

Nome: Nathan Medeiros Cristiano.

Turma: RED129005

LABORATÓRIO 9

USANDO MINHA MÁQUINA / IFSC

DESVENDANDO O TCP - NÚMERO DE SEQUÊNCIA, CONTROLE DE ERROS,
TRANSMISSÃO *FULL-DUPLEX*

PARTE 1 - Transmissão sem erros: Verificação de Número de Sequência e Reconhecimentos

ip.addr==10.0.0.20						
No.	Time	Source	Destination	Protocol	Length Info	
7	19.841362	10.0.0.20	10.0.0.21	TCP	74	38370 → 5555 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1203243719 TSecr=0 WS=128
8	19.841381	10.0.0.21	10.0.0.20	TCP	74	5555 → 38370 [SYN, ACK] Seq=0 Ack=1 Win=10 Len=0 MSS=1460 SACK_PERM TSval=2533205043 TSecr=1203243719 WS=4
9	19.841395	10.0.0.20	10.0.0.21	TCP	66	38370 → 5555 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1203243719 TSecr=2533205043
10	20.048281	10.0.0.20	10.0.0.21	TCP	76	[TCP Window Full] 38370 → 5555 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=10 TSval=1203243926 TSecr=2533205043
11	20.048319	10.0.0.21	10.0.0.20	TCP	66	[TCP ZeroWindow] 5555 → 38370 [ACK] Seq=1 Ack=11 Win=0 Len=0 TSval=2533205250 TSecr=1203243926
12	20.048426	10.0.0.21	10.0.0.20	TCP	66	[TCP Window Update] 5555 → 38370 [ACK] Seq=1 Ack=11 Win=12 Len=0 TSval=2533205250 TSecr=1203243926
13	20.256275	10.0.0.20	10.0.0.21	TCP	78	[TCP Window Full] 38370 → 5555 [PSH, ACK] Seq=11 Ack=1 Win=64256 Len=12 TSval=1203244134 TSecr=2533205250
14	20.256305	10.0.0.21	10.0.0.20	TCP	66	[TCP ZeroWindow] 5555 → 38370 [ACK] Seq=1 Ack=23 Win=0 Len=0 TSval=2533205458 TSecr=1203244134
15	20.256368	10.0.0.21	10.0.0.20	TCP	66	[TCP Window Update] 5555 → 38370 [ACK] Seq=1 Ack=23 Win=12 Len=0 TSval=2533205458 TSecr=1203244134
16	20.256391	10.0.0.20	10.0.0.21	TCP	74	38370 → 5555 [PSH, ACK] Seq=23 Ack=1 Win=64256 Len=8 TSval=1203244134 TSecr=2533205458
17	20.256405	10.0.0.21	10.0.0.20	TCP	66	5555 → 38370 [ACK] Seq=1 Ack=31 Win=12 Len=0 TSval=2533205458 TSecr=1203244134
20	32.869827	10.0.0.21	10.0.0.20	TCP	66	5555 → 38370 [FIN, ACK] Seq=1 Ack=31 Win=12 Len=0 TSval=2533218071 TSecr=1203244134
21	32.869916	10.0.0.20	10.0.0.21	TCP	66	38370 → 5555 [FIN, ACK] Seq=31 Ack=2 Win=64256 Len=0 TSval=1203256747 TSecr=2533218071
22	32.869931	10.0.0.21	10.0.0.20	TCP	66	5555 → 38370 [ACK] Seq=2 Ack=32 Win=12 Len=0 TSval=2533218072 TSecr=1203256747

- Qual o número de sequência de cada segmento de dados transmitido (do Transmissor para o Receptor) e qual o significado do número de reconhecimento em cada um deles?

Os números de sequência do transmissor são:

- 0; 1; 11; 23; 31.

O significado em cada um deles é dado pelo segmento de cada byte da cadeia que foi segmentada.

