;;;;

A PERSPECTIVE ON THE ARPANET REFERENCE MODEL

M.A. PADLIPSKY
THE MITRE CORPORATION
Bedford, Massachusetts

Abstract

     The paper, by one of its developers, describes the
conceptual framework in which the ARPANET intercomputer
networking protocol suite, including the DoD standard
Transmission Control Protocol (TCP) and Internet Protocol (IP),
were designed. It also compares and contrasts several aspects of
the ARPANET Reference Model (ARM) with the more widely publicized
International Standards Organization's Reference Model for Open
System Interconnection (ISORM).

"A PERSPECTIVE ON THE ARPANET REFERENCE MODEL"

M. A. Padlipsky

Introduction

    Despite the fact that "the ARPANET" stands as the
proof-of-concept of intercomputer networking and, as discussed in
more detail below, introduced such fundamental notions as
Layering and Virtualizing to the literature, the wide
availability of material which appeals to the International
Standards Organization's Reference Model for Open System
Interconnection (ISORM) has prompted many new- comers to the
field to overlook the fact that, even though it was largely
tacit, the designers of the ARPANET protocol suite have had a
reference model of their own all the long. That is, since well
before ISO even took an interest in "networking", workers in the
ARPA-sponsored research community have been going about their
business of doing research and development in intercomputer
networking with a particular frame of reference in mind. They
have, unfortunately, either been so busy with their work or were
perhaps somehow unsuited temperamentally to do learned papers on
abstract topics when there are interesting things to be said on
specific topics, that it is only in very recent times that there
has been much awareness in the research community of the impact
of the ISORM on the lay mind. When the author is asked to review
solemn memoranda comparing such things as the ARPANET treatment
of "internetting" with that of CCITT employing the ISORM "as the
frame of reference," however, the time has clearly come to
attempt to enunciate the ARPANET Reference Model (ARM)
publicly--for such comparisons are painfully close to comparing
an orange with an apple using redness and smoothness as the
dominant criteria, given the philosophical closeness of the CCITT
and ISO models and their mutual disparities from the ARPANET
model.

    This paper, then, is primarily intended as a perspective on
the ARM. (Secondarily, it is intended to point out some of the
differences between the ARM and the ISORM. For a perspective on
this subtheme, please see Note [1]) It can't be "the official"
version because the ARPANET Network Working Group (NWG), which
was the collective source of the ARM, hasn't had an official
general meeting since October, 1971, and can scarcely be
resurrected to haggle over it. It does, at least, represent with
some degree of fidelity the views of a number of NWG members as
those views were expressed in NWG general meetings, NWG protocol
design committee meetings, and private conversations over the
intervening years. (Members of the current ARPA Internet Working
Group, which applied

1

and adapted the original model to a broader arena than had
initially been contemplated, were also consulted.) That might
not sound so impressive as a pronunciamento from an international
standards organization, but the reader should be somewhat
consoled by the consideration that not only are the views
expressed here purported to be those of the primary workers in
the field, but also at least one Englishman helped out in the
review process.

                    Historical/Philosophical Context

     Although rigorous historians of science might quibble as to
whether they were "invented" by a particular group, it is an
historical fact that many now widely-accepted, fundamental
concepts of intercomputer networking were original to the ARPANET
Network Working Group. [2] Before attempting to appreciate the
implications of that assertion, let's attempt to define its two
key terms and then cite the concepts it alludes to:

     By "intercomputer networking" we mean the attachment of
multiple, usually general-purpose computer systems--in the sense
of Operating Systems of potentially different manufacture (i.e.,
"Heterogeneous Operating Systems")--to some communications
network, or communications networks somehow interconnected, for
the purpose of achieving resource sharing amongst the
participating operating systems, usually called Hosts. (By
"resource sharing" we mean the potential ability for programs on
each of the Hosts to interoperate with programs on the other
Hosts and for data housed on each of the Hosts to be made
available to the other Hosts in a more general and flexible
fashion than merely enabling users on each of the Hosts to be
able to login to the other Hosts as if they were local; that is,
we expect to do more than mere "remote access" to intercomputer
networked Hosts.) By "the ARPANET Network Working Group," we
mean those system programmers and computer scientists from
numerous Defense Advanced Research Projects Agency-sponsored
installations whose home operating systems were intended to
become early Hosts on the ARPANET. (By "the ARPANET" we mean,
depending on context, either that communications network
sponsored by DARPA which served as proof-of-concept for the
communications technology known as "packet switching," or,
consistent with common usage, the intercomputer network which was
evolved by the NWG that uses that communications network--or
"comm subnet"--as its inter-Host data transmission medium.)

     The concepts of particular interest are as follows: By
analogy to the use of the term in traditional communications, the
NWG decided that the key to the mechanization of the
resource-sharing goal (which in turn had been posited in their
informal charter)

would be "protocols" that Hosts would interpret both in
communicating with the comm subnet and in communicating with each
other. Because the active entities in Hosts (the programs in
execution) were widely referred to in Computer Science as
"processes," it seemed clear that the mechanization of resource
sharing had to involve interprocess communication; protocols that
enabled and employed interprocess communication became, almost
axiomatically, the path to the goal. Perhaps because the
limitations of mere remote access were perceived early on, or
perhaps simply by analogy to the similar usage with regard to
distinguishing between physical tape drives and tape drives
associated with some conventionally-defined function like the
System Input stream or the System Output stream in batch
operating systems, the discernible communications paths (or
"channels") through the desired interprocess communication
mechanism became known as "logical connections"--the intent of
the term being to indicate that the physical path didn't matter
but the designator (number) of the logical connection could have
an assigned meaning, just like logical tape drive numbers.
Because "modularity" was an important issue in Computer Science
at the time, and because the separation of Hosts and Interface
Message Processors (IMP's) was a given, the NWG realized that the
protocols it designed should be "layered," in the sense that a
given set of related functions (e.g., the interprocess
communication mechanism, or "primitives," as realized in a
Host-to-Host protocol) should not take special cognizance of the
detailed internal mechanics of another set of related functions
(e.g., the comm subnet attachment mechanism, as realized in a
Host-Comm Subnet Processor protocol), and that, indeed, protocols
may be viewed as existing in a hierarchy.

