

## MC833 - Tarefa 6

Erik de Godoy Perillo - RA: 135582

1 de junho de 2016

1. –

- `ssize_t recvfrom(int sockfd, void* buf, size_t len, int flags, struct sockaddr* src_addr, socklen_t *addrlen)`

Essa função pode ser usada para receber mensagens pela rede por protocolos sem conexão. Por `sockfd` especifica-se o *socket* do sistema operacional a se “ouvir” na espera de uma nova mensagem, que é guardada em `buf` e tem o tamanho máximo especificado por `len`. `flags` pode ser usado para especificar opções sobre o comportamento da função, como setá-la para *blocking/nonblocking*, por exemplo. Se não forem `NULL`, `src_addr` e `addrlen` são preenchidos com o endereço de quem enviou a mensagem e o tamanho da estrutura que representa o endereço, respectivamente. O valor retornado vai ser o número de *bytes* lidos em caso de sucesso ou -1 caso contrário.

- `ssize_t sendto(int sockfd, const void* buf, size_t len, int flags, const struct sockaddr* dest_addr, socklen_t addrlen)`

A contraparte de `recvfrom`. Pode ser usada para mandar mensagens pela rede por protocolos sem conexão. Por `sockfd` especifica-se o *socket* do sistema operacional a se usar para mandar a mensagem, que é especificada por `buf` e tem tamanho em *bytes* dado por `len`. Por `flags` pode-se configurar algumas partes do comportamento da função, como setar para *blocking/nonblocking*. O endereço destino da mensagem é especificado por `dest_addr` e seu tamanho é especificado em `addrlen`. Em sucesso, retorna o número de bytes enviados. Se falhar, retorna -1.

2. O servidor e cliente foram implementados. Uma demonstração é ilustrada abaixo, onde o cliente feito e o programa `nc` são usados ao mesmo

tempo para mandar mensagens ao servidor, que as manda de volta para os respectivos clientes:

<pre>trab_6  ./server_udp 1234 [received from 127.0.0.1] oi [received from 127.0.0.1] uau!</pre>	<pre>trab_6  ./client_udp localhost 1234 &gt;&gt;&gt; oi [received from 101.0.95.95] oi &gt;&gt;&gt;  ~  nc 127.0.0.1 1234 -u uau! uau!</pre>
--	---

3. A principal diferença entre o protocolo UDP e o TCP é que o primeiro não é orientado a conexão, enquanto o outro é. Assim, UDP é efêmero no sentido que só existe a troca da mensagem entre um *host* e outro, enquanto que no TCP uma conexão é mantida até que não se queira/possa mais trocar mensagens. O UDP usa o método conhecido como *best effort*, isto é: tenta-se enviar a mensagem ao destinatário, mas nada é garantido. Em TCP, há diversas medidas que asseguram que a mensagem foi enviada ao destinatário, como mensagens de confirmação etc. Em TCP também garante-se que os pacotes chegam em ordem, em UDP, não. Por esses motivos e outros, o TCP é naturalmente mais complexo que o UDP. Tende-se a usar o TCP em casos em que a) pode-se pagar pelo *overhead* e b) é importante que todas as mensagens cheguem e o façam em ordem. UDP é mais usado em casos onde não é terrível a perda de alguns pacotes e um custo computacional baixo é apreciado/necessário.
4. O servidor foi modificado para mostrar a porta de quem o envia as mensagens. Assim, por meio do uso do `nc`, pôde-se enviar uma mensagem para o cliente, a qual ele aceitou sem problema algum.

