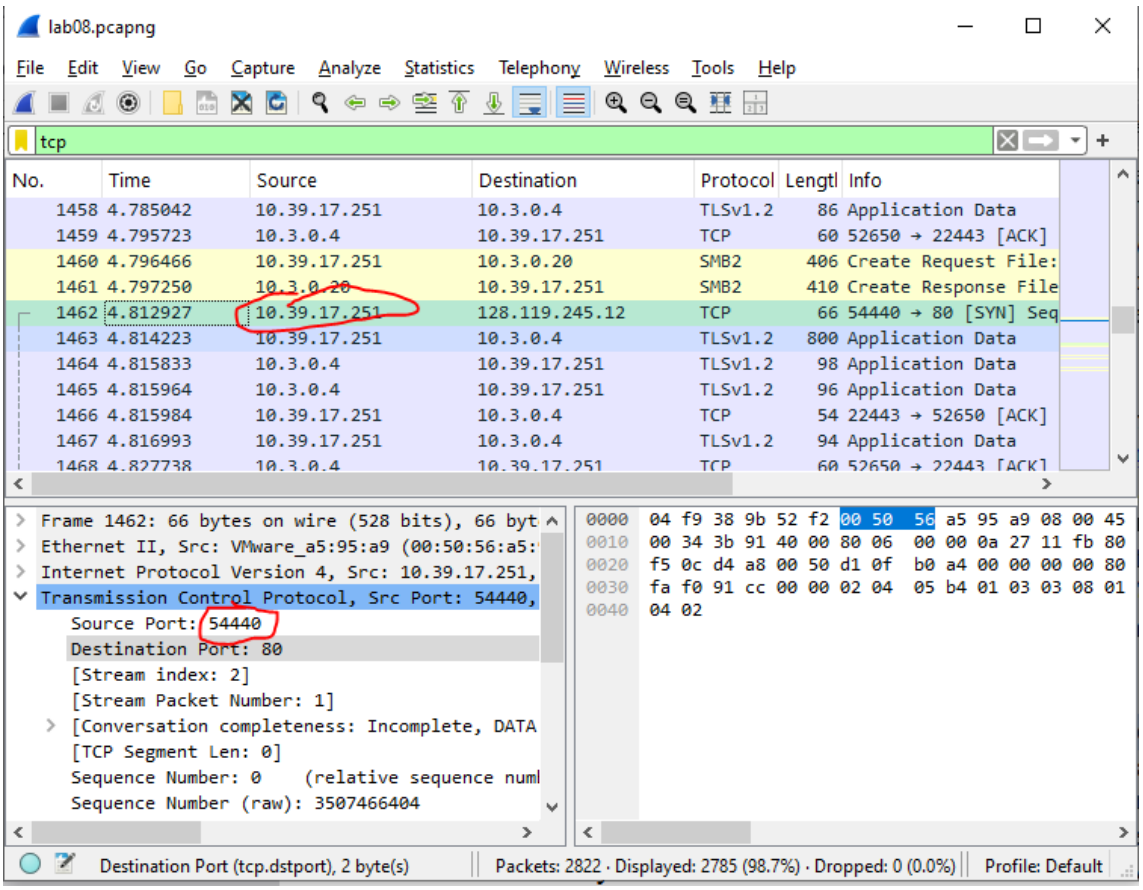
TCP

1

IP port: 10.39.17.251

Source port: 54440



2

IP dest: 128.119.245.12

Source port: 80

3

Mesmo que na 1.

