Internet 主机必要条件——应用程序和支持

本备忘录状态

本 RFC 文档是 Internet community 的正式规范。内容涉及有关主机的基本协议标准文档的修正和补充。发布本备忘录不受限制。

概要

本 RFC 文档是对 Internet 主机软件需求的定义和讨论的二个文档之一。本文档适用于应用程序和支持协议，另一文档 RFC-1122 适用于通信协议层：链路层，IP 层，和传输层。

目　　录

1.介绍

本 RFC 文档是对 Internet 主机系统的协议族执行需求的定义和讨论的二个文档之一。本文档适用于应用程序和支持协议，另一文档“Internet 主机必要条件——通信层”[INTRO:1]适用于较低层的协议：链路层，IP 层，和传输层。

本文档的目的是为 Internet 通信软件的厂商，执行者和用户提供指导。它代表大多数团体的相关经验和学问的意见，由 Internet 研究机构和厂商团体的成员共同研讨提供。

本 RFC 列举了连接到 Internet 的主机必须使用的标准协议，和描述这些协议的相关 RFC 和其他文档。本文纠正了这些参考文档中的错误，附加了额外的讨论，指导执行者执行。

本文中的每一个协议，都包含清楚的必要条件，建议和可选项。读者必须了解，本文中的必要条件列表是不完善的，Internet 主机完整的必要条件主要在它的正式协议规范文档中定义，及本文包含的改进，修正和补充。

一个协议的优秀具体实现是在仔细地阅读 RFC 文档，与 Internet 技术团体进行充分交流，并从事优秀的通信软件实践后提出的，它应与本文中的必要条件只有少许不同。所以，许多情况下，本文中的“必要条件”已经在正式的协议文档中作出规定， 这些内容可能让人感到有些多余。不过，之所以包含这些内容，是因为有些过去的具体实现已经进行了错误的选择，引起了互用性，执行和健壮性的问题。

本文档包含许多必要条件和建议的讨论和解释。必要条件的简单列表是不安全的，因为：

- 有些需要的特性比其他特性要重要得多，而另一些特性却是可选的。
- 另一个原因是有些特殊在厂商产品因环境的限制而设计，但使用了不同的规范。不过，本文档的规范必须可以适应Internet系统中内部操作具有多样性和复杂性的任意主机的一般目的。尽管目前大部分的执行在不同的情况下未能适合这些必要条件，但是，对于有些主要的或次要的系统，本规范都是我们要达到的理想状态。

这些必要条件基于当前的 Internet 体系结构水平。本文档对于还要发展的规范将继续更新，以提供这些领域的附加解释或包含附加的信息。

本介绍节首先给出对主机软件厂商一般的建议，还给出了阅读文档其他部分的指导。第 2 节包含可能用到的所有应用程序和支持协议的必要条件。第 3，4，5 节包含主要协议（分别是 Telnet ，文件传输，和电子邮件）的必要条件。第 6 节包括支持程序：域名系统，系统初始化和管理。最后，第七节列出所有的参考资料。

1．1 Internet 体系结构

从主机的观点对 Internet 体系结构作的基本介绍，请参看[INTRO:1]的第 1.1 节。该节还包含被推荐的有关 Internet 体系结构一般背景的参考资料。

1．2 一般考虑

有两个重要的课程需要 Internet 主机软件厂商学习，新开发商也应仔细考虑。

1．2．1 持续的 Internet 发展

高速成长的 Internet 已经显示出管理的问题，并描绘了一个巨大的基于包交换的通信系统。这些问题已经着手解决，导致的结果是本文中描述的规范将持续发展。这些改变会小心地有计划有控制地进行，因为该计划有广泛的负责网络运行的厂商和团体参与其中。

发展，演变和修正是今天计算机网络协议的特征，这种情况还将持续很多年。一个厂商开发了有关 Internet 协议族（或其他任何协议族！）的计算机通信软件，但是因为协议规范的发展，厂商对软件维护和更新的失败将给消费者造成很大的损失。Internet 是一个巨大的通信网络，用户不间断地与它保持连接。经验表明，开发商软件的缺陷将通过 Internet 技术社区迅速地传播。

1．2．2 健壮性法则

在协议的每一层，都有相应的有关应用的常用标准，它对健壮性和互用性的指导非常有益。

要编写的软件应对每个可能的错误进行处理，尽管这样的错误看起来不太可能发生，但迟早会有携带特殊错误的信息包出现。除非软件作好了准备，否则将引起混乱。通常，最好假定网络中充满不怀好意的用户，他们会发送经过特殊设计的将引起最坏结果的信息包。 这个假定将指引我们作出有保护的设计，虽然在Internet上由于低概率的事件引发的未曾想象到的反应已经造成了严重的问题。纯粹的怀有恶意的人将不会再使发展过程变得如此曲折。

Internet 主机软件对变化的适应性应在每一层中都进行设计。举一个简单的例子，一个协议规范包含的特定头域的值——比如，一个类型域，一个端口号，或一个错误代码，必须假定这些值组成的列表是不完善的。所以，如果一个协议规范定义了四个可能的错误代码，软件必须保证不会因被第五种代码的出现而中断。一个未被定义过的代码可以出现，但是不能因此引起系统故障。

法则的第二部分也同样重要：其他主机上的软件可能有包括正巧合法但却含义模糊的协议特征的缺点。这种不聪明地远离易理解性和简单性，唯恐在其他地方引起不幸的结果。这样的必然结果是“提防行为不当的主机”。

1．2．3 错误日志

Internet包含多种多样的主机和网关系统，每一个都执行许多协议和协议层，其中一些的Internet协议软件中有错误和不正确的特征。存在如此复杂、如此大的差异和多种功能分类的结果，通常是使对用户问题的诊断非常困难。

如果主机的执行中包含了一个经过仔细设计的工具，它可以记录错误的和“陌生的”协议事件，将对问题的诊断有所帮助。当错误发生时，记录下

大量可能包含的诊断信息非常重要。特别是，通常将引起错误的信息包的头信息记录下来会很有用。不过，必须认真设计，以保证错误日志不会消耗大量的资源或对主机操作的其他方面造成干扰。

一个反常但是无害的趋势是，错误日志文件会因协议事件的发生而溢出，这可以通过使用“循环”日志来消除，或使日志文件仅对诊断已知的故障进行记录。对连续重复的消息的过滤和计数是有益的。好像有一个策略工作得很好：（1）在管理协议期间经常对异常状况和这些异常状况的带来的影响进行记录（参看 6.3 节）。（2）允许大量多样的事件的记录有选择地被激活。例如，能够“记录所有事件”或“对主机 X 记录所有事件” 是有益的。

注意，关于主机中正常有效的错误记录的数量因处理的不同而有不同的策略。有些人会说，“如果它没有伤害到我，不想知道关于它的事情”，另一些人则对它更加警惕，并用积极的态度去侦测和解除协议的紊乱。

1．2．4 配置

如果主机对 Internet 协议族的实现可以完全自动配置，那是最理想的情况了。这将允许整个协议族在 ROM 中或在硅片中执行，它将简化为无盘工作站，也将因局域网管理员和系统厂商的消失而我们带来无尽的好处。但是，我们还没有达到这样的理想状态，甚至还没有接近。

在本文中的许多地方，你会发现对一个参数的可配置性是其必要条件之一。在这样的需求背后有几个不同的原因。少数情况下，关于最好的值还没有确定或仍有争执，在未来可能需要更新被推荐的值。其他情况下，该值依赖于外部因素——比如，主机的容量和它对通信负荷的分配，或者附近网络的速度和技术，并且自调整算法是不可行的或是不完善的。有些情况下，可配置能力是因为管理的需要。

最后，有些对可选项的配置，是因为要与已废除的或不正确的协议实现，或无消息来源的分配进行通信的需要，不幸的是这些现象在 Internet 的许多地方顽固地存在着。为了让正确的系统与这些不完善的系统共存，管理员通常要对正确的系统进行“错误的配置”。这个问题会随着不完善系统的退役而逐渐纠正，但是这个问题不能被厂商忽略。

当我们说一个参数必须是可配置的时候，并没有要求它的值在每次启动的时候都从一个配置文件中读取。我们推荐执行者为每个参数都设置一个缺省值，这样配置文件就只有在特殊的安装情况下因缺省值的不合适才会需要。所以，对可配置性的需求是为了确保超出缺省值的可能情况的发生，甚至是在一个仅二进制或基于 ROM 的产品中。

本文档需要在某些情况下指出缺省值的数值。当配置的项目对现有的不完善系统进行适应性调节时，对缺省值的选择会变得非常敏感。如果 Internet 因完全的互用性而成功地集合成一个整体，那么缺省值的设置必须执行正式协议，而不是“错误的配置”以适应不完善的实现。虽然有些厂商为了销售的利润把错误的配置作为缺省值，但我们强烈要求厂商遵照标准选择缺省值。

最后，我们提出，厂商需要为所有配置参数的限制和影响提供足够的参考文件。

1．3 阅读本文档

1．3．1 组织结构

通常，每个主要的节都由下列的子章节组成：

（1）介绍

（2）协议准备——考虑到协议规范文档是一段段的，声明的需求也可能是模糊的或定义有误的，纠正错误，并提供更清晰的解释说明。

（3）特殊问题——讨论协议准备中没有包含的协议设计和执行实现的问题。

（4）界面——讨论其后更高层的服务界面。

（5）总结——包含节中必要条件的一览表。

在本文档的一些个别主题中，会包含标记为“讨论”或“执行”的附加资料。这些资料对前面给出的必要条件文本进行更清晰的解释说明。也包含对未来可能的发展方向的建设。执行材料包含执行者要考虑的问题的建议方法。

总结节对文本进行指南和索引，但它必定是含糊的和不完善的。总结节决不能从完整的 RFC 中个别地使用或引用。

1．3．2 必要条件

在本文档中，用来定义每一个特殊的必要条件的重要性的单词都是大写的。这些单词是：

- “MUST”（必须）

这个单词或形容词“REQUIRED”（必需的）意味着该项目是本规范的绝对必要条件。

- “SHOULD”（应该）

这个单词或形容词“RECOMMENDED”（推荐的）意味着在特殊的环境下存在忽略该项目的有效原因，但是完整的具体实现应能理解，并且在选择一个不同的进程时应对这种情况进行仔细的斟酌。

- “MAY”（可以）

这个单词或形容词“OPTIONAL”（可选的）意味着这个项目是真正可选的。厂商选择包含这个项目是因为特殊的市场需求或提高产品的竞争力，比如，其他厂商忽略了这个项目。

具体实现如果不能满足一个或多个协议执行的“必须”的必要条件，它就没有遵守协议。执行实现满足所有“必须”的和所有的“应该”的必要条件，就称它是“无条件服从”；如果满足所有“必须”的必要条件，但是未满足所有“应该”的必要条件，就称它是“有条件服从”。

1．3．3 术语

本文档使用了下列技术术语：

Segment（段）

段是 TCP 协议中终端到终端传输的单位。一个段由 TCP 头信息和其后的应用数据组成。段以封装到 IP 数据包中的形式来传输。

Message（消息）

这个术语在一些应用层协议（特别是 SMTP）中用来表示应用数据单位。

Datagram（数据包）

[UDP]数据包是 UDP 协议中终端到终端传输的单位。

Multihomed（多源的）

一个主机如果有多个 IP 地址连接网络，它就被称为是多源的。

1．4 感谢

（略——译者注）

2．一般问题

本节包含适用于所有应用层协议的一般必要条件。

2．1 主机名和编号

RFC-952[DNS:4]中定义了合法的 Internet 主机名的语法。对主机名语法的形式由此改变：对第一个字符的要求不再严格，允许使用字母或数字。主机软件必须支持这个更加自由的语法。

主机软件必须处理多达 63 个字符的主机名，应该支持多达 255 个字符的主机名。

无论何时用户指定 Internet 主机，应该是以下两种形式：（1）主机域名；（2）由点分隔的十进制数组成的 IP 地址(″#.#.#.#″)。主机应在域名系统中查找之前，检查用点分隔的十进制数组成的 IP 地址。

讨论：

最后一个要求是不能刻意指定输入用点分隔的十进制主机编号的形式，因为这涉及到用户界面的问题。例如，对于 SMTP 邮件，用点分隔的十进制数必须用“[ ]”括起来（参看第 5.2.17 节），这个符号把主机的表示统一起来，简化了用点分隔的十进制数的语法检查。

如果一个用点分隔的十进制数没有用这样的识别符号输入，那么就要执行一个完整的语法检查，因为现在允许主机域名用数字开头，并且全部使用数字也是合法的（参看第 6.1.2.4 节）。不过，一个有效的主机名从来不会用点分隔的十进制数形式(″#.#.#.#″)，因为至少最高级的标志部分是用字母的。

2．2 使用域名服务

主机域名必须如 6.1 节描述的那样被转换成 IP 地址。

使用域名服务的应用程序必须能够应付软件错误的情况。应用程序在因软件错误而造成的连续重试之间应等待合理的时间间隔，必须可以处理因网络问题造成服务被拒绝若干小时甚至若干天的可能性。

应用程序不应该依赖于查找WKS记录，其中包含有在特定主机地址上提供的所有服务清单。因为WKS RR类型通常不是供Internet 站点使用的。为了批准一个提交的服务，应简单地尝试使用一下。

2．3 多源主机上的应用程序

当远程主机是多源主机时，名字-地址转换将返回一个可选的 IP 地址列表。如第 6.1.3.4 节中指出的那样，这个列表应以递减的顺序排列。应用协议的实现应为多次重试列表中的多源地址直到获得成功作好准备。对 SMTP 的更多需求在第 5.3.4 节中给出。

当本地主机是多源的时，一个基于 UDP 的请求/回应应用应该发送带有 IP 源地址的回应，该 IP 源地址与 UDP 请求数据包中“特定的目的地址”相同。“特定的目的地址”在本文的伴随 RFC[INTRO:1]的“IP 地址”一节中定义。

同样，一个为相同客户端建立了多个TCP连接的服务器程序，也应当使用相同的本地IP地址。

2．4 服务类型

当应用程序调用传输层服务时，必须选择合适的 TOS 值，并且这些数值必须是可配置的。注意，TOS 值包含 5 位，仅有 3 位是目前已定义的，其余两位必须为零。

讨论：

因为网关算法的发展已可实现服务类型，所以对于不同的应用协议推荐的值是不同的。另外，特殊的用户和 Internet 路径的组合可能会使用非标准的 TOS 值。因为这些原因，TOS 值必须是可配置的。

参看最新版本的 RFC“编号的分配” [INTRO:5]中对主要应用协议的 TOS 推荐值。

2．5 一般应用程序必要条件一览

特性	章节	SHOULD-0	MUST-0	SHOULD-NOT	MUST-NOT	Footnote
用户界面:						
允许主机名以数字开头	2.1	x				
主机名可达635字符	2.1	x				
主机名可达255字符	2.1		x			
支持以点分隔的十进制数的主机编号	2.1		x			
首先对以点分隔的十进制数进行语法检查	2.1		x			
根据第6.1节映射域名	2.2	x				
应对软件DNS错误	2.2	x				
重试时保留合理的时间间隔	2.2	x				
允许长时间中断	2.2	x				
要求WKS记录可用	2.2			x		

对远程多源主机尝试多个地址	2.3		x			
UDP回应的源地址是请求中特定的目的地址	2.3		x			
对关联的TCP连接使用相同的IP地址	2.3		x			
指定合适的TOS值	2.4	x				
TOS值可配置	2.4	x				
不使用TOS的零位	2.4	x				

3．远程登录——Telnet 协议

3．1 介绍

Telnet 是远程登录的标准 Internet 应用层。它提供了把客户端（“用户”）系统中的键盘/显示与远程服务器系统的命令解释器连接起来的编码规则。Telnet 协议的子集也作为其他应用协议（比如，FTP 和 SMTP）的组成部分。

Telnet 使用单一的 TCP 连接，它的正常数据流（“网络虚拟终端”或“NVT”模式）是用转义符嵌入控制功能的 7 位 ASCII。Telnet 还允许对许多可选的模式和功能进行商谈。

主要的 Telnet 协议规范在 RFC-854[TELNET:1]中定义，可选项在许多其他的 RFC 中定义，参看第 7 节的参考资料。

3．2 协议准备

3．2．1 选项商谈：RFC-854 第 2-3 页

全部的 Telnet 具体实现必须包含选项商谈机制[TELNET:2]。

主机必须仔细遵从 RFC-854 的规则，以避免选项商谈的循环。主机必须拒绝（比如，对 DO/WILL 回应 WONT/DONT）不支持的选项。选项商谈功能必须在 Telnet 连接的整个生存时间内保持有效（即使所有的请求都被拒绝）。

如果选项商谈失败，Telnet 执行必须使用缺省值，并支持 NVT。

讨论：

尽管许多已广泛应用的“终端”和其支持的选项商谈正变成标准，但所有的执行必须为支持一个在任何用户-服务器之间通信的 NVT 做好准备。

3．2．2 Telnet 的 Go-Ahead 功能：RFC-854 第 5 页，和 RFC-858

在一个从不发送 Telnet 的 Go Ahead（GA）指令的主机上，Telnet 服务器必须尝试商谈 Suppress Go Ahead 选项（也就是发送“WILL Suppress Go Ahead”）。用户或服务器 Telnet 必须总可接受 Suppress Go Ahead 选项的商谈。

