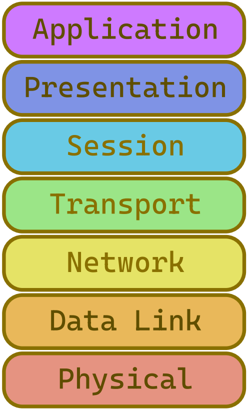
Channel Coding

ARQ and Reliability

November 25, 2021

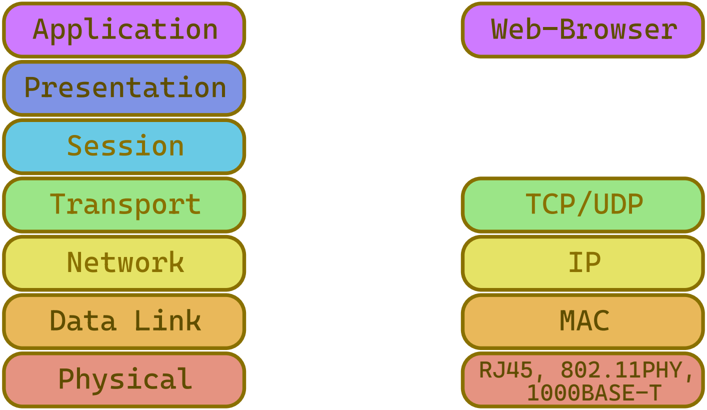
Recall

The Open Systems Interconnection model (OSI)



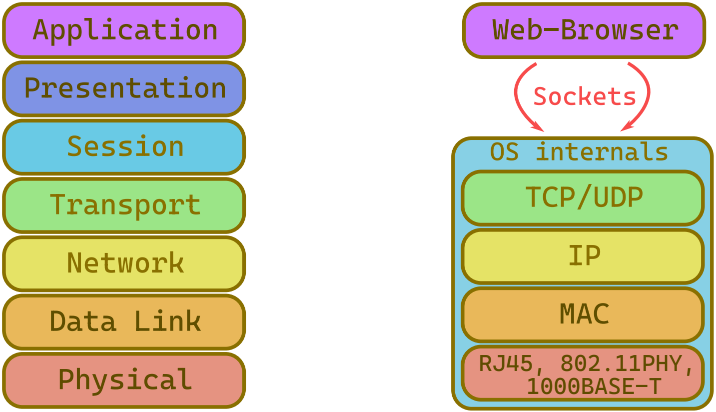
Recall

The Open Systems Interconnection model (OSI)



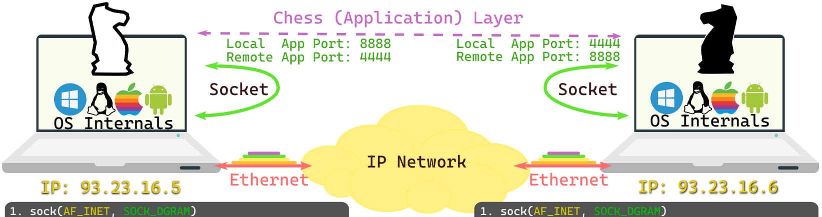
Recall

The Open Systems Interconnection model (OSI)