4

Syn bit = 1

Seq nº = 0

5

Ack = 1 (houve um incremento no seq nº)

Seq nº = 0

6

Seq nº = 1

7

#	Seq Number	Tempo de envio (s)	Próx. ACK esperado	Tempo do ACK (estimado)	RTT (s)
1	152578	3.416831	154038	3.525749	0.1089
2	154038	3.417566	155498	3.530351	0.1128
3	155498	3.417566	156958	3.531622	0.1141
4	156958	3.417566	158418	3.531622	0.1141
5	158418	3.417566	159878	3.531622	0.1141
6	159878	3.417566	161338	3.531622	0.1141

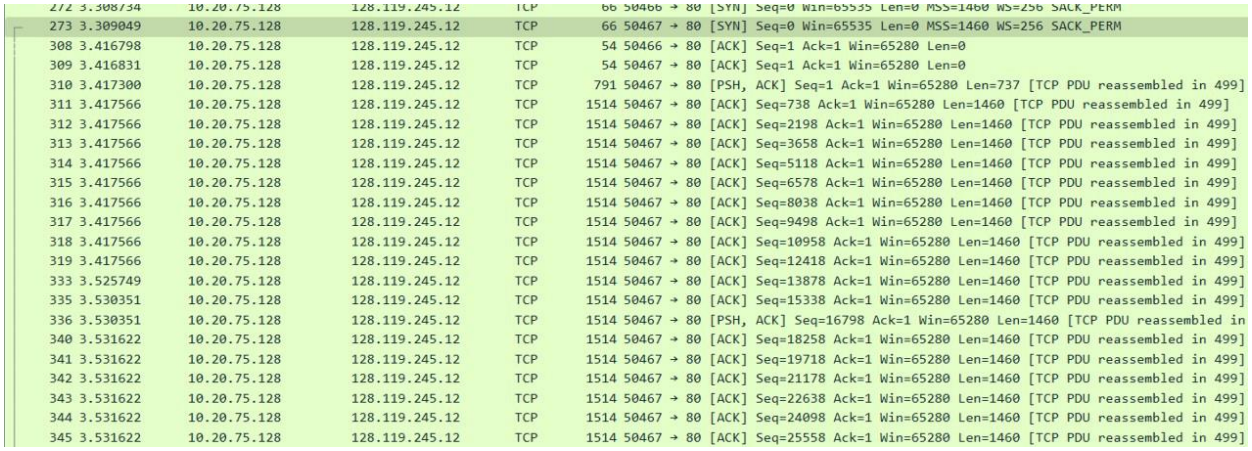
```
EstimatedRTT = (1 - α) * EstimatedRTT_anterior + α * SampleRTT
α = 0.125
```

8

O primeiro segmento TCP (com o HTTP POST) tem 737 bytes de payload.

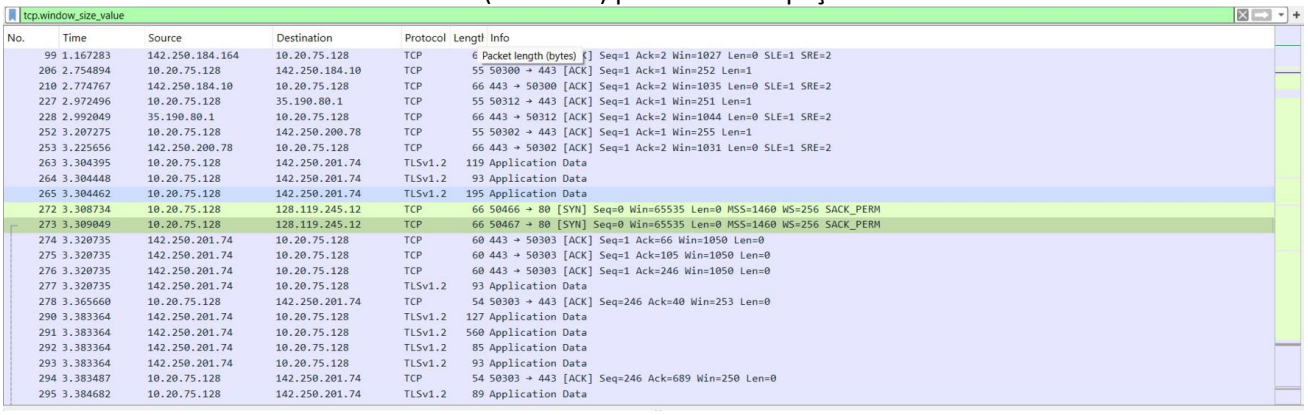
Os cinco seguintes tem o valor padrão de 1460 bytes, que e o Maximum Segment Size (MSS) para redes Ethernet tipicas.