对一个全双工运行的终端来说，GA 是无意义的，用户 Telnet 的具体实现可以忽略 GA 指令。

讨论：

被设计支持Go-Ahead机制的半双工（“锁定键盘”）line-at-a-time终端已经大量消失。它使得在许多操作系统中执行发送Go-Ahead信号变得困难，甚至一些支持自然半双工的终端。该困难是服务代码不能访问关于是否用户进程被锁定等来自Telnet连接的输入，也就是，它没有测定何时发送GA指令的能力。所以大多数的Telnet服务器主机不发送GA指令。

本节的规则的作用是允许 Telnet 连接的任一端可以禁止 GA 指令的使用。

有一类半双工的终端在商业上很重要：“数据输入终端”，它们工作在全屏方式下。不过，支持使用 Telnet 协议的输入终端无需 Go Ahead 信号，参看第 3.3.2 节。

3．2．3 控制功能：RFC-854 第 7-8 页

Telnet指令列表已经被扩展到支持EOR（End-of-Record），使用代码239[TELNET:9]。

用户和服务器Telnet可以支持控制功能EOR, EC, EL, 和Break, 必须支持AO, AYT, DM, IP, NOP, SB, 和 SE。

主机必须可以接收和忽略任何它不支持的 Telnet 控制功能。

讨论：

注意：服务器Telnet需要支持Telnet IP（中断进程）功能，即使服务器主机已经有等价的in-stream 功能（比如，在许多系统中是Control-C）。Telnet IP功能比in-stream中断指令有更好的健壮性，因为TCP加急数据会有超出带宽的事件。

EOR 控制功能可能会限制数据流。一个重要的应用是数据输入终端支持（参看第 3.3.2 节）。应该关注，因为 EOR 没有在 RFC-854 中定义，一个没有准备好正确忽略未知 Telnet 指令的主机可能在它接收到 EOR 时崩溃。为了保护这样的主机，应引入记录结束可选项；不过，一个完全实现的 Telnet 程序不需这样的保护。

3．2．4 Telnet“同步”信号：RFC-854 第 8-10 页

当接收到“加急的”TCP 数据，用户或服务器 Telnet 必须丢弃除 Telnet 指令之外的所有数据，直到接收到 DM（和加急结束符）。

当发送 Telnet IP（中断进程）时，用户 Telnet 应在其后跟随 Telnet“同步信号”序列，也就是发送 TCP 加急数据序列“IAC IP IAC DM”。TCP 加急指示器指出 DM 字节。

当接收到 Telnet IP 指令时，服务器 Telnet 可以给用户发送回一个 Telnet“同步信号”序列，以清洗输出的数据流。当本地用户中断一个进程时，选择权应与服务器操作系统的行为方式保持一致。

当接收到 Telnet AO 指令时，服务器 Telnet 必须给用户发送回一个 Telnet“同步信号”序列，以清洗输出的数据流。

当发送 Telnet IP 时，用户 Telnet 应具有清洗输出的能力，参看第 3.4.5 节。

讨论：

用户 Telnet 清洗服务器输出数据流，有三种可能的方式：

（1）在 IP 后发送 AO。

这将引起服务器主机发送一个“清洗缓冲的输出”信号到它的操作系统。不过，AO 可能在本地不起作用，也就是说，停止在用户 Telnet 端的终端输出，直到服务器 Telnet 接收并处理了 AO，发送回一个“同步信号”为止。

（2）在 IP 后发送 DO TIMING-MARK [TELNET:7]，并丢弃所有的本地输出直到接收到来自服务器 Telnet 的 WILL/WONT TIMING-MARK。

因为 DO TIMING-MARK 将在服务器上在 IP 之后处理，它的回应应该在输出数据流的正确位置。不过，TIMING-MARK 不会发送“清洗缓冲的输出”信号到服务器操作系统。它是否需要依赖于服务器系统。

（3）联合使用前面的两种方式

最好的方式并不明确，因为必须适应大量现有的以各种方式未遵从 Telnet 标准的服务器主机。最安全的方式可能是提供一个用户中选择的选项，让用户去选择（1），（2）或（3）。

3．2．5 NVT 打印机和键盘：RFC-854 第 11 页

在 NVT 模式下，Telnet 不应发送高位为 1 的字符，并且不能把它作为校验位发送。传送高位到应用程序的具体实现应商谈二进制模式（参看第 3.2.6 节）。

讨论：

执行者应知道，按照 RFC-854 严格读取，客户端和服务器允许 NVT ASCII，会忽略对高位进行设置的字符。通常，二进制模式被认为用来在 Telnet 中传输扩展的（超过 7 位的）字符集。

不过，现有的应用程序确实需要 8 位的 NVT 模式，但目前还没有被定义，并且这些现有的应用程序在 Telnet 连接生命的部分或全部时间里，对高位进行了设置。注意，二进制模式与 8 位的 NVT 模式不同，因为二进制模式关闭了行结束进程。因为这个原因，对高位的需求被规定为应该的，而不是必须的。

RFC-854 定义一个“网络虚拟终端”或 NVT 的最小特性集，但不意味着在真实的终端上排除其他特征。Telnet 传输对所有的 7 位 ASCII 字符完全透明，包括任意的 ASCII 控制字符。

例如，终端应支持像 ASCII 转义符序列这样的全屏指令代码，Telnet 的具体实现将以不可解释的数据来接收这些数据。所以，NVT 不会被认为是一个高位受限类型的终端设备。

3．2．6 Telnet 指令体系结构：RFC-854 第 13 页

因为可选项可以出现在数据流中的任意位置，Telnet 转义符（如已知的 IAC，其值为 255）作为数据发送时必须同时发送相同的两个。

3．2．7 Telnet 二进制选项：RFC-856

当二进制选项成功地商谈之后，允许使用任意的 8 位字符。不过，数据流必须仍做 IAC 字符的检查，任何内部的 Telnet 指令必须遵从，数据字节等于 IAC 的必须被复写两次。其他字符进程（比如，将 CR 替换为 CR NUL 或 CR LF）不能执行。特别是，在二进制模式下没有行结束符的约定。

讨论：

二进制选项的商谈通常是双向的，为了将 Telnet 连接从 NVT 模式改变到“二进制模式”。

IAC EOR 序列可以被用来在二进制 Telnet 信息流内部划分数据块。

3．2．8 Telnet 的终端类型选项：RFC-1091

当终端类型对特殊的终端有效时，终端类型选项必须使用在分配编号 RFC[INTRO:5]中定义的正式的终端类型名字。不过，终端类型选项的接收方必须接受任意的名字。

讨论：

RFC-1091[TELNET:10]更新了一个早期在 RFC-930 中定义的终端类型选项的版本。这个早期的版本有能力支持多种终端类型去获取特殊客户终端类型，假定每个物理终端有一个固有的类型。不过，今天的“终端”通常真正是运行在 PC 上的终端模拟程序，可能有模拟多个终端类型的能力。所以，RFC-1091 对允许在用户与服务器 Telnet 之间一般终端类型的商谈规范作了扩充。

3．3 特殊问题

3．3．1 Telnet 行结束符约定

Telnet 协议定义 CR LF 序列的意义是“行结束符”。对终端输入来说，这相当于在用户终端上按下了指令完成或“行结束符”键；在 ASCII 终端上，这个键是 CR，但它也被称作是“Return”或“Enter”。

当服务器 Telnet 接收到来自远程终端输入的 Telnet 行结束符序列 CR LF 时，其作用必须与用户在本地终端按下“行结束符”键的效果一样。在使用 ASCII 的服务器主机上，特别是，当用户在本地终端上按下 CR 键，服务器接收到的 CR LF 序列必须与本地终端的效果一样。所以，当一个 ASCII 服务器主机从 Telnet 连接上接收到 CR 或 CR NUL 时，它们的效果应该一样。

用户 Telnet 必须可以以任何的形式发送：CR LF，CR NUL 和 LF。在一个 ASCII 主机的用户 Telnet 应该有一个用户可控模式，在用户按下“行结束符 ”键时发送 CR LF 或 CR NUL，并且 CR LF 应是缺省的。

Telnet 行结束符序列 CR LF 必须被用来发送 Telnet 数据（比如，将 Telnet 协议并入到其他应用协议中）。

为了保持任意 Telnet 客户端和服务器的交互性，Telnet 协议为行结束符定义了一个标准的表示。因为 ASCII 字符集没有清楚地包含行结束字符，所以系统可以选择不同的表示，比如，CR，LF，和 CR LF 序列。Telnet 协议选择 CR LF 作为网络传输的标准。

不幸的是，在 RFC854 [TELNET:1]中定义的 Telnet 协议规范，已经将用哪个字符作为从客户端发送到服务器的“行结束”键搞得有些不明确。其结果是造成了大量的并还在继续的令人头痛的交互性问题，而且因用户和服务器 Telnet 的各种有缺陷的具体实现而变得更糟。

虽然 Telnet 协议基于一个完美的对称模型来建立，在远程登录的过程中，终端用户的角色与服务器主机的角色是不同的。例如，RFC-854 定义了 CR，LF，和 CR LF 作为从服务器的输出，但并没有定义当用户在终端上按下“行结束”键时,用户 Telnet 应发送什么字符；所以在这点上造成了分歧。

当用户按下“行结束”键时，有些用户 Telnet 具体实现发送 CR LF，而另一些则发送 CR NUL（基于对 RFC-854 中相同句子的不同解释）。这对一个正确实现的 ASCII 服务器主机来说是等价的，如上面所讨论的那样。对其他的服务器，对于用户 Telnet 只需要一种模式。

只能发送 CR NUL 的现有用户 Telnet，当 CR 被按下时就为非 ASCII 的主机造成了两难的选择：它们可以将 CR NUL 视为与 CR LF 等价的输入，所以排除了输入“单独的”CR 的可能性；或者使完整的交互性遭受损失。

假定主机 A 的一个用户 Telnet 到了一个服务器主机 B，然后执行 B 的用户 Telnet 程序登录服务器主机 C。最理想的状态是服务器/用户 Telnet 与 B 的结合是透明的，也就是说，看起来就好象 A 直接连接到了 C。特别是，正确的具体实现将使 B 对 Telnet 行结束符序列透明，除了 CR LF 可能被转换成 CR NUL 或相反的转换。

执行：

为了理解 Telnet 行结束符问题，必须有至少一个将 Telnet 与本地操作系统关联起来的通用模型。服务器 Telnet 进程被结合到作为伪终端的操作系统的终端驱动软件中。由服务器 Telnet 接收到的 Telnet 行结束符序列必须与在一个真正的本地连接终端上按下行结束键的效果相同。

支持每个字符交互性的操作系统的应用程序（比如，editor）对它们的终端 I/O 有两个内部模式：一个是格式化模式，它对行结束符有本地约定，对数据流应用其他的格式规则；另一个是“自然”格式，应用程序对每一个输入的字符进行直接的存取。服务器 Telnet 必须以这样一种方式来实现，即它们的模式对远程终端与本地终端有一样的效果。例如，在一个 ASCII 主机上支持从服务器 Telnet 接收到的 CR LF 或 CR NUL。在自然模式下，CR 字符被传送到应用程序；在格式化模式下，使用本地系统的行结束符约定。

3．3．2 数据输入终端

讨论：

除了导向行和导向字符之外，Telnet 的本意是设计一种 ASCII 终端，有几个视频显示器终端家族如已知的“数据输入终端”或 DET。IBM 3270 家族是一个著名的例子。

为了支持一般的 DET，已经设计了两个 Internet 协议：SUPDUP [TELNET:16, TELNET:17]，和DET选项 [TELNET:18, TELNET:19]。DET选项在一个使用（子）商谈的Telnet连接上驱动数据输入终端。SUPDUP是一个完全分离的终端协议，它可以通过商谈从Telnet进入。虽然SUPDUP和DET选项已经被成功地应用在特殊的环境下，但是两者都没有获得全面的承认或广泛的执行。

一个用于DET交互的不同方案已经被发展成可以通过Telnet支持IBM3270

家族，虽然相同的方案可以被应用到任何DET。它的概念是进入一个“自然DET”模式，该模式下自然的DET输入/输出流被以二进制数据发送。Telnet EOR指令在这些二进制数据流中被用来划分逻辑记录（比如，“screens”）。

执行：

进入和离开自然DET模式的规则如下：

●服务器使用终端类选项[TELNET:10]来获取客户端是一个DET。

●作为惯例，但不是必需的，连接的两端进行EOR选项[TELNET:9]的商谈。

●连接的两端进行二进制选项[TELNET:3]商谈，以进入自然DET模式。

●当连接的任一端提出脱离二进制模式的商谈，另一端也要脱离，并且模式恢复到自然NVT。

3．3．3 选项必要条件

每个Telnet具体实现必须支持二进制选项 [TELNET:3]和Suppress Go Ahead 选项 [TELNET:5]，应该支持Echo[TELNET:4], Status [TELNET:6], End-of-Record [TELNET:9], 和扩展选项列表[TELNET:8]选项。

如果本地的操作系统提供了相应的能力，用户或服务器 Telnet 应该支持 Window Size 选项。

讨论：

注意，End-of-Record 选项只表示 Telnet 可以接收没有崩溃的 Telnet EOR；所以，每个 Telnet 应该愿意接受 End-of-Record 选项的商谈。参看第 3.2.3 节的讨论。

3．3．4 选项初始化

当 Telnet 协议被应用在客户/服务器环境下时，服务器应该对它期望的交互模式进行初始化商谈。

讨论：

Telnet 协议被定义为完全对称的，但它的应用却通常是不对称的。已知远程登录会因连接的双方未对需要的非缺省终端模式进行初始化商谈而失败。决定首选模式的通常是服务器，所以服务器需要初始化商谈，因为商谈是对称的，用户也会初始化。

客户端（用户 Telnet）应该为用户提供一种手段，使其可以禁用或启用选项商谈的初始化。

讨论：

用户有时需要连接到应用服务（比如，FTP 或 SMTP），它们使用 Telnet 作为控制流，但不支持 Telnet 选项。如果选项商谈的初始化是无效的，用户 Telnet 可以用于此目的。

3．3．5 Telnet 行模式选项

一个重要的新 Telnet 选项，行模式[TELNET:12]，已经被提议。行模式选项提供了一个标准的方式，可以让用户 Telnet 和服务器 Telnet 同意客户端而不是服务器来执行终端字符处理。当客户端准备好一个文本的完成行时，（通常）会在一个 TCP 信息包中将它发送到服务器。这个选项将大量减少 Telnet 过程的信息包代价，并将在拥挤的或长时间延迟的网络上给用户提供更好的回应。

行模式选项允许在本地和远程字符处理过程中进行动态的开关转换。例如，一个全屏的编辑器运行时，Telnet 连接将自动商谈单个字符模式，当编辑器结束运行时将返回到行模式。

我们期望在本 RFC 发布后，客户端主机应该实现这个选项，服务器端主机可以实现这个选项。为了在服务器端完全实现，服务器需要能够告诉本地系统不要进行任何输入字符的处理，但是要记住它当前的终端状态，并在状态改变的任何时候告知服务器 Telnet 进程。例如，它将允许口令在屏幕上禁止显示，全屏的编辑器也可以被完全处理。

3．4 Telnet/用户界面

3．4．1 字符集透明性

用户 Telnet 的具体实现应可发送和接收任意的 7 位 ASCII 字符。在可能的情况下，用户主机的操作系统应忽略对任何特殊字符的解释，所以这些字符才能方便地在连接上发送和接收。

在“换码到指令模式”时有些字符值必须被保留。按照惯例，允许两个相同的字符作数据输入。用户应有对特殊字符使用的选择权。

在二进制连接下，如果主机操作系统不允许任意 8 位的数值从键盘输入，那么用户 Telnet 程序可以为此提供一个换码机制。

执行：

透明性问题对服务器来说不太迫切，但执行者应小心处理下列类似情况：在他们到达仅要求 NVT ASCII 的程序之前，掩盖奇偶校验位（由一个旧的，不符合标准的客户端发送来的），适当处理需要 8 位数据流的程序。

