Network Working Group
Request for Comments: 1120

V. Cerf
NRI

September 1989

The Internet Activities Board

Status of this Memo

This RFC provides a history and description of the Internet Activities Board (IAB) and its subsidiary organizations. This memo is for informational use and does not constitute a standard. Distribution of this memo is unlimited.

1. Introduction

In 1968, the U.S. Defense Advanced Research Projects Agency (DARPA) initiated an effort to develop a technology which is now known as packet switching. This technology had its roots in message switching methods, but was strongly influenced by the development of low-cost minicomputers and digital telecommunications techniques during the mid-1960's [BARAN 64, ROBERTS 70, HEART 70, ROBERTS 78]. A very useful survey of this technology can be found in [IEEE 78].

During the early 1970's, DARPA initiated a number of programs to explore the use of packet switching methods in alternative media including mobile radio, satellite and cable [IEEE 78, IEEE 87]. Concurrently, Xerox Palo Alto Research Center (PARC) began an exploration of packet switching on coaxial cable which ultimately led to the development of Ethernet local area networks [METCALFE 76].

The successful implementation of packet radio and packet satellite technology raised the question of interconnecting ARPANET with other types of packet nets. A possible solution to this problem was proposed by Cerf and Kahn [CERF 74] in the form of an internetwork protocol and a set of gateways to connect the different networks. This solution was further developed as part of a research program in internetting sponsored by DARPA and resulted in a collection of computer communications protocols based on the original Transmission Control Protocol (TCP) and its lower level counterpart, Internet Protocol (IP). Together, these protocols, along with many others developed during the course of the research, are referred to as the TCP/IP Protocol Suite [LEINER 85, POSTEL 85, CERF 82, CLARK 86, RFC 1100].

In the early stages of the Internet research program, only a few researchers worked to develop and test versions of the internet protocols. Over time, the size of this activity increased until, in

1979, it was necessary to form an informal committee to guide the technical evolution of the protocol suite. This group was called the Internet Configuration Control Board (ICCB) and was established by Dr. Vinton Cerf who was then the DARPA program manager for the effort. Dr. David C. Clark of the Lab for Computer Science at Massachusetts Institute of Technology was named the chairman of this committee.

In January, 1983, the Defense Communications Agency, then responsible for the operation of the ARPANET, declared the TCP/IP protocol suite to be standard for the ARPANET and all systems on the network converted from the earlier Network Control Program (NCP) to TCP/IP. Late that year, the ICCB was reorganized by Dr. Barry Leiner, Cerf's successor at DARPA, around a series of task forces considering different technical aspects of internetting. The re-organized group was named the Internet Activities Board.

As the Internet expanded, it drew support from U.S. Government organizations including DARPA, the National Science Foundation (NSF), the Department of Energy (DOE) and the National Aeronautics and Space Administration (NASA). Key managers in these organizations, responsible for computer networking research and development, formed an informal Federal Research Internet Coordinating Committee (FRICC) to coordinate U.S. Government support for and development and use of the Internet system. The FRICC sponsors most of the U.S. research on internetting, including support for the Internet Activities Board and its subsidiary organizations.

At the international level, a Coordinating Committee for Intercontinental Research Networks (CCIRN) has been formed which includes the U.S. FRICC and its counterparts in North America and Europe. The CCIRN provides a forum for cooperative planning among the principal North American and European research networking bodies.

2. Internet Activities Board

The Internet Activities Board (IAB) is the coordinating committee for Internet design, engineering and management. The Internet is a collection of over a thousand packet switched networks located principally in the U.S., but also includes systems in many other parts of the world, all interlinked and operating using the protocols of the TCP/IP protocol suite. The IAB is an independent committee of researchers and professionals with a technical interest in the health and evolution of the Internet system. Membership changes with time to adjust to the current realities of the research interests of the participants, the needs of the Internet system and the concerns of the U.S. Government, university and industrial sponsors of the elements of the Internet.

IAB members are deeply committed to making the Internet function effectively and evolve to meet a large scale, high speed future. All IAB members are required to have at least one other major role in the Internet community in addition to their IAB membership. New members are appointed by the chairman of the IAB, with the advice and consent of the remaining members. The chairman serves a term of two years.

The IAB focuses on the TCP/IP protocol suite, and extensions to the Internet system to support multiple protocol suites.

The IAB has two principal subsidiary task forces:

1) Internet Engineering Task Force (IETF)

2) Internet Research Task Force (IRTF)

Each of these Task Forces is led by a chairman and guided by a Steering Group which reports to the IAB through its chairman. Each task force is organized by the chairman, as required, to carry out its charter. For the most part, a collection of Working Groups carries out the work program of each Task Force.

All decisions of the IAB are made public. The principal vehicle by which IAB decisions are propagated to the parties interested in the Internet and its TCP/IP protocol suite is the Request for Comment (RFC) note series. The archival RFC series was initiated in 1969 by Dr. Stephen D. Crocker as a means of documenting the development of the original ARPANET protocol suite [RFC 1000]. The editor-in-chief of this series, Dr. Jonathan B. Postel, has maintained the quality of and managed the archiving of this series since its inception. A small proportion of the RFCs document Internet standards. Most of them are intended to stimulate comment and discussion. The small number which document standards are especially marked in a "status" section to indicate the special status of the document. An RFC summarizing the status of all standard RFCs is published regularly [RFC 1100].

RFCs describing experimental protocols, along with other submissions whose intent is merely to inform, are typically submitted directly to the RFC Editor. A Standard RFC starts out as a Proposed Standard and may be promoted to Draft Standard and finally Standard after suitable review, comment, implementation, and testing.

Prior to publication of a Proposed Standard, Draft Standard or Standard RFC, it is made available for comment through an on-line Internet-Draft directory. Typically, these Internet-Drafts are working documents of the IAB or of the working groups of the Internet Engineering and Research Task Forces. Internet Drafts are either

submitted to the RFC Editor for publication or discarded within three months.

The IAB performs the following functions:

1) Sets Internet Standards,

2) Manages the RFC publication process,

3) Reviews the operation of the IETF and IRTF,

4) Performs strategic planning for the Internet, identifying long-range problems and opportunities,

5) Acts as a technical policy liaison and representative for the Internet community, and

6) Resolves technical issues which cannot be treated within the IETF or IRTF frameworks.

To supplement its work via electronic mail, the IAB meets quarterly to review the condition of the Internet, to review and approve proposed changes or additions to the TCP/IP suite of protocols, to set technical development priorities, to discuss policy matters which may need the attention of the Internet sponsors, and to agree on the addition or retirement of IAB members and on the addition or retirement of task forces reporting to the IAB. Typically, two of the quarterly meetings are by means of video teleconferencing (provided, when possible, through the experimental Internet packet video-conferencing system).

The IAB membership is currently as follows:

Vinton Cerf	- Chairman
David Clark	- IRTF Chairman
Phillip Gross	- IETF Chairman
Jonathan Postel	- RFC Editor
Robert Braden	- Executive Director
Hans-Werner Braun	- Member
Barry Leiner	- Member
Daniel Lynch	- Member
Stephen Kent	- Member

3. The Internet Engineering Task Force

The Internet has grown to encompass a large number of widely geographically dispersed networks in academic and research communities. It now provides an infrastructure for a broad community with various

interests. Moreover, the family of Internet protocols and system components has moved from experimental to commercial development. To help coordinate the operation, management and evolution of the Internet, the IAB established the Internet Engineering Task Force (IETF).

The IETF is chaired by Mr. Phillip Gross and managed by its Internet Engineering Steering Group (IESG). The IAB has delegated to the IESG the general responsibility for making the Internet work and for the resolution of all short- and mid-range protocol and architectural issues required to make the Internet function effectively.

The charter of the IETF includes:

1) Responsibility for specifying the short and mid-term Internet protocols and architecture and recommending standards for IAB approval.

2) Provision of a forum for the exchange of information within the Internet community.

3) Identification of pressing and relevant short- to mid-range operational and technical problem areas and convening of Working Groups to explore solutions.

The Internet Engineering Task Force is a large open community of network designers, operators, vendors, and researchers concerned with the Internet and the Internet protocol suite. It is organized around a set of eight technical areas, each managed by a technical area director. In addition to the IETF Chairman, the area directors make up the IESG membership. Each area director has primary responsibility for one area of Internet engineering activity, and hence for a subset of the IETF Working Groups. The area directors have jobs of critical importance and difficulty and are selected not only for their technical expertise but also for their managerial skills and judgment. At present, the eight technical areas and chairs are:

1) Applications - TBD
2) Host Services - Craig Partridge
3) Internet Services - Noel Chiappa
4) Routing - Robert Hinden
5) Network Management - David Crocker
6) OSI Coexistence - Ross Callon and Robert Hagens
7) Operations - TBD
8) Security - TBD

The work of the IETF is performed by subcommittees known as Working

Groups. There are currently more than 20 of these. Working Groups tend to have a narrow focus and a lifetime bounded by completion of a specific task, although there are exceptions. The IETF is a major source of proposed protocol standards, for final approval by the IAB.

The IETF meets quarterly and extensive minutes of the plenary proceedings as well as reports from each of the working groups are issued by the IAB Secretariat, at the Corporation for National Research Initiatives.

4. The Internet Research Task Force

To promote research in networking and the development of new technology, the IAB established the Internet Research Task Force (IRTF).

In the area of network protocols, the distinction between research and engineering is not always clear, so there will sometimes be overlap between activities of the IETF and the IRTF. There is, in fact, considerable overlap in membership between the two groups. This overlap is regarded as vital for cross-fertilization and technology transfer. In general, the distinction between research and engineering is one of viewpoint and sometimes (but not always) time-frame. The IRTF is generally more concerned with understanding than with products or standard protocols, although specific experimental protocols may have to be developed, implemented and tested in order to gain understanding.

The IRTF is a community of network researchers, generally with an Internet focus. The work of the IRTF is governed by its Internet Research Steering Group (IRSG). The chairman of the IRTF and IRSG is David Clark. The IRTF is organized into a number of Research Groups (RGs) whose chairs are appointed by the chairman of the IRSG. The RG chairs and others selected by the IRSG chairman serve on the IRSG.

These groups typically have 10 to 20 members, and each covers a broad area of research, pursuing specific topics, determined at least in part by the interests of the members and by recommendations of the IAB.

The current members of the IRSG are as follows:

David Clark	- Chairman
Robert Braden	- End-to-End Services
Douglas Comer	- Member at Large
Deborah Estrin	- Autonomous Networks

Stephen Kent	- Privacy and Security
Keith Lantz	- User Interfaces
David Mills	- Member at Large

5. The Near-term Agenda of the IAB

There are seven principal foci of IAB attention for the period 1989 - 1990:

1) Operational Stability
2) User Services
3) OSI Coexistence
4) Testbed Facilities
5) Security
6) Getting Big
7) Getting Fast

Operational stability of the Internet is a critical concern for all of its users. Better tools are needed for gathering operational data, to assist in fault isolation at all levels and to analyze the performance of the system. Opportunities abound for increased cooperation among the operators of the various Internet components [RFC 1109]. Specific, known problems should be dealt with, such as implementation deficiencies in some version of the BIND domain name service resolver software. To the extent that the existing Exterior Gateway Protocol (EGP) is only able to support limited topologies, constraints on topological linkages and allowed transit paths should be enforced until a more general Inter-Autonomous System routing protocol can be specified. Flexibility for Internet implementation would be enhanced by the adoption of a common internal gateway routing protocol by all vendors of internet routers. A major effort is recommended to achieve conformance to the Host Requirements RFCs which are to be published early in the fourth quarter of calendar 1989.

Among the most needed user services, the White Pages (an electronic mailbox directory service) seems the most pressing. Efforts should be focused on widespread deployment of these capabilities in the Internet by mid-1990. The IAB recommends that existing white pages facilities and newer ones, such as X.500, be populated with up-to-date user information and made accessible to Internet users and users of other systems (e.g., commercial email carriers) linked to the Internet. Connectivity with commercial electronic mail carriers should be vigorously pursued, as well as links to other network research communities in Europe and the rest of the world.

Development and deployment of privacy-enhanced electronic mail software should be accelerated in 1990 after release of public domain

software implementing the private electronic mail standards [RFC 1113, RFC 1114, and RFC 1115]. Finally, support for new or enhanced applications such as computer-based conferencing, multi-media messaging and collaboration support systems should be developed.

The National Network Testbed (NNT) resources planned by the FRICC should be applied to support conferencing and collaboration protocol development and application experiments and to support multi-vendor router interoperability testing (e.g., interior and exterior routing, network management, multi-protocol routing and forwarding).

With respect to growth in the Internet, architectural attention should be focused on scaling the system to hundreds of millions of users and hundreds of thousands of networks. The naming, addressing, routing and navigation problems occasioned by such growth should be analyzed. Similarly, research should be carried out on analyzing the limits to the existing Internet architecture, including the ability of the present protocol suite to cope with speeds in the gigabit range and latencies varying from microseconds to seconds in duration.

The Internet should be positioned to support the use of OSI protocols by the end of 1990 or sooner, if possible. Provision for multi-protocol routing and forwarding among diverse vendor routes is one important goal. Introduction of X.400 electronic mail services and interoperation with RFC 822/SMTP [RFC 822, RFC 821, RFC 987, RFC 1026] should be targeted for 1990 as well. These efforts will need to work in conjunction with the White Pages services mentioned above. The IETF, in particular, should establish liaison with various OSI working groups (e.g., at NIST, RARE, Network Management Forum) to coordinate planning for OSI introduction into the Internet and to facilitate registration of information pertinent to the Internet with the various authorities responsible for OSI standards in the United States.

Security Considerations

Finally, with respect to security, a concerted effort should be made to develop guidance and documentation for Internet host managers concerning configuration management, known security problems (and their solutions) and software and technologies available to provide enhanced security and privacy to the users of the Internet.

REFERENCES

[BARAN 64] Baran, P., et al, "On Distributed Communications", Volumes I-XI, RAND Corporation Research Documents, August 1964.

[CERF 74] Cerf V., and R. Kahn, "A Protocol for Packet Network

Interconnection", IEEE Trans. on Communications, Vol. COM-22, No. 5, pp. 637-648, May 1974.

[CERF 82] Cerf V., and E. Cain, "The DoD Internet Protocol Architecture", Proceedings of the SHAPE Technology Center Symposium on Interoperability of Automated Data Systems, November 1982. Also in Computer Networks and ISDN, Vol. 17, No. 5, October 1983.

[CLARK 86] Clark, D., "The Design Philosophy of the DARPA Internet protocols", Proceedings of the SIGCOMM '88 Symposium, Computer Communications Review, Vol. 18, No. 4, pp. 106-114, August 1988.

[HEART 70] Heart, F., R. Kahn, S. Ornstein, W. Crowther, and D. Walden, "The Interface Message Processor for the ARPA Computer Network", AFIPS Conf. Proc. 36, pp. 551-567, June 1970.

[IEEE 78] Kahn, R. (Guest Editor), K. Uncapher, and H. Van Trees (Associate Guest Editors), Proceedings of the IEEE, Special Issue on Packet Communication Networks, Volume 66, No. 11, pp. 1303-1576, November 1978.

[IEEE 87] Leiner, B. (Guest Editor), D. Nielson, and F. Tobagi (Associate Guest Editors), Proceedings of the IEEE, Special Issue on Packet Radio Networks, Volume 75, No. 1, pp. 1-272, January 1987.

[LEINER 85] Leiner, B., R. Cole, J. Postel, and D. Mills, "The DARPA Protocol Suite", IEEE INFOCOM 85, Washington, D.C., March 1985. Also in IEEE Communications Magazine, March 1985.

[METCALFE 76] Metcalfe, R., and D. Boggs, "Ethernet: Distributed Packet for Local Computer Networks", Communications of the ACM, Vol. 19, No. 7, pp. 395-404, July 1976.

[POSTEL 85] Postel, J., "Internetwork Applications Using the DARPA Protocol Suite", IEEE INFOCOM 85, Washington, D.C., March 1985.

[RFC 821] Postel, J., "Simple Mail Transfer Protocol", RFC 821, USC/Information Sciences Institute, August 1982.

[RFC 822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", RFC 822, University of Delaware, August 1982.

[RFC 987] Kille, S., "Mapping between X.400 and RFC 822", University College London, June 1986.

[RFC 1000] Reynolds, J., and J. Postel, "The Request for Comments References Guide", USC/Information Sciences Institute, RFC 1000, August 1987.

[RFC 1026] Kille, S., "Addendum to RFC 987: (Mapping between X.400 and RFC 822)", RFC 1026, University College London, September 1987.

[RFC 1100] Postel, J. (Editor), "IAB Official Protocol Standards", RFC 1100, April 1989.

[RFC 1109] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, NRI, August 1989.

[RFC 1113] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part I -- Message Encipherment and Authentication Procedures", RFC 1113, IAB Privacy Task Force, August 1989.

[RFC 1114] Kent, S., and J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part II -- Certificate-based Key Management", RFC 1114, IAB Privacy Task Force, August 1989.

[RFC 1115] Linn, J., "Privacy Enhancement for Internet Electronic Mail: Part III -- Algorithms, Modes and Identifiers", RFC 1115, IAB Privacy Task Force, August 1989.

[ROBERTS 70] Roberts, L., and B. Wessler, "Computer Network Development to Achieve Resource Sharing", pp. 543-549, Proc. SJCC 1970.

[ROBERTS 78] Roberts, L., "Evolution of Packet Switching", Proc. IEEE, Vol. 66, No. 11, pp. 1307-1313, November 1978.

Note: RFCs are available from the Network Information Center at SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, (1-800-235-3155), or on-line via anonymous file transfer from NIC.DDN.MIL.

Author's Address

Vinton G. Cerf
Corporation for National Research Initiatives
1895 Preston White Drive, Suite 100
Reston, VA 22091

Phone: (703) 620-8990

EMail: VCERF@NRI.RESTON.VA.US

Network Working Group
Request for Comments: 1121

J. Postel (ISI)
L. Kleinrock (UCLA)
V. Cerf (NRI)
B. Boehm (UCLA)
September 1989

Act One - The Poems

Status of this Memo

This RFC presents a collection of poems that were presented at "Act One", a symposium held partially in celebration of the 20th anniversary of the ARPANET. Distribution of this memo is unlimited.

Introduction

The Computer Science Department of the University of California, Los Angeles (UCLA) organized a Symposium on Very High Speed Information Networks as the first in a projected series of meetings on Advanced Computer Technologies, thus ACT ONE. The time was chosen to also commemorate the 20th anniversary of the installation of the first Interface Message Processor (IMP) on the ARPANET which took place at UCLA.

The Symposium took on a theatrical theme and a few of the speakers could not resist the temptation to commit poetry. This memo is an attempt to capture the result.

The Poems

WELCOME
by
Leonard Kleinrock

We've gathered here for two days to examine and debate
And reflect on data networks and as well to celebrate.
To recognize the leaders and recount the path we took.
We'll begin with how it happened; for it's time to take a look.

Yes, the history is legend and the pioneers are here.
Listen to the story - it's our job to make it clear.
We'll tell you where we are now and where we'll likely go.
So welcome to ACT ONE, folks. Sit back - enjoy the show!!

ODE TO A QUEUE
by
Leonard Kleinrock

In the 20 years of funding
Many fields has DARPA led.
But the finest thing that they did bring
Was the analytic thread.

By that I mean they nurtured
Quantitative research tools.
And they always felt for all their gelt
They got principles and rules.

Indeed a wealth of knowledge
Was uncovered and was new.
And the common thread with which we led
Was the analytic queue!

Now a queue may have one server.
If there's more, they form a team.
Its dearest wish is just to fish
In a quiet Poisson stream.

If you want to model networks
Or a complex data flow
A queue's the key to help you see
All the things you need to know.

So the next time you feel lonely
And wonder what to do,
You'll soon feel fine if you join the line
Of an analytic queue!

THE PAST IS PROLOGUE
by
Leonard Kleinrock

The past is prologue so they say.
So Scene 1 was played today.
It set the stage to point the way
To high speed nets on Friday.

And old slow IMP, a costly link,
Codes to fix the lines that stink,
Ideas born in tanks that think,
Tomorrow's distance sure to shrink.

But first tonight we'll drink and eat.
We'll take some time good friends to greet.
Hear Bible class from Danny's seat.
Those good old days were bittersweet!

THE BIG BANG!
(or the birth of the ARPANET)
by
Leonard Kleinrock

It was back in '67 that the clan agreed to meet.
The gangsters and the planners were a breed damned hard to beat.
The goal we set was honest and the need was clear to all:
Connect those big old mainframes and the minis, lest they fall.

The spec was set quite rigid: it must work without a hitch.
It should stand a single failure with an unattended switch.
Files at hefty throughput 'cross the ARPANET must zip.
Send the interactive traffic on a quarter second trip.

The spec went out to bidders and t'was BBN that won.
They worked on soft and hardware and they all got paid for fun.
We decided that the first node would be we who are your hosts
And so today you're gathered here while UCLA boasts.

I suspect you might be asking "What means FIRST node on the net?"
Well frankly, it meant trouble, 'specially since no specs were set.
For you see the interface between the nascent IMP and HOST
Was a confidential secret from us folks on the West coast.

BBN had promised that the IMP was running late.
We welcomed any slippage in the deadly scheduled date.
But one day after Labor Day, it was plopped down at our gate!
Those dirty rotten scoundrels sent the damned thing out air freight!

As I recall that Tuesday, it makes me want to cry.
Everybody's brother came to blame the other guy!
Folks were there from ARPA, GTE and Honeywell.
UCLA and ATT and all were scared as hell.

We cautiously connected and the bits began to flow.
The pieces really functioned - just why I still don't know.
Messages were moving pretty well by Wednesday morn.
All the rest is history - packet switching had been born!

ROSENCRANTZ AND ETHERNET
by
Vint Cerf

All the world's a net! And all the data in it merely packets come to store-and-forward in the queues a while and then are heard no more. 'Tis a network waiting to be switched!

To switch or not to switch? That is the question. Whether 'tis wiser in the net to suffer the store and forward of stochastic networks or to raise up circuits against a sea of packets and, by dedication, serve them.

To net, to switch. To switch, perchance to slip!
Aye, there's the rub. For in that choice of switch,
what loops may lurk, when we have shuffled through
this Banyan net? Puzzles the will, initiates symposia,
stirs endless debate and gives rise to uncontrolled
flights of poetry beyond recompense!

UNTITLED
by
Barry Boehm

Paul Baran came out of the wood
With a message first misunderstood
But despite dangers lurking
The IMP's were soon working
And ARPA did see it was good.

So in place of our early myopia
We now have a net cornucopia
With IMP's, TIP's, and LAN's
Wideband VAN's, MAN's, and WAN's
And prospects of World Net Utopia.

But though we must wind up the clock
With thoughts of downstream feature shock
We all be can mollified
For there's no one more qualified
To discuss this than Leonard Kleinrock.

Notes

The Symposium was held August 17 & 18, 1989, a Thursday and Friday.

"Welcome" was presented on Thursday morning during the Overture.

"Ode to a Queue" was presented in the Thursday morning session on "Giant Steps Forward: Technology Payoffs".

"The Past is Prologue" was presented at the end of the Thursday afternoon sessions.

"The Big Bang!" was presented during the after dinner events on Thursday night.

"Rosencrantz and Ethernet" was presented at the morning session on Friday on "Communication Technologies in the next Millenium" (note that this version may differ slightly from the actual presentation since it was reconstructed from human memory several weeks later).

The untitled poem by Barry Boehm was presented in the Friday afternoon session on "Impact on Government, Commerce and Citizenry". Barry gave his talk on "The Software Challenge to Our Technical Aspirations" then introduced the next speaker with this poem.

Security Considerations

None.

Authors' Addresses

Jon Postel
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Phone: 213-822-1511

EMail: Postel@ISI.EDU

Leonard Kleinrock
University of California
Computer Science Department
3732G Boelter Hall
Los Angeles, CA 90024-1600

Phone: 213-825-2543

EMail: lk@CS.UCLA.EDU

Vinton G. Cerf
Corporation for National Research Initiatives
1895 Preston White Drive, Suite 100
Reston, VA 22091

Phone: 703-620-8990

EMail: VCerf@NRI.RESTON.VA.US

Barry Boehm
University of California
Computer Science Department
3732 Boelter Hall
Los Angeles, CA 90024-1600

Phone: 213-825-8137

EMail: boehm@CS.UCLA.EDU

Network Working Group
Request for Comments: 1122

Internet Engineering Task Force
R. Braden, Editor
October 1989

Requirements for Internet Hosts -- Communication Layers

Status of This Memo

This RFC is an official specification for the Internet community. It incorporates by reference, amends, corrects, and supplements the primary protocol standards documents relating to hosts. Distribution of this document is unlimited.

Summary

This is one RFC of a pair that defines and discusses the requirements for Internet host software. This RFC covers the communications protocol layers: link layer, IP layer, and transport layer; its companion RFC-1123 covers the application and support protocols.

Table of Contents

1. INTRODUCTION

This document is one of a pair that defines and discusses the requirements for host system implementations of the Internet protocol suite. This RFC covers the communication protocol layers: link layer, IP layer, and transport layer. Its companion RFC, "Requirements for Internet Hosts -- Application and Support" [INTRO:1], covers the application layer protocols. This document should also be read in conjunction with "Requirements for Internet Gateways" [INTRO:2].

These documents are intended to provide guidance for vendors, implementors, and users of Internet communication software. They represent the consensus of a large body of technical experience and wisdom, contributed by the members of the Internet research and vendor communities.

This RFC enumerates standard protocols that a host connected to the Internet must use, and it incorporates by reference the RFCs and other documents describing the current specifications for these protocols. It corrects errors in the referenced documents and adds additional discussion and guidance for an implementor.

For each protocol, this document also contains an explicit set of requirements, recommendations, and options. The reader must understand that the list of requirements in this document is incomplete by itself; the complete set of requirements for an Internet host is primarily defined in the standard protocol specification documents, with the corrections, amendments, and supplements contained in this RFC.

A good-faith implementation of the protocols that was produced after careful reading of the RFC's and with some interaction with the Internet technical community, and that followed good communications software engineering practices, should differ from the requirements of this document in only minor ways. Thus, in many cases, the "requirements" in this RFC are already stated or implied in the standard protocol documents, so that their inclusion here is, in a sense, redundant. However, they were included because some past implementation has made the wrong choice, causing problems of interoperability, performance, and/or robustness.

This document includes discussion and explanation of many of the requirements and recommendations. A simple list of requirements would be dangerous, because:

- Some required features are more important than others, and some features are optional.

- There may be valid reasons why particular vendor products that are designed for restricted contexts might choose to use different specifications.

However, the specifications of this document must be followed to meet the general goal of arbitrary host interoperation across the diversity and complexity of the Internet system. Although most current implementations fail to meet these requirements in various ways, some minor and some major, this specification is the ideal towards which we need to move.

These requirements are based on the current level of Internet architecture. This document will be updated as required to provide additional clarifications or to include additional information in those areas in which specifications are still evolving.

This introductory section begins with a brief overview of the Internet architecture as it relates to hosts, and then gives some general advice to host software vendors. Finally, there is some guidance on reading the rest of the document and some terminology.

1.1 The Internet Architecture

General background and discussion on the Internet architecture and supporting protocol suite can be found in the DDN Protocol Handbook [INTRO:3]; for background see for example [INTRO:9], [INTRO:10], and [INTRO:11]. Reference [INTRO:5] describes the procedure for obtaining Internet protocol documents, while [INTRO:6] contains a list of the numbers assigned within Internet protocols.

1.1.1 Internet Hosts

A host computer, or simply "host," is the ultimate consumer of communication services. A host generally executes application programs on behalf of user(s), employing network and/or Internet communication services in support of this function. An Internet host corresponds to the concept of an "End-System" used in the OSI protocol suite [INTRO:13].

An Internet communication system consists of interconnected packet networks supporting communication among host computers using the Internet protocols. The networks are interconnected using packet-switching computers called "gateways" or "IP routers" by the Internet community, and "Intermediate Systems" by the OSI world [INTRO:13]. The RFC "Requirements for Internet Gateways" [INTRO:2] contains the official specifications for Internet gateways. That RFC together with

the present document and its companion [INTRO:1] define the rules for the current realization of the Internet architecture.

Internet hosts span a wide range of size, speed, and function. They range in size from small microprocessors through workstations to mainframes and supercomputers. In function, they range from single-purpose hosts (such as terminal servers) to full-service hosts that support a variety of online network services, typically including remote login, file transfer, and electronic mail.

A host is generally said to be multihomed if it has more than one interface to the same or to different networks. See Section 1.1.3 on "Terminology".

1.1.2 Architectural Assumptions

The current Internet architecture is based on a set of assumptions about the communication system. The assumptions most relevant to hosts are as follows:

(a) The Internet is a network of networks.

Each host is directly connected to some particular network(s); its connection to the Internet is only conceptual. Two hosts on the same network communicate with each other using the same set of protocols that they would use to communicate with hosts on distant networks.

(b) Gateways don't keep connection state information.

To improve robustness of the communication system, gateways are designed to be stateless, forwarding each IP datagram independently of other datagrams. As a result, redundant paths can be exploited to provide robust service in spite of failures of intervening gateways and networks.

All state information required for end-to-end flow control and reliability is implemented in the hosts, in the transport layer or in application programs. All connection control information is thus co-located with the end points of the communication, so it will be lost only if an end point fails.

(c) Routing complexity should be in the gateways.

Routing is a complex and difficult problem, and ought to be performed by the gateways, not the hosts. An important

objective is to insulate host software from changes caused by the inevitable evolution of the Internet routing architecture.

(d) The System must tolerate wide network variation.

A basic objective of the Internet design is to tolerate a wide range of network characteristics -- e.g., bandwidth, delay, packet loss, packet reordering, and maximum packet size. Another objective is robustness against failure of individual networks, gateways, and hosts, using whatever bandwidth is still available. Finally, the goal is full "open system interconnection": an Internet host must be able to interoperate robustly and effectively with any other Internet host, across diverse Internet paths.

Sometimes host implementors have designed for less ambitious goals. For example, the LAN environment is typically much more benign than the Internet as a whole; LANs have low packet loss and delay and do not reorder packets. Some vendors have fielded host implementations that are adequate for a simple LAN environment, but work badly for general interoperation. The vendor justifies such a product as being economical within the restricted LAN market. However, isolated LANs seldom stay isolated for long; they are soon gatewayed to each other, to organization-wide internets, and eventually to the global Internet system. In the end, neither the customer nor the vendor is served by incomplete or substandard Internet host software.

The requirements spelled out in this document are designed for a full-function Internet host, capable of full interoperation over an arbitrary Internet path.

1.1.3 Internet Protocol Suite

To communicate using the Internet system, a host must implement the layered set of protocols comprising the Internet protocol suite. A host typically must implement at least one protocol from each layer.

The protocol layers used in the Internet architecture are as follows [INTRO:4]:

- Application Layer

The application layer is the top layer of the Internet protocol suite. The Internet suite does not further subdivide the application layer, although some of the Internet application layer protocols do contain some internal sub-layering. The application layer of the Internet suite essentially combines the functions of the top two layers -- Presentation and Application -- of the OSI reference model.

We distinguish two categories of application layer protocols: user protocols that provide service directly to users, and support protocols that provide common system functions. Requirements for user and support protocols will be found in the companion RFC [INTRO:1].

The most common Internet user protocols are:

- Telnet (remote login)
- FTP (file transfer)
- SMTP (electronic mail delivery)

There are a number of other standardized user protocols [INTRO:4] and many private user protocols.

Support protocols, used for host name mapping, booting, and management, include SNMP, BOOTP, RARP, and the Domain Name System (DNS) protocols.

- Transport Layer

The transport layer provides end-to-end communication services for applications. There are two primary transport layer protocols at present:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

TCP is a reliable connection-oriented transport service that provides end-to-end reliability, resequencing, and flow control. UDP is a connectionless ("datagram") transport service.

Other transport protocols have been developed by the research community, and the set of official Internet transport protocols may be expanded in the future.

Transport layer protocols are discussed in Chapter 4.

- Internet Layer

  All Internet transport protocols use the Internet Protocol (IP) to carry data from source host to destination host. IP is a connectionless or datagram internetwork service, providing no end-to-end delivery guarantees. Thus, IP datagrams may arrive at the destination host damaged, duplicated, out of order, or not at all. The layers above IP are responsible for reliable delivery service when it is required. The IP protocol includes provision for addressing, type-of-service specification, fragmentation and reassembly, and security information.

  The datagram or connectionless nature of the IP protocol is a fundamental and characteristic feature of the Internet architecture. Internet IP was the model for the OSI Connectionless Network Protocol [INTRO:12].

  ICMP is a control protocol that is considered to be an integral part of IP, although it is architecturally layered upon IP, i.e., it uses IP to carry its data end-to-end just as a transport protocol like TCP or UDP does. ICMP provides error reporting, congestion reporting, and first-hop gateway redirection.

  IGMP is an Internet layer protocol used for establishing dynamic host groups for IP multicasting.

  The Internet layer protocols IP, ICMP, and IGMP are discussed in Chapter 3.

- Link Layer

  To communicate on its directly-connected network, a host must implement the communication protocol used to interface to that network. We call this a link layer or media-access layer protocol.

  There is a wide variety of link layer protocols, corresponding to the many different types of networks. See Chapter 2.

1.1.4 Embedded Gateway Code

Some Internet host software includes embedded gateway functionality, so that these hosts can forward packets as a

gateway would, while still performing the application layer functions of a host.

Such dual-purpose systems must follow the Gateway Requirements RFC [INTRO:2] with respect to their gateway functions, and must follow the present document with respect to their host functions. In all overlapping cases, the two specifications should be in agreement.

There are varying opinions in the Internet community about embedded gateway functionality. The main arguments are as follows:

- Pro: in a local network environment where networking is informal, or in isolated internets, it may be convenient and economical to use existing host systems as gateways.

  There is also an architectural argument for embedded gateway functionality: multihoming is much more common than originally foreseen, and multihoming forces a host to make routing decisions as if it were a gateway. If the multihomed host contains an embedded gateway, it will have full routing knowledge and as a result will be able to make more optimal routing decisions.

- Con: Gateway algorithms and protocols are still changing, and they will continue to change as the Internet system grows larger. Attempting to include a general gateway function within the host IP layer will force host system maintainers to track these (more frequent) changes. Also, a larger pool of gateway implementations will make coordinating the changes more difficult. Finally, the complexity of a gateway IP layer is somewhat greater than that of a host, making the implementation and operation tasks more complex.

  In addition, the style of operation of some hosts is not appropriate for providing stable and robust gateway service.

There is considerable merit in both of these viewpoints. One conclusion can be drawn: an host administrator must have conscious control over whether or not a given host acts as a gateway. See Section 3.1 for the detailed requirements.

1.2 General Considerations

There are two important lessons that vendors of Internet host software have learned and which a new vendor should consider seriously.

1.2.1 Continuing Internet Evolution

The enormous growth of the Internet has revealed problems of management and scaling in a large datagram-based packet communication system. These problems are being addressed, and as a result there will be continuing evolution of the specifications described in this document. These changes will be carefully planned and controlled, since there is extensive participation in this planning by the vendors and by the organizations responsible for operations of the networks.

Development, evolution, and revision are characteristic of computer network protocols today, and this situation will persist for some years. A vendor who develops computer communication software for the Internet protocol suite (or any other protocol suite!) and then fails to maintain and update that software for changing specifications is going to leave a trail of unhappy customers. The Internet is a large communication network, and the users are in constant contact through it. Experience has shown that knowledge of deficiencies in vendor software propagates quickly through the Internet technical community.

1.2.2 Robustness Principle

At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability [IP:1]:

> "Be liberal in what you accept, and
> conservative in what you send"

Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect. This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by unenvisaged mechanisms triggered by low-probability events;

mere human malice would never have taken so devious a course!

Adaptability to change must be designed into all levels of Internet host software. As a simple example, consider a protocol specification that contains an enumeration of values for a particular header field -- e.g., a type field, a port number, or an error code; this enumeration must be assumed to be incomplete. Thus, if a protocol specification defines four possible error codes, the software must not break when a fifth code shows up. An undefined code might be logged (see below), but it must not cause a failure.

The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features. It is unwise to stray far from the obvious and simple, lest untoward effects result elsewhere. A corollary of this is "watch out for misbehaving hosts"; host software should be prepared, not just to survive other misbehaving hosts, but also to cooperate to limit the amount of disruption such hosts can cause to the shared communication facility.

1.2.3 Error Logging

The Internet includes a great variety of host and gateway systems, each implementing many protocols and protocol layers, and some of these contain bugs and mis-features in their Internet protocol software. As a result of complexity, diversity, and distribution of function, the diagnosis of Internet problems is often very difficult.

Problem diagnosis will be aided if host implementations include a carefully designed facility for logging erroneous or "strange" protocol events. It is important to include as much diagnostic information as possible when an error is logged. In particular, it is often useful to record the header(s) of a packet that caused an error. However, care must be taken to ensure that error logging does not consume prohibitive amounts of resources or otherwise interfere with the operation of the host.

There is a tendency for abnormal but harmless protocol events to overflow error logging files; this can be avoided by using a "circular" log, or by enabling logging only while diagnosing a known failure. It may be useful to filter and count duplicate successive messages. One strategy that seems to work well is: (1) always count abnormalities and make such counts accessible through the management protocol (see [INTRO:1]); and (2) allow

the logging of a great variety of events to be selectively enabled. For example, it might useful to be able to "log everything" or to "log everything for host X".

Note that different managements may have differing policies about the amount of error logging that they want normally enabled in a host. Some will say, "if it doesn't hurt me, I don't want to know about it", while others will want to take a more watchful and aggressive attitude about detecting and removing protocol abnormalities.

1.2.4 Configuration

It would be ideal if a host implementation of the Internet protocol suite could be entirely self-configuring. This would allow the whole suite to be implemented in ROM or cast into silicon, it would simplify diskless workstations, and it would be an immense boon to harried LAN administrators as well as system vendors. We have not reached this ideal; in fact, we are not even close.

At many points in this document, you will find a requirement that a parameter be a configurable option. There are several different reasons behind such requirements. In a few cases, there is current uncertainty or disagreement about the best value, and it may be necessary to update the recommended value in the future. In other cases, the value really depends on external factors -- e.g., the size of the host and the distribution of its communication load, or the speeds and topology of nearby networks -- and self-tuning algorithms are unavailable and may be insufficient. In some cases, configurability is needed because of administrative requirements.

Finally, some configuration options are required to communicate with obsolete or incorrect implementations of the protocols, distributed without sources, that unfortunately persist in many parts of the Internet. To make correct systems coexist with these faulty systems, administrators often have to "misconfigure" the correct systems. This problem will correct itself gradually as the faulty systems are retired, but it cannot be ignored by vendors.

When we say that a parameter must be configurable, we do not intend to require that its value be explicitly read from a configuration file at every boot time. We recommend that implementors set up a default for each parameter, so a configuration file is only necessary to override those defaults

that are inappropriate in a particular installation. Thus, the configurability requirement is an assurance that it will be POSSIBLE to override the default when necessary, even in a binary-only or ROM-based product.

This document requires a particular value for such defaults in some cases. The choice of default is a sensitive issue when the configuration item controls the accommodation to existing faulty systems. If the Internet is to converge successfully to complete interoperability, the default values built into implementations must implement the official protocol, not "mis-configurations" to accommodate faulty implementations. Although marketing considerations have led some vendors to choose mis-configuration defaults, we urge vendors to choose defaults that will conform to the standard.

Finally, we note that a vendor needs to provide adequate documentation on all configuration parameters, their limits and effects.

1.3 Reading this Document

1.3.1 Organization

Protocol layering, which is generally used as an organizing principle in implementing network software, has also been used to organize this document. In describing the rules, we assume that an implementation does strictly mirror the layering of the protocols. Thus, the following three major sections specify the requirements for the link layer, the internet layer, and the transport layer, respectively. A companion RFC [INTRO:1] covers application level software. This layerist organization was chosen for simplicity and clarity.

However, strict layering is an imperfect model, both for the protocol suite and for recommended implementation approaches. Protocols in different layers interact in complex and sometimes subtle ways, and particular functions often involve multiple layers. There are many design choices in an implementation, many of which involve creative "breaking" of strict layering. Every implementor is urged to read references [INTRO:7] and [INTRO:8].

This document describes the conceptual service interface between layers using a functional ("procedure call") notation, like that used in the TCP specification [TCP:1]. A host implementation must support the logical information flow

implied by these calls, but need not literally implement the calls themselves. For example, many implementations reflect the coupling between the transport layer and the IP layer by giving them shared access to common data structures. These data structures, rather than explicit procedure calls, are then the agency for passing much of the information that is required.

In general, each major section of this document is organized into the following subsections:

(1) Introduction

(2) Protocol Walk-Through -- considers the protocol specification documents section-by-section, correcting errors, stating requirements that may be ambiguous or ill-defined, and providing further clarification or explanation.

(3) Specific Issues -- discusses protocol design and implementation issues that were not included in the walk-through.

(4) Interfaces -- discusses the service interface to the next higher layer.

(5) Summary -- contains a summary of the requirements of the section.

Under many of the individual topics in this document, there is parenthetical material labeled "DISCUSSION" or "IMPLEMENTATION". This material is intended to give clarification and explanation of the preceding requirements text. It also includes some suggestions on possible future directions or developments. The implementation material contains suggested approaches that an implementor may want to consider.

The summary sections are intended to be guides and indexes to the text, but are necessarily cryptic and incomplete. The summaries should never be used or referenced separately from the complete RFC.

1.3.2 Requirements

In this document, the words that are used to define the significance of each particular requirement are capitalized.

These words are:

* "MUST"

  This word or the adjective "REQUIRED" means that the item is an absolute requirement of the specification.

* "SHOULD"

  This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "MAY"

  This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the MUST requirements for the protocols it implements. An implementation that satisfies all the MUST and all the SHOULD requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements for its protocols is said to be "conditionally compliant".

1.3.3 Terminology

This document uses the following technical terms:

Segment
: A segment is the unit of end-to-end transmission in the TCP protocol. A segment consists of a TCP header followed by application data. A segment is transmitted by encapsulation inside an IP datagram.

Message
: In this description of the lower-layer protocols, a message is the unit of transmission in a transport layer protocol. In particular, a TCP segment is a message. A message consists of a transport protocol header followed by application protocol data. To be transmitted end-to-

end through the Internet, a message must be encapsulated inside a datagram.

IP Datagram
An IP datagram is the unit of end-to-end transmission in the IP protocol. An IP datagram consists of an IP header followed by transport layer data, i.e., of an IP header followed by a message.

In the description of the internet layer (Section 3), the unqualified term "datagram" should be understood to refer to an IP datagram.

Packet
A packet is the unit of data passed across the interface between the internet layer and the link layer. It includes an IP header and data. A packet may be a complete IP datagram or a fragment of an IP datagram.

Frame
A frame is the unit of transmission in a link layer protocol, and consists of a link-layer header followed by a packet.

Connected Network
A network to which a host is interfaced is often known as the "local network" or the "subnetwork" relative to that host. However, these terms can cause confusion, and therefore we use the term "connected network" in this document.

Multihomed
A host is said to be multihomed if it has multiple IP addresses. For a discussion of multihoming, see Section 3.3.4 below.

Physical network interface
This is a physical interface to a connected network and has a (possibly unique) link-layer address. Multiple physical network interfaces on a single host may share the same link-layer address, but the address must be unique for different hosts on the same physical network.

Logical [network] interface
We define a logical [network] interface to be a logical path, distinguished by a unique IP address, to a connected network. See Section 3.3.4.

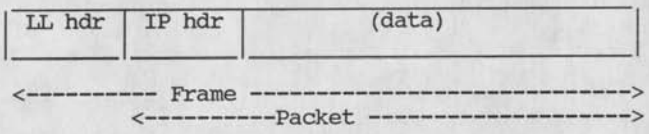
Specific-destination address
This is the effective destination address of a datagram, even if it is broadcast or multicast; see Section 3.2.1.3.

Path
At a given moment, all the IP datagrams from a particular source host to a particular destination host will typically traverse the same sequence of gateways. We use the term "path" for this sequence. Note that a path is uni-directional; it is not unusual to have different paths in the two directions between a given host pair.

MTU
The maximum transmission unit, i.e., the size of the largest packet that can be transmitted.

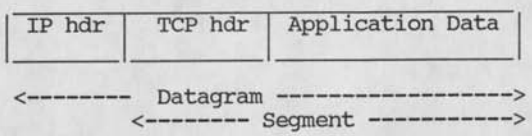
The terms frame, packet, datagram, message, and segment are illustrated by the following schematic diagrams:

A. Transmission on connected network:



B. Before IP fragmentation or after IP reassembly:

or, for TCP:



1.4 Acknowledgments

This document incorporates contributions and comments from a large group of Internet protocol experts, including representatives of university and research labs, vendors, and government agencies. It was assembled primarily by the Host Requirements Working Group of the Internet Engineering Task Force (IETF).

The Editor would especially like to acknowledge the tireless dedication of the following people, who attended many long meetings and generated 3 million bytes of electronic mail over the past 18 months in pursuit of this document: Philip Almquist, Dave Borman (Cray Research), Noel Chiappa, Dave Crocker (DEC), Steve Deering (Stanford), Mike Karels (Berkeley), Phil Karn (Bellcore), John Lekashman (NASA), Charles Lynn (BBN), Keith McCloghrie (TWG), Paul Mockapetris (ISI), Thomas Narten (Purdue), Craig Partridge (BBN), Drew Perkins (CMU), and James Van Bokkelen (FTP Software).

In addition, the following people made major contributions to the effort: Bill Barns (Mitre), Steve Bellovin (AT&T), Mike Brescia (BBN), Ed Cain (DCA), Annette DeSchon (ISI), Martin Gross (DCA), Phill Gross (NRI), Charles Hedrick (Rutgers), Van Jacobson (LBL), John Klensin (MIT), Mark Lottor (SRI), Milo Medin (NASA), Bill Melohn (Sun Microsystems), Greg Minshall (Kinetics), Jeff Mogul (DEC), John Mullen (CMC), Jon Postel (ISI), John Romkey (Epilogue Technology), and Mike StJohns (DCA). The following also made significant contributions to particular areas: Eric Allman (Berkeley), Rob Austein (MIT), Art Berggreen (ACC), Keith Bostic (Berkeley), Vint Cerf (NRI), Wayne Hathaway (NASA), Matt Korn (IBM), Erik Naggum (Naggum Software, Norway), Robert Ullmann (Prime Computer), David Waitzman (BBN), Frank Wancho (USA), Arun Welch (Ohio State), Bill Westfield (Cisco), and Rayan Zachariassen (Toronto).

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

2. LINK LAYER

2.1 INTRODUCTION

All Internet systems, both hosts and gateways, have the same requirements for link layer protocols. These requirements are given in Chapter 3 of "Requirements for Internet Gateways" [INTRO:2], augmented with the material in this section.

2.2 PROTOCOL WALK-THROUGH

None.

2.3 SPECIFIC ISSUES

2.3.1 Trailer Protocol Negotiation

The trailer protocol [LINK:1] for link-layer encapsulation MAY be used, but only when it has been verified that both systems (host or gateway) involved in the link-layer communication implement trailers. If the system does not dynamically negotiate use of the trailer protocol on a per-destination basis, the default configuration MUST disable the protocol.

DISCUSSION:

The trailer protocol is a link-layer encapsulation technique that rearranges the data contents of packets sent on the physical network. In some cases, trailers improve the throughput of higher layer protocols by reducing the amount of data copying within the operating system. Higher layer protocols are unaware of trailer use, but both the sending and receiving host MUST understand the protocol if it is used.

Improper use of trailers can result in very confusing symptoms. Only packets with specific size attributes are encapsulated using trailers, and typically only a small fraction of the packets being exchanged have these attributes. Thus, if a system using trailers exchanges packets with a system that does not, some packets disappear into a black hole while others are delivered successfully.

IMPLEMENTATION:

On an Ethernet, packets encapsulated with trailers use a distinct Ethernet type [LINK:1], and trailer negotiation is performed at the time that ARP is used to discover the link-layer address of a destination system.

Specifically, the ARP exchange is completed in the usual manner using the normal IP protocol type, but a host that wants to speak trailers will send an additional "trailer ARP reply" packet, i.e., an ARP reply that specifies the trailer encapsulation protocol type but otherwise has the format of a normal ARP reply. If a host configured to use trailers receives a trailer ARP reply message from a remote machine, it can add that machine to the list of machines that understand trailers, e.g., by marking the corresponding entry in the ARP cache.

Hosts wishing to receive trailer encapsulations send trailer ARP replies whenever they complete exchanges of normal ARP messages for IP. Thus, a host that received an ARP request for its IP protocol address would send a trailer ARP reply in addition to the normal IP ARP reply; a host that sent the IP ARP request would send a trailer ARP reply when it received the corresponding IP ARP reply. In this way, either the requesting or responding host in an IP ARP exchange may request that it receive trailer encapsulations.

This scheme, using extra trailer ARP reply packets rather than sending an ARP request for the trailer protocol type, was designed to avoid a continuous exchange of ARP packets with a misbehaving host that, contrary to any specification or common sense, responded to an ARP reply for trailers with another ARP reply for IP. This problem is avoided by sending a trailer ARP reply in response to an IP ARP reply only when the IP ARP reply answers an outstanding request; this is true when the hardware address for the host is still unknown when the IP ARP reply is received. A trailer ARP reply may always be sent along with an IP ARP reply responding to an IP ARP request.

2.3.2 Address Resolution Protocol -- ARP

2.3.2.1 ARP Cache Validation

An implementation of the Address Resolution Protocol (ARP) [LINK:2] MUST provide a mechanism to flush out-of-date cache entries. If this mechanism involves a timeout, it SHOULD be possible to configure the timeout value.

A mechanism to prevent ARP flooding (repeatedly sending an ARP Request for the same IP address, at a high rate) MUST be included. The recommended maximum rate is 1 per second per

destination.

DISCUSSION:

The ARP specification [LINK:2] suggests but does not require a timeout mechanism to invalidate cache entries when hosts change their Ethernet addresses. The prevalence of proxy ARP (see Section 2.4 of [INTRO:2]) has significantly increased the likelihood that cache entries in hosts will become invalid, and therefore some ARP-cache invalidation mechanism is now required for hosts. Even in the absence of proxy ARP, a long-period cache timeout is useful in order to automatically correct any bad ARP data that might have been cached.