     With the notion of achieving resource sharing via layered
protocols for interprocess communication over logical connections
fairly firmly in place, the NWG turned to how best to achieve the
first step of intercomputer networking: allowing a distant user
to login to a Host as if local--but with the clear understanding
that the mechanisms employed were to be generalizable to other
types of resource sharing. Here we come to the final fundamental
concept contributed by the NWG, for it was observed that if n
different types of Host (i.e., different operating systems) had
to be made aware of the physical characteristics of m different
types of terminal in order to exercise physical control over
them--or even if n different kinds of Host had to become aware of
the native terminals supported by m other kinds of Hosts if
physical control were to remain local--there would be an
administratively intractable "n x m problem." So the notion of
creating a "virtual terminal" arose, probably by analogy to
"virtual memory" in the sense of something that "wasn't really
there" but could be used as if it

were; that is, a common intermediate representation (CIR) of
terminal characteristics was defined in order to allow the Host
to which a terminal was physically attached to map the particular
characteristics of the terminal into a CIR, so that the Host
being logged into, knowing the CIR as part of the relevant
protocol, could map out of it into a form already acceptable to
the native operating system. And when it came time to develop a
File Transfer Protocol, the same virtualizing or CIR trick was
clearly just as useful as for a terminal oriented protocol, so
virtualizing became part of the axiom set too.

     The NWG, then, at least pioneered and probably invented the
notion of doing intercomputer networking/resource sharing via
hierarchical, layered protocols for interprocess communication
over logical connections of common intermediate representations/
virtualizations. Meanwhile, outside of the ARPA research
community, "the ARPANET" was perceived to be a major
technological advance. "Networking" became the "in" thing. And
along with popular success came the call for standards; in
particular, standards based on a widely-publicized "Reference
Model for Open System Interconnection" promulgated by the
International Standards Organization. Not too surprisingly, Open
System Interconnection looks a lot like resource sharing, the
ISORM posits a layered protocol hierarchy, "connections" occur
frequently, and emerging higher level protocols tend to
virtualize; after all, one expects standards to reflect the state
of the art in question. But even if the ISORM, suitably refined,
does prove to be the wave of the future, this author feels that
the ARM is by no means a whitecap, and deserves explication--both
in its role as the ISORM's "roots" and as the basis of a
still-viable alternative protocol suite.

Axiomatization

     Let's begin with the axioms of the ARPANET Reference Model.
Indeed, let's begin by recalling what an axiom is, in common
usage: a principle the truth of which is deemed self-evident.
Given that definition, it's not too surprising that axioms rarely
get stated or examined in non-mathematical discourse. It turns
out, however, that the axiomatization of the ARM--as best we can
recall and reconstruct it--is not only germane to the enunciation
of the ARM, but is also a source of instructive contrasts with
our view of the axiomatization of the ISORM. (See [1] again.)

Resource Sharing

     The fundamental axiom of the ARM is that intercomputer
networking protocols (as distinct from communications network

protocols) are to enable heterogeneous computer operating systems
("Hosts") to achieve resource sharing. Indeed, the session at
the 1970 SJCC in which the ARPANET entered the open literature
was entitled "Resource Sharing Computer Networks".

     Of course, as self-evident truths, axioms rarely receive
much scrutiny. Just what resource sharing is isn't easy to pin
down--nor, for that matter, is just what Open System
Interconnection is. But it must have something to do with the
ability of the programs and data of the several Hosts to be used
by and with programs and data on other of the Hosts in some sort
of cooperative fashion. It must, that is, confer more
functionality upon the human user than merely the ability to log
in/on to a Host miles away ("remote access").

     A striking property of this axiom is that it renders
protocol suites such as "X.25"/"X.28"/ "X.29" rather
uninteresting for our purposes, for they appear to have as their
fundamental axiom the ability to achieve remote access only. (It
might even be a valid rule of thumb that any "network" which
physically interfaces to Hosts via devices that resemble milking
machines--that is, which attach as if they were just a group of
locally-known types of terminals--isn't a resource sharing
network.)

     Reference [3] addresses the resource sharing vs. remote
access topic in more detail.

Interprocess Communication

     The second axiom of the ARM is that resource sharing will be
achieved via an interprocess communication mechanism of some
sort. Again, the concept isn't particularly well-defined in the
"networking" literature. Here, however, there's some
justification, for the concept is fairly well known in the
Operating Systems branch of the Computer Science literature,
which was the field most of the NWG members came from.
Unfortunately, because intercomputer networking involves
communications devices of several sorts, many whose primary field
is Communications became involved with "networking" but were not
in a position to appreciate the implications of the axiom.

     A process may be viewed as the active element of a Host, or
as an address space in execution, or as a "job", or as a "task",
or as a "control point"--or, actually, as any one (or more) of at
least 29 definitions from at least 28 reputable computer
scientists. What's important for present purposes isn't the
precise definition (even if there were one), but the fact that
the axiom's presence dictates the absence of at least one other
axiom at the same level of

abstraction. That is, we might have chosen to attempt to achieve
resource sharing through an explicitly interprocedure
communication oriented mechanism of some sort--wherein the
entities being enabled to communicate were subroutines, or pieces
of address spaces--but we didn't. Whether this was because
somebody realized that you could do interprocedure communication
(or achieve a "virtual address space" or "distributed operating
system" or some such formulation) on top of an interprocess
communication mechanism (IPC), or whether "it just seemed
obvious" to do IPC doesn't matter very much. What matters is
that the axiom was chosen, assumes a fair degree of familiarity
with Operating Systems, doesn't assume extremely close coupling
of Hosts, and has led to a working protocol suite which does
achieve resource sharing--and certainly does appear to be an
axiom the ISORM tacitly accepted, along with resource sharing.

Logical Connections

     The next axiom has to do with whether and how to demultiplex
IPC "channels", "routes", "paths", "ports", or "sockets". That
is, if you're doing interprocess communication (IPC), you still
have to decide whether a process can communicate with more than
one other process, and, if so, how to distinguish between the bit
streams. (Indeed, even choosing streams rather than blocks is a
decision.) Although it isn't treated particularly explicitly in
the literature, it seems clear that the ARM axiom is to do IPC
over logical connections, in the following sense: Just as batch
oriented operating systems found it useful to allow processes
(usually thought of as jobs--or even "programs") to be insulated
from the details of which particular physical tape drives were
working well enough at a particular moment to spin the System
Input and Output reels, and created the view that a reference to
a "logical tape number" would always get to the right physical
drive for the defined purpose, so too the ARM's IPC mechanism
creates logical connections between processes. That is, the IPC
addressing mechanism has semantics as well as syntax.

     "Socket" n on any participating Host will be defined as the
"Well-Known Socket" (W-KS) where a particular service (as
mechanized by a program which follows, or "interprets", a
particular protocol [4]) is found. (Note that the W-KS is
defined for the "side" of a connection where a given service
resides; the user side will, in order to be able to demultiplex
its network-using processes, of course assign different numbers
to its "sides" of connections to a given W-KS. Also, the serving
side takes cognizance of the using side's Host designation as
well as the proferred socket, so it too can demultiplex.)
Clearly, you want free sockets as well as Well-Known ones, and we
have them. Indeed, at each level of the ARM

hierarchy the addressing entities are divided into assigned and
unassigned sets, and the distinction has proven to be quite
useful to networking researchers in that it confers upon them the
ability to experiment with new functions without interfering with
running mechanisms.

