ANSI/IEEE 标准 802.1D，1998 年版

[纳入 ANSI/IEEE 标准 802.1D，1993 年版，IEEE P802.1p，
IEEE 标准 802.1j-1996、IEEE 标准 802.6k-1992、
IEEE 标准 802.11c-1998 和 IEEE P802.12e]

（经 ISO/IEC 通过并重新指定为
ISO/IEC 15802-3:1998）

IEEE 信息技术标准 — 系统间电信和信息交换 — 局域网和城域网 —

通用规格

第 3 部分：媒体访问控制 (MAC) 桥接器

被 ISO/IEC 采用并重新指定为
ISO/IEC 15802-3:1998

赞助

局域网/城域网标准委员会 的

IEEE 计算机学会

ANSI/IEEE 标准 802.1D，1998 年版

IEEE 标准文件是在 IEEE 学会和 IEEE 标准协会 (IEEE-SA) 标准委员会的标准协调委员会内制定的。委员会成员自愿服务且无偿。他们不一定是研究所的成员。IEEE 内部制定的标准代表了研究所内部对该主题的广泛专业知识以及 IEEE 之外那些表示有兴趣参与标准制定的活动的共识。

使用 IEEE 标准完全是自愿的。IEEE 标准的存在并不意味着没有其他方式来生产、测试 、测量、购买、营销或提供与 IEEE 标准范围相关的其他商品和服务。此外，在标准批准和发布时表达的观点可能会因现有技术的发展和标准用户的评论而发生变化。每项 IEEE 标准至少每五年接受一次审查，以进行修订或重申。如果一份文件超过五年而没有得到重申，那么可以合理地得出结论，其内容虽然仍然有一定价值，但并不完全反映目前的最新技术水平。用户应注意检查以确定他们拥有的是任何 IEEE 标准的最新版本。

欢迎任何感兴趣的人士对 IEEE 标准修订提出意见，无论其是否是 IEEE 会员。对文件更改的建议应以拟议的文本更改形式提出，并附上适当的支持意见。

解释：有时，标准中某些部分的含义可能与具体应用有关，因此可能会出现问题。当 IEEE 注意到需要解释时，该协会将采取行动准备适当的回应。由于 IEEE 标准代表了所有相关利益的共识，因此确保任何解释也得到利益平衡的一致同意非常重要。因此，IEEE 及其协会和标准协调委员会的成员无法立即回应解释请求，除非该问题之前已得到正式考虑。

有关标准的评论和解释请求请发送至：

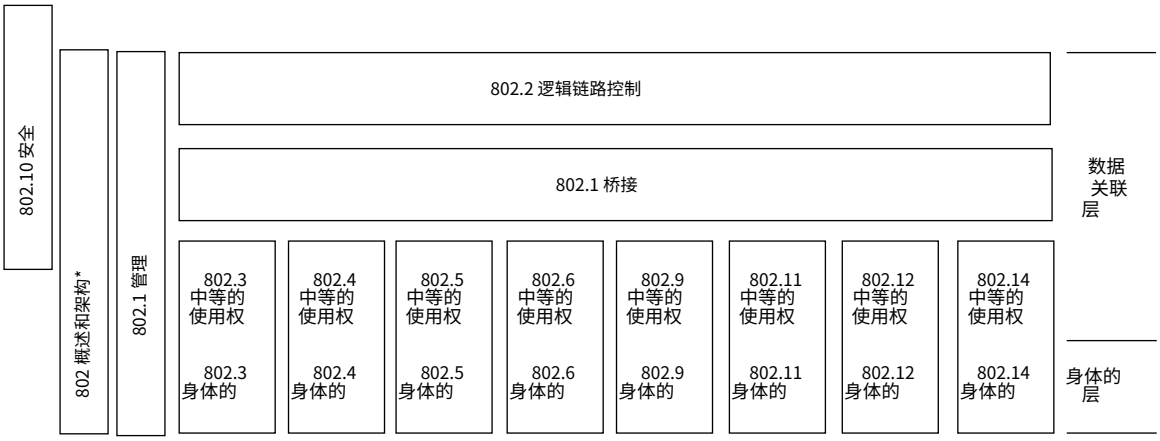
IEEE-SA 标准委员会秘书 445 Hoes Lane
邮政信箱 1331
美国新泽西州皮斯卡塔韦
08855-1331

请注意，实施本标准可能需要使用受专利权保护的主题。发布本标准时，不表明与此相关的任何专利权的存在或有效性。IEEE 不负责识别 IEEE 标准可能需要许可的专利，也不负责调查引起其注意的专利的合法性或范围。专利持有人已提交保证声明，保证将根据这些权利无偿或以合理的费率和非歧视性的合理条款和条件向所有希望获得此类许可的申请人授予许可。IEEE 不对专利持有人提供的许可协议的费率和/或条款和条件的合理性发表任何声明。可从 IEEE 标准部获取更多信息。