IMPLEMENTATION:

Four mechanisms have been used, sometimes in combination, to flush out-of-date cache entries.

(1) Timeout -- Periodically time out cache entries, even if they are in use. Note that this timeout should be restarted when the cache entry is "refreshed" (by observing the source fields, regardless of target address, of an ARP broadcast from the system in question). For proxy ARP situations, the timeout needs to be on the order of a minute.

(2) Unicast Poll -- Actively poll the remote host by periodically sending a point-to-point ARP Request to it, and delete the entry if no ARP Reply is received from N successive polls. Again, the timeout should be on the order of a minute, and typically N is 2.

(3) Link-Layer Advice -- If the link-layer driver detects a delivery problem, flush the corresponding ARP cache entry.

(4) Higher-layer Advice -- Provide a call from the Internet layer to the link layer to indicate a delivery problem. The effect of this call would be to invalidate the corresponding cache entry. This call would be analogous to the "ADVISE_DELIVPROB()" call from the transport layer to the Internet layer (see Section 3.4), and in fact the ADVISE_DELIVPROB routine might in turn call the link-layer advice routine to invalidate

the ARP cache entry.

Approaches (1) and (2) involve ARP cache timeouts on the order of a minute or less. In the absence of proxy ARP, a timeout this short could create noticeable overhead traffic on a very large Ethernet. Therefore, it may be necessary to configure a host to lengthen the ARP cache timeout.

2.3.2.2 ARP Packet Queue

The link layer SHOULD save (rather than discard) at least one (the latest) packet of each set of packets destined to the same unresolved IP address, and transmit the saved packet when the address has been resolved.

DISCUSSION:
Failure to follow this recommendation causes the first packet of every exchange to be lost. Although higher-layer protocols can generally cope with packet loss by retransmission, packet loss does impact performance. For example, loss of a TCP open request causes the initial round-trip time estimate to be inflated. UDP-based applications such as the Domain Name System are more seriously affected.

2.3.3 Ethernet and IEEE 802 Encapsulation

The IP encapsulation for Ethernets is described in RFC-894 [LINK:3], while RFC-1042 [LINK:4] describes the IP encapsulation for IEEE 802 networks. RFC-1042 elaborates and replaces the discussion in Section 3.4 of [INTRO:2].

Every Internet host connected to a 10Mbps Ethernet cable:

- MUST be able to send and receive packets using RFC-894 encapsulation;
- SHOULD be able to receive RFC-1042 packets, intermixed with RFC-894 packets; and
- MAY be able to send packets using RFC-1042 encapsulation.

An Internet host that implements sending both the RFC-894 and the RFC-1042 encapsulations MUST provide a configuration switch to select which is sent, and this switch MUST default to RFC-894.

Note that the standard IP encapsulation in RFC-1042 does not use the protocol id value (K1=6) that IEEE reserved for IP; instead, it uses a value (K1=170) that implies an extension (the "SNAP") which can be used to hold the Ether-Type field. An Internet system MUST NOT send 802 packets using K1=6.

Address translation from Internet addresses to link-layer addresses on Ethernet and IEEE 802 networks MUST be managed by the Address Resolution Protocol (ARP).

The MTU for an Ethernet is 1500 and for 802.3 is 1492.

DISCUSSION:

The IEEE 802.3 specification provides for operation over a 10Mbps Ethernet cable, in which case Ethernet and IEEE 802.3 frames can be physically intermixed. A receiver can distinguish Ethernet and 802.3 frames by the value of the 802.3 Length field; this two-octet field coincides in the header with the Ether-Type field of an Ethernet frame. In particular, the 802.3 Length field must be less than or equal to 1500, while all valid Ether-Type values are greater than 1500.

Another compatibility problem arises with link-layer broadcasts. A broadcast sent with one framing will not be seen by hosts that can receive only the other framing.

The provisions of this section were designed to provide direct interoperation between 894-capable and 1042-capable systems on the same cable, to the maximum extent possible. It is intended to support the present situation where 894-only systems predominate, while providing an easy transition to a possible future in which 1042-capable systems become common.

Note that 894-only systems cannot interoperate directly with 1042-only systems. If the two system types are set up as two different logical networks on the same cable, they can communicate only through an IP gateway. Furthermore, it is not useful or even possible for a dual-format host to discover automatically which format to send, because of the problem of link-layer broadcasts.

2.4 LINK/INTERNET LAYER INTERFACE

The packet receive interface between the IP layer and the link layer MUST include a flag to indicate whether the incoming packet was addressed to a link-layer broadcast address.

DISCUSSION

Although the IP layer does not generally know link layer addresses (since every different network medium typically has a different address format), the broadcast address on a broadcast-capable medium is an important special case. See Section 3.2.2, especially the DISCUSSION concerning broadcast storms.

The packet send interface between the IP and link layers MUST include the 5-bit TOS field (see Section 3.2.1.6).

The link layer MUST NOT report a Destination Unreachable error to IP solely because there is no ARP cache entry for a destination.

2.5 LINK LAYER REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Trailer encapsulation	2.3.1			x			
Send Trailers by default without negotiation	2.3.1					x	
ARP	2.3.2						
Flush out-of-date ARP cache entries	2.3.2.1	x					
Prevent ARP floods	2.3.2.1	x					
Cache timeout configurable	2.3.2.1		x				
Save at least one (latest) unresolved pkt	2.3.2.2		x				
Ethernet and IEEE 802 Encapsulation	2.3.3						
Host able to:	2.3.3						
Send & receive RFC-894 encapsulation	2.3.3	x					
Receive RFC-1042 encapsulation	2.3.3		x				
Send RFC-1042 encapsulation	2.3.3			x			
Then config. sw. to select, RFC-894 dflt	2.3.3	x					
Send K1=6 encapsulation	2.3.3					x	
Use ARP on Ethernet and IEEE 802 nets	2.3.3	x					
Link layer report b'casts to IP layer	2.4	x					
IP layer pass TOS to link layer	2.4	x					
No ARP cache entry treated as Dest. Unreach.	2.4					x	

3. INTERNET LAYER PROTOCOLS

3.1 INTRODUCTION

The Robustness Principle: "Be liberal in what you accept, and conservative in what you send" is particularly important in the Internet layer, where one misbehaving host can deny Internet service to many other hosts.

The protocol standards used in the Internet layer are:

- RFC-791 [IP:1] defines the IP protocol and gives an introduction to the architecture of the Internet.

- RFC-792 [IP:2] defines ICMP, which provides routing, diagnostic and error functionality for IP. Although ICMP messages are encapsulated within IP datagrams, ICMP processing is considered to be (and is typically implemented as) part of the IP layer. See Section 3.2.2.

- RFC-950 [IP:3] defines the mandatory subnet extension to the addressing architecture.

- RFC-1112 [IP:4] defines the Internet Group Management Protocol IGMP, as part of a recommended extension to hosts and to the host-gateway interface to support Internet-wide multicasting at the IP level. See Section 3.2.3.

  The target of an IP multicast may be an arbitrary group of Internet hosts. IP multicasting is designed as a natural extension of the link-layer multicasting facilities of some networks, and it provides a standard means for local access to such link-layer multicasting facilities.

Other important references are listed in Section 5 of this document.

The Internet layer of host software MUST implement both IP and ICMP. See Section 3.3.7 for the requirements on support of IGMP.

The host IP layer has two basic functions: (1) choose the "next hop" gateway or host for outgoing IP datagrams and (2) reassemble incoming IP datagrams. The IP layer may also (3) implement intentional fragmentation of outgoing datagrams. Finally, the IP layer must (4) provide diagnostic and error functionality. We expect that IP layer functions may increase somewhat in the future, as further Internet control and management facilities are developed.

For normal datagrams, the processing is straightforward. For incoming datagrams, the IP layer:

(1) verifies that the datagram is correctly formatted;

(2) verifies that it is destined to the local host;

(3) processes options;

(4) reassembles the datagram if necessary; and

(5) passes the encapsulated message to the appropriate transport-layer protocol module.

For outgoing datagrams, the IP layer:

(1) sets any fields not set by the transport layer;

(2) selects the correct first hop on the connected network (a process called "routing");

(3) fragments the datagram if necessary and if intentional fragmentation is implemented (see Section 3.3.3); and

(4) passes the packet(s) to the appropriate link-layer driver.

A host is said to be multihomed if it has multiple IP addresses. Multihoming introduces considerable confusion and complexity into the protocol suite, and it is an area in which the Internet architecture falls seriously short of solving all problems. There are two distinct problem areas in multihoming:

(1) Local multihoming -- the host itself is multihomed; or

(2) Remote multihoming -- the local host needs to communicate with a remote multihomed host.

At present, remote multihoming MUST be handled at the application layer, as discussed in the companion RFC [INTRO:1]. A host MAY support local multihoming, which is discussed in this document, and in particular in Section 3.3.4.

Any host that forwards datagrams generated by another host is acting as a gateway and MUST also meet the specifications laid out in the gateway requirements RFC [INTRO:2]. An Internet host that includes embedded gateway code MUST have a configuration switch to disable the gateway function, and this switch MUST default to the

non-gateway mode. In this mode, a datagram arriving through one interface will not be forwarded to another host or gateway (unless it is source-routed), regardless of whether the host is single-homed or multihomed. The host software MUST NOT automatically move into gateway mode if the host has more than one interface, as the operator of the machine may neither want to provide that service nor be competent to do so.

In the following, the action specified in certain cases is to "silently discard" a received datagram. This means that the datagram will be discarded without further processing and that the host will not send any ICMP error message (see Section 3.2.2) as a result. However, for diagnosis of problems a host SHOULD provide the capability of logging the error (see Section 1.2.3), including the contents of the silently-discarded datagram, and SHOULD record the event in a statistics counter.

DISCUSSION:
Silent discard of erroneous datagrams is generally intended to prevent "broadcast storms".

3.2 PROTOCOL WALK-THROUGH

3.2.1 Internet Protocol -- IP

3.2.1.1 Version Number: RFC-791 Section 3.1

A datagram whose version number is not 4 MUST be silently discarded.

3.2.1.2 Checksum: RFC-791 Section 3.1

A host MUST verify the IP header checksum on every received datagram and silently discard every datagram that has a bad checksum.

3.2.1.3 Addressing: RFC-791 Section 3.2

There are now five classes of IP addresses: Class A through Class E. Class D addresses are used for IP multicasting [IP:4], while Class E addresses are reserved for experimental use.

A multicast (Class D) address is a 28-bit logical address that stands for a group of hosts, and may be either permanent or transient. Permanent multicast addresses are allocated by the Internet Assigned Number Authority [INTRO:6], while transient addresses may be allocated

dynamically to transient groups. Group membership is determined dynamically using IGMP [IP:4].

We now summarize the important special cases for Class A, B, and C IP addresses, using the following notation for an IP address:

{ <Network-number>, <Host-number> }

or

{ <Network-number>, <Subnet-number>, <Host-number> }

and the notation "-1" for a field that contains all 1 bits. This notation is not intended to imply that the 1-bits in an address mask need be contiguous.

(a) { 0, 0 }

This host on this network. MUST NOT be sent, except as a source address as part of an initialization procedure by which the host learns its own IP address.

See also Section 3.3.6 for a non-standard use of {0,0}.

(b) { 0, <Host-number> }

Specified host on this network. It MUST NOT be sent, except as a source address as part of an initialization procedure by which the host learns its full IP address.

(c) { -1, -1 }

Limited broadcast. It MUST NOT be used as a source address.

A datagram with this destination address will be received by every host on the connected physical network but will not be forwarded outside that network.

(d) { <Network-number>, -1 }

Directed broadcast to the specified network. It MUST NOT be used as a source address.

(e) { <Network-number>, <Subnet-number>, -1 }

Directed broadcast to the specified subnet. It MUST NOT be used as a source address.

(f) { <Network-number>, -1, -1 }

Directed broadcast to all subnets of the specified subnetted network. It MUST NOT be used as a source address.

(g) { 127, <any> }

Internal host loopback address. Addresses of this form MUST NOT appear outside a host.

The <Network-number> is administratively assigned so that its value will be unique in the entire world.

IP addresses are not permitted to have the value 0 or -1 for any of the <Host-number>, <Network-number>, or <Subnet-number> fields (except in the special cases listed above). This implies that each of these fields will be at least two bits long.

For further discussion of broadcast addresses, see Section 3.3.6.

A host MUST support the subnet extensions to IP [IP:3]. As a result, there will be an address mask of the form: {-1, -1, 0} associated with each of the host's local IP addresses; see Sections 3.2.2.9 and 3.3.1.1.

When a host sends any datagram, the IP source address MUST be one of its own IP addresses (but not a broadcast or multicast address).

A host MUST silently discard an incoming datagram that is not destined for the host. An incoming datagram is destined for the host if the datagram's destination address field is:

(1) (one of) the host's IP address(es); or

(2) an IP broadcast address valid for the connected network; or

(3) the address for a multicast group of which the host is a member on the incoming physical interface.

For most purposes, a datagram addressed to a broadcast or multicast destination is processed as if it had been addressed to one of the host's IP addresses; we use the term "specific-destination address" for the equivalent local IP

address of the host. The specific-destination address is defined to be the destination address in the IP header unless the header contains a broadcast or multicast address, in which case the specific-destination is an IP address assigned to the physical interface on which the datagram arrived.

A host MUST silently discard an incoming datagram containing an IP source address that is invalid by the rules of this section. This validation could be done in either the IP layer or by each protocol in the transport layer.

DISCUSSION:

A mis-addressed datagram might be caused by a link-layer broadcast of a unicast datagram or by a gateway or host that is confused or mis-configured.

An architectural goal for Internet hosts was to allow IP addresses to be featureless 32-bit numbers, avoiding algorithms that required a knowledge of the IP address format. Otherwise, any future change in the format or interpretation of IP addresses will require host software changes. However, validation of broadcast and multicast addresses violates this goal; a few other violations are described elsewhere in this document.

Implementers should be aware that applications depending upon the all-subnets directed broadcast address (f) may be unusable on some networks. All-subnets broadcast is not widely implemented in vendor gateways at present, and even when it is implemented, a particular network administration may disable it in the gateway configuration.

3.2.1.4 Fragmentation and Reassembly: RFC-791 Section 3.2

The Internet model requires that every host support reassembly. See Sections 3.3.2 and 3.3.3 for the requirements on fragmentation and reassembly.

3.2.1.5 Identification: RFC-791 Section 3.2

When sending an identical copy of an earlier datagram, a host MAY optionally retain the same Identification field in the copy.

DISCUSSION:

Some Internet protocol experts have maintained that when a host sends an identical copy of an earlier datagram, the new copy should contain the same Identification value as the original. There are two suggested advantages: (1) if the datagrams are fragmented and some of the fragments are lost, the receiver may be able to reconstruct a complete datagram from fragments of the original and the copies; (2) a congested gateway might use the IP Identification field (and Fragment Offset) to discard duplicate datagrams from the queue.

However, the observed patterns of datagram loss in the Internet do not favor the probability of retransmitted fragments filling reassembly gaps, while other mechanisms (e.g., TCP repacketizing upon retransmission) tend to prevent retransmission of an identical datagram [IP:9]. Therefore, we believe that retransmitting the same Identification field is not useful. Also, a connectionless transport protocol like UDP would require the cooperation of the application programs to retain the same Identification value in identical datagrams.

3.2.1.6 Type-of-Service: RFC-791 Section 3.2

The "Type-of-Service" byte in the IP header is divided into two sections: the Precedence field (high-order 3 bits), and a field that is customarily called "Type-of-Service" or "TOS" (low-order 5 bits). In this document, all references to "TOS" or the "TOS field" refer to the low-order 5 bits only.

The Precedence field is intended for Department of Defense applications of the Internet protocols. The use of non-zero values in this field is outside the scope of this document and the IP standard specification. Vendors should consult the Defense Communication Agency (DCA) for guidance on the IP Precedence field and its implications for other protocol layers. However, vendors should note that the use of precedence will most likely require that its value be passed between protocol layers in just the same way as the TOS field is passed.

The IP layer MUST provide a means for the transport layer to set the TOS field of every datagram that is sent; the default is all zero bits. The IP layer SHOULD pass received

TOS values up to the transport layer.

The particular link-layer mappings of TOS contained in RFC-795 SHOULD NOT be implemented.

DISCUSSION:

While the TOS field has been little used in the past, it is expected to play an increasing role in the near future. The TOS field is expected to be used to control two aspects of gateway operations: routing and queueing algorithms. See Section 2 of [INTRO:1] for the requirements on application programs to specify TOS values.

The TOS field may also be mapped into link-layer service selectors. This has been applied to provide effective sharing of serial lines by different classes of TCP traffic, for example. However, the mappings suggested in RFC-795 for networks that were included in the Internet as of 1981 are now obsolete.

3.2.1.7 Time-to-Live: RFC-791 Section 3.2

A host MUST NOT send a datagram with a Time-to-Live (TTL) value of zero.

A host MUST NOT discard a datagram just because it was received with TTL less than 2.

The IP layer MUST provide a means for the transport layer to set the TTL field of every datagram that is sent. When a fixed TTL value is used, it MUST be configurable. The current suggested value will be published in the "Assigned Numbers" RFC.

DISCUSSION:

The TTL field has two functions: limit the lifetime of TCP segments (see RFC-793 [TCP:1], p. 28), and terminate Internet routing loops. Although TTL is a time in seconds, it also has some attributes of a hop-count, since each gateway is required to reduce the TTL field by at least one.

The intent is that TTL expiration will cause a datagram to be discarded by a gateway but not by the destination host; however, hosts that act as gateways by forwarding datagrams must follow the gateway rules for TTL.

A higher-layer protocol may want to set the TTL in order to implement an "expanding scope" search for some Internet resource. This is used by some diagnostic tools, and is expected to be useful for locating the "nearest" server of a given class using IP multicasting, for example. A particular transport protocol may also want to specify its own TTL bound on maximum datagram lifetime.

A fixed value must be at least big enough for the Internet "diameter," i.e., the longest possible path. A reasonable value is about twice the diameter, to allow for continued Internet growth.

3.2.1.8 Options: RFC-791 Section 3.2

There MUST be a means for the transport layer to specify IP options to be included in transmitted IP datagrams (see Section 3.4).

All IP options (except NOP or END-OF-LIST) received in datagrams MUST be passed to the transport layer (or to ICMP processing when the datagram is an ICMP message). The IP and transport layer MUST each interpret those IP options that they understand and silently ignore the others.

Later sections of this document discuss specific IP option support required by each of ICMP, TCP, and UDP.

DISCUSSION:

Passing all received IP options to the transport layer is a deliberate "violation of strict layering" that is designed to ease the introduction of new transport-relevant IP options in the future. Each layer must pick out any options that are relevant to its own processing and ignore the rest. For this purpose, every IP option except NOP and END-OF-LIST will include a specification of its own length.

This document does not define the order in which a receiver must process multiple options in the same IP header. Hosts sending multiple options must be aware that this introduces an ambiguity in the meaning of certain options when combined with a source-route option.

IMPLEMENTATION:

The IP layer must not crash as the result of an option

length that is outside the possible range. For example, erroneous option lengths have been observed to put some IP implementations into infinite loops.

Here are the requirements for specific IP options:

(a) Security Option

Some environments require the Security option in every datagram; such a requirement is outside the scope of this document and the IP standard specification. Note, however, that the security options described in RFC-791 and RFC-1038 are obsolete. For DoD applications, vendors should consult [IP:8] for guidance.

(b) Stream Identifier Option

This option is obsolete; it SHOULD NOT be sent, and it MUST be silently ignored if received.

(c) Source Route Options

A host MUST support originating a source route and MUST be able to act as the final destination of a source route.

If host receives a datagram containing a completed source route (i.e., the pointer points beyond the last field), the datagram has reached its final destination; the option as received (the recorded route) MUST be passed up to the transport layer (or to ICMP message processing). This recorded route will be reversed and used to form a return source route for reply datagrams (see discussion of IP Options in Section 4). When a return source route is built, it MUST be correctly formed even if the recorded route included the source host (see case (B) in the discussion below).

An IP header containing more than one Source Route option MUST NOT be sent; the effect on routing of multiple Source Route options is implementation-specific.

Section 3.3.5 presents the rules for a host acting as an intermediate hop in a source route, i.e., forwarding

a source-routed datagram.

DISCUSSION:

If a source-routed datagram is fragmented, each fragment will contain a copy of the source route. Since the processing of IP options (including a source route) must precede reassembly, the original datagram will not be reassembled until the final destination is reached.

Suppose a source routed datagram is to be routed from host S to host D via gateways G1, G2, ... Gn. There was an ambiguity in the specification over whether the source route option in a datagram sent out by S should be (A) or (B):

(A): {>>G2, G3, ... Gn, D} <--- CORRECT

(B): {S, >>G2, G3, ... Gn, D} <---- WRONG

(where >> represents the pointer). If (A) is sent, the datagram received at D will contain the option: {G1, G2, ... Gn >>}, with S and D as the IP source and destination addresses. If (B) were sent, the datagram received at D would again contain S and D as the same IP source and destination addresses, but the option would be: {S, G1, ...Gn >>}; i.e., the originating host would be the first hop in the route.

(d) Record Route Option

Implementation of originating and processing the Record Route option is OPTIONAL.

(e) Timestamp Option

Implementation of originating and processing the Timestamp option is OPTIONAL. If it is implemented, the following rules apply:

- The originating host MUST record a timestamp in a Timestamp option whose Internet address fields are not pre-specified or whose first pre-specified address is the host's interface address.

- The destination host MUST (if possible) add the current timestamp to a Timestamp option before passing the option to the transport layer or to ICMP for processing.

- A timestamp value MUST follow the rules given in Section 3.2.2.8 for the ICMP Timestamp message.

3.2.2 Internet Control Message Protocol -- ICMP

ICMP messages are grouped into two classes.

*

ICMP error messages:

Destination Unreachable	(see Section 3.2.2.1)
Redirect	(see Section 3.2.2.2)
Source Quench	(see Section 3.2.2.3)
Time Exceeded	(see Section 3.2.2.4)
Parameter Problem	(see Section 3.2.2.5)

*

ICMP query messages:

Echo	(see Section 3.2.2.6)
Information	(see Section 3.2.2.7)
Timestamp	(see Section 3.2.2.8)
Address Mask	(see Section 3.2.2.9)

If an ICMP message of unknown type is received, it MUST be silently discarded.

Every ICMP error message includes the Internet header and at least the first 8 data octets of the datagram that triggered the error; more than 8 octets MAY be sent; this header and data MUST be unchanged from the received datagram.

In those cases where the Internet layer is required to pass an ICMP error message to the transport layer, the IP protocol number MUST be extracted from the original header and used to select the appropriate transport protocol entity to handle the error.

An ICMP error message SHOULD be sent with normal (i.e., zero) TOS bits.

An ICMP error message MUST NOT be sent as the result of receiving:

* an ICMP error message, or

* a datagram destined to an IP broadcast or IP multicast address, or

* a datagram sent as a link-layer broadcast, or

* a non-initial fragment, or

* a datagram whose source address does not define a single host -- e.g., a zero address, a loopback address, a broadcast address, a multicast address, or a Class E address.

NOTE: THESE RESTRICTIONS TAKE PRECEDENCE OVER ANY REQUIREMENT ELSEWHERE IN THIS DOCUMENT FOR SENDING ICMP ERROR MESSAGES.

DISCUSSION:
These rules will prevent the "broadcast storms" that have resulted from hosts returning ICMP error messages in response to broadcast datagrams. For example, a broadcast UDP segment to a non-existent port could trigger a flood of ICMP Destination Unreachable datagrams from all machines that do not have a client for that destination port. On a large Ethernet, the resulting collisions can render the network useless for a second or more.

Every datagram that is broadcast on the connected network should have a valid IP broadcast address as its IP destination (see Section 3.3.6). However, some hosts violate this rule. To be certain to detect broadcast datagrams, therefore, hosts are required to check for a link-layer broadcast as well as an IP-layer broadcast address.

IMPLEMENTATION:
This requires that the link layer inform the IP layer when a link-layer broadcast datagram has been received; see Section 2.4.

3.2.2.1 Destination Unreachable: RFC-792

The following additional codes are hereby defined:

6 = destination network unknown

7 = destination host unknown

8 = source host isolated

9 = communication with destination network administratively prohibited

10 = communication with destination host administratively prohibited

11 = network unreachable for type of service

12 = host unreachable for type of service

A host SHOULD generate Destination Unreachable messages with code:

2 (Protocol Unreachable), when the designated transport protocol is not supported; or

3 (Port Unreachable), when the designated transport protocol (e.g., UDP) is unable to demultiplex the datagram but has no protocol mechanism to inform the sender.

A Destination Unreachable message that is received MUST be reported to the transport layer. The transport layer SHOULD use the information appropriately; for example, see Sections 4.1.3.3, 4.2.3.9, and 4.2.4 below. A transport protocol that has its own mechanism for notifying the sender that a port is unreachable (e.g., TCP, which sends RST segments) MUST nevertheless accept an ICMP Port Unreachable for the same purpose.

A Destination Unreachable message that is received with code 0 (Net), 1 (Host), or 5 (Bad Source Route) may result from a routing transient and MUST therefore be interpreted as only a hint, not proof, that the specified destination is unreachable [IP:11]. For example, it MUST NOT be used as proof of a dead gateway (see Section 3.3.1).

3.2.2.2 Redirect: RFC-792

A host SHOULD NOT send an ICMP Redirect message; Redirects are to be sent only by gateways.

A host receiving a Redirect message MUST update its routing information accordingly. Every host MUST be prepared to

accept both Host and Network Redirects and to process them as described in Section 3.3.1.2 below.

A Redirect message SHOULD be silently discarded if the new gateway address it specifies is not on the same connected (sub-) net through which the Redirect arrived [INTRO:2, Appendix A], or if the source of the Redirect is not the current first-hop gateway for the specified destination (see Section 3.3.1).

3.2.2.3 Source Quench: RFC-792

A host MAY send a Source Quench message if it is approaching, or has reached, the point at which it is forced to discard incoming datagrams due to a shortage of reassembly buffers or other resources. See Section 2.2.3 of [INTRO:2] for suggestions on when to send Source Quench.

If a Source Quench message is received, the IP layer MUST report it to the transport layer (or ICMP processing). In general, the transport or application layer SHOULD implement a mechanism to respond to Source Quench for any protocol that can send a sequence of datagrams to the same destination and which can reasonably be expected to maintain enough state information to make this feasible. See Section 4 for the handling of Source Quench by TCP and UDP.

DISCUSSION:

A Source Quench may be generated by the target host or by some gateway in the path of a datagram. The host receiving a Source Quench should throttle itself back for a period of time, then gradually increase the transmission rate again. The mechanism to respond to Source Quench may be in the transport layer (for connection-oriented protocols like TCP) or in the application layer (for protocols that are built on top of UDP).

A mechanism has been proposed [IP:14] to make the IP layer respond directly to Source Quench by controlling the rate at which datagrams are sent, however, this proposal is currently experimental and not currently recommended.

3.2.2.4 Time Exceeded: RFC-792

An incoming Time Exceeded message MUST be passed to the transport layer.

DISCUSSION:

A gateway will send a Time Exceeded Code 0 (In Transit) message when it discards a datagram due to an expired TTL field. This indicates either a gateway routing loop or too small an initial TTL value.

A host may receive a Time Exceeded Code 1 (Reassembly Timeout) message from a destination host that has timed out and discarded an incomplete datagram; see Section 3.3.2 below. In the future, receipt of this message might be part of some "MTU discovery" procedure, to discover the maximum datagram size that can be sent on the path without fragmentation.

3.2.2.5 Parameter Problem: RFC-792

A host SHOULD generate Parameter Problem messages. An incoming Parameter Problem message MUST be passed to the transport layer, and it MAY be reported to the user.

DISCUSSION:

The ICMP Parameter Problem message is sent to the source host for any problem not specifically covered by another ICMP message. Receipt of a Parameter Problem message generally indicates some local or remote implementation error.

A new variant on the Parameter Problem message is hereby defined:

Code 1 = required option is missing.

DISCUSSION:

This variant is currently in use in the military community for a missing security option.

3.2.2.6 Echo Request/Reply: RFC-792

Every host MUST implement an ICMP Echo server function that receives Echo Requests and sends corresponding Echo Replies. A host SHOULD also implement an application-layer interface for sending an Echo Request and receiving an Echo Reply, for diagnostic purposes.

An ICMP Echo Request destined to an IP broadcast or IP multicast address MAY be silently discarded.

DISCUSSION:
This neutral provision results from a passionate debate between those who feel that ICMP Echo to a broadcast address provides a valuable diagnostic capability and those who feel that misuse of this feature can too easily create packet storms.

The IP source address in an ICMP Echo Reply MUST be the same as the specific-destination address (defined in Section 3.2.1.3) of the corresponding ICMP Echo Request message.

Data received in an ICMP Echo Request MUST be entirely included in the resulting Echo Reply. However, if sending the Echo Reply requires intentional fragmentation that is not implemented, the datagram MUST be truncated to maximum transmission size (see Section 3.3.3) and sent.

Echo Reply messages MUST be passed to the ICMP user interface, unless the corresponding Echo Request originated in the IP layer.

If a Record Route and/or Time Stamp option is received in an ICMP Echo Request, this option (these options) SHOULD be updated to include the current host and included in the IP header of the Echo Reply message, without "truncation". Thus, the recorded route will be for the entire round trip.

If a Source Route option is received in an ICMP Echo Request, the return route MUST be reversed and used as a Source Route option for the Echo Reply message.

3.2.2.7 Information Request/Reply: RFC-792

A host SHOULD NOT implement these messages.

DISCUSSION:
The Information Request/Reply pair was intended to support self-configuring systems such as diskless workstations, to allow them to discover their IP network numbers at boot time. However, the RARP and BOOTP protocols provide better mechanisms for a host to discover its own IP address.

3.2.2.8 Timestamp and Timestamp Reply: RFC-792

A host MAY implement Timestamp and Timestamp Reply. If they are implemented, the following rules MUST be followed.

- The ICMP Timestamp server function returns a Timestamp Reply to every Timestamp message that is received. If this function is implemented, it SHOULD be designed for minimum variability in delay (e.g., implemented in the kernel to avoid delay in scheduling a user process).

The following cases for Timestamp are to be handled according to the corresponding rules for ICMP Echo:

- An ICMP Timestamp Request message to an IP broadcast or IP multicast address MAY be silently discarded.

- The IP source address in an ICMP Timestamp Reply MUST be the same as the specific-destination address of the corresponding Timestamp Request message.

- If a Source-route option is received in an ICMP Echo Request, the return route MUST be reversed and used as a Source Route option for the Timestamp Reply message.

- If a Record Route and/or Timestamp option is received in a Timestamp Request, this (these) option(s) SHOULD be updated to include the current host and included in the IP header of the Timestamp Reply message.

- Incoming Timestamp Reply messages MUST be passed up to the ICMP user interface.

The preferred form for a timestamp value (the "standard value") is in units of milliseconds since midnight Universal Time. However, it may be difficult to provide this value with millisecond resolution. For example, many systems use clocks that update only at line frequency, 50 or 60 times per second. Therefore, some latitude is allowed in a "standard value":

(a) A "standard value" MUST be updated at least 15 times per second (i.e., at most the six low-order bits of the value may be undefined).

(b) The accuracy of a "standard value" MUST approximate that of operator-set CPU clocks, i.e., correct within a few minutes.

3.2.2.9 Address Mask Request/Reply: RFC-950

A host MUST support the first, and MAY implement all three, of the following methods for determining the address mask(s) corresponding to its IP address(es):

(1) static configuration information;

(2) obtaining the address mask(s) dynamically as a side-effect of the system initialization process (see [INTRO:1]); and

(3) sending ICMP Address Mask Request(s) and receiving ICMP Address Mask Reply(s).

The choice of method to be used in a particular host MUST be configurable.

When method (3), the use of Address Mask messages, is enabled, then:

(a) When it initializes, the host MUST broadcast an Address Mask Request message on the connected network corresponding to the IP address. It MUST retransmit this message a small number of times if it does not receive an immediate Address Mask Reply.

(b) Until it has received an Address Mask Reply, the host SHOULD assume a mask appropriate for the address class of the IP address, i.e., assume that the connected network is not subnetted.

(c) The first Address Mask Reply message received MUST be used to set the address mask corresponding to the particular local IP address. This is true even if the first Address Mask Reply message is "unsolicited", in which case it will have been broadcast and may arrive after the host has ceased to retransmit Address Mask Requests. Once the mask has been set by an Address Mask Reply, later Address Mask Reply messages MUST be (silently) ignored.

Conversely, if Address Mask messages are disabled, then no ICMP Address Mask Requests will be sent, and any ICMP Address Mask Replies received for that local IP address MUST be (silently) ignored.

A host SHOULD make some reasonableness check on any address

mask it installs; see IMPLEMENTATION section below.

A system MUST NOT send an Address Mask Reply unless it is an authoritative agent for address masks. An authoritative agent may be a host or a gateway, but it MUST be explicitly configured as a address mask agent. Receiving an address mask via an Address Mask Reply does not give the receiver authority and MUST NOT be used as the basis for issuing Address Mask Replies.

With a statically configured address mask, there SHOULD be an additional configuration flag that determines whether the host is to act as an authoritative agent for this mask, i.e., whether it will answer Address Mask Request messages using this mask.

If it is configured as an agent, the host MUST broadcast an Address Mask Reply for the mask on the appropriate interface when it initializes.

See "System Initialization" in [INTRO:1] for more information about the use of Address Mask Request/Reply messages.

DISCUSSION

Hosts that casually send Address Mask Replies with invalid address masks have often been a serious nuisance. To prevent this, Address Mask Replies ought to be sent only by authoritative agents that have been selected by explicit administrative action.

When an authoritative agent receives an Address Mask Request message, it will send a unicast Address Mask Reply to the source IP address. If the network part of this address is zero (see (a) and (b) in 3.2.1.3), the Reply will be broadcast.

Getting no reply to its Address Mask Request messages, a host will assume there is no agent and use an unsubnetted mask, but the agent may be only temporarily unreachable. An agent will broadcast an unsolicited Address Mask Reply whenever it initializes, in order to update the masks of all hosts that have initialized in the meantime.

IMPLEMENTATION:

The following reasonableness check on an address mask is suggested: the mask is not all 1 bits, and it is

either zero or else the 8 highest-order bits are on.

3.2.3 Internet Group Management Protocol IGMP

IGMP [IP:4] is a protocol used between hosts and gateways on a single network to establish hosts' membership in particular multicast groups. The gateways use this information, in conjunction with a multicast routing protocol, to support IP multicasting across the Internet.

At this time, implementation of IGMP is OPTIONAL; see Section 3.3.7 for more information. Without IGMP, a host can still participate in multicasting local to its connected networks.

3.3 SPECIFIC ISSUES

3.3.1 Routing Outbound Datagrams

The IP layer chooses the correct next hop for each datagram it sends. If the destination is on a connected network, the datagram is sent directly to the destination host; otherwise, it has to be routed to a gateway on a connected network.

3.3.1.1 Local/Remote Decision

To decide if the destination is on a connected network, the following algorithm MUST be used [see IP:3]:

(a) The address mask (particular to a local IP address for a multihomed host) is a 32-bit mask that selects the network number and subnet number fields of the corresponding IP address.

(b) If the IP destination address bits extracted by the address mask match the IP source address bits extracted by the same mask, then the destination is on the corresponding connected network, and the datagram is to be transmitted directly to the destination host.

(c) If not, then the destination is accessible only through a gateway. Selection of a gateway is described below (3.3.1.2).

A special-case destination address is handled as follows:

* For a limited broadcast or a multicast address, simply pass the datagram to the link layer for the appropriate interface.

* For a (network or subnet) directed broadcast, the datagram can use the standard routing algorithms.

The host IP layer MUST operate correctly in a minimal network environment, and in particular, when there are no gateways. For example, if the IP layer of a host insists on finding at least one gateway to initialize, the host will be unable to operate on a single isolated broadcast net.

3.3.1.2 Gateway Selection

To efficiently route a series of datagrams to the same destination, the source host MUST keep a "route cache" of mappings to next-hop gateways. A host uses the following basic algorithm on this cache to route a datagram; this algorithm is designed to put the primary routing burden on the gateways [IP:11].

(a) If the route cache contains no information for a particular destination, the host chooses a "default" gateway and sends the datagram to it. It also builds a corresponding Route Cache entry.

(b) If that gateway is not the best next hop to the destination, the gateway will forward the datagram to the best next-hop gateway and return an ICMP Redirect message to the source host.

(c) When it receives a Redirect, the host updates the next-hop gateway in the appropriate route cache entry, so later datagrams to the same destination will go directly to the best gateway.

Since the subnet mask appropriate to the destination address is generally not known, a Network Redirect message SHOULD be treated identically to a Host Redirect message; i.e., the cache entry for the destination host (only) would be updated (or created, if an entry for that host did not exist) for the new gateway.

DISCUSSION:
This recommendation is to protect against gateways that erroneously send Network Redirects for a subnetted network, in violation of the gateway requirements [INTRO:2].

When there is no route cache entry for the destination host address (and the destination is not on the connected

network), the IP layer MUST pick a gateway from its list of "default" gateways. The IP layer MUST support multiple default gateways.

As an extra feature, a host IP layer MAY implement a table of "static routes". Each such static route MAY include a flag specifying whether it may be overridden by ICMP Redirects.

DISCUSSION:

A host generally needs to know at least one default gateway to get started. This information can be obtained from a configuration file or else from the host startup sequence, e.g., the BOOTP protocol (see [INTRO:1]).

It has been suggested that a host can augment its list of default gateways by recording any new gateways it learns about. For example, it can record every gateway to which it is ever redirected. Such a feature, while possibly useful in some circumstances, may cause problems in other cases (e.g., gateways are not all equal), and it is not recommended.

A static route is typically a particular preset mapping from destination host or network into a particular next-hop gateway; it might also depend on the Type-of-Service (see next section). Static routes would be set up by system administrators to override the normal automatic routing mechanism, to handle exceptional situations. However, any static routing information is a potential source of failure as configurations change or equipment fails.

3.3.1.3 Route Cache

Each route cache entry needs to include the following fields:

(1) Local IP address (for a multihomed host)

(2) Destination IP address

(3) Type(s)-of-Service

(4) Next-hop gateway IP address

Field (2) MAY be the full IP address of the destination

host, or only the destination network number. Field (3), the TOS, SHOULD be included.

See Section 3.3.4.2 for a discussion of the implications of multihoming for the lookup procedure in this cache.

DISCUSSION:

Including the Type-of-Service field in the route cache and considering it in the host route algorithm will provide the necessary mechanism for the future when Type-of-Service routing is commonly used in the Internet. See Section 3.2.1.6.

Each route cache entry defines the endpoints of an Internet path. Although the connecting path may change dynamically in an arbitrary way, the transmission characteristics of the path tend to remain approximately constant over a time period longer than a single typical host-host transport connection. Therefore, a route cache entry is a natural place to cache data on the properties of the path. Examples of such properties might be the maximum unfragmented datagram size (see Section 3.3.3), or the average round-trip delay measured by a transport protocol. This data will generally be both gathered and used by a higher layer protocol, e.g., by TCP, or by an application using UDP. Experiments are currently in progress on caching path properties in this manner.

There is no consensus on whether the route cache should be keyed on destination host addresses alone, or allow both host and network addresses. Those who favor the use of only host addresses argue that:

(1) As required in Section 3.3.1.2, Redirect messages will generally result in entries keyed on destination host addresses; the simplest and most general scheme would be to use host addresses always.

(2) The IP layer may not always know the address mask for a network address in a complex subnetted environment.

(3) The use of only host addresses allows the destination address to be used as a pure 32-bit number, which may allow the Internet architecture to be more easily extended in the future without

any change to the hosts.

The opposing view is that allowing a mixture of destination hosts and networks in the route cache:

(1) Saves memory space.

(2) Leads to a simpler data structure, easily combining the cache with the tables of default and static routes (see below).

(3) Provides a more useful place to cache path properties, as discussed earlier.

IMPLEMENTATION:

The cache needs to be large enough to include entries for the maximum number of destination hosts that may be in use at one time.

A route cache entry may also include control information used to choose an entry for replacement. This might take the form of a "recently used" bit, a use count, or a last-used timestamp, for example. It is recommended that it include the time of last modification of the entry, for diagnostic purposes.

An implementation may wish to reduce the overhead of scanning the route cache for every datagram to be transmitted. This may be accomplished with a hash table to speed the lookup, or by giving a connection-oriented transport protocol a "hint" or temporary handle on the appropriate cache entry, to be passed to the IP layer with each subsequent datagram.

Although we have described the route cache, the lists of default gateways, and a table of static routes as conceptually distinct, in practice they may be combined into a single "routing table" data structure.

3.3.1.4 Dead Gateway Detection

The IP layer MUST be able to detect the failure of a "next-hop" gateway that is listed in its route cache and to choose an alternate gateway (see Section 3.3.1.5).

Dead gateway detection is covered in some detail in RFC-816 [IP:11]. Experience to date has not produced a complete

algorithm which is totally satisfactory, though it has identified several forbidden paths and promising techniques.

* A particular gateway SHOULD NOT be used indefinitely in the absence of positive indications that it is functioning.

* Active probes such as "pinging" (i.e., using an ICMP Echo Request/Reply exchange) are expensive and scale poorly. In particular, hosts MUST NOT actively check the status of a first-hop gateway by simply pinging the gateway continuously.

* Even when it is the only effective way to verify a gateway's status, pinging MUST be used only when traffic is being sent to the gateway and when there is no other positive indication to suggest that the gateway is functioning.

* To avoid pinging, the layers above and/or below the Internet layer SHOULD be able to give "advice" on the status of route cache entries when either positive (gateway OK) or negative (gateway dead) information is available.

DISCUSSION:

If an implementation does not include an adequate mechanism for detecting a dead gateway and re-routing, a gateway failure may cause datagrams to apparently vanish into a "black hole". This failure can be extremely confusing for users and difficult for network personnel to debug.

The dead-gateway detection mechanism must not cause unacceptable load on the host, on connected networks, or on first-hop gateway(s). The exact constraints on the timeliness of dead gateway detection and on acceptable load may vary somewhat depending on the nature of the host's mission, but a host generally needs to detect a failed first-hop gateway quickly enough that transport-layer connections will not break before an alternate gateway can be selected.

Passing advice from other layers of the protocol stack complicates the interfaces between the layers, but it is the preferred approach to dead gateway detection. Advice can come from almost any part of the IP/TCP

architecture, but it is expected to come primarily from the transport and link layers. Here are some possible sources for gateway advice:

- TCP or any connection-oriented transport protocol should be able to give negative advice, e.g., triggered by excessive retransmissions.

- TCP may give positive advice when (new) data is acknowledged. Even though the route may be asymmetric, an ACK for new data proves that the acknowleged data must have been transmitted successfully.

- An ICMP Redirect message from a particular gateway should be used as positive advice about that gateway.

- Link-layer information that reliably detects and reports host failures (e.g., ARPANET Destination Dead messages) should be used as negative advice.

- Failure to ARP or to re-validate ARP mappings may be used as negative advice for the corresponding IP address.

- Packets arriving from a particular link-layer address are evidence that the system at this address is alive. However, turning this information into advice about gateways requires mapping the link-layer address into an IP address, and then checking that IP address against the gateways pointed to by the route cache. This is probably prohibitively inefficient.

Note that positive advice that is given for every datagram received may cause unacceptable overhead in the implementation.

While advice might be passed using required arguments in all interfaces to the IP layer, some transport and application layer protocols cannot deduce the correct advice. These interfaces must therefore allow a neutral value for advice, since either always-positive or always-negative advice leads to incorrect behavior.

There is another technique for dead gateway detection that has been commonly used but is not recommended.

This technique depends upon the host passively receiving ("wiretapping") the Interior Gateway Protocol (IGP) datagrams that the gateways are broadcasting to each other. This approach has the drawback that a host needs to recognize all the interior gateway protocols that gateways may use (see [INTRO:2]). In addition, it only works on a broadcast network.

At present, pinging (i.e., using ICMP Echo messages) is the mechanism for gateway probing when absolutely required. A successful ping guarantees that the addressed interface and its associated machine are up, but it does not guarantee that the machine is a gateway as opposed to a host. The normal inference is that if a Redirect or other evidence indicates that a machine was a gateway, successful pings will indicate that the machine is still up and hence still a gateway. However, since a host silently discards packets that a gateway would forward or redirect, this assumption could sometimes fail. To avoid this problem, a new ICMP message under development will ask "are you a gateway?"

IMPLEMENTATION:

The following specific algorithm has been suggested:

- Associate a "reroute timer" with each gateway pointed to by the route cache. Initialize the timer to a value Tr, which must be small enough to allow detection of a dead gateway before transport connections time out.

- Positive advice would reset the reroute timer to Tr. Negative advice would reduce or zero the reroute timer.

- Whenever the IP layer used a particular gateway to route a datagram, it would check the corresponding reroute timer. If the timer had expired (reached zero), the IP layer would send a ping to the gateway, followed immediately by the datagram.

- The ping (ICMP Echo) would be sent again if necessary, up to N times. If no ping reply was received in N tries, the gateway would be assumed to have failed, and a new first-hop gateway would be chosen for all cache entries pointing to the failed gateway.

Note that the size of Tr is inversely related to the amount of advice available. Tr should be large enough to insure that:

* Any pinging will be at a low level (e.g., <10%) of all packets sent to a gateway from the host, AND

* pinging is infrequent (e.g., every 3 minutes)

Since the recommended algorithm is concerned with the gateways pointed to by route cache entries, rather than the cache entries themselves, a two level data structure (perhaps coordinated with ARP or similar caches) may be desirable for implementing a route cache.

3.3.1.5 New Gateway Selection

If the failed gateway is not the current default, the IP layer can immediately switch to a default gateway. If it is the current default that failed, the IP layer MUST select a different default gateway (assuming more than one default is known) for the failed route and for establishing new routes.

DISCUSSION:

When a gateway does fail, the other gateways on the connected network will learn of the failure through some inter-gateway routing protocol. However, this will not happen instantaneously, since gateway routing protocols typically have a settling time of 30-60 seconds. If the host switches to an alternative gateway before the gateways have agreed on the failure, the new target gateway will probably forward the datagram to the failed gateway and send a Redirect back to the host pointing to the failed gateway (!). The result is likely to be a rapid oscillation in the contents of the host's route cache during the gateway settling period. It has been proposed that the dead-gateway logic should include some hysteresis mechanism to prevent such oscillations. However, experience has not shown any harm from such oscillations, since service cannot be restored to the host until the gateways' routing information does settle down.

IMPLEMENTATION:

One implementation technique for choosing a new default gateway is to simply round-robin among the default gateways in the host's list. Another is to rank the

gateways in priority order, and when the current default gateway is not the highest priority one, to "ping" the higher-priority gateways slowly to detect when they return to service. This pinging can be at a very low rate, e.g., 0.005 per second.

3.3.1.6 Initialization

The following information MUST be configurable:

(1) IP address(es).

(2) Address mask(s).

(3) A list of default gateways, with a preference level.

A manual method of entering this configuration data MUST be provided. In addition, a variety of methods can be used to determine this information dynamically; see the section on "Host Initialization" in [INTRO:1].

DISCUSSION:
Some host implementations use "wiretapping" of gateway protocols on a broadcast network to learn what gateways exist. A standard method for default gateway discovery is under development.

3.3.2 Reassembly

The IP layer MUST implement reassembly of IP datagrams.

We designate the largest datagram size that can be reassembled by EMTU_R ("Effective MTU to receive"); this is sometimes called the "reassembly buffer size". EMTU_R MUST be greater than or equal to 576, SHOULD be either configurable or indefinite, and SHOULD be greater than or equal to the MTU of the connected network(s).

DISCUSSION:
A fixed EMTU_R limit should not be built into the code because some application layer protocols require EMTU_R values larger than 576.

IMPLEMENTATION:
An implementation may use a contiguous reassembly buffer for each datagram, or it may use a more complex data structure that places no definite limit on the reassembled datagram size; in the latter case, EMTU_R is said to be

"indefinite".

Logically, reassembly is performed by simply copying each fragment into the packet buffer at the proper offset. Note that fragments may overlap if successive retransmissions use different packetizing but the same reassembly Id.

The tricky part of reassembly is the bookkeeping to determine when all bytes of the datagram have been reassembled. We recommend Clark's algorithm [IP:10] that requires no additional data space for the bookkeeping. However, note that, contrary to [IP:10], the first fragment header needs to be saved for inclusion in a possible ICMP Time Exceeded (Reassembly Timeout) message.

There MUST be a mechanism by which the transport layer can learn MMS_R, the maximum message size that can be received and reassembled in an IP datagram (see GET_MAXSIZES calls in Section 3.4). If EMTU_R is not indefinite, then the value of MMS_R is given by:

MMS_R = EMTU_R - 20

since 20 is the minimum size of an IP header.

There MUST be a reassembly timeout. The reassembly timeout value SHOULD be a fixed value, not set from the remaining TTL. It is recommended that the value lie between 60 seconds and 120 seconds. If this timeout expires, the partially-reassembled datagram MUST be discarded and an ICMP Time Exceeded message sent to the source host (if fragment zero has been received).

DISCUSSION:

The IP specification says that the reassembly timeout should be the remaining TTL from the IP header, but this does not work well because gateways generally treat TTL as a simple hop count rather than an elapsed time. If the reassembly timeout is too small, datagrams will be discarded unnecessarily, and communication may fail. The timeout needs to be at least as large as the typical maximum delay across the Internet. A realistic minimum reassembly timeout would be 60 seconds.

It has been suggested that a cache might be kept of round-trip times measured by transport protocols for various destinations, and that these values might be used to dynamically determine a reasonable reassembly timeout

value. Further investigation of this approach is required.

If the reassembly timeout is set too high, buffer resources in the receiving host will be tied up too long, and the MSL (Maximum Segment Lifetime) [TCP:1] will be larger than necessary. The MSL controls the maximum rate at which fragmented datagrams can be sent using distinct values of the 16-bit Ident field; a larger MSL lowers the maximum rate. The TCP specification [TCP:1] arbitrarily assumes a value of 2 minutes for MSL. This sets an upper limit on a reasonable reassembly timeout value.

3.3.3 Fragmentation

Optionally, the IP layer MAY implement a mechanism to fragment outgoing datagrams intentionally.

