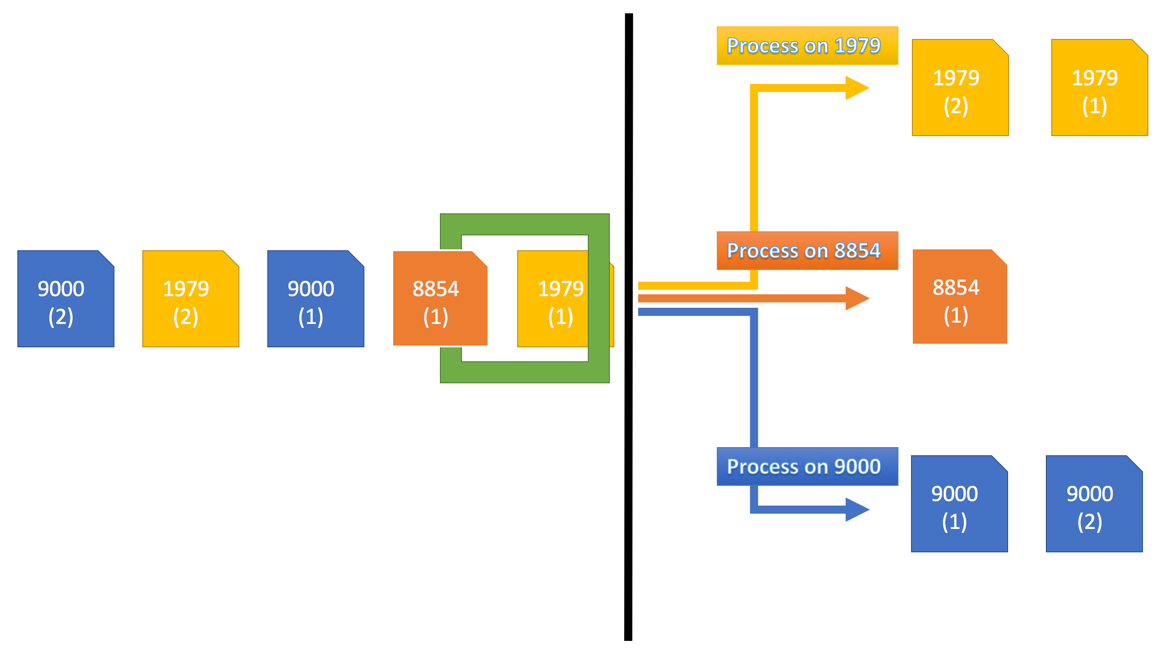
1. Responsibilities of the Transport Layer: Network Layer Problems
   1. Guarantee message delivery
      1. Drops messages
   2. Deliver messages in the same order sent
      1. Reorder Messages
   3. Supports arbitrarily large messages
      1. Limits messages to some finite size



1. Well-Known Port Numbers:
   1. SMTP: 25
   2. HTTP: 80
   3. Kerberos: 88
   4. POP3: 110
   5. BGP: 179
   6. Remote Process Execution: 512
   7. Whoami: 565
   8. LDAP: 389
2. UDP Header:

|  |  |
| --- | --- |
| Source Port | Destination Port |
| 0001010010011000 | 0010001010111000 |
| Length | Checksum |
| 0000000000001100 | 0010011100101110 |
| 01001101 01010100 01010101 00100001 | |

Checksum Math:

0001010010011000 0011011001011100

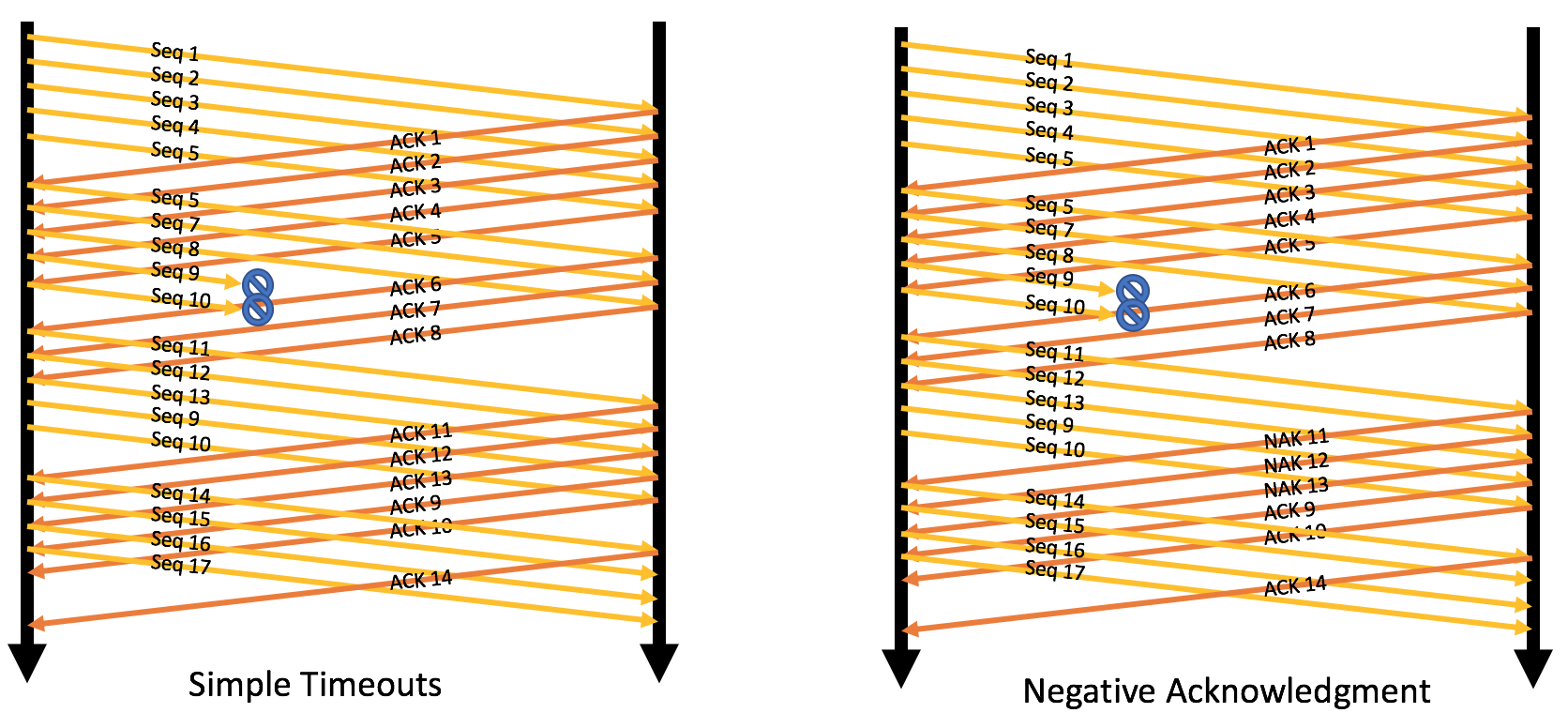
+ 0010001010111000 + 0100110101010100

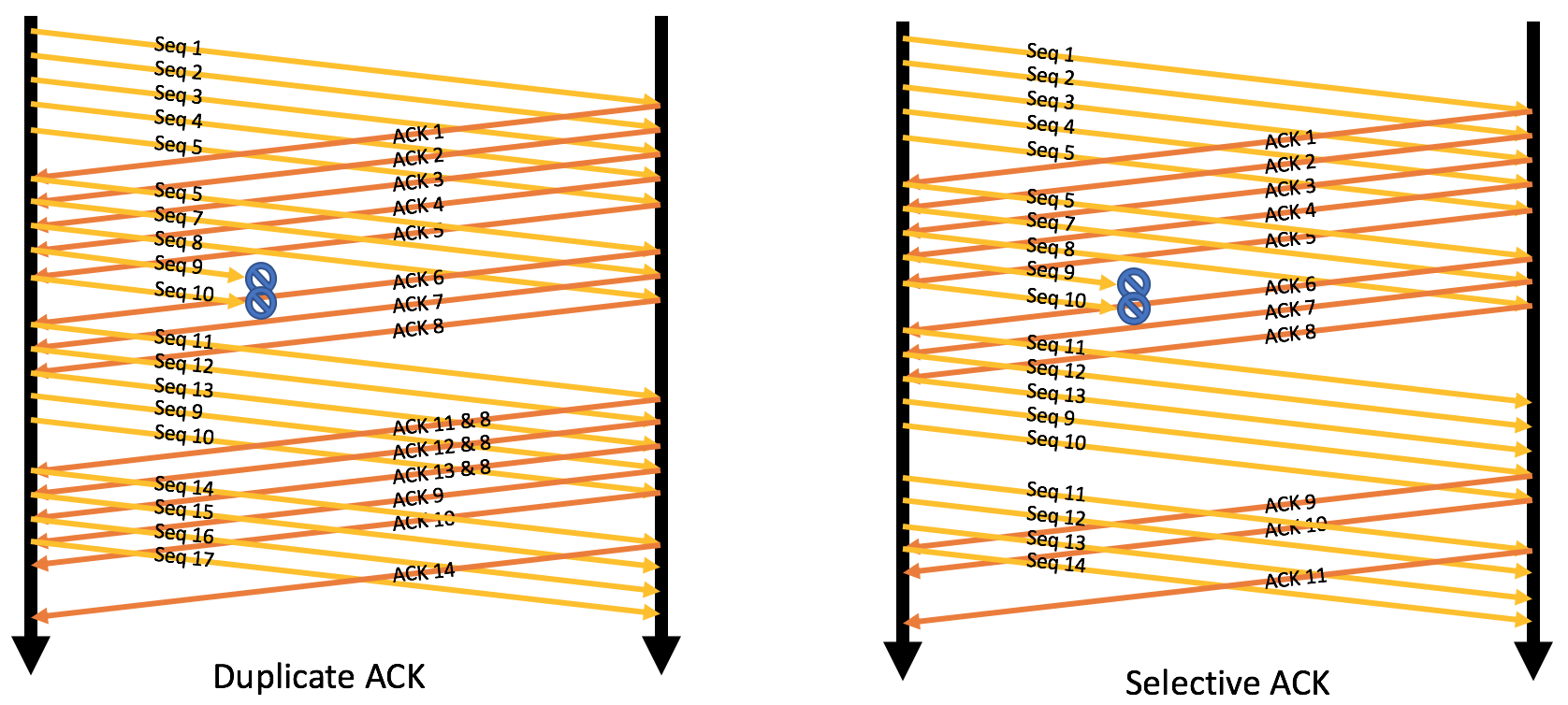
0011011001010000 1000001110110000

+ 0000000000001100 + 0101010100100001

0011011001011100 1101100011010001

1. UDP is used for TFTP (Trivial File Transfer Protocol) as well as SNP (Simple Network Protocol)
2. The two components are acknowledgements and timeouts. If an ACK hasn’t been received within a certain “timeout” frame of time, then the packet is retransmitted automatically.
3. The redundant frame issue is resolved by adding the one bit sequence number. This helps the receiver identify if the frame is new or old by checking the sequence number bit.
4. Packet Exchange Diagrams:





1. 4 bits
2. 2^29 bytes
3. TCP States:
   1. [Close]🡪(appl: active open, sends SYN)🡪[SYN\_SENT]🡪(appl: active close)🡪 [Close]
   2. [Established]🡪(appl: close, send FIN)🡪[FIN\_WAIT\_1]🡪(recv: FIN, send:ACK)🡪[Closing]🡪(recv:ACK)🡪[TimeWait]🡪[Close]
   3. [Close]🡪(appl: active open, sends SYN)🡪[SYN\_SENT]🡪(timeout)🡪 [Close]
4. With the push flag the receiver still needs to wait for all the data first before sending it off to the application. The urgent flag tells the receiver to ignore waiting and just streamline the data directly to the application.