Segmento	Número de Sequência	Comprimento TCP
1 (POST)	152578	737 bytes
2	153315	1460 bytes
3	154775	1460 bytes
4	156235	1460 bytes
5	157695	1460 bytes
6	159155	1460 bytes



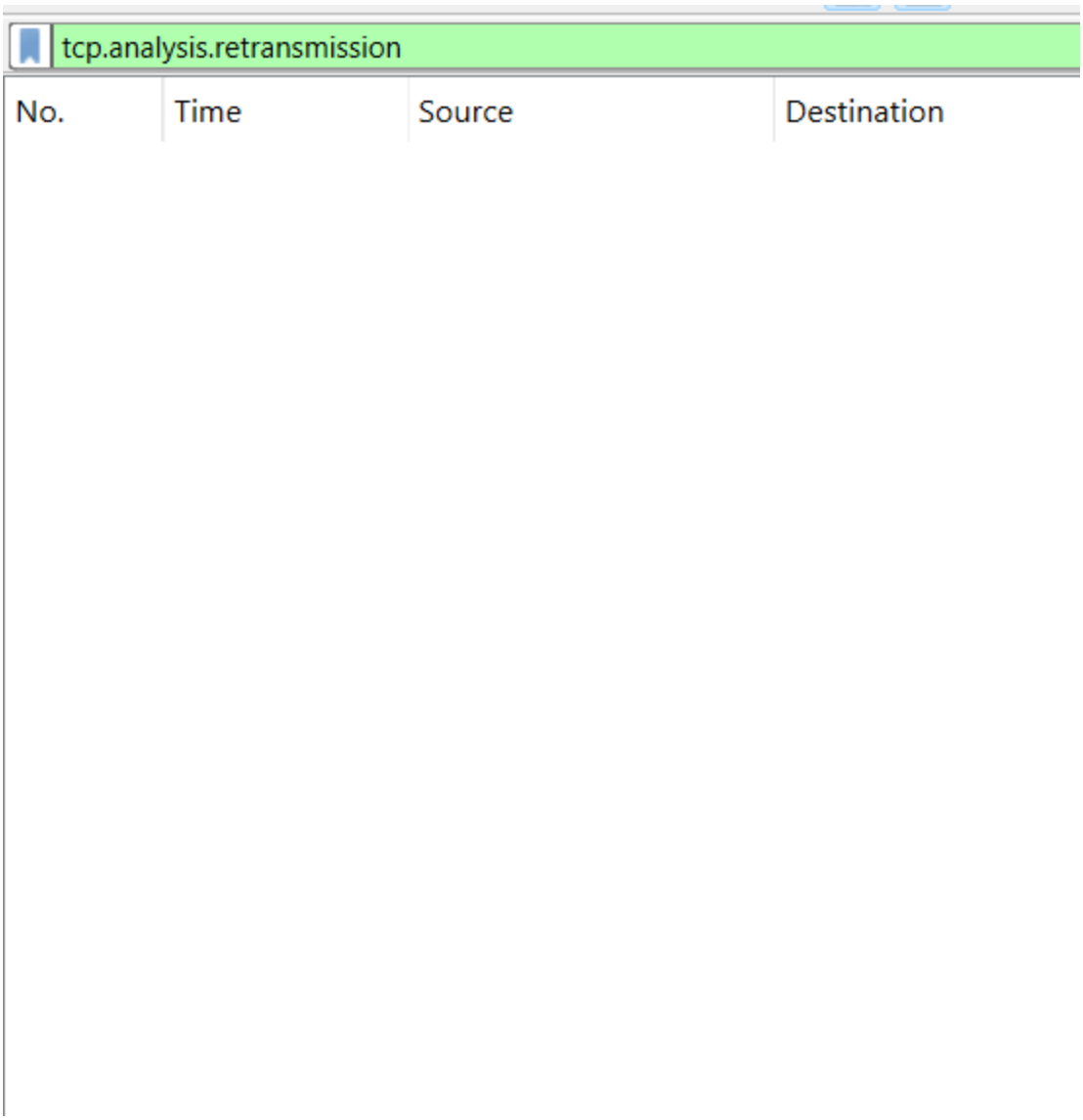
9

O espaço mínimo de buffer anunciado foi: 65280 bytes.
A janela nunca foi reduzida para zero, nem para um valor muito pequeno.
Conclusão: O emissor nunca foi travado (throttled) por falta de espaço no buffer do recetor.



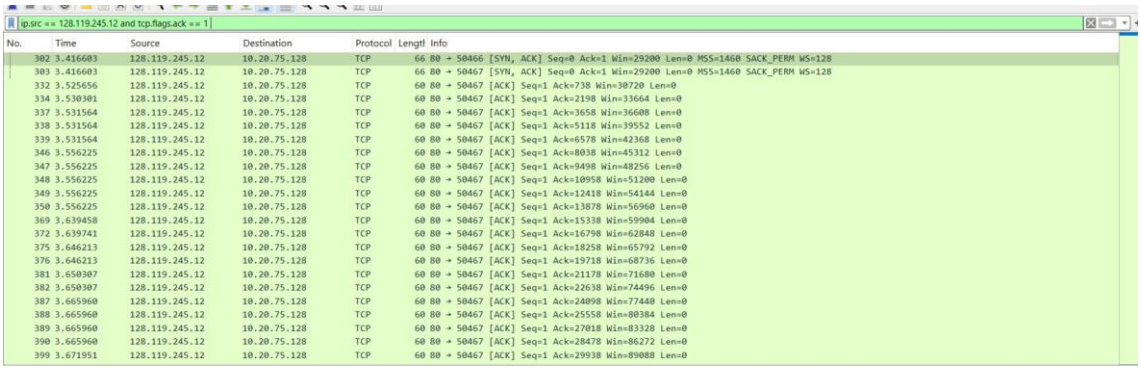
Com essa imagem confirmamos mesmo que nenhum dos pacotes mostra Window size value=0

10



Verificação feita com o filtro: tcp.analysis.retransmission
Resultado: Não foram encontradas retransmissões nos pacotes capturados.
Conclusão: Todos os segmentos TCP foram entregues com sucesso, sem necessidade de reenviar

11



ip.src == 128.119.245.12 and tcp.flags.ack == 1

No.	Time	Source	Destination	Protocol	Length	Info
302	3.416603	128.119.245.12	10.20.75.128	TCP	66	80 → 50466 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM WS=128
303	3.416603	128.119.245.12	10.20.75.128	TCP	66	80 → 50467 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM WS=128
332	3.525656	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=738 Win=30720 Len=0
334	3.530301	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=2198 Win=33664 Len=0
337	3.531564	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=3658 Win=36608 Len=0
338	3.531564	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=5118 Win=39552 Len=0
339	3.531564	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=6578 Win=42368 Len=0
346	3.556225	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=8038 Win=45312 Len=0
347	3.556225	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=9498 Win=48256 Len=0
348	3.556225	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=10958 Win=51200 Len=0
349	3.556225	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=12418 Win=54144 Len=0
350	3.556225	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=13878 Win=56960 Len=0
369	3.639458	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=15338 Win=59904 Len=0
372	3.639741	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=16798 Win=62848 Len=0
375	3.646213	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=18258 Win=65792 Len=0
376	3.646213	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=19718 Win=68736 Len=0
381	3.650307	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=21178 Win=71680 Len=0
382	3.650307	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=22638 Win=74496 Len=0
387	3.665960	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=24098 Win=77440 Len=0
388	3.665960	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=25558 Win=80384 Len=0
389	3.665960	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=27018 Win=83328 Len=0
390	3.665960	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=28478 Win=86272 Len=0
399	3.671951	128.119.245.12	10.20.75.128	TCP	60	80 → 50467 [ACK] Seq=1 Ack=29938 Win=89088 Len=0

