LABORATÓRIO DE REDES DE COMPUTADORES - T1

Alunos: Eduardo Amengual Garcia e João Vitor Schwingel

1 - Aplicação

A aplicação consiste em dois arquivos, o de server e o de cliente, onde optamos por unificar a aplicação em uma só, possibilitando ao usuário escolher o tipo de socket que quer usar, TCP ou UDP, não sendo necessário assim criar múltiplos arquivos para o projeto. A arquitetura consiste em dois sockets tanto para TCP quando para UDP, um socket para gerenciar o controle, cuidado de operações como login e logout e, um socket para cuidar do transporte de dados, sendo encarregado de mandar as mensagens privadas, broadcast e file.

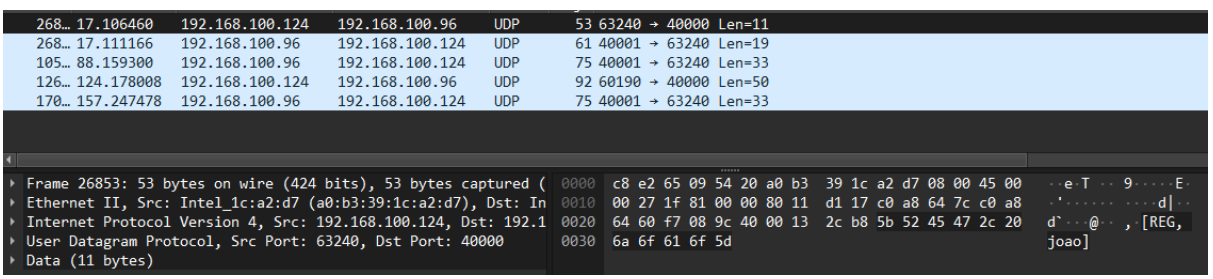
2 - Análise de tráfego

1) Execute o Wireshark para monitorar o tráfego UDP gerado pelo programa. Identifique os pacotes UDP que estão sendo enviados para cada um dos servidores. Quais portas de origem e destino estão sendo utilizadas pelos pacotes?

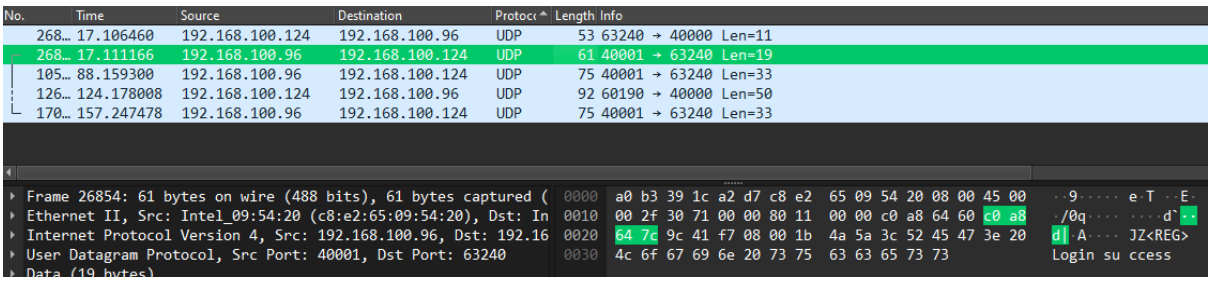
   Resposta: Dado o fluxo de mensagens abaixo:

268…	17.106460	192.168.100.124	192.168.100.96	UDP	53 63240 → 40000 Len=11
268…	17.111166	192.168.100.96	192.168.100.124	UDP	61 40001 → 63240 Len=19
105…	88.159300	192.168.100.96	192.168.100.124	UDP	75 40001 → 63240 Len=33
126…	124.178008	192.168.100.124	192.168.100.96	UDP	92 60190 → 40000 Len=50
170…	157.247478	192.168.100.96	192.168.100.124	UDP	75 40001 → 63240 Len=33

Podemos perceber que a máquina 192.168.128 na porta 63240 (client.py) envia uma mensagem para máquina 192.168.100.96 na porta 40000 (server.py), a mensagem em si, é uma solicitação de login no servidor



Depois o servidor na porta 40001 (porta do servidor udp de server.py) retorna uma mensagem para a máquina que solicitou o login



vale dizer que as portas do servidor são diferentes pois estamos trabalhando com dois sockets, um de controle para tratar os logins (porta 40000) e o outro socket para enviar as mensagens (porta 40001).