- Como foi reconhecido cada segmento enviado? É igual ao número de sequência ou é um número acima? Justifique.

Pelo ACK do receptor, em resposta ao segmento enviado pelo transmissor, é verificado pela ordem sequencial de bytes. Ele não é igual ao número de sequência, é acima, pois **é o número de sequência + length**.

16	20.256391	10.0.0.20	10.0.0.21	TCP	74	38370 → 5555 [PSH, ACK] Seq=23 Ack=1 Win=64256 Len=8 TSval=1203244134 TSecr=2533205458
17	20.256405	10.0.0.21	10.0.0.20	TCP	66	5555 → 38370 [ACK] Seq=1 Ack=31 Win=12 Len=0 TSval=2533205458 TSecr=1203244134

- Qual o significado, funcionalidade e necessidade das mensagens, inseridas pelo Wireshark, "TCP ZeroWindow" e "TCP Window Update"?

Isso, serve para visualização do usuário, para identificar, que a janela(buffer), limitado pelo receptor, foi "estourado" e depois foi reajustado.

- Qual a relação entre os campos "Len=", "Seq=", "Ack=", "Win=" e o tamanho do segmento de dados?

A relação entre eles é mútua, o "LEN" é o tamanho do pacote em bytes, logo o "SEQ" marca onde começa o segmento de dados, o ACK é a operação de adição entre em esses dois confirmado o segmento recebido, o "WIN" define a quantidade de bytes que o Buffer vai ter, sendo o assim cada pacote enviado se adequa ao "WIN" do receptor.

Relative sequence numbers - OFF

No.	Time	Source	Destination	Protocol	Length	Info
7	19.841362	10.0.0.20	10.0.0.21	TCP	74	38370 → 5555 [SYN] Seq=2276423584 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1283243719 TSecr=0 WS=128
8	19.841381	10.0.0.21	10.0.0.20	TCP	74	5555 → 38370 [SYN, ACK] Seq=3845814424 Ack=2276423585 Win=10 Len=0 MSS=1460 SACK_PERM TSval=2533285043 TSecr=1283243719 WS=4
9	19.841395	10.0.0.20	10.0.0.21	TCP	66	38370 → 5555 [ACK] Seq=2276423585 Ack=3845814425 Win=64256 Len=0 TSval=1283243719 TSecr=2533285043
10	20.048281	10.0.0.20	10.0.0.21	TCP	76	[TCP Window Full] 38370 → 5555 [PSH, ACK] Seq=2276423585 Ack=3845814425 Win=64256 Len=10 TSval=1283243926 TSecr=2533285043
11	20.048319	10.0.0.21	10.0.0.20	TCP	66	[TCP ZeroWindow] 5555 → 38370 [ACK] Seq=3845814425 Ack=2276423595 Win=0 Len=0 TSval=2533285250 TSecr=1283243926
12	20.048426	10.0.0.21	10.0.0.20	TCP	66	[TCP Window Update] 5555 → 38370 [ACK] Seq=3845814425 Ack=2276423595 Win=12 Len=0 TSval=2533285250 TSecr=1283243926
13	20.256275	10.0.0.20	10.0.0.21	TCP	78	[TCP Window Full] 38370 → 5555 [PSH, ACK] Seq=2276423595 Ack=3845814425 Win=64256 Len=12 TSval=1283244134 TSecr=2533285250
14	20.256385	10.0.0.21	10.0.0.20	TCP	66	[TCP ZeroWindow] 5555 → 38370 [ACK] Seq=3845814425 Ack=2276423607 Win=0 Len=0 TSval=2533285458 TSecr=1283244134
15	20.256368	10.0.0.21	10.0.0.20	TCP	66	[TCP Window Update] 5555 → 38370 [ACK] Seq=3845814425 Ack=2276423607 Win=12 Len=0 TSval=2533285458 TSecr=1283244134
16	20.256391	10.0.0.20	10.0.0.21	TCP	74	38370 → 5555 [PSH, ACK] Seq=2276423607 Ack=3845814425 Win=64256 Len=8 TSval=1283244134 TSecr=2533285458
17	20.256485	10.0.0.21	10.0.0.20	TCP	66	5555 → 38370 [ACK] Seq=3845814425 Ack=2276423615 Win=12 Len=0 TSval=2533285458 TSecr=1283244134
20	32.869827	10.0.0.21	10.0.0.20	TCP	66	5555 → 38370 [FIN, ACK] Seq=3845814425 Ack=2276423615 Win=12 Len=0 TSval=2533218071 TSecr=1283244134
21	32.869916	10.0.0.20	10.0.0.21	TCP	66	38370 → 5555 [FIN, ACK] Seq=2276423615 Ack=3845814426 Win=64256 Len=0 TSval=1283256747 TSecr=2533218071
22	32.869931	10.0.0.21	10.0.0.20	TCP	66	5555 → 38370 [ACK] Seq=3845814426 Ack=2276423616 Win=12 Len=0 TSval=2533218072 TSecr=1283256747