We designate by EMTU_S ("Effective MTU for sending") the maximum IP datagram size that may be sent, for a particular combination of IP source and destination addresses and perhaps TOS.

A host MUST implement a mechanism to allow the transport layer to learn MMS_S, the maximum transport-layer message size that may be sent for a given {source, destination, TOS} triplet (see GET_MAXSIZES call in Section 3.4). If no local fragmentation is performed, the value of MMS_S will be:

MMS_S = EMTU_S - <IP header size>

and EMTU_S must be less than or equal to the MTU of the network interface corresponding to the source address of the datagram. Note that <IP header size> in this equation will be 20, unless the IP reserves space to insert IP options for its own purposes in addition to any options inserted by the transport layer.

A host that does not implement local fragmentation MUST ensure that the transport layer (for TCP) or the application layer (for UDP) obtains MMS_S from the IP layer and does not send a datagram exceeding MMS_S in size.

It is generally desirable to avoid local fragmentation and to choose EMTU_S low enough to avoid fragmentation in any gateway along the path. In the absence of actual knowledge of the minimum MTU along the path, the IP layer SHOULD use EMTU_S <= 576 whenever the destination address is not on a connected network, and otherwise use the connected network's

MTU.

The MTU of each physical interface MUST be configurable.

A host IP layer implementation MAY have a configuration flag "All-Subnets-MTU", indicating that the MTU of the connected network is to be used for destinations on different subnets within the same network, but not for other networks. Thus, this flag causes the network class mask, rather than the subnet address mask, to be used to choose an EMTU_S. For a multihomed host, an "All-Subnets-MTU" flag is needed for each network interface.

DISCUSSION:

Picking the correct datagram size to use when sending data is a complex topic [IP:9].

(a) In general, no host is required to accept an IP datagram larger than 576 bytes (including header and data), so a host must not send a larger datagram without explicit knowledge or prior arrangement with the destination host. Thus, MMS_S is only an upper bound on the datagram size that a transport protocol may send; even when MMS_S exceeds 556, the transport layer must limit its messages to 556 bytes in the absence of other knowledge about the destination host.

(b) Some transport protocols (e.g., TCP) provide a way to explicitly inform the sender about the largest datagram the other end can receive and reassemble [IP:7]. There is no corresponding mechanism in the IP layer.

A transport protocol that assumes an EMTU_R larger than 576 (see Section 3.3.2), can send a datagram of this larger size to another host that implements the same protocol.

(c) Hosts should ideally limit their EMTU_S for a given destination to the minimum MTU of all the networks along the path, to avoid any fragmentation. IP fragmentation, while formally correct, can create a serious transport protocol performance problem, because loss of a single fragment means all the fragments in the segment must be retransmitted [IP:9].

Since nearly all networks in the Internet currently support an MTU of 576 or greater, we strongly recommend the use of 576 for datagrams sent to non-local networks.

It has been suggested that a host could determine the MTU over a given path by sending a zero-offset datagram fragment and waiting for the receiver to time out the reassembly (which cannot complete!) and return an ICMP Time Exceeded message. This message would include the largest remaining fragment header in its body. More direct mechanisms are being experimented with, but have not yet been adopted (see e.g., RFC-1063).

3.3.4 Local Multihoming

3.3.4.1 Introduction

A multihomed host has multiple IP addresses, which we may think of as "logical interfaces". These logical interfaces may be associated with one or more physical interfaces, and these physical interfaces may be connected to the same or different networks.

Here are some important cases of multihoming:

(a) Multiple Logical Networks

The Internet architects envisioned that each physical network would have a single unique IP network (or subnet) number. However, LAN administrators have sometimes found it useful to violate this assumption, operating a LAN with multiple logical networks per physical connected network.

If a host connected to such a physical network is configured to handle traffic for each of N different logical networks, then the host will have N logical interfaces. These could share a single physical interface, or might use N physical interfaces to the same network.

(b) Multiple Logical Hosts

When a host has multiple IP addresses that all have the same <Network-number> part (and the same <Subnet-number> part, if any), the logical interfaces are known as "logical hosts". These logical interfaces might share a single physical interface or might use separate

physical interfaces to the same physical network.

(c) Simple Multihoming

In this case, each logical interface is mapped into a separate physical interface and each physical interface is connected to a different physical network. The term "multihoming" was originally applied only to this case, but it is now applied more generally.

A host with embedded gateway functionality will typically fall into the simple multihoming case. Note, however, that a host may be simply multihomed without containing an embedded gateway, i.e., without forwarding datagrams from one connected network to another.

This case presents the most difficult routing problems. The choice of interface (i.e., the choice of first-hop network) may significantly affect performance or even reachability of remote parts of the Internet.

Finally, we note another possibility that is NOT multihoming: one logical interface may be bound to multiple physical interfaces, in order to increase the reliability or throughput between directly connected machines by providing alternative physical paths between them. For instance, two systems might be connected by multiple point-to-point links. We call this "link-layer multiplexing". With link-layer multiplexing, the protocols above the link layer are unaware that multiple physical interfaces are present; the link-layer device driver is responsible for multiplexing and routing packets across the physical interfaces.

In the Internet protocol architecture, a transport protocol instance ("entity") has no address of its own, but instead uses a single Internet Protocol (IP) address. This has implications for the IP, transport, and application layers, and for the interfaces between them. In particular, the application software may have to be aware of the multiple IP addresses of a multihomed host; in other cases, the choice can be made within the network software.

3.3.4.2 Multihoming Requirements

The following general rules apply to the selection of an IP source address for sending a datagram from a multihomed

host.

(1) If the datagram is sent in response to a received datagram, the source address for the response SHOULD be the specific-destination address of the request. See Sections 4.1.3.5 and 4.2.3.7 and the "General Issues" section of [INTRO:1] for more specific requirements on higher layers.

Otherwise, a source address must be selected.

(2) An application MUST be able to explicitly specify the source address for initiating a connection or a request.

(3) In the absence of such a specification, the networking software MUST choose a source address. Rules for this choice are described below.

There are two key requirement issues related to multihoming:

(A) A host MAY silently discard an incoming datagram whose destination address does not correspond to the physical interface through which it is received.

(B) A host MAY restrict itself to sending (non-source-routed) IP datagrams only through the physical interface that corresponds to the IP source address of the datagrams.

DISCUSSION:

Internet host implementors have used two different conceptual models for multihoming, briefly summarized in the following discussion. This document takes no stand on which model is preferred; each seems to have a place. This ambivalence is reflected in the issues (A) and (B) being optional.

- Strong ES Model

  The Strong ES (End System, i.e., host) model emphasizes the host/gateway (ES/IS) distinction, and would therefore substitute MUST for MAY in issues (A) and (B) above. It tends to model a multihomed host as a set of logical hosts within the same physical host.

With respect to (A), proponents of the Strong ES model note that automatic Internet routing mechanisms could not route a datagram to a physical interface that did not correspond to the destination address.

Under the Strong ES model, the route computation for an outgoing datagram is the mapping:

```
route(src IP addr, dest IP addr, TOS)
                                    -> gateway
```

Here the source address is included as a parameter in order to select a gateway that is directly reachable on the corresponding physical interface. Note that this model logically requires that in general there be at least one default gateway, and preferably multiple defaults, for each IP source address.

- Weak ES Model

This view de-emphasizes the ES/IS distinction, and would therefore substitute MUST NOT for MAY in issues (A) and (B). This model may be the more natural one for hosts that wiretap gateway routing protocols, and is necessary for hosts that have embedded gateway functionality.

The Weak ES Model may cause the Redirect mechanism to fail. If a datagram is sent out a physical interface that does not correspond to the destination address, the first-hop gateway will not realize when it needs to send a Redirect. On the other hand, if the host has embedded gateway functionality, then it has routing information without listening to Redirects.

In the Weak ES model, the route computation for an outgoing datagram is the mapping:

```
route(dest IP addr, TOS) -> gateway, interface
```

3.3.4.3 Choosing a Source Address

DISCUSSION:

When it sends an initial connection request (e.g., a TCP "SYN" segment) or a datagram service request (e.g., a UDP-based query), the transport layer on a multihomed host needs to know which source address to use. If the application does not specify it, the transport layer must ask the IP layer to perform the conceptual mapping:

```
GET_SRCADDR(remote IP addr, TOS)
                              -> local IP address
```

Here TOS is the Type-of-Service value (see Section 3.2.1.6), and the result is the desired source address. The following rules are suggested for implementing this mapping:

(a) If the remote Internet address lies on one of the (sub-) nets to which the host is directly connected, a corresponding source address may be chosen, unless the corresponding interface is known to be down.

(b) The route cache may be consulted, to see if there is an active route to the specified destination network through any network interface; if so, a local IP address corresponding to that interface may be chosen.

(c) The table of static routes, if any (see Section 3.3.1.2) may be similarly consulted.

(d) The default gateways may be consulted. If these gateways are assigned to different interfaces, the interface corresponding to the gateway with the highest preference may be chosen.

In the future, there may be a defined way for a multihomed host to ask the gateways on all connected networks for advice about the best network to use for a given destination.

IMPLEMENTATION:

It will be noted that this process is essentially the same as datagram routing (see Section 3.3.1), and therefore hosts may be able to combine the

implementation of the two functions.

3.3.5 Source Route Forwarding

Subject to restrictions given below, a host MAY be able to act as an intermediate hop in a source route, forwarding a source-routed datagram to the next specified hop.

However, in performing this gateway-like function, the host MUST obey all the relevant rules for a gateway forwarding source-routed datagrams [INTRO:2]. This includes the following specific provisions, which override the corresponding host provisions given earlier in this document:

(A) TTL (ref. Section 3.2.1.7)

The TTL field MUST be decremented and the datagram perhaps discarded as specified for a gateway in [INTRO:2].

(B) ICMP Destination Unreachable (ref. Section 3.2.2.1)

A host MUST be able to generate Destination Unreachable messages with the following codes:

4 (Fragmentation Required but DF Set) when a source-routed datagram cannot be fragmented to fit into the target network;

5 (Source Route Failed) when a source-routed datagram cannot be forwarded, e.g., because of a routing problem or because the next hop of a strict source route is not on a connected network.

(C) IP Source Address (ref. Section 3.2.1.3)

A source-routed datagram being forwarded MAY (and normally will) have a source address that is not one of the IP addresses of the forwarding host.

(D) Record Route Option (ref. Section 3.2.1.8d)

A host that is forwarding a source-routed datagram containing a Record Route option MUST update that option, if it has room.

(E) Timestamp Option (ref. Section 3.2.1.8e)

A host that is forwarding a source-routed datagram

containing a Timestamp Option MUST add the current timestamp to that option, according to the rules for this option.

To define the rules restricting host forwarding of source-routed datagrams, we use the term "local source-routing" if the next hop will be through the same physical interface through which the datagram arrived; otherwise, it is "non-local source-routing".

- o A host is permitted to perform local source-routing without restriction.
- o A host that supports non-local source-routing MUST have a configurable switch to disable forwarding, and this switch MUST default to disabled.
- o The host MUST satisfy all gateway requirements for configurable policy filters [INTRO:2] restricting non-local forwarding.

If a host receives a datagram with an incomplete source route but does not forward it for some reason, the host SHOULD return an ICMP Destination Unreachable (code 5, Source Route Failed) message, unless the datagram was itself an ICMP error message.

3.3.6 Broadcasts

Section 3.2.1.3 defined the four standard IP broadcast address forms:

Limited Broadcast: {-1, -1}

Directed Broadcast: {<Network-number>,-1}

Subnet Directed Broadcast:
{<Network-number>,<Subnet-number>,-1}

All-Subnets Directed Broadcast: {<Network-number>,-1,-1}

A host MUST recognize any of these forms in the destination address of an incoming datagram.

There is a class of hosts* that use non-standard broadcast address forms, substituting 0 for -1. All hosts SHOULD

*4.2BSD Unix and its derivatives, but not 4.3BSD.

recognize and accept any of these non-standard broadcast addresses as the destination address of an incoming datagram. A host MAY optionally have a configuration option to choose the 0 or the -1 form of broadcast address, for each physical interface, but this option SHOULD default to the standard (-1) form.

When a host sends a datagram to a link-layer broadcast address, the IP destination address MUST be a legal IP broadcast or IP multicast address.

A host SHOULD silently discard a datagram that is received via a link-layer broadcast (see Section 2.4) but does not specify an IP multicast or broadcast destination address.

Hosts SHOULD use the Limited Broadcast address to broadcast to a connected network.

DISCUSSION:

Using the Limited Broadcast address instead of a Directed Broadcast address may improve system robustness. Problems are often caused by machines that do not understand the plethora of broadcast addresses (see Section 3.2.1.3), or that may have different ideas about which broadcast addresses are in use. The prime example of the latter is machines that do not understand subnetting but are attached to a subnetted net. Sending a Subnet Broadcast for the connected network will confuse those machines, which will see it as a message to some other host.

There has been discussion on whether a datagram addressed to the Limited Broadcast address ought to be sent from all the interfaces of a multihomed host. This specification takes no stand on the issue.

3.3.7 IP Multicasting

A host SHOULD support local IP multicasting on all connected networks for which a mapping from Class D IP addresses to link-layer addresses has been specified (see below). Support for local IP multicasting includes sending multicast datagrams, joining multicast groups and receiving multicast datagrams, and leaving multicast groups. This implies support for all of [IP:4] except the IGMP protocol itself, which is OPTIONAL.

DISCUSSION:
IGMP provides gateways that are capable of multicast routing with the information required to support IP multicasting across multiple networks. At this time, multicast-routing gateways are in the experimental stage and are not widely available. For hosts that are not connected to networks with multicast-routing gateways or that do not need to receive multicast datagrams originating on other networks, IGMP serves no purpose and is therefore optional for now. However, the rest of [IP:4] is currently recommended for the purpose of providing IP-layer access to local network multicast addressing, as a preferable alternative to local broadcast addressing. It is expected that IGMP will become recommended at some future date, when multicast-routing gateways have become more widely available.

If IGMP is not implemented, a host SHOULD still join the "all-hosts" group (224.0.0.1) when the IP layer is initialized and remain a member for as long as the IP layer is active.

DISCUSSION:
Joining the "all-hosts" group will support strictly local uses of multicasting, e.g., a gateway discovery protocol, even if IGMP is not implemented.

The mapping of IP Class D addresses to local addresses is currently specified for the following types of networks:

- Ethernet/IEEE 802.3, as defined in [IP:4].
- Any network that supports broadcast but not multicast, addressing: all IP Class D addresses map to the local broadcast address.
- Any type of point-to-point link (e.g., SLIP or HDLC links): no mapping required. All IP multicast datagrams are sent as-is, inside the local framing.

Mappings for other types of networks will be specified in the future.

A host SHOULD provide a way for higher-layer protocols or applications to determine which of the host's connected network(s) support IP multicast addressing.

3.3.8 Error Reporting

Wherever practical, hosts MUST return ICMP error datagrams on detection of an error, except in those cases where returning an ICMP error message is specifically prohibited.

DISCUSSION:
A common phenomenon in datagram networks is the "black hole disease": datagrams are sent out, but nothing comes back. Without any error datagrams, it is difficult for the user to figure out what the problem is.

3.4 INTERNET/TRANSPORT LAYER INTERFACE

The interface between the IP layer and the transport layer MUST provide full access to all the mechanisms of the IP layer, including options, Type-of-Service, and Time-to-Live. The transport layer MUST either have mechanisms to set these interface parameters, or provide a path to pass them through from an application, or both.

DISCUSSION:
Applications are urged to make use of these mechanisms where applicable, even when the mechanisms are not currently effective in the Internet (e.g., TOS). This will allow these mechanisms to be immediately useful when they do become effective, without a large amount of retrofitting of host software.

We now describe a conceptual interface between the transport layer and the IP layer, as a set of procedure calls. This is an extension of the information in Section 3.3 of RFC-791 [IP:1].

* Send Datagram

```
SEND(src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt
    => result )
```

where the parameters are defined in RFC-791. Passing an Id parameter is optional; see Section 3.2.1.5.

* Receive Datagram

```
RECV(BufPTR, prot
    => result, src, dst, SpecDest, TOS, len, opt)
```

All the parameters are defined in RFC-791, except for:

SpecDest = specific-destination address of datagram
(defined in Section 3.2.1.3)

The result parameter dst contains the datagram's destination address. Since this may be a broadcast or multicast address, the SpecDest parameter (not shown in RFC-791) MUST be passed. The parameter opt contains all the IP options received in the datagram; these MUST also be passed to the transport layer.

* Select Source Address

GET_SRCADDR(remote, TOS) -> local

remote = remote IP address
TOS = Type-of-Service
local = local IP address

See Section 3.3.4.3.

* Find Maximum Datagram Sizes

GET_MAXSIZES(local, remote, TOS) -> MMS_R, MMS_S

MMS_R = maximum receive transport-message size.
MMS_S = maximum send transport-message size.
(local, remote, TOS defined above)

See Sections 3.3.2 and 3.3.3.

* Advice on Delivery Success

ADVISE_DELIVPROB(sense, local, remote, TOS)

Here the parameter sense is a 1-bit flag indicating whether positive or negative advice is being given; see the discussion in Section 3.3.1.4. The other parameters were defined earlier.

* Send ICMP Message

SEND_ICMP(src, dst, TOS, TTL, BufPTR, len, Id, DF, opt)
-> result

(Parameters defined in RFC-791).

Passing an Id parameter is optional; see Section 3.2.1.5. The transport layer MUST be able to send certain ICMP messages: Port Unreachable or any of the query-type messages. This function could be considered to be a special case of the SEND() call, of course; we describe it separately for clarity.

* Receive ICMP Message

RECV_ICMP(BufPTR) -> result, src, dst, len, opt

(Parameters defined in RFC-791).

The IP layer MUST pass certain ICMP messages up to the appropriate transport-layer routine. This function could be considered to be a special case of the RECV() call, of course; we describe it separately for clarity.

For an ICMP error message, the data that is passed up MUST include the original Internet header plus all the octets of the original message that are included in the ICMP message. This data will be used by the transport layer to locate the connection state information, if any.

In particular, the following ICMP messages are to be passed up:

- Destination Unreachable
- Source Quench
- Echo Reply (to ICMP user interface, unless the Echo Request originated in the IP layer)
- Timestamp Reply (to ICMP user interface)
- Time Exceeded

DISCUSSION:

In the future, there may be additions to this interface to pass path data (see Section 3.3.1.3) between the IP and transport layers.

3.5 INTERNET LAYER REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Implement IP and ICMP	3.1	x					
Handle remote multihoming in application layer	3.1	x					
Support local multihoming	3.1			x			
Meet gateway specs if forward datagrams	3.1	x					
Configuration switch for embedded gateway	3.1	x					1
Config switch default to non-gateway	3.1	x					1
Auto-config based on number of interfaces	3.1					x	1
Able to log discarded datagrams	3.1		x				
Record in counter	3.1		x				
Silently discard Version != 4	3.2.1.1	x					
Verify IP checksum, silently discard bad dgram	3.2.1.2	x					
Addressing:							
Subnet addressing (RFC-950)	3.2.1.3	x					
Src address must be host's own IP address	3.2.1.3	x					
Silently discard datagram with bad dest addr	3.2.1.3	x					
Silently discard datagram with bad src addr	3.2.1.3	x					
Support reassembly	3.2.1.4	x					
Retain same Id field in identical datagram	3.2.1.5			x			
TOS:							
Allow transport layer to set TOS	3.2.1.6	x					
Pass received TOS up to transport layer	3.2.1.6		x				
Use RFC-795 link-layer mappings for TOS	3.2.1.6				x		
TTL:							
Send packet with TTL of 0	3.2.1.7					x	
Discard received packets with TTL < 2	3.2.1.7					x	
Allow transport layer to set TTL	3.2.1.7	x					
Fixed TTL is configurable	3.2.1.7	x					
IP Options:							
Allow transport layer to send IP options	3.2.1.8	x					
Pass all IP options rcvd to higher layer	3.2.1.8	x					

IP layer silently ignore unknown options	3.2.1.8	x				
Security option	3.2.1.8a			x		
Send Stream Identifier option	3.2.1.8b				x	
Silently ignore Stream Identifer option	3.2.1.8b	x				
Record Route option	3.2.1.8d			x		
Timestamp option	3.2.1.8e			x		
Source Route Option:						
Originate & terminate Source Route options	3.2.1.8c	x				
Datagram with completed SR passed up to TL	3.2.1.8c	x				
Build correct (non-redundant) return route	3.2.1.8c	x				
Send multiple SR options in one header	3.2.1.8c					x
ICMP:						
Silently discard ICMP msg with unknown type	3.2.2		x			
Include more than 8 octets of orig datagram	3.2.2			x		
Included octets same as received	3.2.2		x			
Demux ICMP Error to transport protocol	3.2.2		x			
Send ICMP error message with TOS=0	3.2.2			x		
Send ICMP error message for:						
- ICMP error msg	3.2.2					x
- IP b'cast or IP m'cast	3.2.2					x
- Link-layer b'cast	3.2.2					x
- Non-initial fragment	3.2.2					x
- Datagram with non-unique src address	3.2.2					x
Return ICMP error msgs (when not prohibited)	3.3.8		x			
Dest Unreachable:						
Generate Dest Unreachable (code 2/3)	3.2.2.1			x		
Pass ICMP Dest Unreachable to higher layer	3.2.2.1	x				
Higher layer act on Dest Unreach	3.2.2.1			x		
Interpret Dest Unreach as only hint	3.2.2.1	x				
Redirect:						
Host send Redirect	3.2.2.2				x	
Update route cache when recv Redirect	3.2.2.2	x				
Handle both Host and Net Redirects	3.2.2.2	x				
Discard illegal Redirect	3.2.2.2			x		
Source Quench:						
Send Source Quench if buffering exceeded	3.2.2.3			x		
Pass Source Quench to higher layer	3.2.2.3	x				
Higher layer act on Source Quench	3.2.2.3			x		
Time Exceeded: pass to higher layer	3.2.2.4	x				
Parameter Problem:						
Send Parameter Problem messages	3.2.2.5			x		
Pass Parameter Problem to higher layer	3.2.2.5	x				
Report Parameter Problem to user	3.2.2.5				x	
ICMP Echo Request or Reply:						
Echo server and Echo client	3.2.2.6	x				

Echo client	3.2.2.6		x				
Discard Echo Request to broadcast address	3.2.2.6			x			
Discard Echo Request to multicast address	3.2.2.6			x			
Use specific-dest addr as Echo Reply src	3.2.2.6	x					
Send same data in Echo Reply	3.2.2.6	x					
Pass Echo Reply to higher layer	3.2.2.6	x					
Reflect Record Route, Time Stamp options	3.2.2.6		x				
Reverse and reflect Source Route option	3.2.2.6	x					
ICMP Information Request or Reply:	3.2.2.7				x		
ICMP Timestamp and Timestamp Reply:	3.2.2.8			x			
Minimize delay variability	3.2.2.8		x				1
Silently discard b'cast Timestamp	3.2.2.8			x			1
Silently discard m'cast Timestamp	3.2.2.8			x			1
Use specific-dest addr as TS Reply src	3.2.2.8	x					1
Reflect Record Route, Time Stamp options	3.2.2.6		x				1
Reverse and reflect Source Route option	3.2.2.8	x					1
Pass Timestamp Reply to higher layer	3.2.2.8	x					1
Obey rules for "standard value"	3.2.2.8	x					1
ICMP Address Mask Request and Reply:							
Addr Mask source configurable	3.2.2.9	x					
Support static configuration of addr mask	3.2.2.9	x					
Get addr mask dynamically during booting	3.2.2.9			x			
Get addr via ICMP Addr Mask Request/Reply	3.2.2.9			x			
Retransmit Addr Mask Req if no Reply	3.2.2.9	x					3
Assume default mask if no Reply	3.2.2.9		x				3
Update address mask from first Reply only	3.2.2.9	x					3
Reasonableness check on Addr Mask	3.2.2.9		x				
Send unauthorized Addr Mask Reply msgs	3.2.2.9					x	
Explicitly configured to be agent	3.2.2.9	x					
Static config=> Addr-Mask-Authoritative flag	3.2.2.9		x				
Broadcast Addr Mask Reply when init.	3.2.2.9	x					3
ROUTING OUTBOUND DATAGRAMS:							
Use address mask in local/remote decision	3.3.1.1	x					
Operate with no gateways on conn network	3.3.1.1	x					
Maintain "route cache" of next-hop gateways	3.3.1.2	x					
Treat Host and Net Redirect the same	3.3.1.2		x				
If no cache entry, use default gateway	3.3.1.2	x					
Support multiple default gateways	3.3.1.2	x					
Provide table of static routes	3.3.1.2			x			
Flag: route overridable by Redirects	3.3.1.2			x			
Key route cache on host, not net address	3.3.1.3			x			
Include TOS in route cache	3.3.1.3		x				
Able to detect failure of next-hop gateway	3.3.1.4	x					
Assume route is good forever	3.3.1.4				x		

Ping gateways continuously	3.3.1.4					x	
Ping only when traffic being sent	3.3.1.4	x					
Ping only when no positive indication	3.3.1.4	x					
Higher and lower layers give advice	3.3.1.4		x				
Switch from failed default g'way to another	3.3.1.5	x					
Manual method of entering config info	3.3.1.6	x					
REASSEMBLY and FRAGMENTATION:							
Able to reassemble incoming datagrams	3.3.2	x					
At least 576 byte datagrams	3.3.2	x					
EMTU_R configurable or indefinite	3.3.2		x				
Transport layer able to learn MMS_R	3.3.2	x					
Send ICMP Time Exceeded on reassembly timeout	3.3.2	x					
Fixed reassembly timeout value	3.3.2		x				
Pass MMS_S to higher layers	3.3.3	x					
Local fragmentation of outgoing packets	3.3.3			x			
Else don't send bigger than MMS_S	3.3.3	x					
Send max 576 to off-net destination	3.3.3		x				
All-Subnets-MTU configuration flag	3.3.3			x			
MULTIHOMING:							
Reply with same addr as spec-dest addr	3.3.4.2		x				
Allow application to choose local IP addr	3.3.4.2	x					
Silently discard d'gram in "wrong" interface	3.3.4.2			x			
Only send d'gram through "right" interface	3.3.4.2			x			4
SOURCE-ROUTE FORWARDING:							
Forward datagram with Source Route option	3.3.5			x			1
Obey corresponding gateway rules	3.3.5	x					1
Update TTL by gateway rules	3.3.5	x					1
Able to generate ICMP err code 4, 5	3.3.5	x					1
IP src addr not local host	3.3.5			x			1
Update Timestamp, Record Route options	3.3.5	x					1
Configurable switch for non-local SRing	3.3.5	x					1
Defaults to OFF	3.3.5	x					1
Satisfy gwy access rules for non-local SRing	3.3.5	x					1
If not forward, send Dest Unreach (cd 5)	3.3.5		x				2
BROADCAST:							
Broadcast addr as IP source addr	3.2.1.3					x	
Receive 0 or -1 broadcast formats OK	3.3.6		x				
Config'ble option to send 0 or -1 b'cast	3.3.6			x			
Default to -1 broadcast	3.3.6		x				
Recognize all broadcast address formats	3.3.6	x					
Use IP b'cast/m'cast addr in link-layer b'cast	3.3.6	x					
Silently discard link-layer-only b'cast dg's	3.3.6		x				
Use Limited Broadcast addr for connected net	3.3.6		x				

MULTICAST:						
Support local IP multicasting (RFC-1112)	3.3.7		x			
Support IGMP (RFC-1112)	3.3.7			x		
Join all-hosts group at startup	3.3.7		x			
Higher layers learn i'face m'cast capability	3.3.7		x			
INTERFACE:						
Allow transport layer to use all IP mechanisms	3.4	x				
Pass interface ident up to transport layer	3.4	x				
Pass all IP options up to transport layer	3.4	x				
Transport layer can send certain ICMP messages	3.4	x				
Pass spec'd ICMP messages up to transp. layer	3.4	x				
Include IP hdr+8 octets or more from orig.	3.4	x				
Able to leap tall buildings at a single bound	3.5		x			

Footnotes:

(1) Only if feature is implemented.

(2) This requirement is overruled if datagram is an ICMP error message.

(3) Only if feature is implemented and is configured "on".

(4) Unless has embedded gateway functionality or is source routed.

4. TRANSPORT PROTOCOLS

4.1 USER DATAGRAM PROTOCOL -- UDP

4.1.1 INTRODUCTION

The User Datagram Protocol UDP [UDP:1] offers only a minimal transport service -- non-guaranteed datagram delivery -- and gives applications direct access to the datagram service of the IP layer. UDP is used by applications that do not require the level of service of TCP or that wish to use communications services (e.g., multicast or broadcast delivery) not available from TCP.

UDP is almost a null protocol; the only services it provides over IP are checksumming of data and multiplexing by port number. Therefore, an application program running over UDP must deal directly with end-to-end communication problems that a connection-oriented protocol would have handled -- e.g., retransmission for reliable delivery, packetization and reassembly, flow control, congestion avoidance, etc., when these are required. The fairly complex coupling between IP and TCP will be mirrored in the coupling between UDP and many applications using UDP.

4.1.2 PROTOCOL WALK-THROUGH

There are no known errors in the specification of UDP.

4.1.3 SPECIFIC ISSUES

4.1.3.1 Ports

UDP well-known ports follow the same rules as TCP well-known ports; see Section 4.2.2.1 below.

If a datagram arrives addressed to a UDP port for which there is no pending LISTEN call, UDP SHOULD send an ICMP Port Unreachable message.

4.1.3.2 IP Options

UDP MUST pass any IP option that it receives from the IP layer transparently to the application layer.

An application MUST be able to specify IP options to be sent in its UDP datagrams, and UDP MUST pass these options to the IP layer.

DISCUSSION:
At present, the only options that need be passed through UDP are Source Route, Record Route, and Time Stamp. However, new options may be defined in the future, and UDP need not and should not make any assumptions about the format or content of options it passes to or from the application; an exception to this might be an IP-layer security option.

An application based on UDP will need to obtain a source route from a request datagram and supply a reversed route for sending the corresponding reply.

4.1.3.3 ICMP Messages

UDP MUST pass to the application layer all ICMP error messages that it receives from the IP layer. Conceptually at least, this may be accomplished with an upcall to the ERROR_REPORT routine (see Section 4.2.4.1).

DISCUSSION:
Note that ICMP error messages resulting from sending a UDP datagram are received asynchronously. A UDP-based application that wants to receive ICMP error messages is responsible for maintaining the state necessary to demultiplex these messages when they arrive; for example, the application may keep a pending receive operation for this purpose. The application is also responsible to avoid confusion from a delayed ICMP error message resulting from an earlier use of the same port(s).

4.1.3.4 UDP Checksums

A host MUST implement the facility to generate and validate UDP checksums. An application MAY optionally be able to control whether a UDP checksum will be generated, but it MUST default to checksumming on.

If a UDP datagram is received with a checksum that is non-zero and invalid, UDP MUST silently discard the datagram. An application MAY optionally be able to control whether UDP datagrams without checksums should be discarded or passed to the application.

DISCUSSION:
Some applications that normally run only across local area networks have chosen to turn off UDP checksums for

efficiency. As a result, numerous cases of undetected errors have been reported. The advisability of ever turning off UDP checksumming is very controversial.

IMPLEMENTATION:
There is a common implementation error in UDP checksums. Unlike the TCP checksum, the UDP checksum is optional; the value zero is transmitted in the checksum field of a UDP header to indicate the absence of a checksum. If the transmitter really calculates a UDP checksum of zero, it must transmit the checksum as all 1's (65535). No special action is required at the receiver, since zero and 65535 are equivalent in 1's complement arithmetic.

4.1.3.5 UDP Multihoming

When a UDP datagram is received, its specific-destination address MUST be passed up to the application layer.

An application program MUST be able to specify the IP source address to be used for sending a UDP datagram or to leave it unspecified (in which case the networking software will choose an appropriate source address). There SHOULD be a way to communicate the chosen source address up to the application layer (e.g, so that the application can later receive a reply datagram only from the corresponding interface).

DISCUSSION:
A request/response application that uses UDP should use a source address for the response that is the same as the specific destination address of the request. See the "General Issues" section of [INTRO:1].

4.1.3.6 Invalid Addresses

A UDP datagram received with an invalid IP source address (e.g., a broadcast or multicast address) must be discarded by UDP or by the IP layer (see Section 3.2.1.3).

When a host sends a UDP datagram, the source address MUST be (one of) the IP address(es) of the host.

4.1.4 UDP/APPLICATION LAYER INTERFACE

The application interface to UDP MUST provide the full services of the IP/transport interface described in Section 3.4 of this

document. Thus, an application using UDP needs the functions of the GET_SRCADDR(), GET_MAXSIZES(), ADVISE_DELIVPROB(), and RECV_ICMP() calls described in Section 3.4. For example, GET_MAXSIZES() can be used to learn the effective maximum UDP maximum datagram size for a particular {interface,remote host,TOS} triplet.

An application-layer program MUST be able to set the TTL and TOS values as well as IP options for sending a UDP datagram, and these values must be passed transparently to the IP layer. UDP MAY pass the received TOS up to the application layer.

4.1.5 UDP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
UDP							
UDP send Port Unreachable	4.1.3.1		x				
IP Options in UDP							
- Pass rcv'd IP options to applic layer	4.1.3.2	x					
- Applic layer can specify IP options in Send	4.1.3.2	x					
- UDP passes IP options down to IP layer	4.1.3.2	x					
Pass ICMP msgs up to applic layer	4.1.3.3	x					
UDP checksums:							
- Able to generate/check checksum	4.1.3.4	x					
- Silently discard bad checksum	4.1.3.4	x					
- Sender Option to not generate checksum	4.1.3.4			x			
- Default is to checksum	4.1.3.4	x					
- Receiver Option to require checksum	4.1.3.4			x			
UDP Multihoming							
- Pass spec-dest addr to application	4.1.3.5	x					

- Applic layer can specify Local IP addr	4.1.3.5	x				
- Applic layer specify wild Local IP addr	4.1.3.5	x				
- Applic layer notified of Local IP addr used	4.1.3.5		x			
Bad IP src addr silently discarded by UDP/IP	4.1.3.6	x				
Only send valid IP source address	4.1.3.6	x				
UDP Application Interface Services						
Full IP interface of 3.4 for application	4.1.4	x				
- Able to spec TTL, TOS, IP opts when send dg	4.1.4	x				
- Pass received TOS up to applic layer	4.1.4			x		

4.2 TRANSMISSION CONTROL PROTOCOL -- TCP

4.2.1 INTRODUCTION

The Transmission Control Protocol TCP [TCP:1] is the primary virtual-circuit transport protocol for the Internet suite. TCP provides reliable, in-sequence delivery of a full-duplex stream of octets (8-bit bytes). TCP is used by those applications needing reliable, connection-oriented transport service, e.g., mail (SMTP), file transfer (FTP), and virtual terminal service (Telnet); requirements for these application-layer protocols are described in [INTRO:1].

4.2.2 PROTOCOL WALK-THROUGH

4.2.2.1 Well-Known Ports: RFC-793 Section 2.7

DISCUSSION:

TCP reserves port numbers in the range 0-255 for "well-known" ports, used to access services that are standardized across the Internet. The remainder of the port space can be freely allocated to application processes. Current well-known port definitions are listed in the RFC entitled "Assigned Numbers" [INTRO:6]. A prerequisite for defining a new well-known port is an RFC documenting the proposed service in enough detail to allow new implementations.

Some systems extend this notion by adding a third subdivision of the TCP port space: reserved ports, which are generally used for operating-system-specific services. For example, reserved ports might fall between 256 and some system-dependent upper limit. Some systems further choose to protect well-known and reserved ports by permitting only privileged users to open TCP connections with those port values. This is perfectly reasonable as long as the host does not assume that all hosts protect their low-numbered ports in this manner.

4.2.2.2 Use of Push: RFC-793 Section 2.8

When an application issues a series of SEND calls without setting the PUSH flag, the TCP MAY aggregate the data internally without sending it. Similarly, when a series of segments is received without the PSH bit, a TCP MAY queue the data internally without passing it to the receiving application.

The PSH bit is not a record marker and is independent of segment boundaries. The transmitter SHOULD collapse successive PSH bits when it packetizes data, to send the largest possible segment.

A TCP MAY implement PUSH flags on SEND calls. If PUSH flags are not implemented, then the sending TCP: (1) must not buffer data indefinitely, and (2) MUST set the PSH bit in the last buffered segment (i.e., when there is no more queued data to be sent).

The discussion in RFC-793 on pages 48, 50, and 74 erroneously implies that a received PSH flag must be passed to the application layer. Passing a received PSH flag to the application layer is now OPTIONAL.

An application program is logically required to set the PUSH flag in a SEND call whenever it needs to force delivery of the data to avoid a communication deadlock. However, a TCP SHOULD send a maximum-sized segment whenever possible, to improve performance (see Section 4.2.3.4).

DISCUSSION:

When the PUSH flag is not implemented on SEND calls, i.e., when the application/TCP interface uses a pure streaming model, responsibility for aggregating any tiny data fragments to form reasonable sized segments is partially borne by the application layer.

Generally, an interactive application protocol must set the PUSH flag at least in the last SEND call in each command or response sequence. A bulk transfer protocol like FTP should set the PUSH flag on the last segment of a file or when necessary to prevent buffer deadlock.

At the receiver, the PSH bit forces buffered data to be delivered to the application (even if less than a full buffer has been received). Conversely, the lack of a PSH bit can be used to avoid unnecessary wakeup calls to the application process; this can be an important performance optimization for large timesharing hosts. Passing the PSH bit to the receiving application allows an analogous optimization within the application.

4.2.2.3 Window Size: RFC-793 Section 3.1

The window size MUST be treated as an unsigned number, or else large window sizes will appear like negative windows

and TCP will not work. It is RECOMMENDED that implementations reserve 32-bit fields for the send and receive window sizes in the connection record and do all window computations with 32 bits.

DISCUSSION:
It is known that the window field in the TCP header is too small for high-speed, long-delay paths. Experimental TCP options have been defined to extend the window size; see for example [TCP:11]. In anticipation of the adoption of such an extension, TCP implementors should treat windows as 32 bits.

4.2.2.4 Urgent Pointer: RFC-793 Section 3.1

The second sentence is in error: the urgent pointer points to the sequence number of the LAST octet (not LAST+1) in a sequence of urgent data. The description on page 56 (last sentence) is correct.

A TCP MUST support a sequence of urgent data of any length.

A TCP MUST inform the application layer asynchronously whenever it receives an Urgent pointer and there was previously no pending urgent data, or whenever the Urgent pointer advances in the data stream. There MUST be a way for the application to learn how much urgent data remains to be read from the connection, or at least to determine whether or not more urgent data remains to be read.

DISCUSSION:
Although the Urgent mechanism may be used for any application, it is normally used to send "interrupt"-type commands to a Telnet program (see "Using Telnet Synch Sequence" section in [INTRO:1]).

The asynchronous or "out-of-band" notification will allow the application to go into "urgent mode", reading data from the TCP connection. This allows control commands to be sent to an application whose normal input buffers are full of unprocessed data.

IMPLEMENTATION:
The generic ERROR-REPORT() upcall described in Section 4.2.4.1 is a possible mechanism for informing the application of the arrival of urgent data.

4.2.2.5 TCP Options: RFC-793 Section 3.1

A TCP MUST be able to receive a TCP option in any segment. A TCP MUST ignore without error any TCP option it does not implement, assuming that the option has a length field (all TCP options defined in the future will have length fields). TCP MUST be prepared to handle an illegal option length (e.g., zero) without crashing; a suggested procedure is to reset the connection and log the reason.

4.2.2.6 Maximum Segment Size Option: RFC-793 Section 3.1

TCP MUST implement both sending and receiving the Maximum Segment Size option [TCP:4].

TCP SHOULD send an MSS (Maximum Segment Size) option in every SYN segment when its receive MSS differs from the default 536, and MAY send it always.

If an MSS option is not received at connection setup, TCP MUST assume a default send MSS of 536 (576-40) [TCP:4].

The maximum size of a segment that TCP really sends, the "effective send MSS," MUST be the smaller of the send MSS (which reflects the available reassembly buffer size at the remote host) and the largest size permitted by the IP layer:

```
Eff.snd.MSS =

    min(SendMSS+20, MMS_S) - TCPhdrsize - IPoptionsize
```

where:

* SendMSS is the MSS value received from the remote host, or the default 536 if no MSS option is received.

* MMS_S is the maximum size for a transport-layer message that TCP may send.

* TCPhdrsize is the size of the TCP header; this is normally 20, but may be larger if TCP options are to be sent.

* IPoptionsize is the size of any IP options that TCP will pass to the IP layer with the current message.

The MSS value to be sent in an MSS option must be less than

or equal to:

MMS_R - 20

where MMS_R is the maximum size for a transport-layer message that can be received (and reassembled). TCP obtains MMS_R and MMS_S from the IP layer; see the generic call GET_MAXSIZES in Section 3.4.

DISCUSSION:

The choice of TCP segment size has a strong effect on performance. Larger segments increase throughput by amortizing header size and per-datagram processing overhead over more data bytes; however, if the packet is so large that it causes IP fragmentation, efficiency drops sharply if any fragments are lost [IP:9].

Some TCP implementations send an MSS option only if the destination host is on a non-connected network. However, in general the TCP layer may not have the appropriate information to make this decision, so it is preferable to leave to the IP layer the task of determining a suitable MTU for the Internet path. We therefore recommend that TCP always send the option (if not 536) and that the IP layer determine MMS_R as specified in 3.3.3 and 3.4. A proposed IP-layer mechanism to measure the MTU would then modify the IP layer without changing TCP.

4.2.2.7 TCP Checksum: RFC-793 Section 3.1

Unlike the UDP checksum (see Section 4.1.3.4), the TCP checksum is never optional. The sender MUST generate it and the receiver MUST check it.

4.2.2.8 TCP Connection State Diagram: RFC-793 Section 3.2, page 23

There are several problems with this diagram:

(a) The arrow from SYN-SENT to SYN-RCVD should be labeled with "snd SYN,ACK", to agree with the text on page 68 and with Figure 8.

(b) There could be an arrow from SYN-RCVD state to LISTEN state, conditioned on receiving a RST after a passive open (see text page 70).

(c) It is possible to go directly from FIN-WAIT-1 to the TIME-WAIT state (see page 75 of the spec).

4.2.2.9 Initial Sequence Number Selection: RFC-793 Section 3.3, page 27

A TCP MUST use the specified clock-driven selection of initial sequence numbers.

4.2.2.10 Simultaneous Open Attempts: RFC-793 Section 3.4, page 32

There is an error in Figure 8: the packet on line 7 should be identical to the packet on line 5.

A TCP MUST support simultaneous open attempts.

DISCUSSION:
It sometimes surprises implementors that if two applications attempt to simultaneously connect to each other, only one connection is generated instead of two. This was an intentional design decision; don't try to "fix" it.

4.2.2.11 Recovery from Old Duplicate SYN: RFC-793 Section 3.4, page 33

Note that a TCP implementation MUST keep track of whether a connection has reached SYN_RCVD state as the result of a passive OPEN or an active OPEN.

4.2.2.12 RST Segment: RFC-793 Section 3.4

A TCP SHOULD allow a received RST segment to include data.

DISCUSSION
It has been suggested that a RST segment could contain ASCII text that encoded and explained the cause of the RST. No standard has yet been established for such data.

4.2.2.13 Closing a Connection: RFC-793 Section 3.5

A TCP connection may terminate in two ways: (1) the normal TCP close sequence using a FIN handshake, and (2) an "abort" in which one or more RST segments are sent and the connection state is immediately discarded. If a TCP

connection is closed by the remote site, the local application MUST be informed whether it closed normally or was aborted.

The normal TCP close sequence delivers buffered data reliably in both directions. Since the two directions of a TCP connection are closed independently, it is possible for a connection to be "half closed," i.e., closed in only one direction, and a host is permitted to continue sending data in the open direction on a half-closed connection.

A host MAY implement a "half-duplex" TCP close sequence, so that an application that has called CLOSE cannot continue to read data from the connection. If such a host issues a CLOSE call while received data is still pending in TCP, or if new data is received after CLOSE is called, its TCP SHOULD send a RST to show that data was lost.

When a connection is closed actively, it MUST linger in TIME-WAIT state for a time 2xMSL (Maximum Segment Lifetime). However, it MAY accept a new SYN from the remote TCP to reopen the connection directly from TIME-WAIT state, if it:

(1) assigns its initial sequence number for the new connection to be larger than the largest sequence number it used on the previous connection incarnation, and

(2) returns to TIME-WAIT state if the SYN turns out to be an old duplicate.

DISCUSSION:

TCP's full-duplex data-preserving close is a feature that is not included in the analogous ISO transport protocol TP4.

Some systems have not implemented half-closed connections, presumably because they do not fit into the I/O model of their particular operating system. On these systems, once an application has called CLOSE, it can no longer read input data from the connection; this is referred to as a "half-duplex" TCP close sequence.

The graceful close algorithm of TCP requires that the connection state remain defined on (at least) one end of the connection, for a timeout period of 2xMSL, i.e., 4 minutes. During this period, the (remote socket,

local socket) pair that defines the connection is busy and cannot be reused. To shorten the time that a given port pair is tied up, some TCPs allow a new SYN to be accepted in TIME-WAIT state.

4.2.2.14 Data Communication: RFC-793 Section 3.7, page 40

Since RFC-793 was written, there has been extensive work on TCP algorithms to achieve efficient data communication. Later sections of the present document describe required and recommended TCP algorithms to determine when to send data (Section 4.2.3.4), when to send an acknowledgment (Section 4.2.3.2), and when to update the window (Section 4.2.3.3).

DISCUSSION:

One important performance issue is "Silly Window Syndrome" or "SWS" [TCP:5], a stable pattern of small incremental window movements resulting in extremely poor TCP performance. Algorithms to avoid SWS are described below for both the sending side (Section 4.2.3.4) and the receiving side (Section 4.2.3.3).

In brief, SWS is caused by the receiver advancing the right window edge whenever it has any new buffer space available to receive data and by the sender using any incremental window, no matter how small, to send more data [TCP:5]. The result can be a stable pattern of sending tiny data segments, even though both sender and receiver have a large total buffer space for the connection. SWS can only occur during the transmission of a large amount of data; if the connection goes quiescent, the problem will disappear. It is caused by typical straightforward implementation of window management, but the sender and receiver algorithms given below will avoid it.

Another important TCP performance issue is that some applications, especially remote login to character-at-a-time hosts, tend to send streams of one-octet data segments. To avoid deadlocks, every TCP SEND call from such applications must be "pushed", either explicitly by the application or else implicitly by TCP. The result may be a stream of TCP segments that contain one data octet each, which makes very inefficient use of the Internet and contributes to Internet congestion. The Nagle Algorithm described in Section 4.2.3.4 provides a simple and effective solution to this problem. It does have the effect of clumping

characters over Telnet connections; this may initially surprise users accustomed to single-character echo, but user acceptance has not been a problem.

Note that the Nagle algorithm and the send SWS avoidance algorithm play complementary roles in improving performance. The Nagle algorithm discourages sending tiny segments when the data to be sent increases in small increments, while the SWS avoidance algorithm discourages small segments resulting from the right window edge advancing in small increments.

A careless implementation can send two or more acknowledgment segments per data segment received. For example, suppose the receiver acknowledges every data segment immediately. When the application program subsequently consumes the data and increases the available receive buffer space again, the receiver may send a second acknowledgment segment to update the window at the sender. The extreme case occurs with single-character segments on TCP connections using the Telnet protocol for remote login service. Some implementations have been observed in which each incoming 1-character segment generates three return segments: (1) the acknowledgment, (2) a one byte increase in the window, and (3) the echoed character, respectively.

4.2.2.15 Retransmission Timeout: RFC-793 Section 3.7, page 41

The algorithm suggested in RFC-793 for calculating the retransmission timeout is now known to be inadequate; see Section 4.2.3.1 below.

Recent work by Jacobson [TCP:7] on Internet congestion and TCP retransmission stability has produced a transmission algorithm combining "slow start" with "congestion avoidance". A TCP MUST implement this algorithm.

If a retransmitted packet is identical to the original packet (which implies not only that the data boundaries have not changed, but also that the window and acknowledgment fields of the header have not changed), then the same IP Identification field MAY be used (see Section 3.2.1.5).

IMPLEMENTATION:
Some TCP implementors have chosen to "packetize" the data stream, i.e., to pick segment boundaries when

segments are originally sent and to queue these segments in a "retransmission queue" until they are acknowledged. Another design (which may be simpler) is to defer packetizing until each time data is transmitted or retransmitted, so there will be no segment retransmission queue.

In an implementation with a segment retransmission queue, TCP performance may be enhanced by repacketizing the segments awaiting acknowledgment when the first retransmission timeout occurs. That is, the outstanding segments that fitted would be combined into one maximum-sized segment, with a new IP Identification value. The TCP would then retain this combined segment in the retransmit queue until it was acknowledged. However, if the first two segments in the retransmission queue totalled more than one maximum-sized segment, the TCP would retransmit only the first segment using the original IP Identification field.

4.2.2.16 Managing the Window: RFC-793 Section 3.7, page 41

A TCP receiver SHOULD NOT shrink the window, i.e., move the right window edge to the left. However, a sending TCP MUST be robust against window shrinking, which may cause the "useable window" (see Section 4.2.3.4) to become negative.

If this happens, the sender SHOULD NOT send new data, but SHOULD retransmit normally the old unacknowledged data between SND.UNA and SND.UNA+SND.WND. The sender MAY also retransmit old data beyond SND.UNA+SND.WND, but SHOULD NOT time out the connection if data beyond the right window edge is not acknowledged. If the window shrinks to zero, the TCP MUST probe it in the standard way (see next Section).

DISCUSSION:

Many TCP implementations become confused if the window shrinks from the right after data has been sent into a larger window. Note that TCP has a heuristic to select the latest window update despite possible datagram reordering; as a result, it may ignore a window update with a smaller window than previously offered if neither the sequence number nor the acknowledgment number is increased.

4.2.2.17 Probing Zero Windows: RFC-793 Section 3.7, page 42

Probing of zero (offered) windows MUST be supported.

A TCP MAY keep its offered receive window closed indefinitely. As long as the receiving TCP continues to send acknowledgments in response to the probe segments, the sending TCP MUST allow the connection to stay open.

DISCUSSION:

It is extremely important to remember that ACK (acknowledgment) segments that contain no data are not reliably transmitted by TCP. If zero window probing is not supported, a connection may hang forever when an ACK segment that re-opens the window is lost.