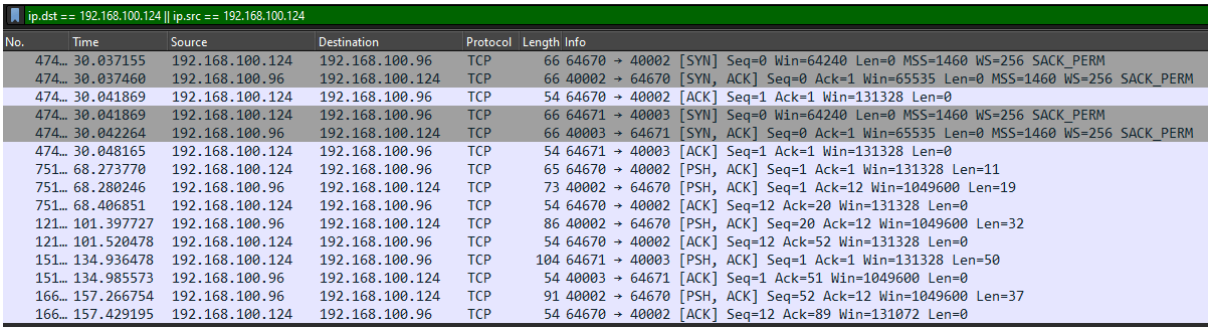
2) Há diferença, em termos de volume de tráfego na rede, entre a aplicação com socket TCP e a aplicação com socket UDP?

Resposta: Sim, há muita diferença, o TCP é naturalmente mais volumoso em questão de tráfego do que o UDP justamente pelo TCP ter um processo de conexão estabelecida e ter outras questões como transmissão de pacotes. Para demonstrar, fizemos o mesmo chat de conversa tanto com o UDP, quanto TCP, onde fizemos o seguinte passo a passo:

- ligamos o servidor na máquina 192.168.100.96
- conectamos um cliente TCP/UDP da máquina 192.168.100.124 chamado joao
- conectamos um cliente TCP/UDP da máquina 192.168.100.96 chamado garcia
- garcia manda mensagem pra joao
- joao responde mensagem para garcia
- garcia responde

Realizando esse bate papo entre as máquinas 192.168.100.96 e 192.168.100.124 temos os seguintes resultados de volume de tráfego:

No.	Time	Source	Destination	Protocol	Length	Info
268…	17.106460	192.168.100.124	192.168.100.96	UDP	53	63240 → 40000 Len=11
268…	17.111166	192.168.100.96	192.168.100.124	UDP	61	40001 → 63240 Len=19
105…	88.159300	192.168.100.96	192.168.100.124	UDP	75	40001 → 63240 Len=33
126…	124.178008	192.168.100.124	192.168.100.96	UDP	92	60190 → 40000 Len=50
170…	157.247478	192.168.100.96	192.168.100.124	UDP	75	40001 → 63240 Len=33



