

## Relatório da Elaboração de Sockets para UDP e TCP

Nome(s): Gabriel Fanto Stundner, Luiz Guerra

Data: 14/09/2020

Link do Repositório: <https://github.com/F4NT0/sockets-cliente-servidor>

### 1) Rodando Socket TCP com arquivo menor que 1500 bytes

Quando inicializado o Wireshark e inicializado o Servidor e o Cliente a primeira coisa que aparece é uma conexão onde o meu IP da máquina (192.168.0.21) está fazendo uma conexão externa no IP 35.222.85.5, onde está pegando a porta interna 46434 e enviando para a porta externa 80 (de HTTP)

A imagem abaixo é todas as conexões feitas no envio do menor arquivo

210	40.557332333	192.168.0.21	35.222.85.5	TCP	74 46434 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM...
212	40.724926182	192.168.0.21	35.222.85.5	TCP	66 46434 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=318203...
213	40.725194007	192.168.0.21	35.222.85.5	HTTP	153 GET / HTTP/1.1
216	40.892820635	192.168.0.21	35.222.85.5	TCP	78 [TCP Dup ACK 212#1] 46434 → 80 [ACK] Seq=88 Ack=1 Win=642...
219	40.892954951	192.168.0.21	35.222.85.5	TCP	66 46434 → 80 [ACK] Seq=88 Ack=150 Win=64128 Len=0 TSval=318...
220	40.893276243	192.168.0.21	35.222.85.5	TCP	66 46434 → 80 [FIN, ACK] Seq=88 Ack=150 Win=64128 Len=0 TSva...
211	40.724818086	35.222.85.5	192.168.0.21	TCP	74 80 → 46434 [SYN, ACK] Seq=0 Ack=1 Win=28160 Len=0 MSS=142...
214	40.892304791	35.222.85.5	192.168.0.21	TCP	66 80 → 46434 [ACK] Seq=1 Ack=88 Win=29312 Len=0 TSval=34655...
215	40.892772115	35.222.85.5	192.168.0.21	TCP	66 [TCP Previous segment not captured] 80 → 46434 [FIN, ACK]...
217	40.892885122	35.222.85.5	192.168.0.21	TCP	66 80 → 46434 [ACK] Seq=1 Ack=88 Win=29312 Len=0 TSval=34655...
218	40.892906611	35.222.85.5	192.168.0.21	TCP	214 [TCP Out-Of-Order] 80 → 46434 [PSH, ACK] Seq=1 Ack=88 Win...
221	41.070460814	35.222.85.5	192.168.0.21	TCP	66 80 → 46434 [ACK] Seq=150 Ack=89 Win=29312 Len=0 TSval=346...

Quando iniciado o Servidor, ele envia da porta definida do meu programa para a porta 80 de saída (talvez devido que eu tenha escolhido a porta 6789), assim que o servidor é iniciado ele espera um arquivo de retorno pela mesma porta. Quando iniciamos o Cliente, o programa pergunta se queremos enviar o arquivo pequeno ou o arquivo grande pré-definidos.

Como mostrado acima, quando selecionamos a primeira conexão bem no canto Esquerdo possui uma linha, mostrando que todas as conexões estão conectadas, onde cada conexão é um passo da conexão.

A conexão de Número **210** é a conexão de acesso para o localhost quando iniciamos o Servidor, depois na conexão de Número **212** é o Acknowledgement (confirmação) desse acesso externo.

Na conexão **213** já é a conexão externa, onde é feito um GET no servidor, onde agora temos a confirmação que está iniciado.

Da conexão **219** a **214** são a inicialização do Cliente e a chamada pelo arquivo menor, onde o arquivo vai ser aberto e lido linha por linha, onde cada linha vai sendo enviada para o Servidor e o Servidor vai enviando uma Resposta de que recebeu a linha para o cliente.

Abaixo a imagem do Servidor e do Cliente depois de rodado para ser capturado no Wireshark

```
M4TRIX:TCP (master) > ./RunServer.sh
Servidor Iniciado!
Iniciado o Processo de Leitura
[ VALOR RECEBIDO ] : Exemplo de Envio de dados para o Servidor
[ VALOR RECEBIDO ] : Somente esta sendo lido as linhas desse arquivo
[ VALOR RECEBIDO ] : uuuuu! funcionou!
Processo de Leitura Finalizado!
M4TRIX:TCP (master) >

M4TRIX:TCP (master) > ./RunClient.sh
Conectando ao servidor localhost:6789
Digite o nome do Arquivo que deseja enviar [teste-pequeno | teste-grande]
teste-pequeno
Dado sendo Enviado: Exemplo de Envio de dados para o Servidor
Dado sendo Enviado: Somente esta sendo lido as linhas desse arquivo
Dado sendo Enviado: uuuuu! funcionou!
Dado sendo Enviado: end
Recebido do servidor: [ OK ] Leitura Realizado com Sucesso
M4TRIX:TCP (master) >
```

O programa para de enviar quando chega na palavra *end* dentro do arquivo texto, assim ele para o envio do cliente e fecha o Socket, causando também a finalização do Servidor, para que ele não fique rodando sem motivos.