Sockets

Network stack from users perspective

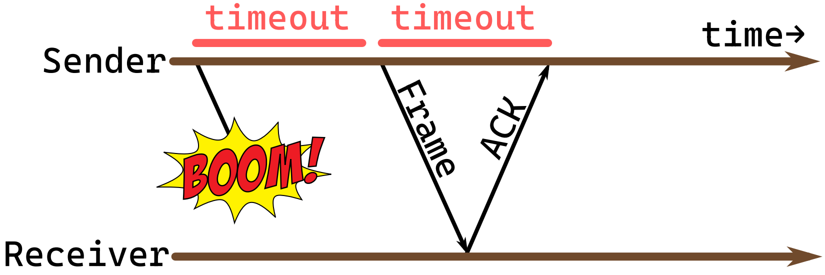


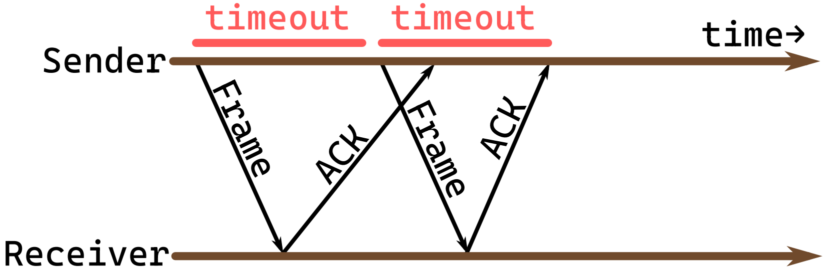

```
1. sock(AF_INET, SOCK_DGRAM)
2.
3. while True: # Gaming Phase
4.     sendto("e2-e4", ('93.23.16.6', 4444))
5.     recvfrom()
```

```
1. sock(AF_INET, SOCK_DGRAM)
2. bind(('0.0.0.0', 4444))
3. while True: # Gaming Phase
4.     recvfrom()
5.     sendto("e7-e5", ('93.23.16.5', 8888))
```

Application Layer
Transport Layer
IP Layer
Data Link Layer
Physical Layer

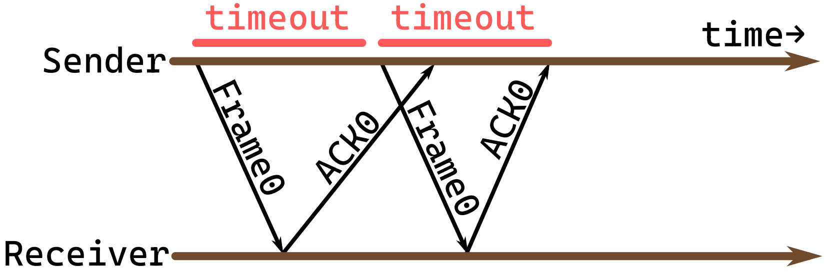

time→
Sender
BOOM!
Receiver

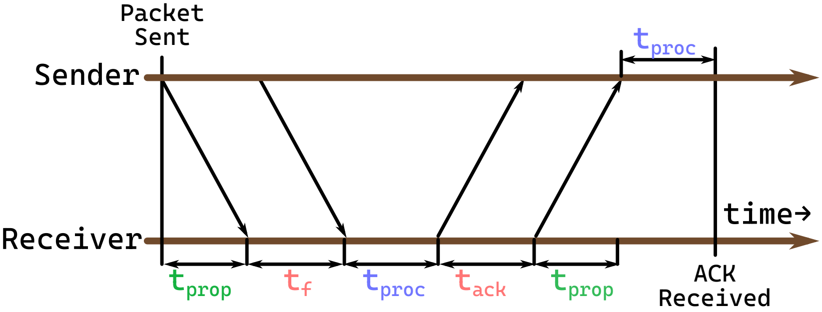

timeout
timeout
time→
Sender
Frame
ACK
BOOM!
Receiver


timeout
timeout
time→
Sender
Receiver
Frame
ACK
Frame
ACK

ARQ

Sequence number. Stop-and-wait.





$$t_0 = 2t_{prop} + t_f + 2t_{proc} + t_{ack} \approx RTT + 2t_{proc} + \frac{n_f + n_{ack}}{Rate}$$

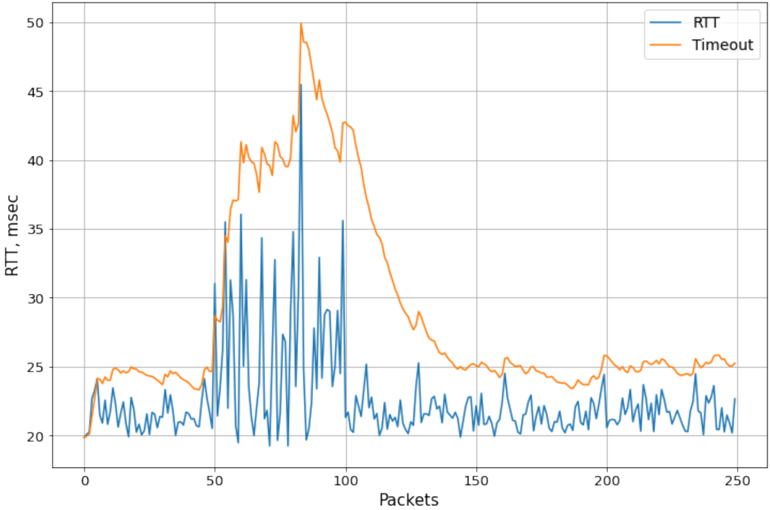
- Which timeout should we choose?
  - Not too big
  - Not too small
- Easy to define for specific LAN. Little variation.
- Difficult over the Internet. High variation.