- A principal diferença aqui é que o número de sequência agora é o "Real", definido pelo TCP aleatoriamente pelas suas 4 bilhões de possibilidades (2^{32}). Isso serve a fim de evitar número de sequências iguais entre a transmissão.

9.3 PARTE 2 - Transmissão com erros: retransmissões

No.	Time	Source	Destination	Protocol	Length	Info
17	21.312371	10.0.0.21	10.0.0.20	TCP	66	5555 → 51954 [ACK] Seq=2272387323 Ack=179813960
18	21.312376	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [ACK] Seq=179813960 Ack=2272387323
19	21.312379	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [PSH, ACK] Seq=179814684 Ack=22723
20	21.312384	10.0.0.21	10.0.0.20	TCP	78	[TCP Dup ACK 17#1] 5555 → 51954 [ACK] Seq=22723
21	21.323943	10.0.0.20	10.0.0.21	TCP	790	[TCP Out-Of-Order] 51954 → 5555 [ACK] Seq=17981
22	21.324030	10.0.0.21	10.0.0.20	TCP	66	5555 → 51954 [ACK] Seq=2272387323 Ack=179815408
23	21.324038	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [ACK] Seq=179815408 Ack=2272387323
24	21.324041	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [PSH, ACK] Seq=179816132 Ack=22723
25	21.324058	10.0.0.21	10.0.0.20	TCP	66	5555 → 51954 [ACK] Seq=2272387323 Ack=179816856
26	21.324062	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [ACK] Seq=179816856 Ack=2272387323
27	21.324064	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [PSH, ACK] Seq=179817580 Ack=22723
28	21.324097	10.0.0.21	10.0.0.20	TCP	66	5555 → 51954 [ACK] Seq=2272387323 Ack=179818304
29	21.324101	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [ACK] Seq=179818304 Ack=2272387323
30	21.324103	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [PSH, ACK] Seq=179819028 Ack=22723
31	21.324114	10.0.0.21	10.0.0.20	TCP	66	5555 → 51954 [ACK] Seq=2272387323 Ack=179819752
32	21.324117	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [ACK] Seq=179819752 Ack=2272387323
33	21.324119	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [PSH, ACK] Seq=179820476 Ack=22723
34	21.367956	10.0.0.21	10.0.0.20	TCP	66	5555 → 51954 [ACK] Seq=2272387323 Ack=179820476
35	21.367965	10.0.0.20	10.0.0.21	TCP	790	51954 → 5555 [ACK] Seq=179821200 Ack=2272387323
36	22.623984	10.0.0.20	10.0.0.21	TCP	790	[TCP Retransmission] 51954 → 5555 [ACK] Seq=179
37	22.624056	10.0.0.21	10.0.0.20	TCP	86	[TCP Dup ACK 34#1] 5555 → 51954 [ACK] Seq=22723
38	22.624078	10.0.0.20	10.0.0.21	TCP	790	[TCP Retransmission] 51954 → 5555 [PSH, ACK] Se

- Houve perda de pacotes? Como você identificou isso?