## 2) Rodando Socket TCP com Arquivo maior que 10000 bytes

Para fazer esse teste pegamos um arquivo bem grande de texto para enviar para o Servidor, onde nos deu o seguinte resultado:

2010	18.190531896	192.168.0.21	64.4.54.254	TCP	74 47654 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SA...
2034	18.381297751	64.4.54.254	192.168.0.21	TCP	74 443 → 47654 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 M...
2035	18.381370602	192.168.0.21	64.4.54.254	TCP	66 47654 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval...
2061	18.592285260	64.4.54.254	192.168.0.21	TCP	1514 443 → 47654 [ACK] Seq=1 Ack=518 Win=263424 Len=1448...
2062	18.592365607	192.168.0.21	64.4.54.254	TCP	66 47654 → 443 [ACK] Seq=518 Ack=1449 Win=64128 Len=0 ...
2063	18.593118425	64.4.54.254	192.168.0.21	TCP	1514 443 → 47654 [ACK] Seq=1449 Ack=518 Win=263424 Len=1...
2064	18.593151546	192.168.0.21	64.4.54.254	TCP	66 47654 → 443 [ACK] Seq=518 Ack=2897 Win=64128 Len=0 ...
2066	18.593265851	192.168.0.21	64.4.54.254	TCP	66 47654 → 443 [ACK] Seq=518 Ack=4026 Win=63104 Len=0 ...
2090	18.790345186	192.168.0.21	64.4.54.254	TCP	66 47654 → 443 [ACK] Seq=611 Ack=4077 Win=64128 Len=0 ...
2112	18.981613255	64.4.54.254	192.168.0.21	TCP	66 443 → 47654 [ACK] Seq=4077 Ack=1426 Win=262400 Len=...
2121	19.038273908	192.168.0.21	64.4.54.254	TCP	66 47654 → 443 [FIN, ACK] Seq=1457 Ack=4392 Win=64128 ...
2143	19.224647070	64.4.54.254	192.168.0.21	TCP	66 [TCP Dup ACK 2112#1] 443 → 47654 [ACK] Seq=4392 Ack=...
2144	19.225027923	64.4.54.254	192.168.0.21	TCP	66 443 → 47654 [ACK] Seq=4392 Ack=1458 Win=262400 Len=...

Deu um maior numero de acessos que o arquivo menor, onde o IP de destino da minha maquina foi para o IP **64.4.54.254**.

A maior parte do acesso é ACK para confirmar o envio, mas a maior mudança do arquivo menor para o maior é que como ele le mais linhas e envia um por um é maior o número de envios para o externo.

A imagem da conexão são as seguintes:

<b>M4TRIX:TCP (master) &gt; ./RunServer.sh</b> Servidor Iniciado! Iniciado o Processo de Leitura [ VALOR RECEBIDO ] : Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Viverra nam libero justo laoreet sit amet cursus sit. Elit at imperdiet dui accumsan sit amet nulla facilisi morbi. Sapien faucibus et molestie ac feugiat. Enim neque volutpat ac tincidunt. Condimentum id venenatis a condimentum vitae sapien pellentesque habitant. Dui nunc mattis enim ut. Enim neque volutpat ac tincidunt vitae. Hendrerit dolor magna eget est lorem. Vitae proin sagittis nisl rhoncus. Ullamcorper dignissim cras tincidunt lobortis. Tempor orci dapibus ultrices in iaculis nunc sed. [ VALOR RECEBIDO ] : Mauris pellentesque pulvinar pellentesque habitant morbi tristique senectus et netus. Nibh sed pulvinar proin gravida. Venenatis cras sed felis eget velit aliquet. Faucibus in ornare quam viverra orci sagittis eu volutpat. Est ullamcorper eget nulla facilisi etiam dignissim diam quis enim. Cursum turpis massa tincidunt dui ut. Enim nunc faucibus a pellentesque sit amet. Felis bibendum ut tristique et egestas quis ipsum. At va	<b>M4TRIX:TCP (master) &gt; ./RunClient.sh</b> Conectando ao servidor localhost:6789 Digite o nome do Arquivo que deseja enviar [teste-pequeno   teste-grande] teste-grande Dado sendo Enviado: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Viverra nam libero justo laoreet sit amet cursus sit. Elit at imperdiet dui accumsan sit amet nulla facilisi morbi. Sapien faucibus et molestie ac feugiat. Enim neque volutpat ac tincidunt. Condimentum id venenatis a condimentum vitae sapien pellentesque habitant. Dui nunc mattis enim ut. Enim neque volutpat ac tincidunt vitae. Hendrerit dolor magna eget est lorem. Vitae proin sagittis nisl rhoncus. Ullamcorper dignissim cras tincidunt lobortis. Tempor orci dapibus ultrices in iaculis nunc sed. Dado sendo Enviado: Mauris pellentesque pulvinar pellentesque habitant morbi tristique senectus et netus. Nibh sed pulvinar proin gravida. Venenatis cras sed felis eget velit aliquet. Faucibus in ornare quam viverra orci sagittis eu volutpat. Est ullamcorper eget nulla facilisi etiam dignissim diam quis enim. Cursum turpis massa tincidunt dui ut. Enim nunc f
--	---