The delay in opening a zero window generally occurs when the receiving application stops taking data from its TCP. For example, consider a printer daemon application, stopped because the printer ran out of paper.

The transmitting host SHOULD send the first zero-window probe when a zero window has existed for the retransmission timeout period (see Section 4.2.2.15), and SHOULD increase exponentially the interval between successive probes.

DISCUSSION:

This procedure minimizes delay if the zero-window condition is due to a lost ACK segment containing a window-opening update. Exponential backoff is recommended, possibly with some maximum interval not specified here. This procedure is similar to that of the retransmission algorithm, and it may be possible to combine the two procedures in the implementation.

4.2.2.18 Passive OPEN Calls: RFC-793 Section 3.8

Every passive OPEN call either creates a new connection record in LISTEN state, or it returns an error; it MUST NOT affect any previously created connection record.

A TCP that supports multiple concurrent users MUST provide an OPEN call that will functionally allow an application to LISTEN on a port while a connection block with the same local port is in SYN-SENT or SYN-RECEIVED state.

DISCUSSION:

Some applications (e.g., SMTP servers) may need to handle multiple connection attempts at about the same time. The probability of a connection attempt failing is reduced by giving the application some means of listening for a new connection at the same time that an earlier connection attempt is going through the three-way handshake.

IMPLEMENTATION:
Acceptable implementations of concurrent opens may permit multiple passive OPEN calls, or they may allow "cloning" of LISTEN-state connections from a single passive OPEN call.

4.2.2.19 Time to Live: RFC-793 Section 3.9, page 52

RFC-793 specified that TCP was to request the IP layer to send TCP segments with TTL = 60. This is obsolete; the TTL value used to send TCP segments MUST be configurable. See Section 3.2.1.7 for discussion.

4.2.2.20 Event Processing: RFC-793 Section 3.9

While it is not strictly required, a TCP SHOULD be capable of queueing out-of-order TCP segments. Change the "may" in the last sentence of the first paragraph on page 70 to "should".

DISCUSSION:
Some small-host implementations have omitted segment queueing because of limited buffer space. This omission may be expected to adversely affect TCP throughput, since loss of a single segment causes all later segments to appear to be "out of sequence".

In general, the processing of received segments MUST be implemented to aggregate ACK segments whenever possible. For example, if the TCP is processing a series of queued segments, it MUST process them all before sending any ACK segments.

Here are some detailed error corrections and notes on the Event Processing section of RFC-793.

(a) CLOSE Call, CLOSE-WAIT state, p. 61: enter LAST-ACK state, not CLOSING.

(b) LISTEN state, check for SYN (pp. 65, 66): With a SYN

bit, if the security/compartment or the precedence is wrong for the segment, a reset is sent. The wrong form of reset is shown in the text; it should be:

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

(c) SYN-SENT state, Check for SYN, p. 68: When the connection enters ESTABLISHED state, the following variables must be set:
SND.WND <- SEG.WND
SND.WL1 <- SEG.SEQ
SND.WL2 <- SEG.ACK

(d) Check security and precedence, p. 71: The first heading "ESTABLISHED STATE" should really be a list of all states other than SYN-RECEIVED: ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, and TIME-WAIT.

(e) Check SYN bit, p. 71: "In SYN-RECEIVED state and if the connection was initiated with a passive OPEN, then return this connection to the LISTEN state and return. Otherwise...".

(f) Check ACK field, SYN-RECEIVED state, p. 72: When the connection enters ESTABLISHED state, the variables listed in (c) must be set.

(g) Check ACK field, ESTABLISHED state, p. 72: The ACK is a duplicate if SEG.ACK =< SND.UNA (the = was omitted). Similarly, the window should be updated if: SND.UNA =< SEG.ACK =< SND.NXT.

(h) USER TIMEOUT, p. 77:

It would be better to notify the application of the timeout rather than letting TCP force the connection closed. However, see also Section 4.2.3.5.

4.2.2.21 Acknowledging Queued Segments: RFC-793 Section 3.9

A TCP MAY send an ACK segment acknowledging RCV.NXT when a valid segment arrives that is in the window but not at the left window edge.

DISCUSSION:

RFC-793 (see page 74) was ambiguous about whether or not an ACK segment should be sent when an out-of-order segment was received, i.e., when SEG.SEQ was unequal to RCV.NXT.

One reason for ACKing out-of-order segments might be to support an experimental algorithm known as "fast retransmit". With this algorithm, the sender uses the "redundant" ACK's to deduce that a segment has been lost before the retransmission timer has expired. It counts the number of times an ACK has been received with the same value of SEG.ACK and with the same right window edge. If more than a threshold number of such ACK's is received, then the segment containing the octets starting at SEG.ACK is assumed to have been lost and is retransmitted, without awaiting a timeout. The threshold is chosen to compensate for the maximum likely segment reordering in the Internet. There is not yet enough experience with the fast retransmit algorithm to determine how useful it is.

4.2.3 SPECIFIC ISSUES

4.2.3.1 Retransmission Timeout Calculation

A host TCP MUST implement Karn's algorithm and Jacobson's algorithm for computing the retransmission timeout ("RTO").

- Jacobson's algorithm for computing the smoothed round-trip ("RTT") time incorporates a simple measure of the variance [TCP:7].

- Karn's algorithm for selecting RTT measurements ensures that ambiguous round-trip times will not corrupt the calculation of the smoothed round-trip time [TCP:6].

This implementation also MUST include "exponential backoff" for successive RTO values for the same segment.
Retransmission of SYN segments SHOULD use the same algorithm as data segments.

DISCUSSION:

There were two known problems with the RTO calculations specified in RFC-793. First, the accurate measurement of RTTs is difficult when there are retransmissions. Second, the algorithm to compute the smoothed round-trip time is inadequate [TCP:7], because it incorrectly

assumed that the variance in RTT values would be small and constant. These problems were solved by Karn's and Jacobson's algorithm, respectively.

The performance increase resulting from the use of these improvements varies from noticeable to dramatic. Jacobson's algorithm for incorporating the measured RTT variance is especially important on a low-speed link, where the natural variation of packet sizes causes a large variation in RTT. One vendor found link utilization on a 9.6kb line went from 10% to 90% as a result of implementing Jacobson's variance algorithm in TCP.

The following values SHOULD be used to initialize the estimation parameters for a new connection:

(a) RTT = 0 seconds.

(b) RTO = 3 seconds. (The smoothed variance is to be initialized to the value that will result in this RTO).

The recommended upper and lower bounds on the RTO are known to be inadequate on large internets. The lower bound SHOULD be measured in fractions of a second (to accommodate high speed LANs) and the upper bound should be 2*MSL, i.e., 240 seconds.

DISCUSSION:
Experience has shown that these initialization values are reasonable, and that in any case the Karn and Jacobson algorithms make TCP behavior reasonably insensitive to the initial parameter choices.

4.2.3.2 When to Send an ACK Segment

A host that is receiving a stream of TCP data segments can increase efficiency in both the Internet and the hosts by sending fewer than one ACK (acknowledgment) segment per data segment received; this is known as a "delayed ACK" [TCP:5].

A TCP SHOULD implement a delayed ACK, but an ACK should not be excessively delayed; in particular, the delay MUST be less than 0.5 seconds, and in a stream of full-sized segments there SHOULD be an ACK for at least every second segment.

DISCUSSION:

A delayed ACK gives the application an opportunity to update the window and perhaps to send an immediate response. In particular, in the case of character-mode remote login, a delayed ACK can reduce the number of segments sent by the server by a factor of 3 (ACK, window update, and echo character all combined in one segment).

In addition, on some large multi-user hosts, a delayed ACK can substantially reduce protocol processing overhead by reducing the total number of packets to be processed [TCP:5]. However, excessive delays on ACK's can disturb the round-trip timing and packet "clocking" algorithms [TCP:7].

4.2.3.3 When to Send a Window Update

A TCP MUST include a SWS avoidance algorithm in the receiver [TCP:5].

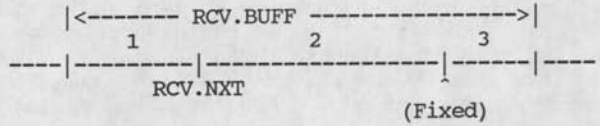
IMPLEMENTATION:

The receiver's SWS avoidance algorithm determines when the right window edge may be advanced; this is customarily known as "updating the window". This algorithm combines with the delayed ACK algorithm (see Section 4.2.3.2) to determine when an ACK segment containing the current window will really be sent to the receiver. We use the notation of RFC-793; see Figures 4 and 5 in that document.

The solution to receiver SWS is to avoid advancing the right window edge RCV.NXT+RCV.WND in small increments, even if data is received from the network in small segments.

Suppose the total receive buffer space is RCV.BUFF. At any given moment, RCV.USER octets of this total may be tied up with data that has been received and acknowledged but which the user process has not yet consumed. When the connection is quiescent, RCV.WND = RCV.BUFF and RCV.USER = 0.

Keeping the right window edge fixed as data arrives and is acknowledged requires that the receiver offer less than its full buffer space, i.e., the receiver must specify a RCV.WND that keeps RCV.NXT+RCV.WND constant as RCV.NXT increases. Thus, the total buffer space RCV.BUFF is generally divided into three parts:



1 - RCV.USER = data received but not yet consumed;
2 - RCV.WND = space advertised to sender;
3 - Reduction = space available but not yet advertised.

The suggested SWS avoidance algorithm for the receiver is to keep RCV.NXT+RCV.WND fixed until the reduction satisfies:

RCV.BUFF - RCV.USER - RCV.WND >=

min(Fr * RCV.BUFF, Eff.snd.MSS)

where Fr is a fraction whose recommended value is 1/2, and Eff.snd.MSS is the effective send MSS for the connection (see Section 4.2.2.6). When the inequality is satisfied, RCV.WND is set to RCV.BUFF-RCV.USER.

Note that the general effect of this algorithm is to advance RCV.WND in increments of Eff.snd.MSS (for realistic receive buffers: Eff.snd.MSS < RCV.BUFF/2). Note also that the receiver must use its own Eff.snd.MSS, assuming it is the same as the sender's.

4.2.3.4 When to Send Data

A TCP MUST include a SWS avoidance algorithm in the sender.

A TCP SHOULD implement the Nagle Algorithm [TCP:9] to coalesce short segments. However, there MUST be a way for an application to disable the Nagle algorithm on an individual connection. In all cases, sending data is also subject to the limitation imposed by the Slow Start algorithm (Section 4.2.2.15).

DISCUSSION:
The Nagle algorithm is generally as follows:

If there is unacknowledged data (i.e., SND.NXT > SND.UNA), then the sending TCP buffers all user

data (regardless of the PSH bit), until the outstanding data has been acknowledged or until the TCP can send a full-sized segment (Eff.snd.MSS bytes; see Section 4.2.2.6).

Some applications (e.g., real-time display window updates) require that the Nagle algorithm be turned off, so small data segments can be streamed out at the maximum rate.

IMPLEMENTATION:

The sender's SWS avoidance algorithm is more difficult than the receivers's, because the sender does not know (directly) the receiver's total buffer space RCV.BUFF. An approach which has been found to work well is for the sender to calculate Max(SND.WND), the maximum send window it has seen so far on the connection, and to use this value as an estimate of RCV.BUFF. Unfortunately, this can only be an estimate; the receiver may at any time reduce the size of RCV.BUFF. To avoid a resulting deadlock, it is necessary to have a timeout to force transmission of data, overriding the SWS avoidance algorithm. In practice, this timeout should seldom occur.

The "useable window" [TCP:5] is:

U = SND.UNA + SND.WND - SND.NXT

i.e., the offered window less the amount of data sent but not acknowledged. If D is the amount of data queued in the sending TCP but not yet sent, then the following set of rules is recommended.

Send data:

(1) if a maximum-sized segment can be sent, i.e, if:

min(D,U) >= Eff.snd.MSS;

(2) or if the data is pushed and all queued data can be sent now, i.e., if:

[SND.NXT = SND.UNA and] PUSHED and D <= U

(the bracketed condition is imposed by the Nagle algorithm);

(3) or if at least a fraction Fs of the maximum window can be sent, i.e., if:

[SND.NXT = SND.UNA and]

min(D.U) >= Fs * Max(SND.WND);

(4) or if data is PUSHed and the override timeout occurs.

Here Fs is a fraction whose recommended value is 1/2. The override timeout should be in the range 0.1 - 1.0 seconds. It may be convenient to combine this timer with the timer used to probe zero windows (Section 4.2.2.17).

Finally, note that the SWS avoidance algorithm just specified is to be used instead of the sender-side algorithm contained in [TCP:5].

4.2.3.5 TCP Connection Failures

Excessive retransmission of the same segment by TCP indicates some failure of the remote host or the Internet path. This failure may be of short or long duration. The following procedure MUST be used to handle excessive retransmissions of data segments [IP:11]:

(a) There are two thresholds R1 and R2 measuring the amount of retransmission that has occurred for the same segment. R1 and R2 might be measured in time units or as a count of retransmissions.

(b) When the number of transmissions of the same segment reaches or exceeds threshold R1, pass negative advice (see Section 3.3.1.4) to the IP layer, to trigger dead-gateway diagnosis.

(c) When the number of transmissions of the same segment reaches a threshold R2 greater than R1, close the connection.

(d) An application MUST be able to set the value for R2 for a particular connection. For example, an interactive application might set R2 to "infinity," giving the user control over when to disconnect.

(d) TCP SHOULD inform the application of the delivery problem (unless such information has been disabled by the application; see Section 4.2.4.1), when R1 is reached and before R2. This will allow a remote login (User Telnet) application program to inform the user, for example.

The value of R1 SHOULD correspond to at least 3 retransmissions, at the current RTO. The value of R2 SHOULD correspond to at least 100 seconds.

An attempt to open a TCP connection could fail with excessive retransmissions of the SYN segment or by receipt of a RST segment or an ICMP Port Unreachable. SYN retransmissions MUST be handled in the general way just described for data retransmissions, including notification of the application layer.

However, the values of R1 and R2 may be different for SYN and data segments. In particular, R2 for a SYN segment MUST be set large enough to provide retransmission of the segment for at least 3 minutes. The application can close the connection (i.e., give up on the open attempt) sooner, of course.

DISCUSSION:
Some Internet paths have significant setup times, and the number of such paths is likely to increase in the future.

4.2.3.6 TCP Keep-Alives

Implementors MAY include "keep-alives" in their TCP implementations, although this practice is not universally accepted. If keep-alives are included, the application MUST be able to turn them on or off for each TCP connection, and they MUST default to off.

Keep-alive packets MUST only be sent when no data or acknowledgement packets have been received for the connection within an interval. This interval MUST be configurable and MUST default to no less than two hours.

It is extremely important to remember that ACK segments that contain no data are not reliably transmitted by TCP. Consequently, if a keep-alive mechanism is implemented it MUST NOT interpret failure to respond to any specific probe as a dead connection.

An implementation SHOULD send a keep-alive segment with no data; however, it MAY be configurable to send a keep-alive segment containing one garbage octet, for compatibility with erroneous TCP implementations.

DISCUSSION:

A "keep-alive" mechanism periodically probes the other end of a connection when the connection is otherwise idle, even when there is no data to be sent. The TCP specification does not include a keep-alive mechanism because it could: (1) cause perfectly good connections to break during transient Internet failures; (2) consume unnecessary bandwidth ("if no one is using the connection, who cares if it is still good?"); and (3) cost money for an Internet path that charges for packets.

Some TCP implementations, however, have included a keep-alive mechanism. To confirm that an idle connection is still active, these implementations send a probe segment designed to elicit a response from the peer TCP. Such a segment generally contains SEG.SEQ = SND.NXT-1 and may or may not contain one garbage octet of data. Note that on a quiet connection SND.NXT = RCV.NXT, so that this SEG.SEQ will be outside the window. Therefore, the probe causes the receiver to return an acknowledgment segment, confirming that the connection is still live. If the peer has dropped the connection due to a network partition or a crash, it will respond with a RST instead of an acknowledgment segment.

Unfortunately, some misbehaved TCP implementations fail to respond to a segment with SEG.SEQ = SND.NXT-1 unless the segment contains data. Alternatively, an implementation could determine whether a peer responded correctly to keep-alive packets with no garbage data octet.

A TCP keep-alive mechanism should only be invoked in server applications that might otherwise hang indefinitely and consume resources unnecessarily if a client crashes or aborts a connection during a network failure.

4.2.3.7 TCP Multihoming

If an application on a multihomed host does not specify the local IP address when actively opening a TCP connection, then the TCP MUST ask the IP layer to select a local IP address before sending the (first) SYN. See the function GET_SRCADDR() in Section 3.4.

At all other times, a previous segment has either been sent or received on this connection, and TCP MUST use the same local address is used that was used in those previous segments.

4.2.3.8 IP Options

When received options are passed up to TCP from the IP layer, TCP MUST ignore options that it does not understand.

A TCP MAY support the Time Stamp and Record Route options.

An application MUST be able to specify a source route when it actively opens a TCP connection, and this MUST take precedence over a source route received in a datagram.

When a TCP connection is OPENed passively and a packet arrives with a completed IP Source Route option (containing a return route), TCP MUST save the return route and use it for all segments sent on this connection. If a different source route arrives in a later segment, the later definition SHOULD override the earlier one.

4.2.3.9 ICMP Messages

TCP MUST act on an ICMP error message passed up from the IP layer, directing it to the connection that created the error. The necessary demultiplexing information can be found in the IP header contained within the ICMP message.

- Source Quench

  TCP MUST react to a Source Quench by slowing transmission on the connection. The RECOMMENDED procedure is for a Source Quench to trigger a "slow start," as if a retransmission timeout had occurred.

- Destination Unreachable -- codes 0, 1, 5

  Since these Unreachable messages indicate soft error

conditions, TCP MUST NOT abort the connection, and it SHOULD make the information available to the application.

DISCUSSION:
TCP could report the soft error condition directly to the application layer with an upcall to the ERROR_REPORT routine, or it could merely note the message and report it to the application only when and if the TCP connection times out.

- Destination Unreachable -- codes 2-4

  These are hard error conditions, so TCP SHOULD abort the connection.

- Time Exceeded -- codes 0, 1

  This should be handled the same way as Destination Unreachable codes 0, 1, 5 (see above).

- Parameter Problem

  This should be handled the same way as Destination Unreachable codes 0, 1, 5 (see above).

4.2.3.10 Remote Address Validation

A TCP implementation MUST reject as an error a local OPEN call for an invalid remote IP address (e.g., a broadcast or multicast address).

An incoming SYN with an invalid source address must be ignored either by TCP or by the IP layer (see Section 3.2.1.3).

A TCP implementation MUST silently discard an incoming SYN segment that is addressed to a broadcast or multicast address.

4.2.3.11 TCP Traffic Patterns

IMPLEMENTATION:
The TCP protocol specification [TCP:1] gives the implementor much freedom in designing the algorithms that control the message flow over the connection -- packetizing, managing the window, sending

acknowledgments, etc. These design decisions are difficult because a TCP must adapt to a wide range of traffic patterns. Experience has shown that a TCP implementor needs to verify the design on two extreme traffic patterns:

- o Single-character Segments

  Even if the sender is using the Nagle Algorithm, when a TCP connection carries remote login traffic across a low-delay LAN the receiver will generally get a stream of single-character segments. If remote terminal echo mode is in effect, the receiver's system will generally echo each character as it is received.

- o Bulk Transfer

  When TCP is used for bulk transfer, the data stream should be made up (almost) entirely of segments of the size of the effective MSS. Although TCP uses a sequence number space with byte (octet) granularity, in bulk-transfer mode its operation should be as if TCP used a sequence space that counted only segments.

Experience has furthermore shown that a single TCP can effectively and efficiently handle these two extremes.

The most important tool for verifying a new TCP implementation is a packet trace program. There is a large volume of experience showing the importance of tracing a variety of traffic patterns with other TCP implementations and studying the results carefully.

4.2.3.12 Efficiency

IMPLEMENTATION:

Extensive experience has led to the following suggestions for efficient implementation of TCP:

(a) Don't Copy Data

In bulk data transfer, the primary CPU-intensive tasks are copying data from one place to another and checksumming the data. It is vital to minimize the number of copies of TCP data. Since

the ultimate speed limitation may be fetching data across the memory bus, it may be useful to combine the copy with checksumming, doing both with a single memory fetch.

(b) Hand-Craft the Checksum Routine

A good TCP checksumming routine is typically two to five times faster than a simple and direct implementation of the definition. Great care and clever coding are often required and advisable to make the checksumming code "blazing fast". See [TCP:10].

(c) Code for the Common Case

TCP protocol processing can be complicated, but for most segments there are only a few simple decisions to be made. Per-segment processing will be greatly speeded up by coding the main line to minimize the number of decisions in the most common case.

4.2.4 TCP/APPLICATION LAYER INTERFACE

4.2.4.1 Asynchronous Reports

There MUST be a mechanism for reporting soft TCP error conditions to the application. Generically, we assume this takes the form of an application-supplied ERROR_REPORT routine that may be upcalled [INTRO:7] asynchronously from the transport layer:

ERROR_REPORT(local connection name, reason, subreason)

The precise encoding of the reason and subreason parameters is not specified here. However, the conditions that are reported asynchronously to the application MUST include:

* ICMP error message arrived (see 4.2.3.9)

* Excessive retransmissions (see 4.2.3.5)

* Urgent pointer advance (see 4.2.2.4).

However, an application program that does not want to receive such ERROR_REPORT calls SHOULD be able to

effectively disable these calls.

DISCUSSION:
These error reports generally reflect soft errors that can be ignored without harm by many applications. It has been suggested that these error report calls should default to "disabled," but this is not required.

4.2.4.2 Type-of-Service

The application layer MUST be able to specify the Type-of-Service (TOS) for segments that are sent on a connection. It not required, but the application SHOULD be able to change the TOS during the connection lifetime. TCP SHOULD pass the current TOS value without change to the IP layer, when it sends segments on the connection.

The TOS will be specified independently in each direction on the connection, so that the receiver application will specify the TOS used for ACK segments.

TCP MAY pass the most recently received TOS up to the application.

DISCUSSION
Some applications (e.g., SMTP) change the nature of their communication during the lifetime of a connection, and therefore would like to change the TOS specification.

Note also that the OPEN call specified in RFC-793 includes a parameter ("options") in which the caller can specify IP options such as source route, record route, or timestamp.

4.2.4.3 Flush Call

Some TCP implementations have included a FLUSH call, which will empty the TCP send queue of any data for which the user has issued SEND calls but which is still to the right of the current send window. That is, it flushes as much queued send data as possible without losing sequence number synchronization. This is useful for implementing the "abort output" function of Telnet.

4.2.4.4 Multihoming

The user interface outlined in sections 2.7 and 3.8 of RFC-793 needs to be extended for multihoming. The OPEN call MUST have an optional parameter:

```
OPEN( ... [local IP address,] ... )
```

to allow the specification of the local IP address.

DISCUSSION:
Some TCP-based applications need to specify the local IP address to be used to open a particular connection; FTP is an example.

IMPLEMENTATION:
A passive OPEN call with a specified "local IP address" parameter will await an incoming connection request to that address. If the parameter is unspecified, a passive OPEN will await an incoming connection request to any local IP address, and then bind the local IP address of the connection to the particular address that is used.

For an active OPEN call, a specified "local IP address" parameter will be used for opening the connection. If the parameter is unspecified, the networking software will choose an appropriate local IP address (see Section 3.3.4.2) for the connection

4.2.5 TCP REQUIREMENT SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Push flag							
Aggregate or queue un-pushed data	4.2.2.2			x			
Sender collapse successive PSH flags	4.2.2.2		x				
SEND call can specify PUSH	4.2.2.2			x			

If cannot: sender buffer indefinitely	4.2.2.2					x	
If cannot: PSH last segment	4.2.2.2	x					
Notify receiving ALP of PSH	4.2.2.2			x			1
Send max size segment when possible	4.2.2.2		x				
Window							
Treat as unsigned number	4.2.2.3	x					
Handle as 32-bit number	4.2.2.3		x				
Shrink window from right	4.2.2.16				x		
Robust against shrinking window	4.2.2.16	x					
Receiver's window closed indefinitely	4.2.2.17			x			
Sender probe zero window	4.2.2.17	x					
First probe after RTO	4.2.2.17		x				
Exponential backoff	4.2.2.17		x				
Allow window stay zero indefinitely	4.2.2.17	x					
Sender timeout OK conn with zero wind	4.2.2.17					x	
Urgent Data							
Pointer points to last octet	4.2.2.4	x					
Arbitrary length urgent data sequence	4.2.2.4	x					
Inform ALP asynchronously of urgent data	4.2.2.4	x					1
ALP can learn if/how much urgent data Q'd	4.2.2.4	x					1
TCP Options							
Receive TCP option in any segment	4.2.2.5	x					
Ignore unsupported options	4.2.2.5	x					
Cope with illegal option length	4.2.2.5	x					
Implement sending & receiving MSS option	4.2.2.6	x					
Send MSS option unless 536	4.2.2.6		x				
Send MSS option always	4.2.2.6			x			
Send-MSS default is 536	4.2.2.6	x					
Calculate effective send seg size	4.2.2.6	x					
TCP Checksums							
Sender compute checksum	4.2.2.7	x					
Receiver check checksum	4.2.2.7	x					
Use clock-driven ISN selection	4.2.2.9	x					
Opening Connections							
Support simultaneous open attempts	4.2.2.10	x					
SYN-RCVD remembers last state	4.2.2.11	x					
Passive Open call interfere with others	4.2.2.18					x	
Function: simultan. LISTENs for same port	4.2.2.18	x					
Ask IP for src address for SYN if necc.	4.2.3.7	x					
Otherwise, use local addr of conn.	4.2.3.7	x					
OPEN to broadcast/multicast IP Address	4.2.3.14					x	
Silently discard seg to bcast/mcast addr	4.2.3.14	x					

Closing Connections							
RST can contain data	4.2.2.12		x				
Inform application of aborted conn	4.2.2.13	x					
Half-duplex close connections	4.2.2.13			x			
Send RST to indicate data lost	4.2.2.13		x				
In TIME-WAIT state for 2xMSL seconds	4.2.2.13	x					
Accept SYN from TIME-WAIT state	4.2.2.13			x			
Retransmissions							
Jacobson Slow Start algorithm	4.2.2.15	x					
Jacobson Congestion-Avoidance algorithm	4.2.2.15	x					
Retransmit with same IP ident	4.2.2.15			x			
Karn's algorithm	4.2.3.1	x					
Jacobson's RTO estimation alg.	4.2.3.1	x					
Exponential backoff	4.2.3.1	x					
SYN RTO calc same as data	4.2.3.1		x				
Recommended initial values and bounds	4.2.3.1		x				
Generating ACK's:							
Queue out-of-order segments	4.2.2.20		x				
Process all Q'd before send ACK	4.2.2.20	x					
Send ACK for out-of-order segment	4.2.2.21			x			
Delayed ACK's	4.2.3.2		x				
Delay < 0.5 seconds	4.2.3.2	x					
Every 2nd full-sized segment ACK'd	4.2.3.2	x					
Receiver SWS-Avoidance Algorithm	4.2.3.3	x					
Sending data							
Configurable TTL	4.2.2.19	x					
Sender SWS-Avoidance Algorithm	4.2.3.4	x					
Nagle algorithm	4.2.3.4		x				
Application can disable Nagle algorithm	4.2.3.4	x					
Connection Failures:							
Negative advice to IP on R1 retxs	4.2.3.5	x					
Close connection on R2 retxs	4.2.3.5	x					
ALP can set R2	4.2.3.5	x					1
Inform ALP of R1<=retxs<R2	4.2.3.5		x				1
Recommended values for R1, R2	4.2.3.5		x				
Same mechanism for SYNs	4.2.3.5	x					
R2 at least 3 minutes for SYN	4.2.3.5	x					
Send Keep-alive Packets:	4.2.3.6			x			
- Application can request	4.2.3.6	x					
- Default is "off"	4.2.3.6	x					
- Only send if idle for interval	4.2.3.6	x					
- Interval configurable	4.2.3.6	x					

- Default at least 2 hrs.	4.2.3.6	x					
- Tolerant of lost ACK's	4.2.3.6	x					
IP Options							
Ignore options TCP doesn't understand	4.2.3.8	x					
Time Stamp support	4.2.3.8			x			
Record Route support	4.2.3.8			x			
Source Route:							
ALP can specify	4.2.3.8	x					1
Overrides src rt in datagram	4.2.3.8	x					
Build return route from src rt	4.2.3.8	x					
Later src route overrides	4.2.3.8		x				
Receiving ICMP Messages from IP	4.2.3.9	x					
Dest. Unreach (0,1,5) => inform ALP	4.2.3.9		x				
Dest. Unreach (0,1,5) => abort conn	4.2.3.9					x	
Dest. Unreach (2-4) => abort conn	4.2.3.9		x				
Source Quench => slow start	4.2.3.9		x				
Time Exceeded => tell ALP, don't abort	4.2.3.9		x				
Param Problem => tell ALP, don't abort	4.2.3.9		x				
Address Validation							
Reject OPEN call to invalid IP address	4.2.3.10	x					
Reject SYN from invalid IP address	4.2.3.10	x					
Silently discard SYN to bcast/mcast addr	4.2.3.10	x					
TCP/ALP Interface Services							
Error Report mechanism	4.2.4.1	x					
ALP can disable Error Report Routine	4.2.4.1		x				
ALP can specify TOS for sending	4.2.4.2	x					
Passed unchanged to IP	4.2.4.2		x				
ALP can change TOS during connection	4.2.4.2		x				
Pass received TOS up to ALP	4.2.4.2			x			
FLUSH call	4.2.4.3			x			
Optional local IP addr parm. in OPEN	4.2.4.4	x					
---	---	-	-	-	-	-	--
---	---	-	-	-	-	-	--

FOOTNOTES:

(1) "ALP" means Application-Layer program.

5. REFERENCES

INTRODUCTORY REFERENCES

[INTRO:1] "Requirements for Internet Hosts -- Application and Support," IETF Host Requirements Working Group, R. Braden, Ed., RFC-1123, October 1989.

[INTRO:2] "Requirements for Internet Gateways," R. Braden and J. Postel, RFC-1009, June 1987.

[INTRO:3] "DDN Protocol Handbook," NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

[INTRO:4] "Official Internet Protocols," J. Reynolds and J. Postel, RFC-1011, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

[INTRO:5] "Protocol Document Order Information," O. Jacobsen and J. Postel, RFC-980, March 1986.

[INTRO:6] "Assigned Numbers," J. Reynolds and J. Postel, RFC-1010, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

[INTRO:7] "Modularity and Efficiency in Protocol Implementations," D. Clark, RFC-817, July 1982.

[INTRO:8] "The Structuring of Systems Using Upcalls," D. Clark, 10th ACM SOSP, Orcas Island, Washington, December 1985.

Secondary References:

[INTRO:9] "A Protocol for Packet Network Intercommunication," V. Cerf and R. Kahn, IEEE Transactions on Communication, May 1974.

[INTRO:10] "The ARPA Internet Protocol," J. Postel, C. Sunshine, and D. Cohen, Computer Networks, Vol. 5, No. 4, July 1981.

[INTRO:11] "The DARPA Internet Protocol Suite," B. Leiner, J. Postel, R. Cole and D. Mills, Proceedings INFOCOM 85, IEEE, Washington DC,

March 1985. Also in: IEEE Communications Magazine, March 1985. Also available as ISI-RS-85-153.

[INTRO:12] "Final Text of DIS8473, Protocol for Providing the Connectionless Mode Network Service," ANSI, published as RFC-994, March 1986.

[INTRO:13] "End System to Intermediate System Routing Exchange Protocol," ANSI X3S3.3, published as RFC-995, April 1986.

LINK LAYER REFERENCES

[LINK:1] "Trailer Encapsulations," S. Leffler and M. Karels, RFC-893, April 1984.

[LINK:2] "An Ethernet Address Resolution Protocol," D. Plummer, RFC-826, November 1982.

[LINK:3] "A Standard for the Transmission of IP Datagrams over Ethernet Networks," C. Hornig, RFC-894, April 1984.

[LINK:4] "A Standard for the Transmission of IP Datagrams over IEEE 802 "Networks," J. Postel and J. Reynolds, RFC-1042, February 1988.

This RFC contains a great deal of information of importance to Internet implementers planning to use IEEE 802 networks.

IP LAYER REFERENCES

[IP:1] "Internet Protocol (IP)," J. Postel, RFC-791, September 1981.

[IP:2] "Internet Control Message Protocol (ICMP)," J. Postel, RFC-792, September 1981.

[IP:3] "Internet Standard Subnetting Procedure," J. Mogul and J. Postel, RFC-950, August 1985.

[IP:4] "Host Extensions for IP Multicasting," S. Deering, RFC-1112, August 1989.

[IP:5] "Military Standard Internet Protocol," MIL-STD-1777, Department of Defense, August 1983.

This specification, as amended by RFC-963, is intended to describe

the Internet Protocol but has some serious omissions (e.g., the mandatory subnet extension [IP:3] and the optional multicasting extension [IP:4]). It is also out of date. If there is a conflict, RFC-791, RFC-792, and RFC-950 must be taken as authoritative, while the present document is authoritative over all.

[IP:6] "Some Problems with the Specification of the Military Standard Internet Protocol," D. Sidhu, RFC-963, November 1985.

[IP:7] "The TCP Maximum Segment Size and Related Topics," J. Postel, RFC-879, November 1983.

Discusses and clarifies the relationship between the TCP Maximum Segment Size option and the IP datagram size.

[IP:8] "Internet Protocol Security Options," B. Schofield, RFC-1108, October 1989.

[IP:9] "Fragmentation Considered Harmful," C. Kent and J. Mogul, ACM SIGCOMM-87, August 1987. Published as ACM Comp Comm Review, Vol. 17, no. 5.

This useful paper discusses the problems created by Internet fragmentation and presents alternative solutions.

[IP:10] "IP Datagram Reassembly Algorithms," D. Clark, RFC-815, July 1982.

This and the following paper should be read by every implementor.

[IP:11] "Fault Isolation and Recovery," D. Clark, RFC-816, July 1982.

SECONDARY IP REFERENCES:

[IP:12] "Broadcasting Internet Datagrams in the Presence of Subnets," J. Mogul, RFC-922, October 1984.

[IP:13] "Name, Addresses, Ports, and Routes," D. Clark, RFC-814, July 1982.

[IP:14] "Something a Host Could Do with Source Quench: The Source Quench Introduced Delay (SQUID)," W. Prue and J. Postel, RFC-1016, July 1987.

This RFC first described directed broadcast addresses. However, the bulk of the RFC is concerned with gateways, not hosts.

UDP REFERENCES:

[UDP:1] "User Datagram Protocol," J. Postel, RFC-768, August 1980.

TCP REFERENCES:

[TCP:1] "Transmission Control Protocol," J. Postel, RFC-793, September 1981.

[TCP:2] "Transmission Control Protocol," MIL-STD-1778, US Department of Defense, August 1984.

This specification as amended by RFC-964 is intended to describe the same protocol as RFC-793 [TCP:1]. If there is a conflict, RFC-793 takes precedence, and the present document is authoritative over both.

[TCP:3] "Some Problems with the Specification of the Military Standard Transmission Control Protocol," D. Sidhu and T. Blumer, RFC-964, November 1985.

[TCP:4] "The TCP Maximum Segment Size and Related Topics," J. Postel, RFC-879, November 1983.

[TCP:5] "Window and Acknowledgment Strategy in TCP," D. Clark, RFC-813, July 1982.

[TCP:6] "Round Trip Time Estimation," P. Karn & C. Partridge, ACM SIGCOMM-87, August 1987.

[TCP:7] "Congestion Avoidance and Control," V. Jacobson, ACM SIGCOMM-88, August 1988.

SECONDARY TCP REFERENCES:

[TCP:8] "Modularity and Efficiency in Protocol Implementation," D. Clark, RFC-817, July 1982.

[TCP:9] "Congestion Control in IP/TCP," J. Nagle, RFC-896, January 1984.

[TCP:10] "Computing the Internet Checksum," R. Braden, D. Borman, and C. Partridge, RFC-1071, September 1988.

[TCP:11] "TCP Extensions for Long-Delay Paths," V. Jacobson & R. Braden, RFC-1072, October 1988.

Security Considerations

There are many security issues in the communication layers of host software, but a full discussion is beyond the scope of this RFC.

The Internet architecture generally provides little protection against spoofing of IP source addresses, so any security mechanism that is based upon verifying the IP source address of a datagram should be treated with suspicion. However, in restricted environments some source-address checking may be possible. For example, there might be a secure LAN whose gateway to the rest of the Internet discarded any incoming datagram with a source address that spoofed the LAN address. In this case, a host on the LAN could use the source address to test for local vs. remote source. This problem is complicated by source routing, and some have suggested that source-routed datagram forwarding by hosts (see Section 3.3.5) should be outlawed for security reasons.

Security-related issues are mentioned in sections concerning the IP Security option (Section 3.2.1.8), the ICMP Parameter Problem message (Section 3.2.2.5), IP options in UDP datagrams (Section 4.1.3.2), and reserved TCP ports (Section 4.2.2.1).

Author's Address

Robert Braden
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Phone: (213) 822 1511

EMail: Braden@ISI.EDU

Network Working Group
Request for Comments: 1123

Internet Engineering Task Force
R. Braden, Editor
October 1989

Requirements for Internet Hosts -- Application and Support

Status of This Memo

This RFC is an official specification for the Internet community. It incorporates by reference, amends, corrects, and supplements the primary protocol standards documents relating to hosts. Distribution of this document is unlimited.

Summary

This RFC is one of a pair that defines and discusses the requirements for Internet host software. This RFC covers the application and support protocols; its companion RFC-1122 covers the communication protocol layers: link layer, IP layer, and transport layer.

Table of Contents

1. INTRODUCTION

This document is one of a pair that defines and discusses the requirements for host system implementations of the Internet protocol suite. This RFC covers the applications layer and support protocols. Its companion RFC, "Requirements for Internet Hosts -- Communications Layers" [INTRO:1] covers the lower layer protocols: transport layer, IP layer, and link layer.

These documents are intended to provide guidance for vendors, implementors, and users of Internet communication software. They represent the consensus of a large body of technical experience and wisdom, contributed by members of the Internet research and vendor communities.

This RFC enumerates standard protocols that a host connected to the Internet must use, and it incorporates by reference the RFCs and other documents describing the current specifications for these protocols. It corrects errors in the referenced documents and adds additional discussion and guidance for an implementor.

For each protocol, this document also contains an explicit set of requirements, recommendations, and options. The reader must understand that the list of requirements in this document is incomplete by itself; the complete set of requirements for an Internet host is primarily defined in the standard protocol specification documents, with the corrections, amendments, and supplements contained in this RFC.

A good-faith implementation of the protocols that was produced after careful reading of the RFC's and with some interaction with the Internet technical community, and that followed good communications software engineering practices, should differ from the requirements of this document in only minor ways. Thus, in many cases, the "requirements" in this RFC are already stated or implied in the standard protocol documents, so that their inclusion here is, in a sense, redundant. However, they were included because some past implementation has made the wrong choice, causing problems of interoperability, performance, and/or robustness.

This document includes discussion and explanation of many of the requirements and recommendations. A simple list of requirements would be dangerous, because:

- Some required features are more important than others, and some features are optional.

- There may be valid reasons why particular vendor products that

are designed for restricted contexts might choose to use different specifications.

However, the specifications of this document must be followed to meet the general goal of arbitrary host interoperation across the diversity and complexity of the Internet system. Although most current implementations fail to meet these requirements in various ways, some minor and some major, this specification is the ideal towards which we need to move.

These requirements are based on the current level of Internet architecture. This document will be updated as required to provide additional clarifications or to include additional information in those areas in which specifications are still evolving.

This introductory section begins with general advice to host software vendors, and then gives some guidance on reading the rest of the document. Section 2 contains general requirements that may be applicable to all application and support protocols. Sections 3, 4, and 5 contain the requirements on protocols for the three major applications: Telnet, file transfer, and electronic mail, respectively. Section 6 covers the support applications: the domain name system, system initialization, and management. Finally, all references will be found in Section 7.

1.1 The Internet Architecture

For a brief introduction to the Internet architecture from a host viewpoint, see Section 1.1 of [INTRO:1]. That section also contains recommended references for general background on the Internet architecture.

1.2 General Considerations

There are two important lessons that vendors of Internet host software have learned and which a new vendor should consider seriously.

1.2.1 Continuing Internet Evolution

The enormous growth of the Internet has revealed problems of management and scaling in a large datagram-based packet communication system. These problems are being addressed, and as a result there will be continuing evolution of the specifications described in this document. These changes will be carefully planned and controlled, since there is extensive participation in this planning by the vendors and by the organizations responsible for operations of the networks.

Development, evolution, and revision are characteristic of computer network protocols today, and this situation will persist for some years. A vendor who develops computer communication software for the Internet protocol suite (or any other protocol suite!) and then fails to maintain and update that software for changing specifications is going to leave a trail of unhappy customers. The Internet is a large communication network, and the users are in constant contact through it. Experience has shown that knowledge of deficiencies in vendor software propagates quickly through the Internet technical community.

1.2.2 Robustness Principle

At every layer of the protocols, there is a general rule whose application can lead to enormous benefits in robustness and interoperability:

> "Be liberal in what you accept, and
> conservative in what you send"

Software should be written to deal with every conceivable error, no matter how unlikely; sooner or later a packet will come in with that particular combination of errors and attributes, and unless the software is prepared, chaos can ensue. In general, it is best to assume that the network is filled with malevolent entities that will send in packets designed to have the worst possible effect. This assumption will lead to suitable protective design, although the most serious problems in the Internet have been caused by unenvisaged mechanisms triggered by low-probability events; mere human malice would never have taken so devious a course!

Adaptability to change must be designed into all levels of Internet host software. As a simple example, consider a protocol specification that contains an enumeration of values for a particular header field -- e.g., a type field, a port number, or an error code; this enumeration must be assumed to be incomplete. Thus, if a protocol specification defines four possible error codes, the software must not break when a fifth code shows up. An undefined code might be logged (see below), but it must not cause a failure.

The second part of the principle is almost as important: software on other hosts may contain deficiencies that make it unwise to exploit legal but obscure protocol features. It is unwise to stray far from the obvious and simple, lest untoward effects result elsewhere. A corollary of this is "watch out

for misbehaving hosts"; host software should be prepared, not just to survive other misbehaving hosts, but also to cooperate to limit the amount of disruption such hosts can cause to the shared communication facility.

1.2.3 Error Logging

The Internet includes a great variety of host and gateway systems, each implementing many protocols and protocol layers, and some of these contain bugs and mis-features in their Internet protocol software. As a result of complexity, diversity, and distribution of function, the diagnosis of user problems is often very difficult.

Problem diagnosis will be aided if host implementations include a carefully designed facility for logging erroneous or "strange" protocol events. It is important to include as much diagnostic information as possible when an error is logged. In particular, it is often useful to record the header(s) of a packet that caused an error. However, care must be taken to ensure that error logging does not consume prohibitive amounts of resources or otherwise interfere with the operation of the host.

There is a tendency for abnormal but harmless protocol events to overflow error logging files; this can be avoided by using a "circular" log, or by enabling logging only while diagnosing a known failure. It may be useful to filter and count duplicate successive messages. One strategy that seems to work well is: (1) always count abnormalities and make such counts accessible through the management protocol (see Section 6.3); and (2) allow the logging of a great variety of events to be selectively enabled. For example, it might useful to be able to "log everything" or to "log everything for host X".

Note that different managements may have differing policies about the amount of error logging that they want normally enabled in a host. Some will say, "if it doesn't hurt me, I don't want to know about it", while others will want to take a more watchful and aggressive attitude about detecting and removing protocol abnormalities.

1.2.4 Configuration

It would be ideal if a host implementation of the Internet protocol suite could be entirely self-configuring. This would allow the whole suite to be implemented in ROM or cast into silicon, it would simplify diskless workstations, and it would

be an immense boon to harried LAN administrators as well as system vendors. We have not reached this ideal; in fact, we are not even close.

At many points in this document, you will find a requirement that a parameter be a configurable option. There are several different reasons behind such requirements. In a few cases, there is current uncertainty or disagreement about the best value, and it may be necessary to update the recommended value in the future. In other cases, the value really depends on external factors -- e.g., the size of the host and the distribution of its communication load, or the speeds and topology of nearby networks -- and self-tuning algorithms are unavailable and may be insufficient. In some cases, configurability is needed because of administrative requirements.

Finally, some configuration options are required to communicate with obsolete or incorrect implementations of the protocols, distributed without sources, that unfortunately persist in many parts of the Internet. To make correct systems coexist with these faulty systems, administrators often have to "mis-configure" the correct systems. This problem will correct itself gradually as the faulty systems are retired, but it cannot be ignored by vendors.

When we say that a parameter must be configurable, we do not intend to require that its value be explicitly read from a configuration file at every boot time. We recommend that implementors set up a default for each parameter, so a configuration file is only necessary to override those defaults that are inappropriate in a particular installation. Thus, the configurability requirement is an assurance that it will be POSSIBLE to override the default when necessary, even in a binary-only or ROM-based product.

This document requires a particular value for such defaults in some cases. The choice of default is a sensitive issue when the configuration item controls the accommodation to existing faulty systems. If the Internet is to converge successfully to complete interoperability, the default values built into implementations must implement the official protocol, not "mis-configurations" to accommodate faulty implementations. Although marketing considerations have led some vendors to choose mis-configuration defaults, we urge vendors to choose defaults that will conform to the standard.

Finally, we note that a vendor needs to provide adequate

documentation on all configuration parameters, their limits and effects.

1.3 Reading this Document

1.3.1 Organization

In general, each major section is organized into the following subsections:

(1) Introduction

(2) Protocol Walk-Through -- considers the protocol specification documents section-by-section, correcting errors, stating requirements that may be ambiguous or ill-defined, and providing further clarification or explanation.

(3) Specific Issues -- discusses protocol design and implementation issues that were not included in the walk-through.

(4) Interfaces -- discusses the service interface to the next higher layer.

(5) Summary -- contains a summary of the requirements of the section.

Under many of the individual topics in this document, there is parenthetical material labeled "DISCUSSION" or "IMPLEMENTATION". This material is intended to give clarification and explanation of the preceding requirements text. It also includes some suggestions on possible future directions or developments. The implementation material contains suggested approaches that an implementor may want to consider.

The summary sections are intended to be guides and indexes to the text, but are necessarily cryptic and incomplete. The summaries should never be used or referenced separately from the complete RFC.

1.3.2 Requirements

In this document, the words that are used to define the significance of each particular requirement are capitalized. These words are:

* "MUST"

  This word or the adjective "REQUIRED" means that the item is an absolute requirement of the specification.

* "SHOULD"

  This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "MAY"

  This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the MUST requirements for the protocols it implements. An implementation that satisfies all the MUST and all the SHOULD requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements for its protocols is said to be "conditionally compliant".

1.3.3 Terminology

This document uses the following technical terms:

Segment
: A segment is the unit of end-to-end transmission in the TCP protocol. A segment consists of a TCP header followed by application data. A segment is transmitted by encapsulation in an IP datagram.

Message
: This term is used by some application layer protocols (particularly SMTP) for an application data unit.

Datagram
: A [UDP] datagram is the unit of end-to-end transmission in the UDP protocol.

Multihomed
A host is said to be multihomed if it has multiple IP addresses to connected networks.

1.4 Acknowledgments

This document incorporates contributions and comments from a large group of Internet protocol experts, including representatives of university and research labs, vendors, and government agencies. It was assembled primarily by the Host Requirements Working Group of the Internet Engineering Task Force (IETF).

The Editor would especially like to acknowledge the tireless dedication of the following people, who attended many long meetings and generated 3 million bytes of electronic mail over the past 18 months in pursuit of this document: Philip Almquist, Dave Borman (Cray Research), Noel Chiappa, Dave Crocker (DEC), Steve Deering (Stanford), Mike Karels (Berkeley), Phil Karn (Bellcore), John Lekashman (NASA), Charles Lynn (BBN), Keith McCloghrie (TWG), Paul Mockapetris (ISI), Thomas Narten (Purdue), Craig Partridge (BBN), Drew Perkins (CMU), and James Van Bokkelen (FTP Software).

In addition, the following people made major contributions to the effort: Bill Barns (Mitre), Steve Bellovin (AT&T), Mike Brescia (BBN), Ed Cain (DCA), Annette DeSchon (ISI), Martin Gross (DCA), Phill Gross (NRI), Charles Hedrick (Rutgers), Van Jacobson (LBL), John Klensin (MIT), Mark Lottor (SRI), Milo Medin (NASA), Bill Melohn (Sun Microsystems), Greg Minshall (Kinetics), Jeff Mogul (DEC), John Mullen (CMC), Jon Postel (ISI), John Romkey (Epilogue Technology), and Mike StJohns (DCA). The following also made significant contributions to particular areas: Eric Allman (Berkeley), Rob Austein (MIT), Art Berggreen (ACC), Keith Bostic (Berkeley), Vint Cerf (NRI), Wayne Hathaway (NASA), Matt Korn (IBM), Erik Naggum (Naggum Software, Norway), Robert Ullmann (Prime Computer), David Waitzman (BBN), Frank Wancho (USA), Arun Welch (Ohio State), Bill Westfield (Cisco), and Rayan Zachariassen (Toronto).

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

2. GENERAL ISSUES

This section contains general requirements that may be applicable to all application-layer protocols.

2.1 Host Names and Numbers

The syntax of a legal Internet host name was specified in RFC-952 [DNS:4]. One aspect of host name syntax is hereby changed: the restriction on the first character is relaxed to allow either a letter or a digit. Host software MUST support this more liberal syntax.

Host software MUST handle host names of up to 63 characters and SHOULD handle host names of up to 255 characters.

Whenever a user inputs the identity of an Internet host, it SHOULD be possible to enter either (1) a host domain name or (2) an IP address in dotted-decimal ("#.#.#.#") form. The host SHOULD check the string syntactically for a dotted-decimal number before looking it up in the Domain Name System.

DISCUSSION:

This last requirement is not intended to specify the complete syntactic form for entering a dotted-decimal host number; that is considered to be a user-interface issue. For example, a dotted-decimal number must be enclosed within "[ ]" brackets for SMTP mail (see Section 5.2.17). This notation could be made universal within a host system, simplifying the syntactic checking for a dotted-decimal number.