ip.dst == 192.168.100.124 || ip.src == 192.168.100.124

No.	Time	Source	Destination	Protocol	Length	Info
474…	30.037155	192.168.100.124	192.168.100.96	TCP	66	64670 → 40002 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
474…	30.037460	192.168.100.96	192.168.100.124	TCP	66	40002 → 64670 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
474…	30.041869	192.168.100.124	192.168.100.96	TCP	54	64670 → 40002 [ACK] Seq=1 Ack=1 Win=131328 Len=0
474…	30.041869	192.168.100.124	192.168.100.96	TCP	66	64671 → 40003 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
474…	30.042264	192.168.100.96	192.168.100.124	TCP	66	40003 → 64671 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
474…	30.048165	192.168.100.124	192.168.100.96	TCP	54	64671 → 40003 [ACK] Seq=1 Ack=1 Win=131328 Len=0
751…	68.273770	192.168.100.124	192.168.100.96	TCP	65	64670 → 40002 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=11
751…	68.280246	192.168.100.96	192.168.100.124	TCP	73	40002 → 64670 [PSH, ACK] Seq=1 Ack=12 Win=1049600 Len=19
751…	68.406851	192.168.100.124	192.168.100.96	TCP	54	64670 → 40002 [ACK] Seq=12 Ack=20 Win=131328 Len=0
121…	101.397727	192.168.100.96	192.168.100.124	TCP	86	40002 → 64670 [PSH, ACK] Seq=20 Ack=12 Win=1049600 Len=32
121…	101.520478	192.168.100.124	192.168.100.96	TCP	54	64670 → 40002 [ACK] Seq=12 Ack=52 Win=131328 Len=0
151…	134.936478	192.168.100.124	192.168.100.96	TCP	104	64671 → 40003 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=50
151…	134.985573	192.168.100.96	192.168.100.124	TCP	54	40003 → 64671 [ACK] Seq=1 Ack=51 Win=1049600 Len=0
166…	157.266754	192.168.100.96	192.168.100.124	TCP	91	40002 → 64670 [PSH, ACK] Seq=52 Ack=12 Win=1049600 Len=37
166…	157.429195	192.168.100.124	192.168.100.96	TCP	54	64670 → 40002 [ACK] Seq=12 Ack=89 Win=131072 Len=0

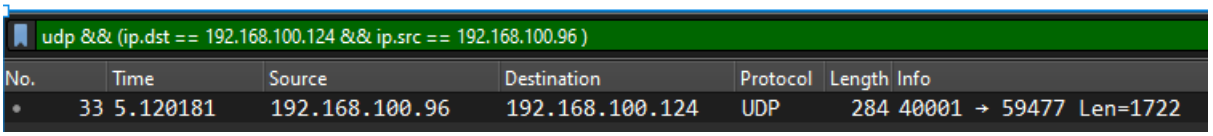
Pelas imagens, confirma-se que o TCP tem um volume de tráfego mais extenso devido ao estabelecimento das conexões

3) Há diferença, em termos de desempenho da aplicação, entre a aplicação com socket TCP e a aplicação com socket UDP?

Resposta: O TCP oferece um desempenho mais confiável: Devido ao controle de fluxo, retransmissão e garantias de entrega, o TCP é mais adequado para aplicações que exigem confiabilidade e integridade dos dados, como em um sistema de chat ou envio de arquivos. No entanto, esse nível de confiabilidade vem ao custo de maior latência e menor taxa de transferência em redes sobrecarregadas ou de baixa qualidade. Já o UDP melhora a velocidade e a latência focando na entrega imediata dos dados porém negligenciando a garantia da entrega

4) Compare a transmissão de um arquivo de 1200 bytes usando a socket TCP e socket UDP.

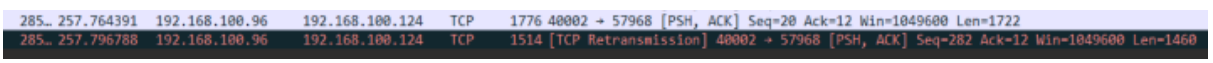
Resposta: A transmissão foi um pouco diferente entre os arquivos, no UDP a length total do pacote (contendo o cabeçalho das outras camadas) enviado contendo o file de 1200 bytes foi de 284



udp && (ip.dst == 192.168.100.124 && ip.src == 192.168.100.96)

No.	Time	Source	Destination	Protocol	Length	Info
33	5.120181	192.168.100.96	192.168.100.124	UDP	284	40001 → 59477 Len=1722

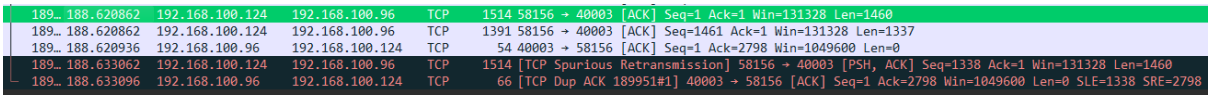
Já usando o TCP houve a necessidade de realizar um reenvio de pacotes e o tamanho total do pacote é bem maior, sendo de 1776.