     On this axiom, the ISORM differs from the ARM. ISORM
"peer-peer" connections (or "associations") appear to be used
only for demultiplexing, with the number assigned by the receive
side rather than the send side. That is, a separate protocol is
intro- duced to establish that a particular "transport"
connection will be used in the present "session" for some
particular service. At the risk of editorializing, logical
connections seem much cleaner than "virtual" connections (using
virtual in the sense of something that "isn't really there" but
can be used as if it were, by analogy to virtual memory, as noted
above, and in deference to the X.25 term "virtual circuit", which
appears to have dictated the receiver-assigned posture the ISORM
takes at its higher levels.) Although the ISORM view "works", the
W-KS approach avoids the introduction of an extra protocol.

Layering

     The next axiom is perhaps the best-known, and almost
certainly the worst-understood. As best we can reconstruct
things, the NWG was much taken with the Computer Science buzzword
of the times, "modularity". "Everybody knew" modularity was a
Good Thing. In addition, we were given a head start because the
IMP's weren't under our direct control anyway, but could possibly
change at some future date, and we didn't want to be "locked in"
to the then-current IMP-Host protocol. So it was enunciated that
protocols which were to be members of the ARM suite (ARMS, for
future reference, although at the time nobody used "ARM", much
less "ARMS") were to be layered. It was widely agreed that this
meant a given protocol's control information (i.e., the control
information exchanged by counterpart protocol interpreters, or
"peer entities" in ISORM terms) should be treated strictly as
data by a protocol "below" it, so that you could invoke a
protocol interpreter (PI) through a known interface, but if
either protocol changed there would not be any dependencies in
the other on the former details of the one, and as long as the
interface didn't change you wouldn't have to change the PI of the
protocol which hadn't changed.

     All well and good, if somewhat cryptic. The important point
for present purposes, however, isn't a seemingly-rigorous
definition of Layering, but an appreciation of what the axiom
meant in the evolution of the ARM. What it meant was that we
tried to come up

with protocols that represented reasonable "packagings" of
functionality. For reasons that are probably unknowable, but
about which some conjectures will be offered subsequently, the
ARM and the ISORM agree strongly on the presence of Layering in
their respective axiomatizations but differ strikingly as to what
packagings of functionality are considered appropriate. To
anticipate a bit, the ARM concerns itself with three layers and
only one of them is mandatorily traversed; whereas the ISORM,
again as everybody knows, has, because of emerging "sub-layers",
what must be viewed as at least seven layers, and many who have
studied it believe that all of the layers must be traversed on
each transmission/reception of data.

    Perhaps the most significant point of all about Layering is
that the most frequently-voiced charge at NWG protocol committee
design meetings was, "That violates Layering!" even though nobody
had an appreciably-clearer view of what Layering meant than has
been presented here, yet the ARMS exists. We can only guess what
goes on in the design meetings for protocols to become members of
the ISORM suite (ISORMS), but it doesn't seem likely that having
more layers could possibly decrease the number of arguments....

    Indeed, it's probably fair to say that the ARM view of
Layering is to treat layers as quite broad functional groupings
(Network Interface, Host-Host, and Process-Level, or
Applications), the constituents of which are to be modular.
E.g., in the Host-Host layer of the current ARMS, the Internet
Protocol, IP, packages internet addressing--among other
things--for both the Transmission Control Protocol, TCP, which
packages reliable interprocess communication, and UDP--the less
well-known User Datagram Protocol--which packages only
demultiplexable interprocess communication ... and for any other
IPC packaging which should prove desirable. The ISORM view, on
the other hand, fundamentally treats layers as rather narrow
functional groupings, attempting to force modularity by requiring
additional layers for additional functions (although the
"classes" view of the proposed ECMA-sponsored ISORM Transport
protocol tends to mimic the relations between TCP, UDP, and IP).

    It is, by the way, forcing this view of modularity by
multiplying layers rather than by trusting the designers of a
given protocol to make it usable by other protocols within its
own layer that we suspect to be a major cause of the divergence
between the ISORM and the ARM, but, as indicated, the issue
almost certainly is not susceptible of proof. (The less
structured view of modularity will be returned to in the next
major section.) At any rate, the notion that "N-entities" must
communicate with one another by means of "N-1 entities" does seem
to us to take the ISORM out of its

intended sphere of description into the realm of prescription,
where we believe it should not be, if for no other reason than
that for a reference model to serve a prescriptive role levies
unrealizable requirements of precision, and of familiarity with
all styles of operating systems, on its expositors. In other
words, as it is currently presented, the ISORM hierarchy of
protocols turns out to be a rather strict hierarchy, with
required, "chain of command" implications akin to the Elizabethan
World Picture's Great Chain of Being some readers might recall if
they've studied Shakespeare, whereas in the ARM a cat can even
invoke a king, much less look at one.

Common Intermediate Representations

     The next axiom to be considered might well not be an axiom
in a strict sense of the term, for it is susceptible of "proof"
in some sense. That is, when it came time to design the first
Process-Level (roughly equivalent to ISORM Level 5.3 [5] through
7) ARMS protocol, it did seem self-evident that a "virtual
terminal" was a sound conceptual model--but it can also be
demonstrated that it is. The argument, customarily shorthanded
as "the N X M Problem", was sketched above; it goes as follows:
If you want to let users at remote terminals log in/on to Hosts
(and you do--resource sharing doesn't preclude remote access, it
subsumes it), you have a problem with Hosts' native terminal
control software or "access methods", which only "know about"
certain kinds/brands/types of terminals, but there are many more
terminals out there than any Host has internalized (even those
whose operating systems take a generic view of I/O and don't
allow applications programs to "expect" particular terminals).

     You don't want to make N different types of Host/Operating
System have to become aware of M different types of terminal.
You don't want to limit access to users who are at one particular
type of terminal even if all your Hosts happen to have one in
common. Therefore, you define a common intermediate
representation (CIR) of the properties of terminals--or create a
Network Virtual Terminal (NVT), where "virtual" is used by
analogy to "virtual memory" in the sense of something that isn't
necessarily really present physically but can be used as if it
were. Each Host adds one terminal to its set of supported types,
the NVT--where adding means translating/mapping from the CIR to
something acceptable to the rest of the programs on your system
when receiving terminal-oriented traffic "from the net", and
translating/mapping to the CIR from whatever your acceptable
native representation was when sending terminal-oriented traffic
"to the net". (And the system to which the terminal is
physically attached does the same things.)

