The Hitchhikers Guide to the Internet

25 August 1987

Ed Krol
krol@uxc.cso.uiuc.edu

This document was produced through funding of the National Science Foundation.

Purpose and Audience

This document assumes that one is familiar with the workings of a non-connected simple IP network (e.g. a few 4.2 BSD systems on an Ethernet not connected to anywhere else). Appendix A contains remedial information to get one to this point. Its purpose is to get that person, familiar with a simple net, versed in the ‘‘oral tradition’’ of the Internet to the point that that net can be connected to the Internet with little danger to either. It is not a tutorial, it consists of pointers to other places, literature, and hints which are not normally documented. Since the Internet is a dynamic environment, changes to this document will be made regularly. The author welcomes comments and suggestions. This is especially true of terms for the glossary (definitions are not necessary).

What is the Internet?

In the beginning there was the ARPAnet, a wide area experimental network connecting hosts and terminal servers together. Procedures were set up to regulate the allocation of addresses and to create voluntary standards for the network. As local area networks became more pervasive, many hosts became gateways to local networks. A network layer to allow the interoperation of these networks was developed and called IP (Internet Protocol). Over time other groups created long haul IP based networks (NASA, NSF, states...). These nets, too, inter-operate because of IP. The collection of all of these interoperating networks is the Internet.

Two groups do much of the research and information work of the Internet (ISI and SRI). ISI (the Informational Sciences Institute) does much of the research, standardization, and allocation work of the Internet. SRI International provides information services for the Internet. In fact, after you are connected to the Internet most of the information in this document can be retrieved from the Network Information Center (NIC) run by SRI.

Operating the Internet

Each network, be it the ARPAnet, NSFnet or a regional network, has its own operations center. The ARPAnet is run by BBN, Inc. under contract from DARPA. Their facility is called the Network Operations Center or NOC. Cornell University temporarily operates NSFnet (called the Network Information Service Center, NISC). It goes on to the regionals having similar facilities to monitor and keep watch over the goings on of their portion of the Internet. In addition, they all should have some knowledge of what is happening to the Internet in total. If a problem comes up, it is suggested that a campus network liaison should contact the network operator to which he is directly connected. That is, if you are connected to a regional network (which is gatewayed to the NSFnet, which is connected to the ARPAnet...) and have a problem, you should contact your regional network operations center.

RFCs

The internal workings of the Internet are defined by a set of documents called RFCs (Request for Comments). The general process for creating an RFC is for someone wanting something formalized to write a document describing the issue and mailing it to Jon Postel (postel@isi.edu). He acts as a referee for the proposal. It is then commented upon by all those wishing to take part in the discussion (electronically of course). It may go through multiple revisions. Should it be generally accepted as a good idea, it will be assigned a number and filed with the RFCs.

The RFCs can be divided into five groups: required, suggested, directional, informational and obsolete. Required RFC’s (e.g. RFC-791, The Internet Protocol) must be implemented on any host connected to the Internet. Suggested RFCs are generally implemented by network hosts. Lack of them does not preclude access to the Internet, but may impact its usability. RFC-793 (Transmission Control Protocol) is a suggested RFC. Directional RFCs were discussed and agreed to, but their application has never come into wide use. This may be due to the lack of wide need for the specific application (RFC-937 The Post Office Protocol) or that, although technically superior, ran against other pervasive approaches (RFC-891 Hello). It is suggested that should the facility be required by a particular site, an implementation be done in accordance with the RFC. This insures that, should the idea be one whose time has come, the implementation will be in accordance with some standard and will be generally usable. Informational RFCs contain factual information about the Internet and its operation (RFC-990, Assigned Numbers). Finally, as the Internet and technology have grown, some RFCs have become unnecessary. These obsolete RFCs cannot be ignored, however. Frequently when a change is made to some RFC that causes a new one to be issued obsoleting others, the new RFC only contains explanations and motivations for the change. Understanding the model on which the whole facility is based may involve reading the original and subsequent RFCs on the topic. (Appendix B contains a list of what are considered to be the major RFCs necessary for understanding the Internet).