285…	257.764391	192.168.100.96	192.168.100.124	TCP	1776	40002 → 57968 [PSH, ACK] Seq=20 Ack=12 Win=1049600 Len=1722
285…	257.796788	192.168.100.96	192.168.100.124	TCP	1514	[TCP Retransmission] 40002 → 57968 [PSH, ACK] Seq=282 Ack=12 Win=1049600 Len=1460

5) Compare a transmissão de um arquivo de 2000 bytes usando a socket TCP e socket UDP.

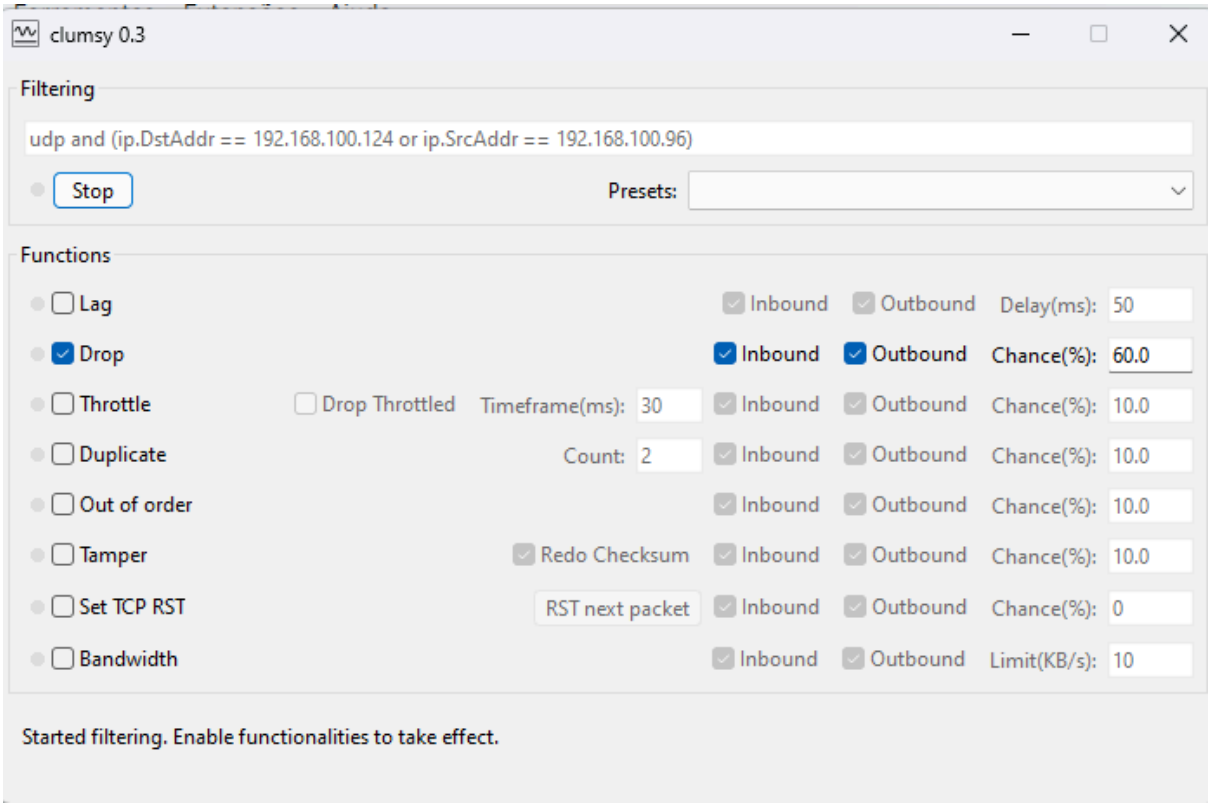
Resposta: Ao mandar um arquivo de 2000 bytes pelo TCP, houve retransmissão e o arquivo chegou em múltiplos pacotes