"Virtualizing" worked so well for the protocol in question
("Telnet", for TELetypewriter NETwork) that when it came time to
design a File Transfer Protocol (FTP), it was employed again--in
two ways, as it happens. (It also worked so well that in some
circles, "Telnet" is used as a generic term for "Virtual Terminal
Protocol", just like "Kleenex" for "disposable handkerchief".)
The second way in which FTP (another generic-specific) used
Common Intermediate Representations is well-known: you can make
your FTP protocol interpreters (PI's) use certain "virtual" file
types in ARMS FTP's and in proposed ISORMS FTP's. The first way
a CIR was used deserved more publicity, though: We decided to
have a command-oriented FTP, in the sense of making it possible
for users to cause files to be deleted from remote directories,
for example, as well as simply getting a file added to a remote
directory. (We also wanted to be able to designate some files to
be treated as input to the receiving Hosts' native "mail" system,
if it had one.) Therefore, we needed an agreed-upon
representation of the commands--not only spelling the names, but
also defining the character set, indicating the ends of lines,
and so on. In less time than it takes to write about, we
realized we already had such a CIR: "Telnet".

So we "used Telnet", or at any rate the NVT aspects of that
protocol, as the "Presentation" protocol for the control aspects
of FTP--but we didn't conclude from that that Telnet was a lower
layer than FTP. Rather, we applied the principles of modularity
to make use of a mechanism for more than one purpose--and we
didn't presume to know enough about the internals of everybody
else's Host to dictate how the program(s) that conferred the FTP
functionality interfaced with the program(s) that conferred the
Telnet functionality. That is, on some operating systems it
makes sense to let FTP get at the NVT CIR by means of closed
subroutine calls, on others through native IPC, and on still
others by open subroutine calls (in the sense of replicating the
code that does the NVT mapping within the FTP PI). Such
decisions are best left to the system programmers of the several
Hosts. Although the ISORM takes a similar view in principle, in
practice many ISORM advocates take the model prescriptively
rather than descriptively and construe it to require that PI's at
a given level must communicate with each other via an "N-1
entity" even within the same Host. (Still other ISORMites
construe the model as dictating "monolithic" layers--i.e., single
protocols per level--but this view seems to be abating.)

One other consideration about virtualizing bears mention:
it's a good servant but a bad master. That is, when you're
dealing with the amount of traffic that traverses a
terminal-oriented logical (or even virtual) connection, you don't
worry much about how many CPU cycles you're "wasting" on mapping
into and out of the NVT CIR; but

when you're dealing with files that can be millions of bits long,
you probably should worry--for those CPU cycles are in a fairly
real sense the resources you're making sharable. Therefore, when
it comes to (generic) FTP's, even though we've seen it in one or
two ISORM L6 proposals, having only a virtual file conceptual
model is not wise. You'd rather let one side or the other map
directly between native representations where possible, to
eliminate the overhead for going into and out of the CIR--for
long enough files, anyway, and provided one side or the other is
both willing and able to do the mapping to the intended
recipient's native representation.

Efficiency

     The last point leads nicely into an axiom that is rarely
acknowledged explicitly, but does belong in the ARM list of
axioms: Efficiency is a concern, in several ways. In the first
place, protocol mechanisms are meant to follow the design
principle of Parsimony, or Least Mechanism; witness the argument
immediately above about making FTP's be able to avoid the double
mapping of a Virtual File approach when they can. In the second
place, witness the argument further above about leaving
implementation decisions to implementers. In the author's
opinion, the worst mistake in the ISORM isn't defining seven (or
more) layers, but decreeing that "N-entities" must communicate
via "N-1 entities" in a fashion which supports the interpretation
that it applies intra-Host as well as inter-Host. If you picture
the ISORM as a highrise apartment building, you are constrained
to climb down the stairs and then back up to visit a neighbor
whose apartment is on your own floor. This might be good
exercise, but CPU's don't need aerobics as far as we know.

     Recalling that this paper is only secondarily about ARM
"vs." ISORM, let's duly note that in the ARM there is a concern
for efficiency from the perspective of participating Hosts'
resources (e.g., CPU cycles and, it shouldn't be overlooked,
"core") expended on interpreting protocols, and pass on to the
final axiom without digressing to one or two proposed specific
ISORM mechanisms which seem to be extremely inefficient.

Equity

     The least known of the ARM axioms has to do with a concern
over whether particular protocol mechanisms would entail undue
perturbation of native mechanisms if implemented in particular
Hosts. That is, however reluctantly, the ARMS designers were
willing to listen to claims that "you can't implement that in my
system" when particular tactics were proposed and, however

grudgingly, retreat from a mechanism that seemed perfectly
natural on their home systems to one which didn't seriously
discommode a colleague's home system. A tacit design principle
based on equity was employed. The classic example had to do with
"electronic mail", where a desire to avoid charging for incoming
mail led some FTP designers to think that the optionally
mandatory "login" commands of the protocol shouldn't be mandatory
after all. But the commands were needed by some operating
systems to actuate not only accounting mechanisms but
authentication mechanisms as well, and the process which
"fielded" FTP connections was too privileged (and too busy) to
contain the FTP PI as well. So (to make a complex story
cryptic), a common name and password were advertised for a "free"
account for incoming mail, and the login commands remained
mandatory (in the sense that any Host could require their
issuance before it participated in FTP).

     Rather than attempt to clarify the example, let's get to its
moral: The point is that how well protocol mechanisms integrate
with particular operating systems can be extremely subtle, so in
order to be equitable to participating systems, you must either
have your designers be sophisticated implementers or subject your
designs to review by sophisticated implementers (and grant veto
power to them in some sense).

     It is important to note that, in the author's view, the
ISORM not only does not reflect application of the Principle of
Equity, but it also fails to take any explicit cognizance of the
necessity of properly integrating its protocol interpreters into
continuing operating systems. Probably motivated by Equity
considerations, ARMS protocols, on the other hand, represent the
result of intense implementation discussion and testing.

Articulation

     Given the foregoing discussion of its axioms, and a reminder
that we find it impossible in light of the existence of dozens of
definitions of so fundamental a notion as "process" to believe in
rigorous definitions, the ARPANET Reference Model is not going to
require much space to articulate. Indeed, given further the
observation that we believe reference models are supposed to be
descriptive rather than prescriptive, the articulation of the ARM
can be almost terse.

     In order to achieve efficient, equitable resource sharing
among dissimilar operating systems, a layered set of interprocess
communication oriented protocols is posited which typically
employ common intermediate representations over logical
connections. Three

layers are distinguished, each of which may contain a number of protocols.