The Network Information Center

The NIC is a facility available to all Internet users which provides information to the community. There are three means of NIC contact: network, telephone, and mail. The network accesses are the most prevalent. Interactive access is frequently used to do queries of NIC service overviews, look up user and host names, and scan lists of NIC documents. It is available by using

```
%telnet sri-nic.arpa
```

on a BSD system and following the directions provided by a user friendly prompter. From poking around in the databases provided one might decide that a document named NETINFO:NUG.DOC (The Users Guide to the ARPAnet) would be worth having. It could be retrieved via an anonymous FTP. An anonymous FTP would proceed something like the following. (The dialogue may vary slightly depending on the implementation of FTP you are using).

```
%ftp sri-nic.arpa
Connected to sri-nic.arpa.
220 SRI_NIC.ARPA FTP Server Process 5Z(47)-6 at Wed 17-Jun-87 12:00 PDT
Name (sri-nic.arpa:myname): anonymous
331 ANONYMOUS user ok, send real ident as password.
Password: myname
230 User ANONYMOUS logged in at Wed 17-Jun-87 12:01 PDT, job 15.
ftp> get netinfo:nug.doc
200 Port 18.144 at host 128.174.5.50 accepted.
150 ASCII retrieve of <NETINFO>NUG.DOC.11 started.
226 Transfer Completed 157675 (8) bytes transferred
local: netinfo:nug.doc  remote:netinfo:nug.doc
157675 bytes in 4.5e+02 seconds (0.34 Kbytes/s)
ftp> quit
221 QUIT command received. Goodbye.
```

(Another good initial document to fetch is NETINFO:WHAT-THE-NIC-DOES.TXT)!

Questions of the NIC or problems with services can be asked of or reported to using electronic mail. The following addresses can be used:

NIC@SRI-NIC.ARPA	General user assistance, document requests

REGISTRAR@SRI-NIC.ARPA	User registration and WHOIS updates
HOSTMASTER@SRI-NIC.ARPA	Hostname and domain changes and updates
ACTION@SRI-NIC.ARPA	SRI-NIC computer operations
SUGGESTIONS@SRI-NIC.ARPA	Comments on NIC publications and services

For people without network access, or if the number of documents is large, many of the NIC documents are available in printed form for a small charge. One frequently ordered document for starting sites is a compendium of major RFCs. Telephone access is used primarily for questions or problems with network access. (See appendix B for mail/telephone contact numbers).

The NSFnet Network Service Center

The NSFnet Network Service Center (NNSC) is funded by NSF to provide a first level of aid to users of NSFnet should they have questions or encounter problems traversing the network. It is run by BBN Inc. Karen Roubicek (roubicek@nnsc.nsf.net) is the NNSC user liaison.

The NNSC, which currently has information and documents online and in printed form, plans to distribute news through network mailing lists, bulletins, newsletters, and online reports. The NNSC also maintains a database of contact points and sources of additional information about NSFnet component networks and supercomputer centers.

Prospective or current users who do not know whom to call concerning questions about NSFnet use, should contact the NNSC. The NNSC will answer general questions, and, for detailed information relating to specific components of the Internet, will help users find the appropriate contact for further assistance. (Appendix B)

Mail Reflectors

The way most people keep up to date on network news is through subscription to a number of mail reflectors. Mail reflectors are special electronic mailboxes which, when they receive a message, resend it to a list of other mailboxes. This in effect creates a discussion group on a particular topic. Each subscriber sees all the mail forwarded by the reflector, and if one wants to put his ‘‘two cents’’ in sends a message with the comments to the reflector....

The general format to subscribe to a mail list is to find the address reflector and append the string –REQUEST to the mailbox name (not the host name). For example, if you wanted to take part in the mailing list for NSFnet reflected by NSFNET@NNSC.NSF.NET, one sends a request to NSFNET–REQUEST@NNSC.NSF.NET. This may be a wonderful scheme, but the problem is that you must know the list exists in the first place. It is suggested that, if you are interested, you read the mail from one list (like NSFNET) and you will probably become familiar with the existence of others. A registration service for mail reflectors is provided by the NIC in the files NETINFO:INTEREST-GROUPS-1.TXT, NETINFO:INTEREST-GROUPS-2.TXT, and NETINFO:INTEREST-GROUPS-3.TXT.

The NSFNET mail reflector is targeted at those people who have a day to day interest in the news of the NSFnet (the backbone, regional network, and Internet inter-connection site workers). The messages are reflected by a central location and are sent as separate messages to each subscriber. This creates hundreds of messages on the wide area networks where bandwidth is the scarcest.