<pre>trab_6  ./server_udp 1234 [received from 127.0.0.1:42015] oi [received from 127.0.0.1:42015] alo server</pre>	<pre>trab_6  ./client_udp localhost 1234 &gt;&gt;&gt; oi [received from 127.0.0.1:1234] oi &gt;&gt;&gt; alo server [received from 127.0.0.1:33864] oi sou o server :) &gt;&gt;&gt; ■  trab_6  nc -u 127.0.0.1 42015 oi sou o server :)</pre>
--	--

5. O cliente foi modificado para checar o endereço/porta de quem o man-

dou a mensagem. Se eles forem de algum modo diferentes dos especificados como sendo do servidor, a mensagem é ignorada e um aviso é mostrado, como ilustrado abaixo:

<pre>trab_6  ./server_udp 1234 [received from 127.0.0.1:52598] ola [received from 127.0.0.1:52598] hey b0ss</pre>	<pre>trab_6  ./client_udp 127.0.0.1 1234 &gt;&gt;&gt; ola [received from 127.0.0.1:1234] ola &gt;&gt;&gt; hey b0ss someone is pretending to be the server and I'm scared. the perverse person's address is 127.0.0.1:49120 &gt;&gt;&gt; █</pre>
	<pre>trab_6  nc -u 127.0.0.1 52598 oi sou o server :D</pre>

## 1 Erros assíncronos e sockets UDP conectados

A ferramenta `tcpdump` foi utilizada para monitorar a interface *loopback* do sistema. De fato, quando enviou-se um pacote UDP para o servidor sem ele estar ativo, a mensagem de erro ICMP apareceu pouco depois.

<pre>tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes 04:37:46.239375 IP localhost.localdomain.33806 &gt; localhost.localdomain.search-agent: UDP, length 5 04:37:46.239396 IP localhost.localdomain &gt; localhost.localdomain: ICMP localhost.localdomain udp port search-agent unreachable, length 41</pre>	
<pre>trab_6 </pre>	<pre>trab_6  ./client_udp 127.0.0.1 1234 &gt;&gt;&gt; oi? █</pre>

6. –

- (a) As mensagens do `tcpdump` não ficaram diferentes, mas o cliente conectado mostrou um erro, o que não ocorreu com a versão anterior. Para o `tcpdump` foi indiferente pois, na camada de transporte, as coisas continuam as mesmas, ou seja, o protocolo se comporta da mesma forma.

<pre>~  sudo tcpdump -i lo tcpdump: verbose output suppressed, use -v or -vv for full protocol decode listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes 04:53:06.062685 IP localhost.localdomain.50242 &gt; localhost.localdomain.search-agent: UDP, length 4 04:53:06.062712 IP localhost.localdomain &gt; localhost.localdomain: ICMP localhost.localdomain udp port search-agent unreachable, length 40</pre>	
<pre>trab_6 </pre>	<pre>trab_6  ./client_udp_connected 127.0.0.1 1234 &gt;&gt;&gt; oi [udp_client] error receiving message! : connection refused trab_6  █</pre>

- (b) Rodando o servidor, depois o cliente conectado, o nc caiu. Usando ncat, um simular que tem mensagens mais informativas, obteve-se um erro de *connection refused*.

```
trab_6| ./server_udp 1234
[received from 127.0.0.1:35428] oi
[received from 127.0.0.1:35428] alo

>>> alo
received from server: alo
>>>

trab_6| nc -u 127.0.0.1 35428
oi
oieeeee
trab_6| ncat -v -u 127.0.0.1 35428
Ncat: Version 7.12 ( https://nmap.org/ncat )
Ncat: Connected to 127.0.0.1:35428.
alo
Ncat: Connection refused.
trab_6|
```

7. Há uma pequena diferença constante de desempenho (testei em torno de 8 vezes) onde a versão conectada é algo em torno de um segundo mais rápida que a não conectada para o envio de dez milhões de pacotes. Fontes que pesquisei [1] indicam que o socket UDP conectado passa por menos etapas para o envio de uma mensagem, pois já sabe de antemão o seu destino.

```
trab_6| time ./perf_client_udp localhost 1234

real    0m35.729s
user    0m0.883s
sys     0m34.823s
trab_6|

trab_6| time ./perf_client_udp_connected localhost 1234

real    0m34.996s
user    0m0.930s
sys     0m34.030s
trab_6|
```

## Referências

- [1] <http://goo.gl/uC6QSa>