The Network Interface layer contains those protocols which are presented as interfaces by communications subnetwork processors ("CSNP"; e.g., packet switches, bus interface units, etc.) The CSNP's are assumed to have their own protocol or protocols among themselves, which are not directly germane to the model. In particular, no assumption is made that CSNP's of different types can be directly interfaced to one another; that is, "internetting" will be accomplished by Gateways, which are special purpose systems that attach to CSNP's as if they were Hosts (see also "Gateways" below). The most significant property of the Network Interface layer is that bits presented to it by an attached Host will probably be transported by the underlying CSNP's to an addressed Host (or Hosts) (i.e., "reliable" comm subnets are not posited--although they are, of course, allowed). A Network layer protocol interpreter ("module") is normally invoked by a Host-Host protocol PI, but may be invoked by a Process Level/Applications protocol PI, or even by a Host process interpreting no formal protocol whatsoever.

The Host-Host layer contains those protocols which confer interprocess communication functionality. In the current "internet" version of the ARM, the most significant property of such protocols is the ability to direct such IPC to processes on Hosts attached to "proximate networks" (i.e., to CSNP's of various autonomous communications subnetworks) other than that of the Host at hand, in addition to those on a given proximate net. (You can, by the way, get into some marvelous technicoaesthetic arguments over whether there should be a separate Internet layer; for present purposes, we assume that the Principle of Parsimony dominates.) Another significant property of Host-Host protocols, although not a required one, is the ability to do such IPC over logical connections. Reliability, flow control, and the ability to deal with "out-of-band signals" are other properties of Host-Host protocols which may be present. (See also "TCP/IP Design Goals and Constraints", below.) A Host-Host PI is normally invoked by a Process Level/Applications PI, but may also be invoked by a Host process interpreting no formal protocol whatsoever. Also, a Host need not support more than a single, possibly notional, process (that is, the code running in an "intelligent terminal" might not be viewed by its user--or even its creator--as a formal "process", but it stands as a de facto one).

The Process Level/Applications layer contains those protocols which perform specific resource sharing and remote access functions such as allowing users to log in/on to foreign Hosts, transferring files, exchanging messages, and the like. Protocols in this layer

will often employ common intermediate representations, or
"virtual- izations", to perform their functions, but this is not
a necessary condition. They are also at liberty to use the
functions performed by other protocols within the same layer,
invoked in whatever fashion is appropriate within a given
operating system context.

    Orthogonal to the layering, but consistent with it, is the
notion that a "Host-Front End" protocol (H-FP), or "Host-Outboard
Processing Environment" protocol, may be employed to offload
Network and Host-Host layer PI's from Hosts, to Outboard
Processing Environments (e.g., to "Network Front Ends", or to
BIU's, where the actual PI's reside, to be invoked by the H-FP as
a distributed processing mechanism), as well as portions of
Process Level/Applications protocols' functionality. The most
significant property of an H-FP attached Host is that it be
functionally identical to a Host with inboard PI's in operation,
when viewed from another Host. (That is, Hosts which outboard
PI's will be attached to in a flexible fashion via an explicit
protocol, rather than in a rigid fashion via the emulation of
devices already known to the operating system in question.)

    Whether inboard or outboard of the Host, it is explicitly
assumed that PI's will be appropriately integrated into the
containing operating systems. The Network and Host-Host layers
are, that is, effectively system programs (although this
observation should not be construed as implying that any of their
PI's must of necessity be implemented in a particular operating
system's "hard-core supervisor" or equivalent) and their PI's
must be able to behave as such.

                         Visualization

    Figures 1 and 2 (adapted from [6]) present, respectively, an
abstract rendition of the ARPANET Reference Model and a
particular version of a protocol suite designed to that model.
Just as one learns in Geometry that one cannot "prove" anything
from the figures in the text, they are intended only to
supplement the prose description above. (At least they bear no
resemblance to highrise apartment houses.)

                 TCP/IP Design Goals and Constraints

    The foregoing description of the ARM, in the interests of
conciseness, deferred detailed discussion of two rather relevant
topics: just what TCP and IP (the Transmission Control Protocol
and the Internet Protocol) are "about", and just what role
Gateways are

                              14

expected to play in the model. We turn to those topics now,
under separate headings.

     As has been stated, with the success of the ARPANET [7] as
both a proof-of-concept of intercomputer resource sharing via a
packet-switched communications subnetwork and a (still)
functional resource sharing network, a number of other bodies,
research and commercial, developed "their own networks." Often
just the communications subnetwork was intended, with the goal
being to achieve remote access to attached Hosts rather than
resource sharing among them, but nonetheless new networks
abounded. Hosts attached to the original ARPANET or to DoD nets
meant to be transferences of ARPANET technology should, it was
perceived in the research community, be able to do resource
sharing (i.e., interpret common high level protocols) with Hosts
attached to these other networks. Thus, the first discernible
goal of what was to become TCP/IP was to develop a protocol to
achieve "internetting".

     At roughly the same time--actually probably chronologically
prior, but not logically prior--the research community came to
understand that the original ARPANET Host-Host Protocol or AH-HP
(often miscalled NCP because it was the most visible component of
the Network Control Program of the early literature) was somewhat
flawed, particularly in the area of "robustness." The comm
subnet was not only relied upon to deliver messages accurately
and in order, but it was even expected to manage the transfer of
bits from Hosts to and from its nodal processors over a hardware
interface and "link protocol" that did no error checking. So,
although the ARPANET-as-subnet has proven to be quite good in
managing those sorts of things, surely if internetting were to be
achieved over subnets potentially much less robust than the
ARPANET subnet, the second discernible goal must be the
reliability of the Host-to-Host protocol. That is, irrespective
of the properties of the communications subnetworks involved in
internetting, TCP is to furnish its users--whether they be
processes interpreting formal protocols or simply processes
communicating in an ad hoc fashion--with the ability to
communicate as if their respective containing Hosts were attached
to the best comm subnet possible (e.g., a hardwired connection).

     The mechanizations considered to achieve reliability and
even those for internetting were alien enough to AH-HP's style,
though, and the efficiency of several of AH-HP's native
mechanisms (particularly Flow Control and the notion of a Control
Link) had been questioned often enough, that a good Host-Host
protocol could not be a simple extension of AH-HP. Thus, along
with the desire for reliability came a necessity to furnish a
good Host-Host protocol, a

design goal easy to overlook. This is a rather subtle issue in
that it brings into play a wealth of prior art. For present
purposes, in practical terms it means that the "good" ideas
(according to the technical intuition of the designers) of
AH-HP--such as sockets, logical connections, Well-Known Sockets,
and in general the interprocess communication premise--are
retained in TCP without much discussion, while the "bad" ideas
are equally tacitly jettisoned in favor of ones deemed either
more appropriate in their own right or more consistent with the
other two goals.