3．4．2 Telnet 指令

用户 Telnet 程序必须为用户提供输入任意 Telnet 控制功能 IP、AO 或 AYT 的能力，并应该提供输入 EC、EL 和 Break 的能力。

3．4．3 TCP 连接错误

用户 Telnet 程序应向用户报告任何传输层报告来的 TCP 错误（参看[INTRO:1]的“TCP/应用层界面”一节）。

3．4．4 非缺省的 Telnet 连接端口

用户 Telnet 程序应允许用户可选地指定在服务器 Telnet 主机上的非标准连接端口号。

3．4．5 清洗输出

用户 Telnet 程序应该为用户提供在 IP 被发送时，指定是否清洗输出的能力。参看第 3.2.4 节。

对任何会引起用户Telnet清洗本地输出直到接收到来自服务器的Telnet信号，应该有让用户手动地恢复正常输出的手段，以防万一服务器发送信号失败。

3．5 Telnet 必要条件一览

特性	章节	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
选项商谈	3.2.1	x					
避免商谈循环	3.2.1	x					
拒绝不支持的选项	3.2.1	x					
在连接的任何时间可商谈	3.2.1		x				
缺省NVT	3.2.1	x					
在术语类型选项中发送正式名字	3.2.8	x					
在术语类型选项中接受任何名字	3.2.8	x					
实现Binary、Suppress-GA选项	3.3.3	x					
Echo, Status, EOL, Ext-Opt-List 选项	3.3.3		x				
在适当的时候实现Window-Size选项	3.3.3		x				
服务器初始模式商谈	3.3.4		x				
用户可以允许/禁止初始商谈	3.3.4		x				
Go-Aheads							
非GA服务商谈SUPPRESS-GA选项	3.2.2	x					
用户或服务器接受SUPPRESS-GA选项	3.2.2	x					
用户Telnet忽略GA	3.2.2			x			
控制功能							
支持SE NOP DM IP AO AYT SB	3.2.3	x					

支持EOR EC EL Break	3.2.3			x		
忽略不支持的控制功能	3.2.3	x				
用户，服务器丢弃加急数据直到DM	3.2.4	x				
用户Telnet在IP, AO, AYT之后发送“同步信号”	3.2.4		x			
服务器Telnet对IP回应同步信号	3.2.4			x		
服务器Telnet对AO回应同步信号	3.2.4	x				
用户Telnet当发送IP时可清洗输出	3.2.4		x			
编码						
在NVT模式下发送Send高位	3.2.5				x	
把高位作为奇偶校验位发送	3.2.5					x
商谈BINARY是否可传递高位到应用程序	3.2.5		x			
总是重复IAC数据字节	3.2.6	x				
在二进制模式下重复两次IAC数据	3.2.7	x				
在二进制模式下执行Telnet指令	3.2.7	x				
二进制模式下用CR NUL作行结束符	3.2.7					x
行结束符						
在服务器上的EOL作为本地行结束符	3.3.1	x				
ASCII服务接受CR LF 或 CR NUL 作为行结束符	3.3.1	x				
用户Telnet可以发送CR LF, CR NUL, 或 LF	3.3.1	x				
ASCII用户可以选择CR LF/CR NUL	3.3.1		x			
用户Telnet缺省模式是CR LF	3.3.1		x			
非交互式用户使用CR LF作为EOL	3.3.1	x				
用户Telnet界面						
输入&输出所有的7位字符	3.4.1		x			
绕过本地操作系统的解释	3.4.1		x			
转义字符	3.4.1	x				
用户可设置转义字符	3.4.1		x			
避免输入8位的值	3.4.1			x		
可以输入IP, AO, AYT	3.4.2	x				
可以输入EC, EL, Break	3.4.2		x			
向用户报告TCP连接错误	3.4.3		x			
可选的非缺省连接端口	3.4.4		x			
可指定：当IP发送时清洗输出	3.4.5		x			
可手动恢复输出模式	3.4.5		x			

4． 文件传输

4．1 文件传输协议——FTP

4．1．1 介绍

文件传输协议 FTP 是文件传输的主要 Internet 标准。当前规范定义于 RFC-959[FTP:1]中。

FTP 将同时发生的 TCP 控制连接和数据传输分隔开。FTP 包含许多特征，其中一些在普通执行时用不到。不过，每个 FTP 特征都至少有一个实现。RFC-959 定义的最小执行标准太小了，所以这里定义了一个稍微大一些的执行标准。

Internet 用户不必负担多年来 FTP 具体实现的不足。协议的执行者已经因为把 FTP 当作小的和价值不高的作业付出了代价。因为 FTP 有用户界面，因为它要处理可能发生的通信的多样性和操作系统的错误，还因为它要处理世界上大量多样的文件系统。

4．1．2 协议准备

4．1．2．1 本地类型：RFC-959 第 3.1.1.4 节

FTP 程序必须支持 TYPE I（“IMAGE”或二进制类型），也要支持 TYPE L 8（逻辑字节大小为 8 的“本地”类型）。还需一个机制，如果主机的内存用 m 位的字来组织，并且 m 不是 8 的倍数，那么也要支持 TYPE L m。

讨论：

“TYPE L 8”指令经常用于在一个内存用 36 位的字组织的机器和一个用 8 位字节组织的机器之间传输二进数据。对一个 8 位字节的机器来说，TYPE L 8 与 IMAGE 类型相同。

“TYPE L m”有时用来指定在两台 m 位的字的机器上的 FTP 程序，以确保从一台机器传本地模式的二进制文件到另一台机器。不过，该指令应像“TYPE I”一样对两台机器均有作用。

4．1．2．2 Telnet 格式控制：RFC-959 的第 3.1.1.5.2 节

如果主机对TYPE N 和TYPE T没有差别，应执行TYPE T。

讨论：

该规定可以减少有这样差异的主机的内部操作。

许多主机对文本文件用 ASCII 字符串表示，使用内含的 ASCII 格式控制字符（LF，BS，FF，……）控制文件打印时的格式。对这样的主机来说，“打印”文件和其他文件没有差别。不过，系统使用特定的记录结构文件就需要特殊的格式来处理可打印文件（比如，ASA 走纸控制）。对后一种主机，FTP 允许从 TYPE N 和 TYPE T 中选一。

4．1．2．3 页结构：RFC-959 的第 3.1.2.3 节和附录 I

通常不推荐页结构的执行实现。不过，如果一个主机系统需要 FTP 执行“随机存取”或“holey”文件，那么就必须使用已定义的页结构而不是定义一个新的私有的 FTP 格式。

4．1．2．4 数据结构传输：RFC-959 的第 3.1.2 节

记录结构和文件结构之间的FTP转换应是可逆的，当目的主机有扩展需求的可能时，在记录结构和文件结构之间的FTP传输应是可逆的。

讨论：

RFC-959 要求记录结构和文件结构之间严格可逆。但为了高效和方便使用，通常要牺牲可逆性。传输一个文件通常有两个不同目的：在目标主机处理或仅是为了存储。对处理来说，要使用应用程序以主机期望的格式来建立文件。

举一个引起冲突的例子。假定一个面向记录的操作系统要求文件严格遵守每记录 80 字节的规定，当在这样的主机执行 STOR 指令存储文件时，FTP 服务器必须可以将每一行或记录加到 80 字节，但是以后要想恢复这个文件却不能严格地逆操作了。

4．1．2．5 数据连接管理：RFC-959 的第 3.3 节

使用流模式的 User-FTP 在发送任意的传输指令之前，应先发送一条 PORT 指令来分配一个非缺省的数据端口。

讨论：

这样的方式是必需的。因为在一个TCP连接关闭之后要有一个长时间的延迟，直到它的连接对可以重新使用，这样就允许在一个FTP过程中可以有多个传输操作。发送一个PORT指令可以避免如果使用了非流模式的传输模式，通过离开在传输之间建立的数据传输连接。

4．1．2．6 PASV 指令：RFC-959 的第 4.1.2 节

server-FTP 必须实现 PASV 指令。

如果复杂的三方传输在相同的任务中被执行，在任何传输指令发送之前必须先发送一个新的 PASV 指令，它包含唯一的端口对。

执行：

PASV 指令的 227 回应的格式没有完全标准化。特别是当 FTP 客户端不能采用 RFC-959 的第 40 页给出的圆括号（事实上，第 43 页的图 3 忽略了它们）。所以，解释 PASV 回应的 user-FTP 程序必须检测主机的第一个数字和端口号。

注意：主机编号 h1,h2,h3,h4 是发送回应的服务器主机的 IP 地址，p1,p2 是 PASV 分配的非缺省数据传输端口。

4．1．2．7 LIST 和 NLST 指令：RFC-959 的第 4.1.3 节

NLST 指令返回的数据必须仅包含一个合法路径名的单一列表，它们应该可以直接被服务器作为后面的某个文件的传输指令中的参数。

LIST 和 NLST 指令返回的数据应使用一个隐含的类型 AN，除非当前的类型是 EBCDIC，其他情况下默认的类型应是 EN。

讨论：

许多 FTP 客户端支持微指令，它们可以在 NLST 指令中通过通配符包含路径名的列表，以上传或下载多个文件。“多文件上传”的扩展只需客户机的本地实现，但是“多文件下载”却需要服务器的协作。

LIST 和 NLST 的默认类型为现有的 user-FTP，特别是“多文件下载”指令提供兼容性。

4．1．2．8 SITE 指令：RFC-959 的第 4.1.3 节

server-FTP 应为非标准的特征使用 SITE 指令，而不是发明新的私有指令或对现有的指令进行非标准的扩充。

4．1．2．9 STOU 指令：RFC-959 的第 4.1.3 节

STOU 指令以唯一的名字存储文件。当 server-FTP 接收到 STOU 指令，它必须在传输开始之前，在“125 传输开始”或“150 打开数据连接”消息中返回一个真实的文件名（RFC-959 中提及的 250 回应码是错误的）。这些消息严格的格式因此被定义如下：

```
125 FILE: pppp
150 FILE: pppp
```

这里，pppp 代表将要存储的唯一文件名。

4．1．2．10 Telnet 行结束码：RFC-959 第 34 页

执行者不能在读取控制连接的分界符和 Telnet 的行结束符序列之间进行任何通信。

讨论：

所以，在处理指令（或回应）之前，server-FTP 必须继续从控制连接读取字符串，直到遇到完整的 Telnet 行结束符序列。相反地，一个从控制连接的单独读取应包含多于一个的 FTP 指令。

4．1．2．11 FTP 回应：RFC-959，第 35 页第 4.2 节

server-FTP 必须在控制连接上发送最合适的正确格式的回应。注意：RFC-959（不同于早期版本的 FTP 规范）没有包含对“自然产生的”回应消息的说明。

server-FTP 在任何时候的回应均应使用 RFC-959 中定义的回应码，不过，只要 4.2 节的规则允许，server-FTP 可以在需要的时候使用不同的回应码。当执行者可以在 4xx 和 5xx 回应码之间选择其一时，如果存在让失败的

FTP 等待一段时间之后仍可执行的任何合理的可能性，server-FTP 应发送 4xx（暂时失败）回应码。

user-FTP 应可仅通过 3 位回应码的最高位就可作出程序的大致决定，这可以防止 server-FTP 使用非标准的回应码时造成的麻烦。

user-FTP 必须可以处理多行的回应。如果执行实现对行数作了限制或超出了限制的行数，user-FTP 必须重新获取回应。比如，忽略多余的行直到收到多行回应的结束符。

user-FTP 不用对 421 回应码（服务不可用，关闭控制连接）作特殊的解释，但是应可检测服务器对控制连接的关闭。

讨论：

服务器的执行实现如果未能严格遵循回应规则，通常会引起 FTP 用户程序中止。注意：RFC-959 对早期创立的 FTP 规范或必须遵循的回应规则的说明是含糊的。

正确地区分暂时性和永久性失败来选择 FTP 回应码是很重要的，它允许文件传输的客户 daemon 程序的成功使用。这些程序依赖回应码来决定是否对失败的传输重新操作，对暂时错误使用永久失败码（5xx）将引起这些程序不必要的放弃。

当回应的含义严格地 RFC-959 中给出的文本时，一字不差地对 RFC-959 文本的引用将增强一致性。不过，在适当的时候鼓励执行者选择回应文本以传达特定的依赖于系统的信息。

4．1．2．12 连接：RFC-959 的第 5.2 节

RFC-959 这节的第二段中的“and the port used”（和使用的端口）是错误的（历史错误），应被忽略。

在多源服务器主机上，缺省的数据传输端口（L-1）必须与相同的本地 IP 地址相关联，因为相应的控制连接在端口 L 上。

user-FTP 不能在 FTP 控制连接上发送除 SYNCH 和 IP 以外的任何 Telnet 控制，特别是不能在控制连接上试图商谈 Telnet 的可选项。不过，server-FTP 必须有允许和拒绝 Telnet 商谈的能力（比如，发送 DONT/WONT）。

讨论：

虽然 RFC 中说：“服务器和用用户进程应遵循 Telnet 协议的规定……[在控制连接上]”，但不意味可以商谈 Telnet 的可选项。

4．1．2．13 最小实现：RFC-959 的第 5.1 节

下列的指令和可选项必须被每一个 server-FTP 和 user-FTP 支持，除非底层的文件系统或操作系统不允许或不支持特殊的指令。

类型：ASCII 非打印，IMAGE，LOCAL 8

模式：流模式

结构：文件，记录*

指令：

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.

*记录结构仅为支持记录结构的文件系统所需要。

讨论：

鼓励厂商支持更大的协议子集。例如，协议中对某些 Internet 用户很有帮助的健壮性特征（比如，Restart，ABOR，块模式）也很重要，但没有被广泛地被实现。

若主机的文件系统没有记录结构，它仍可接受记录结构的文件，逐字地以字节流形式记录。

4．1．3 特殊问题

4．1．3．1 非标准的指令动词

FTP 允许“实验的”指令，它们的名字以“X”开始。如果这些指令以后被作为标准采用，它们仍将在现有的执行中保持使用“X”形式。目前，目录指令适用于此：

RFC-959	“实验的”
MKD	XMKD
RMD	XRMD
PWD	XPWD
CDUP	XCUP
CWD	XCWD

所有的执行都应认可这些指令的两种形式，通过在指令查找表中的额外条目简单地使用它们相等来实现。

执行：

user-FTP 要访问仅支持“X”形式的服务器，可以通过执行一个模式开关实现，也可以使用下面的程序自动实现：如果上面某个指令的 RFC-959 形式被拒绝，并返回 500 或 502 回应码，那就尝试使用实验的形式；任何其他回应被视为允许。

4．1．3．2 超时设定

server-FTP 进程应有超时设定，当服务器很长一段时间没有活动时（比如，进程没有指令或数据传输），终止进程和关闭控制连接。超时设定的时间长度应是可配置的，缺省值至少 5 分钟。

客户 FTP 进程（在 RFC-959 中称为“user-PI”）仅在被一个程序调用时在回应中需要超时设定。

讨论：

如果没有超时设定，在相应的客户端崩溃而没有关闭控制连接时，server-FTP 将无限期地停止不动。

4．1．3．3 数据和控制的并发

讨论：

这对 FTP 开发者意味着用户可在数据传输进行的任一时刻发送 STAT 指令，并且服务器会立即给出状态的回应——例如，到此时已传输的字节数。同样，ABOR 指令在数据传输期间也可能发出。

不幸的是，有些小机器的操作系统使这样的并发程序设计变得困难，还有些执行者追求最小的解决方案，所以这些 FTP 的执行实现不允许数据和控制连接的并发执行。即使一个小服务器也必须准备好允许和延期在数据传输期间到达的 STAR 或 ABOR 指令。

4．1．3．4 FTP 重新开始机制

RFC-959 的第 40-41 页对 110 回应的描述是不正确的，正确的描述如下。重新开始的回应信息，通过控制连接从接收方 FTP 到 user-FTP，格式如下：

110 MARK ssss = rrrr

在此：

- ssss 是一个文本串，给出数据流的重新开始标记和发送方文件系统某位置的编码。
- rrrr 是接收方文件系统相应位置的编码。

编码对特殊的文件系统和网络实现也是特殊的，它总是在相同的系统中被产生和解释，不管是发送方还是接收方。

当一个支持重新开始的 FTP 在数据流中接收到重新开始标记，它应强迫数据转到这个在对应的位置 rrrr 编码之前写入可靠存储中的位置。FTP 发送重新开始标记不能假定 110 回应会与数据同步返回，也就是说，它不能在发送更多数据之前等待 110 回应。

为重新开始传输操作要遭遇的错误定义了两个新回应码：

554 需求的动作没有发生：非法的 REST 参数。

554 回应应该由跟随在 REST 指令后的 FTP 服务指令产生。该回应指出了 server-FTP 中不能被 REST 重新配置的现有文件。

555 需求的动作没有发生：类型或结构不匹配。

555 回应应该由跟随在 REST 指令后的 APPE 指令或任何 FTP 服务指令产生。该回应指出在当前传输参数（类型或结构）和现有文件的属性之间存在不匹配的现象。

讨论：

注意：FTP 重新开始机制要求数据传输采用块或压缩模式，允许重新开始标记包含在数据流中。重新开始标记的频率可以很低。

重新开始标记标记了数据流中的某个位置，但是发送方可能需要在将数据进行可靠的存储时对数据进行转换。通常，接收方的编码必须包含任何需要的信息，以便在 FTP 数据流的任何位置重新开始转换。例如，在类型 A 的传输中，有些发送方主机把 CR LF 序列转换成磁盘中的单一 LF 字符。如果重新开始标记发生在 CR 和 LF 之间，接收方必须在 rrrr 编码，传输必须在“CR 已被发现并丢弃”的状态下重新开始。