Simple Timeout calculation scheme[1]. Smoothed RTT + variance.

- $SRTT_{N+1} = 0.9 \cdot SRTT_N + 0.1 \cdot RTT_{N+1}$
- $Svar_{N+1} = 0.9 \cdot Svar_N = 0.1 \cdot |RTT_{N+1} - SRTT_{N+1}|$
- $Timeout_N = SRTT_N + 4 \cdot Svar_N$

[1] rfc2988

ARQ
Adaptive Timeout



- Probability of Failure[1]:

$$P_f = 1 - (1 - plr)^2$$

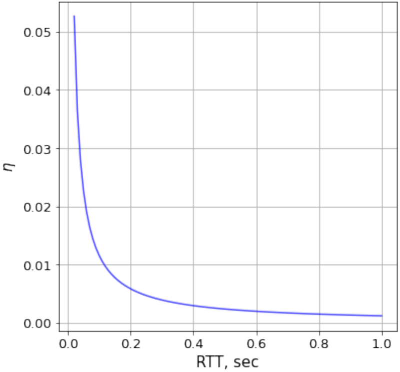
- Average total time to transmit a packet [**1**]:

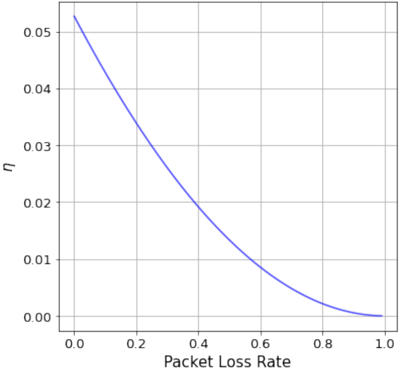
$$E[t_{packet}] = t_0 + \frac{t_{out} P_f}{1 - P_f}$$

- Effective information transmission rate: $R_{eff} = \dfrac{n_f - n_{headers}}{E[t_{packet}]}$
- Associated transmission efficiency: $\eta = \dfrac{R_{eff}}{Rate}$