There are two ways in which a campus could spread the news and not cause these messages to inundate the wide area networks. One is to re-reflect the message on the campus. That is, set up a reflector on a local machine which forwards the message to a campus distribution list. The other is to create an alias on a campus machine which places the messages into a notesfile on the topic. Campus users who want the information could access the notesfile and see the messages that have been sent since their last access. One might also elect to have the campus wide area network liaison screen the messages in either case and only forward those which are considered of merit. Either of these schemes allows one message to be sent to the campus, while allowing wide distribution within.

Address Allocation

Before a local network can be connected to the Internet it must be allocated a unique IP address. These addresses are allocated by ISI. The allocation process consists of getting an application form received from ISI. (Send a message to hostmaster@sri-nic.arpa and ask for the template for a connected address). This template is filled out and mailed back to hostmaster. An address is allocated and e-mailed back to you. This can also be done by postal mail (Appendix B).

IP addresses are 32 bits long. It is usually written as four decimal numbers separated by periods (e.g., 192.17.5.100). Each number is the value of an octet of the 32 bits. It was seen from the beginning that some networks might choose to organize themselves as very flat (one net with a lot of nodes) and some might organize hierarchically (many interconnected nets with fewer nodes each and a backbone). To provide for these cases, addresses were differentiated into class A, B, and C networks. This classification had to with the interpretation of the octets. Class A networks have the first octet as a network address and the remaining three as a host address on that network. Class C addresses have three octets of network address and one of host. Class B is split two and two. Therefore, there is an address space for a few large nets, a reasonable number of medium nets and a large number of small nets. The top two bits in the first octet are coded to tell the address format. All of the class A nets have been allocated. So one has to choose between Class B and Class C when placing an order. (There are also class D (Multicast) and E (Experimental) formats. Multicast addresses will likely come into greater use in the near future, but are not frequently used now).

In the past sites requiring multiple network addresses requested multiple discrete addresses (usually Class C). This was done because much of the software available (notably 4.2BSD) could not deal with subnetted addresses. Information on how to reach a particular network (routing information) must be stored in Internet gateways and packet switches. Some of these nodes have a limited capability to store and exchange routing information (limited to about 300 networks). Therefore, it is suggested that any campus announce (make known to the Internet) no more than two discrete network numbers.

If a campus expects to be constrained by this, it should consider subnetting. Subnetting (RFC-932) allows one to announce one address to the Internet and use a set of addresses on the campus. Basically, one defines a mask which allows the network to differentiate between the network portion and host portion of the address. By using a different mask on the Internet and the campus, the address can be interpreted in multiple ways. For example, if a campus requires two networks internally and has the 32,000 addresses beginning 128.174.X.X (a Class B address) allocated to it, the campus could allocate 128.174.5.X to one part of campus and 128.174.10.X to another. By advertising 128.174 to the Internet with a subnet mask of FF.FF.00.00, the Internet would treat these two addresses as one. Within the campus a mask of FF.FF.FF.00 would be used, allowing the campus to treat the addresses as separate entities. (In reality you don’t pass the subnet mask of FF.FF.00.00 to the Internet, the octet meaning is implicit in its being a class B address).

A word of warning is necessary. Not all systems know how to do subnetting. Some 4.2BSD systems require additional software. 4.3BSD systems subnet as released. Other devices and operating systems vary in the problems they have dealing with subnets. Frequently these machines can be used as a leaf on a network but not as a gateway within the subnetted portion of the network. As time passes and more systems become 4.3BSD based, these problems should disappear.

There has been some confusion in the past over the format of an IP broadcast address. Some machines used an address of all zeros to mean broadcast and some all ones. This was confusing when machines of both type were connected to the same network. The broadcast address of all ones has been adopted to end the grief. Some systems (e.g. 4.2 BSD) allow one to choose the format of the broadcast address. If a system does allow this choice, care should be taken that the all ones format is chosen. (This is explained in RFC-1009 and RFC-1010).

Internet Problems

There are a number of problems with the Internet. Solutions to the problems range from software changes to long term research projects. Some of the major ones are detailed below:

Number of Networks

When the Internet was designed it was to have about 50 connected networks. With the explosion of networking, the number is now approaching 300. The software in a group of critical gateways (called the core gateways of the ARPAnet) are not able to pass or store much more than that number. In the short term, core reallocation and recoding has raised the number slightly. By the summer of ’88 the current PDP-11 core gateways will be replaced with BBN Butterfly gateways which will solve the problem.