注意：重新开始标记需要被编码成可打印的 ASCII 字符串，而不管数据是什么类型。

RFC-959 说重新开始信息被返回“给用户”，但是它并没有真正发生。通常，user-FTP 把重新开始信息（ssss,rrrr）进行可靠的存储，比如，把它添加到重新开始控制文件中。当传输第一次开始时会自动建立一个空的重新开始控制文件，在传输成功完成之后该文件被自动删除。建议这个文件根据正在传输的文件或远程主机的名字取一个容易确认格式的名字，这与许多文本编辑器命名“backup”文件的方法类似。

FTP 重新开始有三种情况：

（1）从用户到服务器传输

user-FTP 在数据流中合适的位置放置重新开始标记<ssss>。当 server-FTP 接收到一个标记，它将此前所有的数据写入磁盘，对它的文件系统位置进行编码，转换成 rrrr 的方式，通过控制连接返回“110 MARK ssss=rrrr”的回应。user-FTP 将这对（ssss,rrrr）添加到重新开始控制文件中。

重新开始传输时，user-FTP 从重新开始控制文件中取出最后一对（ssss,rrrr），重新配置本地文件系统，使用 ssss 的形式进行转换，发送“REST rrrr”指令给 server-FTP。

（2）从服务器到用户传输

server-FTP 在数据流中合适的位置放置重新开始标记<ssss>。当 user-FTP 接收到一个标记，它将此前所有的数据写入磁盘，对它的文件系统位置进行编码，转换成 rrrr 的方式。user-FTP 将这对（ssss,rrrr）添加到重新开始控制文件中。

重新开始传输时，user-FTP 从重新开始控制文件中取出最后一对（ssss,rrrr），重新配置本地文件系统，使用 rrrr 的形式进行转换，发送“REST ssss”指令给 server-FTP。

（3）服务器到服务器（“三方”）传输

发送方 server-FTP 在数据流中合适的位置放置重新开始标记<ssss>。当接收方 server-FTP 接收到一个标记，它将此前所有的数据写入磁盘，对它的文件系统位置进行编码，转换成 rrrr 的方式，通过控制连接返回“110 MARK

ssss=rrrr”的回应给用户。user-FTP 将这对（ssss,rrrr）添加到重新开始控制文件中。

重新开始传输时，user-FTP 从重新开始控制文件中取出最后一对（ssss,rrrr），发送“REST ssss”给发送方 server-FTP，发送“REST rrrr”给接收方 server-FTP。

4.1.4 FTP/用户界面

本节讨论 user-FTP 程序的用户界面。

4.1.4.1 路径名规范

因为 FTP 在多样的环境中使用，user-FTP 的实现必须支持以任意字符串构成的远程路径名，所以它们的形式和内容均不受本地操作系统约定的限制。

讨论：

另外，远程路径名可以是任意长度的，也应允许使用所有的可打印 ASCII 字符和空格（0x20）。RFC-959 允许路径名包含除 CR 和 LF 以外的任何 7 位 ASCII 字符。

4.1.4.2 “QUOTE”指令

user-FTP 程序必须实现“QUOTE”指令，这将可以传送任意字符串给服务器，显示所有作为结果的回应信息给用户。

为了让“QUOTE”指令可用，user-FTP 应在用户进入它们时发送传输控制指令到服务器，而不是存储所有的指令，然后仅在数据传输开始时把它们发送给服务器。

讨论：

“QUOTE”指令实质上允许用户采用需要的特殊系统指令（比如：SITE 或 ALLO）访问服务器，或调用 user-FTP 没有实现的新的或可选的特征。例如，“QUOTE”可以被用来指定“类型 A T”，来发送一个打印文件给主机，甚至 user-FTP 不支持这个类型。

4.1.4.3 显示给用户的回应

user-FTP 应显示给用户它接收到的所有错误回应信息的完整文本。应有一个“详细”模式，显示所有发送的指令和所有接收到的完整文本及回应码，用户问题的诊断。

4.1.4.4 维持同步

为了维持与服务器的指令同步，user-FT 应对丢失的或意外的回应信息有较强的适应能力。

4.1.5 FTP 必要条件一览

特性	章节	MUST	SHOULD	MUST NOT	SHOULD NOT	MAY	Footnote
如果与 TYPE N 相同就执行 TYPE T	4.1.2.2		x				
可能的文件/记录转换	4.1.2.4		x				
User-FTP在流模式下发送PORT指令	4.1.2.5		x				
Server-FTP 实现 PASV	4.1.2.6	x					
PASV 在传输之前	4.1.2.6	x					
NLST回应在RETR指令中可用	4.1.2.7	x					
LIST和NLST的隐含类型	4.1.2.7		x				
可使用非标准特征的SITE指令	4.1.2.8		x				
STOU返回指定的路径名	4.1.2.9	x					
在控制连接上使用TCP READ范围	4.1.2.10					x	
Server-FTP仅发送正确的回应格式	4.1.2.11	x					
Server-FTP在可能的情况下使用定义的回应代码	4.1.2.11		x				
遵从4.2节的 新回应代码	4.1.2.11			x			
User-FTP仅使用回应的高位	4.1.2.11		x				
User-FTP处理多行回应行	4.1.2.11	x					
User-FTP处理特殊的421回应	4.1.2.11				x		
缺省的数据端口与控制连接上的IP地址相同	4.1.2.12	x					
User-FTP发送Telnet指令SYNCH, IP	4.1.2.12					x	
User-FTP可商谈Telnet可选项	4.1.2.12					x	
Server-FTP处理Telnet可选项	4.1.2.12	x					
处理"实验性的"目录指令	4.1.3.1		x				
server-FTP的空闲超时	4.1.3.2		x				
空闲超时可配置	4.1.3.2		x				
在重新开始标记的接收方检查点	4.1.3.4		x				
发送方假定110回应是同步的	4.1.3.4					x	
支持的类型:							
ASCII - 非打印(AN)	4.1.2.13	x					

ASCII - Telnet (AT) --如果与AN相同	4.1.2.2		x			
ASCII – 走纸控制(AC)	959 3.1.1.5.2			x		
EBCDIC - (任意形式)	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
支持的模式:						
流模式	4.1.2.13	x				
块模式	959 3.4.2			x		
支持的结构:						
文件结构	4.1.2.13	x				
记录结构	4.1.2.13	x				3
页结构	4.1.2.3				x	
支持的指令:						
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				
用户界面:						
任意路径名	4.1.4.1	x				
实现 "QUOTE" 指令	4.1.4.2	x				
立即传送控制指令	4.1.4.2		x			
为用户显示错误消息	4.1.4.3		x			
详细模式	4.1.4.3		x			
与服务器保持同步	4.1.4.4		x			

注：

（1）这些标准以前曾给出过。

（2）m 是内存字的位数。

（3）对采用记录结构的文件系统的主机的来说，这个必要条件是可选的

4．2 琐碎文件传输协议——TFTP

4．2．1 介绍

锁碎文件传输协议在 RFC-783[TFTP:1]中定义。

TFTP 用 UDP 为自己提供了可靠的传送方法，以此作为传输协议，使用了一个简单的停止-等待确认系统。因为 TFTP 有一个有效的仅有 512 个八位字节段的窗口，它可以仅通过拥有延时带宽的通路提供良好的性能。如果没有访问控制或安全顾虑的话，TFTP 的文件界面非常简单。

TFTP 的最重要应用是通过局域网络自启动主机，因为它既简单又小巧，所以足够在 EPROM[BOOT:1, BOOT:2]中容易地实现。强烈要求厂商支持 TFTP 用来启动。

4．2．2 协议准备

TFTP 规范[TFTP:1]是以开放的风格写成的，所以它没有完整地详细说明协议为数众多的组成部分。

4．2．2．1 传输模式：RFC-783 第 3 页

“邮件”传输模式不应被支持。

4．2．2．2 UDP 信息头：RFC-783 第 17 页

UDP 信息头的长度域定义得不正确，它包括 UDP 信息头的长度（8）。

4．2．3 特殊问题

4．2．3．1 1　魔术师学徒综合症

在协议规范中，有一个严重的错误，就是已广为人知的“魔术师学徒综合症”。它不会引起传输的错误运行（如果传输可以完成，文件总会被正确地传输），这个错误会引起过多的重传，这将导致传输超时。

具体实现必须包含对此问题的修正：发送者（也就是创建数据包的一方）必须在收到重复的 ACK 时不再重传当前的数据包。

讨论：

该错误是由任意一方的协议规则引起的，在接收一个旧重复数据包时，可能重传当前数据包。如果网络中的一个信息包延迟，并稍后成功地到达，但此时某一方已因超时并重传数据包，那么将产生一个回应的复本。如果另一方重复地对此作出响应，那么剩余的数据将被重传（除非数据包丢失而导致中断）。更糟糕的是，因为延迟通常由网络堵塞引起，而重发通常会导致更严重的堵塞，由此引发更多的延迟信息包，等等。

下例将帮助我们阐明这个问题。

TFTP A　　TFTP B

(1)　Receive ACK X-1

```
     Send DATA X
(2)                          Receive DATA X
                             Send ACK X
        (ACK X 在网络中延时，并且A超时)
(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
```

注意一旦延迟的 ACK 到达，协议将安心地重复所有另外的信息包（序列 5-8 和 9-12）。该问题并不是由一方的超时引起，而是由于当双方接收到复本后重传了当前的数据包造成的。

纠正错误的方法是如前面所指出的中断重传循环，这与 TCP 的行为类似。接收方可能删除重传计时器，因为重发的 ACK 将不再引起任何操作，这将有助于当 TFTP 应用在自启动程序时简化操作。允许保持计时器也可以，如果重传的 ACK 确实在网络中丢失了，它将有所帮助，当然，此时发送方仍需一个重传计时器。

4．2．3．2 超时规则

TFTP 必须使用自适应的超时规则。

执行：

TCP重传规则提供了一个有用的工作基础。至少需要一个重传指数补偿的超时规则。

4．2．3．3 扩展

非标准的多种扩展已经被应用到 TFTP，包括另外的传输模式和一个安全操作模式（使用口令）。但它们中还没有一个被标准化。

4．2．3．4 访问控制

一个服务器的 TFTP 具体实现应该包括一些可配置的访问控制。

4．2．3．5 广播请求

一个被发送到广播地址的 TFTP 请求应被静静地忽略。

讨论：

由于 TFTP 脆弱的访问控制能力，TFTP 请求被广播到任意的网络将导致一个严重的安全漏洞。

4．2．4 TFTP 必要条件一览

特性	章节				SHOULD	MUST	Footnote
修正“魔术师学徒综合症”	4.2.3.1	x					
传输模式:							
netascii	RFC-783	x					
octet	RFC-783	x					
mail	4.2.2.1				x		
extensions	4.2.3.3			x			
使用自适应的超时规则	4.2.3.2	x					
访问控制的可配置能力	4.2.3.4		x				
静静地忽略广播请求	4.2.3.5		x				

5．电子邮件——SMTP 和 RFC-822

5．1 介绍

在 TCP/IP 协议族中，电子邮件的格式规范在 RFC-822[SMTP:2]中定义，它用 RFC-821[SMTP:1]中定义的简单邮件传输协议（SMTP）来传送。

SMTP 协议已经多年未变，在 Internet 社区中已经有几个经过改变的 SMTP 在应用。特别是，对域名系统（DNS）的转换已经引起地址格式和邮件路由的变化。本节中，我们假定已精通 DNS 的概念和技术，它的必要条件在 6.1 节给出。

RFC-822 详细说明了电子邮件消息的 Internet 标准格式。 RFC-822 取代了一个旧标准，RFC-733，虽然它已被废除，但仍在一些地方使用。这两种格式有时简单地用数字提及（“822”和“733”）。

RFC-822 被应用在拥有不同于 SMTP 的电子传输协议的一些非 Internet 邮件环境中，并且 SMTP 也适用于一些非 Internet 环境。本文介绍的使用 SMTP 和 RFC-822 的规则仅限于 Internet 环境，其他使用不同协议的邮件环境应定义他们自己的规则。

5．2 协议准备

本节内容涉及 RFC-821 和 RFC-822。

RFC-821 定义的 SMTP 协议清楚地包含了众多示例，所以执行者发现它很容易理解。本节对它作了简单更新和评注，以使它可适应当前应用。

RFC-822 是一个冗长的文档，定义了丰富的语法。不幸的是，对 RFC-822 的不完全或不良执行是常见的现象。事实上，实际应用中，有多种不同的 RFC-822 实现方式，所以具体实现通常需要接受和正确地解释所有 RFC-822 语法。

5．2．1 SMTP 模型：RFC-821 第 2 节

讨论：

邮件在客户端（SMTP“发送者”）和服务器（SMTP“接收者”）之间通过一系列的请求/回复处理实现传送的目的。传送包括（1）正确的消息，由消息头和消息体组成，（2）作为外壳的 SMTP 起始和目的地址。

SMTP 程序与 X.400 的消息传送代理（MTAs）类似。它是协议软件的另一层，与终端用户更接近，有责任构成和分析 RFC-822 消息头，这部分就像已知的 X.400 中的“用户代理”，本文中也使用这个术语。在用户代理和 SMTP 执行之间有明确的逻辑区别，因为它们处理协议的不同层。注意，这个区别并不会严格地反映 Internet 邮件典型执行的结构。通常，具有如已知的“mailer”程序一样的程序执行 SMTP 和用户代理的部分功能，其余的用户代理功能包含在一个用于进入和阅读邮件的用户界面中。

SMTP 外壳由发送的站点构造，典型的是消息第一次在 SMTP 发送者程序中排队时由用户代理构造。外壳地址可能是来自于消息头，由用户界面提供（比如，执行一个 bcc 请求）或来自于本地构造信息（比如，邮件列表的

扩展）。SMTP 通常不会在消息交付的以后阶段再次包含在消息头中，所以，外壳通过使用 SMTP 的 MAIL 和 RCPT 指令从它自己的消息中分离出来进行传送。

RFC-821 假定邮件被交付到主机的一个单独用户。随着域名系统的出现和邮件路由使用邮件交换（MX）资源记录，执行者现在应考虑交付邮件给某个域的用户，它可能是（或不是）一个特殊的主机。但这不会改变 SMTP 仍是一个主机到主机的邮件交换协议的事实。

5．2．2 标准化：RFC-821 第 3.1 节

SMTP 发送者在 MAIL 和 RCPT 指令中的域名必须被“标准化”，也就是说，它们必须是完全合格的首要名字或域字母，而不是昵称或域缩写。一个标准化的名字可以直接地识别一台主机或是一个 MX 名字，但不能是 CNAME。

5．2．3 VRFY 和 EXPN 指令：RFC-821 第 3.3 节

SMTP 发送者必须实现 VRFY，也应实现 EXPN（这个需求已超过 RFC-821）。不过，在特殊的安装中，配置信息会使 VRFY 和 EXPN 无效，甚至 EXPN 对选择列表也是无效的。

为 VRFY 指令定义了一个新的回应码：

252 不是 VRFY 用户（比如，信息不是本地的），但将把消息传送给用户并尝试交付。

讨论：

SMTP 用户和管理员已经习惯于用这些指令来诊断邮件交付问题。随着多级邮件列表扩展（有时多于两级）应用的增多，使用 EXPN 来诊断因疏忽造成的邮件循环已经变得越来越重要。另一方面，有些人认为 EXPN 泄露了个人隐私，可能甚至带来安全问题。

5．2．4 SEND，SOML 和 SAML 指令：RFC-821 第 3.4 节

SMTP 应实现发送消息到用户终端的指令：SEND，SOML 和 SAML。

讨论：

已经提到过，邮件转发 MX 记录，与直接立即地交付消息到用户终端的意图是不一致的。不过，不能直接发送到用户终端的 SMTP 接收者，应对跟随在 SEND 后面的 RCPT 指令返回一个“251 用户非本地”的回应，来告知创建者交付可能延期。

5．2．5 HELO 指令：RFC-821 第 3.5 节

SMTP 发送者必须保证 HELO 指令中的<domain>参数对客户端主机来说是有效的首要域名。其结果是，SMTP 的接收者不必为了使 HELO 参数生效而对该名字进行 MX 转换。

HELO 的接收者应对 HELO 参数进行校验，以确保其与发送者的 IP 地址一致。不过，接收者不能拒绝接收消息，即使发送者的 HELO 指令校验失败。

讨论：

校验 HELO 参数需要对域名进行查找，这会耽误一些时间。对跟踪伪造邮件来源的问题，下面（参看“DATA 指令”）提出了一个可选的工具。

还应注意，HELO 的参数仍需要有效的<domain>语法，因为它被视为一个 Received:行；否则，将发送一个 501 错误。

执行：

当 HELO 参数确认失败，建议插入一个关于发送者的未知真实性的注释到消息头中（比如，在“Received:”行中）。

5．2．6 邮件转发：RFC-821 第 3.6 节

我们区别三种邮件（存储和）转发：

（1）使用关于接收者的私有消息的一个简单转发程序或“mail exchanger”转发消息。参看 RFC-821 第 3.2 节。

（2）在 SMTP 邮件环境中由于明确的来源路径（如 RFC-821 第 3.6 节中定义的那样）的 SMTP 邮件“转发”消息。SMTP 转发功能使用遵照 RFC-822 的来源路由的“@…:”格式（参看下面的第 5.2.19 节）。