[1] plr stands for Packet Loss Rate

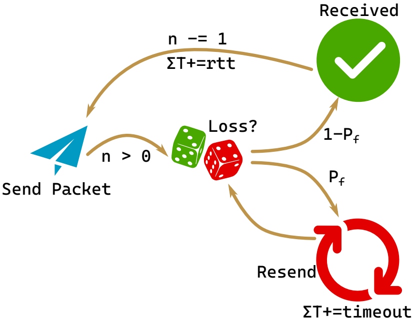
Stop And Wait
Efficiency

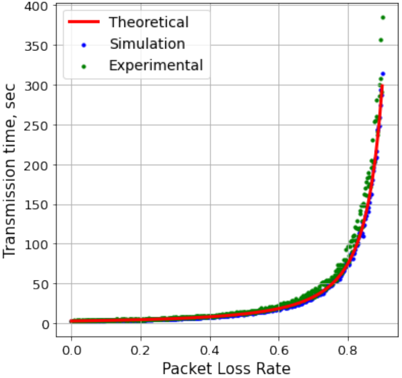




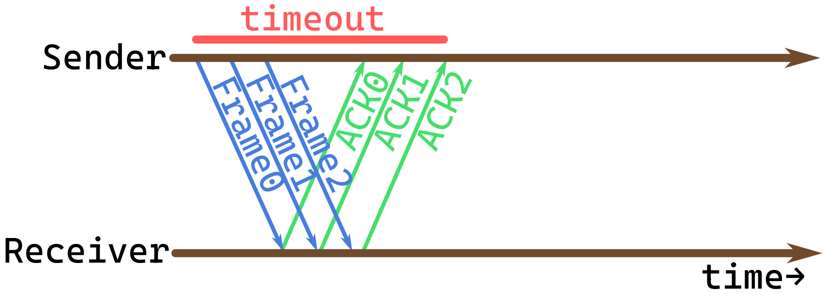
Stop And Wait

Send time simulation

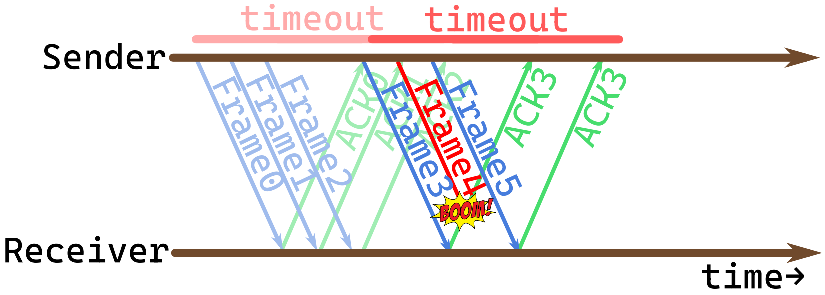




Sliding Window
Go Back N. Principle

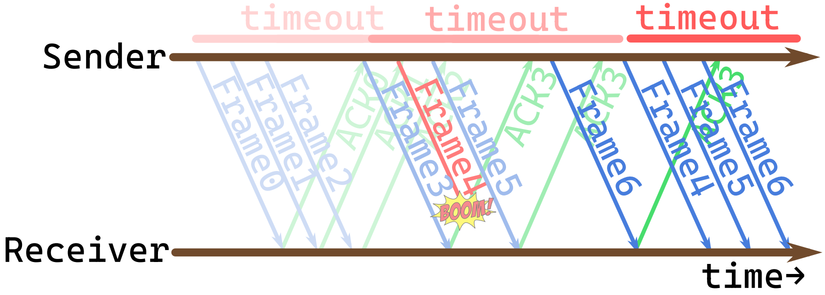


Sliding Window
Go Back N. Principle



Sliding Window

Go Back N. Principle

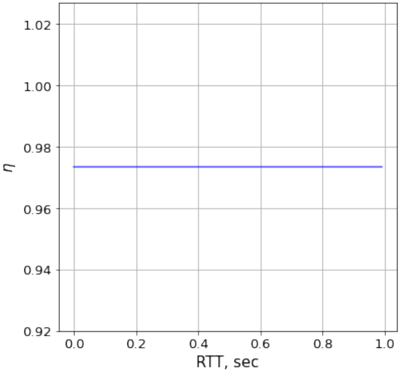


[1] Demo: 1, 2

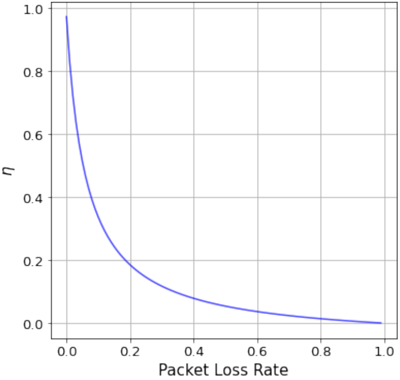
- Probability of Failure: $P_f = plr$
- Average total time to transmit a packet [**1**]. Windows size W_s should be selected so that the channel will be busy all the time.

$$E[t_{packet}] = t_f \frac{1 + (W_s - 1)P_f}{1 - P_f}$$

- Effective information transmission rate: $R_{eff} = \frac{n_f - n_{headers}}{E[t_{packet}]}$
- Associated transmission efficiency: $\eta = \frac{R_{eff}}{Rate}$

