PROPOSED HOST-FRONT END PROTOCOL

Status Of This Memo

   The reader should be aware of several things in regard to what the
   present document is up to. First and foremost, IT IS A PROPOSAL FOR
   A STANDARD, NOT A STANDARD ITSELF. Next, it assumes that the
   separate document, RFC 928, which is an introduction to the present
   document, has been read before it is. Next, it should be understood
   that "final cut" over this version of the document has been exercised
   by the author of RFC 928, not by the primary author of the present
   document, so any readers bothered by style considerations should feel
   free to blame the former, who's used to it, rather than the latter,
   who may well be guiltless. (Editing at a distance finally become too
   hard to manage, so if I'm typing it myself I'm going to fiddle with
   it myself too, including, but not limited to, sticking my own section
   on the Conceptual Model in before Joel's words start, rather than
   leaving it in the Introduction. MAP)

   Finally, it should be noted that this is not a finished document.
   That is, the intent is eventually to supply appendices for all of the
   protocol offloadings, describing their uses of protocol idiosyncratic
   parameters and even their interpretations of the standard per-command
   parameters, but in order to get what we've got into circulation we
   haven't waited until all such appendices have been written up. (We
   do have notes on how to handle FTP, e.g., and UDP will be pretty
   straightforward, but getting them ready would have delayed things
   into still another calendar year, which would have been very annoying
   ... not to say embarrassing.) For that matter, it's not even a
   finished document with respect to what is here. Not only is it our
   stated intention to revise the protocol based upon implementation
   experience gained from volunteer test implementations, but it's also
   the case that it hasn't proven feasible to iron out all known
   wrinkles in what is being presented. For example, the response codes
   almost certainly need clarification and expansion, and at least one
   of us doesn't think mandatory initial parameters need control flags.
   However, to try too hard for polish would be to stay in subcommittee
   for the better part of forever, so what you see is what we've got,
   but certainly isn't meant to be what you or we are stuck with.

   This RFC suggests a proposed protocol for the ARPA-Internet
   community, and requests discussion and suggestions for improvements.
   Distribution of this memo is unlimited.

Conceptual Model

    There are two fundamental motivations for doing outboard processing.
    One is to conserve the Hosts' resources (CPU cycles and memory) in a
    resource sharing intercomputer network, by offloading as much of the
    required networking software from the Hosts to Outboard Processing
    Environments (or "Network Front-Ends") as possible. The other is to
    facilitate procurement of implementations of the various
    intercomputer networking protocols for the several types of Host in
    play in a typical heterogeneous intercomputer network, by employing
    common implementations in the OPE. A third motivation, of basing a
    network security approach on trusted mandatory OPEs, will not be
    dealt with here, but is at least worthy of mention.

    Neither motivation should be allowed to detract from the underlying,
    assumed desire to perform true intercomputer networking, however.
    Therefore, it is further assumed that OPEs will be attached to Hosts
    via a flexible attachment strategy, as described in [1]. That is, at
    the software level an explicit Host-Front End Protocol (H-FP) will be
    employed between Hosts and OPEs, rather than having OPEs emulate
    devices or device controllers already "known" to Host operating
    systems (in order to avoid introducing new code into the Host).

    For reasons discussed in the Introduction, an H-FP resolves into
    three layers. The Link layer enables the exchange of bits between
    Host and OPE. The Channel layer enables the bit streams to be
    demultiplexed and flow controlled (both the Channel and Link layers
    may use preexisting per-Host mechanizations, it should be recalled).
    The Command (or "Service Access") layer is our primary concern at
    present. It serves as the distributed processing mechanism which
    allows processes on Hosts to manipulate protocol interpreters (PIs)
    in OPEs on their behalf; for convenience, it will be referred to as
    "the H-FP" here. (It should be noted that the Link and Channel
    layers may be viewed as roughly equivalent to the inboard processing
    investment for a Host-comm subnet processor PI and device driver, so
    in practical terms the savings of resources achieved by outboard
    processing come from making the H-FP "smaller" than the inboard
    implementations of the protocols it allows to be offloaded.)

    The crucial property of the H-FP conceptually is that it stands as
    the interface between a (Host) process and a PI (which is actually
    outboard). Usually, the model is that of a closed subroutine
    interface, although in some cases an interprocess communication
    mechanism model must be appealed to. That is, the interactions
    between cooperating H-FP PIs in some sense mimic subroutine or IPC
    calls, from the perspective of Host processes calling upon their own
    H-FP PIs, which in turn are of course interfacing via just such

mechanisms themselves. Another way of putting it is that "if the
protocols were inboard," the processes invoking H-FP wouldn't know
the difference. H-FP, then, may be viewed as a roundabout way of
letting Host processes "get at" various PIs.

Naturally, the mechanization of the desired concept cannot be
particularly literal. After all, the Hosts and the OPEs are
different processors, so we're not envisioning a passing through of
parameters in an exact fashion. However, in broad terms the model is
just that of a somewhat funny interface between a process and a PI.
(This should not be construed as ruling out the occurrence of events
which prompt the OPE to initiate an exchange of commands with the
Host, though; see the Introduction for more on the topic of
"Symmetric Begins.")

Interaction Discipline

The interaction between the Host and the OPE must be capable of
providing a suitable interface between processes (or protocol
interpreters) in the Host and the off-loaded protocol interpreters in
the OPE. This interaction must not, however, burden the Host more
heavily than would have resulted from supporting the protocols
inboard, lest the advantage of using an OPE be overridden.
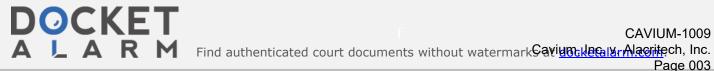
Channel Level Interaction

As stated elsewhere, the Channel level protocol (implicitly in
conjunction with the Link level) provides two major functions. These
are demultiplexing the traffic from the Link level into distinct data
streams, and providing flow control between the Host and the OPE on a
per stream basis. These hold even if the Host-OPE attachment is DMA.