O pacote verde é o primeiro pacote da transferência do arquivo, onde o Seq=1 e a Len= 1460, depois o próximo pacote, que é a continuação começa com Seq=1461 e Len=1337 e por fim, o terceiro pacote confirmando que todo o conteúdo chegou com Seq=1 e Ack=2789 = (1461+1337). Já no UDP os dados até são vistos na lista do wireshark mas não chegam ao cliente e, como não há nenhum mecanismo de re-envio e garantia de entrega devido ao fato do UDP não ser orientado à conexão, os dados são perdidos.

6) Configure a interface de rede da máquina para incluir perda de pacotes. Qual a diferença, em termos de tráfego na rede, entre o socket TCP e UDP? Houve alguma retransmissão usando alguma retransmissão usando TCP?

Resposta: Configurarmos a interface do Clumsy na máquina do servidor windows (192.168.100.96) e criamos um cliente garcia para tentar mandar mensagens para outro cliente chamado joao na máquina 192.168.100.124



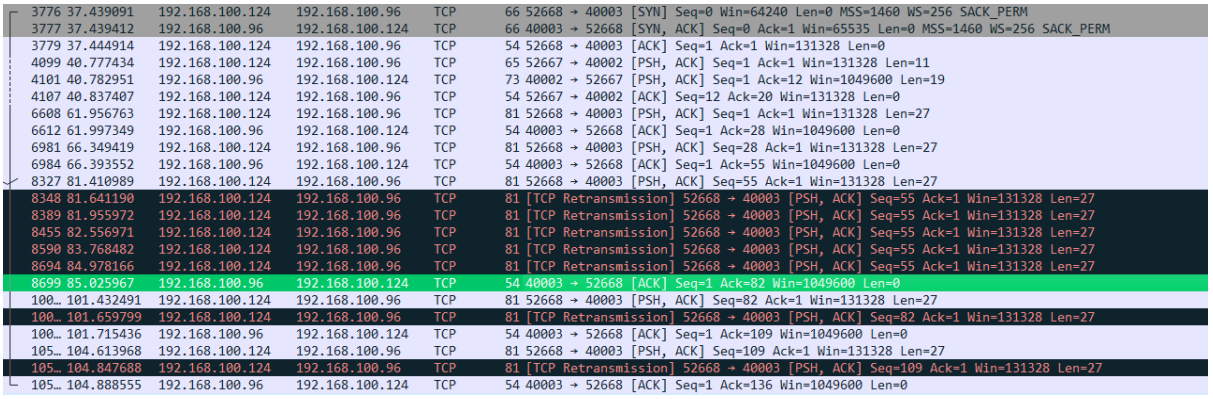
Como pode ser visto pela imagem, configuramos a rede para ter uma chance de 60% de perda de pacotes e segue abaixo o fluxo de mensagens UDP do cliente 192.168.100.96 para o cliente 192.168.100.124

```
dudu on T1_labRedes [ main] - 5m 33s 165ms
# python .\client.py 192.168.100.96 udp
/REG garcia
<REG> Login success
/MSG joao teste1
/MSG joao teste2
/MSG joao teste3
/MSG joao teste4
/MSG joao teste5
/MSG joao teste6
/MSG joao teste7
/MSG joao teste8
/MSG joao teste9
/MSG joao teste10
/MSG joao teste11
/MSG joao teste12
/MSG joao teste13
/MSG joao teste14
/MSG joao teste15
/MSG joao teste16
/MSG joao teste17
/MSG joao teste18
/MSG joao teste19
/MSG joao teste20
```

como podemos ver, mandamos 20 mensagens para o cliente 192.168.100.124, porém ao olharmos no wireshark, apenas 5 mensagens chegaram ao destino: teste2, teste7, teste8, teste15 e teste16. Vale dizer que os dois primeiros pacotes na imagens são respectivamente a máquina pedindo para fazer login e recebendo login success do servidor.