（3）一个邮件“网关”在两个不同的环境下传递消息。邮件网关的规则在下面第 5.3.7 节讨论。

一个转发邮件但不是不同邮件环境网关的 Internet 主机（也就是说，被归入（1）或（2））不应该改变任何已存在的消息头域，虽然主机会根据第 5.2.8 节的要求添加一个适当的 Received:行。

SMTP 发送方不应发送用“@…:”地址形式包含清楚地表示来源路由的 RCPT TO:指令。所以，转发功能不能使用 RFC-821 第 3.6 节的定义。

讨论：

邮件交付的意图是在 Internet 环境下阻挡所有的来源路由和破坏明确的来源路径。来源路由不是必需的，得意的目标地址“user@domain”就可满足。这是由于清楚的结构决定了使用全体名字而不是邮件的来源路由。所以，SMTP 提供端到端的连通性，DNS 提供全球的唯一的独立的定位名。MX 记录处理当来源路由有其他需要时的情况。

SMTP 接收方必须接受信封中的明确的来源路径语法，但是它可能执行 RFC-821 第 3.6 节定义的转发功能。如果它不能执行转发功能，它将尝试直接将消息传递给最右边“@”符号的右边的主机。

讨论：

例如，假定一个没有实现转发功能的主机接收到一个SMTP指令：“RCPT TO:<@ALPHA,@BETA:joe@GAMMA>”,这里，ALPHA, BETA 和 GAMMA表示域名。不是根据RFC-821第20页建议的使用一个550错误回应立即拒绝该消息，而是尝试用“RCPT TO:<joe@GAMMA>”直接转发消息到GAMMA。因为主机不支持转发，它无需更新反向路径。

有的建议是有时需要手动路由以绕过失败，不过，这个需要的重要性是有争议的。为此目的使用清楚的SMTP邮件路由是被阻止的，并且事实上它也不成功，因此许多主机系统并不支持它。有些为此目的使用了“%-hack”

（参看第5.2.16节）。

5．2．7 RCPT 指令：RFC-821 第 4.1.1 节

一个支持 SMTP 接收方的主机必须支持保留邮箱“Postmaster”。

SMTP 接收方可以在 RCPT 指令到达时对其参数进行校验，不过，RCPT 回应的延迟不能超过合理的时间（参看第 5.3.2 节）。

所以，一个 RCPT 的“250 OK”回应并不必然代表交付的地址是有效的。在消息接受之后发现的错误将用一个邮件携带通知消息报告给适当的地址（参看第 5.3.3 节）。

讨论：

RCPT 的参数条件设置是否会立即生效是一个工程方面的设计选择。在邮件被交付之前报告目的地邮箱错误给 SMTP 发送者，对节省时间和网络带宽是有益的，但是如果 RCPT 验证时间过长，这些优势就没有了。

例如，接收者可以立即核实本地任何简单的本地的参考资料，比如专门的本地已注册邮箱的信息。但另一方面，对直到消息被交付和接受之后进行的邮件列表的验证，通常意味着“合理时间”的限制会被拖延，因为验证一个巨大的邮件列表将花费非常长的时间。具体实现可以（或不可以）选择推迟对非本地地址的确认，并且还因此需要 DNS 查询。如果 DNS 查询被执行，但发生了软件域系统错误（比如，超时），则必须假定其有效。

5．2．8 DATA 指令：RFC-821 第 4.1.1 节

每个 SMTP 接收方（不仅是“接受作为转发的或作为最终交付的消息”[SMTP:1]）必须在消息的开头插入一个“Received:”行。这个在 RFC-821 中被称为“时间邮戳行”的行中：

- FROM 域应该包含两项：（1）与 HELO 指令中给出的相同的来源主机的名字；（2）一个包含来源主机 IP 地址的域文字，它由 TCP 连接决定。
- ID 域可以包含 RFC-822 中建议的象“@”这样的字符，但不是必需的。
- 当给出多个 RCPT 指令时，FOR 域可以包含一个<path>条目的列表。

Internet 邮件程序不能改变先前添加到消息头中的 Received:行。

讨论：

在 Received:行中包含来源主机和 IP 来源地址，将为跟踪非法的邮件来源和消校验 HELO 参数的需要提供足够的信息。

Received:行最初的意图是为了人类用户可以跟踪邮件路由，最初是用来进行故障诊断的。参看下面第 5.3.7 节中的讨论。

当 SMTP 接收方使消息可以“最终交付”时，它必须在消息的 SMTP 信封中传递 MAIL FROM: 地址，为了假如有一个必须在以后被发送错误通知时使用（参看第 5.3.3 节）。当从 Internet 到不同的邮件环境中间有网关时，有一个类似的需求，参看第 5.3.7 节。

讨论：

注意，对 DATA 指令的最后回应仅依赖于消息的成功传输的存储。任何与目的地址有关的问题必须或者（1）在对 RCPT 指令的回应中作 SMTP 错误的报告；或者（2）在稍后给发送方发送一个错误消息邮件。

执行：

MAIL FROM:信息可以作为一个参数或嵌入在消息开头的 Return-Path:行中来传递。

5．2．9 指令语法：RFC-821 第 4.1.2 节

RFC-821 给出的 MAIL FROM:指令的语法忽略了一种空路径：“MAIL FROM:<>”（参看 RFC-821 第 15 页）的情况。空的反向路径必须被支持。

5．2．10 SMTP 回应：RFC-821 第 4.2 节

SMTP 接收方应该仅发送 RFC-821 的第 4.2.2 节中或本文中列出的回应码。SMTP 接收方应该使用在 RFC-821 的示例中给出的适当的文本。

SMTP 接收方必须仅由回应码而不是由回应文本（除了 251 和 551 回应以外）来决定它的动作。任何文本，包括根本没有内容的文本，必须被接受。跟随在回应码后的空格被看成是文本的一部分。只要可能，SMTP 发送方应该仅检测回应码的第一个数字，它们在 RFC-821 的附录 E 中进行了详细说明。

讨论：

当 SMTP 系统使用了未在 RFC-821 第 4.3 节中明确列出的回应码，但它却合法地遵照附录 E 中的回应码说明原理，此时，就会产生交互问题。

5．2．11 透明性：RFC-821 第 4.5.2 节

执行者必须确定他们的邮件系统总能添加和删除句号，以确保消息透明性。

5．2．12 在 MX 处理中使用 WKS：RFC-974 第 5 页

RFC-974 [SMTP:3]建议域服务器可以被查询 WKS（″Well-Known Service″，已知服务）记录，来检验每一个被提名的邮件目标支持 SMTP。后来的经验显示 WKS 未被广泛支持，所以不应让 WKS 介入到 MX 处理中。

下列在 RFC-822 中被注释的信息，由该文档的相应章节组织。

5．2．13 RFC-822 消息规范：RFC-822 第 4 节

过去的Return-path:行的语法忽略了空的返回路径的可能性，它用来防止错误通告的循环（参看第5.3.3节）。完整的语法是：

```
return = ″Return-path″  ″:″ route-addr
     / ″Return-path″  ″:″ ″<″ ″>″
```

可选的头域的集因此扩展到可以包含 RFC-1049[SMTP:7]中定义的Content-Type（目录类型）域。这个域“允许邮件读取系统来自动确定构造消息体和处理其显示的类型”。[SMTP:7]用代理可以支持此类型。

5．2．14 RFC-822 日期和时间规范：RFC-822 第 5 节

日期的语法因此变为：

```
date = 1*2DIGIT month 2*4DIGIT
```

所有的邮件软件应使用 4 位数字表示日期中的年份，这便于到下个世纪的转换。

有一个很强的趋势是数字时间区域指示器的使用，具体实现应使用数字的时间区域代替时间区域名。不过，所有的具体实现必须接受这两种符号表示法。如果使用了时间区域名，它们必须遵循 RFC-822 正确地定义。

RFC-822 中的军用时间区域定义得不正确：由于 UT 使用了错误的方式来计算（符号被颠倒了）。结果是，RFC-822 中的军用时间区域头中没有携带信息。

最后要注意，在附录 D 的语法一览中对“区域”的定义有一个印刷错误；正确的定义出现在 RFC-822 的第 3 节中。

5．2．15 RFC-822 语法变化：RFC-822 第 6.1 节

根据 RFC-822 中句法定义的“mailbox”（邮箱）在此变为：

```
mailbox =  addr-spec             ; simple address
        / [phrase] route-addr    ; name & addr-spec
```

这里，路由地址之前的短语现在是可选的。这个变化使得下列头域合法，例如：

```
From: <craig@nnsc.nsf.net>
```

5．2．16 RFC-822本地部分：RFC-822第6.2节

基本的信箱地址规范形式是：“local-part@domain”。这里“local-part”有时称为地址的“左边”，它依赖于域。

转发邮件但不是最终目的地的主机意味着右边的“domain”不能解释或修改地址的“本地部分”。

当邮件通过网关从Internet邮件环境进入到外部邮件环境时（参看第5.3.7节），外部环境的路由信息可以被嵌入到地址的“本地部分”中去。网关将根据外部邮件环境适当地解释本地部分。

讨论：

虽然来源路由在Internet中是被阻止的（参看第5.2.6节），但有的非

Internet邮件环境的交付机制依赖于来源路由。当邮件通过Internet时，Internet以外环境的来源路由通常可以被隐藏在地址的“本地部分”中（参看第5.2.16节）。当邮件到达合适的Internet邮件网关时，网关将解释本地部分，并为目标邮件环境创建需要的地址或路由。

例如，一台Internet主机可以发送邮件到“a!b!c!user@gateway-domain”。这个复杂的本地部分“a!b!c!user”在Internet域中不会被解释，但可以在特定的邮件网关中被解析和理解。

一个内嵌的来源路由有时会被编码到“本地部分”中，这时使用“%”作为右侧绑定路由操作符。例如，

```
user%domain%relay3%relay2@relay1
```

在上句中，“%”约定意味着邮件被从“relay1”发送，途经“relay2”，“relay3”，最后到达在“domain”的“user”。通常已知的如“%-hack”。建议“%”比其他隐藏在本地部分中的路由操作符（比如，“!”）的优先级低；例如，“a!b%c”将被解释为“(a!b)%c”。

仅有目标主机（此时是“relay1”）被准许分析本地部分“user%domain%relay3%relay2”。

5．2．17 域文字：RFC-822 第 6.2.3 节

邮件程序必须可以接受并解析由点分隔的十进制组成的主机地址的 Internet 域文字（“dtext”，参看 RFC-822）。这就满足了 2.1 节中对邮件场合下的必要条件。

SMTP 必须接受并识别出任何其自己的 IP 地址的域文字。

5．2．18 常见的地址格式化错误：RFC-822 第 6.1 节

遗憾的是，地址格式化或解析错误是常见的。本节仅提到最常见的错误。用户代理必须接受所有合法的 RFC-822 地址格式，并不能产生非法的地址语法。

●一个常见的错误是省略了一群标识符之后的分号。

●有些系统在它们产生的消息中未能提供完全符合规则的域名。头部地址“@”符号的右边域必须是完全符合规则的域名。

例如，有些系统未能提供完全符合规则的 From:地址，这妨碍了在来自于自动构造的返回地址的用户界面中的“reply”指令。

讨论：

虽然 RFC-822 允许在一个域中局部地使用域名的缩写，但是 RFC-822 在 Internet 邮件的应用中不允许这样。其目的是 Internet 主机不能发送在地址域中包含域名缩写的 SMTP 消息头。这允许头部的地址域在 Internet 中被解析时无需变更，正如第 5.2.6 节要求的那样。

●有些系统错误地解析多次跳跃的明确的来源路径，例如：

@relay1,@relay2,@relay3:user@domain。

●有些系统在地址中或消息标识中的符合规则的域名上添加了一个跟踪圆点。这违反了 RFC-822 的语法。

5．2．19 明确的来源路径：RFC-822 第 6.2.7 节

Internet 主机软件不应该建立包含有明确的来源路径地址的 RFC-822 头部，但是必须接受这样的头部，以兼容早期的系统。

讨论：

在一个保留的陈述中，RFC-822 说“阻止明确的来源路径的应用”。许多主机不正确地执行了 RFC-822 的来源路由，因此语法在实践中不能被含糊地使用。许多用户觉得语法令人讨厌。明确的来源路径在交付的邮件环境中是不需要的，参看第 5.2.6 节。因为所有这些原因，使用 RFC-822 符号的明确的来源路径不能在 Internet 邮件头部中使用。

在第 5.2.16 节的规定中，需要允许一个明确的来源路径被隐藏在地址的本地部分，比如，使用“%-hack”，目的是为了允许邮件可以通过路由到达需要明确的来源路径的其他环境中。应该警惕，当目的地在 Internet 中时，用户代理无法察觉和防止这样隐含的来源路由的使用。我们只能在 Internet 中阻止任何类型的来源路径，因为它们是不需要的和多余的。

5．3 特殊问题

5．3．1 SMTP 队列策略

一个具体实现 SMTP 的主机一般的结构包括：用户邮箱，为传输的消息设置的一个或多个队列区，一个或多个为发送和接收邮件而设置的 deamon 进程。确切的结构依赖于主机上用户的需求和主机所能支持的邮件列表的数量和大小而变化。我们描述了几个最优化方案，它们已被证明对支持高传输级别的邮件程序有显著的帮助。

任何队列策略必须包括：

●涉及所有活动的超时机制。

●从不会对错误的消息发送包含错误消息的回应。

5．3．1．1 发送策略

SMTP 发送方的一般模型是一个或多个周期性地尝试发送外发邮件的进程。在一个典型的系统中，调解消息的程序有一些方法来请求立即关注一篇新的外发邮件。当邮件不能被立即传输时，必须由发送方对其进行排队并周期性地重发。邮件队列的条目将不仅包含消息本身，而且还包括信封信息。

在一次尝试失败以后，发送方必须延迟，再重发到特定的目的地。通常，重试的时间间隔至少为 30 分钟；不过，当 SMTP 发送方可以测定未交付的原因时，更成熟和多变的策略会有益处。

重试会一直持续，直到消息被传输或发送方放弃；放弃的时间通常至少需要 4-5 天。重试算法的参数必须是可配置的。

发送方应该保留一个不能到达和通讯超时的主机列表，而不是仅仅重试排队的邮件条目。

讨论：

经验建议失败通常是暂时现象（目标系统已崩溃），赞成这样一种策略，即两个连接在第一个小时尝试队列中的消息，以后减少到每次两到三个小时。

SMTP 发送方可以通过与 SMTP 接收方合作来缩短队列的延迟时间。特别是，如果邮件从一个指定的地址接收到，很明显，任何该主机的邮件队列现都可以被发送了。

策略可以被进一步修改，对于每个主机有多个地址的结果（参看第 5.3.4 节），来优化交付的时间和资源的利用。

SMTP 发送方可以为每一个无效目的地主机设置一个巨大的消息队列，在每一次重试循环中，如果重试所有的消息，将消耗过多的 Internet 资源，并且 deamon 将被长期阻塞。注意，SMTP 通常仅可以测定几分钟超时后的一个失败的交付尝试。如果对一打甚至数百个排队的消息，每个连接一分钟的超时，都将导致巨大的延迟。

当相同的消息被交付到同一机的多个用户时，仅应传输一个消息的复本。也就是，SMTP 发关方应该使用指令序列：RCPT, RCPT,... RCPT, DATA 来代替序列：RCPT, DATA, RCPT, DATA,... RCPT, DATA。强烈推荐使用这种高效的具体实现。

类似地，SMTP 发送方可以支持多个同时外发邮件的处理，以完成及时的交付。不过，由于使用了所有的资源来处理邮件，应该为保护主机强加一些限制。

多源主机的多个不同地址的应用将在下面讨论。

5．3．1．2 接收策略

SMTP 接收方应该努力保持在所有时间对 SMTP 端口的监听。这将需要为 SMTP 引入多重 TCP 连接的支持。可以强加一些限制。

执行：

当 SMTP 接收方从一个特定的主机地址接收到邮件时，它可以通知 SMTP 发送方重试所有关于这个主机址的未完成邮件。

5．3．2 SMTP 的超时设定

对于 SMTP 发送方的超时设定有两种方案：（a）为每个 SMTP 指令分别地限制时间；或（b）为单独邮件消息的所有 SMTP 对话限制时间。SMTP 发送方应该使用（a），对每个指令进行超时设定。超时设定应该可以被容易地重新配置。

讨论：

超时设定是 SMTP 具体实现的一个基本特性。如果超时设定的时间太长（或者更糟地，没有超时设定），Internet 通信失败或 SMTP 程序中的软件错

误将不确定地阻碍 SMTP 进程。如果超时设定的时间太短，资源将因频繁的重试而被浪费。

如果使用（b）方案，超时设定可以非常巨大，比如，一个小时，以允许时间适应非常大的邮件列表。超时设定也会因为消息的大小而直线地增大，来解决传输一个巨大消息所需要的时间。一个巨大的固定的超时设定将导致两个问题：一个失败可能仍会阻碍发送方很长一段时间，并且非常大的消息可能仍假装超时（这是一个奢侈的失败！）。