Os Ack Numbers estão a subir de forma constante
O recetor confirma 1460 bytes por ACK, ou seja, um segmento de cada vez.
Não existem casos em que o recetor envia ACKs para dois segmentos de cada vez (2920 bytes).
Isso indica que o recetor está a usar um comportamento de ACK por segmento, típico em redes estáveis e rápidas.

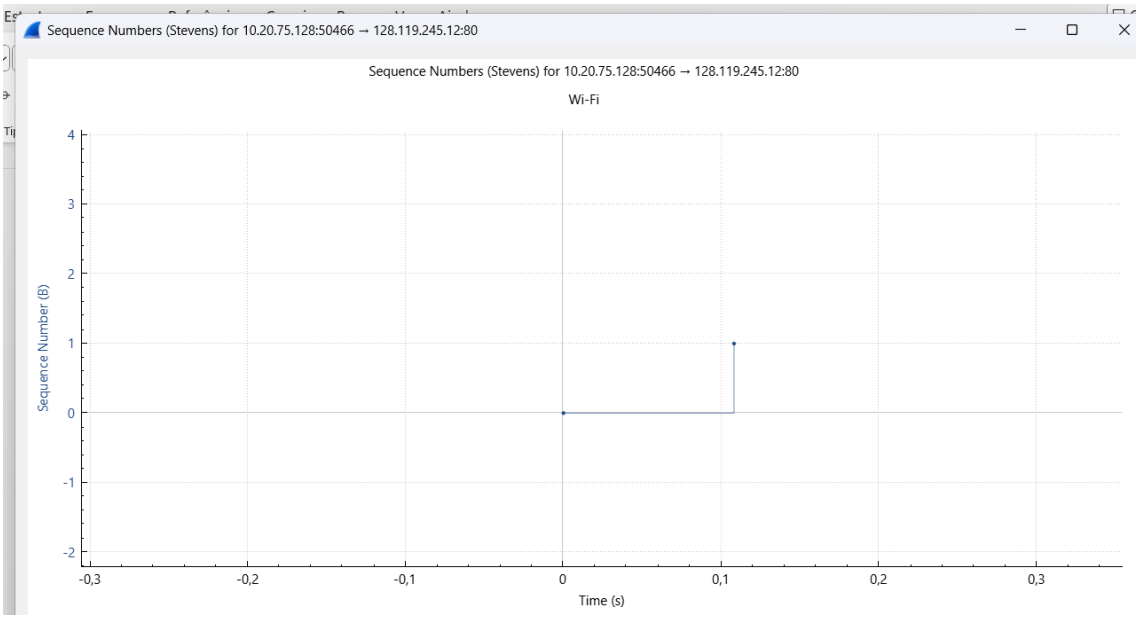
12

Throughput = Total de dados transferidos / Tempo total da transferência
Total de dados transferidos = 29938 - 152578 = 1460 * 10 = 14.600 bytes

13

Na minha própria captura, o gráfico “Time-Sequence Graph (Stevens)” não apresentou dados suficientes para observar claramente as fases de slow start e congestion avoidance.
Isso pode ter acontecido porque:

- A captura foi demasiado curta.
- Ou nem todos os pacotes foram capturados no início ou fim da transmissão.



14

Mesmo sem um gráfico visível, a análise dos segmentos enviados e reconhecidos indica que o TCP iniciou com slow start, enviando pacotes em ritmo acelerado, e depois entrou na fase de congestion avoidance, com ritmo de envio estabilizado e constante — em conformidade com o comportamento esperado do TCP