

Re: recvfrom() strange operation

Source: <http://linux.derkeiler.com/Newsgroups/comp.os.linux.development.system/2006-09/msg00076.html>

- *From:* kzsolt@xxxxxxxxxxx
 - *Date:* 9 Sep 2006 02:28:13 -0700
-

"3) Have the ..."

But in that case not working, because the framing always require buffer scroll (for escape sequence). Other side our task is "using with standard calls", this mean we cannot go deeply to kernel, and we can migrate any time to x86solaris.

I have only one escape way for this kind of UDP operation. In case of TCP receive you always send data to upper level in sequence, because you know the packet squence. In case of UDP you do not know the packet sequence, maybe at revc() octett_n and octett_n+1 sended in reverse (IP network nature). This is maybe the only reason why UDP do not enable to read packet as octett only as packet.

But "application" can (must) discover packet squence reverse from octett stream, so this is pain for us....

Other sides the UDP packet sequence problem littlebit mistificated. If you have complex IP network (maybe generate packet squence reverse) then you must use TCP instead of UDP. Otherwords in this kind of network UDP has more disadvantage then advantage. If you have easy network topology you must use UDP instead of TCP because the troughput advantage.

In case of standard IP services you have no way to select which host use UDP and which TCP for the same service. Our application can do it. If a network have potential to serve it we make mouch higher troughput and lower delay than TCP. Our higher level applications need it...

steve_schefter@xxxxxxxxxxx írta:

kzsolt@xxxxxxxxxxx wrote:

"Any other copying as it moves through the layers of the protocol stack is probably a poor design."

If you think about all of the usefull framing algorithm you have two way:

- 1). "other copying"
- 2). read data byte by byte...

3) Have the lower level of the stack read the complete message

Re: recvfrom() strange operation

into a buffer accessible by all layers and then pass on the address of the data rather than doing a copy. That's the way most protocol stacks these days work.

"then why would this same code exist within user space"

I developed a full IP protocol stack including UDP and TCP for our company. This has no relation to x86 and unix kernel, but I know it from inside.

To receive UDP or TCP packet you need some buffering of received data. This mean the packet arrived with n byte user data, and the user can read it sequecially by segmenets or by octetts. This is a same algo like you read sequecially from the file (blocks). If this algo used for buffering of TCP packet I do not undersatnd why not used for UDP packets.

That's because TCP is a streaming (byte-oriented) protocol while UDP is a message-oriented protocol. If the kernel did what you wanted, then I as a programmer could not UDP recv() and know that what I got was sent as a single message by the remote. I may need to. It's designed this way on purpose, it just doesn't suit your purpose.

Reading between the lines, it appears as though you are moving from kernel space to user space just to be able to access Linux sockets. This is not necessary. You can access sockets in the kernel. See tux for example.

Regards,
Steve

Steve Schefter phone: +1 705 725 9999 x26
The Software Group Limited fax: +1 705 725 9666
642 Welham Road,
Barrie, Ontario CANADA L4N 9A1 Web: www.wanware.com