Routing Issues

Along with sheer mass of the data necessary to route packets to a large number of networks, there are many problems with the updating, stability, and optimality of the routing algorithms. Much research is being done in the area, but the optimal solution to these routing problems is still years away. In most cases the the routing we have today works, but sub-optimally and sometimes unpredictably.

Trust Issues

Gateways exchange network routing information. Currently, most gateways accept on faith that the information provided about the state of the network is correct. In the past this was not a big problem since most of the gateways belonged to a single administrative entity (DARPA). Now with multiple wide area networks under different administrations, a rogue gateway somewhere in the net could cripple the Internet. There is design work going on to solve both the problem of a gateway doing unreasonable things and providing enough information to reasonably route data between multiply connected networks (multi-homed networks).

Capacity & Congestion

Many portions of the ARPAnet are very congested during the busy part of the day. Additional links are planned to alleviate this congestion, but the implementation will take a few months.

These problems and the future direction of the Internet are determined by the Internet Architect (Dave Clark of MIT) being advised by the Internet Activities Board (IAB). This board is composed of chairmen of a number of committees with responsibility for various specialized areas of the Internet. The committees composing the IAB and their chairmen are:

Committee	*Chair*
Autonomous Networks	Deborah Estrin
End-to-End Services	Bob Braden
Internet Architecture	Dave Mills
Internet Engineering	Phil Gross
EGP2	Mike Petry
Name Domain Planning	Doug Kingston
Gateway Monitoring	Craig Partridge
Internic	Jake Feinler

Performance & Congestion Control	Robert Stine
NSF Routing	Chuck Hedrick
Misc. MilSup Issues	Mike St. Johns
Privacy Steve Kent	
IRINET Requirements	Vint Cerf
Robustness & Survivability	Jim Mathis
Scientific Requirements	Barry Leiner

Note that under Internet Engineering, there are a set of task forces and chairs to look at short term concerns. The chairs of these task forces are not part of the IAB.

Routing

Routing is the algorithm by which a network directs a packet from its source to its destination. To appreciate the problem, watch a small child trying to find a table in a restaurant. From the adult point of view the structure of the dining room is seen and an optimal route easily chosen. The child, however, is presented with a set of paths between tables where a good path, let alone the optimal one to the goal is not discernible.

A little more background might be appropriate. IP gateways (more correctly routers) are boxes which have connections to multiple networks and pass traffic between these nets. They decide how the packet is to be sent based on the information in the IP header of the packet and the state of the network. Each interface on a router has an unique address appropriate to the network to which it is connected. The information in the IP header which is used is primarily the destination address. Other information (e.g. type of service) is largely ignored at this time. The state of the network is determined by the routers passing information among themselves. The distribution of the database (what each node knows), the form of the updates, and metrics used to measure the value of a connection, are the parameters which determine the characteristics of a routing protocol.

Under some algorithms each node in the network has complete knowledge of the state of the network (the adult algorithm). This implies the nodes must have larger amounts of local storage and enough CPU to search the large tables in a short enough time (remember this must be done for each packet). Also, routing updates usually contain only changes to the existing information (or you spend a large amount of the network capacity passing around megabyte routing updates). This type of algorithm has several problems. Since the only way the routing information can be passed around is across the network and the propagation time is non-trivial, the view of the network at each node is a correct historical view of the network at varying times in the past. (The adult algorithm, but rather than looking directly at the dining area, looking at a photograph of the dining room. One is likely to pick the optimal route and find a bus-cart has moved in to block the path after the photo was taken). These inconsistencies can cause circular routes (called routing loops) where once a packet enters it is routed in a closed path until its time to live (TTL) field expires and it is discarded.

Other algorithms may know about only a subset of the network. To prevent loops in these protocols, they are usually used in a hierarchical network. They know completely about their own area, but to leave that area they go to one particular place (the default gateway). Typically these are used in smaller networks (campus, regional...).

Routing protocols in current use:

Static (no protocol-table/default routing)

> Don’t laugh. It is probably the most reliable, easiest to implement, and least likely to get one into trouble for a small network or a leaf on the Internet. This is, also, the only method available on some CPU–operating system combinations. If a host is connected to an Ethernet which has only

one gateway off of it, one should make that the default gateway for the host and do no other routing. (Of course that gateway may pass the reachablity information somehow on the other side of itself).