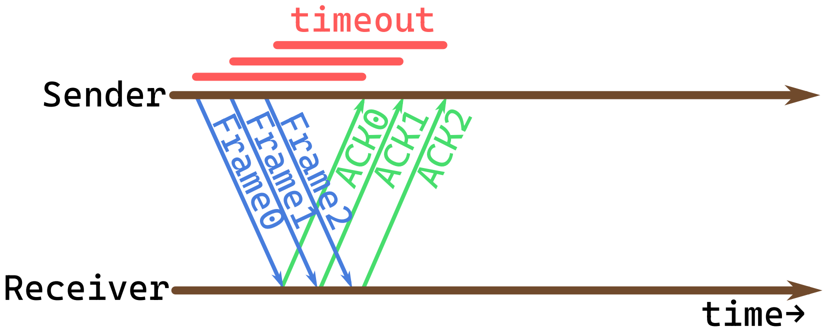


1.02
1.00
0.98
0.96
0.94
0.92
η
0.0
0.2
0.4
0.6
0.8
1.0
RTT, sec



1.0
0.8
0.6
0.4
0.2
0.0
η
0.0
0.2
0.4
0.6
0.8
1.0
Packet Loss Rate

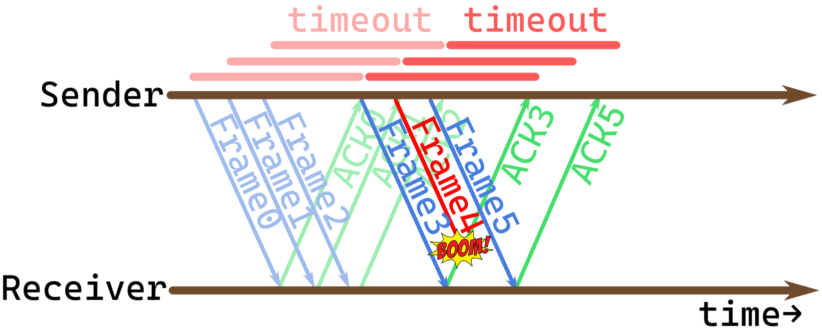
Sliding Window
Selective Repeat. Principle



[1] Demo: 1, 2

Sliding Window

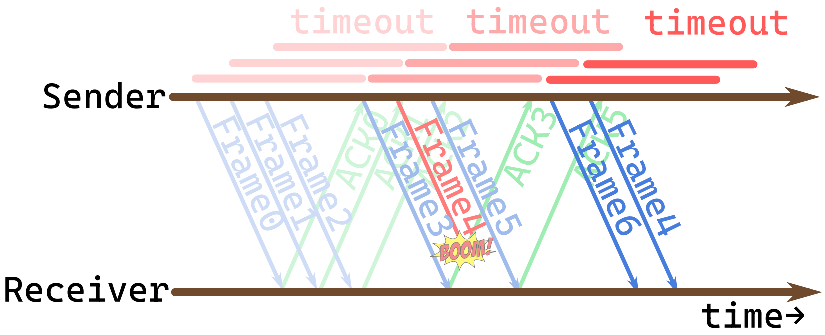
Selective Repeat. Principle



[1] Demo: 1, 2

Sliding Window

Selective Repeat. Principle

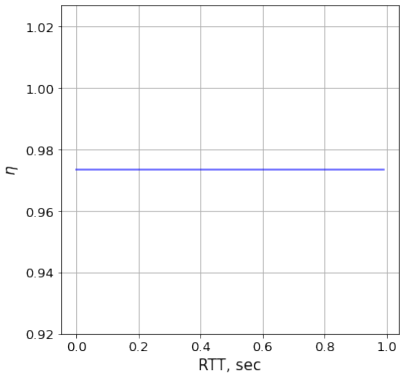


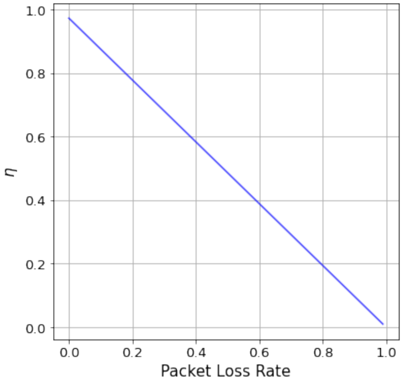
[1]Demo: 1, 2

- Probability of Failure: $P_f = plr$
- Average total time to transmit a packet [**1**]. Windows size W_s should be selected so that the channel will be busy all the time.