It could be argued that other goals are discernible, but the
three cited--which may be restated and compressed as a desire to
offer a good Host-Host protocol to achieve reliable
internetting--are challenging enough, when thought about hard for
a few years, to justify a document of even more than this one's
length. What of the implied and/or accepted design constraints,
though?

The first discernible design constraint borders on the
obvious: Just as the original ARPANET popularized
packet-switching (and, unfortunately to a lesser extent, resource
sharing), its literature popularized the notion of "Layering."
Mechanistically, layering is easy to describe: the control
information of a given protocol must be treated strictly as data
by the next "lower" protocol (with processes "at the top," and
the/a transmission medium "at the bottom"), as discussed earlier.
Philosophically, the notion is sufficiently subtle that even
today researchers of good will still argue over what "proper"
layering implies, also as discussed earlier. For present
purposes, however, it suffices to observe the following:
Layering is a useful concept. The precise set of functions
offered by a given layer is open to debate, as is the precise
number of layers necessary for a complete protocol suite to
achieve resource sharing. (Most researchers from the ARPANET
"world" tend to think of only three layers--the process,
applications, or user level; the Host-Host level; and the network
level--though if pressed they acknowledge that "the IMPs must
have a protocol too." Adherents of the International Standards
Organization's "Open System Interconnection" program--which
appears to be how they spell resource sharing--claim that seven
is the right number of levels--though if pressed they acknowledge
that "one or two of them have sublevels." And adherents of the
Consultative Committee for International Telephony and Telegraphy
don't seem particularly concerned with resource sharing to begin
with.) At any rate, TCP and IP are constrained to operate in a
(or possibly in more than one) layered protocol hierarchy.
Indeed, although it is not the sole reason, this fact is the
primary rationale for separating the internetting mechanization
into a discrete protocol (the Internet Protocol: IP). In other
words, although designed

16

"for" the ARM, TCP and IP are actually so layered as to be useful
even outside the ARM.

     It should be noted that as a direct consequence of the
Layering constraint, TCP must be capable of operating "above" a
functionally- equivalent protocol other than IP (e.g., an
interface protocol directly into a proximate comm subnet, if
internetting is not being done), and IP must be capable of
supporting user protocols other than TCP (e.g., a non-reliable
"Real-Time" protocol).

     Resisting the temptation to attempt to do justice to the
complexities of Layering, we move on to a second design
constraint, which also borders on the obvious: Only minimal
assumptions can be made about the properties of the various
communications subnetworks in play. (The "network" composed of
the concatenation of such subnets is sometimes called "a
catenet," though more often--and less picturesquely--merely "an
internet.") After all, the main goal is to let processes on
Hosts attached to, essentially, "any old (or new) net"
communicate, and to limit that communication to processes on
Hosts attached to comm subnets that, say, do positive
acknowledgments of message delivery would be remiss. [8]

     Given this constraint, by the way, it is quite natural to
see the more clearly Host-to-Host functions vested in TCP and the
more clearly Host-to-catenet functions vested in IP. It is,
however, a misconception to believe that IP was designed in the
expectation that comm subnets "should" present only the "lowest
common denominator" of functionality; rather, IP furnishes TCP
with what amounts to an abstraction (some would say a
virtualization--in the ARPANET Telnet Protocol sense of
virtualizing as meaning mapping from/to a common intermediate
representation to/from a given native representation) of the
properties of "any" comm subnet including, it should be noted,
even one which presents an X.25 interface. That is, IP allows
for the application to a given transmission of whatever generic
properties its proximate subnet offers equivalents for; its
design neither depends upon nor ignores the presence of any
property other than the ability to try to get some packet of bits
to some destination, which surely is an irreducible minimum for
the functionality of anything one would be willing to call a
network.

     Finally, we take note of a design constraint rarely
enunciated in the literature, but still a potent factor in the
design process: Probably again stemming from the popularity of
the original ARPANET, as manifested in the number of types of
Hosts (i.e., operating systems) attached to it, minimal
assumptions are made about the nature or even the "power" of the
Hosts which could implement TCP/IP. Clearly, some notion of
process is necessary if there is to

be interprocess communication, but even here the entire Host
might constitute a single process from the perspective of the
catenet. Less clearly, but rather importantly, Hosts must either
"be able to tell time" or at least be able to "fake" that
ability; this is in order to achieve the reliability goal, which
leads to a necessity for Hosts to retransmit messages (which may
have gotten lost or damaged in the catenet), which in turn leads
to a necessity to know when to retransmit. It should be noted,
however, that this does not preclude a (presumably quite modestly
endowed) Host's simply going into a controlled loop between
transmissions and retransmitting after enough megapasses through
the loop have been made--if, of course, the acknowledgment of
receipt of the transmission in question has not already arrived
"in the meantime."

     To conclude with a formulation somewhere between the concise
and the terse, TCP/IP are to constitute a means for processes on
Hosts about which minimal assumptions are made to do reliable
interprocess communication in a layered protocol suite over a
catenet consisting of communications subnetworks about which
minimal assumptions are made. Though it nearly goes without
saying, we would probably be remiss not to conclude by observing
that that's a lot harder to do than to say.

                              Gateways

     One other aspect of the ARPANET Reference Model bears
separate mention. Even though it is an exceedingly fine point as
to whether it's actually "part" of the Model or merely a sine qua
non contextual assumption, the role of Gateways is of
considerable importance to the functioning of the Internet
Protocol, IP.

     As noted, the defining characteristic of a Gateway is that
it attaches to two or more proximate comm subnets as if it were a
Host. That is, from "the network's" point of view, Gateways are
not distinguished from Hosts; rather, "normal" traffic will go to
them, addressed according to the proximate net's interface
protocol. However, the most important property of Gateways is
that they interpret a full version of IP which deals with
internet routing (Host IP interpreters are permitted to take a
static view of routing, sending datagrams which are destined for
Hosts not directly attached to the proximate net to a known
Gateway, or Gateways, addressed on the proximate net), as well of
course, as with fragmentation of datagrams which, although of
permissible size on one of their proximate nets, are too large
for the next proximate net (which contains either the target Host
or still another Gateway).