One word of warning, it is only with extreme caution that one should use static routes in the middle of a network which is also using dynamic routing. The routers passing dynamic information are sometimes confused by conflicting dynamic and static routes. If your host is on an ethernet with multiple routers to other networks on it and the routers are doing dynamic routing among themselves, it is usually better to take part in the dynamic routing than to use static routes.

RIP

RIP is a routing protocol based on XNS (Xerox Network System) adapted for IP networks. It is used by many routers (Proteon, cisco, UB...) and many BSD Unix systems. BSD systems typically run a program called *routed* to exchange information with other systems running RIP. RIP works best for nets of small diameter where the links are of equal speed. The reason for this is that the metric used to determine which path is best is the hop-count. A hop is a traversal across a gateway. So, all machines on the same Ethernet are zero hops away. If a router connects connects two networks directly, a machine on the other side of the router is one hop away.... As the routing information is passed through a gateway, the gateway adds one to the hop counts to keep them consistent across the network. The diameter of a network is defined as the largest hop-count possible within a network. Unfortunately, a hop count of 16 is defined as infinity in RIP meaning the link is down. Therefore, RIP will not allow hosts separated by more than 15 gateways in the RIP space to communicate.

The other problem with hop-count metrics is that if links have different speeds, that difference is not reflected in the hop-count. So a one hop satellite link (with a .5 sec delay) at 56kb would be used instead of a two hop T1 connection. Congestion can be viewed as a decrease in the efficacy of a link. So, as a link gets more congested, RIP will still know it is the best hop-count route and congest it even more by throwing more packets on the queue for that link.

The protocol is not well documented. A group of people are working on producing an RFC to both define the current RIP and to do some extensions to it to allow it to better cope with larger networks. Currently, the best documentation for RIP appears to be the code to BSD *routed.*

Routed

The *routed* program, which does RIP for 4.2BSD systems, has many options. One of the most frequently used is: *routed –q* (quiet mode) which means listen to RIP information but never broadcast it. This would be used by a machine on a network with multiple RIP speaking gateways. It allows the host to determine which gateway is best (hopwise) to use to reach a distant network. (Of course you might want to have a default gateway to prevent having to pass all the addresses known to the Internet around with RIP).

There are two ways to insert static routes into *routed*, the */etc/gateways* file and the *route add* command. Static routes are useful if you know how to reach a distant network, but you are not receiving that route using RIP. For the most part the *route add* command is preferable to use. The reason for this is that the command adds the route to that machine's routing table but does not export it through RIP. The */etc/gateways* file takes precedence over any routing information received through a RIP update. It is also broadcast as fact in RIP updates produced by the host without question, so if a mistake is made in the */etc/gateways* file, that mistake will soon permeate the RIP space and may bring the network to its knees.

One of the problems with *routed* is that you have very little control over what gets broadcast and what doesn't. Many times in larger networks where various parts of the network are under different administrative controls, you would like to pass on through RIP only nets which you receive from RIP and you know are reasonable. This prevents people from adding IP addresses to the network which may be illegal and you being responsible for passing them on to the Internet. This type of reasonability checks are not available with *routed* and leave it usable, but inadequate for large networks.

Hello (RFC-891)

Hello is a routing protocol which was designed and implemented in a experimental software router called a ‘‘Fuzzball’’ which runs on a PDP-11. It does not have wide usage, but is the routing protocol currently used on the NSFnet backbone. The data transferred between nodes is similar to RIP (a list of networks and their metrics). The metric, however, is milliseconds of delay. This allows Hello to be used over nets of various link speeds and performs better in congestive situations.

One of the most interesting side effects of Hello based networks is their great timekeeping ability. If you consider the problem of measuring delay on a link for the metric, you find that it is not an easy thing to do. You cannot measure round trip time since the return link may be more congested, of a different speed, or even not there. It is not really feasible for each node on the network to have a builtin WWV (nationwide radio time standard) receiver. So, you must design an algorithm to pass around time between nodes over the network links where the delay in transmission can only be approximated. Hello routers do this and in a nationwide network maintain synchronized time within milliseconds.