ip.dst == 192.168.100.124 || ip.src == 192.168.100.124

No.	Time	Source	Destination	Protocol	Length	Info
442…	27.477669	192.168.100.124	192.168.100.96	UDP	53	51462 → 40000 Len=11
442…	27.485904	192.168.100.96	192.168.100.124	UDP	61	40001 → 51462 Len=19
709…	57.497483	192.168.100.96	192.168.100.124	UDP	62	40001 → 51462 Len=20
709…	65.191595	192.168.100.96	192.168.100.124	UDP	62	40001 → 51462 Len=20
710…	66.732229	192.168.100.96	192.168.100.124	UDP	62	40001 → 51462 Len=20
893…	78.431961	192.168.100.96	192.168.100.124	UDP	63	40001 → 51462 Len=21
894…	80.221686	192.168.100.96	192.168.100.124	UDP	63	40001 → 51462 Len=21

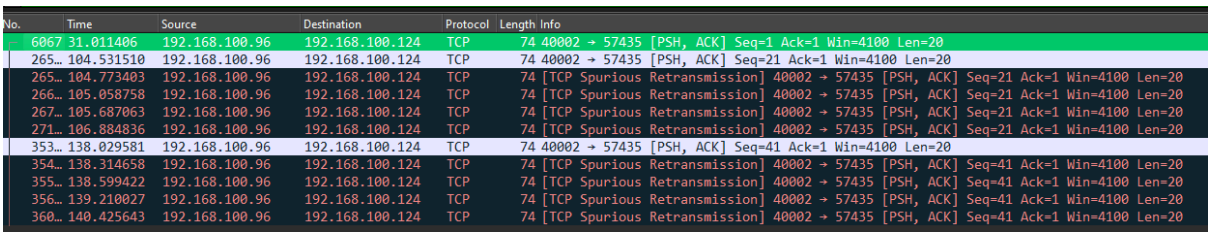


3776	37.439091	192.168.100.124	192.168.100.96	TCP	66 52668 → 40003 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
3777	37.439412	192.168.100.96	192.168.100.124	TCP	66 40003 → 52668 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM
3779	37.444914	192.168.100.124	192.168.100.96	TCP	54 52668 → 40003 [ACK] Seq=1 Ack=1 Win=131328 Len=0
4099	40.777434	192.168.100.124	192.168.100.96	TCP	65 52667 → 40002 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=11
4101	40.782951	192.168.100.96	192.168.100.124	TCP	73 40002 → 52667 [PSH, ACK] Seq=1 Ack=12 Win=1049600 Len=19
4107	40.837407	192.168.100.124	192.168.100.96	TCP	54 52667 → 40002 [ACK] Seq=12 Ack=20 Win=131328 Len=0
6608	61.956763	192.168.100.124	192.168.100.96	TCP	81 52668 → 40003 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=27
6612	61.997349	192.168.100.96	192.168.100.124	TCP	54 40003 → 52668 [ACK] Seq=1 Ack=28 Win=1049600 Len=0
6981	66.349419	192.168.100.124	192.168.100.96	TCP	81 52668 → 40003 [PSH, ACK] Seq=28 Ack=1 Win=131328 Len=27
6984	66.393552	192.168.100.96	192.168.100.124	TCP	54 40003 → 52668 [ACK] Seq=1 Ack=55 Win=1049600 Len=0
8327	81.410989	192.168.100.124	192.168.100.96	TCP	81 52668 → 40003 [PSH, ACK] Seq=55 Ack=1 Win=131328 Len=27
8348	81.641190	192.168.100.124	192.168.100.96	TCP	81 [TCP Retransmission] 52668 → 40003 [PSH, ACK] Seq=55 Ack=1 Win=131328 Len=27
8389	81.955972	192.168.100.124	192.168.100.96	TCP	81 [TCP Retransmission] 52668 → 40003 [PSH, ACK] Seq=55 Ack=1 Win=131328 Len=27
8455	82.556971	192.168.100.124	192.168.100.96	TCP	81 [TCP Retransmission] 52668 → 40003 [PSH, ACK] Seq=55 Ack=1 Win=131328 Len=27
8590	83.768482	192.168.100.124	192.168.100.96	TCP	81 [TCP Retransmission] 52668 → 40003 [PSH, ACK] Seq=55 Ack=1 Win=131328 Len=27
8694	84.978166	192.168.100.124	192.168.100.96	TCP	81 [TCP Retransmission] 52668 → 40003 [PSH, ACK] Seq=55 Ack=1 Win=131328 Len=27
8699	85.025967	192.168.100.96	192.168.100.124	TCP	54 40003 → 52668 [ACK] Seq=1 Ack=82 Win=1049600 Len=0
100…	101.432491	192.168.100.124	192.168.100.96	TCP	81 52668 → 40003 [PSH, ACK] Seq=82 Ack=1 Win=131328 Len=27
100…	101.659799	192.168.100.124	192.168.100.96	TCP	81 [TCP Retransmission] 52668 → 40003 [PSH, ACK] Seq=82 Ack=1 Win=131328 Len=27
100…	101.715436	192.168.100.96	192.168.100.124	TCP	54 40003 → 52668 [ACK] Seq=1 Ack=109 Win=1049600 Len=0
105…	104.613968	192.168.100.124	192.168.100.96	TCP	81 52668 → 40003 [PSH, ACK] Seq=109 Ack=1 Win=131328 Len=27
105…	104.847688	192.168.100.124	192.168.100.96	TCP	81 [TCP Retransmission] 52668 → 40003 [PSH, ACK] Seq=109 Ack=1 Win=131328 Len=27
105…	104.888555	192.168.100.96	192.168.100.124	TCP	54 40003 → 52668 [ACK] Seq=1 Ack=136 Win=1049600 Len=0