     Aside from their role in routing, another property of
Gateways is also of significance: Gateways do not deal with
protocols above IP. That is, it is an explicit assumption of the
ARM that the catenet will be "protocol compatible", in the sense
that no attempt will be made to translate or map between
dissimilar Host-Host protocols (e.g., TCP and AH-HP) or
dissimilar Process-level protocols (e.g., ARPANET FTP and EDN
FTP) at the Gateways. The justifications for this position are
somewhat complex; the interested reader is encouraged to see
Reference [10]. For present purposes, however, it should suffice
to note that the case against translating/mapping Gateways is a
sound one, and that, as with the ARMS protocols, the great
practical virtue of what are sometimes called "IP Gateways" is
that they are in place and running.

                    "Architectural" Highlights

     As was implied earlier, one of the problems with viewing a
reference model prescriptively rather than descriptively is that
the articulation of the model must be more precise than appears
to be humanly possible. That the ISORM, in striving for
superhuman precision, fails to achieve it is not grounds for
censure. However, by reaching a degree of apparent precision
that has enticed at least some of its readers to attempt to use
it in a prescriptive fashion, the ISORM has introduced a number
of ambiguities which have been attributed as well to the ARM by
relative laymen in intercomputer networking whose initial
exposure to the field was the ISORM. Therefore, we conclude this
not-very-rigorous paper with a highly informal treatment of
various points of confusion stemming from attempting to apply the
ISORM to the ARM.

     (It should be noted, by the way, that one of the most
striking ambiguities about the ISORM is just what role X.25 plays
in it: We have been informed by a few ISORMites that X.25 "is"
Levels 1-3, and we accepted that as factual until we were told
during the review process of the present paper that "that's not
what we believe in the U.K." What follows, then, is predicated
on the assumption that the earlier reports were probably but not
definitely accurate--and if it turns out to be in time to help
prevent ISO from embracing X.25 exclusively by pointing out some
of the problems entailed, so much the better.)

"Customized Parking Garages"

     The typical picture of the ISORM shows what looks like two
highrises with what looks like two parking garages between them.
(That is, seven layers of protocol per "Data Terminal Equipment",
three layers per "Data Circuit Terminating Equipment".) The
problem

is that only one "style" of parking garage--i.e., one which
presents an X.25 interface--is commonly understood to be
available to stand beside an ISORM DTE by those who believe that
ISO has adopted X.25 as its L1-3. In the ARM, on the other hand,
no constraints are levied on the Communications Subnetwork
Processors. Thus, satellite communications, "Packet Radios",
"Ethernets" and the like are all accommodated by the ARM.

     Also, the sort of Outboard Processing Environment mentioned
earlier in which networking protocols are interpreted on behalf
of the Host in a distributed processing fashion is quite
comfortably accommodated by the ARM. This is not to say that one
couldn't develop an OPE for/to the ISORM, but rather that doing
so does not appear to us to be natural to it, for at least two
reasons: 1. The Session Level associates sockets with processes,
hence it belongs "inboard". The Presentation Level involves
considerable bit-diddling, hence it belongs "outboard". The
Presentation Level is, unfortunately, above the Session Level.
This seems to indicate that outboard processing wasn't taken into
account by the formulators of the ISORM. 2. Although some
ISORMites have claimed that "X.25 can be used as a Host-Front End
Protocol", it doesn't look like one to us, even if the ability to
do end-to-end things via what is nominally the Network interface
is somewhat suggestive. (Those who believe that you need a
protocol as strong as TCP below X.25 to support the virtual
circuit illusion might argue that you've actually outboarded the
Host-Host layer, but both the X.25 spec and the ISORM appeal to
protocols above X.25 for full L II functionality.) Perhaps, with
sufficient ingenuity, one might use X.25 to convey an H-FP, but
it seems clear it isn't meant to be one in and of itself.

"Plenty of Roads"

     Based upon several pictures presented at conferences and in
articles, DCE's in the X.25-based ISORM appear to many to be
required to present X.25 interfaces to each other as well as to
their DTE's. Metaphorically, the parking garages have single
bridges between them. In the ARM, the CSNP-CSNP protocol is
explicitly outside the model, thus there can be as many "roads"
as needed between the ARM equivalent to ISORM parking garages.
This also allays fears about the ability to take advantage of
alternate routing in X.25 subnets or in X.75 internets (because
both X.25 and X.75 are "hop-by-hop" oriented, and would not seem
to allow for alternate routing without revision).

"Multiple Apartments Per Floor"

     As noted, the ISORM's strictures on inter-entity
communication within each "highrise" are equivalent to having to
climb downstairs and then back up to visit another apartment on
your own floor. The ARM explicitly expects PI's within a layer
to interface directly with one another when appropriate,
metaphorically giving the effect of multiple apartments on each
floor off a common hallway. (Also, for those who believe the
ISORM implies only one protocol/apartment per layer/story, again
the ARM is more flexible.)

"Elevators"

     The ISORM is widely construed as requiring each layer to be
traversed on every transmission (although there are rumors of the
forthcoming introduction of "null layers"), giving the effect of
having to climb all seven stories' worth of stairs every time you
enter the highrise. In the ARM, only Layer I, the Network
Interface layer, must be traversed; protocols in Layers II and/or
III need not come into play, giving the effect of being able to
take an elevator rather than climb the stairs.

"Straight Clotheslines"

     Because they appear to have to go down to L3 for their
initiation, the ISORM's Session and Transport connections are, to
us, metaphorically tangled clotheslines; the ARM's logical
connections are straight (and go from the second floor to the
second floor without needing a pole that gets in the way of the
folks on the third floor--if that doesn't make a weak metaphor
totally feeble.)

"Townhouse Styles Available"

     Should ISORM Level 6 and 7 protocols eventuate which are
desirable, the "two-story townhouse style apartments" they
represent can be erected on an ARM L I - L II (Network Interface
and Host-Host Layers) "foundation". With some clever carpentry,
even ISORM L5 might be cobbled in.

"Manned Customs Sheds"

     Although it's straining the architectural metaphor quite
hard, one of the unfortunate implications of the ISORM's failure
to address operating system integration issues is that the notion
of "Expedited Data" exchanges between "peer entities" might only
amount to an SST flight to a foreign land where there's no one on
duty at

the Customs Shed (and the door to the rest of the airport is
locked from the other side). By clearly designating the
Host-Host (L II) mechanism(s) which are to be used by Layer III
(Process-Level/ Applications) protocols to convey "out-of-band
signals", the ARM gives the effect of keeping the Customs Sheds
manned at all times. (It should be noted, by the way, that we
acknowledge the difficulty of addressing system integration
issues without biasing the discussion toward particular systems;
we feel, however, that not trying to do so is far worse than
trying and failing to avoid all parochialism.)