使用推荐的方案（a），将为每个 SMTP 指令设置计时器，并为每个数据传输设置缓冲区。上句话后面的意思是全部的超时设定与消息的大小固定成比例。

基于对忙碌的邮件转发主机的丰富经验，每条指令最小的超时设定值应该如下：

●最初的 220 消息：5 分钟

SMTP 发送方进程需要区别失败的 TCP 连接和接收到最初的 220 欢迎消息的延迟。许多 SMTP 接收方接受 TC 连接，但是要延迟 220 消息的交付，直到它们的系统负荷允许更多的邮件被处理。

●MAIL 指令：5 分钟

●RCPT 指令：5 分钟

如果邮件列表或别名的处理未被延期，直到消息被接受之后，这就需要一个更长的超时设定。

●数据初始化：2 分钟

这是等候对 DATA 指令的“354 开始输入”回应的时间。

●数据块：3 分钟

这是每个 TCP SEND 调用传输一大块数据完成所需等候的时间。

●DATA 终止：10 分钟

这是等候“250 OK”回应的时间。当接收方获得使消息数据终结的最后周期时，它应该执行处理以交付消息到用户邮箱。此时一个不合逻辑的超时设定是非常浪费的，因为消息已被成功发送。

SMTP 接收方应该有至少 5 分钟的超时设定，来等待发送方的下一条指令。

5．3．3 可靠的邮件收条

当 SMTP 接收方接受了一个邮件（通过为 DATA 指令的回应中发送“250 OK”消息），它也接受了交付和转发邮件的责任。它必须认真履行这个责任，也就是说，不能因为琐碎的原因丢失消息，比如，因为主机稍后的清洗或可预知的资源不足。

如果在接受消息后交付失败，SMTP 发送方必须将原因明确表达，并在一个包含通知消息的邮件中传送。这个通知必须在信封中使用一个空的（“<>”）反向路径来发送，参看 RFC-821 的第 3.6 节。通知的接收者应是信封反向路径（或 Return-Path:行）的地址。不过，如果地址是空的

（“<>”），SMTP 接收方就不能发送这个通知了。如果地址是一个明确的来源路径，它应该被剥离到它的最后一次跳跃。

讨论：

例如，如果到达时是“MAIL FROM:<@a,@b:user@d>”，假定一个错误通知必须被发送，那么通知消息应被发送：“RCPT TO:<user@d>”。

SMTP 中一些消息被接受后的交付失败是不可避免的。例如，由于“软件”的错误，SMTP 接收方不可能使所有的 RCPT 指令中的交付地址都有效，也由于目标是一个邮件列表（参看早期关于 RCPT 的讨论）。

为了避免重复地接收消息，SMTP 接收方必须搜索作为消息传输结尾的“.”，以最小化回应的时间。参看 RFC-1047 [SMTP:4]获取关于本问题的讨论。

5．3．4 可靠的邮件传输

为了传输消息，SMTP 发送方从信封中的目的地地址中测定目标主机的 IP 地址。特别是，它将“@”符号右边的字符串映射为 IP 地址。该映射或它自身的传输可能因软件错误而失败，这个软件错误导致 SMTP 发送方对外发邮件进行重新排队以便稍后重发，如第 5.3.1.1 节要求的那样。

当成功时，映射可能导致一个供选择的交付地址列表而不是一个单一的地址，因为（a）多个 MX 记录，（b）多源的，或同时因为（a）和（b）。为了提供可靠的邮件传输，SMTP 必须能尝试（和重试）这个有序列表中的每一个地址，直到交付成功。不过，也可以对尝试的交付地址的数量进行可配置的限制。在任何情况下，主机应该尝试至少两个地址。

下列信息用来对主机地址进行排队：

（1）多个 MX 记录——它们包含了将要被分类的优先级。如果一个相同的优先级有多个目的地并且没有明确的原因支持其中的某个地址（比如，按地址优先级排序）那么 SMTP 发送方会随机地选取一个，通过负载到多个邮件交换来传送。注意，这是[DNS:3]中程序的明确表达。

（2）多源主机——目的地主机（可能降低 MX 记录的优先级）可以是多源的，这种情况下，域名解决程序将返回一个可选择的 IP 地址列表。域名解决程序界面有责任将列表按降序排列，SMTP 必须按给出的次序尝试它们。

讨论：

虽然尝试多个可选地址的能力是必需的，但是在特殊的安装环境中可能对可选择地址进行限制甚至禁用。是否应重试一个多源主机的多个地址已经引起争论。争论的焦点是为了及时交付可以重试的多个地址的最大数量。这个参数可能会导致不必要的资源消耗。

注意，由于在 5.3.1 中讨论的发送策略，资源消耗也起着有力的作用。

5．3．5 域名支持

SMTP 的具体实必须使用 6.1 节中定义的机制进行域名和 IP 地址之间的映射。这个所有 Internet SMTP 必须包含的方法是为了支持 Internet DNS。

特别是，SMTP 发送方必须支持 MX 记录方案[SMTP:3]。也可参看[DNS:2]的第 7.4 节获取关于 SMTP 域名支持的信息。

5．3．6 邮件列表和别名

为了多次交付，有 SMTP 能力的主机应该支持地址扩展的别名和列表形式。当消息被交付或转发到扩展列表形式的每个地址时，信封中的寄信人地址（“MAIL FROM:”）必须变为列表管理员的个人地址，但是消息头的左边不能改变，特别是，消息的“From”域不能改变。

一个重要的邮件工具，是通过把伪邮箱地址改变或“扩展”为目的邮箱地址的列表，将把单一的消息交付到多个目的地的机制。当消息被发送到伪邮箱时（有时称为“exploder”），复本会被转发或重新分配到扩展列表的每一个邮箱中去。我们根据扩展有规则，将这样的伪邮箱分为“别名”或“列表”。

（a）别名

为了扩展别名，接收方邮件程序简单地将信封中的伪邮箱地址用每一个扩展的地址轮流替换，信封的其余部分和消息体没有变化。消息然后被交付或转发到每一个扩展的邮箱。

（b）列表

邮箱列表可以被称为由“重新分配”而不是由“转发”来操作。为了扩展列表，接收者邮件程序将信封中的伪邮箱地址用每个扩展地址轮流替换。信封中的返回地址被改变，所以在最终交付时产生的所有错误消息将被返回到列表管理员那里，而水是邮件的创建者。管理员通常不会控制列表的内容，并将查找令人讨厌的错误消息。

5．3．7 邮件网关

网关将邮件在不同的邮件环境之间传递，也就是说，它们拥有不同的邮件格式和协议，复杂但不容易被标准化。参看示例[SMTP:5a],[SMTP:5b]。不过，有些一般需求可以通过在 Internet 和其他邮件环境之间的网关实现。

（A）当消息通过网关在不同的邮件环境中穿行时，头域可以被重新写入。

讨论：

它可以包括目的地址的本的部分的解释，如第 5.2.16 节中建议的那样。

其他通过网关进入Internet的邮件系统通常使用RFC-822头的子集，但有些并不等价于SMTP信封。所以，当消息离开Internet环境，它可能需要将SMTP环境信息合并到消息头中去。一个可能的解决方案是创建一个新的头域来携带环境信息（比如，″X-SMTP-MAIL:″ 和 ″X-SMTP-RCPT:″）；不过，这将需要改变不同环境下的邮件程序。

（B）当转发一个消息到Internet以外的环境时，网关必须预先准备好Received:行，但不能改变任何已经存在于头中的Received:行。

讨论：

该需求是第5.2.8节对一般“Received:”行要求的子集，在此强调重申。

由其他环境建立的消息Received:域可能不严格符合RFC-822的规范。不过大多数重要的Received:行的应用已进行了邮件除错处理。

强烈建议在Received域提供的“via”条目中指出网关的环境和协议。

（C）对Intenet方面，网关应该接受SMTP指令中和RFC-822头中所有合法的地址格式，和所有合法的RFC-822消息。虽然网关必须接受在RFC-822头中或信封中的RFC-822明确的来源路径（形式为“@…:”），但它可以（或不可以）对来源路径起作用，参看第5.2.6节和5.2.19节。

讨论：

经常冒着危险去限制可接受的邮件网关的地址范围，以使到过程环境的地址转换简单化。这个实践基于用户可以控制发送到邮件路网关的地址的假设。不过，实际上，用户仅对最终发送的地址有很小的控制力，它们的邮件程序可以从容地转换地址到任何合法的RFC-822格式。

（D）网关必须确保所有的到Internet的消息的头域适合Internet 邮件的必要条件。特别是，在“From”，“To”，“Cc:”等域中的所有的地址必须被转换（如果需要）到符合RFC-822标准的语法，并且它们必须对发送回应来说是有效的和有用的。

（E）用来转换邮件从 Internet 协议到另一个环境协议的转换算法，应该试着确保从外界的邮件环境来的错误消息可以被传递到 SMTP 信封中的返回路径中，而不是发送者在 RFC-822 消息的“From:”域中列出的。

（F）同样地，当从另外的环境转发一个消息到 Internet 中时，网关应该进行设置，以使信封的返回路径与错误消息的返回路径保持一致，即便要为不同的环境提供。

5．3．8 消息的最大尺寸

邮件程序必须可以发送和接收至少长度为 64K 字节的消息（包括头信息），非常期望支持的最大尺寸可以很大。

讨论：

虽然 SMTP 没有定义消息的最大尺寸，但是，因为电子邮件已应用于多种目的，以致产生了非常大的消息。例如，邮件通常用来代替 FTP 来传输 ASCII 文件，特别是传输完整的文档。其结果是消息会有 1M 字节甚至更大。我们注意到现在的文档，和它的更底层伴随文档都可容纳 0.5M 字节。

5．4 SMTP 必要条件一览

特性	章节	MUST	SHOULD	MAY	SHOULD NOT	MUST NOT	Footnote
SMTP发送方:							
实现VRFY	5.2.3	x					
实现EXPN	5.2.3		x				
EXPN, VRFY可配置	5.2.3			x			
实现SEND, SOML, SAML	5.2.4			x			
校验HELO的参数	5.2.5			x			
拒绝不正确的HELO消息	5.2.5					x	
接受明确的来源路径语法	5.2.6	x					
支持“postmaster”	5.2.7	x					
当接收到RCPT时进行处理（除了列表）	5.2.7			x			
RCPT回应长时间延迟	5.2.7					x	
添加Received:行	5.2.8	x					
Received: 行包括域文字	5.2.8		x				
改变前面的Received:行	5.2.8					x	
传递返回路径信息	5.2.8	x					
支持空的反向路径	5.2.9	x					
仅发送正式的回应代码	5.2.10		x				
适当时发送来自于RFC-821的文本	5.2.10		x				
为透明性删除“.”	5.2.11	x					
接受并识别自身域文字	5.2.17	x					
关于错误消息的错误消息	5.3.1					x	
保持对SMTP对未决定的监听	5.3.1.2		x				
提供并发限制	5.3.1.2			x			
对下一条发送方的指令等待至少5分钟	5.3.2		x				
在“250 OK”之后可避免交付失败	5.3.3					x	

在接受后发送错误通知消息	5.3.3	x				
发送使用空返回路径	5.3.3	x				
发送给信封返回路径	5.3.3		x			
发送到地址	5.3.3					x
剥离明确的来源路径	5.3.3		x			
最小的可接受延迟(RFC-1047)	5.3.3	x				
SMTP发送方:						
MAIL，RCPT中的域名标准化	5.2.2	x				
实现 SEND, SOML, SAML	5.2.4			x		
在HELO中发送有效的首要主机名	5.2.5	x				
在RCPT TO:中发送明确的来源路径	5.2.6				x	
仅使用因应码决定行动	5.2.10	x				
当可能时仅使用回应码的高位	5.2.10		x			
为透明性添加“.”	5.2.11	x				
软件失败后重发消息	5.3.1.1	x				
重试之前延时	5.3.1.1	x				
可配置重试参数	5.3.1.1	x				
重试每个排队的目的主机	5.3.1.1		x			
对相同的DATA可有多个RCPT	5.3.1.1		x			
支持多并发传输	5.3.1.1			x		
提供并发的限制	5.3.1.1		x			
对所有动作进行超时设定	5.3.1	x				
每个指令进行超时设定	5.3.2		x			
超时设定可容易地重新配置	5.3.2		x			
推荐的时间	5.3.2		x			
轮流尝试队列中的地址	5.3.4	x				
对轮流尝试限制可配置	5.3.4			x		
尝试至少两个地址	5.3.4		x			
Load-split across equal MX alternates	5.3.4		x			
使用域名系统	5.3.5	x				
支持MX记录	5.3.5	x				
在MX处理中使用WKS记录	5.2.12				x	
邮件转发:						
改变现有的头域	5.2.6				x	
实现转发功能：: 821第3.6节	5.2.6			x		

如果不可以，交付到RHS域	5.2.6		x			
解释地址的“本地部分”	5.2.16					x
邮件列表和别名						
支持两者	5.3.6		x			
为本地管理员报告邮件列表错误	5.3.6	x				
邮件网关:						
在本地部分中嵌入外来的邮件路径	5.2.16			x		
当需要时重写头域	5.3.7			x		
预先准备Received:行	5.3.7	x				
改变现有的Received:行	5.3.7					x
在Internet方面接受全部RFC-822	5.3.7		x			
对RFC-822的明确的来源路径起作用	5.3.7			x		
在Internet方面仅发送有效的RFC-822	5.3.7	x				
交付错误消息到信封地址	5.3.7		x			
根据错误返回地址设置返回路径	5.3.7		x			
用户代理-- RFC-822						
允许用户输入<route>地址	5.2.6				x	
支持RFC-1049目录类型域	5.2.13			x		
使用4个数字表示年份	5.2.14		x			
产生数字时间区域	5.2.14		x			
接受所有时间区域	5.2.14	x				
根据RFC-822使用非数字的时间区域	5.2.14	x				
省略在路径地址之前的短语	5.2.15			x		
接受并解析以点分隔的十进制域文字	5.2.17	x				
接受所有RFC-822地址格式	5.2.18	x				
产生非法的RFC-822地址格式	5.2.18					x
在头中完全符合域名	5.2.18	x				
在头中建立明确的来源路径	5.2.19				x	
在头中接受明确的来源路径	5.2.19	x				
发送/接收至少64KB的消息	5.3.8	x				

6．支持服务

6．1 域名转换

6．1．1 介绍

所有的主机必须实现对域名系统的解决方案，它必须提供一种机制，用来把主机名转换为 IP 地址，及相反的转换。

除了 DNS 外，主机也应实现查找本地 Internet 主机目录来转换主机名的机制。参看第 6.1.3.8 获取关于这个可选需求的更多信息。

讨论：

Internet 域名的转换最初是通过查找一个所有主机目录的本地复本实现的。随着不断的更新，这个目录越来越大，对于更新和发布而言它太大了，难以适应众多主机，所以，产生了 DNS。

DNS 建立了一个分布式数据库，主要用来在主机名和主机地址之间进行转换。实现 DNS 的软件也是必需的。DNS 包含两个逻辑上独立的部分：名字服务和解决程序（虽然实现时通常因为效率的原因将它们组合在一起）[DNS:2]。

域名服务器存储有关数据库确定部分的权威数据，并回答关于这些数据的查询。域解决程序为用户进程查询域名服务器。所以所有的主机都需要一个DNS解决程序，有些主机还需要运行域名服务程序。因为没有服务器有完整的信息，所以，通常为了解决一个查询需要从多个名字服务器获取信息。

6．1．2 协议准备

执行者应仔细研究参考资料[DNS:1] and [DNS:2]。它们提供了关于域名系统的彻底的原理、协议和执行实现的内容，反映出多年的经验。

6．1．2．1 零 TTL 的资源记录：RFC-1035 第 3.2.1 节

所有的 DNS 服务器和解决程序必须完全掌握零 TTL 的 RR：返回 RR 给客户端，但不能将其存在缓存中。

讨论：

零 TTL 值的意思是 RR 仅能在进程中处理，不应存入缓存，它们对非常不稳定的数据很有用。

6．1．2．2 QCLASS 值：RFC-1035 第 3.2.5 节

“QCLASS=*”的查询不应该被除查询者外的人使用，从多于一级的系统中查找数据。特别是，如果查询者仅对 Internet 数据类型感兴趣，必须使用 QCLASS=IN。

6．1．2．3 未使用的范围：RFC-1035 第 4.1.1 节

查询或回应的消息中未使用的范围必须是零。

6．1．2．4 压缩：RFC-1035 第 4.1.4 节

名字服务器必须在回应中使用压缩。

讨论：

压缩实质上是为了避免 UDP 信息包的溢出；参看第 6.1.3.2 节。

6．1．2．5 误用配置信息：RFC-1035 第 6.1.2 节

递归的名字服务器和完整服务解决程序通常有若干配置信息，包含关于根目录或本地名字服务器定位的线索。具体实现有可能在回应中包含任何这样的线索。

讨论：

许多执行者已经发现存储这些线索（对数据进行缓存）会带来方便，但忽视了要确保这些缓存的数据不被包含在回应中。因为线索的陈旧无效或不正确已经导致了 Internet 中的严重问题。