$$E[t_{packet}] = \frac{t_f}{1 - P_f}$$

- Effective information transmission rate: $R_{eff} = \frac{n_f - n_{headers}}{E[t_{packet}]}$
- Associated transmission efficiency: $\eta = \frac{R_{eff}}{Rate}$


1.02
1.00
0.98
0.96
0.94
0.92
η
0.0
0.2
0.4
0.6
0.8
1.0
RTT, sec


1.0
0.8
0.6
0.4
0.2
0.0
η
0.0
0.2
0.4
0.6
0.8
1.0
Packet Loss Rate

Task 1

Echo Server

```
user@pc1:~$ ./echo-server -p 8888 --proto udp
Server is listenning on UDP port 8888....
Connection from 192.168.0.6...
Client message: Your mouth is deep as a cave!
Client message: I took it okay the first time mr dentist....
Client message: That was an echo))))
```

```
user@pc2:~$ ./echo-client -a 192.168.0.7 -p 8888 --proto udp
>>>>: Your mouth is deep as a cave!
echo: Your mouth is deep as a cave!
>>>>: I took it okay the first time mr dentist....
echo: I took it okay the first time mr dentist....
>>>>: That was an echo))))
echo: That was an echo))))
```

[1] Demo: 1, 2

Your task is to create Python program which transmits files via network under harsh network conditions.

Filesize: 1 GB Bandwidth: 100 Mbps MTU: 1512 B

RTT, msec	PLR	4 credits	8 credits	11 credits	15 credits
1	0 %	25 min	20 min	15 min	10 min
10	1 %	30 min	25 min	20 min	15 min
10	10 %		30 min	25 min	20 min
100	10 %			30 min	25 min
1000	1 %				30 min

Table 1: Cases and credits

Stop-and-Wait gets 4 credits. Go-Back-N gets 8 credits. Selective repeate gets 11 credits.

Task 2

Network condition simulation

```
# to set delays and losses on eth0 interface
tc qdisc add dev eth0 root netem delay 10ms loss 1.0%
# to remove delays and losses on eth0 interface
tc qdisc del dev eth0 root netem delay 10ms loss 1.0%
# to limit bandwith on eth0 interface
tc qdisc add dev eth0 root tbf rate 100mbit
# to check network parammeters
iperf3 -s -p 8888 # server side
iperf3 -c 127.0.0.1 -p 8888 -u -b 1000m # client side
```

[1] How to limit bandwidth on Linux to better test your applications

Task 2

How it should look like

```
user@pc1:~$ ./server -p 8888 > file
Server is listening on UDP port 8888....
Connection from 192.168.0.6...
Transmission is over.
Transmission time is 123.01 sec.
user@pc1:~$ ls -lah file
-rwxrwxrwx 1 user user 1G Oct 28 18:47 file
user@pc1:~$ md5sum ./file
a7931b2aa3348a0b68286c8ea4ba6a11 file
```

```
user@pc2:~$ dd if=/dev/urandom of=file bs=1G count=1
user@pc2:~$ ls -lah file
-rwxrwxrwx 1 user user 1G Oct 28 18:21 file
user@pc2:~$ md5sum ./file
a7931b2aa3348a0b68286c8ea4ba6a11 file
user@pc2:~$ cat file | ./sender -a 192.168.0.5 -p 8888
Transmission started.
Transmission is over.
Transmission time is 122 sec
```

References and further readings

[1] Leon-Garcia, A., & Widjaja, I. (2000). Communication networks: fundamental concepts and key architectures (Vol. 2). New York: McGraw-Hill.

[2] Computer Networking: A Top-Down Approach. / Interactive Animations

[3] TU Berlin. Computer Networks - An Animated Approach

[4] University Washington. Computer Networks - Retransmissions

[5] University of Colorado. Peer-to-Peer Protocols and Local Area Networks

Thanks for your attention