Não vai caber todo o texto enviado, devido que é um texto de 49 linhas.

## 3) Rodando Socket UDP com arquivo menor que 1500 bytes

Quando tentei rodar o Socket UDP, por algum motivo ele não pegou o protocolo UDP pela minha maquina, mesmo tendo rodado a leitura ele não me deu a resposta do protocolo UDP e sim um DNS, como abaixo:

286	41.417114890	192.168.0.21	201.21.192.112	DNS	108 Standard query 0x70e5 A asimov.vortex.data.trafficm...
287	41.417682903	192.168.0.21	201.21.192.112	DNS	108 Standard query 0x9c09 AAAA asimov.vortex.data.traff...
288	41.429247098	201.21.192.112	192.168.0.21	DNS	145 Standard query response 0x70e5 A asimov.vortex.data...

```

▶ Frame 286: 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface wlp6s0, id 0
▶ Ethernet II, Src: HonHaiPr_fc:9d:2b (70:18:8b:fc:9d:2b), Dst: HUMAX_b8:9d:0e (4c:d0:8a:b8:9d:0e)
▶ Internet Protocol Version 4, Src: 192.168.0.21, Dst: 201.21.192.112
▼ User Datagram Protocol, Src Port: 37451, Dst Port: 53
    Source Port: 37451
    Destination Port: 53
    Length: 74
    Checksum: 0x0928 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 10]
    ▶ [Timestamps]
▶ Domain Name System (query)

```

Mas vendo melhor dentro do DNS eu vi que tinha tido um Acesso UDP, onde isso significa que o Socket UDP possui um DNS também, onde quando foi feito a conexão deu essa resposta.

#### 4) Rodando Socket UDP com arquivo maior que 10000 bytes

Agora o teste do arquivo maior, onde me deu o seguinte resultado:

140	17.004628258	192.168.0.21	201.21.192.112	DNS	108 Standard query 0xba11 A asimov.vortex.data.trafficm...
141	17.005242446	192.168.0.21	201.21.192.112	DNS	108 Standard query 0x095d AAAA asimov.vortex.data.traff...
142	17.928014423	192.168.0.21	201.21.192.164	DNS	97 Standard query 0xba11 A asimov.vortex.data.trafficm...
151	23.177761845	2804:14d:4c88:8108:8982:8...	2804:14d:4c10:672:2...	DNS	128 Standard query 0xba11 A asimov.vortex.data.trafficm...
152	23.193710001	2804:14d:4c10:672:201:21:...	2804:14d:4c88:8108:...	DNS	165 Standard query response 0xba11 A asimov.vortex.data...

```

▶ Frame 140: 108 bytes on wire (864 bits), 108 bytes captured (864 bits) on interface wlp6s0, id 0
▶ Ethernet II, Src: HonHaiPr_fc:9d:2b (70:18:8b:fc:9d:2b), Dst: HUMAX_b8:9d:0e (4c:d0:8a:b8:9d:0e)
▶ Internet Protocol Version 4, Src: 192.168.0.21, Dst: 201.21.192.112
▼ User Datagram Protocol, Src Port: 49313, Dst Port: 53
    Source Port: 49313
    Destination Port: 53
    Length: 74
    Checksum: 0x91a5 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 14]
    ▶ [Timestamps]
▶ Domain Name System (query)

```

vendo sobre o arquivo pequeno para o grande vejo que continua a questão do DNS no acesso e retorno, mas teve 3 conexões como as do arquivo pequeno mas também teve 2 conexões de uma porta que não sei qual é, mas parece ser uma resposta global de DNS.

a) O tráfego de rede do UDP construi um Domain Name System quando é feito a conexão, diferente do TCP, que faz mais conexões quando o arquivo é maior mas não possui uma construção de DNS como UDP, onde UDP precisa de mais informação que o TCP na hora que foi construído o programa

b) TCP: são necessários somente um pacote para enviar o TCP, onde ele não cuida do acesso, sendo mais fácil de enviar

UDP: são necessários mais de um pacote e é criado um acesso com DNS para a saída