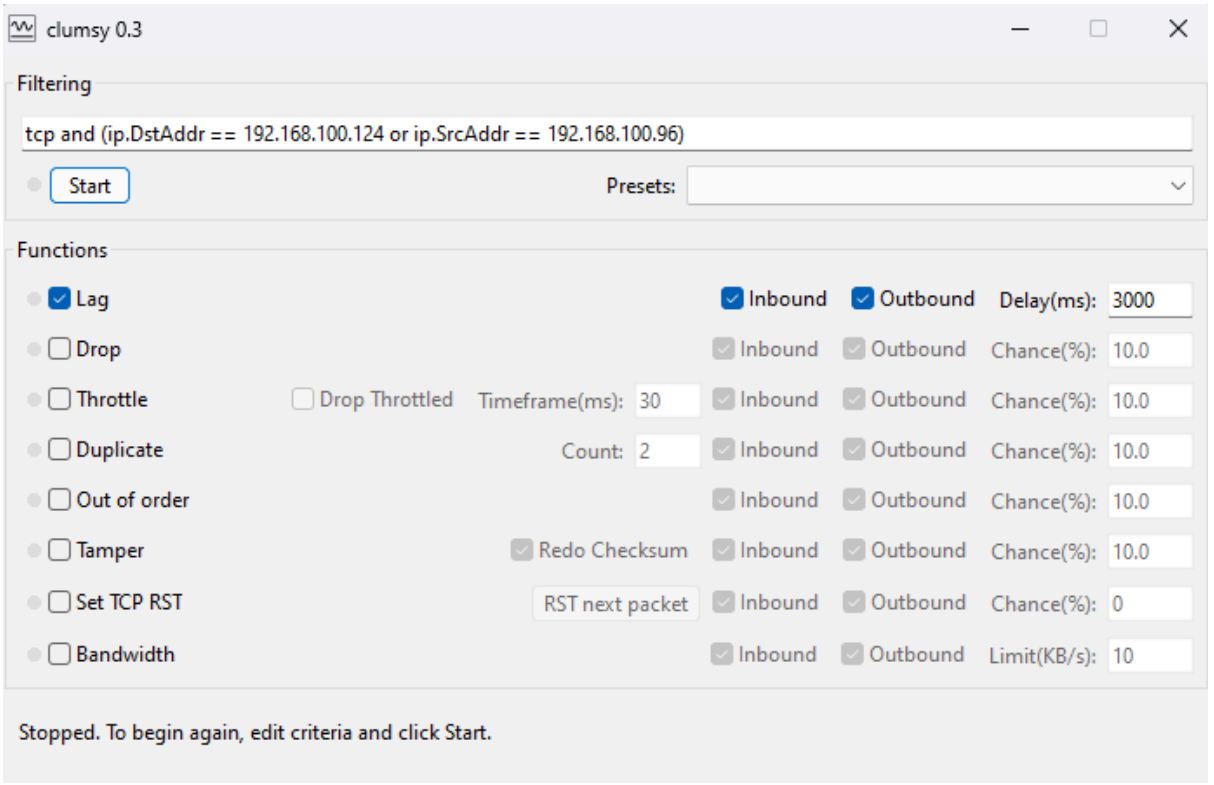
Já na figura acima, podemos ver que o TCP ao perder um pacote, fica tentando transmitir o pacote até receber um ack do servidor informando o recebimento, garantindo que mandamos todas as mensagens e estas são recebidas com sucesso.

