Network Working Group
Request for Comments # 451
NIC # 14135

M.A. Padlipsky
MIT-Multics
February 22, 1973

Tentative Proposal for a Unified User Level Protocol

Now that proposals for expansions to the Telnet Protocol are in vogue again (RFC's 426 and 435, for example), I'd like to promote some discussion of a particular favorite of my own. Please note that this is presented as a tentative proposal: it's an attempt to consider the desirability of a new approach, not a rigorous specification. To begin somewhat obliquely, for some time I've felt that we (the NWG) have fallen into a trap in regard to the Initial Connection Protocol. The point is that even though the ICP gives us the ability to define a "family" of ICPlets by varying the contact socket, there's no compelling reason why we should do so. That we have done so in the FTP and RJEP I view as unfortunate--but also undesirable and unnecessary.

To take the "undesirable" aspects first, consider the following: If we continue to define a new contact socket for every new "user level" protocol we come up with, we'll continue to need another new mechanism (process, procedure, or patch) to respond to requests for connection for each new protocol. By Occam's Razor (or the principle of economy of mechanism, if you prefer), this is a bad thing. Irrespective of the relative difficulty of implementing such mechanisms on the various Hosts, to implement them at all leads to a kind of conceptual clutter. Further, a different kind of confusion is introduced by the notion which some of our number seem to be entertaining, that the "later" user level protocols such as FTP are somehow still another level of abstraction up from Telnet. So it seems to me that we could spare ourselves a lot of bother, both practical and theoretical, if we could avoid spawning contact sockets needlessly.

Turning to the "unnecessary" aspects, I think that even if the case against the current approach isn't completely convincing the case for a particular alternative might be. So to show that the multiple contact socket ICP is unnecessary, I'll try to show that what I call the "unified user level protocol" (UULP) is better. The first thing to notice is that all the "later" protocols "speak Telnet". This is sensible: Telnet works, by and large. Why not make use of it? Right. But why not make even more use of it? In view of the fact that FTP, RJEP, and even the initiating part of the Network Graphics Protocol, are really just ways of letting a user say to a Server "I don't know what you call it on your system, but please perform the whatever function (push or pull a file, start or stop a batch job, funnel some of my output through the Network Virtual Graphics Terminal module) for me now,

why not simply put hooks in Telnet to indicate that a Network Generic Function is wanted instead of a Host-specific one at a given point in time? Then everybody can come in through Telnet in ways that are already known (and usually debugged and optimized) and fan out to other services through a single mechanism, where that single mechanism can be whatever is most appropriate to a particular Host. This view has the additional virtue of keeping the Host "Answering Service"-equivalent processes out of the act when new protocols come along -- where by Answering Service, I mean that process which manages logins in general for a given Host. This process is, of course, a particularly sensitive one on those systems which worry about accounting and security.

That's all probably a bit vague. Perhaps some idea of how I think the UULP would work will cast some light on what I think it is. What's needed is a way of letting the Server know that it's being given a generic command (I decline to call it a Network Virtual command, but I'm afraid that might be what I mean) like "STOR" or "RETR" rather than a local command like "who" or "sys". What could be simpler than defining a Telnet Control Code (TCC) for "Network Generic Function Follows"? So if the Server Telnet receives a command line beginning with the NGF TCC (say, 277 octal), it just feeds the line to the appropriate process or procedure (depending on the structure of its operating system). This approach also offers a handy way of communicating back the fact that a particular protocol or piece thereof isn't available: define a TCC for "Unimplemented Generic Function". This feels a lot cleaner than having a close on socket 3 mean anything from "no FTP Server exists here" to "the FTP Server happens to be busy."

Notice that the UULP automatically provides the answer to such objections as the one Braden raises in RFC 430, that "there is no mechanism within the FTP for _changing_ a password. A user shouldn't have to use a different protocol ... to merely change his password". With the UULP, any system which has a password changing ability would have it available for all user level protocols because all of its abilities are made available by the generic login. This seems clearly superior to having to retrofit afterthought after afterthought to the various user level protocols as we come to realize that life is more convenient when we get away from the view that each protocol lives in its own island universe. I understand that one of the main motivations for going the multiple contact socket route was to avoid syntactic (and semantic) conflicts between the protocol and the particular Host's "normal" command processor; however, locking ourselves in to special command processors exclusively is awfully procrustean. So instead of cutting off the limbs to fit the bed, why not use the UULP to expand the bed.

Although this is a tentative proposal and not meant to be a detailed design spec, one elaboration suggests itself which might make the general idea more attractive: For ease of implementation on some systems, it would probably be a good idea to define additional TCC's for "Begin User Protocol". That is, the user side starts the FTP by sending the "Begin FTP" Telnet Control Code, waits for the Server to send either the same code or the one for "Unimplemented Generic Function", and then proceeds (or not) to send STOR's and RETR's and the like. (It could also follow the "I will"/"I won't" style discipline of RFC 435 if we like.) Probably each line is preceded by the Network Generic Function TCC so that systems which don't pass input off to some other process can still distinguish between input to the system command processor and input to the procedure(s) which perform(s) the protocol in question, although perhaps it would be preferable to have an "End Protocol" TCC.

Now, I'm the first to admit that what makes sense to me, on my system, may not make sense on somebody else's. But it does seem plausible to me that the unified user level protocol I've sketched here ought to be no harder to implement than the multiple contact socket (MCS) ICP is. And the advantages of the UULP over the MCS ICP in terms of ease of extension and (at least in my mind, if not in this paper) clarity make it seem worthwhile to consider further. So rather than try to refine it here, let me simply ask for comments both on the general notion and on the necessary iteration of the design from sketch to spec. (The Multics scenario in ICCC booklet shows how to get "mail" to me, for those who don't feel like RFCing or phoning.)