If a dotted-decimal number can be entered without such identifying delimiters, then a full syntactic check must be made, because a segment of a host domain name is now allowed to begin with a digit and could legally be entirely numeric (see Section 6.1.2.4). However, a valid host name can never have the dotted-decimal form #.#.#.#, since at least the highest-level component label will be alphabetic.

2.2 Using Domain Name Service

Host domain names MUST be translated to IP addresses as described in Section 6.1.

Applications using domain name services MUST be able to cope with soft error conditions. Applications MUST wait a reasonable interval between successive retries due to a soft error, and MUST

allow for the possibility that network problems may deny service for hours or even days.

An application SHOULD NOT rely on the ability to locate a WKS record containing an accurate listing of all services at a particular host address, since the WKS RR type is not often used by Internet sites. To confirm that a service is present, simply attempt to use it.

2.3 Applications on Multihomed hosts

When the remote host is multihomed, the name-to-address translation will return a list of alternative IP addresses. As specified in Section 6.1.3.4, this list should be in order of decreasing preference. Application protocol implementations SHOULD be prepared to try multiple addresses from the list until success is obtained. More specific requirements for SMTP are given in Section 5.3.4.

When the local host is multihomed, a UDP-based request/response application SHOULD send the response with an IP source address that is the same as the specific destination address of the UDP request datagram. The "specific destination address" is defined in the "IP Addressing" section of the companion RFC [INTRO:1].

Similarly, a server application that opens multiple TCP connections to the same client SHOULD use the same local IP address for all.

2.4 Type-of-Service

Applications MUST select appropriate TOS values when they invoke transport layer services, and these values MUST be configurable. Note that a TOS value contains 5 bits, of which only the most-significant 3 bits are currently defined; the other two bits MUST be zero.

DISCUSSION:

As gateway algorithms are developed to implement Type-of-Service, the recommended values for various application protocols may change. In addition, it is likely that particular combinations of users and Internet paths will want non-standard TOS values. For these reasons, the TOS values must be configurable.

See the latest version of the "Assigned Numbers" RFC [INTRO:5] for the recommended TOS values for the major application protocols.

2.5 GENERAL APPLICATION REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
User interfaces:							
Allow host name to begin with digit	2.1	x					
Host names of up to 635 characters	2.1	x					
Host names of up to 255 characters	2.1		x				
Support dotted-decimal host numbers	2.1		x				
Check syntactically for dotted-dec first	2.1		x				
Map domain names per Section 6.1	2.2	x					
Cope with soft DNS errors	2.2	x					
Reasonable interval between retries	2.2	x					
Allow for long outages	2.2	x					
Expect WKS records to be available	2.2				x		
Try multiple addr's for remote multihomed host	2.3		x				
UDP reply src addr is specific dest of request	2.3		x				
Use same IP addr for related TCP connections	2.3		x				
Specify appropriate TOS values	2.4	x					
TOS values configurable	2.4	x					
Unused TOS bits zero	2.4	x					

3. REMOTE LOGIN -- TELNET PROTOCOL

3.1 INTRODUCTION

Telnet is the standard Internet application protocol for remote login. It provides the encoding rules to link a user's keyboard/display on a client ("user") system with a command interpreter on a remote server system. A subset of the Telnet protocol is also incorporated within other application protocols, e.g., FTP and SMTP.

Telnet uses a single TCP connection, and its normal data stream ("Network Virtual Terminal" or "NVT" mode) is 7-bit ASCII with escape sequences to embed control functions. Telnet also allows the negotiation of many optional modes and functions.

The primary Telnet specification is to be found in RFC-854 [TELNET:1], while the options are defined in many other RFCs; see Section 7 for references.

3.2 PROTOCOL WALK-THROUGH

3.2.1 Option Negotiation: RFC-854, pp. 2-3

Every Telnet implementation MUST include option negotiation and subnegotiation machinery [TELNET:2].

A host MUST carefully follow the rules of RFC-854 to avoid option-negotiation loops. A host MUST refuse (i.e, reply WONT/DONT to a DO/WILL) an unsupported option. Option negotiation SHOULD continue to function (even if all requests are refused) throughout the lifetime of a Telnet connection.

If all option negotiations fail, a Telnet implementation MUST default to, and support, an NVT.

DISCUSSION:
Even though more sophisticated "terminals" and supporting option negotiations are becoming the norm, all implementations must be prepared to support an NVT for any user-server communication.

3.2.2 Telnet Go-Ahead Function: RFC-854, p. 5, and RFC-858

On a host that never sends the Telnet command Go Ahead (GA), the Telnet Server MUST attempt to negotiate the Suppress Go Ahead option (i.e., send "WILL Suppress Go Ahead"). A User or Server Telnet MUST always accept negotiation of the Suppress Go

Ahead option.

When it is driving a full-duplex terminal for which GA has no meaning, a User Telnet implementation MAY ignore GA commands.

DISCUSSION:
Half-duplex ("locked-keyboard") line-at-a-time terminals for which the Go-Ahead mechanism was designed have largely disappeared from the scene. It turned out to be difficult to implement sending the Go-Ahead signal in many operating systems, even some systems that support native half-duplex terminals. The difficulty is typically that the Telnet server code does not have access to information about whether the user process is blocked awaiting input from the Telnet connection, i.e., it cannot reliably determine when to send a GA command. Therefore, most Telnet Server hosts do not send GA commands.

The effect of the rules in this section is to allow either end of a Telnet connection to veto the use of GA commands.

There is a class of half-duplex terminals that is still commercially important: "data entry terminals," which interact in a full-screen manner. However, supporting data entry terminals using the Telnet protocol does not require the Go Ahead signal; see Section 3.3.2.

3.2.3 Control Functions: RFC-854, pp. 7-8

The list of Telnet commands has been extended to include EOR (End-of-Record), with code 239 [TELNET:9].

Both User and Server Telnets MAY support the control functions EOR, EC, EL, and Break, and MUST support AO, AYT, DM, IP, NOP, SB, and SE.

A host MUST be able to receive and ignore any Telnet control functions that it does not support.

DISCUSSION:
Note that a Server Telnet is required to support the Telnet IP (Interrupt Process) function, even if the server host has an equivalent in-stream function (e.g., Control-C in many systems). The Telnet IP function may be stronger than an in-stream interrupt command, because of the out-of-band effect of TCP urgent data.

The EOR control function may be used to delimit the

stream. An important application is data entry terminal support (see Section 3.3.2). There was concern that since EOR had not been defined in RFC-854, a host that was not prepared to correctly ignore unknown Telnet commands might crash if it received an EOR. To protect such hosts, the End-of-Record option [TELNET:9] was introduced; however, a properly implemented Telnet program will not require this protection.

3.2.4 Telnet "Synch" Signal: RFC-854, pp. 8-10

When it receives "urgent" TCP data, a User or Server Telnet MUST discard all data except Telnet commands until the DM (and end of urgent) is reached.

When it sends Telnet IP (Interrupt Process), a User Telnet SHOULD follow it by the Telnet "Synch" sequence, i.e., send as TCP urgent data the sequence "IAC IP IAC DM". The TCP urgent pointer points to the DM octet.

When it receives a Telnet IP command, a Server Telnet MAY send a Telnet "Synch" sequence back to the user, to flush the output stream. The choice ought to be consistent with the way the server operating system behaves when a local user interrupts a process.

When it receives a Telnet AO command, a Server Telnet MUST send a Telnet "Synch" sequence back to the user, to flush the output stream.

A User Telnet SHOULD have the capability of flushing output when it sends a Telnet IP; see also Section 3.4.5.

DISCUSSION:

There are three possible ways for a User Telnet to flush the stream of server output data:

(1) Send AO after IP.

This will cause the server host to send a "flush-buffered-output" signal to its operating system. However, the AO may not take effect locally, i.e., stop terminal output at the User Telnet end, until the Server Telnet has received and processed the AO and has sent back a "Synch".

(2) Send DO TIMING-MARK [TELNET:7] after IP, and discard all output locally until a WILL/WONT TIMING-MARK is

received from the Server Telnet.

Since the DO TIMING-MARK will be processed after the IP at the server, the reply to it should be in the right place in the output data stream. However, the TIMING-MARK will not send a "flush buffered output" signal to the server operating system. Whether or not this is needed is dependent upon the server system.

(3) Do both.

The best method is not entirely clear, since it must accommodate a number of existing server hosts that do not follow the Telnet standards in various ways. The safest approach is probably to provide a user-controllable option to select (1), (2), or (3).

3.2.5 NVT Printer and Keyboard: RFC-854, p. 11

In NVT mode, a Telnet SHOULD NOT send characters with the high-order bit 1, and MUST NOT send it as a parity bit. Implementations that pass the high-order bit to applications SHOULD negotiate binary mode (see Section 3.2.6).

DISCUSSION:

Implementors should be aware that a strict reading of RFC-854 allows a client or server expecting NVT ASCII to ignore characters with the high-order bit set. In general, binary mode is expected to be used for transmission of an extended (beyond 7-bit) character set with Telnet.

However, there exist applications that really need an 8-bit NVT mode, which is currently not defined, and these existing applications do set the high-order bit during part or all of the life of a Telnet connection. Note that binary mode is not the same as 8-bit NVT mode, since binary mode turns off end-of-line processing. For this reason, the requirements on the high-order bit are stated as SHOULD, not MUST.

RFC-854 defines a minimal set of properties of a "network virtual terminal" or NVT; this is not meant to preclude additional features in a real terminal. A Telnet connection is fully transparent to all 7-bit ASCII characters, including arbitrary ASCII control characters.

For example, a terminal might support full-screen commands coded as ASCII escape sequences; a Telnet implementation would pass these sequences as uninterpreted data. Thus, an NVT should not be conceived as a terminal type of a highly-restricted device.

3.2.6 Telnet Command Structure: RFC-854, p. 13

Since options may appear at any point in the data stream, a Telnet escape character (known as IAC, with the value 255) to be sent as data MUST be doubled.

3.2.7 Telnet Binary Option: RFC-856

When the Binary option has been successfully negotiated, arbitrary 8-bit characters are allowed. However, the data stream MUST still be scanned for IAC characters, any embedded Telnet commands MUST be obeyed, and data bytes equal to IAC MUST be doubled. Other character processing (e.g., replacing CR by CR NUL or by CR LF) MUST NOT be done. In particular, there is no end-of-line convention (see Section 3.3.1) in binary mode.

DISCUSSION:

The Binary option is normally negotiated in both directions, to change the Telnet connection from NVT mode to "binary mode".

The sequence IAC EOR can be used to delimit blocks of data within a binary-mode Telnet stream.

3.2.8 Telnet Terminal-Type Option: RFC-1091

The Terminal-Type option MUST use the terminal type names officially defined in the Assigned Numbers RFC [INTRO:5], when they are available for the particular terminal. However, the receiver of a Terminal-Type option MUST accept any name.

DISCUSSION:

RFC-1091 [TELNET:10] updates an earlier version of the Terminal-Type option defined in RFC-930. The earlier version allowed a server host capable of supporting multiple terminal types to learn the type of a particular client's terminal, assuming that each physical terminal had an intrinsic type. However, today a "terminal" is often really a terminal emulator program running in a PC, perhaps capable of emulating a range of terminal types. Therefore, RFC-1091 extends the specification to allow a

more general terminal-type negotiation between User and Server Telnets.

3.3 SPECIFIC ISSUES

3.3.1 Telnet End-of-Line Convention

The Telnet protocol defines the sequence CR LF to mean "end-of-line". For terminal input, this corresponds to a command-completion or "end-of-line" key being pressed on a user terminal; on an ASCII terminal, this is the CR key, but it may also be labelled "Return" or "Enter".

When a Server Telnet receives the Telnet end-of-line sequence CR LF as input from a remote terminal, the effect MUST be the same as if the user had pressed the "end-of-line" key on a local terminal. On server hosts that use ASCII, in particular, receipt of the Telnet sequence CR LF must cause the same effect as a local user pressing the CR key on a local terminal. Thus, CR LF and CR NUL MUST have the same effect on an ASCII server host when received as input over a Telnet connection.

A User Telnet MUST be able to send any of the forms: CR LF, CR NUL, and LF. A User Telnet on an ASCII host SHOULD have a user-controllable mode to send either CR LF or CR NUL when the user presses the "end-of-line" key, and CR LF SHOULD be the default.

The Telnet end-of-line sequence CR LF MUST be used to send Telnet data that is not terminal-to-computer (e.g., for Server Telnet sending output, or the Telnet protocol incorporated another application protocol).

DISCUSSION:

To allow interoperability between arbitrary Telnet clients and servers, the Telnet protocol defined a standard representation for a line terminator. Since the ASCII character set includes no explicit end-of-line character, systems have chosen various representations, e.g., CR, LF, and the sequence CR LF. The Telnet protocol chose the CR LF sequence as the standard for network transmission.

Unfortunately, the Telnet protocol specification in RFC-854 [TELNET:1] has turned out to be somewhat ambiguous on what character(s) should be sent from client to server for the "end-of-line" key. The result has been a massive and continuing interoperability headache, made worse by various faulty implementations of both User and Server

Telnets.

Although the Telnet protocol is based on a perfectly symmetric model, in a remote login session the role of the user at a terminal differs from the role of the server host. For example, RFC-854 defines the meaning of CR, LF, and CR LF as output from the server, but does not specify what the User Telnet should send when the user presses the "end-of-line" key on the terminal; this turns out to be the point at issue.

When a user presses the "end-of-line" key, some User Telnet implementations send CR LF, while others send CR NUL (based on a different interpretation of the same sentence in RFC-854). These will be equivalent for a correctly-implemented ASCII server host, as discussed above. For other servers, a mode in the User Telnet is needed.

The existence of User Telnets that send only CR NUL when CR is pressed creates a dilemma for non-ASCII hosts: they can either treat CR NUL as equivalent to CR LF in input, thus precluding the possibility of entering a "bare" CR, or else lose complete interworking.

Suppose a user on host A uses Telnet to log into a server host B, and then execute B's User Telnet program to log into server host C. It is desirable for the Server/User Telnet combination on B to be as transparent as possible, i.e., to appear as if A were connected directly to C. In particular, correct implementation will make B transparent to Telnet end-of-line sequences, except that CR LF may be translated to CR NUL or vice versa.

IMPLEMENTATION:

To understand Telnet end-of-line issues, one must have at least a general model of the relationship of Telnet to the local operating system. The Server Telnet process is typically coupled into the terminal driver software of the operating system as a pseudo-terminal. A Telnet end-of-line sequence received by the Server Telnet must have the same effect as pressing the end-of-line key on a real locally-connected terminal.

Operating systems that support interactive character-at-a-time applications (e.g., editors) typically have two internal modes for their terminal I/O: a formatted mode, in which local conventions for end-of-line and other

formatting rules have been applied to the data stream, and a "raw" mode, in which the application has direct access to every character as it was entered. A Server Telnet must be implemented in such a way that these modes have the same effect for remote as for local terminals. For example, suppose a CR LF or CR NUL is received by the Server Telnet on an ASCII host. In raw mode, a CR character is passed to the application; in formatted mode, the local system's end-of-line convention is used.

3.3.2 Data Entry Terminals

DISCUSSION:

In addition to the line-oriented and character-oriented ASCII terminals for which Telnet was designed, there are several families of video display terminals that are sometimes known as "data entry terminals" or DETs. The IBM 3270 family is a well-known example.

Two Internet protocols have been designed to support generic DETs: SUPDUP [TELNET:16, TELNET:17], and the DET option [TELNET:18, TELNET:19]. The DET option drives a data entry terminal over a Telnet connection using (sub-) negotiation. SUPDUP is a completely separate terminal protocol, which can be entered from Telnet by negotiation. Although both SUPDUP and the DET option have been used successfully in particular environments, neither has gained general acceptance or wide implementation.

A different approach to DET interaction has been developed for supporting the IBM 3270 family through Telnet, although the same approach would be applicable to any DET. The idea is to enter a "native DET" mode, in which the native DET input/output stream is sent as binary data. The Telnet EOR command is used to delimit logical records (e.g., "screens") within this binary stream.

IMPLEMENTATION:

The rules for entering and leaving native DET mode are as follows:

- The Server uses the Terminal-Type option [TELNET:10] to learn that the client is a DET.
- It is conventional, but not required, that both ends negotiate the EOR option [TELNET:9].
- Both ends negotiate the Binary option [TELNET:3] to

enter native DET mode.

- When either end negotiates out of binary mode, the other end does too, and the mode then reverts to normal NVT.

3.3.3 Option Requirements

Every Telnet implementation MUST support the Binary option [TELNET:3] and the Suppress Go Ahead option [TELNET:5], and SHOULD support the Echo [TELNET:4], Status [TELNET:6], End-of-Record [TELNET:9], and Extended Options List [TELNET:8] options.

A User or Server Telnet SHOULD support the Window Size Option [TELNET:12] if the local operating system provides the corresponding capability.

DISCUSSION:
Note that the End-of-Record option only signifies that a Telnet can receive a Telnet EOR without crashing; therefore, every Telnet ought to be willing to accept negotiation of the End-of-Record option. See also the discussion in Section 3.2.3.

3.3.4 Option Initiation

When the Telnet protocol is used in a client/server situation, the server SHOULD initiate negotiation of the terminal interaction mode it expects.

DISCUSSION:
The Telnet protocol was defined to be perfectly symmetrical, but its application is generally asymmetric. Remote login has been known to fail because NEITHER side initiated negotiation of the required non-default terminal modes. It is generally the server that determines the preferred mode, so the server needs to initiate the negotiation; since the negotiation is symmetric, the user can also initiate it.

A client (User Telnet) SHOULD provide a means for users to enable and disable the initiation of option negotiation.

DISCUSSION:
A user sometimes needs to connect to an application service (e.g., FTP or SMTP) that uses Telnet for its

control stream but does not support Telnet options. User Telnet may be used for this purpose if initiation of option negotiation is disabled.

3.3.5 Telnet Linemode Option

DISCUSSION:

An important new Telnet option, LINEMODE [TELNET:12], has been proposed. The LINEMODE option provides a standard way for a User Telnet and a Server Telnet to agree that the client rather than the server will perform terminal character processing. When the client has prepared a complete line of text, it will send it to the server in (usually) one TCP packet. This option will greatly decrease the packet cost of Telnet sessions and will also give much better user response over congested or long-delay networks.

The LINEMODE option allows dynamic switching between local and remote character processing. For example, the Telnet connection will automatically negotiate into single-character mode while a full screen editor is running, and then return to linemode when the editor is finished.

We expect that when this RFC is released, hosts should implement the client side of this option, and may implement the server side of this option. To properly implement the server side, the server needs to be able to tell the local system not to do any input character processing, but to remember its current terminal state and notify the Server Telnet process whenever the state changes. This will allow password echoing and full screen editors to be handled properly, for example.

3.4 TELNET/USER INTERFACE

3.4.1 Character Set Transparency

User Telnet implementations SHOULD be able to send or receive any 7-bit ASCII character. Where possible, any special character interpretations by the user host's operating system SHOULD be bypassed so that these characters can conveniently be sent and received on the connection.

Some character value MUST be reserved as "escape to command mode"; conventionally, doubling this character allows it to be entered as data. The specific character used SHOULD be user selectable.

On binary-mode connections, a User Telnet program MAY provide an escape mechanism for entering arbitrary 8-bit values, if the host operating system doesn't allow them to be entered directly from the keyboard.

IMPLEMENTATION:
The transparency issues are less pressing on servers, but implementors should take care in dealing with issues like: masking off parity bits (sent by an older, non-conforming client) before they reach programs that expect only NVT ASCII, and properly handling programs that request 8-bit data streams.

3.4.2 Telnet Commands

A User Telnet program MUST provide a user the capability of entering any of the Telnet control functions IP, AO, or AYT, and SHOULD provide the capability of entering EC, EL, and Break.

3.4.3 TCP Connection Errors

A User Telnet program SHOULD report to the user any TCP errors that are reported by the transport layer (see "TCP/Application Layer Interface" section in [INTRO:1]).

3.4.4 Non-Default Telnet Contact Port

A User Telnet program SHOULD allow the user to optionally specify a non-standard contact port number at the Server Telnet host.

3.4.5 Flushing Output

A User Telnet program SHOULD provide the user the ability to specify whether or not output should be flushed when an IP is sent; see Section 3.2.4.

For any output flushing scheme that causes the User Telnet to flush output locally until a Telnet signal is received from the Server, there SHOULD be a way for the user to manually restore normal output, in case the Server fails to send the expected signal.

3.5. TELNET REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Option Negotiation	3.2.1	x					
Avoid negotiation loops	3.2.1	x					
Refuse unsupported options	3.2.1	x					
Negotiation OK anytime on connection	3.2.1		x				
Default to NVT	3.2.1	x					
Send official name in Term-Type option	3.2.8	x					
Accept any name in Term-Type option	3.2.8	x					
Implement Binary, Suppress-GA options	3.3.3	x					
Echo, Status, EOL, Ext-Opt-List options	3.3.3		x				
Implement Window-Size option if appropriate	3.3.3		x				
Server initiate mode negotiations	3.3.4		x				
User can enable/disable init negotiations	3.3.4		x				
Go-Aheads							
Non-GA server negotiate SUPPRESS-GA option	3.2.2	x					
User or Server accept SUPPRESS-GA option	3.2.2	x					
User Telnet ignore GA's	3.2.2			x			
Control Functions							
Support SE NOP DM IP AO AYT SB	3.2.3	x					
Support EOR EC EL Break	3.2.3			x			
Ignore unsupported control functions	3.2.3	x					
User, Server discard urgent data up to DM	3.2.4	x					
User Telnet send "Synch" after IP, AO, AYT	3.2.4		x				
Server Telnet reply Synch to IP	3.2.4			x			
Server Telnet reply Synch to AO	3.2.4	x					
User Telnet can flush output when send IP	3.2.4		x				
Encoding							
Send high-order bit in NVT mode	3.2.5				x		
Send high-order bit as parity bit	3.2.5					x	
Negot. BINARY if pass high-ord. bit to applic	3.2.5		x				
Always double IAC data byte	3.2.6	x					

Double IAC data byte in binary mode	3.2.7	x				
Obey Telnet cmds in binary mode	3.2.7	x				
End-of-line, CR NUL in binary mode	3.2.7					x
End-of-Line						
EOL at Server same as local end-of-line	3.3.1	x				
ASCII Server accept CR LF or CR NUL for EOL	3.3.1	x				
User Telnet able to send CR LF, CR NUL, or LF	3.3.1	x				
ASCII user able to select CR LF/CR NUL	3.3.1		x			
User Telnet default mode is CR LF	3.3.1		x			
Non-interactive uses CR LF for EOL	3.3.1	x				
User Telnet interface						
Input & output all 7-bit characters	3.4.1		x			
Bypass local op sys interpretation	3.4.1		x			
Escape character	3.4.1	x				
User-settable escape character	3.4.1		x			
Escape to enter 8-bit values	3.4.1			x		
Can input IP, AO, AYT	3.4.2	x				
Can input EC, EL, Break	3.4.2		x			
Report TCP connection errors to user	3.4.3		x			
Optional non-default contact port	3.4.4		x			
Can spec: output flushed when IP sent	3.4.5		x			
Can manually restore output mode	3.4.5		x			

4. FILE TRANSFER

4.1 FILE TRANSFER PROTOCOL -- FTP

4.1.1 INTRODUCTION

The File Transfer Protocol FTP is the primary Internet standard for file transfer. The current specification is contained in RFC-959 [FTP:1].

FTP uses separate simultaneous TCP connections for control and for data transfer. The FTP protocol includes many features, some of which are not commonly implemented. However, for every feature in FTP, there exists at least one implementation. The minimum implementation defined in RFC-959 was too small, so a somewhat larger minimum implementation is defined here.

Internet users have been unnecessarily burdened for years by deficient FTP implementations. Protocol implementors have suffered from the erroneous opinion that implementing FTP ought to be a small and trivial task. This is wrong, because FTP has a user interface, because it has to deal (correctly) with the whole variety of communication and operating system errors that may occur, and because it has to handle the great diversity of real file systems in the world.

4.1.2. PROTOCOL WALK-THROUGH

4.1.2.1 LOCAL Type: RFC-959 Section 3.1.1.4

An FTP program MUST support TYPE I ("IMAGE" or binary type) as well as TYPE L 8 ("LOCAL" type with logical byte size 8). A machine whose memory is organized into m-bit words, where m is not a multiple of 8, MAY also support TYPE L m.

DISCUSSION:

The command "TYPE L 8" is often required to transfer binary data between a machine whose memory is organized into (e.g.) 36-bit words and a machine with an 8-bit byte organization. For an 8-bit byte machine, TYPE L 8 is equivalent to IMAGE.

"TYPE L m" is sometimes specified to the FTP programs on two m-bit word machines to ensure the correct transfer of a native-mode binary file from one machine to the other. However, this command should have the same effect on these machines as "TYPE I".

4.1.2.2 Telnet Format Control: RFC-959 Section 3.1.1.5.2

A host that makes no distinction between TYPE N and TYPE T SHOULD implement TYPE T to be identical to TYPE N.

DISCUSSION:
This provision should ease interoperation with hosts that do make this distinction.

Many hosts represent text files internally as strings of ASCII characters, using the embedded ASCII format effector characters (LF, BS, FF, ...) to control the format when a file is printed. For such hosts, there is no distinction between "print" files and other files. However, systems that use record structured files typically need a special format for printable files (e.g., ASA carriage control). For the latter hosts, FTP allows a choice of TYPE N or TYPE T.

4.1.2.3 Page Structure: RFC-959 Section 3.1.2.3 and Appendix I

Implementation of page structure is NOT RECOMMENDED in general. However, if a host system does need to implement FTP for "random access" or "holey" files, it MUST use the defined page structure format rather than define a new private FTP format.

4.1.2.4 Data Structure Transformations: RFC-959 Section 3.1.2

An FTP transformation between record-structure and file-structure SHOULD be invertible, to the extent possible while making the result useful on the target host.

DISCUSSION:
RFC-959 required strict invertibility between record-structure and file-structure, but in practice, efficiency and convenience often preclude it. Therefore, the requirement is being relaxed. There are two different objectives for transferring a file: processing it on the target host, or just storage. For storage, strict invertibility is important. For processing, the file created on the target host needs to be in the format expected by application programs on that host.

As an example of the conflict, imagine a record-oriented operating system that requires some data files to have exactly 80 bytes in each record. While STORing

a file on such a host, an FTP Server must be able to pad each line or record to 80 bytes; a later retrieval of such a file cannot be strictly invertible.

4.1.2.5 Data Connection Management: RFC-959 Section 3.3

A User-FTP that uses STREAM mode SHOULD send a PORT command to assign a non-default data port before each transfer command is issued.

DISCUSSION:
This is required because of the long delay after a TCP connection is closed until its socket pair can be reused, to allow multiple transfers during a single FTP session. Sending a port command can avoided if a transfer mode other than stream is used, by leaving the data transfer connection open between transfers.

4.1.2.6 PASV Command: RFC-959 Section 4.1.2

A server-FTP MUST implement the PASV command.

If multiple third-party transfers are to be executed during the same session, a new PASV command MUST be issued before each transfer command, to obtain a unique port pair.

IMPLEMENTATION:
The format of the 227 reply to a PASV command is not well standardized. In particular, an FTP client cannot assume that the parentheses shown on page 40 of RFC-959 will be present (and in fact, Figure 3 on page 43 omits them). Therefore, a User-FTP program that interprets the PASV reply must scan the reply for the first digit of the host and port numbers.

Note that the host number h1,h2,h3,h4 is the IP address of the server host that is sending the reply, and that p1,p2 is a non-default data transfer port that PASV has assigned.

4.1.2.7 LIST and NLST Commands: RFC-959 Section 4.1.3

The data returned by an NLST command MUST contain only a simple list of legal pathnames, such that the server can use them directly as the arguments of subsequent data transfer commands for the individual files.

The data returned by a LIST or NLST command SHOULD use an

implied TYPE AN, unless the current type is EBCDIC, in which case an implied TYPE EN SHOULD be used.

DISCUSSION:
Many FTP clients support macro-commands that will get or put files matching a wildcard specification, using NLST to obtain a list of pathnames. The expansion of "multiple-put" is local to the client, but "multiple-get" requires cooperation by the server.

The implied type for LIST and NLST is designed to provide compatibility with existing User-FTPs, and in particular with multiple-get commands.

4.1.2.8 SITE Command: RFC-959 Section 4.1.3

A Server-FTP SHOULD use the SITE command for non-standard features, rather than invent new private commands or unstandardized extensions to existing commands.

4.1.2.9 STOU Command: RFC-959 Section 4.1.3

The STOU command stores into a uniquely named file. When it receives an STOU command, a Server-FTP MUST return the actual file name in the "125 Transfer Starting" or the "150 Opening Data Connection" message that precedes the transfer (the 250 reply code mentioned in RFC-959 is incorrect). The exact format of these messages is hereby defined to be as follows:

125 FILE: pppp
150 FILE: pppp

where pppp represents the unique pathname of the file that will be written.

4.1.2.10 Telnet End-of-line Code: RFC-959, Page 34

Implementors MUST NOT assume any correspondence between READ boundaries on the control connection and the Telnet EOL sequences (CR LF).

DISCUSSION:
Thus, a server-FTP (or User-FTP) must continue reading characters from the control connection until a complete Telnet EOL sequence is encountered, before processing the command (or response, respectively). Conversely, a single READ from the control connection may include

more than one FTP command.

4.1.2.11 FTP Replies: RFC-959 Section 4.2, Page 35

A Server-FTP MUST send only correctly formatted replies on the control connection. Note that RFC-959 (unlike earlier versions of the FTP spec) contains no provision for a "spontaneous" reply message.

A Server-FTP SHOULD use the reply codes defined in RFC-959 whenever they apply. However, a server-FTP MAY use a different reply code when needed, as long as the general rules of Section 4.2 are followed. When the implementor has a choice between a 4xx and 5xx reply code, a Server-FTP SHOULD send a 4xx (temporary failure) code when there is any reasonable possibility that a failed FTP will succeed a few hours later.

A User-FTP SHOULD generally use only the highest-order digit of a 3-digit reply code for making a procedural decision, to prevent difficulties when a Server-FTP uses non-standard reply codes.

A User-FTP MUST be able to handle multi-line replies. If the implementation imposes a limit on the number of lines and if this limit is exceeded, the User-FTP MUST recover, e.g., by ignoring the excess lines until the end of the multi-line reply is reached.

A User-FTP SHOULD NOT interpret a 421 reply code ("Service not available, closing control connection") specially, but SHOULD detect closing of the control connection by the server.

DISCUSSION:

Server implementations that fail to strictly follow the reply rules often cause FTP user programs to hang. Note that RFC-959 resolved ambiguities in the reply rules found in earlier FTP specifications and must be followed.

It is important to choose FTP reply codes that properly distinguish between temporary and permanent failures, to allow the successful use of file transfer client daemons. These programs depend on the reply codes to decide whether or not to retry a failed transfer; using a permanent failure code (5xx) for a temporary error will cause these programs to give up unnecessarily.

When the meaning of a reply matches exactly the text shown in RFC-959, uniformity will be enhanced by using the RFC-959 text verbatim. However, a Server-FTP implementor is encouraged to choose reply text that conveys specific system-dependent information, when appropriate.

4.1.2.12 Connections: RFC-959 Section 5.2

The words "and the port used" in the second paragraph of this section of RFC-959 are erroneous (historical), and they should be ignored.

On a multihomed server host, the default data transfer port (L-1) MUST be associated with the same local IP address as the corresponding control connection to port L.

A user-FTP MUST NOT send any Telnet controls other than SYNCH and IP on an FTP control connection. In particular, it MUST NOT attempt to negotiate Telnet options on the control connection. However, a server-FTP MUST be capable of accepting and refusing Telnet negotiations (i.e., sending DONT/WONT).

DISCUSSION:

Although the RFC says: "Server- and User- processes should follow the conventions for the Telnet protocol...[on the control connection]", it is not the intent that Telnet option negotiation is to be employed.

4.1.2.13 Minimum Implementation; RFC-959 Section 5.1

The following commands and options MUST be supported by every server-FTP and user-FTP, except in cases where the underlying file system or operating system does not allow or support a particular command.

Type: ASCII Non-print, IMAGE, LOCAL 8
Mode: Stream
Structure: File, Record*
Commands:
USER, PASS, ACCT,
PORT, PASV,
TYPE, MODE, STRU,
RETR, STOR, APPE,
RNFR, RNTO, DELE,
CWD, CDUP, RMD, MKD, PWD,

LIST, NLST,
SYST, STAT,
HELP, NOOP, QUIT.

*Record structure is REQUIRED only for hosts whose file systems support record structure.

DISCUSSION:
Vendors are encouraged to implement a larger subset of the protocol. For example, there are important robustness features in the protocol (e.g., Restart, ABOR, block mode) that would be an aid to some Internet users but are not widely implemented.

A host that does not have record structures in its file system may still accept files with STRU R, recording the byte stream literally.

4.1.3 SPECIFIC ISSUES

4.1.3.1 Non-standard Command Verbs

FTP allows "experimental" commands, whose names begin with "X". If these commands are subsequently adopted as standards, there may still be existing implementations using the "X" form. At present, this is true for the directory commands:

RFC-959 "Experimental"

MKD	XMKD
RMD	XRMD
PWD	XPWD
CDUP	XCUP
CWD	XCWD

All FTP implementations SHOULD recognize both forms of these commands, by simply equating them with extra entries in the command lookup table.

IMPLEMENTATION:
A User-FTP can access a server that supports only the "X" forms by implementing a mode switch, or automatically using the following procedure: if the RFC-959 form of one of the above commands is rejected with a 500 or 502 response code, then try the experimental form; any other response would be passed to the user.

4.1.3.2 Idle Timeout

A Server-FTP process SHOULD have an idle timeout, which will terminate the process and close the control connection if the server is inactive (i.e., no command or data transfer in progress) for a long period of time. The idle timeout time SHOULD be configurable, and the default should be at least 5 minutes.

A client FTP process ("User-PI" in RFC-959) will need timeouts on responses only if it is invoked from a program.

DISCUSSION:

Without a timeout, a Server-FTP process may be left pending indefinitely if the corresponding client crashes without closing the control connection.

4.1.3.3 Concurrency of Data and Control

DISCUSSION:

The intent of the designers of FTP was that a user should be able to send a STAT command at any time while data transfer was in progress and that the server-FTP would reply immediately with status -- e.g., the number of bytes transferred so far. Similarly, an ABOR command should be possible at any time during a data transfer.

Unfortunately, some small-machine operating systems make such concurrent programming difficult, and some other implementers seek minimal solutions, so some FTP implementations do not allow concurrent use of the data and control connections. Even such a minimal server must be prepared to accept and defer a STAT or ABOR command that arrives during data transfer.

4.1.3.4 FTP Restart Mechanism

The description of the 110 reply on pp. 40-41 of RFC-959 is incorrect; the correct description is as follows. A restart reply message, sent over the control connection from the receiving FTP to the User-FTP, has the format:

110 MARK ssss = rrrr

Here:

* ssss is a text string that appeared in a Restart Marker

in the data stream and encodes a position in the sender's file system;

* rrrr encodes the corresponding position in the receiver's file system.

The encoding, which is specific to a particular file system and network implementation, is always generated and interpreted by the same system, either sender or receiver.

When an FTP that implements restart receives a Restart Marker in the data stream, it SHOULD force the data to that point to be written to stable storage before encoding the corresponding position rrrr. An FTP sending Restart Markers MUST NOT assume that 110 replies will be returned synchronously with the data, i.e., it must not await a 110 reply before sending more data.

Two new reply codes are hereby defined for errors encountered in restarting a transfer:

554 Requested action not taken: invalid REST parameter.

A 554 reply may result from a FTP service command that follows a REST command. The reply indicates that the existing file at the Server-FTP cannot be repositioned as specified in the REST.

555 Requested action not taken: type or stru mismatch.

A 555 reply may result from an APPE command or from any FTP service command following a REST command. The reply indicates that there is some mismatch between the current transfer parameters (type and stru) and the attributes of the existing file.

DISCUSSION:

Note that the FTP Restart mechanism requires that Block or Compressed mode be used for data transfer, to allow the Restart Markers to be included within the data stream. The frequency of Restart Markers can be low.

Restart Markers mark a place in the data stream, but the receiver may be performing some transformation on the data as it is stored into stable storage. In general, the receiver's encoding must include any state information necessary to restart this transformation at any point of the FTP data stream. For example, in TYPE

A transfers, some receiver hosts transform CR LF sequences into a single LF character on disk. If a Restart Marker happens to fall between CR and LF, the receiver must encode in rrrr that the transfer must be restarted in a "CR has been seen and discarded" state.

Note that the Restart Marker is required to be encoded as a string of printable ASCII characters, regardless of the type of the data.

RFC-959 says that restart information is to be returned "to the user". This should not be taken literally. In general, the User-FTP should save the restart information (ssss,rrrr) in stable storage, e.g., append it to a restart control file. An empty restart control file should be created when the transfer first starts and deleted automatically when the transfer completes successfully. It is suggested that this file have a name derived in an easily-identifiable manner from the name of the file being transferred and the remote host name; this is analogous to the means used by many text editors for naming "backup" files.

There are three cases for FTP restart.

(1) User-to-Server Transfer

The User-FTP puts Restart Markers <ssss> at convenient places in the data stream. When the Server-FTP receives a Marker, it writes all prior data to disk, encodes its file system position and transformation state as rrrr, and returns a "110 MARK ssss = rrrr" reply over the control connection. The User-FTP appends the pair (ssss,rrrr) to its restart control file.

To restart the transfer, the User-FTP fetches the last (ssss,rrrr) pair from the restart control file, repositions its local file system and transformation state using ssss, and sends the command "REST rrrr" to the Server-FTP.

(2) Server-to-User Transfer

The Server-FTP puts Restart Markers <ssss> at convenient places in the data stream. When the User-FTP receives a Marker, it writes all prior data to disk, encodes its file system position and

transformation state as rrrr, and appends the pair (rrrr,ssss) to its restart control file.

To restart the transfer, the User-FTP fetches the last (rrrr,ssss) pair from the restart control file, repositions its local file system and transformation state using rrrr, and sends the command "REST ssss" to the Server-FTP.

(3) Server-to-Server ("Third-Party") Transfer

The sending Server-FTP puts Restart Markers <ssss> at convenient places in the data stream. When it receives a Marker, the receiving Server-FTP writes all prior data to disk, encodes its file system position and transformation state as rrrr, and sends a "110 MARK ssss = rrrr" reply over the control connection to the User. The User-FTP appends the pair (ssss,rrrr) to its restart control file.

To restart the transfer, the User-FTP fetches the last (ssss,rrrr) pair from the restart control file, sends "REST ssss" to the sending Server-FTP, and sends "REST rrrr" to the receiving Server-FTP.

4.1.4 FTP/USER INTERFACE

This section discusses the user interface for a User-FTP program.

4.1.4.1 Pathname Specification

Since FTP is intended for use in a heterogeneous environment, User-FTP implementations MUST support remote pathnames as arbitrary character strings, so that their form and content are not limited by the conventions of the local operating system.

DISCUSSION:
In particular, remote pathnames can be of arbitrary length, and all the printing ASCII characters as well as space (0x20) must be allowed. RFC-959 allows a pathname to contain any 7-bit ASCII character except CR or LF.

4.1.4.2 "QUOTE" Command

A User-FTP program MUST implement a "QUOTE" command that will pass an arbitrary character string to the server and display all resulting response messages to the user.

To make the "QUOTE" command useful, a User-FTP SHOULD send transfer control commands to the server as the user enters them, rather than saving all the commands and sending them to the server only when a data transfer is started.

DISCUSSION:

The "QUOTE" command is essential to allow the user to access servers that require system-specific commands (e.g., SITE or ALLO), or to invoke new or optional features that are not implemented by the User-FTP. For example, "QUOTE" may be used to specify "TYPE A T" to send a print file to hosts that require the distinction, even if the User-FTP does not recognize that TYPE.

4.1.4.3 Displaying Replies to User

A User-FTP SHOULD display to the user the full text of all error reply messages it receives. It SHOULD have a "verbose" mode in which all commands it sends and the full text and reply codes it receives are displayed, for diagnosis of problems.

4.1.4.4 Maintaining Synchronization

The state machine in a User-FTP SHOULD be forgiving of missing and unexpected reply messages, in order to maintain command synchronization with the server.

4.1.5 FTP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Implement TYPE T if same as TYPE N	4.1.2.2		x				
File/Record transform invertible if poss.	4.1.2.4		x				
User-FTP send PORT cmd for stream mode	4.1.2.5		x				
Server-FTP implement PASV	4.1.2.6	x					
PASV is per-transfer	4.1.2.6	x					
NLST reply usable in RETR cmds	4.1.2.7	x					
Implied type for LIST and NLST	4.1.2.7		x				
SITE cmd for non-standard features	4.1.2.8		x				
STOU cmd return pathname as specified	4.1.2.9	x					
Use TCP READ boundaries on control conn.	4.1.2.10					x	
Server-FTP send only correct reply format	4.1.2.11	x					
Server-FTP use defined reply code if poss.	4.1.2.11		x				
New reply code following Section 4.2	4.1.2.11			x			
User-FTP use only high digit of reply	4.1.2.11		x				
User-FTP handle multi-line reply lines	4.1.2.11	x					
User-FTP handle 421 reply specially	4.1.2.11				x		
Default data port same IP addr as ctl conn	4.1.2.12	x					
User-FTP send Telnet cmds exc. SYNCH, IP	4.1.2.12					x	
User-FTP negotiate Telnet options	4.1.2.12					x	
Server-FTP handle Telnet options	4.1.2.12	x					
Handle "Experimental" directory cmds	4.1.3.1		x				
Idle timeout in server-FTP	4.1.3.2		x				
Configurable idle timeout	4.1.3.2		x				
Receiver checkpoint data at Restart Marker	4.1.3.4		x				
Sender assume 110 replies are synchronous	4.1.3.4					x	
Support TYPE:							
ASCII - Non-Print (AN)	4.1.2.13	x					
ASCII - Telnet (AT) -- if same as AN	4.1.2.2		x				
ASCII - Carriage Control (AC)	959 3.1.1.5.2			x			
EBCDIC - (any form)	959 3.1.1.2			x			
IMAGE	4.1.2.1	x					
LOCAL 8	4.1.2.1	x					

LOCAL m	4.1.2.1			x		2
Support MODE:						
Stream	4.1.2.13	x				
Block	959 3.4.2			x		
Support STRUCTURE:						
File	4.1.2.13	x				
Record	4.1.2.13	x				3
Page	4.1.2.3				x	
Support commands:						
USER	4.1.2.13	x				
PASS	4.1.2.13	x				
ACCT	4.1.2.13	x				
CWD	4.1.2.13	x				
CDUP	4.1.2.13	x				
SMNT	959 5.3.1			x		
REIN	959 5.3.1			x		
QUIT	4.1.2.13	x				
PORT	4.1.2.13	x				
PASV	4.1.2.6	x				
TYPE	4.1.2.13	x				1
STRU	4.1.2.13	x				1
MODE	4.1.2.13	x				1
RETR	4.1.2.13	x				
STOR	4.1.2.13	x				
STOU	959 5.3.1			x		
APPE	4.1.2.13	x				
ALLO	959 5.3.1			x		
REST	959 5.3.1			x		
RNFR	4.1.2.13	x				
RNTO	4.1.2.13	x				
ABOR	959 5.3.1			x		
DELE	4.1.2.13	x				
RMD	4.1.2.13	x				
MKD	4.1.2.13	x				
PWD	4.1.2.13	x				
LIST	4.1.2.13	x				
NLST	4.1.2.13	x				
SITE	4.1.2.8			x		
STAT	4.1.2.13	x				
SYST	4.1.2.13	x				
HELP	4.1.2.13	x				
NOOP	4.1.2.13	x				

User Interface:						
Arbitrary pathnames	4.1.4.1	x				
Implement "QUOTE" command	4.1.4.2	x				
Transfer control commands immediately	4.1.4.2		x			
Display error messages to user	4.1.4.3		x			
Verbose mode	4.1.4.3		x			
Maintain synchronization with server	4.1.4.4		x			

Footnotes:

(1) For the values shown earlier.

(2) Here m is number of bits in a memory word.

(3) Required for host with record-structured file system, optional otherwise.

4.2 TRIVIAL FILE TRANSFER PROTOCOL -- TFTP

4.2.1 INTRODUCTION

The Trivial File Transfer Protocol TFTP is defined in RFC-783 [TFTP:1].

TFTP provides its own reliable delivery with UDP as its transport protocol, using a simple stop-and-wait acknowledgment system. Since TFTP has an effective window of only one 512 octet segment, it can provide good performance only over paths that have a small delay*bandwidth product. The TFTP file interface is very simple, providing no access control or security.

TFTP's most important application is bootstrapping a host over a local network, since it is simple and small enough to be easily implemented in EPROM [BOOT:1, BOOT:2]. Vendors are urged to support TFTP for booting.

4.2.2 PROTOCOL WALK-THROUGH

The TFTP specification [TFTP:1] is written in an open style, and does not fully specify many parts of the protocol.

4.2.2.1 Transfer Modes: RFC-783, Page 3

The transfer mode "mail" SHOULD NOT be supported.

4.2.2.2 UDP Header: RFC-783, Page 17

The Length field of a UDP header is incorrectly defined; it includes the UDP header length (8).

4.2.3 SPECIFIC ISSUES

4.2.3.1 Sorcerer's Apprentice Syndrome

There is a serious bug, known as the "Sorcerer's Apprentice Syndrome," in the protocol specification. While it does not cause incorrect operation of the transfer (the file will always be transferred correctly if the transfer completes), this bug may cause excessive retransmission, which may cause the transfer to time out.

Implementations MUST contain the fix for this problem: the sender (i.e., the side originating the DATA packets) must never resend the current DATA packet on receipt of a

duplicate ACK.

DISCUSSION:

The bug is caused by the protocol rule that either side, on receiving an old duplicate datagram, may resend the current datagram. If a packet is delayed in the network but later successfully delivered after either side has timed out and retransmitted a packet, a duplicate copy of the response may be generated. If the other side responds to this duplicate with a duplicate of its own, then every datagram will be sent in duplicate for the remainder of the transfer (unless a datagram is lost, breaking the repetition). Worse yet, since the delay is often caused by congestion, this duplicate transmission will usually causes more congestion, leading to more delayed packets, etc.

The following example may help to clarify this problem.

```
    TFTP A                          TFTP B

(1) Receive ACK X-1
    Send DATA X
(2)                                 Receive DATA X
                                    Send ACK X
       (ACK X is delayed in network,
        and  A times out):
(3) Retransmit DATA X

(4)                                 Receive DATA X again
                                    Send ACK X again
(5) Receive (delayed) ACK X
    Send DATA X+1
(6)                                 Receive DATA X+1
                                    Send ACK X+1
(7) Receive ACK X again
    Send DATA X+1 again
(8)                                 Receive DATA X+1 again
                                    Send ACK X+1 again
(9) Receive ACK X+1
    Send DATA X+2
(10)                                Receive DATA X+2
                                    Send ACK X+3
(11) Receive ACK X+1 again
     Send DATA X+2 again
(12)                                Receive DATA X+2 again
                                    Send ACK X+3 again
```

Notice that once the delayed ACK arrives, the protocol settles down to duplicate all further packets (sequences 5-8 and 9-12). The problem is caused not by either side timing out, but by both sides retransmitting the current packet when they receive a duplicate.

The fix is to break the retransmission loop, as indicated above. This is analogous to the behavior of TCP. It is then possible to remove the retransmission timer on the receiver, since the resent ACK will never cause any action; this is a useful simplification where TFTP is used in a bootstrap program. It is OK to allow the timer to remain, and it may be helpful if the retransmitted ACK replaces one that was genuinely lost in the network. The sender still requires a retransmit timer, of course.

4.2.3.2 Timeout Algorithms

A TFTP implementation MUST use an adaptive timeout.

IMPLEMENTATION:
TCP retransmission algorithms provide a useful base to work from. At least an exponential backoff of retransmission timeout is necessary.

4.2.3.3 Extensions

A variety of non-standard extensions have been made to TFTP, including additional transfer modes and a secure operation mode (with passwords). None of these have been standardized.

4.2.3.4 Access Control

A server TFTP implementation SHOULD include some configurable access control over what pathnames are allowed in TFTP operations.

4.2.3.5 Broadcast Request

A TFTP request directed to a broadcast address SHOULD be silently ignored.

DISCUSSION:
Due to the weak access control capability of TFTP, directed broadcasts of TFTP requests to random networks

could create a significant security hole.

4.2.4 TFTP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Fix Sorcerer's Apprentice Syndrome	4.2.3.1	x					
Transfer modes:							
netascii	RFC-783	x					
octet	RFC-783	x					
mail	4.2.2.1				x		
extensions	4.2.3.3			x			
Use adaptive timeout	4.2.3.2	x					
Configurable access control	4.2.3.4		x				
Silently ignore broadcast request	4.2.3.5		x				

5. ELECTRONIC MAIL -- SMTP and RFC-822

5.1 INTRODUCTION

In the TCP/IP protocol suite, electronic mail in a format specified in RFC-822 [SMTP:2] is transmitted using the Simple Mail Transfer Protocol (SMTP) defined in RFC-821 [SMTP:1].

While SMTP has remained unchanged over the years, the Internet community has made several changes in the way SMTP is used. In particular, the conversion to the Domain Name System (DNS) has caused changes in address formats and in mail routing. In this section, we assume familiarity with the concepts and terminology of the DNS, whose requirements are given in Section 6.1.

RFC-822 specifies the Internet standard format for electronic mail messages. RFC-822 supercedes an older standard, RFC-733, that may still be in use in a few places, although it is obsolete. The two formats are sometimes referred to simply by number ("822" and "733").

RFC-822 is used in some non-Internet mail environments with different mail transfer protocols than SMTP, and SMTP has also been adapted for use in some non-Internet environments. Note that this document presents the rules for the use of SMTP and RFC-822 for the Internet environment only; other mail environments that use these protocols may be expected to have their own rules.

5.2 PROTOCOL WALK-THROUGH

This section covers both RFC-821 and RFC-822.

The SMTP specification in RFC-821 is clear and contains numerous examples, so implementors should not find it difficult to understand. This section simply updates or annotates portions of RFC-821 to conform with current usage.