The data streams between the Host and the OPE are bidirectional. In
this document, the basic unit of data transferred by the Channel
level is referred to as a "chunk". The primary motivation for this
terminology is that the H-FP permits the Channel level to be one of
several possible protocols, each with its own terminology. For
example, a chunk on an X.25 Channel would be a packet, while a chunk
on the DTI H-FP channel would be a message. While the Command level
is, in a sense, "more efficient" when the chunk size is permitted to
be large, the flexibility permitted in the choice of protocols at the
Channel level precludes any assumptions about the chunk size.

Each data stream is fully asynchronous. A Channel protocol user can
send data at any time, once the channel has been properly opened.
(The Command level's logic may render some actions meaningless,
however.) The data transfer service provided by the Channel protocol

is reliable; this entails delivery in the correct order, without
duplication, and checked for bit errors. All retransmission, error
checking, and duplicate detection is provided by this protocol in a
way that is transparent to the user. (If the attachment is DMA,
stream identification and chunk length must still be provided for.)

The flow control at the Channel level is provided to prevent the OPE
and the Host from overloading each other's resources by excessive
transmissions. In general, this flow control should not directly
affect the outboard protocol interpreters' operation. On the other
had, this flow control has the same effect as explicit interface
events that provide flow control between the user and the protocol
interpreter (e.g., the Allocate event of the interface specification
for TCP found in MIL-STD 1778). Hence, such events do not need to be
communicated explicitly at the Command level. (If the attachment is
DMA, flow control must still be provided for.)

Should Hosts require an OPE to be attached via a Link Level that
furnishes physical demultiplexing (e.g., a group of RS232 ports), any
attempt to avoid furnishing reliability and explicit flow control, is
done at their peril; we have not chosen to assist such an
enterprise, but neither have we precluded it. (It would certainly
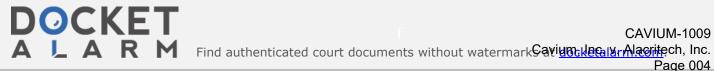violate the spirit of the thing, however.)

Command Level Interaction

The approach chosen for this H-FP is to base the interaction on a
small set of commands, separately applicable to a given Channel Level
channel. The commands are simple, but sufficiently flexible to permit
the off-loading of the interpreters of the large number of protocols
at various levels in the hierarchy. This flexibility is made
possible in part by the similar nature of the interfaces to most
protocols, combined with the provision of "protocol idiosyncratic
parameters". These parameters are defined for each offloaded protocol
interpreter in the OPE. The use of such parameters does not
complicate the basic design of the OPE, since it must be customized
for each off-loaded protocol anyway, and all that is required of the
OPE for those parameters is to pass them to the off-loaded protocol
interpreter. Hence, an interface tailored to a particular protocol
can be created in a straightforward and cost-effective way.

The command dialog is more or less asynchronous. Commands can be
issued at any particular time (except when there is a pending
command, which will be discussed below), and there is no need for
dummy traffic on a channel when no commands are issued.

Associated with each command is a response. The purpose of this

response is to indicate, at some level that depends in part on the
particular protocol interpreter that is offloaded to the OPE, whether
the command was successfully executed, and if unsuccessful, the
reason. Often, generating the response involves interaction with the
protocol interpreter before a response can be generated.

When a command is issued, the issuer must wait for a response before
another command is issued. The nature of the communication between
the Host and the OPE is thus a lock step command/response dialog.
There are two major exceptions to this principle, however. One
exception is the abrupt form of the End command, which can be issued
at any time to cancel any previously issued commands, and indicate
that services are no longer desired. The other exception is the
Signal command. Since a Signal is out-of-band and usually of high
importance, forcing it to wait on a response would be undesirable.
Hence, a Signal command can be issued while commands (other than
Signal) are pending. However, a Signal command should not be issued
before a successful response to the Begin command has been received.
Since it is possible for more than one command of different types to
be pending at one time, a mechanism to distinguish responses is
needed. Since there are never two commands of the same type pending,
including the command name in the response is sufficient to make this
distinction.

A special case command is the Transmit command. Details of the
Transmit command are provided in the next section. Essentially, the
Transmit command is used to invoke the data transfer services of the
off-loaded protocol (when issued by the Host) or to indicate the
arrival of new data from the network (when issued by the OPE). The
nature of specific protocol interfaces for these events varies widely
between protocols. Some may block until the data is accepted by the
remote counterpart (or "peer") protocol interpreter, while others may
not. Hence, there is a special parameter which indicates the nature
of the Transmit command interface. It can either require that the
response should be generated immediately after determining the
Transmit command is complete and formed properly, or can indicate
that the response should not be generated until the appropriate
interface event is given by the remote protocol interpreter. The
default action for all Transmit commands can be initialized using the
Begin command and changed using the Condition command. Also, the
default action can be temporarily overridden by specifying a
parameter with the Transmit command. The net result of this mechanism
is to allow the Host to determine within reason just how lock-stepped
transmissions are to be. (It is assumed that the usual case will be
to transfer the burden of buffering to the OPE by taking immediate
responses, provided that doing so "makes sense" with the particular
offloaded protocol in play.)

Lilienkamp & Mandell & Padlipsky   [Page 5]

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.