Exterior Gateway Protocol (EGP RFC-904)

EGP is not strictly a routing protocol, it is a reachability protocol. It tells only if nets can be reached through a particular gateway, not how good the connection is. It is the standard by which gateways to local nets inform the ARPAnet of the nets they can reach. There is a metric passed around by EGP but its usage is not standardized formally. Its typical value is value is 1 to 8 which are arbitrary goodness of link values understood by the internal DDN gateways. The smaller the value the better and a value of 8 being unreachable. A quirk of the protocol prevents distinguishing between 1 and 2, 3 and 4..., so the usablity of this as a metric is as three values and unreachable. Within NSFnet the values used are 1, 3, and unreachable. Many routers talk EGP so they can be used for ARPAnet gateways.

Gated

So we have regional and campus networks talking RIP among themselves, the NSFnet backbone talking Hello, and the DDN speaking EGP.

How do they interoperate? In the beginning there was static routing, assembled into the Fuzzball software configured for each site. The problem with doing static routing in the middle of the network is that it is broadcast to the Internet whether it is usable or not. Therefore, if a net becomes unreachable and you try to get there, dynamic routing will immediately issue a net unreachable to you. Under static routing the routers would think the net could be reached and would continue trying until the application gave up (in 2 or more minutes). Mark Fedor of Cornell (fedor@devvax.tn.cornell.edu) attempted to solve these problems with a replacement for *routed* called *gated*.

Gated talks RIP to RIP speaking hosts, EGP to EGP speakers, and Hello to Hello’ers. These speakers frequently all live on one Ethernet, but luckily (or unluckily) cannot understand each others ruminations. In addition, under configuration file control it can filter the conversion. For example, one can produce a configuration saying announce RIP nets via Hello only if they are specified in a list and are reachable by way of a RIP broadcast as well. This means that if a rogue network appears in your local site’s RIP space, it won’t be passed through to the Hello side of the world. There are also configuration options to do static routing and name trusted gateways.

This may sound like the greatest thing since sliced bread, but there is a catch called metric conversion. You have RIP measuring in hops, Hello measuring in milliseconds, and EGP using arbitrary small numbers. The big questions is how many hops to a millisecond, how many milliseconds in the EGP number 3.... Also, remember that infinity (unreachability) is 16 to RIP, 30000 or so to Hello, and 8 to the DDN with EGP. Getting all these metrics to work well together is no small feat. If done incorrectly and you translate an RIP of 16 into an EGP of 6, everyone in the ARPAnet will still think your gateway can reach the unreachable and will send every packet in the world your way. For these reasons, Mark requests that you consult closely with him when configuring and using *gated*.

Names

All routing across the network is done by means of the IP address associated with a packet. Since humans find it difficult to remember addresses like 128.174.5.50, a symbolic name register was set up at the NIC where people would say ‘‘I would like my host to be named ’uiucuxc’’’. Machines connected to the Internet across the nation would connect to the NIC in the middle of the night, check modification dates on the hosts file, and if modified move it to their local machine. With the advent of workstations and micros, changes to the host file would have to be made nightly. It would also be very labor intensive and consume a lot of network bandwidth. RFC-882 and a number of others describe domain name service, a distributed data base system for mapping names into addresses.

We must look a little more closely into what’s in a name. First, note that an address specifies a particular connection on a specific network. If the machine moves, the address changes. Second, a machine can have one or more names and one or more network addresses (connections) to different networks. Names point to a something which does useful work (i.e. the machine) and IP addresses point to an interface on that provider. A name is a purely symbolic representation of a list of addresses on the network. If a machine moves to a different network, the addresses will change but the name could remain the same.

Domain names are tree structured names with the root of the tree at the right. For example:

uxc.cso.uiuc.edu

is a machine called ’uxc’ (purely arbitrary), within the subdomains method of allocation of the U of I) and ’uiuc’ (the University of Illinois at Urbana), registered with ’edu’ (the set of educational institutions).

A simplified model of how a name is resolved is that on the user’s machine there is a resolver. The resolver knows how to contact across the network a root name server. Root servers are the base of the tree structured data retrieval system. They know who is responsible for handling first level domains (e.g. ’edu’). What root servers to use is an installation parameter. From the root server the resolver finds out who provides ’edu’ service. It contacts the ’edu’ name server which supplies it with a list of addresses of servers for the subdomains (like ’uiuc’). This action is repeated with the subdomain servers until the final subdomain returns a list of addresses of interfaces on the host in question. The user’s machine then has its choice of which of these addresses to use for communication.