Sim, o pode-se observar nos pacotes "Out-Of-Order", e o pacote de "Dup Ack"

- Os pacotes perdidos foram retransmitidos? Justifique.

Sim! Pode se observar nos pacotes 36 e 38 "TCP Retransmission" Cada pacote tem um temporizador, após esse temporizador se esgotar, o TCP retransmite o pacote perdido, e também se o TCP perceber que os números de ACKS forem maior que 3, ele envia um pacote de retransmissão referente aquele pacote perdido.

- Qual o significado da mensagem, inserida pelo Wireshark, "TCP Retransmission"? Como você justificaria uma perda de segmento sem acesso a essa informação?

Significa que o TCP retransmite o referido pacote perdido. Sem acesso a essa informação, basta vermos os ACKS dos pacotes, caso tivesse o pacote original mais dois duplicados, saberíamos que este pacote foi perdido.

- Qual o significado das cores diferenciadas, inseridas pelo Wireshark, nos diversos segmentos apresentados?

Pacotes pretos, informam problemas na rede.

9.4 PARTE 3 - Testando a capacidade do TCP de enviar dados de forma duplex

- Os arquivos foram corretamente trocados entre as duas máquinas? Dica: Responda observando o conteúdo dos arquivos, que são exclusivos e bem criativos :).

Sim foram enviados corretamente!

[illegible]

Perguntas:

- Onde pode ser observado a comunicação *full-duplex*? Obs: Foque a análise nos segmentos que contém [PSH, ACK].

25	85.329223	10.0.0.21	10.0.0.20	TCP	1514 44076 → 5555 [PSH, ACK] Seq=3682048335 Ack=2155850857
26	85.329224	10.0.0.21	10.0.0.20	TCP	1514 44076 → 5555 [ACK] Seq=3682049783 Ack=2155850857 Win=
27	85.329234	10.0.0.20	10.0.0.21	TCP	1514 5555 → 44076 [PSH, ACK] Seq=2155850857 Ack=3682049783

Aqui observamos que tanto o receptor quanto o transmissor estão trocando pacotes simultaneamente, observando o "PSH,ACK", .

- Qual é a relação entre os comandos no terminal tanto do cliente como do servidor com a comunicação *full-duplex*?

O comando do transmissor deixa ele em modo escuta na porta 5555, esperando uma conexão para enviar o arquivo "Servidor.txt", e qualquer outra coisa que chegar do outro lado, guardar em "Arq_recebido.txt". Já no receptor falamos que quando a conexão for estabelecida ele enviará o arquivo "Cliente.txt" para o servidor "10.0.0.20 5555", e qualquer coisa que o servidor enviar, guardar no em "Arq_recebido.txt"

- Como os ACKs são propagados, em pacotes exclusivos ou de carona (*piggyback*) com os dados?

Eles estão sendo enviados em ambos, podemos ver nos seguintes pacotes.

- **PIGGYBACK**

14	85.329...	10.0.0.21	10.0.0.20	TCP	1... 44076 → 5555 [PSH, ACK] Seq=3682045439 Ack=2155843617 Win=4380 Len=1448
----	-----------	-----------	-----------	-----	--

- **ACKS EXCLUSIVOS**

seguintes pacotes.

21	85.329183	10.0.0.20	10.0.0.21	TCP	66 5555 → 44076 [ACK] Seq=2155847961 Ack=3682045439 Win=2896 Len=0
----	-----------	-----------	-----------	-----	--

18	85.329158	10.0.0.21	10.0.0.20	TCP	66 44076 → 5555 [ACK] Seq=3682048335 Ack=2155845065 Win=2932 Len=0
----	-----------	-----------	-----------	-----	--