RFC-822 is a long and dense document, defining a rich syntax. Unfortunately, incomplete or defective implementations of RFC-822 are common. In fact, nearly all of the many formats of RFC-822 are actually used, so an implementation generally needs to recognize and correctly interpret all of the RFC-822 syntax.

5.2.1 The SMTP Model: RFC-821 Section 2

DISCUSSION:

Mail is sent by a series of request/response transactions between a client, the "sender-SMTP," and a server, the

"receiver-SMTP". These transactions pass (1) the message proper, which is composed of header and body, and (2) SMTP source and destination addresses, referred to as the "envelope".

The SMTP programs are analogous to Message Transfer Agents (MTAs) of X.400. There will be another level of protocol software, closer to the end user, that is responsible for composing and analyzing RFC-822 message headers; this component is known as the "User Agent" in X.400, and we use that term in this document. There is a clear logical distinction between the User Agent and the SMTP implementation, since they operate on different levels of protocol. Note, however, that this distinction is may not be exactly reflected the structure of typical implementations of Internet mail. Often there is a program known as the "mailer" that implements SMTP and also some of the User Agent functions; the rest of the User Agent functions are included in a user interface used for entering and reading mail.

The SMTP envelope is constructed at the originating site, typically by the User Agent when the message is first queued for the Sender-SMTP program. The envelope addresses may be derived from information in the message header, supplied by the user interface (e.g., to implement a bcc: request), or derived from local configuration information (e.g., expansion of a mailing list). The SMTP envelope cannot in general be re-derived from the header at a later stage in message delivery, so the envelope is transmitted separately from the message itself using the MAIL and RCPT commands of SMTP.

The text of RFC-821 suggests that mail is to be delivered to an individual user at a host. With the advent of the domain system and of mail routing using mail-exchange (MX) resource records, implementors should now think of delivering mail to a user at a domain, which may or may not be a particular host. This DOES NOT change the fact that SMTP is a host-to-host mail exchange protocol.

5.2.2 Canonicalization: RFC-821 Section 3.1

The domain names that a Sender-SMTP sends in MAIL and RCPT commands MUST have been "canonicalized," i.e., they must be fully-qualified principal names or domain literals, not nicknames or domain abbreviations. A canonicalized name either identifies a host directly or is an MX name; it cannot be a

CNAME.

5.2.3 VRFY and EXPN Commands: RFC-821 Section 3.3

A receiver-SMTP MUST implement VRFY and SHOULD implement EXPN (this requirement overrides RFC-821). However, there MAY be configuration information to disable VRFY and EXPN in a particular installation; this might even allow EXPN to be disabled for selected lists.

A new reply code is defined for the VRFY command:

252 Cannot VRFY user (e.g., info is not local), but will take message for this user and attempt delivery.

DISCUSSION:
SMTP users and administrators make regular use of these commands for diagnosing mail delivery problems. With the increasing use of multi-level mailing list expansion (sometimes more than two levels), EXPN has been increasingly important for diagnosing inadvertent mail loops. On the other hand, some feel that EXPN represents a significant privacy, and perhaps even a security, exposure.

5.2.4 SEND, SOML, and SAML Commands: RFC-821 Section 3.4

An SMTP MAY implement the commands to send a message to a user's terminal: SEND, SOML, and SAML.

DISCUSSION:
It has been suggested that the use of mail relaying through an MX record is inconsistent with the intent of SEND to deliver a message immediately and directly to a user's terminal. However, an SMTP receiver that is unable to write directly to the user terminal can return a "251 User Not Local" reply to the RCPT following a SEND, to inform the originator of possibly deferred delivery.

5.2.5 HELO Command: RFC-821 Section 3.5

The sender-SMTP MUST ensure that the <domain> parameter in a HELO command is a valid principal host domain name for the client host. As a result, the receiver-SMTP will not have to perform MX resolution on this name in order to validate the HELO parameter.

The HELO receiver MAY verify that the HELO parameter really

corresponds to the IP address of the sender. However, the receiver MUST NOT refuse to accept a message, even if the sender's HELO command fails verification.

DISCUSSION:

Verifying the HELO parameter requires a domain name lookup and may therefore take considerable time. An alternative tool for tracking bogus mail sources is suggested below (see "DATA Command").

Note also that the HELO argument is still required to have valid <domain> syntax, since it will appear in a Received: line; otherwise, a 501 error is to be sent.

IMPLEMENTATION:

When HELO parameter validation fails, a suggested procedure is to insert a note about the unknown authenticity of the sender into the message header (e.g., in the "Received:" line).

5.2.6 Mail Relay: RFC-821 Section 3.6

We distinguish three types of mail (store-and-) forwarding:

(1) A simple forwarder or "mail exchanger" forwards a message using private knowledge about the recipient; see section 3.2 of RFC-821.

(2) An SMTP mail "relay" forwards a message within an SMTP mail environment as the result of an explicit source route (as defined in section 3.6 of RFC-821). The SMTP relay function uses the "@...:" form of source route from RFC-822 (see Section 5.2.19 below).

(3) A mail "gateway" passes a message between different environments. The rules for mail gateways are discussed below in Section 5.3.7.

An Internet host that is forwarding a message but is not a gateway to a different mail environment (i.e., it falls under (1) or (2)) SHOULD NOT alter any existing header fields, although the host will add an appropriate Received: line as required in Section 5.2.8.

A Sender-SMTP SHOULD NOT send a RCPT TO: command containing an explicit source route using the "@...:" address form. Thus, the relay function defined in section 3.6 of RFC-821 should not be used.

DISCUSSION:
The intent is to discourage all source routing and to abolish explicit source routing for mail delivery within the Internet environment. Source-routing is unnecessary; the simple target address "user@domain" should always suffice. This is the result of an explicit architectural decision to use universal naming rather than source routing for mail. Thus, SMTP provides end-to-end connectivity, and the DNS provides globally-unique, location-independent names. MX records handle the major case where source routing might otherwise be needed.

A receiver-SMTP MUST accept the explicit source route syntax in the envelope, but it MAY implement the relay function as defined in section 3.6 of RFC-821. If it does not implement the relay function, it SHOULD attempt to deliver the message directly to the host to the right of the right-most "@" sign.

DISCUSSION:
For example, suppose a host that does not implement the relay function receives a message with the SMTP command: "RCPT TO:<@ALPHA,@BETA:joe@GAMMA>", where ALPHA, BETA, and GAMMA represent domain names. Rather than immediately refusing the message with a 550 error reply as suggested on page 20 of RFC-821, the host should try to forward the message to GAMMA directly, using: "RCPT TO:<joe@GAMMA>". Since this host does not support relaying, it is not required to update the reverse path.

Some have suggested that source routing may be needed occasionally for manually routing mail around failures; however, the reality and importance of this need is controversial. The use of explicit SMTP mail relaying for this purpose is discouraged, and in fact it may not be successful, as many host systems do not support it. Some have used the "%-hack" (see Section 5.2.16) for this purpose.

5.2.7 RCPT Command: RFC-821 Section 4.1.1

A host that supports a receiver-SMTP MUST support the reserved mailbox "Postmaster".

The receiver-SMTP MAY verify RCPT parameters as they arrive; however, RCPT responses MUST NOT be delayed beyond a reasonable time (see Section 5.3.2).

Therefore, a "250 OK" response to a RCPT does not necessarily

imply that the delivery address(es) are valid. Errors found after message acceptance will be reported by mailing a notification message to an appropriate address (see Section 5.3.3).

DISCUSSION:

The set of conditions under which a RCPT parameter can be validated immediately is an engineering design choice. Reporting destination mailbox errors to the Sender-SMTP before mail is transferred is generally desirable to save time and network bandwidth, but this advantage is lost if RCPT verification is lengthy.

For example, the receiver can verify immediately any simple local reference, such as a single locally-registered mailbox. On the other hand, the "reasonable time" limitation generally implies deferring verification of a mailing list until after the message has been transferred and accepted, since verifying a large mailing list can take a very long time. An implementation might or might not choose to defer validation of addresses that are non-local and therefore require a DNS lookup. If a DNS lookup is performed but a soft domain system error (e.g., timeout) occurs, validity must be assumed.

5.2.8 DATA Command: RFC-821 Section 4.1.1

Every receiver-SMTP (not just one that "accepts a message for relaying or for final delivery" [SMTP:1]) MUST insert a "Received:" line at the beginning of a message. In this line, called a "time stamp line" in RFC-821:

* The FROM field SHOULD contain both (1) the name of the source host as presented in the HELO command and (2) a domain literal containing the IP address of the source, determined from the TCP connection.

* The ID field MAY contain an "@" as suggested in RFC-822, but this is not required.

* The FOR field MAY contain a list of <path> entries when multiple RCPT commands have been given.

An Internet mail program MUST NOT change a Received: line that was previously added to the message header.

DISCUSSION:
Including both the source host and the IP source address in the Received: line may provide enough information for tracking illicit mail sources and eliminate a need to explicitly verify the HELO parameter.

Received: lines are primarily intended for humans tracing mail routes, primarily of diagnosis of faults. See also the discussion under 5.3.7.

When the receiver-SMTP makes "final delivery" of a message, then it MUST pass the MAIL FROM: address from the SMTP envelope with the message, for use if an error notification message must be sent later (see Section 5.3.3). There is an analogous requirement when gatewaying from the Internet into a different mail environment; see Section 5.3.7.

DISCUSSION:
Note that the final reply to the DATA command depends only upon the successful transfer and storage of the message. Any problem with the destination address(es) must either (1) have been reported in an SMTP error reply to the RCPT command(s), or (2) be reported in a later error message mailed to the originator.

IMPLEMENTATION:
The MAIL FROM: information may be passed as a parameter or in a Return-Path: line inserted at the beginning of the message.

5.2.9 Command Syntax: RFC-821 Section 4.1.2

The syntax shown in RFC-821 for the MAIL FROM: command omits the case of an empty path: "MAIL FROM: <>" (see RFC-821 Page 15). An empty reverse path MUST be supported.

5.2.10 SMTP Replies: RFC-821 Section 4.2

A receiver-SMTP SHOULD send only the reply codes listed in section 4.2.2 of RFC-821 or in this document. A receiver-SMTP SHOULD use the text shown in examples in RFC-821 whenever appropriate.

A sender-SMTP MUST determine its actions only by the reply code, not by the text (except for 251 and 551 replies); any text, including no text at all, must be acceptable. The space (blank) following the reply code is considered part of the text. Whenever possible, a sender-SMTP SHOULD test only the

first digit of the reply code, as specified in Appendix E of RFC-821.

DISCUSSION:
Interoperability problems have arisen with SMTP systems using reply codes that are not listed explicitly in RFC-821 Section 4.3 but are legal according to the theory of reply codes explained in Appendix E.

5.2.11 Transparency: RFC-821 Section 4.5.2

Implementors MUST be sure that their mail systems always add and delete periods to ensure message transparency.

5.2.12 WKS Use in MX Processing: RFC-974, p. 5

RFC-974 [SMTP:3] recommended that the domain system be queried for WKS ("Well-Known Service") records, to verify that each proposed mail target does support SMTP. Later experience has shown that WKS is not widely supported, so the WKS step in MX processing SHOULD NOT be used.

The following are notes on RFC-822, organized by section of that document.

5.2.13 RFC-822 Message Specification: RFC-822 Section 4

The syntax shown for the Return-path line omits the possibility of a null return path, which is used to prevent looping of error notifications (see Section 5.3.3). The complete syntax is:

```
return = "Return-path"  ":" route-addr
       / "Return-path"  ":" "<" ">"
```

The set of optional header fields is hereby expanded to include the Content-Type field defined in RFC-1049 [SMTP:7]. This field "allows mail reading systems to automatically identify the type of a structured message body and to process it for display accordingly". [SMTP:7] A User Agent MAY support this field.

5.2.14 RFC-822 Date and Time Specification: RFC-822 Section 5

The syntax for the date is hereby changed to:

```
date = 1*2DIGIT month 2*4DIGIT
```

All mail software SHOULD use 4-digit years in dates, to ease the transition to the next century.

There is a strong trend towards the use of numeric timezone indicators, and implementations SHOULD use numeric timezones instead of timezone names. However, all implementations MUST accept either notation. If timezone names are used, they MUST be exactly as defined in RFC-822.

The military time zones are specified incorrectly in RFC-822: they count the wrong way from UT (the signs are reversed). As a result, military time zones in RFC-822 headers carry no information.

Finally, note that there is a typo in the definition of "zone" in the syntax summary of appendix D; the correct definition occurs in Section 3 of RFC-822.

5.2.15 RFC-822 Syntax Change: RFC-822 Section 6.1

The syntactic definition of "mailbox" in RFC-822 is hereby changed to:

```
mailbox =  addr-spec            ; simple address
        / [phrase] route-addr   ; name & addr-spec
```

That is, the phrase preceding a route address is now OPTIONAL. This change makes the following header field legal, for example:

```
From: <craig@nnsc.nsf.net>
```

5.2.16 RFC-822 Local-part: RFC-822 Section 6.2

The basic mailbox address specification has the form: "local-part@domain". Here "local-part", sometimes called the "left-hand side" of the address, is domain-dependent.

A host that is forwarding the message but is not the destination host implied by the right-hand side "domain" MUST NOT interpret or modify the "local-part" of the address.

When mail is to be gatewayed from the Internet mail environment into a foreign mail environment (see Section 5.3.7), routing information for that foreign environment MAY be embedded within the "local-part" of the address. The gateway will then interpret this local part appropriately for the foreign mail environment.

DISCUSSION:

Although source routes are discouraged within the Internet (see Section 5.2.6), there are non-Internet mail environments whose delivery mechanisms do depend upon source routes. Source routes for extra-Internet environments can generally be buried in the "local-part" of the address (see Section 5.2.16) while mail traverses the Internet. When the mail reaches the appropriate Internet mail gateway, the gateway will interpret the local-part and build the necessary address or route for the target mail environment.

For example, an Internet host might send mail to: "a!b!c!user@gateway-domain". The complex local part "a!b!c!user" would be uninterpreted within the Internet domain, but could be parsed and understood by the specified mail gateway.

An embedded source route is sometimes encoded in the "local-part" using "%" as a right-binding routing operator. For example, in:

user%domain%relay3%relay2@relay1

the "%" convention implies that the mail is to be routed from "relay1" through "relay2", "relay3", and finally to "user" at "domain". This is commonly known as the "%-hack". It is suggested that "%" have lower precedence than any other routing operator (e.g., "!") hidden in the local-part; for example, "a!b%c" would be interpreted as "(a!b)%c".

Only the target host (in this case, "relay1") is permitted to analyze the local-part "user%domain%relay3%relay2".

5.2.17 Domain Literals: RFC-822 Section 6.2.3

A mailer MUST be able to accept and parse an Internet domain literal whose content ("dtext"; see RFC-822) is a dotted-decimal host address. This satisfies the requirement of Section 2.1 for the case of mail.

An SMTP MUST accept and recognize a domain literal for any of its own IP addresses.

5.2.18 Common Address Formatting Errors: RFC-822 Section 6.1

Errors in formatting or parsing 822 addresses are unfortunately common. This section mentions only the most common errors. A User Agent MUST accept all valid RFC-822 address formats, and MUST NOT generate illegal address syntax.

- o A common error is to leave out the semicolon after a group identifier.

- o Some systems fail to fully-qualify domain names in messages they generate. The right-hand side of an "@" sign in a header address field MUST be a fully-qualified domain name.

  For example, some systems fail to fully-qualify the From: address; this prevents a "reply" command in the user interface from automatically constructing a return address.

  DISCUSSION:
  Although RFC-822 allows the local use of abbreviated domain names within a domain, the application of RFC-822 in Internet mail does not allow this. The intent is that an Internet host must not send an SMTP message header containing an abbreviated domain name in an address field. This allows the address fields of the header to be passed without alteration across the Internet, as required in Section 5.2.6.

- o Some systems mis-parse multiple-hop explicit source routes such as:

  @relay1,@relay2,@relay3:user@domain.

- o Some systems over-qualify domain names by adding a trailing dot to some or all domain names in addresses or message-ids. This violates RFC-822 syntax.

5.2.19 Explicit Source Routes: RFC-822 Section 6.2.7

Internet host software SHOULD NOT create an RFC-822 header containing an address with an explicit source route, but MUST accept such headers for compatibility with earlier systems.

DISCUSSION:

In an understatement, RFC-822 says "The use of explicit source routing is discouraged". Many hosts implemented RFC-822 source routes incorrectly, so the syntax cannot be used unambiguously in practice. Many users feel the syntax is ugly. Explicit source routes are not needed in the mail envelope for delivery; see Section 5.2.6. For all these reasons, explicit source routes using the RFC-822 notations are not to be used in Internet mail headers.

As stated in Section 5.2.16, it is necessary to allow an explicit source route to be buried in the local-part of an address, e.g., using the "%-hack", in order to allow mail to be gatewayed into another environment in which explicit source routing is necessary. The vigilant will observe that there is no way for a User Agent to detect and prevent the use of such implicit source routing when the destination is within the Internet. We can only discourage source routing of any kind within the Internet, as unnecessary and undesirable.

5.3 SPECIFIC ISSUES

5.3.1 SMTP Queueing Strategies

The common structure of a host SMTP implementation includes user mailboxes, one or more areas for queueing messages in transit, and one or more daemon processes for sending and receiving mail. The exact structure will vary depending on the needs of the users on the host and the number and size of mailing lists supported by the host. We describe several optimizations that have proved helpful, particularly for mailers supporting high traffic levels.

Any queueing strategy MUST include:

- Timeouts on all activities. See Section 5.3.2.
- Never sending error messages in response to error messages.

5.3.1.1 Sending Strategy

The general model of a sender-SMTP is one or more processes that periodically attempt to transmit outgoing mail. In a typical system, the program that composes a message has some method for requesting immediate attention for a new piece of outgoing mail, while mail that cannot be transmitted

immediately MUST be queued and periodically retried by the sender. A mail queue entry will include not only the message itself but also the envelope information.

The sender MUST delay retrying a particular destination after one attempt has failed. In general, the retry interval SHOULD be at least 30 minutes; however, more sophisticated and variable strategies will be beneficial when the sender-SMTP can determine the reason for non-delivery.

Retries continue until the message is transmitted or the sender gives up; the give-up time generally needs to be at least 4-5 days. The parameters to the retry algorithm MUST be configurable.

A sender SHOULD keep a list of hosts it cannot reach and corresponding timeouts, rather than just retrying queued mail items.

DISCUSSION:

Experience suggests that failures are typically transient (the target system has crashed), favoring a policy of two connection attempts in the first hour the message is in the queue, and then backing off to once every two or three hours.

The sender-SMTP can shorten the queueing delay by cooperation with the receiver-SMTP. In particular, if mail is received from a particular address, it is good evidence that any mail queued for that host can now be sent.

The strategy may be further modified as a result of multiple addresses per host (see Section 5.3.4), to optimize delivery time vs. resource usage.

A sender-SMTP may have a large queue of messages for each unavailable destination host, and if it retried all these messages in every retry cycle, there would be excessive Internet overhead and the daemon would be blocked for a long period. Note that an SMTP can generally determine that a delivery attempt has failed only after a timeout of a minute or more; a one minute timeout per connection will result in a very large delay if it is repeated for dozens or even hundreds of queued messages.

When the same message is to be delivered to several users on the same host, only one copy of the message SHOULD be transmitted. That is, the sender-SMTP should use the command sequence: RCPT, RCPT,... RCPT, DATA instead of the sequence: RCPT, DATA, RCPT, DATA,... RCPT, DATA. Implementation of this efficiency feature is strongly urged.

Similarly, the sender-SMTP MAY support multiple concurrent outgoing mail transactions to achieve timely delivery. However, some limit SHOULD be imposed to protect the host from devoting all its resources to mail.

The use of the different addresses of a multihomed host is discussed below.

5.3.1.2 Receiving strategy

The receiver-SMTP SHOULD attempt to keep a pending listen on the SMTP port at all times. This will require the support of multiple incoming TCP connections for SMTP. Some limit MAY be imposed.

IMPLEMENTATION:
When the receiver-SMTP receives mail from a particular host address, it could notify the sender-SMTP to retry any mail pending for that host address.

5.3.2 Timeouts in SMTP

There are two approaches to timeouts in the sender-SMTP: (a) limit the time for each SMTP command separately, or (b) limit the time for the entire SMTP dialogue for a single mail message. A sender-SMTP SHOULD use option (a), per-command timeouts. Timeouts SHOULD be easily reconfigurable, preferably without recompiling the SMTP code.

DISCUSSION:
Timeouts are an essential feature of an SMTP implementation. If the timeouts are too long (or worse, there are no timeouts), Internet communication failures or software bugs in receiver-SMTP programs can tie up SMTP processes indefinitely. If the timeouts are too short, resources will be wasted with attempts that time out part way through message delivery.

If option (b) is used, the timeout has to be very large, e.g., an hour, to allow time to expand very large mailing lists. The timeout may also need to increase linearly

with the size of the message, to account for the time to transmit a very large message. A large fixed timeout leads to two problems: a failure can still tie up the sender for a very long time, and very large messages may still spuriously time out (which is a wasteful failure!).

Using the recommended option (a), a timer is set for each SMTP command and for each buffer of the data transfer. The latter means that the overall timeout is inherently proportional to the size of the message.

Based on extensive experience with busy mail-relay hosts, the minimum per-command timeout values SHOULD be as follows:

- Initial 220 Message: 5 minutes

  A Sender-SMTP process needs to distinguish between a failed TCP connection and a delay in receiving the initial 220 greeting message. Many receiver-SMTPs will accept a TCP connection but delay delivery of the 220 message until their system load will permit more mail to be processed.

- MAIL Command: 5 minutes

- RCPT Command: 5 minutes

  A longer timeout would be required if processing of mailing lists and aliases were not deferred until after the message was accepted.

- DATA Initiation: 2 minutes

  This is while awaiting the "354 Start Input" reply to a DATA command.

- Data Block: 3 minutes

  This is while awaiting the completion of each TCP SEND call transmitting a chunk of data.

- DATA Termination: 10 minutes.

  This is while awaiting the "250 OK" reply. When the receiver gets the final period terminating the message data, it typically performs processing to deliver the message to a user mailbox. A spurious timeout at this point would be very wasteful, since the message has been

successfully sent.

A receiver-SMTP SHOULD have a timeout of at least 5 minutes while it is awaiting the next command from the sender.

5.3.3 Reliable Mail Receipt

When the receiver-SMTP accepts a piece of mail (by sending a "250 OK" message in response to DATA), it is accepting responsibility for delivering or relaying the message. It must take this responsibility seriously, i.e., it MUST NOT lose the message for frivolous reasons, e.g., because the host later crashes or because of a predictable resource shortage.

If there is a delivery failure after acceptance of a message, the receiver-SMTP MUST formulate and mail a notification message. This notification MUST be sent using a null ("<>") reverse path in the envelope; see Section 3.6 of RFC-821. The recipient of this notification SHOULD be the address from the envelope return path (or the Return-Path: line). However, if this address is null ("<>"), the receiver-SMTP MUST NOT send a notification. If the address is an explicit source route, it SHOULD be stripped down to its final hop.

DISCUSSION:

For example, suppose that an error notification must be sent for a message that arrived with: "MAIL FROM:<@a,@b:user@d>". The notification message should be sent to: "RCPT TO:<user@d>".

Some delivery failures after the message is accepted by SMTP will be unavoidable. For example, it may be impossible for the receiver-SMTP to validate all the delivery addresses in RCPT command(s) due to a "soft" domain system error or because the target is a mailing list (see earlier discussion of RCPT).

To avoid receiving duplicate messages as the result of timeouts, a receiver-SMTP MUST seek to minimize the time required to respond to the final "." that ends a message transfer. See RFC-1047 [SMTP:4] for a discussion of this problem.

5.3.4 Reliable Mail Transmission

To transmit a message, a sender-SMTP determines the IP address of the target host from the destination address in the envelope. Specifically, it maps the string to the right of the

"@" sign into an IP address. This mapping or the transfer itself may fail with a soft error, in which case the sender-SMTP will requeue the outgoing mail for a later retry, as required in Section 5.3.1.1.

When it succeeds, the mapping can result in a list of alternative delivery addresses rather than a single address, because of (a) multiple MX records, (b) multihoming, or both. To provide reliable mail transmission, the sender-SMTP MUST be able to try (and retry) each of the addresses in this list in order, until a delivery attempt succeeds. However, there MAY also be a configurable limit on the number of alternate addresses that can be tried. In any case, a host SHOULD try at least two addresses.

The following information is to be used to rank the host addresses:

(1) Multiple MX Records -- these contain a preference indication that should be used in sorting. If there are multiple destinations with the same preference and there is no clear reason to favor one (e.g., by address preference), then the sender-SMTP SHOULD pick one at random to spread the load across multiple mail exchanges for a specific organization; note that this is a refinement of the procedure in [DNS:3].

(2) Multihomed host -- The destination host (perhaps taken from the preferred MX record) may be multihomed, in which case the domain name resolver will return a list of alternative IP addresses. It is the responsibility of the domain name resolver interface (see Section 6.1.3.4 below) to have ordered this list by decreasing preference, and SMTP MUST try them in the order presented.

DISCUSSION:

Although the capability to try multiple alternative addresses is required, there may be circumstances where specific installations want to limit or disable the use of alternative addresses. The question of whether a sender should attempt retries using the different addresses of a multihomed host has been controversial. The main argument for using the multiple addresses is that it maximizes the probability of timely delivery, and indeed sometimes the probability of any delivery; the counter argument is that it may result in unnecessary resource use.

Note that resource use is also strongly determined by the

sending strategy discussed in Section 5.3.1.

5.3.5 Domain Name Support

SMTP implementations MUST use the mechanism defined in Section 6.1 for mapping between domain names and IP addresses. This means that every Internet SMTP MUST include support for the Internet DNS.

In particular, a sender-SMTP MUST support the MX record scheme [SMTP:3]. See also Section 7.4 of [DNS:2] for information on domain name support for SMTP.

5.3.6 Mailing Lists and Aliases

An SMTP-capable host SHOULD support both the alias and the list form of address expansion for multiple delivery. When a message is delivered or forwarded to each address of an expanded list form, the return address in the envelope ("MAIL FROM:") MUST be changed to be the address of a person who administers the list, but the message header MUST be left unchanged; in particular, the "From" field of the message is unaffected.

DISCUSSION:

An important mail facility is a mechanism for multi-destination delivery of a single message, by transforming or "expanding" a pseudo-mailbox address into a list of destination mailbox addresses. When a message is sent to such a pseudo-mailbox (sometimes called an "exploder"), copies are forwarded or redistributed to each mailbox in the expanded list. We classify such a pseudo-mailbox as an "alias" or a "list", depending upon the expansion rules:

(a) Alias

To expand an alias, the recipient mailer simply replaces the pseudo-mailbox address in the envelope with each of the expanded addresses in turn; the rest of the envelope and the message body are left unchanged. The message is then delivered or forwarded to each expanded address.

(b) List

A mailing list may be said to operate by "redistribution" rather than by "forwarding". To

expand a list, the recipient mailer replaces the pseudo-mailbox address in the envelope with each of the expanded addresses in turn. The return address in the envelope is changed so that all error messages generated by the final deliveries will be returned to a list administrator, not to the message originator, who generally has no control over the contents of the list and will typically find error messages annoying.

5.3.7 Mail Gatewaying

Gatewaying mail between different mail environments, i.e., different mail formats and protocols, is complex and does not easily yield to standardization. See for example [SMTP:5a], [SMTP:5b]. However, some general requirements may be given for a gateway between the Internet and another mail environment.

(A) Header fields MAY be rewritten when necessary as messages are gatewayed across mail environment boundaries.

DISCUSSION:
This may involve interpreting the local-part of the destination address, as suggested in Section 5.2.16.

The other mail systems gatewayed to the Internet generally use a subset of RFC-822 headers, but some of them do not have an equivalent to the SMTP envelope. Therefore, when a message leaves the Internet environment, it may be necessary to fold the SMTP envelope information into the message header. A possible solution would be to create new header fields to carry the envelope information (e.g., "X-SMTP-MAIL:" and "X-SMTP-RCPT:"); however, this would require changes in mail programs in the foreign environment.

(B) When forwarding a message into or out of the Internet environment, a gateway MUST prepend a Received: line, but it MUST NOT alter in any way a Received: line that is already in the header.

DISCUSSION:
This requirement is a subset of the general "Received:" line requirement of Section 5.2.8; it is restated here for emphasis.

Received: fields of messages originating from other

environments may not conform exactly to RFC822. However, the most important use of Received: lines is for debugging mail faults, and this debugging can be severely hampered by well-meaning gateways that try to "fix" a Received: line.

The gateway is strongly encouraged to indicate the environment and protocol in the "via" clauses of Received field(s) that it supplies.

(C) From the Internet side, the gateway SHOULD accept all valid address formats in SMTP commands and in RFC-822 headers, and all valid RFC-822 messages. Although a gateway must accept an RFC-822 explicit source route ("@...:" format) in either the RFC-822 header or in the envelope, it MAY or may not act on the source route; see Sections 5.2.6 and 5.2.19.

DISCUSSION:
It is often tempting to restrict the range of addresses accepted at the mail gateway to simplify the translation into addresses for the remote environment. This practice is based on the assumption that mail users have control over the addresses their mailers send to the mail gateway. In practice, however, users have little control over the addresses that are finally sent; their mailers are free to change addresses into any legal RFC-822 format.

(D) The gateway MUST ensure that all header fields of a message that it forwards into the Internet meet the requirements for Internet mail. In particular, all addresses in "From:", "To:", "Cc:", etc., fields must be transformed (if necessary) to satisfy RFC-822 syntax, and they must be effective and useful for sending replies.

(E) The translation algorithm used to convert mail from the Internet protocols to another environment's protocol SHOULD try to ensure that error messages from the foreign mail environment are delivered to the return path from the SMTP envelope, not to the sender listed in the "From:" field of the RFC-822 message.

DISCUSSION:
Internet mail lists usually place the address of the mail list maintainer in the envelope but leave the

original message header intact (with the "From:" field containing the original sender). This yields the behavior the average recipient expects: a reply to the header gets sent to the original sender, not to a mail list maintainer; however, errors get sent to the maintainer (who can fix the problem) and not the sender (who probably cannot).

(F) Similarly, when forwarding a message from another environment into the Internet, the gateway SHOULD set the envelope return path in accordance with an error message return address, if any, supplied by the foreign environment.

5.3.8 Maximum Message Size

Mailer software MUST be able to send and receive messages of at least 64K bytes in length (including header), and a much larger maximum size is highly desirable.

DISCUSSION:

Although SMTP does not define the maximum size of a message, many systems impose implementation limits.

The current de facto minimum limit in the Internet is 64K bytes. However, electronic mail is used for a variety of purposes that create much larger messages. For example, mail is often used instead of FTP for transmitting ASCII files, and in particular to transmit entire documents. As a result, messages can be 1 megabyte or even larger. We note that the present document together with its lower-layer companion contains 0.5 megabytes.

5.4 SMTP REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
RECEIVER-SMTP:							
Implement VRFY	5.2.3	x					
Implement EXPN	5.2.3		x				
EXPN, VRFY configurable	5.2.3			x			
Implement SEND, SOML, SAML	5.2.4			x			
Verify HELO parameter	5.2.5			x			
Refuse message with bad HELO	5.2.5					x	
Accept explicit src-route syntax in env.	5.2.6	x					
Support "postmaster"	5.2.7	x					
Process RCPT when received (except lists)	5.2.7			x			
Long delay of RCPT responses	5.2.7					x	
Add Received: line	5.2.8	x					
Received: line include domain literal	5.2.8		x				
Change previous Received: line	5.2.8					x	
Pass Return-Path info (final deliv/gwy)	5.2.8	x					
Support empty reverse path	5.2.9	x					
Send only official reply codes	5.2.10		x				
Send text from RFC-821 when appropriate	5.2.10		x				
Delete "." for transparency	5.2.11	x					
Accept and recognize self domain literal(s)	5.2.17	x					
Error message about error message	5.3.1					x	
Keep pending listen on SMTP port	5.3.1.2		x				
Provide limit on recv concurrency	5.3.1.2			x			
Wait at least 5 mins for next sender cmd	5.3.2		x				
Avoidable delivery failure after "250 OK"	5.3.3					x	
Send error notification msg after accept	5.3.3	x					
Send using null return path	5.3.3	x					
Send to envelope return path	5.3.3		x				
Send to null address	5.3.3					x	
Strip off explicit src route	5.3.3		x				
Minimize acceptance delay (RFC-1047)	5.3.3	x					

SENDER-SMTP:							
Canonicalized domain names in MAIL, RCPT	5.2.2	x					
Implement SEND, SOML, SAML	5.2.4			x			
Send valid principal host name in HELO	5.2.5	x					
Send explicit source route in RCPT TO:	5.2.6				x		
Use only reply code to determine action	5.2.10	x					
Use only high digit of reply code when poss.	5.2.10		x				
Add "." for transparency	5.2.11	x					
Retry messages after soft failure	5.3.1.1	x					
Delay before retry	5.3.1.1	x					
Configurable retry parameters	5.3.1.1	x					
Retry once per each queued dest host	5.3.1.1		x				
Multiple RCPT's for same DATA	5.3.1.1		x				
Support multiple concurrent transactions	5.3.1.1			x			
Provide limit on concurrency	5.3.1.1		x				
Timeouts on all activities	5.3.1	x					
Per-command timeouts	5.3.2		x				
Timeouts easily reconfigurable	5.3.2		x				
Recommended times	5.3.2		x				
Try alternate addr's in order	5.3.4	x					
Configurable limit on alternate tries	5.3.4			x			
Try at least two alternates	5.3.4		x				
Load-split across equal MX alternates	5.3.4		x				
Use the Domain Name System	5.3.5	x					
Support MX records	5.3.5	x					
Use WKS records in MX processing	5.2.12				x		
MAIL FORWARDING:							
Alter existing header field(s)	5.2.6				x		
Implement relay function: 821/section 3.6	5.2.6			x			
If not, deliver to RHS domain	5.2.6		x				
Interpret 'local-part' of addr	5.2.16					x	
MAILING LISTS AND ALIASES							
Support both	5.3.6		x				
Report mail list error to local admin.	5.3.6	x					
MAIL GATEWAYS:							
Embed foreign mail route in local-part	5.2.16			x			
Rewrite header fields when necessary	5.3.7			x			
Prepend Received: line	5.3.7	x					
Change existing Received: line	5.3.7					x	
Accept full RFC-822 on Internet side	5.3.7		x				
Act on RFC-822 explicit source route	5.3.7			x			

Send only valid RFC-822 on Internet side	5.3.7	x				
Deliver error msgs to envelope addr	5.3.7		x			
Set env return path from err return addr	5.3.7		x			
USER AGENT -- RFC-822						
Allow user to enter <route> address	5.2.6				x	
Support RFC-1049 Content Type field	5.2.13			x		
Use 4-digit years	5.2.14		x			
Generate numeric timezones	5.2.14		x			
Accept all timezones	5.2.14	x				
Use non-num timezones from RFC-822	5.2.14	x				
Omit phrase before route-addr	5.2.15			x		
Accept and parse dot.dec. domain literals	5.2.17	x				
Accept all RFC-822 address formats	5.2.18	x				
Generate invalid RFC-822 address format	5.2.18					x
Fully-qualified domain names in header	5.2.18	x				
Create explicit src route in header	5.2.19				x	
Accept explicit src route in header	5.2.19	x				
Send/recv at least 64KB messages	5.3.8	x				

6. SUPPORT SERVICES

6.1 DOMAIN NAME TRANSLATION

6.1.1 INTRODUCTION

Every host MUST implement a resolver for the Domain Name System (DNS), and it MUST implement a mechanism using this DNS resolver to convert host names to IP addresses and vice-versa [DNS:1, DNS:2].

In addition to the DNS, a host MAY also implement a host name translation mechanism that searches a local Internet host table. See Section 6.1.3.8 for more information on this option.

DISCUSSION:

Internet host name translation was originally performed by searching local copies of a table of all hosts. This table became too large to update and distribute in a timely manner and too large to fit into many hosts, so the DNS was invented.

The DNS creates a distributed database used primarily for the translation between host names and host addresses. Implementation of DNS software is required. The DNS consists of two logically distinct parts: name servers and resolvers (although implementations often combine these two logical parts in the interest of efficiency) [DNS:2].

Domain name servers store authoritative data about certain sections of the database and answer queries about the data. Domain resolvers query domain name servers for data on behalf of user processes. Every host therefore needs a DNS resolver; some host machines will also need to run domain name servers. Since no name server has complete information, in general it is necessary to obtain information from more than one name server to resolve a query.

6.1.2 PROTOCOL WALK-THROUGH

An implementor must study references [DNS:1] and [DNS:2] carefully. They provide a thorough description of the theory, protocol, and implementation of the domain name system, and reflect several years of experience.

6.1.2.1 Resource Records with Zero TTL: RFC-1035 Section 3.2.1

All DNS name servers and resolvers MUST properly handle RRs with a zero TTL: return the RR to the client but do not cache it.

DISCUSSION:
Zero TTL values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached; they are useful for extremely volatile data.

6.1.2.2 QCLASS Values: RFC-1035 Section 3.2.5

A query with "QCLASS=*" SHOULD NOT be used unless the requestor is seeking data from more than one class. In particular, if the requestor is only interested in Internet data types, QCLASS=IN MUST be used.

6.1.2.3 Unused Fields: RFC-1035 Section 4.1.1

Unused fields in a query or response message MUST be zero.

6.1.2.4 Compression: RFC-1035 Section 4.1.4

Name servers MUST use compression in responses.

DISCUSSION:
Compression is essential to avoid overflowing UDP datagrams; see Section 6.1.3.2.

6.1.2.5 Misusing Configuration Info: RFC-1035 Section 6.1.2

Recursive name servers and full-service resolvers generally have some configuration information containing hints about the location of root or local name servers. An implementation MUST NOT include any of these hints in a response.

DISCUSSION:
Many implementors have found it convenient to store these hints as if they were cached data, but some neglected to ensure that this "cached data" was not included in responses. This has caused serious problems in the Internet when the hints were obsolete or incorrect.

6.1.3 SPECIFIC ISSUES

6.1.3.1 Resolver Implementation

A name resolver SHOULD be able to multiplex concurrent requests if the host supports concurrent processes.

In implementing a DNS resolver, one of two different models MAY optionally be chosen: a full-service resolver, or a stub resolver.

(A) Full-Service Resolver

A full-service resolver is a complete implementation of the resolver service, and is capable of dealing with communication failures, failure of individual name servers, location of the proper name server for a given name, etc. It must satisfy the following requirements:

- The resolver MUST implement a local caching function to avoid repeated remote access for identical requests, and MUST time out information in the cache.

- The resolver SHOULD be configurable with start-up information pointing to multiple root name servers and multiple name servers for the local domain. This insures that the resolver will be able to access the whole name space in normal cases, and will be able to access local domain information should the local network become disconnected from the rest of the Internet.

(B) Stub Resolver

A "stub resolver" relies on the services of a recursive name server on the connected network or a "nearby" network. This scheme allows the host to pass on the burden of the resolver function to a name server on another host. This model is often essential for less capable hosts, such as PCs, and is also recommended when the host is one of several workstations on a local network, because it allows all of the workstations to share the cache of the recursive name server and hence reduce the number of domain requests exported by the local network.

At a minimum, the stub resolver MUST be capable of directing its requests to redundant recursive name servers. Note that recursive name servers are allowed to restrict the sources of requests that they will honor, so the host administrator must verify that the service will be provided. Stub resolvers MAY implement caching if they choose, but if so, MUST timeout cached information.

6.1.3.2 Transport Protocols

DNS resolvers and recursive servers MUST support UDP, and SHOULD support TCP, for sending (non-zone-transfer) queries. Specifically, a DNS resolver or server that is sending a non-zone-transfer query MUST send a UDP query first. If the Answer section of the response is truncated and if the requester supports TCP, it SHOULD try the query again using TCP.

DNS servers MUST be able to service UDP queries and SHOULD be able to service TCP queries. A name server MAY limit the resources it devotes to TCP queries, but it SHOULD NOT refuse to service a TCP query just because it would have succeeded with UDP.

Truncated responses MUST NOT be saved (cached) and later used in such a way that the fact that they are truncated is lost.

DISCUSSION:

UDP is preferred over TCP for queries because UDP queries have much lower overhead, both in packet count and in connection state. The use of UDP is essential for heavily-loaded servers, especially the root servers. UDP also offers additional robustness, since a resolver can attempt several UDP queries to different servers for the cost of a single TCP query.

It is possible for a DNS response to be truncated, although this is a very rare occurrence in the present Internet DNS. Practically speaking, truncation cannot be predicted, since it is data-dependent. The dependencies include the number of RRs in the answer, the size of each RR, and the savings in space realized by the name compression algorithm. As a rule of thumb, truncation in NS and MX lists should not occur for answers containing 15 or fewer RRs.

Whether it is possible to use a truncated answer depends on the application. A mailer must not use a truncated MX response, since this could lead to mail loops.

Responsible practices can make UDP suffice in the vast majority of cases. Name servers must use compression in responses. Resolvers must differentiate truncation of the Additional section of a response (which only loses extra information) from truncation of the Answer section (which for MX records renders the response unusable by mailers). Database administrators should list only a reasonable number of primary names in lists of name servers, MX alternatives, etc.

However, it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that applies to UDP, and hence will require TCP. Thus, resolvers and name servers should implement TCP services as a backup to UDP today, with the knowledge that they will require the TCP service in the future.

By private agreement, name servers and resolvers MAY arrange to use TCP for all traffic between themselves. TCP MUST be used for zone transfers.

A DNS server MUST have sufficient internal concurrency that it can continue to process UDP queries while awaiting a response or performing a zone transfer on an open TCP connection [DNS:2].

A server MAY support a UDP query that is delivered using an IP broadcast or multicast address. However, the Recursion Desired bit MUST NOT be set in a query that is multicast, and MUST be ignored by name servers receiving queries via a broadcast or multicast address. A host that sends broadcast or multicast DNS queries SHOULD send them only as occasional probes, caching the IP address(es) it obtains from the response(s) so it can normally send unicast queries.

DISCUSSION:

Broadcast or (especially) IP multicast can provide a way to locate nearby name servers without knowing their IP addresses in advance. However, general broadcasting of recursive queries can result in excessive and unnecessary load on both network and servers.

6.1.3.3 Efficient Resource Usage

The following requirements on servers and resolvers are very important to the health of the Internet as a whole, particularly when DNS services are invoked repeatedly by higher level automatic servers, such as mailers.

(1) The resolver MUST implement retransmission controls to insure that it does not waste communication bandwidth, and MUST impose finite bounds on the resources consumed to respond to a single request. See [DNS:2] pages 43-44 for specific recommendations.

(2) After a query has been retransmitted several times without a response, an implementation MUST give up and return a soft error to the application.

(3) All DNS name servers and resolvers SHOULD cache temporary failures, with a timeout period of the order of minutes.

DISCUSSION:
This will prevent applications that immediately retry soft failures (in violation of Section 2.2 of this document) from generating excessive DNS traffic.

(4) All DNS name servers and resolvers SHOULD cache negative responses that indicate the specified name, or data of the specified type, does not exist, as described in [DNS:2].

(5) When a DNS server or resolver retries a UDP query, the retry interval SHOULD be constrained by an exponential backoff algorithm, and SHOULD also have upper and lower bounds.

IMPLEMENTATION:
A measured RTT and variance (if available) should be used to calculate an initial retransmission interval. If this information is not available, a default of no less than 5 seconds should be used. Implementations may limit the retransmission interval, but this limit must exceed twice the Internet maximum segment lifetime plus service delay at the name server.

(6) When a resolver or server receives a Source Quench for

a query it has issued, it SHOULD take steps to reduce the rate of querying that server in the near future. A server MAY ignore a Source Quench that it receives as the result of sending a response datagram.

IMPLEMENTATION:
One recommended action to reduce the rate is to send the next query attempt to an alternate server, if there is one available. Another is to backoff the retry interval for the same server.

6.1.3.4 Multihomed Hosts

When the host name-to-address function encounters a host with multiple addresses, it SHOULD rank or sort the addresses using knowledge of the immediately connected network number(s) and any other applicable performance or history information.

DISCUSSION:
The different addresses of a multihomed host generally imply different Internet paths, and some paths may be preferable to others in performance, reliability, or administrative restrictions. There is no general way for the domain system to determine the best path. A recommended approach is to base this decision on local configuration information set by the system administrator.

IMPLEMENTATION:
The following scheme has been used successfully:

(a) Incorporate into the host configuration data a Network-Preference List, that is simply a list of networks in preferred order. This list may be empty if there is no preference.

(b) When a host name is mapped into a list of IP addresses, these addresses should be sorted by network number, into the same order as the corresponding networks in the Network-Preference List. IP addresses whose networks do not appear in the Network-Preference List should be placed at the end of the list.

6.1.3.5 Extensibility

DNS software MUST support all well-known, class-independent formats [DNS:2], and SHOULD be written to minimize the trauma associated with the introduction of new well-known types and local experimentation with non-standard types.

DISCUSSION:

The data types and classes used by the DNS are extensible, and thus new types will be added and old types deleted or redefined. Introduction of new data types ought to be dependent only upon the rules for compression of domain names inside DNS messages, and the translation between printable (i.e., master file) and internal formats for Resource Records (RRs).

Compression relies on knowledge of the format of data inside a particular RR. Hence compression must only be used for the contents of well-known, class-independent RRs, and must never be used for class-specific RRs or RR types that are not well-known. The owner name of an RR is always eligible for compression.

A name server may acquire, via zone transfer, RRs that the server doesn't know how to convert to printable format. A resolver can receive similar information as the result of queries. For proper operation, this data must be preserved, and hence the implication is that DNS software cannot use textual formats for internal storage.

The DNS defines domain name syntax very generally -- a string of labels each containing up to 63 8-bit octets, separated by dots, and with a maximum total of 255 octets. Particular applications of the DNS are permitted to further constrain the syntax of the domain names they use, although the DNS deployment has led to some applications allowing more general names. In particular, Section 2.1 of this document liberalizes slightly the syntax of a legal Internet host name that was defined in RFC-952 [DNS:4].

6.1.3.6 Status of RR Types

Name servers MUST be able to load all RR types except MD and MF from configuration files. The MD and MF types are obsolete and MUST NOT be implemented; in particular, name servers MUST NOT load these types from configuration files.

DISCUSSION:

The RR types MB, MG, MR, NULL, MINFO and RP are considered experimental, and applications that use the DNS cannot expect these RR types to be supported by most domains. Furthermore these types are subject to redefinition.

The TXT and WKS RR types have not been widely used by Internet sites; as a result, an application cannot rely on the the existence of a TXT or WKS RR in most domains.

6.1.3.7 Robustness

DNS software may need to operate in environments where the root servers or other servers are unavailable due to network connectivity or other problems. In this situation, DNS name servers and resolvers MUST continue to provide service for the reachable part of the name space, while giving temporary failures for the rest.

DISCUSSION:

Although the DNS is meant to be used primarily in the connected Internet, it should be possible to use the system in networks which are unconnected to the Internet. Hence implementations must not depend on access to root servers before providing service for local names.

6.1.3.8 Local Host Table

DISCUSSION:

A host may use a local host table as a backup or supplement to the DNS. This raises the question of which takes precedence, the DNS or the host table; the most flexible approach would make this a configuration option.

Typically, the contents of such a supplementary host table will be determined locally by the site. However, a publically-available table of Internet hosts is maintained by the DDN Network Information Center (DDN NIC), with a format documented in [DNS:4]. This table can be retrieved from the DDN NIC using a protocol described in [DNS:5]. It must be noted that this table contains only a small fraction of all Internet hosts. Hosts using this protocol to retrieve the DDN NIC host table should use the VERSION command to check if the

table has changed before requesting the entire table with the ALL command. The VERSION identifier should be treated as an arbitrary string and tested only for equality; no numerical sequence may be assumed.

The DDN NIC host table includes administrative information that is not needed for host operation and is therefore not currently included in the DNS database; examples include network and gateway entries. However, much of this additional information will be added to the DNS in the future. Conversely, the DNS provides essential services (in particular, MX records) that are not available from the DDN NIC host table.

6.1.4 DNS USER INTERFACE

6.1.4.1 DNS Administration

This document is concerned with design and implementation issues in host software, not with administrative or operational issues. However, administrative issues are of particular importance in the DNS, since errors in particular segments of this large distributed database can cause poor or erroneous performance for many sites. These issues are discussed in [DNS:6] and [DNS:7].

6.1.4.2 DNS User Interface

Hosts MUST provide an interface to the DNS for all application programs running on the host. This interface will typically direct requests to a system process to perform the resolver function [DNS:1, 6.1:2].

At a minimum, the basic interface MUST support a request for all information of a specific type and class associated with a specific name, and it MUST return either all of the requested information, a hard error code, or a soft error indication. When there is no error, the basic interface returns the complete response information without modification, deletion, or ordering, so that the basic interface will not need to be changed to accommodate new data types.

DISCUSSION:
The soft error indication is an essential part of the interface, since it may not always be possible to access particular information from the DNS; see Section 6.1.3.3.

A host MAY provide other DNS interfaces tailored to particular functions, transforming the raw domain data into formats more suited to these functions. In particular, a host MUST provide a DNS interface to facilitate translation between host addresses and host names.

6.1.4.3 Interface Abbreviation Facilities

User interfaces MAY provide a method for users to enter abbreviations for commonly-used names. Although the definition of such methods is outside of the scope of the DNS specification, certain rules are necessary to insure that these methods allow access to the entire DNS name space and to prevent excessive use of Internet resources.

If an abbreviation method is provided, then:

(a) There MUST be some convention for denoting that a name is already complete, so that the abbreviation method(s) are suppressed. A trailing dot is the usual method.

(b) Abbreviation expansion MUST be done exactly once, and MUST be done in the context in which the name was entered.

DISCUSSION:
For example, if an abbreviation is used in a mail program for a destination, the abbreviation should be expanded into a full domain name and stored in the queued message with an indication that it is already complete. Otherwise, the abbreviation might be expanded with a mail system search list, not the user's, or a name could grow due to repeated canonicalizations attempts interacting with wildcards.