A group may apply for its own domain name (like ’uiuc’ above). This is done in a manner similar to the IP address allocation. The only requirements are that the requestor have two machines reachable from the Internet, which will act as name servers for that domain. Those servers could also act as servers for subdomains or other servers could be designated as such. Note that the servers need not be located in any particular place, as long as they are reachable for name resolution. (U of I could ask Michigan State to act on its behalf and that would be fine). The biggest problem is that someone must do maintenance on the database. If the machine is not convenient, that might not be done in a timely fashion. The other thing to note is that once the domain is allocated to an administrative entity, that entity can freely allocate subdomains using what ever manner it sees fit.

The Berkeley Internet Name Domain (BIND) Server implements the Internet name server for UNIX systems. The name server is a distributed data base system that allows clients to name resources and to share that information with other network hosts. BIND is integrated with 4.3BSD and is used to lookup and store host names, addresses, mail agents, host information, and more. It replaces the */etc/hosts* file for host name lookup. BIND is still an evolving program. To keep up with reports on operational problems, future design decisions, etc, join the BIND mailing list by sending a request to *bind-request@ucbarpa.Berkeley.EDU*. BIND can also be obtained via anonymous FTP from ucbarpa.berkley.edu.

There are several advantages in using BIND. One of the most important is that it frees a host from relying on */etc/hosts* being up to date and complete. Within the .uiuc.edu domain, only a few hosts are included in the host table distributed by SRI. The remainder are listed locally within the BIND tables on uxc.cso.uiuc.edu (the server machine for most of the .uiuc.edu domain). All are equally reachable from any other Internet host running BIND.

BIND can also provide mail forwarding information for interior hosts not directly reachable from the Internet. These hosts can either be on non-advertised networks, or not connected to a network at all, as in the case of UUCP-reachable hosts. More information on BIND is available in the ‘‘Name Server Operations Guide for BIND’’ in *UNIX System Manager’s Manual,* 4.3BSD release.

There are a few special domains on the network, like SRI-NIC.ARPA. The ’arpa’ domain is historical, referring to hosts registered in the old hosts database at the NIC. There are others of the form NNSC.NSF.NET. These special domains are used sparingly and require ample justification. They refer to servers under the administrative control of the network rather than any single organization. This allows for the actual server to be moved around the net while the user interface to that machine remains constant. That is, should BBN relinquish control of the NNSC, the new provider would be pointed to by that name.

In actuality, the domain system is a much more general and complex system than has been described. Resolvers and some servers cache information to allow steps in the resolution to be skipped. Information provided by the servers can be arbitrary, not merely IP addresses. This allows the system to be used both by non-IP networks and for mail, where it may be necessary to give information on intermediate mail bridges.

What’s wrong with Berkeley Unix

University of California at Berkeley has been funded by DARPA to modify the Unix system in a number of ways. Included in these modifications is support for the Internet protocols. In earlier versions (e.g. BSD 4.2) there was good support for the basic Internet protocols (TCP, IP, SMTP, ARP) which allowed it to perform nicely on IP ethernets and smaller Internets. There were deficiencies, however, when it was connected to complicated networks. Most of these problems have been resolved under the newest release (BSD 4.3). Since it is the springboard from which many vendors have launched Unix implementations (either by porting the existing code or by using it as a model), many implementations (e.g. Ultrix) are still based on BSD 4.2. Therefore, many implementations still exist with the BSD 4.2 problems. As time goes on, when BSD 4.3 trickles through vendors as new release, many of the problems will be resolved. Following is a list of some problem scenarios and their handling under each of these releases.

ICMP redirects

> Under the Internet model, all a system needs to know to get anywhere in the Internet is its own address, the address of where it wants to go, and how to reach a gateway which knows about the Internet. It doesn’t have to be the best gateway. If the system is on a network with multiple gateways, and a host sends a packet for delivery to a gateway which feels another directly connected gateway is more appropriate, the gateway sends the sender a message. This message is an ICMP redirect, which politely says "I’ll deliver this message for you, but you really ought to use that gateway over there to reach this host". BSD 4.2 ignores these messages. This creates more stress

on the gateways and the local network, since for every packet sent, the gateway sends a packet to the originator. BSD 4.3 uses the redirect to update its routing tables, will use the route until it times out, then revert to the use of the route it thinks is should use. The whole process then repeats, but it is far better than one per packet.