7) Configurar a interface de rede da máquina para incluir latência variável. Qual a diferença, em termos de tráfego na rede, entre o socket TCP e UDP? Houve alguma retransmissão usando TCP?

   Resposta: Em termos de tráfego de tráfego de rede, o UDP não apresentou diferenças comparado ao tráfego sem o Clumsy configurada, as mensagens apenas demoravam para chegar ao destino. Porém já no TCP houve sim uma diferença. O remetente envia o pacote original porém, graças a latência adicionada ele demora a receber um ack e retransmite o pacote várias vezes porém, antes desses pacotes retransmitidos chegarem, o pacote original chega ao seu destino, tornando os pacotes retransmitidos “Spurious Retransmission” como pode ser visto na imagem abaixo.



No.	Time	Source	Destination	Protocol	Length	Info
6067	31.011406	192.168.100.96	192.168.100.124	TCP	74	40002 → 57435 [PSH, ACK] Seq=1 Ack=1 Win=4100 Len=20
265…	104.531510	192.168.100.96	192.168.100.124	TCP	74	40002 → 57435 [PSH, ACK] Seq=21 Ack=1 Win=4100 Len=20
265…	104.773403	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=21 Ack=1 Win=4100 Len=20
266…	105.058758	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=21 Ack=1 Win=4100 Len=20
267…	105.687063	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=21 Ack=1 Win=4100 Len=20
271…	106.884836	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=21 Ack=1 Win=4100 Len=20
353…	138.029581	192.168.100.96	192.168.100.124	TCP	74	40002 → 57435 [PSH, ACK] Seq=41 Ack=1 Win=4100 Len=20
354…	138.314658	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=41 Ack=1 Win=4100 Len=20
355…	138.599422	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=41 Ack=1 Win=4100 Len=20
356…	139.210027	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=41 Ack=1 Win=4100 Len=20
360…	140.425643	192.168.100.96	192.168.100.124	TCP	74	[TCP Spurious Retransmission] 40002 → 57435 [PSH, ACK] Seq=41 Ack=1 Win=4100 Len=20


clumsy 0.3
Filtering
tcp and (ip.DstAddr == 192.168.100.124 or ip.SrcAddr == 192.168.100.96)
Start
Presets:
Functions
Lag
Inbound
Outbound
Delay(ms):
3000
Drop
Inbound
Outbound
Chance(%):
10.0
Throttle
Drop Throttled
Timeframe(ms):
30
Inbound
Outbound
Chance(%):
10.0
Duplicate
Count:
2
Inbound
Outbound
Chance(%):
10.0
Out of order
Inbound
Outbound
Chance(%):
10.0
Tamper
Redo Checksum
Inbound
Outbound
Chance(%):
10.0
Set TCP RST
RST next packet
Inbound
Outbound
Chance(%):
0
Bandwidth
Inbound
Outbound
Limit(KB/s):
10
Stopped. To begin again, edit criteria and click Start.