6．1．3 特殊问题

6．1．3．1 解决程序的执行

如果主机支持并发进程，名字解决程序就能进行多路的同步请求。

在 DNS 解决程序的执行中，可以任意选择两个不同模型之一：完整的服务解决程序，或部分的解决程序。

（A）完整的服务解决程序

完整的服务解决程序是解决程序服务的完整实现，并且有处理通信失败，个别名字服务器失败，对给定的名字进行合适的名字服务器定位的能力等。它必须满足下列必要条件：

●解决程序必须实现本地缓存功能，以避免对同一请求进行重复的远程访问，必须对缓存中的信息进行超时无效处理。

●解决程序应该可以根据指出本地域的多个根名服务器和多个名字服务器的开始信息来进行配置。这确保了解决程序在一般情况下能访问整个名字空间，并能在万一本地网络与Internet断开连接的情况下，访问本地信息。

（B）部分的解决程序

“部分的解决程序”依赖于已连接网络或“附近的”网络上的递归名字服务器的服务。该方案允许主机将解决程序的任务传递给其他主机的名字服务器。该模式通常应用在像 PC 这样能力较小的主机上，而且也推荐当主机是本地网络多个工作站之一时使用该模式，因为它允许所有的工作站共享递归名字服务器的缓存，并且减少本地网络域请求出口的数量。

作为最低限度，部分的解决程序必须有把它的请求提供给其它递归名字服务器的能力。注意：递归名字服务器可以限制请求的来源，所以，主机管理员必须校验要提供的服务。部分的解决程序可以选择实现缓存，但是假如那样的话，必须要对缓存中的信息进行超时无效处理。

6．1．3．2 传输协议

为了发送（非环形传送）查询，DNS 解决程序和递归服务器必须支持 UDP，并且应该支持 TCP。特别是，DNS 解决程序或服务器在发送非环形传

送查询之前必须先发送一个 UDP 查询。如果回应的 answer 部分被删除并且如果请求者支持 TCP，它应再次尝试用 TCP 进行查询。

DNS 服务必须能对 UDP 查询提供服务，并且应该能对 TCP 查询提供服务。名字服务器可限制用于 TCP 查询的资源，但不应拒绝为 TCP 查询提供服务。

被删除的回应不能被保存（不能被缓存），并且不能以这样的方式在以后使用。

讨论：

查询时 UDP 应作为首选，因为 UDP 查询在信息包数量和连接状态两方面占用更低的资源， UDP 的使用实质上加重了服务器的负担，特别是根服务器的负担。另外 UDP 也更健壮，因为解决方案解决程序可以尝试几个 UDP 查询到不同的服务器而只相当于付出一个 TCP 查询的代价。

DNS 回应可能被删除，虽然在现在的 Internet DNS 中这种情况极少发生。实际上，删除不可预知，因为它是依赖于数据的。这些依赖包括回答中 RR 的数量，每个 RR 的大小，和用名字压缩算法实现了的空间存储。作为一个简单的规则，当回答中包含 15 个或更少的 RR 时，NS 和 MX 列表中不会发生删除。

无论是否有使用依赖于应用程序的被删除回答的可能性，邮寄者都不能使用被删除的 MX 回应，因为这将导致邮件循环。

有责任让UDP满足绝大多数情况。名字服务器在回应中必须使用压缩。解决程序必须能够区分回应中Additional部分的切断（这仅丢失了附加信息）和Answer部分的切断（MX记录给出了邮件程序不可使用的回应）。数据库管理员应仅列出一个合理数量的名字服务器列表中主要名字，MX是可选的，及其他。

不过很清楚，未来将会定义新的 DNS 记录类型，它们包含的信息将突破 UDP 对 512 字节的限制，所以那时就要用到 TCP 了。所以，解决程序和名字服务器应实现 TCP 服务作为今天 UDP 的备份，可以推测未来将需要 TCP 服务。

经由私人同意，名字服务和解决程序可以使用 TCP 在他们之间进行所有的传输。环形传送必须使用 TCP。

DNS 服务器必有充分的内部并发能力，使它能够在等待回应时处理 UDP 查询，或在开放的 TCP 连接[DNS:2]上执行环形传输。

服务器可以支持一个用IP广播或多点传送地址交付的UDP查询。不过，Recursion Desired位不能被在多点传送查询中设置，并必须可被通过广播或多点传送地址被接收查询的名字服务器忽略。发送广播或多点传送DNS请求的服务器应该发送把它们当作偶然的探测器来发送，隐藏从回应中获得的IP地址。

讨论：

广播或（特别是）IP 多点传送可以提供一种方法，用来定位预先不知道 IP 地址的附近的名字服务器。不过，一般递归查询的广播会导致网络和服务器额外的和不必要的负担。

6．1．3．3 高效的资源用法

下列服务器或解决程序的必要条件对作为整体的 Internet 的状态非常重要，尤其当 DNS 服务被像 mailer 这样高级的自动服务器反复调用时。

（1）解决程序必须实现重传控制，以确保不会浪费通信带宽，也必须利用有限的带宽响应每个单独的请求。参看[DNS:2]第 43-44 页获取特别建议。

（2）当一个查询已经重传了几次而没有回应后，具体实现必须放弃并为应用程序返回一个软件错误。

（3）所有的名字服务器和解决程序应该对暂时错误进行缓存，保持大约几分钟的超时周期。

讨论：

这将防止应用程序立即重试因过多的 DNS 流量而引起的软件错误（违反本文档第 2.2 节的规定）。

（4）所有的 DNS 名字服务器和解决程序应该对以下的拒绝回应进行缓存：在[DNS:2]中定义的，不存在的特定名字，或特定类型的数据。

（5）当 DNS 服务器或解决程序回重试 UDP 查询时，其重试的时间间隔应该由一个指数补偿算法来限定，并且也应该有高端和低端的限制。

执行：

一个标准的 RTT 和数据偏离值（如果可用）应被用来计算初始重传时间间隔。如果这个信息无效，应使用不少于 5 秒的缺省值。具体实现应限制重传的时间间隔，但是此限制必须超过两个 Internet 最大段的生存时间加上名字服务器的服务延迟的时间。

（6）当一个解决程序或服务器接收到一个因查询已发出而产生的 Source Quench 时，它应该采取措施，减少最近查询该服务器的速度。服务器可以忽略它接收到的 Source Quench，把它当作是发送的回应数据流。

执行：

一个减少速度的推荐方案，是发送下一次查询到另一台备用的服务器，如果有可用的备用服务器的话。另一个方案是对相同的服务器进行重试的时间间隔补偿。

6．1．3．4 多源主机

当主机的名字-地址转换功能遇到拥有多源地址的主机时，它应使用直接连接的网络数量的信息和任何其他适用的性能或历史信息来对地址进行排列或排序。

讨论：

多源主机的不同地址通常意味着不同的 Internet 路径，和其他在性能、可靠性或管理限制方面更可取的路径。对测定域系统的最佳路径没有常规的方法。一个推荐的方案，是基于对由系统管理员配置的本地信息的测定。

执行：

下列方案已被应用成功：

（a）将网络优先级列表合并到主机配置数据中，这是一个网络优先级顺序的简单列表。该列表在没有优先级时可以是空的。

（b）当主机名被映射到一个 IP 地址的列表中时，这些地址应该按网络编号排序，并以相应的顺序排列以网络优先级列表中。未出现在网络优先级列表中的的网络 IP 地址应被排到列表的末尾。

6．1．3．5 可扩展性

DNS 软件必须支持所有已知的独立类型的格式[DNS:2]，并应减少因新的已知类型和本地非标准的实验类型的引入而造成的损害。

讨论：

DNS 使用的数据类型和种类是可扩展的，所以可以添加新类型，旧类型会被删除或废除。新数据类型的引入应仅依赖于 DNS 消息内的域名压缩规则和资源记录（RR）的内部格式与可打印格式之间的转换。

压缩依赖于特定 RR 内的数据格式的状态。所以压缩必须仅应用在已知的独立类型的 RR 的内容上，并且决不能用在特定种类的 RR 或未知的 RR 类型上。一个 RR 的自己的名字总是符合压缩的规则。

名字服务器可能通过环带传输获得服务器不知如何转换到可打印格式的 RR。解决程序可以接收类似的信息作为查询的结果。因为特殊的处理，数据必须被存储，所以这意味着 DNS 软件不能使用原文的格式进行内部存储。

DNS 定义了非常通过的域名语法——每个标志字符串包含最多 63 个 8 位字节，用圆点分隔，总的最大长度 255 字节。DNS 的特殊应用被允许可对它们使用的域名语法的约束进行扩充，域名的配置已经允许一些应用使用更一般的名字。特别是，本文的第 2.1 节对 RFC-952[DNS:4]中定义的合法 Internet 主机名的语法放宽了限制。

6．1．3．6 RR 类型的状态

名字服务器必须可以从配置文件装载除 MD 和 MF 以外的所有 RR 类型。MD 和 MF 类型已被废除，并不能再被执行；特别是名字服务器不能从配置文件中装载这些类型。

讨论：

RR 类型 MB，MG，MR，NULL，MINFO 和 RP 被视为是实验性的，使用 DNS 的应用程序不能期望这些类型被大多数域支持。另外这些类型会被重新定义。

TXT 和 WKS RR 类型还没有被 Internet 站点广泛地使用。其结果是应用程序不能依赖大多数域中存在的 TXT 或 WKS RR。

6．1．3．7 健壮性

当由于网络连接或其他问题导致根服务器或其他服务器无效时，DNS 软件可能需要在这样的环境下工作。此时，DNS 名字服务器和解决程序必须继续根据可获得的部分名字空间来提供服务，对其他请求给出暂时失败的回应。

讨论：

虽然 DNS 的本意起初是在连接到 Internet 的环境下使用的，但是有将它应用在未连接到 Internet 的网络系统中的可能。所以执行者在提供本地名字服务之前不能依赖于对根服务器的访问。

6．1．3．8 本地主机目录

讨论：

主机可以使用一个本地主机目录作为 DNS 的后备或补充。新增加的问题是如何设置优先级，是 DNS 还是主机目录？大多数灵活的方案都为此提供了配置的可选项。

其特征是，这种附加的主机的内容目录将由附近的站点来决定。不过，一个公用的 Internet 主机目录由 DDN 网络信息中心（DDN NIC）根据[DNS:4]的格式来维护。该目录可以使用[DNS:5]中描述的协议从 DDN NIC 重新得到。请注意，该目录仅包含全部 Internet 主机的一小部分。使用这个协议得到 DDN NIC 主机目录的主机，应在用 ALL 指令请求全部的目录之前用 VERSION 指令来检查表格是否已经改变。VERSION 标识符应被视为一个随机的字符串，并仅为用来进行平等的测试，不采用数字表示的序列。

DDN NIC 主机目录包含管理信息，因其对主机操作无用，所以现在还未被包含在 DNS 数据库内；另外还包含网络各网关条目的样本。不过，大量的这类附加信息将来将被加入到 DNS 中。相反地，DNS 提供 DDN NIC 主机目录无法提供的基本服务（特别是 MX 记录）。

6.1.4 DNS 用户界面

6．1．4．1 DNS 管理

本文与主软件中的设计和执行问题相关，与管理和操作问题无关。不过，管理问题在 DNS 中非常重要，因为这个巨大的分布式数据库的特别部分中的错误，可以导致某些站点的少量或不正确的执行。这些问题在[DNS:6]和[DNS:7]中讨论。

6．1．4．2 DNS 用户界面

主机必须为运行在主机上的所有应用 DNS 程序提供界面。界面将直接请求系统进程执行解决程序功能[DNS:1, 6.1:2]。

作为最小的实现，基本界面必须支持与指定名字相关联的、特定类型的所有信息的请求，也必须返回所有被请求的信息，一个确定的错误代码，或一个软件错误标志。当没有错误时，基本界面返回未经修改、删除、排序的完整回应信息，所以基本的界面无需为新的数据类型而改变。

讨论：

软件错误标志是界面的基本部分，因为并不一直有访问来自 DNS 的特殊信息的可能性，参看第 6.1.3.3 节。

主机可以提供为特殊功能定制的 DNS 界面，将自然状态的域数据转换到更适合这些功能的格式。特别是，主机必须提供方便地在主机地址和主机名之间转换的界面。

6．1．4．3 界面缩写工具

用户界面可以为用户提供一种输入缩写的常用名字的方法。虽然这种方法的定义已超出了 DNS 规范的范围，但需要可靠的规则，以确保这些方法可以有权使用全部的 DNS 名字空间，并预防对 Internet 资源的过量使用。

如果要提供一个缩写在方法，那么：

（a）必须方便地指出一个名字已经完成，以结束缩写在方式。通常的方法是其后跟一个圆点。

（b）缩写必须一次就正确地扩充成功，必须在已经输入的上下文中成功。

讨论：

例如，

如果一个缩写用来代表邮件程序的接收方，缩写应该写出完整的域名，结尾有一个指示已经完成的标志，存储在消息队列中。另外，缩写可以写出邮件系统搜索列表，由于按照交互的通配符形式而增加的反复尝试。

有两最常用的缩写方案：

（1）界面级别名

界面级别名以一个别名/域名成对的列表的概念实现的。列表可以按照用户或按照主机排序，因不同的功能来分隔列表，比如，一个列表专为名字-地址的转换，另一个不同的列表专为邮件域。当用户输入一个名字，界面尝试用名字去匹配列表条目中的的别名成分，如果可以找到匹配的条目，名字就被相应的域名代替。

注意：界面级别名和 CNAME 是完独立的机制；界面级别名是本地事件，而 CNAME 是在广泛的 Internet 别名机制，它是任何 DNS 具体实现必需的一部分。

（2）查找列表

查找列表是以一个有序的域名列表的概念实现的。当用户输入一个名字，查找列表中的域名作为后缀加入到用户提供的名字，一个接一个地，直到关联的域名被找到，或列表查找完毕。查找列表通常包含本地主机父域的名字或其他祖先域。查找列表通常按用户或按进程排序。

管理员将查找列表工具设成无效是可能的。可以证明，管理方面的拒绝在一些情况下，防止了 DNS 的滥用。

有一个危险，就是当检查用户是否输入了完整的域名，是否缺乏结尾的句号作为完成的标记，查找列表机制将产生对根服务器的过多查询。查找列表机制必须有下列一个或两个预防措施来防止这样的情况：

（a）本地的解决程序/名字服务器可以执行对拒绝的回应的缓存（参看第 6.1.3.3 节）。

（b）查找列表在尝试使用名字在非本地的域服务器上（如根服务器）进行查询之前，可以要求在产生的域名中加入两个或更多的内部句号。

讨论：

该必要条件的意图是当查找列表被测试时，避免对用户过多的延迟，更重要的是防止对根服务器或其他高级别服务器的过多访问量。例如，如果用户提供了一个名字“X”，并且查找列表将根服务器作为组成成分，查询在访问下一个可选择的查找列表时将访问根服务器。其结果是根服务器和其附近的网关的负荷将因 Internet 上的大量主机而增加。

可选的拒绝缓存限制了名字在第一时间被使用的事件。内部的句号规则仅可预防一些顶级名字的简单应用。

6．1．5 域名系统必要条件一览

特性	章节	SHOULD-USE	MUST-UNDERSTAND	MUST	SHOULD-NOT	MUST-NOT	Footnote
一般问题							
实现DNS名字到地址的转换	6.1.1	x					
实现DNS地址到名字的转换	6.1.1	x					
支持使用主机目录转换	6.1.1			x			
适当处理零TTL的RR	6.1.2.1	x					
使用QCLASS=*不是必要的	6.1.2.2		x				
为Internet种类使用QCLASS=IN	6.1.2.2	x					
不使用的域零	6.1.2.3	x					
在回应中使用压缩	6.1.2.4	x					
在回应中包含配置信息	6.1.2.5					x	
支持所有已知的，独立的类型	6.1.3.5	x					
容易地扩展类型列表	6.1.3.5		x				
装载所有RR类型（除了MD和MF）	6.1.3.6	x					
装载MD或MF类型	6.1.3.6					x	
在根服务器等无效时仍可运行	6.1.3.7	x					

解决程序的问题:							
解决程序支持多个并发请求	6.1.3.1		x				
完整的服务解决程序:	6.1.3.1			x			
本地缓存	6.1.3.1	x					
本缓存中的超时信息	6.1.3.1	x					
开始信息的可配置能力	6.1.3.1		x				
部分的解决程序：	6.1.3.1			x			
使用多余的递归名字服务器	6.1.3.1	x					
本地缓存	6.1.3.1			x			
本缓存中的超时信息	6.1.3.1	x					
对远程多源主机的支持：							
按优先级列表排列多个地址	6.1.3.4		x				
传输协议：							
支持UDP查询	6.1.3.2	x					
支持TCP查询	6.1.3.2		x				
首先使用UDP发送查询	6.1.3.2	x					1
如果UDP回答被删除，尝试TCP	6.1.3.2		x				
名字服务器限制TCP的查询资源	6.1.3.2			x			
消耗不必要的TCP查询	6.1.3.2				x		
使用被删除的数据	6.1.3.2					x	
仅使用TCP的私有认可	6.1.3.2			x			
对环形传输使用TCP	6.1.3.2	x					
TCP的使用不防碍UDP查询	6.1.3.2	x					
支持广播和多点传送查询	6.1.3.2			x			
查询中的RD bit设置	6.1.3.2					x	
被多点传送的或广播的服务器忽略RD位	6.1.3.2	x					
仅把地址作为偶然的探测器发送	6.1.3.2		x				
资源利用：							
按照[DNS:2]进行传输控制	6.1.3.3	x					
每个请求有限的带宽	6.1.3.3	x					
重试后失败=> 软件错误	6.1.3.3	x					
缓存暂时性失败	6.1.3.3		x				
缓存拒绝回应	6.1.3.3		x				
使用指数补偿后再试	6.1.3.3		x				