The two most common abbreviation methods are:

(1) Interface-level aliases

Interface-level aliases are conceptually implemented as a list of alias/domain name pairs. The list can be per-user or per-host, and separate lists can be associated with different functions, e.g. one list for host name-to-address translation, and a different list for mail domains. When the user enters a name, the interface attempts to match the name to the alias component of a list entry, and if a matching entry can

be found, the name is replaced by the domain name found in the pair.

Note that interface-level aliases and CNAMEs are completely separate mechanisms; interface-level aliases are a local matter while CNAMEs are an Internet-wide aliasing mechanism which is a required part of any DNS implementation.

(2) Search Lists

A search list is conceptually implemented as an ordered list of domain names. When the user enters a name, the domain names in the search list are used as suffixes to the user-supplied name, one by one, until a domain name with the desired associated data is found, or the search list is exhausted. Search lists often contain the name of the local host's parent domain or other ancestor domains. Search lists are often per-user or per-process.

It SHOULD be possible for an administrator to disable a DNS search-list facility. Administrative denial may be warranted in some cases, to prevent abuse of the DNS.

There is danger that a search-list mechanism will generate excessive queries to the root servers while testing whether user input is a complete domain name, lacking a final period to mark it as complete. A search-list mechanism MUST have one of, and SHOULD have both of, the following two provisions to prevent this:

(a) The local resolver/name server can implement caching of negative responses (see Section 6.1.3.3).

(b) The search list expander can require two or more interior dots in a generated domain name before it tries using the name in a query to non-local domain servers, such as the root.

DISCUSSION:
The intent of this requirement is to avoid excessive delay for the user as the search list is tested, and more importantly to prevent excessive traffic to the root and other high-level servers. For example, if the user supplied a name "X" and the search list contained the root as a component,

a query would have to consult a root server before the next search list alternative could be tried. The resulting load seen by the root servers and gateways near the root would be multiplied by the number of hosts in the Internet.

The negative caching alternative limits the effect to the first time a name is used. The interior dot rule is simpler to implement but can prevent easy use of some top-level names.

6.1.5 DOMAIN NAME SYSTEM REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
GENERAL ISSUES							
Implement DNS name-to-address conversion	6.1.1	x					
Implement DNS address-to-name conversion	6.1.1	x					
Support conversions using host table	6.1.1			x			
Properly handle RR with zero TTL	6.1.2.1	x					
Use QCLASS=* unnecessarily	6.1.2.2		x				
Use QCLASS=IN for Internet class	6.1.2.2	x					
Unused fields zero	6.1.2.3	x					
Use compression in responses	6.1.2.4	x					
Include config info in responses	6.1.2.5					x	
Support all well-known, class-indep. types	6.1.3.5	x					
Easily expand type list	6.1.3.5		x				
Load all RR types (except MD and MF)	6.1.3.6	x					
Load MD or MF type	6.1.3.6					x	
Operate when root servers, etc. unavailable	6.1.3.7	x					
RESOLVER ISSUES:							
Resolver support multiple concurrent requests	6.1.3.1		x				
Full-service resolver:	6.1.3.1			x			
Local caching	6.1.3.1	x					

Information in local cache times out	6.1.3.1	x					
Configurable with starting info	6.1.3.1		x				
Stub resolver:	6.1.3.1			x			
Use redundant recursive name servers	6.1.3.1	x					
Local caching	6.1.3.1			x			
Information in local cache times out	6.1.3.1	x					
Support for remote multi-homed hosts:							
Sort multiple addresses by preference list	6.1.3.4		x				
TRANSPORT PROTOCOLS:							
Support UDP queries	6.1.3.2	x					
Support TCP queries	6.1.3.2		x				
Send query using UDP first	6.1.3.2	x					1
Try TCP if UDP answers are truncated	6.1.3.2		x				
Name server limit TCP query resources	6.1.3.2			x			
Punish unnecessary TCP query	6.1.3.2				x		
Use truncated data as if it were not	6.1.3.2					x	
Private agreement to use only TCP	6.1.3.2			x			
Use TCP for zone transfers	6.1.3.2	x					
TCP usage not block UDP queries	6.1.3.2	x					
Support broadcast or multicast queries	6.1.3.2			x			
RD bit set in query	6.1.3.2					x	
RD bit ignored by server is b'cast/m'cast	6.1.3.2	x					
Send only as occasional probe for addr's	6.1.3.2		x				
RESOURCE USAGE:							
Transmission controls, per [DNS:2]	6.1.3.3	x					
Finite bounds per request	6.1.3.3	x					
Failure after retries => soft error	6.1.3.3	x					
Cache temporary failures	6.1.3.3		x				
Cache negative responses	6.1.3.3		x				
Retries use exponential backoff	6.1.3.3		x				
Upper, lower bounds	6.1.3.3		x				
Client handle Source Quench	6.1.3.3		x				
Server ignore Source Quench	6.1.3.3			x			
USER INTERFACE:							
All programs have access to DNS interface	6.1.4.2	x					
Able to request all info for given name	6.1.4.2	x					
Returns complete info or error	6.1.4.2	x					
Special interfaces	6.1.4.2			x			
Name<->Address translation	6.1.4.2	x					
Abbreviation Facilities:	6.1.4.3			x			

Convention for complete names	6.1.4.3	x					
Conversion exactly once	6.1.4.3	x					
Conversion in proper context	6.1.4.3	x					
Search list:	6.1.4.3			x			
Administrator can disable	6.1.4.3		x				
Prevention of excessive root queries	6.1.4.3	x					
Both methods	6.1.4.3		x				
---	---	-	-	-	-	-	--
---	---	-	-	-	-	-	--

1. Unless there is private agreement between particular resolver and particular server.

6.2 HOST INITIALIZATION

6.2.1 INTRODUCTION

This section discusses the initialization of host software across a connected network, or more generally across an Internet path. This is necessary for a diskless host, and may optionally be used for a host with disk drives. For a diskless host, the initialization process is called "network booting" and is controlled by a bootstrap program located in a boot ROM.

To initialize a diskless host across the network, there are two distinct phases:

(1) Configure the IP layer.

Diskless machines often have no permanent storage in which to store network configuration information, so that sufficient configuration information must be obtained dynamically to support the loading phase that follows. This information must include at least the IP addresses of the host and of the boot server. To support booting across a gateway, the address mask and a list of default gateways are also required.

(2) Load the host system code.

During the loading phase, an appropriate file transfer protocol is used to copy the system code across the network from the boot server.

A host with a disk may perform the first step, dynamic configuration. This is important for microcomputers, whose floppy disks allow network configuration information to be mistakenly duplicated on more than one host. Also, installation of new hosts is much simpler if they automatically obtain their configuration information from a central server, saving administrator time and decreasing the probability of mistakes.

6.2.2 REQUIREMENTS

6.2.2.1 Dynamic Configuration

A number of protocol provisions have been made for dynamic configuration.

- ICMP Information Request/Reply messages

This obsolete message pair was designed to allow a host to find the number of the network it is on. Unfortunately, it was useful only if the host already knew the host number part of its IP address, information that hosts requiring dynamic configuration seldom had.

- Reverse Address Resolution Protocol (RARP) [BOOT:4]

  RARP is a link-layer protocol for a broadcast medium that allows a host to find its IP address given its link layer address. Unfortunately, RARP does not work across IP gateways and therefore requires a RARP server on every network. In addition, RARP does not provide any other configuration information.

- ICMP Address Mask Request/Reply messages

  These ICMP messages allow a host to learn the address mask for a particular network interface.

- BOOTP Protocol [BOOT:2]

  This protocol allows a host to determine the IP addresses of the local host and the boot server, the name of an appropriate boot file, and optionally the address mask and list of default gateways. To locate a BOOTP server, the host broadcasts a BOOTP request using UDP. Ad hoc gateway extensions have been used to transmit the BOOTP broadcast through gateways, and in the future the IP Multicasting facility will provide a standard mechanism for this purpose.

The suggested approach to dynamic configuration is to use the BOOTP protocol with the extensions defined in "BOOTP Vendor Information Extensions" RFC-1084 [BOOT:3]. RFC-1084 defines some important general (not vendor-specific) extensions. In particular, these extensions allow the address mask to be supplied in BOOTP; we RECOMMEND that the address mask be supplied in this manner.

DISCUSSION:

Historically, subnetting was defined long after IP, and so a separate mechanism (ICMP Address Mask messages) was designed to supply the address mask to a host. However, the IP address mask and the corresponding IP address conceptually form a pair, and for operational

simplicity they ought to be defined at the same time and by the same mechanism, whether a configuration file or a dynamic mechanism like BOOTP.

Note that BOOTP is not sufficiently general to specify the configurations of all interfaces of a multihomed host. A multihomed host must either use BOOTP separately for each interface, or configure one interface using BOOTP to perform the loading, and perform the complete initialization from a file later.

Application layer configuration information is expected to be obtained from files after loading of the system code.

6.2.2.2 Loading Phase

A suggested approach for the loading phase is to use TFTP [BOOT:1] between the IP addresses established by BOOTP.

TFTP to a broadcast address SHOULD NOT be used, for reasons explained in Section 4.2.3.4.

6.3 REMOTE MANAGEMENT

6.3.1 INTRODUCTION

The Internet community has recently put considerable effort into the development of network management protocols. The result has been a two-pronged approach [MGT:1, MGT:6]: the Simple Network Management Protocol (SNMP) [MGT:4] and the Common Management Information Protocol over TCP (CMOT) [MGT:5].

In order to be managed using SNMP or CMOT, a host will need to implement an appropriate management agent. An Internet host SHOULD include an agent for either SNMP or CMOT.

Both SNMP and CMOT operate on a Management Information Base (MIB) that defines a collection of management values. By reading and setting these values, a remote application may query and change the state of the managed system.

A standard MIB [MGT:3] has been defined for use by both management protocols, using data types defined by the Structure of Management Information (SMI) defined in [MGT:2]. Additional MIB variables can be introduced under the "enterprises" and "experimental" subtrees of the MIB naming space [MGT:2].

Every protocol module in the host SHOULD implement the relevant MIB variables. A host SHOULD implement the MIB variables as defined in the most recent standard MIB, and MAY implement other MIB variables when appropriate and useful.

6.3.2 PROTOCOL WALK-THROUGH

The MIB is intended to cover both hosts and gateways, although there may be detailed differences in MIB application to the two cases. This section contains the appropriate interpretation of the MIB for hosts. It is likely that later versions of the MIB will include more entries for host management.

A managed host must implement the following groups of MIB object definitions: System, Interfaces, Address Translation, IP, ICMP, TCP, and UDP.

The following specific interpretations apply to hosts:

- ipInHdrErrors

  Note that the error "time-to-live exceeded" can occur in a host only when it is forwarding a source-routed datagram.

- ipOutNoRoutes

  This object counts datagrams discarded because no route can be found. This may happen in a host if all the default gateways in the host's configuration are down.

- ipFragOKs, ipFragFails, ipFragCreates

  A host that does not implement intentional fragmentation (see "Fragmentation" section of [INTRO:1]) MUST return the value zero for these three objects.

- icmpOutRedirects

  For a host, this object MUST always be zero, since hosts do not send Redirects.

- icmpOutAddrMaskReps

  For a host, this object MUST always be zero, unless the host is an authoritative source of address mask information.

- ipAddrTable

  For a host, the "IP Address Table" object is effectively a table of logical interfaces.

- ipRoutingTable

  For a host, the "IP Routing Table" object is effectively a combination of the host's Routing Cache and the static route table described in "Routing Outbound Datagrams" section of [INTRO:1].

  Within each ipRouteEntry, ipRouteMetric1...4 normally will have no meaning for a host and SHOULD always be -1, while ipRouteType will normally have the value "remote".

  If destinations on the connected network do not appear in the Route Cache (see "Routing Outbound Datagrams section of [INTRO:1]), there will be no entries with ipRouteType of "direct".

DISCUSSION:

The current MIB does not include Type-of-Service in an ipRouteEntry, but a future revision is expected to make

this addition.

We also expect the MIB to be expanded to allow the remote management of applications (e.g., the ability to partially reconfigure mail systems). Network service applications such as mail systems should therefore be written with the "hooks" for remote management.

6.3.3 MANAGEMENT REQUIREMENTS SUMMARY

FEATURE	SECTION	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
Support SNMP or CMOT agent	6.3.1		x				
Implement specified objects in standard MIB	6.3.1		x				

7. REFERENCES

This section lists the primary references with which every implementer must be thoroughly familiar. It also lists some secondary references that are suggested additional reading.

INTRODUCTORY REFERENCES:

[INTRO:1] "Requirements for Internet Hosts -- Communication Layers," IETF Host Requirements Working Group, R. Braden, Ed., RFC-1122, October 1989.

[INTRO:2] "DDN Protocol Handbook," NIC-50004, NIC-50005, NIC-50006, (three volumes), SRI International, December 1985.

[INTRO:3] "Official Internet Protocols," J. Reynolds and J. Postel, RFC-1011, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

[INTRO:4] "Protocol Document Order Information," O. Jacobsen and J. Postel, RFC-980, March 1986.

[INTRO:5] "Assigned Numbers," J. Reynolds and J. Postel, RFC-1010, May 1987.

This document is republished periodically with new RFC numbers; the latest version must be used.

TELNET REFERENCES:

[TELNET:1] "Telnet Protocol Specification," J. Postel and J. Reynolds, RFC-854, May 1983.

[TELNET:2] "Telnet Option Specification," J. Postel and J. Reynolds, RFC-855, May 1983.

[TELNET:3] "Telnet Binary Transmission," J. Postel and J. Reynolds, RFC-856, May 1983.

[TELNET:4] "Telnet Echo Option," J. Postel and J. Reynolds, RFC-857, May 1983.

[TELNET:5] "Telnet Suppress Go Ahead Option," J. Postel and J.

Reynolds, RFC-858, May 1983.

[TELNET:6] "Telnet Status Option," J. Postel and J. Reynolds, RFC-859, May 1983.

[TELNET:7] "Telnet Timing Mark Option," J. Postel and J. Reynolds, RFC-860, May 1983.

[TELNET:8] "Telnet Extended Options List," J. Postel and J. Reynolds, RFC-861, May 1983.

[TELNET:9] "Telnet End-Of-Record Option," J. Postel, RFC-855, December 1983.

[TELNET:10] "Telnet Terminal-Type Option," J. VanBokkelen, RFC-1091, February 1989.

This document supercedes RFC-930.

[TELNET:11] "Telnet Window Size Option," D. Waitzman, RFC-1073, October 1988.

[TELNET:12] "Telnet Linemode Option," D. Borman, RFC-1116, August 1989.

[TELNET:13] "Telnet Terminal Speed Option," C. Hedrick, RFC-1079, December 1988.

[TELNET:14] "Telnet Remote Flow Control Option," C. Hedrick, RFC-1080, November 1988.

SECONDARY TELNET REFERENCES:

[TELNET:15] "Telnet Protocol," MIL-STD-1782, U.S. Department of Defense, May 1984.

This document is intended to describe the same protocol as RFC-854. In case of conflict, RFC-854 takes precedence, and the present document takes precedence over both.

[TELNET:16] "SUPDUP Protocol," M. Crispin, RFC-734, October 1977.

[TELNET:17] "Telnet SUPDUP Option," M. Crispin, RFC-736, October 1977.

[TELNET:18] "Data Entry Terminal Option," J. Day, RFC-732, June 1977.

[TELNET:19] "TELNET Data Entry Terminal option -- DODIIS Implementation," A. Yasuda and T. Thompson, RFC-1043, February 1988.

FTP REFERENCES:

[FTP:1] "File Transfer Protocol," J. Postel and J. Reynolds, RFC-959, October 1985.

[FTP:2] "Document File Format Standards," J. Postel, RFC-678, December 1974.

[FTP:3] "File Transfer Protocol," MIL-STD-1780, U.S. Department of Defense, May 1984.

This document is based on an earlier version of the FTP specification (RFC-765) and is obsolete.

TFTP REFERENCES:

[TFTP:1] "The TFTP Protocol Revision 2," K. Sollins, RFC-783, June 1981.

MAIL REFERENCES:

[SMTP:1] "Simple Mail Transfer Protocol," J. Postel, RFC-821, August 1982.

[SMTP:2] "Standard For The Format of ARPA Internet Text Messages," D. Crocker, RFC-822, August 1982.

This document obsoleted an earlier specification, RFC-733.

[SMTP:3] "Mail Routing and the Domain System," C. Partridge, RFC-974, January 1986.

This RFC describes the use of MX records, a mandatory extension to the mail delivery process.

[SMTP:4] "Duplicate Messages and SMTP," C. Partridge, RFC-1047, February 1988.

[SMTP:5a] "Mapping between X.400 and RFC 822," S. Kille, RFC-987, June 1986.

[SMTP:5b] "Addendum to RFC-987," S. Kille, RFC-???, September 1987.

The two preceding RFC's define a proposed standard for gatewaying mail between the Internet and the X.400 environments.

[SMTP:6] "Simple Mail Transfer Protocol," MIL-STD-1781, U.S. Department of Defense, May 1984.

This specification is intended to describe the same protocol as does RFC-821. However, MIL-STD-1781 is incomplete; in particular, it does not include MX records [SMTP:3].

[SMTP:7] "A Content-Type Field for Internet Messages," M. Sirbu, RFC-1049, March 1988.

DOMAIN NAME SYSTEM REFERENCES:

[DNS:1] "Domain Names - Concepts and Facilities," P. Mockapetris, RFC-1034, November 1987.

This document and the following one obsolete RFC-882, RFC-883, and RFC-973.

[DNS:2] "Domain Names - Implementation and Specification," RFC-1035, P. Mockapetris, November 1987.

[DNS:3] "Mail Routing and the Domain System," C. Partridge, RFC-974, January 1986.

[DNS:4] "DoD Internet Host Table Specification," K. Harrenstein, RFC-952, M. Stahl, E. Feinler, October 1985.

SECONDARY DNS REFERENCES:

[DNS:5] "Hostname Server," K. Harrenstein, M. Stahl, E. Feinler, RFC-953, October 1985.

[DNS:6] "Domain Administrators Guide," M. Stahl, RFC-1032, November 1987.

[DNS:7] "Domain Administrators Operations Guide," M. Lottor, RFC-1033, November 1987.

[DNS:8] "The Domain Name System Handbook," Vol. 4 of Internet Protocol Handbook, NIC 50007, SRI Network Information Center, August 1989.

SYSTEM INITIALIZATION REFERENCES:

[BOOT:1] "Bootstrap Loading Using TFTP," R. Finlayson, RFC-906, June 1984.

[BOOT:2] "Bootstrap Protocol (BOOTP)," W. Croft and J. Gilmore, RFC-951, September 1985.

[BOOT:3] "BOOTP Vendor Information Extensions," J. Reynolds, RFC-1084, December 1988.

Note: this RFC revised and obsoleted RFC-1048.

[BOOT:4] "A Reverse Address Resolution Protocol," R. Finlayson, T. Mann, J. Mogul, and M. Theimer, RFC-903, June 1984.

MANAGEMENT REFERENCES:

[MGT:1] "IAB Recommendations for the Development of Internet Network Management Standards," V. Cerf, RFC-1052, April 1988.

[MGT:2] "Structure and Identification of Management Information for TCP/IP-based internets," M. Rose and K. McCloghrie, RFC-1065, August 1988.

[MGT:3] "Management Information Base for Network Management of TCP/IP-based internets," M. Rose and K. McCloghrie, RFC-1066, August 1988.

[MGT:4] "A Simple Network Management Protocol," J. Case, M. Fedor, M. Schoffstall, and C. Davin, RFC-1098, April 1989.

[MGT:5] "The Common Management Information Services and Protocol over TCP/IP," U. Warrier and L. Besaw, RFC-1095, April 1989.

[MGT:6] "Report of the Second Ad Hoc Network Management Review Group," V. Cerf, RFC-1109, August 1989.

Security Considerations

There are many security issues in the application and support programs of host software, but a full discussion is beyond the scope of this RFC. Security-related issues are mentioned in sections concerning TFTP (Sections 4.2.1, 4.2.3.4, 4.2.3.5), the SMTP VRFY and EXPN commands (Section 5.2.3), the SMTP HELO command (5.2.5), and the SMTP DATA command (Section 5.2.8).

Author's Address

Robert Braden
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Phone: (213) 822 1511

EMail: Braden@ISI.EDU

Network Working Group
Request for Comments: 1124

B. Leiner
RIACS
September 1989

Policy Issues in Interconnecting Networks

Status of this Memo

To support the activities of the Federal Research Internet Coordinating Committee (FRICC) in creating an interconnected set of networks to serve the research community, two workshops were held to address the technical support of policy issues that arise when interconnecting such networks. Held under the auspices of the Internet Activities Board at the request of the FRICC, and sponsored by NASA through RIACS, the workshops addressed the required and feasible technologies and architectures that could be used to satisfy the desired policies for interconnection.

The purpose of this RFC is to report the results of these workshops. Distribution of this memo is unlimited.

TABLE OF CONTENTS

Preface

This report documents the results of two workshops held at the request of the Federal Research Internet Coordinating Committee and under the auspices of the Internet Activities Board. As such, this report represents the work of a large number of people (listed in Section 7). Without their efforts, these results would not have been possible. The author (really more of an editor) would like to acknowledge their efforts and contributions, and thank them for their cooperation in making the workshops a success.

1. Introduction

Computer networking has become pervasive and basic to the conduct of scientific and academic activities. To provide the needed networking support to these activities, each of the agencies funding research has proceeded to establish one or more agency funded computer networks.

Recognizing the importance of such networking support, the Office of Science and Technology Policy (OSTP) working with the appropriate personnel from the research-funding agencies on the Federal Coordinating Council on Science Engineering and Technology (FCCSET) Committee on High-Speed Networks developed a set of recommendations for the evolution and enhancements of scientific and academic networks. These recommendations are described in three phases. The first phase addresses the interconnection of the various agency networks into a ubiquitous networking capability serving several hundred universities and research institutions with a backbone network operating 1.5 Mb/s. The second phase involves upgrading the network backbone to 45 Mb/s and connecting additional universities and other research institutions. The third phase involves the development and installation of a high bandwidth (Gb/s) networking capability.

The motivation for the first two phases are to achieve good performance in a cost effective manner. The scientific and academic community is best served by an interconnected ubiquitous networking capability rather than a set of partitioned networks supporting only subsets of the community. Costs can be reduced and performance improved through sharing of resources and using cross-support (e.g., using one agency's network to serve an institution for another agency's purposes rather than having to connect each institution to every network.)

To accomplish these objectives, the Federal Research Internet Coordinating Committee (FRICC) was formed. Consisting of representatives from the key research agencies (NSF, DARPA, NASA, and DOE), this ad hoc group has been developing strategies for interconnection of networks and evolution of the Internet in accordance with the OSTP recommendations for Phases 1-3. In the process of developing such plans, it became apparent that a set of issues needed to be addressed concerning the various agency policies for their research networks in light of the desire to interconnect such networks.

This report documents the results of a series of two workshops (18-20 June 1988 at NASA Ames Research Center and 8-10 November 1988 at MIT) held to address these issues. Held under the auspices of the Internet Activities Board (IAB) at the request of the FRICC, and sponsored by NASA through RIACS, the workshops addressed the required and feasible technologies and architectures that could be used to satisfy the desired policies for interconnection.

The issues were divided into four categories, and working groups established within the workshops to address each area. The first working group addressed the policies themselves. Working with the members of the FRICC, the initial statements

of agency policies were refined so that the rest of the workshop attendees could better understand the desired and required policies. The second working group addressed issues associated with access control to network resources. The third working group addressed the techniques required to support the sharing of networking resources in accordance with agreed upon policies. The fourth working group focussed on the end-to-end services required to support an interconnected set of networks.

Each of the working groups prepared summary reports of their deliberations. These reports are contained in Sections 3-6 of this document. The report of the policy working group attempts to summarize the existing policies of each of the agencies, particularly with respect to interconnection with other networks. The other three working groups focussed on the technology issues needed to be addressed in light of those policies. In each case, the working group report discusses the issues and develops an evolutionary capability with the goal of fully addressing the agency policies. Summaries of these reports are contained in the next section.

It is hoped that the results documented in this report will help the FRICC and the rest of the research community in achieving this exciting objective: a national research networking capability.

2. Workshop Summary

Driving the workshop were the policies of the individual agencies and a desire to interconnect the networks in a way that was satisfactory to those agencies. A prime policy driver appeared to be OMB Circular A130, which states that appropriate mechanisms must be used to assure some level of accounting for the use of the various networks. Another important policy driver was the need for agencies to assure that sharing of networks did not adversely impact the support of the individual agency users on their specific networks. This led in some cases to the need to be able to dedicate a portion (sometimes all during a specified time period) of an agency network to supporting its own users. Finally, the need to provide appropriate supporting end-to-end services, including security issues, led to the need for coordinating such services.

To facilitate the discussion of the technology issues and the presentation of results, it was decided to describe the evolution of capability in four phases. Phase 0 represented currently deployed and available capability. While not necessarily being currently used for the support of the policy issues, the capabilities of Phase 0 were viewed as being currently available and could be used starting today. Phase 1 consisted of capabilities that were developed and deployed at a limited number of sites. Thus, the issues involved in using such capabilities involved mainly those of widespread deployment (plus perhaps some limited amount of development associated with, e.g., porting of software). Phase 2 represented capabilities that were relatively well understood (little research required) but would require development activity before they could be used to support the policies for interconnection. Phase 3 capabilities require research to achieve, and thus represent the most future capability.

While these phases of capability represent evolution in availability, they should not be viewed as evolution in starting time for action. In all cases, research and development activities would have to start today in order that these capabilities be available in a timely manner.

As the working group on access control discussed the required technologies and mechanisms, it became clear that an important technology driver was the need to label packets with the appropriate information to make determinations of routing and resource allocation internal to the interconnected networks. For example, if certain links in a NASA network was to be restricted to use only by NASA users (even if accessing the network through an NSF network), it would be necessary to provide such labelling information in the packet. The report of the working group discusses the information that needs to be carried in such labels, requirements for authentication, and some potential experiments and development that should be carried out to achieve the required capability.

The working group on resource sharing focussed on the technologies that would allow fair sharing of resources between the participating agencies. The key issue that emerged from the discussions of this working group was the need to develop global

algorithms that permitted sharing and prioritization of the use of resources. As an example, it is relatively easy for an agency to block low-priority traffic from traversing its network during a period of high internal requirement. It is not so easy to do so and assure that the external users still can receive the resources they need from the interconnected internet.

The working group on end-to-end services focussed on those services that are required from a user's perspective from the overall system, and need to be coordinated across the interconnected networks. For example, directory and security services must be provided across the interconnected system. The key element emerging from the group discussions was the need to establish a consistent set of mechanisms to interconnect the various end-to-end services. These must be provided in a secure manner to assure that the security services fulfill their function.

The working groups identified the need to carry out supporting experiments and analysis to carry forward the interconnection of the networks, e.g., to make decisions about the need for stream versus transaction support. Each group developed a set of possible experiments and activities in accordance with the phases of development discussed above. These are summarized in Tables I-III.

A number of possible follow-on activities were identified to be passed on to the various Task Forces of the IAB. These are shown in Table IV.

In summary, the workshop identified a number of critical issues and identified areas where further research and experimentation is required. It is hoped that these results help provide a "road map" for how to satisfy agency policies and requirements in the interconnection of networks.

Table I
Access Control Projects

Phase 0	Access Control based on source/destination access matrix (for traffic not transiting network)
Phase 1	*Statspy* experiment to determine and define requirement for transactions
	"ESnet hack" for limited access control based on source/destination addresses.
	"Xerox hack" for limited access control based on source/destination addresses.
Phase 2	Coloring of stream packets
	Simple colors/labelling
	Route filtering for access control using source/destination addresses
	Incorporate "Xerox hack" into other gateways
	Authentication and signature architecture
Phase 3	Use of complex credentials
	Use of policy gateways in route computation

Table II
Resource Sharing Projects

Phase 0	Simple route filtering
Phase 1	Run *Statspy* to determine source/destination traffic flows (to comply with A130 traffic monitoring requirements)
Phase 2/3	50/50 resource management for link sharing
	Color packets and observe behavior to improve traffic monitoring
	Fast encryption of route and certificate packets, to secure traffic monitoring and control
	Fast mapping from source/destination to packet label/color
	Demonstration of gateway using soft state
	Define and support policy source routing
	Synthesis of source route
	Management controls and protocols
	Composition of policy terms
	Define and structure route set-up protocols

Table III
End-to-End Services Projects

Phase 0	User/process authentication using passwords (origin authentication)
	Mail relays for both function and system isolation
	Name domains system for host name to address mapping
Phase 1	User/process authentication using challenge/response or some other protocol (origin authentication)
	Secure-ID or other authentication technologies
	Challenge/response technologies (overlaps with the previous line)
	Kerberos (authentication server)
Phase 2	Authentication using certificates
	Integrity (MACs, checksums) and labelling
	Key distribution and management
	Secure mail (see RFC 1113)
	Certificates (see same RFC)
	Security of distributed white pages
	Integrity labelling, tools (MACs, checksums)
	Distributed white pages for the entire Internet
Phase 3	Use of VISAs
	Certification across peer domains
	Distributed computation
	National file system
	Trusted accounting
	Firewalls for end-to-end services
	Integrity of data across international boundaries with agreed upon cryptographic technologies
	Use zero-sum knowledge to have a third party to assure integrity without secrecy for such cases

Table IV
Projects for IAB Task Forces

ETETF	Handling of quality of service in gateways
ANTF	Phases 2 and 3 of resource sharing activities
IETF	Policy routing
Privacy	End-to-end privacy services
???	End-to-end services

3. Working Group on Interconnection Policies

Working Group 0 Members

Steve Wolff (Chair)	NSF
Guy Almes	Rice
Matt Bishop	Dartmouth
Brian Boesch	DARPA
Scott Brim	Cornell
Phill Gross	NRI
Dan Hitchcock	DoE
Russ Mundy	DCA
Tony Villasenor	NASA

Network resource sharing is encouraged by the potential for economies of scale both in communication link acquisition cost and in provision of value-added network services (the latter not yet demonstrated in the Internet, but consistent with telephone company experience); it is suggested by the Congressionally-ordered network study that resulted in the OSTP report *A Research and Development Strategy for High Performance Computing;* and it is mandated by OMB Circular A-130. Technical forces in the same direction include the additional connectivity each agency provides to its clients (actual or potential) by acquiring the use of nets belonging to other agencies at little or no additional cost, and the robustness afforded by the sharing of redundant paths or other forms of "excess" capacity.

The agencies represented on the FRICC, however, have differing missions and requirements, and these differences are reflected in differing rules and procedures for network usage. WG0 was created to explicate the rules for network use of the FRICC agencies, for those rules -- particularly the differences among them -- form the foundation upon which the technical specifications of "policy-based routing" must be built. This report, therefore, is the primary input to the technical Working Groups WG1, WG2, and WG3.

Making all FRICC agencies' network use rules the same is NOT a goal of WG0. Each FRICC agency has more-or-less well-formulated rules for the use of its network in the absence of explicit interconnection with other networks and the attendant "foreign" traffic. These rules are given below. Currently, no agency has rules for interconnection with:

- networks of other FRICC agencies,
- networks of other countries,
- commercial networks, or
- "sensitive" networks (e.g., SDInet, NASA mission-critical nets);

consistent formulation of such rules will be discussed in future FRICC meetings.

It was however noted that, in dealing with subordinate (not peer) networks, NSF has required traffic presented to the NSFnet backbone to conform to NSF rules of acceptable use; DoE on the other hand is tending to the more liberal policy of carrying any traffic that meets the rules for acceptable use of the agency network offering the traffic.

3.1. Existing Policies, Summarized

The following is a summary of the existing policies for network usage of the FRICC member agencies.

NSF (draft, summarized):

- Purpose is to support scientific research and other scholarly activities.
- Use to support research or instruction at not-for-profit institutions of instruction and/or research is acceptable, whether all parties to the use are located or employed at such institutions or not.
- Activities in direct support of acceptable use are acceptable.
- Use for research or instruction by for-profit institutions may or may not be acceptable, and will be reviewed case-by-case.
- Commercial use by for-profit institutions is generally not acceptable.

DoE (draft, summarized):

- Use in which at least one party is supported by Energy Sciences funds is acceptable.
- Use by persons at DoE sites is acceptable, even if they are not supported by Energy Sciences funds.
- Advertising or promotional activities are not acceptable.
- Use in direct competition with commercial services is not acceptable.

NASA (draft, summarized):

- Purposes are to support NASA space science programs, to support collaborating science activities (e.g., with ESA, NOAA, USGS), and to support NASA contractors (e.g., those involved in building scientific sensors and spaceborne hardware).
- Other activities may be supported on a case-by-case basis, provided there is no impact to the NASA programs.
- No Eastern bloc access.
- Shared use of network facilities must be controllable and annually accounted for.
- NASA networking facilities may be made available for other uses and users on a cost-reimbursable basis.

- Direct competition with commercial services is not acceptable.

DARPA:

- Purpose is to support network research and other DARPA research objectives.
- There may be "forbidden routes" for some traffic.

DDN (excluding ARPANET and the proposed DRI):

- Use is for DoD business only, unless otherwise approved by JCS.
- All connections to other nets strictly regulated by mailbridges (now) or trusted guard gateways (future).
- Facilities must comply with DoD Security Architecture and with DoD Directive 5200.28 which requires C2 certification for sensitive unclassified information.

3.2. Refined Policy Statements

As a result of the first workshop discussions on policy, Dr. Cerf met with the various agency representatives to refine the policy statements. The results of these meetings were as follows. Note that these statements are those of the workshop and do not represent official agency policies. Each policy is represented in Clark's Policy Term (PT) notation[1] and then described in English. The standard Clark Form for PTs (Hsrc,ARsrc,ARent)(Hdst,ARdst,ARexit){UCI}{Cg} FRICC={DOE,NASA,DCA,NSF} where H=Host, AR=Autonomous Region, src=source, dst=destination, ent=entry (previous hope), exit=exit (last hop, F=Federal Agency Net, Re=Regional, U=University, Co=Commercial Corporation, and Cc=Commercial Carrier. All PTs are assumed to be symmetrical in these examples.

NSF

NSF1: (*,*,{F/Re})(*,*,{F/Re}){research,support}{unauthenticated UCI, no-per-pkt charge}

i.e., NSF will carry traffic for any host connected to a F/Re network talking to any other host connected to a F/Re via any F/Re entry and exit network, so long as it is being used for research or support. There is no authentication of the UCI and no per packet charging. NSFnet is a backbone and so does not connect directly to universities or companies. Thus the indication of {F/Re} instead of {F/Re/U/Co} as ARent and ARexit.[2]

NSF2: ({User svcs, Expert Svcs}, {NSF},{F/Re})(*,{F/Re},{F/Re})

i.e., NSF will carry traffic to user and expert services hosts in NSF Autonomous Region (AR) to/from any F/Re AR, via any F/Re AR. These are the only things that

[1] D.D. Clark, "Policy Routing in Internet Protocols," Version 1.1, May 19, 1988.

[2] Note: I can't actually decide whether it should be as stated above or (*,{F/Re},{F/Re})(*,{F/Re},{F/Re})

directly connect to NSFnet.

DOE

DOE1: (*,DOE,-)(*,*,*){research,support}{unauthenticated UCI, no-per-packet charge}

i.e., DOE will carry traffic to and from any host directly connected to DOE so long as it is used for research or support. There is no authentication of the UCI and no per packet charging.

DOE2: (*,*,{F/Re})(*,*,{F/Re}){}{unauthenticated UCI, no-per-pkt charge}

i.e., DOE will carry traffic for any host connected to a F/Re network talking to any other host connected to a F/Re via any F/Re entry and exit network without regard to the UCI. There is no authentication of the UCI and no per packet charging. (In other words, DOE is more restrictive with its own traffic than with traffic it is carrying as part of a resource sharing arrangement.)

NASA

NASA1: (*,*,*)(*,NASA,-){NASA-research,support}{unauthenticated UCI,no-per-packet-charge}

i.e., NASA will accept any traffic to/from members of the NASA AR, but no transit. No UCI authentication and no per packet charge.

NASA2: (*,{F},*)(*,{F},*){research,support}{per-packet accounting, limited to n% of available BW}

i.e., NASA will carry transit traffic to/from other federal agency networks if they are for research and if the total use of available BW by non-NASA Federal agencies is below n%.[3]

NASA3: (*,{Co},*) (*,{F/R/U},-) {NASA research,support} {not authenticated UCI, no per packet charge}

i.e., NASA will carry commercial traffic to federal, regional, and university ARs for NASA research or support, but it will not allow transit. The particular entry AR is not important.

NASA4: (*,*,-)(*,*,-){}{per-packet-charge to recoup cost, limited to n% of available BW}

i.e., On a case by case basis, NASA will consider non-NASA traffic on a cost-reimbursed basis. It will not carry transit traffic on this basis.

[3] Note that this non-interference policy type needs some more work in terms of integrating it into the routing algorithms.

DARPA

DARPA1: (*,*,*)(*,DARPA,-){research,support}{unauthenticated-UCI, no per packet charge}

i.e., DARPA will carry traffic to/from any host in DARPA AR from any external host that can get it there so long as UCI is research or support. No UCI authentication or per packet charge.

DARPA2: (*,*,{F/R/U/Co})(*,*,{F/R/U/Co}){research,support}{unauthenticated-UCI, no per packet charge, non-interference basis}

i.e., DARPA will carry traffic for any host connected to a F/Re/U/Co network talking to any other host connected to a F/Re/U/Co via any F/Re/U/Co entry and exit network, so long as it is being used for research or support, and the network is not heavily congested! There is no authentication of the UCI and no per packet charging.[4]

DCA

DDN1: (mailbridge,DDN,-)(*,{F/Re},{F/Re}){research,support}{unauthenticated UCI, all incoming packets marked, per-kilopacket charge}

i.e., DDN will not carry any transit traffic. It will only accept and send traffic to and from its mailbridge(s) and only from and to hosts on other F/Re nets.

An Example Regional[5]

Regional1: (*,{F/Re/U},{F/Re/U})(*,{F/Re/U},NSF){research,support} {unauthenticated UCI, no-per-packet charge}

i.e., The Regional will carry traffic from/to any directly connected F/Re/U network to any F/Re/U network via NSF if it is for a research or support UCI. (NSF requires that all Regional networks only forward to it traffic that complies with its, NSF's, policies!)

Regional2: (*,{F/Re/U},{F/Re/U})(*,{F/Re/U},Cc){}{unauthenticated UCI, per-kilopacket charge}

i.e., The Regional will carry traffic from/to any directly connected F/Re/U network to any F/Re/U network via a commercial carrier regardless of its UCI. In this case, the packets are charged for since the commercial carrier charges per kilopacket.

---

[4] Note: DARPA would like to say something about the need to enter the DARPA AR at the point closest to the destination, but I don't know how to express this.

[5] Note: No interview was done for this one. This is just a guess.

4. Access Control for Network Switching and Transmission Resources

Working Group 1 Members

Steve Kent (Chair)	BBN
Guy Almes	Rice
Bill Bostwick	Los Alamos
Marsha Branstad	DoD
Vint Cerf	NRI
Deborah Estrin	USC
Tony Hain	Livermore
Dan Lynch	ACE
Russ Mundy	DCA
Anita Holmgren	Unisys

4.1. Introduction

This report reflects discussions among the members of working group with regard to network access control for the National Research Internet (NRI). The NRI will be composed of network resources contributed by various organizations (primarily agencies of the Federal government). The operational model for the NRI is that of a collection of autonomous, administrative domains (referred to as "domains" within this report), each of which manages a collection of network transmission and/or switching resources. (Other, higher level resources also may be shared across domain boundaries, but these are not the focus of the access controls discussed herein.) Some of these network resources are owned or leased exclusively on behalf of the administrative domain responsible for the resource, whereas other resources may be jointly paid for and administered.

There is a perceived requirement that a domain provide access control for the network transmission and switching resources that comprise it. This form of access control is distinguished from measures oriented toward controlling access to subscriber resources, e.g., workstations, file servers, etc. Rather, these measures are intended to apply to communication paths which transit gateways, circuits, networks, etc.

There are several motivations for introducing network resource access controls. The organizations which will contribute network resources or funding for shared resources to the NRI need to be satisfied that sharing of these network resources can be controlled in such a fashion as to accord priority to designated users or groups of users and to account for resource usage in accordance with OMB guidelines. It may be necessary to bill for usage of some resources, especially commercial facilities connected to the NRI. Some organizations have adopted policies that prohibit transport of data from certain classes of users across their networks.

This report examines various aspects of network resource access control measures in the NRI context, including bases for making access control decisions (policy inputs), communication scenarios to be supported, mechanisms for enforcing access control policies, and assurance issues associated with enforcement. Formulation of specific access control policies is outside the scope of this report and is addressed by the report of Policy Working Group.

This report has been prepared by the members of the working group as a result of discussions that took place at workshops sponsored by NASA on June 15-17, 1988 and November 8-10, 1988. Additional inputs have been prepared by working group members during the interval between these workshops and co-ordinated by the chair.

4.2. Access Control Policy Issues

4.2.1. Policies and Models

Any discussion of access control measures should begin with a characterization of the policies which the measures are to enforce, and a definition of the model that underlies the policies. There are various ways to characterize access control policies, one of which (ISO 7498-2) considers two axes: 1) the basis on which access control decisions are made (rule-based or identity-based), and 2) the entity who defines the policy (user-directed or administratively directed). For the NRI environment, we anticipate the policies are all administratively directed since they represent constraints imposed by organizations which contribute resources to the NRI, not individual subscribers.

Discussions with organizational representatives suggest that both identity-based and rule-based policies may be employed. For example, in some circumstances an access control decision will be made based on the identity of the user (or a class of which the user is a member) requesting access. In many cases, possession of a token indicating agency authorization for resource use, perhaps coupled with time and day of week inputs, will form the basis for the access control decision. These two examples illustrate identity-based and rule-based policies and policies that combine both policy bases are also possible.

The security access model we assume for the NRI environment is a traditional one involving subjects and objects. Subjects are active entities (e.g., processes) which are accorded some access privileges with respect to objects. The processes execute in various subscriber equipments (hosts, workstations, servers, etc.) either acting on behalf of users (individuals or groups) or acting as entities independent of any specific, human user. Objects in this context are typically data paths through the NRI, and thus they implicitly entail the use of transmission and switching resources. (Alternatively, we could consider these resources individually as the objects and the paths as compositions of the component parts.)

4.2.2. Policy Inputs

A refinement of policy characterization is provided by considering the range of inputs on which access control decisions will be made. These inputs can be divided into two categories (somewhat arbitrarily): 1) data implicitly available to the enforcement entities, e.g., time and date or utilization and connectivity status, and 2) data explicitly provided by subjects, e.g., in packet headers. Note that this characterization does not specify whether the explicit inputs are provided in every packet or only in some packets, how the inputs are validated, etc. These details are critical components of an architecture, not just an implementation, and thus the final form of this list should take into account these considerations as well as the rationale provided below.

Based on inputs from agency representatives present at the workshops, it appears desirable that information on local resource utilization and global connectivity be major implicit inputs in access control decisions. The rationale is that many agencies appear to be adopting policies which permit sharing of resources by "outside subjects" on a "non-interference" basis. This requires that the enforcement mechanisms be cognizant of the resource utilization status (congestion measures) so as to determine what constitutes non-interfering sharing.[6] It also requires some explicit identification of subjects to determine whether the non-interference criteria should be applied. More refined sharing policies could take into account relative priorities for various subjects, type of service (TOS)-based routing decisions, etc. The Resource Sharing Working Group is focusing on routing issues which take into account quantitative measures related to TOS. In contrast, this group has focused more on policies in which such quantitative measures are not primary inputs to the access control decision. This suggests that a combination of the architectural proposal from both groups will be required to address some of the access control policy requirements described at the workshops.

Data that might be explicitly required from a subject was the topic of much discussion. A list of candidate data items was developed and is discussed below. Although not all administrative domains might require all of these inputs for an access control decision, it has been suggested that the list be universally agreed upon among all domains. The argument is that global routing determinations are affected by local access control decisions and that it is desirable to enable subscribers (or their local policy route servers) to calculate permitted routes before initiating transmission of data along a path. In order to perform such calculations, each domain must publish its access control policy and the inputs to the policy must be universally interpretable. Thus there is a strong motivation to define a minimum set of explicit inputs to these

[6] There is a potential conflict here in using local congestion measures as inputs to an access control decision. It is desirable for a remote subject (e.g., policy controller) to determine in advance if a specified transmission resource can be used in constructing a (policy) route between two points in the NRI, for reasons elucidated by Dave Clark in his policy routing paper. Thus the conflict arises if either the remote subject cannot obtain the necessary local congestion measures or if these measures are very dynamic.

policies.

At one point in the discussion it was suggested that any inputs to access control decisions that were not universally interpretable could be accommodated by allowing for "domain specific" data items. Such data items would be interpreted by only a few domains (perhaps only a single domain) along a route. However, we note that this concept does not seem to be in concert with the principle cited earlier (and discussed in Clark's paper), i.e., subjects should be able to predict access control decisions for any domain through which they might construct a route. Thus the concept of a domain-specific access control data item as an "escape" mechanism for including additional inputs to access control decisions may not be appropriate. Recall that no domain is required to employ all the supplied inputs in making an access control decision and thus inclusion of a data item in a widely known collection need not impose on domains that do not wish to make use of the data item.

Since the administrative domains often represent federal agencies (e.g., DOE, NASA, NSF), it was perceived that there should be some means of representing an agency's granting authorization for resource use to the subject. This might be a hierarchic data item, specifying both an agency identifier and further defining the subject's privileges as granted by the agency. For example, an agency such as DOE might grant somewhat different privileges to its employees, to its grantees and their staff, and to other individuals engaged in work that is viewed as supportive to the agency mission (though not necessarily funded by the agency). This effect might be achieved by issuing to each of these subjects credentials that specify some form of affiliation with the agency in question but with different qualifiers, depending on the nature of the affiliation. Thus we envision a compound access control data item that will specify an AGENCY AFFILIATION INDICATOR, consisting of an AGENCY ID and AFFILIATION CLASS.

It is anticipated that some form of accounting for use of resources will be required in many circumstances within the NRI. OMB regulations requires this accounting at the agency level, and thus it might be sufficient to rely on the agency affiliation data to satisfy this requirement. In other cases, an orthogonal account identifier might be required and so we allow for inclusion of a BILLING CODE[7] as part of the explicit access control data. This may prove especially important in contexts where commercial facilities are employed.

In the most extreme cases it may be necessary for an individual subject to be identified, either for accounting or for access authorization. Although details for such an identifier were not discussed, it seems likely that a hierarchic data item would be appropriate, with a domain identifier used to specify the authority that vouches for the subject's identity, plus a subject identifier that is unique within the domain. Even if users need not be identified as individuals, groups of users may be identified for

[7]Note that this item may enter into the decision process or may be employed only for accounting.

authorization purposes. Hence we expect to see a SUBJECT ID compound data item consisting of a DOMAIN ID and a USER ID, where this later data item may represent a group of users rather than a single individual.

The (ultimate) internet layer (IP or CLNP) source and destination addresses associated with a packet, possibly including protocol identification data, are also viewed as legitimate inputs to access control decisions, but for different reasons that the other data items described above. Use of addresses provides a convenient means of prohibiting access by specific devices or groups of devices (e.g., entire LANs) should it become necessary to revoke access at this granularity. Also, one can imagine simple access control policies that might be employed initially in the NRI and which would be based only (or primarily) on these values. Finally, we note that these data items are already included in every packet and are examined in the course of effecting the routing decisions which are the heart of the internet switching system and which are thus intimately related to the objects being protected. Thus even if these data items are not used in formulating an access control decision, they play an important role in the enforcement of the policies. It is worth noting that the preceding discussion of data items which are candidates as explicit inputs to access control decisions does not address how or when these data items are created, distributed, validated, or transported in subscriber traffic. These are important architectural issues, some of which are addressed in later portions of this document.

4.3. Communication Scenarios

4.3.1. Connection-Oriented Communication

Different types of communication scenarios may impose differing requirements on access control mechanisms. We observe that fine-grained access control mechanisms for connection-oriented communications are better understood and easier to implement than corresponding mechanisms for connectionless communication. The rationale behind this observation is that connection-oriented communication implies some connection establishment procedure. This procedure is a natural place to perform access control checks and to terminate the procedure if the checks fail. Moreover, the processing and bandwidth overhead associated with connection establishment procedures makes the added burden of transporting and processing access control information less onerous. In contrast, additional processing and bandwidth for access control applied to individual packets is much more likely to result in an unacceptable overhead if comparable levels of assurance and granularity of enforcement are sought.

The NRI is expected to provide (lower layer 3) connectionless service as its basic interface. Many proposed designs for IP or CLNP switches for this network environment introduce a notion of "soft-state" for connectionless traffic which is roughly analogous to treating this traffic as though it were connection-oriented. This soft state is usually cited as a prerequisite for providing better congestion control facilities in the Internet and for supporting more sophisticated routing, e.g., type of

service (TOS) routing with support for bandwidth guarantees.

We anticipate that designated IP/CLNP switches in the NRI will act as enforcement mechanisms for the transmission and switching access control policy, an assumption that matches Clark's policy routing model. The switches, designated "policy gateways" in Clark's paper, are ideal candidates for this role as they provide the interfaces between domains and thus have direct control over packet transport at domain boundaries. Based on these observations, it seems reasonable to pursue access control mechanisms which assume that some form of connection abstraction can be imposed on most (though perhaps not all) communications. The intent is that the soft-state database could be augmented to include additional data required for access control enforcement.

Throughout this report we shall employ the term "connection" in this broad sense when discussing path establishment procedures, even if the internet and transport layer protocols employed by the end points do not provide a true connection service. Only when the characteristics of a communication activity cannot be effectively modelled as a connection in this soft state sense (as would be the case in many brief, transaction-oriented communication scenarios) will we use the term "connectionless" to describe the activity.

This orientation is further motivated by the relative ease with which one can devise mechanisms for communication scenarios in which there is a well defined "initiator" of a "connection" and this initiator can be called upon to supply inputs to the access control process. For example, traditional virtual terminal communication involves establishing an actual connection, in real time, between two processes. The initiator of the connection is required to supply authorization data to the target of the connection before access is granted to the computation resources at the target (though this occurs after the connection itself is established). The same holds true for traditional file transfer scenarios, even though 3-way file transfer facilities have been defined which may not precisely fit this model.