5. Sequence Numbers and ACKS in order:
   1. Seq = 0x00110CFF, ACK = 0x00110D00
   2. Seq = 0x00111009, ACK = 0x0011100A
   3. Seq = 0x001110DE, ACK = 0x001110DF
6. Client Actions:
   1. Sends one 1000 byte segment and waits for next window.
   2. Sends it all.
   3. Sends it all.
7. Original Algorithm: α = 0.875
   1. 105.625 = α x 100.000ms + (1 - α ) × 145ms
   2. 119.297 = α x 105.625ms + (1 - α ) × 145ms
   3. 123.760 = α x 119.297ms + (1 - α ) × 155ms
   4. 126.290 = α x 123.760ms + (1 - α ) × 144ms
   5. 123.004 = α x 126.290ms + (1 - α ) × 100ms
   6. 139.504 = α x 123.004ms + (1 - α ) × 255ms
   7. 147.066 = α x 139.504ms + (1 - α ) × 200ms
   8. 151.808 = α x 147.066ms + (1 - α ) × 185ms
   9. 153.582 = α x 151.808ms + (1 - α ) × 166ms
   10. 152.509 = α x 153.582ms + (1 - α ) × 145ms
8. Jacobson/Karels equations: EstimatedRTT’s
   1. Note: Calculated using Script:
9. foreach ($Sample in $Samples){
10. >> $ERTT = $ERTT + ($Delta \* ($Sample - $ERTT))
11. >> $Rtts += ,$Ertt

20. >> }

b. 139.375

c. 144.296875

d. 153.662109375

e. 145.207763671875

f. 105.650970458984

g. 236.331371307373

h. 204.541421413422

i. 187.442677676678

j. 168.680334709585

h. 147.960041838698