高端，低端限制	6.1.3.3		x				
客户端处理Source Quench	6.1.3.3		x				
服务器忽略Source Quench	6.1.3.3			x			
用户界面：							
所有的程序有权使用DNS界面	6.1.4.2	x					
可以对给定的名字请求所有的信息	6.1.4.2	x					
返回完整的信息或错误	6.1.4.2	x					
特殊界面	6.1.4.2			x			
名字<->地址转换	6.1.4.2	x					
缩写工具：	6.1.4.3			x			
完整名字的规定	6.1.4.3	x					
一次正确的规定	6.1.4.3	x					
在适当的上下文的规定	6.1.4.3	x					
查找列表：	6.1.4.3			x			
管理员可以禁用	6.1.4.3		x				
过多根服务器查询的预防	6.1.4.3	x					
两种方案	6.1.4.3		x				

注：

1．除非在特定的解决程序和特定的服务器之间有私有认可。

6．2 主机初始化

6．2．1 介绍

本节讨论在已连接网络上的，或更通常的在 Internet 路径上的，主机软件的初始化。这对无盘主机是必需的，对有磁盘驱动器的主机是可选的。对于无盘主机，初始化过程称为“网络启动”，它由位于启动 ROM 芯片中的启动引导程序控制。

为了通过网络初始化无盘主机，有两个完全不同的阶段：

（1）配置 IP 层。

无盘的机器通常不能对网络配置信息进行永久的存储，所以合适的配置信息必须动态地获得，以支持它遵从的装载阶段。信息必须至少包含主机的 IP 地址和启动服务器。为了支持通过网关的启动，地址掩码和缺省的网关列表也是必需的。

（2）装载主机系统代码

在装载阶段，用一个合适的文件传输协议通过网络从启动服务器拷贝系统代码。

有盘主机可以执行第一步进行动态的配置。这对微型计算机非常重要，它们的软盘允许网络配置信息被复制到多台主机中。还有，新主机的安装，如果从中心服务器自动获得配置信息，将变得非常简单，而且节约管理员的时间，减少出错的可能性。

6．2．2 必要条件

6．2．2．1 动态配置

许多协议已经可以进行动态配置。

●ICMP 信息请求/回应消息

这个已废除的消息对被设计用来允许一台主机查找它所在网络的编号。不幸的是，它只在已知它的 IP 地址的主机编号对时才可用，主机需要的信息的动态配置很难实现。

●逆地址解决协议（RARP）[BOOT:4]

RARP 适用于广播方法的链路层协议，它允许在给出的链路层地址中查找 IP 地址。不幸的是，RARP 不能通过 IP 网关工作，所以在每一个网络上都需要一个 RARP 服务器。另外，RARP 不能提供任何其他的配置信息。

●ICMP 地址掩码请求/回应消息

ICMP 消息允许一台主机利用特殊的网络界面获取地址掩码。

●BOOTP 协议[BOOT:2]

该协议允许一台主机检测本地主机和启动服务器的 IP 地址，适当的启动文件名，任意的地址掩码和缺省网关的列表。为了定位 BOOTP 服务，主

机使用 UDP 广播一个 BOOTP 请求。ad hoc 网关扩展被通过网关传送 BOOTP 广播，将来，IP 多点传送工具将为此用途提供标准的机制。

动态配置的建设方法是使用 RFC-1084 “BOOTP 厂商信息扩展”[BOOT:3]中定义的 BOOTP 协议扩展。RFC-1084 定义了一些重要的全面的（而不是特定厂商的）扩展。特别是，这些扩展允许在 BOOTP 中提供地址掩码，我们推荐在这种方式下提供地址掩码。

讨论：

在历史上，子网在 IP 之后很长一段时间才设计出来，所以一个单独的机制（ICMP 地址掩码消息）被设计用来提供地址掩码给主机。不过，IP 地址掩码和相应的 IP 地址组成一对，为了操作简单，它们应在相同的时间，用相同的机制来定义，不论配置文件或与 BOOTP 相似的动态机制如何。

注意：BOOTP 没有十分全面地说明多源主机所有界面的配置。多源主机必须为每个界面使用单独的 BOOTP，或使用 BOOTP 配置一个界面来执行装载，并且用稍后的一个文件执行完全的初始化。

应用层配置信息也被扩展到从装载系统代码以后的文件中获得。

6．2．2．2 装载阶段

装载阶段建议的方法是在由 BOOTP 确定的 IP 地址之间使用 TFTP[BOOT:1]。

不应该使用 TFTP 到一个广播地址，其原因在已第 4.2.3.4 节中解释。

6．3 远程管理

6．3．1 介绍

Internet 社区最近已经致力于网络管理协议的发展。其结果是产生了两种方法[MGT:1, MGT:6]：简单网络管理协议（SNMP）[MGT:4]和基于 TCP 的公共管理信息协议（CMOT）[MGT:5]。

为了可以管理使用 SNMP 和 CMOT，主机必须实现适当的管理代理。Internet 主机应该包含 SNMP 或 CMOT 之一的代理。

SNMP 和 CMOT 都对被称为管理信息库（MIB）的管理数值的集合起作用。通过读取和设置这些值，远程应用程序可以查询和改变被管理系统的状态。

一个标准的MIB[MGT:3]已经被定义来为双方的管理协议使用，使用的数据类型由定义在[MGT:2]中的管理信息结构（SMI）来定义。附加的MIB变量可以被引入到MIB名字空间[MGT：2]的“计划”和“实验”子树下。

所有协议模型都应实现与 MIB 有关的变量。主机应实现最近大多数标准 MIB 中定义的 MIB 变量，并可以在适当的或需要的时候实现其他 MIB 变量。

6．3．2 协议准备

MIB 是为了适用于主机和网关，虽然在 MIB 应用到这两种情况时可能会有细微的差别。本节包含了主机 MIB 的适当解释。可能 MIB 以后的版本会包含更多有关主机管理的项目。

一个被管理的主机必须实现下列MIB对象群的定义：System, Interfaces, Address Translation,IP, ICMP, TCP, and UDP.

下列详细的解释将应用于主机：

- ipInHdrErrors

注意，“超出生存时间”错误可能仅在主机转发一个来源路由数据包时发生。

- ipOutNoRoutes

本对象对因没有找到路由而被丢弃的数据包进行计数。它在主机所有缺省配置的网关都无效时发生作用。

- ipFragOKs, ipFragFails, ipFragCreates

一个没有故意执行分裂的主机（参看[INTRO:1]的“分裂”一节）必须为这三个对象返回零值。

- icmpOutRedirects

对主机来说，这个对象必须始终为零，因为主机不能转发。

- icmpOutAddrMaskReps

对主机来说，这个对象必须始终为零，除非主机是权威的地址掩码信息提供者。

- ipAddrTable

对主机来说，“IP Address Table”对象是一个有效的逻辑界面表格。

- ipRoutingTable

对主机来说，“IP Routing Table”对象是一个有效的主机路由缓存和在[INTRO:1]的“路由外出的数据包”中描述的静态路由表的组合。

在每个 ipRouteEntry 中，ipRouteMetric1~4 通常对主机来说没有含义，并且应该总是-1，同时 ipRouteType 的值通常是“remote”。

如果已连接的网络上的目的地没有出现在路由缓存中（参看[INTRO:1]的“路由外出的数据包”一节），将没有有关“目的”ipRouteType的条目。

讨论：

目前的 MIB 在 inRouteEntry 中不包含服务类型，但期望未来的扩展会将其加入。

我们也希望MIB经过扩展可以允许应用程序的远程管理（比如，部分地重新配置邮件系统的能力）。像邮件系统这样的网络管理应用程序应该为远程管理被填充“hook”。

6．3．3 管理必要条件一览

```
|          |  |  |  |S|  |
|          |  |  |  |H|  |F
|          |  |  |  |O|M|o
```

			S		U	U	o
			H		L	S	t
		M	O		D	T	n
		U	U	M			o
		S	L	A	N	N	t
		T	D	Y	O	O	t
特性	章节				T	T	e
支持SNMP或CMOT代理	6.3.1		x				
执行在标准MIB中指定的对象	6.3.1		x				

7．参考资料

本节列出了主要的参考资料，每个执行者必须彻底地熟悉它们。还列出了次要的参考资料，建议作为附加的阅读。

介绍的参考资料：

[INTRO:1] "Internet 主机必要条件——通信层,"
IETF 主机必要条件工作组, R. Braden, Ed., RFC-1122, 1989 年 8 月。

[INTRO:2] "DDN 协议手册,"
NIC-50004, NIC-50005, NIC-50006, (三册), SRI International, 1985 年 12 月。

[INTRO:3] "正式 Internet 协议,"
J. Reynolds and J. Postel, RFC-1011, 1987 年 5 月。
本文档周期性地出版关于新 RFC 编号的内容，必须使用最新版。

[INTRO:4] "协议文档状态信息,"
O. Jacobsen and J.Postel, RFC-980, 1986 年 3 月。

[INTRO:5] "编号的分配,"
J. Reynolds and J. Postel, RFC-1010,1987 年 3 月。
本文档周期性地出版关于新 RFC 编号的内容，必须使用最新版。

TELNET的参考资料：

[TELNET:1] "Telnet协议规范,"
J. Postel and J. Reynolds, RFC-854, 1983年5月。

[TELNET:2] "Telnet可选项规范,"
J. Postel and J. Reynolds,RFC-855, 1983年5月。

[TELNET:3] "Telnet二进制传输,"
J. Postel and J. Reynolds,RFC-856, 1983年5月。

[TELNET:4] "Telnet立即响应可选项,"
J. Postel and J. Reynolds, RFC-857, 1983年5月。

[TELNET:5] "Telnet对Go Ahead可选项的禁用,"
J. Postel and J. Reynolds, RFC-858, 1983年5月。

[TELNET:6] "Telnet状态可选项,"

J. Postel and J. Reynolds, RFC-859, 1983年5月。

[TELNET:7] "Telnet定时标记可选项,"
J. Postel and J. Reynolds, RFC-860, 1983年5月。

[TELNET:8] "Telnet扩展可选项列表,"
J. Postel and J.Reynolds, RFC-861, 1983年5月。

[TELNET:9] "Telnet记录结束可选项,"
J. Postel, RFC-855, 1983年12月。

[TELNET:10] "Telnet终端类型可选项,"
J. VanBokkelen, RFC-1091, 1989年2月。
本文档替代了RFC-930。

[TELNET:11] "Telnet窗口尺寸可选项,"
D. Waitzman, RFC-1073, 1988年10月。

[TELNET:12] "Telnet行模式可选项,"
D. Borman, RFC-1116, 1989年8月。

[TELNET:13] "Telnet终端速度可选项,"
C. Hedrick, RFC-1079, 1988年12月。

[TELNET:14] "Telnet远程流控制可选项,"
C. Hedrick, RFC- 1080, 1988年11月。

次要的TELNET参考资料:

[TELNET:15] "Telnet 协议,"
MIL-STD-1782, U.S. Department of Defense, 1984年5月。
本文档的目的是要描述一个与RFC-854相同的协议，如果造成冲突，
RFC-854有更高的优先权。

[TELNET:16] "SUPDUP协议,"
M. Crispin, RFC-734, 1977年10月。

[TELNET:17] "Telnet的SUPDUP可选项,"
M. Crispin, RFC-736, 1977年10月。

[TELNET:18] "数据登录终端可选项,"
J. Day, RFC-732, 1977年6月。

[TELNET:19] "TELNET数据登录终端可选项-- DODIIS实现,"
A. Yasuda and T. Thompson, RFC-1043, 1988年2月。

FTP参考资料:

[FTP:1] "文件传输协议,"
J. Postel and J. Reynolds, RFC-959, 1985年10月。

[FTP:2] "文档文件格式标准,"
J. Postel, RFC-678, December 1974年12月。

[FTP:3] "文件传输协议,"
MIL-STD-1780, U.S. Department of Defense, May 1984年5月。
本文档基于一个早期的FTP规范版本，它已被废除。

TFTP的参考资料：

[TFTP:1] "TFTP协议修订版2,"
K. Sollins, RFC-783, June1981年6月。

邮件的参考资料：

[SMTP:1] "简单邮件传输协议,"
J. Postel, RFC-821, 1982年8月。

[SMTP:2] "ARPA Internet文本消息格式的标准,"
D. Crocker, RFC-822, 1982年8月。
本文档废除了一个早期的规范， RFC-733.

[SMTP:3] "邮件路由和域系统,"
C. Partridge, RFC-974, 1986年1月。
本RFC描述了MX记录的应用， 一个邮件交付过程的强制扩展。

[SMTP:4] "重复消息和SMTP,"
C. Partridge, RFC-1047, 1988年2月。

[SMTP:5a] "X.400与RFC 822之间的映射,"
S. Kille, RFC-987,June 1986.

[SMTP:5b] ″ RFC-987的附录,″
S. Kille, RFC-???, September 1987.
上述两个RFC定义了一个在Internet和X.400环境下网关邮件的提议标准。

[SMTP:6] ″简单邮件传输协议,″
MIL-STD-1781, U.S. Department of Defense, 1984年5月。
本规范的目的是描述一个与RFC-821相同的协议，但MIL-STD-1781是不完整的，特别是它没有包含MX记录[SMTP:3]。

[SMTP:7] ″ Internet Messages的信文类型域,″
M. Sirbu, RFC-1049, 1988年3月。

域名系统的参考资料:

[DNS:1] ″域名——原理和工具,″
P. Mockapetris, RFC-1034, 1987年11月。
本文档和下列一个已被废除的RFC-882, RFC-883, and RFC-973.

[DNS:2] ″域名——实现和规范,″
RFC-1035,P. Mockapetris, 1987年11月。

[DNS:3] ″邮件路由和域系统,″
C. Partridge, RFC-974,1986年1月。

[DNS:4] ″DoD Internet主机表格规范,″
K. Harrenstein,RFC-952, M. Stahl, E. Feinler, October 1985.

次要的DNS参考资料:

[DNS:5] ″主机名字服务器,″
K. Harrenstein, M. Stahl, E. Feinler, RFC-953, 1985年10月。

[DNS:6] ″域管理员指南,″
M. Stahl, RFC-1032, 1987年11月。

[DNS:7] ″域管理员操作指南,″
M. Lottor, RFC-1033, 1987年11月。

[DNS:8] "域名系统手册,"
Internet协议手册第4卷， NIC 50007, SRI 网络信息中心，1989年8月。

系统初始化的参考资料：

[BOOT:1] "使用TFTP装载引导程序,"
R. Finlayson, RFC-906, 1984年6月。

[BOOT:2] "引导程序协议 (BOOTP),"
W. Croft and J. Gilmore, RFC-951, 1985年9月。

[BOOT:3] "BOOTP厂商信息扩展,"
J. Reynolds, RFC-1084, 1988年12月。
注意：本RFC修正并废除了RFC-1048.

[BOOT:4] "逆地址解决协议,"
R. Finlayson, T. Mann, J. Mogul, and M. Theimer, RFC-903, 1984年6月。

管理的参考资料：

[MGT:1] "IAB对Internet网络管理标准发展的建议,"
V. Cerf, RFC-1052, April 1988.

[MGT:2] "基于TCP/IP的互联网的管理信息的结构和认证,"
M. Rose and K. McCloghrie, RFC-1065, 1988年8月。

[MGT:3] "基于TCP/IP的互联网的网络管理信息的基础,"
M. Rose and K. McCloghrie, RFC-1066, August 1988.

[MGT:4] "一个简单的网络管理协议,"
J. Case, M. Fedor,M. Schoffstall, and C. Davin, RFC-1098, 1989年4月。

[MGT:5] "基于TCP/IP的公共管理信息服务和协议,"
U. Warrier and L. Besaw, RFC-1095, 1989年4月。

[MGT:6] "Second Ad Hoc Network Management Review Group的报告,"
V. Cerf, RFC-1109, 1989年8月。

安全考虑

在主机软件的应用程序和支持程序中有许多安全问题，但是关于它的完整讨论已超出了本RFC的范围。有关安全的问题在以下章节中涉及：有关TFTP（第4.2.1,4.2.3.4,4.2.3.5节），SMTP的VRFY和EXPN指令（第5.2.3节），SMTP的HELO指令（第5.2.5节）和SMTP的DATA指令（第5.2.8节）。

作者地址

（略——译者注）

原文：RFC-1123

《Requirements for Internet Hosts -- Application and Support》

译者：comehope，2002 年 7 月

博客：http://www.comehope.com