4.3.2. Variations on Connection-Oriented Scenarios

When the scenario does not embody the concept of an initiator, then it may become more difficult to devise simple mechanisms for acquiring the authorization data prior to authorizing transmission of data on the connection in question. The example of simultaneous connection initiation by two TCP instances was cited as an example of this sort of deviation from our simple connection establishment scenario. The concern here is not an access control issue per se, but rather that two simplex connections would be separately routed instead of one duplex connection, a situation which could lead to anomalous behavior (in terms of performance). Note also that ISO transport protocols (TP0-4) do not support such simultaneous connection initiation and so the criticality of supporting such "dual initiator" situations is not clear.

Another concern was voiced over situations in which the initiator of a connection is readily identified but permission to traverse a path is a function of the authorization

of the computing resources being accessed, not of the subscriber initiating the connection. The assumption underlying this concern is that the initiator of the connection would not be capable of supplying the necessary, validated authorization data to the satisfaction of the policy gateways because such inputs would be available only at the destination. However, if the host being accessed could distribute appropriate credentials to the user prior to his access, the simple initiator scenario might suffice.

These two examples indicate how discussion of access control in the context of specific communication scenarios can be highly dependent on underlying assumptions about details of enforcement mechanisms. Many such discussions cannot take place without a straw man architecture for such mechanisms, and the straw man must address assurance issues, etc. Nonetheless, it is worthwhile to characterize the range of communication scenarios which need be supported in order to establish a reference for evaluating such straw men. Thus we will continue exploring communication scenarios and postpone enforcement mechanism discussion until the next section.

4.3.3. Electronic Messaging

Electronic mail poses something of a problem for connection-oriented access control models for several reasons. First, the initiator of a connection established for mail transfer is generally not the message originator and may not even have any relationship to the originator or a recipient. In fact, staged delivery of mail permits relay points which have no affiliation with the message originator or any recipient. This decoupling raises concerns with respect to assurance of access control inputs. Second, identifying a single subject for access control purposes becomes difficult in this context as multiple message originators may be served by a single mail transfer connection. Third, if traffic destinations are included in an access control decision, the multi-recipient characteristic of many messages further complicates the process.

We could accommodate mail transfer by treating mail transfer agents (MTAs) as subjects, and according to them a set of privileges appropriate to ensure mail delivery throughout the NRI, though that may not translate into allowing every MTA to access every other MTA directly or via any possible network path. This approach sacrifices fine granularity access control, and possibly efficiency of mail transfer, for simplicity. The fact that mail generally does not require the low delay paths[8] (which we anticipate will be the most scarce resources) may make this approach more palatable. If commercial paths are employed and fine grained billing is required, this approach delegates responsibility for per-user billing to the message handling system (as envisioned in X.400 recommendations). This approach is analogous to the access control technique typically adopted for end-system access control with regard to mail.

[8] If electronic mail offered priority service categories which imposed stringent limits on delivery delays, then this general comment might not hold.

4.3.4. Transaction-Oriented Communication

Various brief, connectionless interactions will take place between servers. Interactions are so brief, and may be so dispersed over time that they do not fit the connection abstraction noted above. Nonetheless, some form of access control must be allied to all traffic if the access control facilities are to be effective (complete mediation). Such interactions may best be accommodated by not requiring any connection-like authorization procedure, but rather by requiring the access control enforcement points to recognize such interactions (perhaps based on source/destination addresses) and permit them on the basis of fairly static authorizations. This "special case" treatment for connectionless traffic is likely to be acceptable only if the resulting traffic volume is fairly low. Some form of auditing of these traffic flows would still be necessary[9] to support the accounting requirements cited in section 1 and would provide a basis for detecting anomalous patterns that might be indicative of misuse.

File server interactions may not fit this profile, despite the fact that they are transaction-orientated communications. If the quantity of data returned in response to a small query is quite large, e.g., an entire file or directory, then the traffic volume would likely be too large to treat as above. Fortunately, most file server interactions would likely be local and thus not subject to the access controls we are discussing, i.e., the transfers would not cross domain boundaries. However, a homogeneous collection of file servers in different geographic locations might generate significant amounts of traffic in response to user commands. This poses the potential problem of large data transfers initiated from hosts which employ connectionless protocols and which operate on behalf of (non-resident) users. The first aspect of this problem could be addressed by requiring use of connection-oriented protocols for such transfers (a not unreasonable suggestion for other than local transfers anyway). The second aspect of the problem either requires enforcement mechanisms which support such "proxy" operations or adoption of policies which do not require fine grained access control (so that identification of the file server rather than the specific user is sufficient).

4.3.5. Multicast Communication

One other class of communication was very briefly discussed which was also not well represented by our simple connection-oriented model, i.e., multicast communication. At least some of the concerns about support for multicast seem to have arisen in conjunction with discussion of the need to factor in the authorization associated with the destination of a packet as well as its source. Again, the underlying assumption seems to be that the destination might be required to provide some authorization information data which only it would possess and acquiring this data would become even more complex in scenarios where the packet is addressed to multiple destinations.

---

[9] If the volume is sufficiently low, the traffic might be considered part of the "noise floor" for the NRI and not explicitly accounted for, as would be the case for routing updates, etc.

One can distinguish two classes of multicast communication: transaction-oriented and stream-oriented. The latter has been typical of conferencing communication while the former is typical of server location queries, etc. Transaction-oriented multicast communication might be accommodated by the static, address-based access control mechanisms discussed in section 4.3.4. Stream-oriented multicast typically involves some form of stream establishment procedure prior to transmission of user data and it does not seem unreasonable to augment such procedures to accommodate authorization data transfer. Thus multicast communication may not be so difficult to accommodate as originally suggested.

4.4. Access Control Architectures

Access control policies can be examined independent of enforcement mechanisms and architectural details, but there are limitations to such isolated examination, as noted in section 4.3. There are several reasons for adopting a (straw man) architecture in which to consider such policies. First, one must identify the transmission costs, e.g., in terms of processing overhead or bandwidth reduction, associated with enforcement mechanisms in support of policies. Second, one must understand how policies' representations and authorization data are managed in order to estimate the infrastructure costs (additional servers and databases, dissemination of authorization data, human management for the databases and equipment, etc.) associated with such policies. Third, one must understand where trust is vested in the architecture in order to gage its social acceptability and establish the level of assurance that might be accorded the resulting access control system.

In this section, we discuss how operating system security principles might be applied in this access control context.

4.4.1. Analogies with Operating System Security

In discussing mechanisms for network resource access control, it is useful to compare them to some of the enforcement precepts generally applied to operating system access control mechanisms. In the context of computer systems (subscriber resources), the concept of a ''reference monitor'' is widely used. A reference monitor mediates all accesses by subjects to objects. (For any reasonable degree of implementation assurance the reference monitor must itself be protected from tampering so that it cannot be circumvented.) Before any object is accessed, the authorization of the subject to access the object, and to operate on it in the fashion requested, is checked. This a priori checking is deemed essential if the reference monitor is to prevent the unauthorized release or modification of data. Despite the use of reference monitors, even in relatively high assurance operating system implementations, there are usually covert channels via which data can be released to

unauthorized subjects at relatively low data rates.[10] Complete elimination of these covert channels is usually deemed impractical except in the most sensitive applications. Auditing of object accesses is often performed in addition to the access control enforcement described above and post access analysis may be carried out. However, this analysis is best viewed as a damage control measure and a possible means of detecting anomalous usage patters, not a primary enforcement mechanism.

In the context of network resource access control, neither disclosure nor modification of subscriber data is at risk. (Recall that traffic analysis is not a service considered here, but rather is a subscriber security service considered by the End-to-End Working Group). Instead, the primary concern is transmission of packets via paths which are not unauthorized, i.e., unauthorized consumption of resources. A major failure of these controls could result in denial of service for authorized users, but minor failures result only in some small amount of "theft of service". The impression provided by the report of the Policy Working Group is that such minor violations would be acceptable in the context of most, though not all, of the articulated access control policies for switching and transmission resources.[11]

This suggests that it is appropriate to adopt enforcement mechanisms which are resistant to attacks which would result in major violations of the access control policies, but that perfect control of traffic flows is not essential (analogous to information disclosure via covert channels in the operating system context). It also suggests that post access auditing is appropriate as a damage control measure and to verify that authorized subjects have not engaged in usage patterns which call into question their trustworthiness. Thus we suggest adopting a reference monitor-like approach for our access control policies, but with the understanding that perfect access mediation is probably infeasible and unnecessary.

4.4.2. Clark's Policy Routing Model and Access Control

We adopted as a strawman architecture the design presented by Dave Clark in his paper on policy routing.[12] Many of our discussions were influenced by the concepts and mechanisms proposed in the paper. In this section, we review those aspects of the design which are relevant to our access control concerns, discuss areas which were not completely specified in Clark's paper, and explore some modifications and extensions to this design.

Clark's paper defines three new entities in the Internet which participate in policy routing and thus network resource access control. Enforcement of policy route constraints is the responsibility of policy gateways. These gateways are present at the

[10] Data rates on the order of 1-10 bits per second are typical for covert channels in this context.

[11] It is clear that some access control policies would not be satisfied by inherent limitations of the type suggested here and thus would not be accommodated by the architectures proposed herein. For example, NASA is unlikely to trust such architectures to enforce a non-interference policy for network resources critical to shuttle operations during a mission.

interfaces between domains[13] and thus are capable of controlling the flow of all traffic into or out of a domain. Within each domain are one or more policy servers.[14] These devices serve several functions and are, in many respects, the heart of the access control system proposed by Clark. A policy server serves as the repository for and the management interface to inter-domain access control policies for its domain. Thus it provides representations of these policies to policy servers in other domains and it acquires from them policies applicable to their domains. A policy server responds to queries from subjects on hosts within its domain, synthesizing valid routes based on the subject's communication requirements, the PS's knowledge of current internet connectivity, and of applicable inter-domain access control policies. A policy server provides the selected policy route(s) to the subject, along with authorization and billing data, cryptographically sealed by the policy server. This operation is best viewed as a digital signature process.

A central feature of this proposal is that it requires the policy gateways to trust the policy servers that represent a domain, but does not require this trust to be extended to each subject within the domain. Clark assumes that domains are mutually trustworthy to the extent that the policy gateways rely on the source policy server to have correctly evaluated the subject's authorization to make use of a given policy route. Since domains in the NRI represent organizations (e.g., Federal agencies), there may be a reasonable basis for assuming that the individuals managing a policy server on behalf of a domain can be relied upon to operate in a responsible manner. (The trustworthiness of the hardware and software upon which a policy server is implemented is a separate concern.) Note that the means by which a policy server ensures that a validated route is properly bound to an authorized subject within the domain is a local matter, not specified by the architecture.

Signing of this collection of data serves several purposes. As noted above, the policy server for a domain is vouching for any identification and billing data and is also stating that it has selected a route which is allowed by the access control policies provided by other domains. Clark notes that this does not preclude checking of route validity by policy gateways, but it does allow mutually trusting domains to rely on these checks performed by the originating domain's policy server. It is advantageous that the signature be generated using asymmetric cryptography so that the policy gateways have a non-repudiable record of these claims by a policy server (which might prove useful should disputes arise or in isolating faults). Since only policy servers generate the signatures, the task of managing keys for signature validation becomes manageable.

---

[12] "Policy Routing in Internet Protocols," Version 1.1, May 19, 1988.

[13] Clark employed the term "Administrative Region" but we adopted the term "Administrative Domain" to avoid any implications of geographic locality.

[14] Clark designated these devices "Policy Controllers" but we have adopted our current designation to avoid confusion that might from use of the acronym "PC."

Clark proposed that an initial packet include an IP option consisting of signed policy route data (including billing and authorization information), but that subsequent packets contain only a short form of the policy route option with a "handle" from the option in the original packet. The handle would be generated by the policy server in the source domain and would uniquely identify the current route (based on the combination of the domain identifier and the route identifier). The policy gateways would cache the policy route using the handle as a search key and subsequent packets would be validated by determining if the handle was present in the cache and by processing the packets according to the policy route associated with the cache entry.

This approach to individual packet validation differs from others which have been proposed, e.g., Estrin's VISA schemes,[15] in that is does not assume a crypto checksum binding authorization data to packet contents. Thus it is possible to copy a valid header from a legitimate packet and prepend it to a packet content not associated with the valid header. Clark argues that this is an acceptable vulnerability since the access control afforded here only applies to transmission and switching resource utilization, not information disclosure. The utility of "appropriating" valid packet headers is limited so long as the policy gateways match source and destination addresses against those held in the cache (as specified in the signed, policy route option). However, in circumstances where use of resources results in actual bills, unauthorized transmission of packets using copied, valid headers or forgery of valid headers could result in spurious charges to legitimate users.

In his paper, Clark proposes inclusion of a 16-bit signature and a handle composed of a 16-bit domain identifier, and a 16-bit route identified unique within the domain in the policy route option. It was not clear if the short form of this option would also contain a signature, though most of the working group membership believed this might have been implied. We observe that a 16-bit signature is probably insufficient to preclude forgery; a more appropriate size quantity would be on the order of 128 or 256 bits. It is critical that the policy route option be unforgeable and thus the extra overhead implied by the larger signature is justified.

On individual packets traversing an established route there is a diminished need for short form option integrity and authenticity, except to prevent malicious, spurious charges. As noted above, if policy gateways check the source and destination address in the packet against that recorded in the cache, there is relatively little to be gained from forging a short form option. Since it is already possible to copy a legitimate short form option from a valid packet, it isn't clear how much additional assurance is provided by incorporating authenticity measures in short form options.[16] Perhaps a prudent safeguard is for policy servers to adopt a process for selecting route identifiers

[15] "VISA Scheme for Inter-Organization Network Security," D. Estrin and G. Tsudik, Proceedings of the 1987 IEEE Symposium on Security and Privacy.

[16] We also note that the computational overhead of validating a crypto-seal (or reasonable size) on every packet is probably prohibitive.

so as to minimize the likelihood that they can be guessed, e.g., using a pseudorandom process. We do recommend that the policy route option be expanded to include some indication of lifetime, either measured in time or in number of packets or both. This limit on the lifetime of a route further reduces its vulnerability to exploitation by unauthorized subjects and a packet quota could provide an additional means for detecting misuse.[17]

4.4.3. Clark's Architecture in Retrospect

Now that we have reviewed the architecture presented in Clark's paper and made some local observations and suggestions, it is useful to view the architecture in the context of our previous discussions. For example, the architecture described in this paper supports both identity-based and rule based, administratively-directed access control policies. It adopts a security model in which the objects are routes through the Internet (which correspond to use of switching and transmission resources) and the subjects are processes executing on behalf of users or groups of users and, hosts or groups of hosts (perhaps entire domains).

Clark's architecture embodies the connection-oriented (single originator) access control model discussed in section 4.3.1 above and thus this class of communication is especially well served by this architecture. Communication scenarios that deviate from this model must be examined to determine how they can be accommodated. For example, electronic messaging would probably be handled by viewing the MTAs as subjects rather than trying to control access on the basis of individual message originators, as suggested in section 4.3.3. Stream-oriented multicast communication could be accommodated as described in section 4.3.5.

Transaction-oriented communication, whether point-to-point or multicast, may not be served very well by this architecture, i.e., it may be difficult to amortize the cost of policy route options in these communication scenarios. However, if cache entries in policy gateways can include "wild card" entries for addresses, then it might be possible for a policy server to seed routes for access to commonly accessed collections of servers, etc. on behalf of all (many?) of the hosts in its domain and pass out the identifiers for these routes to members of the domain.

The remaining deviant case involves dual-initiator connections, a scenario of undetermined criticality. The source and destination hosts could discover that different route identifiers were assigned to a single transport layer connection and co-operate to use only one of the routes (using some unambiguous criteria such as comparing route identifiers as unsigned integers and selecting the larger value route identifier). However, this solution may be viewed as being outside of the architecture in that it does not involve the policy gateways, policy servers, etc. Another aspect of support for some communication scenarios which generated some concern is also outside the

[17] If a packet quota were imposed on a route and the route were used by an unauthorized subject, the authorized subject might detect this if the route were to become invalid due to exhaustion of the packet quota.

scope of the architecture, i.e., the need for proxy authorization. The possible need for such a facility was noted in conjunction with file server communication on behalf of users, e.g., transfer of a file between two file servers. It appears that the architecture in Clark's paper could support such communication authorization, but the means by which the initiating policy server determines that the communication is on behalf of a specified user, rather than the file server itself, is a local matter not part of the architecture.

In section 4.3.2, a concern was raised about supporting route establishment when permission for a route was dependent on authorization of the destination, not the initiator. In Clark's architecture, this case would not be treated any differently since it is the initiator's policy server which evaluates the access control policy and makes the decision, and all the inputs required to make the decision are available to that policy server. For the most part, the architecture assumes the policy gateways trust the initiating policy server to interpret the access control policies correctly at the time it generates the sealed route option and supplies it to a subject in the local domain. Intermediate policy gateways can review the data provided in the policy route to confirm the decision, but the paper seems to suggest that this independent confirmation would not usually be carried out during route establishment, for reasons of efficiency, though the signature should be checked.

4.4.4. Trust Implications and Possible Remedies

In Clark's architecture, the ability of policy gateways to validate an access control decision is limited because the authorization data included in the signed route option does not incorporate any independent validation mechanisms. For example, the policy gateways must trust the initiating policy server to have verified the user ID, agency affiliation, etc., because there is no means for the policy gateways to verify these access control inputs directly. The route verification that can be performed by policy gateways is based on checking the signature (thus verifying the integrity and authenticity of the route) and on matching the supplied access control inputs against the policy in effect. Rather, the assumption is that access control policy terms and conditions are distributed and that the data items against which the policy terms and conditions can be matched are all locally validated quantities, i.e., they are vouched for solely by the initiating domain through its policy server. Thus the architecture relies on mutual trust among domains, non-repudiable (signed) policy routes, and post-hoc auditing to reconcile conformance.

If this level of mutual trust proves unacceptable in the NRI, it is worth exploring how one might extend the architecture to incorporate independently verifiable "credentials". First we need to identify which credentials might need to be independently verifiable. One candidate is the AGENCY AFFILIATION INDICATOR. If a connection is initiated with a policy route that claims an affiliation for which the initiating domain is not the certifying domain, then it might be reasonable to require that the AGENCY AFFILIATION INDICATOR be

independently verifiable.

A BILLING CODE might require independent verification if the code is one which does not somehow imply charges to the initiating domain.[18] An analogy can be made with long distance telephone charging. A direct dialed call from a home number is assumed to be legitimate, whereas a similar call from a pay phone or hotel room requires an independently verifiable account number unless the charges are borne locally (via coins or billed to your room). Thus BILLING CODEs also appear to be good candidates for independent verification, at least in some circumstances.

Finally, the other major credential considered for inclusion in policy routes was the SUBJECT ID. Again, the circumstances in which independent verification is likely to be of interest are those in which the subject's domain differs from the initiating domain. Since the SUBJECT ID already includes an indication of the domain which vouches for the subject's identity, it is easy to determine if independent verification is required. Thus in all cases the motivation for an independent verification facility arises only when the certifying domain for a credential differs from the initiating domain for the connection.

In order for a domain to certify a credential for independent verification, the resulting data should be bound to a subject (or class of subjects) so as to render it useless to other subjects. This is easily accomplished by including the subjects (subject class) to whom the credential is issued as part of the signed credential. Note that this also allows the issuer to distribute the credentials directly to subjects, not only through domains, if that proves useful. Thus a domain such as DOE might issue a BILLING CODE and AGENCY AFFILIATION ID to a researcher at a university, binding it to his SUBJECT ID. The researcher could present the credentials to his local policy server for consideration in selecting routes and that policy server could include the credential along with the policy route option.

Policy gateways could verify that DOE had granted permission to use the BILLING CODE to this subject and that the subject was affiliated with DoE by verifying the seal on the credential and matching the included SUBJECT ID against that in the policy route. As above, it might not be feasible for every policy gateway to perform this independent verification prior to processing packets for the connection, but the option would exist and post hoc auditing is feasible. These credentials should contain a validity date range to constrain their lifetime, and some form of hot list would also need to be maintained by each issuing domain and distributed to policy servers and gateways to revoke credentials, e.g., upon termination of affiliation.

This technique would reduce the level of trust accorded the policy server at the university since it could not forge the credential. This binding does not ensure that the subject and the source address are correctly paired. However, if the SUBJECT ID

[18]Clark suggested that such codes might incorporate an AD identifier which would explicitly establish the requisite binding. However, he was concerned that a strict requirement for a billing code to be bound to the initiating AD would unduly restrict mobile users.

indicates that the initiating domain is the certifying domain for the subject, then one must ultimately rely on that domain to correctly maintain subject-address bindings. If the subject is foreign to the initiating domain (as might be the case for a mobile user), the incremental assurance offered by independently verifiable credentials seems fairly small. It is not clear what form of credential binding would be useful for mobile users. The ''home domain'' for a mobile user could certify that he was temporarily associated with another (specified) domain, thus lending credence to a claim by the initiating domain that the ''foreign'' user was in residence. If the logistics of generating and transferring some sort of travel credential (''hall pass''?) could be made acceptable to users, this might prove to be a viable means of addressing this problem. For these credentials, even more than most, validity dates should be included to limit their lifetime.

5. Resource Sharing

Working Group 2 Members

David Clark (Chair)	MIT
Guy Almes	Rice
Bob Braden	USC-ISI
Scott Brim	Cornell
Jon Crowcroft	University College London
Deborah Estrin	USC
Steve Goldstein	Mitre
Phill Gross	NRI
Bill Jones	NASA/Ames
Dan Nessett	NMFECC
Ari Ollikainen	RIACS
Mike St. Johns	DCA
Tony Villasenor	NASA HQ

5.1. Introduction

This working group was asked to consider the question of mechanism necessary to insure "fair" sharing of resources, in particular bandwidth.

The group proposed, as a starting position, that to permit sharing of resources, such as networks or links, among agencies (for example), the following questions must be answered.

- What sorts of service classes will be required? Which are possible?
- How must the users of the resources be categorized?
- What sort of accounting for the resources are required?
- What levels of assurance are required?
- How global is the impact of various sorts of service classes?
- What management tools are required to control multi-agency policy mechanisms?

Two ideas are central to the discussion: service class and category.

5.2. Service Class

The idea of service class is that in order to provide a controlled sharing of a resource, it is necessary to define how the sharing will be measured. The measurement represents a way of specifying a service class.

In the workshop, most service classes related to policy concerns were defined in terms of relative bandwidth. The following examples were often proposed:

- A link is shared by two (or more) service classes, each of which gets a guaranteed fraction of the link capacity under overload.
- A link is shared by two (or more) service classes, some of which may not interfere with others. That is, they are excluded from the resource if demand is excessive.

An example of a service policy requirement not directly related to bandwidth is mutual aid: two agencies that agree to carry the other's traffic if the resources of the one is down. Half of the mechanism necessary to support this is easy: one could define a service class for traffic belonging to the other agency, and define the service constraint for that class. The hard part of the mechanism is to define how the switch is to know that the other resource is down, so that the usage by that class should be permitted.

In the discussion of service classes, the following comments arose:

- Outside the arena of policy control, there are much broader requirements for service classes, in order to support new sorts of applications. For example, some applications require control of delay. This broader problem is usually called the "Type of Service" or TOS problem (also called quality of service or QOS in ISO). In this respect, the mechanism required of the switch for specifying and measuring the services classes is just a subset of that required for support of multiple classes of service to support applications.
- Some (non-policy) examples of service classes are very difficult to support, e.g., those for real-time speech, or variable rate encoders (that can adjust to changing bandwidth allocation, but must KNOW what rate they are being offered).
- We believe it is not difficult to provide commitment of resources to simple service classes. For example, a gateway could be constructed that would take packets in two service classes, and ensure that under overload each class received equal access to a link. The problems in doing this are to control the overhead in the gateway, which would have an impact on high-speed networks, and to understand the global impact of such guarantees (see below).
- The definition of service classes must be understood globally.

5.3. User Categories

In order to ensure that some user receives some service, it is necessary to identify the packets associated with that user. This is a very hard problem, perhaps harder than supporting reasonable service classes.

Current IP packets do not have user names in them, just source and destination Internet addresses. But a single machine might support users with different privileges, or a user wanting to use different privileges at different times.

In the discussion of user categories, the following points came up:

- To support the sorts of requirements that were offered as examples (e.g., put all NASA packets in service class X), it will be necessary to have some explicit tag in the packet to indicate the packet category. This is a new IP level mechanism.
- The level of "user granularity" is not clear. Would one tag for all of NASA be sufficient, for example?
- It might be necessary for a packet to carry more than one tag, to permit a user with multiple privileges to use them at the same time. Perhaps tags could be approximate, and could resolve in different manners in different parts of the net.
- The level of trust needed for the tag is unclear.
- If a tag is abused, the use must be traced back to an accountable entity, which ought to be a human.
- A very hard problem is multicast: one packet going down several paths that might require different user privileges.

5.4. Additional Discussion

The following comments were made about the other points in the list above.

5.4.1. Accounting for usage:

A clear requirement was that the usage of resources by different user categories be accounted. However, the details of the requirement were not clear. It does not seem too hard to provide a simple measure of total bytes or packets used by each class. As noted above, the hard part is defining the classes, and inserting the class information into the packet.

If a more dynamic accounting for usage is required, then a mechanism can probably be defined to account for usage by any pre-defined measure, but arbitrary measures will be real hard.

5.4.2. Levels of assurance:

There seem to be two obvious levels of assurance as to enforcement of service classes and user categories.

- Separation of traffic into classes, and enforcing and accounting for the usage of each class, will be performed properly so long as the switch elements belonging to each agency operate properly.
- Proper separation and accounting must occur even if the switches of one agency are mis-programmed or malicious.

The latter would be required (probably) in a network operating in hostile circumstances; it corresponds to mechanisms to prevent denial of service. It is a level of assurance that is hard to achieve.

The former level of assurance is much easier. It corresponds roughly to the operation of the Internet today. If one set of gateways is not operating properly, there

may be bad global effects that the other gateways cannot prevent. The problem is cured, not by robust dynamic algorithms, but by detection and correction (e.g., by humans) of the problem.

For many circumstances, e.g., conformance to OMB regulations, the weaker form of assurance is probably sufficient. But DARPA, for example, expressed an interest in as robust an assurance as possible.

5.4.3. Global effects:

The problem of global effects of policy is a very serious issue, the impact of which does not appear to be sufficiently appreciated.

Certain resource constraints, most obviously non-interference (a service class that is excluded when a resource is overloaded), cannot be implemented except in the context of a global routing algorithm that knows about the constraint.

The problem is the following. At the moment, the Internet supports the idea that for any destination address, there is one route out of a switch. If we now support two service classes going to that destination, then each will be sent by the same route, given the current routing algorithm. If one of these service classes is now blocked from a congested resource, there is no mechanism to reroute that class to another resource. The result is that the service class is totally disabled.

In other words, today if a gateway makes a local decision to discriminate against certain users, those users perceive a global disruption of their service.

The problem of propagating and responding to local controls is not impossible. While this section stresses the need to understand the problem, we believe that solutions exist. It will be necessary, however, to contemplate a major adjustment to the current philosophy of Internet routing. In particular, most of the promising approaches are based on some form of source routing.

Above it was asserted that it was not difficult to build a gateway that would make simple resource guarantees. The difficulty is propagating the knowledge of that local guarantee. There are some guarantees that could be enforced in today's Internet without the necessity of global knowledge. For example, if a gateway provided equal sharing of a link under overload to each of two classes, then the global impact would be that of a link whose capacity changed by 50%. A fluctuation of this magnitude could not be globally distinguished from other current forms of congestion. So there are some local controls that can be applied safely in today's Internet, and others (such as non-interference) that can only be contemplated in the context of a global architecture.

5.5. Conclusions

The problem of making a local modification to a gateway to enforce a bandwidth usage limit to a identified category of users seemed reasonable.

Associating a user category with a packet is very hard. The actual requirements are not clear (are one or several categories required, what is the level of assurance that the specified category is legitimate, and so on). In addition, the mechanism is not obvious. This matter is addressed in the report of working group 1.

The problem of level of assurance is also very hard, again because the actual requirement is not clear.

Accounting for usage is probably not too hard.

The hardest problem is redefining the routing algorithms of the Internet to correctly propagate and respond to the impact of local policy controls.

There are several hard and interesting research questions:

- How do service guarantees compose?
- Is it possible to build multi-region systems that are resistant to attack by malicious third-party regions?
- How could user categories be managed? Are they multi-valued, hierarchical or flat?
- How can fault isolation and service assurance be performed?
- What is the relation between statistical resource allocation and possible guarantees of access?

To avoid solving too general a problem, several questions should be asked of the agencies.

- What level of assurance is required?
- What sort of user categories will be required?

5.6. Recommendations

The group proposed a number of experiments and changes that could be undertaken at once, to better understand the problems of policy routing and resource control, and to provide operational facilities toward these goals.

These goals are organized in three categories, things that could be done at once using existing tools, projects with a short time frame, to provide better capabilities and understanding quickly, and finally, projects that would require longer to complete.

5.6.1. Instant projects

Statspy

Although source and destination addresses are not a precise indicator of service class, they do provide much useful information. The so-called *statspy* tool has been used in the past to collect a matrix of traffic sorted by source/destination address. This information could be collected for shared links today to provide a first cut at accounting for the resource.

Route filtering

Route filtering provides a way to instruct a gateway to believe only part of an incoming routing packet, or to change parts of that incoming data, e.g., the cost metric of a proposed path. This capability, available in most commercial gateways and in the gated software for Unix, provides a way to control which destinations are reached by which paths. It cannot separate service classes, but can be used for very rough divisions of traffic based on destination address.

5.6.2. Short-term experiments

These are experiments that could be undertaken at once, with the expectation that they would yield results in the short term. They are not thought to contain high-risk research questions. They might provide some increase in operational capabilities in one to two years.

Simple resource guarantee

A gateway could be programmed to sort incoming packets into two service classes (based on some simple if unrealistic characteristic of the packet, such as addresses or TOS flags), and then divide the use of a link fairly between these classes. That is, in underloaded conditions, each could operate without constraint, but in overload each class would have a fair share of the link.

This would be a first demonstration of allocation of resources to service classes, and would provide a practical way to share a link.

Observe tagged packets

Above, it was noted that the *statspy* program could be used to count packets based on source and destination addresses. One could define a simple IP option, which carried a user identification, and then use the same *statspy* to count these packets. A simple use of this option would be to tag the packet with an indicator of which agency had "sponsored" the packet.

Putting a new IP option into a packet is not hard; some systems like Unix 4.3 BSD provide the hooks to do this today. A simple and general way to find the proper value of the option field would be to implement a very simple form of "Policy Server", which could be a user process on a Unix system. One would send a packet

to the server with the source and destination addresses, the name of the sponsoring agency, and other credentials. In return, one would get the suitable IP option, which would just be inserted into the packet.

This would provide a more accurate accounting of shared resources, and a first demonstration of the concept of the policy server.

Fast encryption of the policy information

In order to ensure that policy routes, authentications and so on are not forged, it will be necessary to seal them in some way. The obvious technology is encryption. A demonstration is needed of a sealing technique that runs at tolerable speeds. This would permit the introduction of a high level of trust into the accounting.

Demonstration of "soft state" in gateway

Several propositions for management of resources in gateways require that the gateway remember some aspect of the packet sequences passing through it. The idea of "soft state" has been proposed to capture the idea of cached information in the gateway which can be reconstituted if lost without terminating the higher level connection.

A first project is to program a gateway to show that this sort of state can be managed effectively, with acceptable overhead. The information stored in the state could initially be rather simple, for example the resource guarantees mentioned above, or logging of packet tags, or enforcement of source/destination access control.

Demonstration of policy routing with Loose Source Route

Once we have demonstrated the tagging of packets, we have all the pieces of a first demonstration of policy routing. A Policy Server module can be programmed to take the source/destination addresses, sponsor and so on, and receive in return a Loose Source Route IP option. This could be placed in the outgoing packet to achieve controlled routing of the packet.

5.6.3. Longer-term experiments

The following are experiments that have a longer term focus. They deal with harder problems, will take longer, and yield an increased functionality. They represent steps that can be undertaken now, and should be if increased functionality is to be achieved in the next few years.

Define and support Policy Source Route option

Above we described a simple demonstration based on the IP Loose Source Route. While this represents a useful first demonstration, the LSR is not suited for real policy routing, because it binds the route to specific gateways, which is too concrete, and because it has no fields to carry policy information.

What is needed is a new IP option to define a Policy Source Route, a more abstract form of source route containing policy information. There is general agreement on the need for this class of mechanism and the general form it would take. A detailed design is now needed.

Tools for Synthesis of PSR

The Policy source route described above would be generated using information exchanged by the various Policy Servers and Policy Gateways. Algorithms for this have been proposed; a concrete design should now be undertaken.

Define protocols for control interaction

To provide the information for the routing algorithm, it will be necessary for policy gateways, policy servers and hosts to exchange information. Protocols for these exchanges must be designed.

Management Tools for Policy Controls

Current experience teaches us that we must develop suitable management tools for a mechanism at the time that we develop the mechanism itself. The problems of policy control are complex, and can be expected to lead to complex management problems. We must begin the design of a management architecture for policy mechanisms.

Analysis of composability of local policies

We assume that an administrator of a region will express policies reflecting the local concerns of that region. These various local policies must be composed to provide an end to end service. It is necessary to ensure that the various local policies do indeed combine to permit a reasonable global service. It would be nice to have some formal understanding of what sorts of local policies can be composed, and some tools for checking that the actual proposed local policies are reasonable.

Architecture for signatures and sealing

To ensure the needed level of assurance, an overall strategy must be devised to define the trust that holds between the different components of the system, and the mechanism needed to insure the integrity of Policy Routes and related messages.

6. End-to-End Security Services

Working Group 3 Members

Dennis Branstad (Chair)	NIST
Matt Bishop	Dartmouth
Brian Boesch	DARPA
Anita Holmgren	Unisys
Barry Howard	Livermore
James Morrill	Sparta
Dan Nessett	NMFECC
David Peters	NASA
Steve Wolff	NSF

6.1. Introduction

This section deals with end-to-end security services for the National Research Internet (NRI). As described previously, the NRI consists of multiple, autonomous, mutually-suspicious, administrative domains. The NRI is an open environment with a dynamic security perimeter. Each domain may have its own security policy and offers a unique set of security services to its own community. However, if secure interoperation is desired across domains, these security policies must belong to a set of hierarchical, consistent policies, and certain cross-domain agreements with respect to security are needed. Working Group 3 focused on the nature and content of such inter-domain cross-agreements.

A security architecture for the federally-funded research networks (which make up the NRI) was proposed. The architecture consists of security sevices, where they are needed, example mechanisms, and the implied common technologies and common policies necessary to support interoperation.

First we offer the strawman architecture. Next, we introduce the concept of a "security domain"; we discuss multi-administrative higher-level security services in detail; then, using the workshop model (of phase 0-3 technologies), suggest a phased approach to making the architecture a reality.

6.2. Multi-administrative Security Architecture

We define security to include, not only protection from unwanted disclosure, but also, protection from unwanted modification and prevention of denial-of-service. This working group suggests that a small number of security services are necessary, and that these security services need to be repeated at various layers in the protocol and system architecture. The following chart illustrates some candidate security services such as: confidentiality, integrity, authentication, access control and service assurance; suggests placement in the architecture such as: user-level, host-level, gateway; and suggests common technologies and common policies that are needed to support these security services across domains.

Security Services in a Multi-Administrative Domain Environment			
Security Services	*Example Mechanisms*	*Common Technologies*	*Common Policies*
Origin Authentication			
-user/process	secure-ID card	Key Distribution	global ID
-host	certificates	(common protocols	conventions
-gateway	certificates	and standards)	
-realtime/deferred	challenge/response	Directory Services	
- certificates	(object registration)		
Origin Access Control			
-user	login	can we use policy	global ID
-host	visa	servers?	conventions
-gateway	policy routing		
Object Integrity			
-msg	MACs		
-file	MACs	common format for	global ID
-datagram	MACs	integrity labels	conventions
-connection	MACs		
-field	MACs		
Object Confidentiality	protected wire	Encryption- (common protocols and standards)	Key Distribution agreement
Service Assurance	routing	Byzantine Robust Management	Multi-domain Network agreement

The International Organization of Standards has recently adopted an International Standard Security Architecture (IS 7498/2) that specifies five security services in the Open Systems Interconnection model of computer networks. The five services and a short definition of each are:

- Authentication: verifying the identity of communicating entities (e.g., computer, software programs) in a network;
- Access Control: restricting access to the information and proccessing capabilities of a network to authorized entities;
- Confidentiality: preventing the unauthorized disclosure of information;
- Integrity: detecting the unauthorized modification of information;
- Non-repudiation: preventing the denial of transmitting or receiving certain information.

A security label is security relevant information that is attached to other information to assist in providing the above named security services. The U.S. Department of Defense (DOD) has specified the format of a security label to be used at the Internet Protocol (IP) layer of the DOD suite of protocols. This label is used primarily to state the classification of the information in an IP packet. The security mechanisms then use the label to control the routing of the packet through the network (based on the security of alternate routes) and the confidentiality protection to be provided to the packet.

6.2.1. Security Domains

Security needs to be considered from an end-to-end perspective. Secure interactions across administrative domains, a security perimeter must be defined. A hierarchical set of ''security domains'' could be established for the research internet. A global security domain could then have a security policy and a set of security services that would be enforced and supported throughout the internet. Each sub-security domain could then have additional security services. Security interfaces between security domains would then be defined. Rules for data to cross these interfaces would need to be established and enforced by ''interdomain gateways''.

6.3. Higher-Level End-to-End Services

In this section, we discuss services in terms of ''administrative domains", which are collections of machines and supporting hardware (nets, etc.) controlled by a set of people who have the (recognized or assumed) power to choose what services that set of entities will offer to other entities. We assume that entities in different administrative domains are mutually suspicious but wish to provide some set of services to each other. Note that the managers of each domain will define their own policies towards the provision of services, so the entities must interact in light of the relevant policies. These policies must be consistent; however, this is not a great restriction, since the policies will either be imposed by an authority encompassing both administrative domains or (more likely) by bi- or multi- lateral agreements or adherence to a mutually agreed upon standard.

We describe a set of supportive services designed to provide the basis for other, productive services visible to the users; we also suggest some useful productive services. The distinction between the two is crucial; supportive services, invisible to the user, are essentially a set of library routines designed to provide security and integrity functions in a manner dictated by the administrative domain. Two domains must decree some format for the interchange of information such as user IDs or file checksums, but (for example) the NASA administrative domain may require use of the File Transfer Protocol (FTP) be allowed only to authenticated individual users, whereas the Dartmouth administrative domain may allow any user from an authorized host to access files using FTP. In this case, the supportive services (authentication of the source of the FTP request) for NASA must support per-user authentication, whereas Dartmouth need only support per-host authentication; however, if NASA

allows FTP access by users in the Dartmouth administrative domain, some accommodation must be made by policy (either by NASA, to accept per-host authorization when users from entities at Dartmouth FTP, or by Dartmouth, to enable per-user authentication when dealing with FTP requests to entities in the NASA administrative domain). Productive services simply request of the supportive services whether some condition is met (is the user allowed to use the service, has the file been altered in transit, etc.) and proceed on that basis.

We describe the supportive and productive services separately.

6.3.1. Supportive Services

Supportive services supply the basis for an entity in one administrative domain accessing the services supplied by another entity in another administrative domain. To this end, they provide access control, authentication, integrity, and confidentiality checking.

The first class of supportive services is origin authentication. There are several subclasses. A policy may require per-process (i.e., per-user) authentication, using mechanisms such as SecureID(tm) cards; this will require some common technology for key distribution among the co-operating domains. A policy may require authentication at the host or gateway level, using certificates; here; a set of directory services such as an object registry must be common to co-operating domains. Note that there are really two flavors of authentication here, real-time authentication in which the origin must identify itself immediately (possibly using a challenge/response protocol), and deferred authentication, in which the origin need only identify itself at some time, the identification being preserved using certificates. Finally, regardless of the type of origin authentication done, all administrative domains must have some global object identification convention that all domains respect.

The second class of supportive services provides access control based on origin. For example, access to a user account might depend on the identity of the requester; on 4.2BSD UNIX systems, access is controlled by the .rhosts file in the target account, with each line of that file specifying a user/host pair authorized to access the account. The system assumes authentication has already been done, and controls access strictly based on the user/host names of the requestor. Similarly, if one host needed to access services on another, it might present a VISA or a service-specific certificate entitling it to use that service. A policy might allow or deny access to networks based on the source or destination of a packet (policy routing). In any case, as with the first class, this class of supportive services requires a global object identification convention. The technology which must be shared by administrative domains co-operating to provide these services is not clear; perhaps policy servers would suffice.

The third class of supportive services provides object integrity. A policy might require that the integrity of any (or all) of messages, files, datagrams, fields, etc., be verifiable, possibly using MACs or other integrity checking mechanisms. In this case, administrative domains enforcing this policy must agree on a common format for

integrity labels as well as a common set of mechanisms.

The fourth class of supportive services provides object confidentiality, for example by encrypting files or protecting the network wires. If cryptography is used, some key distribution mechanism must be agreed upon in order that keys for objects in one administrative domain be available to authorized clients in another. The administrative domains must also agree on the encryption algorithms to be used and some common technology for making keys available is necessary.

The fifth class, non-repudiation, will simply ensure that a requestor (or user) of a service cannot deny that that user made the request (use) of the service. Again, the administrative domains must agree on what types of requests are to be subject to this service, and on the mechanism to be used for inter-domain non-repudiations. Further, the granularity of the non-repudiation records must be decided; this impinges on accounting. For example, NASA may bill on a per-project basis, so if a request came from Dartmouth and the non-repudiation mechanism ensured non-repudiation only in that the request came from Dartmouth, the mechanism would be insufficient for NASA's purpose; again, this must be settled by inter-domain multi-lateral agreement or decree from a higher authority.

In terms of the four phases used to characterize the evolution of capability, at phase 0 is process (user) authentication with passwords; at phase 1 is process (user) authentication using other technologies such as challenge/response protocols; at phase 2 are authentication using certificates, integrity checking mechanisms such as MACs, integrity labeling, methods for non-repudiation, and issues of key distribution and management. Phase 3 issues include the use of VISAs for policy routine and certification across peer administrative domains.

6.3.2. Productive Services

Differing administrative domains provide varied services, but most will want to allow entities at other administrative domains to use one or more of the following services on one or more entities in the local domain. This list is by no means exhaustive; we have simply discussed the more common currently-provided productive services. Undoubtedly, equally or more important ones will arise in the future, or inter-domain policies and agreements will require new ones.

Remote job execution will be essential within domains and given the advances in the use of collaborative support services and distributed computations, important in inter-domain support. Currently, mail transfer by far dominates this area, with file transfers coming a close second. Both raise issues of inter-domain use of remote resources such as disk space and CPU time, as well as confidentiality and integrity issues (can only those authorized to read the file/mail do so? can the file/mail be altered?) Further, authentication of the sender/author (was the letter telling me I got my raise a forgery?) and access control will also be essential. Some of these issues are being addressed by Steve Kent's privacy task force (see RFC 1113), which has been examining secure and private electronic mail for some time. Finally, non-

repudiation of mail is important when electronic mail is used to make agreements or convey sensitive information that the sender may wish to deny having sent. Extensions to more sophisticated forms of collaborative support, such as multi-media mail or electronic ''whiteboards", will require the same level of supportive services. (Note that the ''support" service is a production, rather than a ''supportive'' service. This terminology is confusing, to say the least, but it is also standard.)

Remote access of computers (e.g., via Telnet) and distributed computations, the other forms of remote job execution, will all require similar supportive services -- that is, authentication, access control, integrity, and confidentiality. In all remote job execution schemes, if the execution is done inter-domain, the administrative domains must use a mutually agreed upon set of control protocols; this may be established either by multi-lateral agreements or by some superior authority (for example, an act of Congress dictating a protocol to administratively-independent agencies.)

Remote access comes in many forms; some computers will simply supply services such as directory services and not allow other forms of remote access. These services will require the usual supportive services, but will also require that the client be able to authenticate the server so the client can be sure it is connected to the intended directory and the server can be sure the client is authorized to access the information. Note that this need not be necessary for non-directory services, since if access is made through a directory server and a session key is obtained, should the client then access a bogus (non-directory) server using the session key the bogus server will not be able to respond. Similarly, user authentication as a productive service will be essential when dealing with certificates designed to be used in a productive service. For example, the use of laptop computers will require the availability of user authentication at this level.

Another resource requiring distributed use of computers would be a ''national'' file system, allowing remote hosts throughout the country to access a shared set of files; it will require not only mechanisms for the usual supportive services but also a common interface protocol and a common file exchange protocol to allow systems with very different file accessing semantics to use the national file system.

Due to Office of Management and Budget (OMB) constraints at the federal level, and bookkeeping concerns in other agencies, businesses, and institutions, accounting for resources used in and by other administrative domains will be required; since (for example) the Dartmouth administrative domain will not trust the NASA administrative domain to account for the use of electronic mail sent from Dartmouth to NASA, both NASA and Dartmouth would undoubtedly track such mail and check the relevant bills. Non-repudiation of use of service is at this point essential.

Key distribution in support of secure mail, authentication mechanisms, and other services will require protocols and standards agreed to by different administrative domains. Such services may be integrated with directory servers but this is a matter of policy.

Finally, as different administrative domains communicate, network management and control information will have to be passed between administrative domains, raising issues of object integrity, confidentiality, and access control.

In terms of the four phases used to characterize the evolution of capability, at phase 0 is mail relaying, transfer, and name domains. Phase 1 technologies are authentication technologies such as secure-ID, challenge/ response protocols, and authentication servers such as Kerberos. On the border between phases 1 and 2 are the distributed white pages for the entire Internet. Phase 2 mechanisms such as secure mail and key distribution and management mechanisms are currently under development by the IAB Task Force on Privacy; other phase 2 items are certificates, and security of distributed directory servers (white pages). Distributed computation protocols and controls for a national file system, and accounting mechanisms are phase 3. Also phase 3 are "firewalls" for end-to-end services, so that if the services fail over a portion of the Internet the rest of the Internet may continue to rely on the service being correct and functional (this would limit the damage of incidents like the Internet worm of November 1988) and also the integrity of data across international borders, since most nations restrict the transborder use of cryptographic algorithms that can be used for secrecy, which is true of the base algorithms used in the computation of cryptographic checksums for integrity. Hence a solution requires the development of a cryptographic algorithm that can be used for integrity and authenticity, but not secrecy. One possibility is to use zero-sum knowledge mechanisms to have a third party assure integrity without secrecy, might be feasible. Such a solution is Phase 4 (very long range research).

6.4. Projects

The above suggests several projects that the FRICC or some constituent agency should pursue:

- End-to-end private mail is currently in the experimental phase; encryption is done using the DES, and authentication involves certificates built using RSA. The mechanism allows both privacy and integrity of sent mail.
- A national file system will raise issues of access control, authentication, confidentiality, and integrity.
- Directory services should provide white pages for mail and multi- domain object registration; issues to be addressed include registration of services, distributed list service, and authenticity.
- Finally, questions of multi-domain network monitoring and control are at the heart of interconnected network operations and raise issues of access control, authentication, and integrity.

Some common or interoperable approach to authentication, integrity, and access control, as well as the tools and services to be provided, is necessary; note the policies may differ across administrative domains, but the mechanisms must be able to

communicate with one another. They need not rely on each other, however; that is a policy issue. Whether or not these inter-domain mechanisms can be built with common facilities, the specific protocol base (such as OSI or TCP/IP) that these projects are to be conducted, how results are to be transferred into GOSIP and a European context, the role of vendors as opposed to researchers, and the IETF, IAB, and other such organizations, and which agency or agencies shall take the lead, are all issues that can be resolved in the longer range.

Notes: Reference for the use of productive and supportive services is the ECMA (European Computer Manufacturers Assoociation) Security in Open Systems, A Security Framework document, ECMA TR/46, July 1988.

7. Workshop Attendees

Guy Almes	Rice
Matt Bishop	Dartmouth
Brian Boesch	DARPA
Bill Bostwick	Los Alamos
Dennis Branstad	NIST
Hans-Werner Braun	Merit
Scott Brim	Cornell
Ross Callon	DEC
Vint Cerf	NRI
David Clark	MIT
Mike Corrigan	DoD
Jon Crowcroft	UCL
Richard desJardins	CTA
Deborah Estrin	USC
Steve Goldstein	Mitre
Phill Gross	NRI
Tony Hain	Livermore
Jim Hart	NASA
Jack Haverty	BBN
Dan Hitchcock	DoE
Anita Holmgren	Unisys
Barry Howard	Livermore
Bill Jones	NASA
Steve Kent	BBN
Larry Landweber	Wisconsin
Jim Leighton	Livermore
Barry Leiner	RIACS
Dan Lynch	ACE
Sandy Merola	Lawrence Berkeley Labs
James Morrill	Sparta
Russ Mundy	DCA
Dan Nessett	Livermore
Ari Ollikainen	RIACS
David Peters	NASA
Nachum Shacham	SRI
Henry Sowizral	RIACS
Mike St. Johns	DCA
Paul Tsuchiya	Mitre
Tony Villasenor	NASA
Steve Walker	TIS
Jil Westcott	BBN
Steve Wolff	NSF

Lixia Zhang MIT

8. Glossary

AR	Autonomous Region
CLNP	Connectionless Network Protocol
DARPA	Defense Advanced Research Projects Agency
DES	Data Encryption Standard
DoE	Department of Energy
ECMA	European Computer Manufacturers Association
FRICC	Federal Research Internet Coordinating Committee
GOSIP	Government OSI Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Standards Organization
LAN	Local Area Network
MTA	Mail Transfer Agent
NASA	National Aeronautics and Space Administration
NRI	National Research Internet
NSF	National Science Foundation
OMB	Office of Management and Budget
OSTP	White House Office of Science and Technology Policy
PS	Policy Server
PT	Policy Term
RSA	Rivest Shamir Algorithm
TAC	Terminal Access Controller
TOS	Type of Service
QOS	Quality of Service

Security Considerations

None.

Author's Address

Barry Leiner
Research Institute for Advanced Computer Science
National Aeronautics and Space Administration
Ames Research Center
Mail Stop 230-5
Moffett Field, CA 94035

Phone: (415) 694-5402

EMail: LEINER@RIACS.EDU