Trailers

An application (like FTP) sends a string of octets to TCP which breaks it into chunks, and adds a TCP header. TCP then sends blocks of data to IP which adds its own headers and ships the packets over the network. All this prepending of the data with headers causes memory moves in both the sending and the receiving machines. Someone got the bright idea that if packets were long and they stuck the headers on the end (they became trailers), the receiving machine could put the packet on the beginning of a page boundary and if the trailer was OK merely delete it and transfer control of the page with no memory moves involved. The problem is that trailers were never standardized and most gateways don't know to look for the routing information at the end of the block. When trailers are used, the machine typically works fine on the local network (no gateways involved) and for short blocks through gateways (on which trailers aren't used). So TELNET and FTP's of very short files work just fine and FTP's of long files seem to hang. On BSD 4.2 trailers are a boot option and one should make sure they are off when using the Internet. BSD 4.3 negotiates trailers, so it uses them on its local net and doesn't use them when going across the network.

Retransmissions

TCP fires off blocks to its partner at the far end of the connection. If it doesn't receive an acknowledgement in a reasonable amount of time it retransmits the blocks. The determination of what is reasonable is done by TCP's retransmission algorithm. There is no correct algorithm but some are better than others, where better is measured by the number of retransmissions done unnecessarily. BSD 4.2 had a retransmission algorithm which retransmitted quickly and often. This is exactly what you would want if you had a bunch of machines on an ethernet (a low delay network of large bandwidth). If you have a network of relatively longer delay and scarce bandwidth (e.g. 56kb lines), it tends to retransmit too aggressively. Therefore, it makes the networks and gateways pass more traffic than is really necessary for a given conversation. Retransmission algorithms do adapt to the delay of the network after a few packets, but 4.2's adapts slowly in delay situations. BSD 4.3 does a lot better and tries to do the best for both worlds. It fires off a few retransmissions really quickly assuming it is on a low delay network, and then backs off very quickly. It also allows the delay to be about 4 minutes before it gives up and declares the connection broken.

Appendix A
References to Remedial Information

Quaterman and Hoskins, ‘‘Notable Computer Networks’’, *Communications of the ACM,* Vol 29, #10, pp. 932-971 (October, 1986).

Tannenbaum, Andrew S., *Computer Networks,* Prentice Hall, 1981.

Hedrick, Chuck, *Introduction to the Internet Protocols,* Anonymous FTP from topaz.rutgers.edu, directory pub/tcp-ip-docs, file tcp-ip-intro.doc.

Appendix B
List of Major RFCs

RFC-768	User Datagram Protocol (UDP)
RFC-791	Internet Protocol (IP)
RFC-792	Internet Control Message Protocol (ICMP)
RFC-793	Transmission Control Protocol (TCP)
RFC-821	Simple Mail Transfer Protocol (SMTP)
RFC-822	Standard for the Format of ARPA Internet Text Messages
RFC-854	Telnet Protocol
RFC-917 *	Internet Subnets
RFC-919 *	Broadcasting Internet Datagrams
RFC-922 *	Broadcasting Internet Datagrams in the Presence of Subnets
RFC-940 *	Toward an Internet Standard Scheme for Subnetting
RFC-947 *	Multi-network Broadcasting within the Internet
RFC-950 *	Internet Standard Subnetting Procedure
RFC-959	File Transfer Protocol (FTP)
RFC-966 *	Host Groups: A Multicast Extension to the Internet Protocol
RFC-988 *	Host Extensions for IP Multicasting
RFC-997 *	Internet Numbers
RFC-1010 *	Assigned Numbers
RFC-1011 *	Official ARPA-Internet Protocols

RFC’s marked with the asterisk (*) are not included in the 1985 DDN Protocol Handbook.

Note: This list is a portion of a list of RFC’s by topic retrieved from the NIC under NETINFO:RFC-SETS.TXT (anonymous FTP of course).

The following list is not necessary for connection to the Internet, but is useful in understanding the domain system, mail system, and gateways:

RFC-882	Domain Names - Concepts and Facilities
RFC-883	Domain Names - Implementation
RFC-973	Domain System Changes and Observations
RFC-974	Mail Routing and the Domain