"Ready For Immediate Occupancy"

     The ARM protocol suite has been implemented on a number of
different operating systems. The ISORM protocol suite
"officially" offers at most (and not in the U.K., it should be
recalled) only the highly constraining functionality of X.25 as
L1-L3; L4-L7 are still in the design and agreement processes,
after which they must presumably be subjected to stringent
checkout in multiple implementations before becoming useful
standards. The metaphorical highrises, then, are years away from
being fit for occupancy, even if one is willing to accept the
taste of the interior decorators who seem to insist on building
in numerous features of dubious utility and making you take fully
furnished apartments whether you like it or not; the ARM
buildings, on the other hand, offer stoves and refrigerators, but
there's plenty of room for your own furniture-- and they're ready
for immediate occupancy.

                              Conclusion

     The architectural metaphor might have been overly extended
as it was, but it could have been drawn out even further to point
up more issues on which the ARM appears to us to be superior to
the ISORM, if our primary concern were which is "better". In
fairness, the one issue it omitted which many would take to be in
the ISORM's favor is that "vendor support" of interpreters of the
ISORM protocols will eventually amount to a desirable
"prefabrication", while the building of the ARM PI's is believed
to be labor-intensive. That would indeed be a good point, if it
were well-founded. Unfortunately for its proponents, however,
close scrutiny of the vendor support idea suggests that it is
largely illusory (vide [11]), especially in light of the amount
of time it will take for the international standardization
process to run its course, and the likelihood that specification
ambiguities and optional features will handicap interoperability.
Rather than extend the present paper even further, then, it seems
fair to conclude that with the possible exception of "vendor
support" (with which exception we take

                                 22

exception, for it should be noted that a number of vendors are
already offering support for TCP/IP), the ARPANET Reference Model
and the protocols designed in conformance with it are at least
worthy of consideration by anybody who's planning to do real
inter- computer networking in the next several years--especially
if they have operating systems with counterparts on the present
ARPANET, so that most if not all of the labor intensive part has
been taken care of already--irrespective of one's views on how
good the ISORM protocols eventually will be.

Acknowledgments

        Although it has seldom been more germane to observe that
"any remaining shortcomings are the author's responsibility",
this paper has benefited tremendously from the close scrutiny and
constructive comments of several distinguished members of both
the research community and the (DoD) Protocol Standards Technical
Panel. The author is not only extremely grateful to, but is also
extremely pleased to acknowledge his indebtedness to the
following individuals (cited in alphabetical order): Mr. Trevor
Benjamin, Royal Signals and Radar Establishment (U.K.); Mr.
Edward Cain, Chairman of the PSTP; Dr. Vinton Cerf, DARPA/IPTO
(at the time this was written); Dr. David Clark, M.I.T.
Laboratory for Computer Science (formerly Project MAC); and Dr.
Jonathan Postel, U.S.C. Information Sciences Institute.
Posterity may or may not thank them for their role in turning an
act of personal catharsis into a fair semblance of a "real"
paper, but the author emphatically does.

Notes and References

[1] It almost goes without saying that the subtheme is certainly
     not intended to be a definitive statement of the relative
     merits of the two approaches, although, as will be seen, the
     ARM comes out ahead, in our view. But then, the reader
     might well say, what else should I expect from a paper
     written by one of the developers of the ARM? To attempt to
     dispel thoughts of prejudgment, the author would observe
     that although he is indeed an Old Network Boy of the
     ARPANET, he was not a member of the TCP/IP (the keystone of
     the current ARM) design team, and that he began looking into
     ARM "vs." ISORM from the position of "a plague on both your
     houses". That he has concluded that the differences between
     TCP/IP-based ARM intercomputer networking and X.25-based
     ISORM intercomputer networking are like day and night may be
     taken as indicative of something, but that he also holds
     that the day is at least partly cloudy and the night is not
     altogether moonless should at least meliorate fears of
     prejudice. That is, of course the

ISORM has its merits and the ARM its demerits neither of
which are dealt with here. But "A Perspective" really means
"My Perspective", and the author really is more concerned in
this context with exposition of the ARM than with twitting
the ISORM, even if he couldn't resist including the
comparisons subtheme because of the one-sidedness of the
ISORM publicity he has perceived of late.

[2] Source material for this section was primarily drawn from
     the author's personal experience as a member the NWG and
     from numerous conversations with Dr. Jonathan B. Postel,
     long-time Chairman of the NWG and participant in the design
     meetings prior to the author's involvement. (See also
     Acknowledgments.)

[3] Padlipsky, M. A. "The Elements of Networking Style", M81-41,
     The MITRE Corporation, Bedford, MA, October 1981

[4] Yes, the notion of using "protocols" might well count as an
     axiom in its own right, but, no, we're not going to pretend
     to be that rigorous.

[5] That is, about three tenths of the possible span of
     "Session" functionality, which has to do with making up for
     the lack of Well-Known Sockets, isn't subsumed by the ARM
     Process-Level protocols, but the rest is, or could be.

[6] Davidson, J., et al., "The ARPANET Telnet Protocol: Its
     Purpose, Principles, Implementation, and Impact on Host
     Operating System Design," Proc Fifth Data Communications
     Symposium, ACM/IEEE, Snowbird, Utah, September, 1977.

[7] See Proceedings of the 1970 SJCC, "Resource Sharing Computer
     Networks" session, and Proceedings of the 1972 SJCC, "The
     ARPA Network" session for the standard open literature
     references to the early ARPANET. Other source material for
     this chapter is drawn from the author's personal
     conversations with TCP/IP's principal developers; see also
     Acknowledgments.

[8] A strong case can be made for desiring that the comm subnets
     make a "datagram" (or "connectionless") mode of interface
     available, based upon the desire to support such
     functionality as Packetized Speech, broadcast addressing,
     and mobile subscribers, among other things. For a more
     complete description of this point of view, see [9]. For
     present

purposes, we do not cite the presentation of a datagram mode
interface as a design constraint because it is
possible--albeit undesirable--to operate IP "on top of" a
comm subnet which does not present such an interface.

[9] Cerf, V. G. and R. E. Lyons, "Military Requirements for
     Packet-Switched Networks and for Their Protocol
     Standardization" Proc EASCON 1982.

[10] Padlipsky, M. A., "Gateways, Architectures and Heffalumps",
     M82-51, The MITRE Corporation, Bedford, MA, September 1982.

[11] ---------- "The Illusion of Vendor Support", M82-49, The
     MITRE Corporation, Bedford, MA, September 1982.

NOTE: Figure 1: ARM in the Abstract, and Figure 2: ARMS,
Somewhat Particularized, may be obtained by writing to: Mike
Padlipsky, MITRE Corporation, P.O. Box 208, Bedford,
Massachusetts 01730, or sending computer mail to
Padlipsky@USC-ISIA.