ANSI/IEEE Std 802.1D 简介，1998 年版

[本介绍不是 ANSI/IEEE Std 802.1D（1998 年版）《信息技术 — 系统间电信和信息交换 — 局域网和城域网 — 通用规范 — 第 3 部分：媒体访问控制 (MAC) 桥》的一部分。]

此标准是局域网和城域网标准系列的一部分。该标准与该系列其他成员之间的关系如下所示。（图中的数字指的是 IEEE 标准编号。）



* 以前称为 IEEE 标准 802.1A。

该系列标准涉及国际标准化组织 (ISO) 开放系统互连 (OSI) 基本参考模型 (ISO/IEC 7498-1 : 1994) 定义的物理层和数据链路层。访问标准定义了七种类型的介质访问技术及相关物理介质，每种类型均适用于特定应用或系统目标。其他类型正在研究中。

定义上述技术的标准如下：

- IEEE 标准 802 — *概述和架构。*本标准概述了 IEEE 802 标准系列。
- ANSI/IEEE 标准 802.1B [ISO/IEC 15802-2] — *局域网/城域网管理。*定义 OSI 管理兼容架构和 802.1k 以及用于在LAN/MAN环境中执行远程管理的服务和协议元素。
- ANSI/IEEE 标准 802.1D [ISO/IEC 15802-3] — *媒体访问控制 (MAC) 桥。*指定 MAC 服务边界以下 IEEE 802 LAN 互连的架构和协议。
- ANSI/IEEE 标准 802.1E [ISO/IEC 15802-4] — *系统加载协议。*为与 IEEE 802 LAN 上的系统加载有关的管理方面指定一组服务和协议。
- ANSI/IEEE 标准 802.1F — *IEEE 802 管理信息的通用定义和程序*
- ANSI/IEEE 标准 802.1G [ISO/IEC 15802-5] — *远程媒体访问控制 (MAC) 桥接.* 使用非 LAN 通信技术，指定在逻辑链路控制协议级别以下地理上分离的 IEEE 802 LAN 之间的互连扩展。
- ANSI/IEEE 标准 802.2 [ISO/IEC 8802-2] — *逻辑链路控制*

- ANSI/IEEE 标准 802.3 [ISO/IEC 8802-3] *CSMA/CD 访问方法和物理层规范*
- ANSI/IEEE 标准 802.4 [ISO/IEC 8802-4] *令牌传递总线访问方法和物理层规范*
- ANSI/IEEE 标准 802.5 [ISO/IEC 8802-5] *令牌环访问方法和物理层规范*
- ANSI/IEEE 标准 802.6 [ISO/IEC 8802-6] *分布式队列双总线 (DQDB) 访问方法和物理层规范*
- ANSI/IEEE 标准 802.9 [ISO/IEC 8802-9] *介质访问控制 (MAC) 层和物理 (PHY) 层的综合服务 (IS) LAN 接口*
- ANSI/IEEE 标准 802.10 *可互操作的 LAN/MAN 安全性*
- ANSI/IEEE 标准 802.11 [ISO/IEC DIS 8802-11] *无线局域网介质访问控制 (MAC) 和物理层规范*
- ANSI/IEEE 标准 802.12 [ISO/IEC 8802-12] *需求优先接入方式、物理层和中继器规范*

除了系列标准之外，以下是针对常见物理层技术的推荐做法：

- IEEE 标准 802.7 *IEEE 宽带局域网推荐规范*

以下附加工作组已授权正在开发的标准项目：

- IEEE 802.14 *有线电视宽带通信网络标准协议*

一致性测试方法

已建立了一个附加标准系列，编号为 1802，用于标识 802 系列标准的一致性测试方法文档。因此，802.3 的一致性测试文档编号为 1802.3。

ANSI/IEEE 标准 802.1D，1998 年版

MAC 桥接标准化活动促成了 IEEE 标准 802.1D-1990（随后重新发布为 ISO/IEC 10038:1993 [IEEE 标准 802.1D，1993 版]）的开发，该标准规定了 MAC 服务边界以下 IEEE 802 LAN 互连的架构和协议。IEEE 标准 802.1D-1990 还引入了桥接 LAN 中的过滤服务概念，以及获取此类 LAN 中的过滤信息并将其保存在过滤数据库中的机制。ISO/IEC 10038:1993 的此修订版扩展了过滤服务的概念，以定义桥接 LAN 中的其他功能，旨在实现以下目标：

a) 提供加速流量能力，以支持在局域网环境中传输时间关键型信息；

b) 提供支持 LAN 环境中组动态定义和建立的过滤服务，以及通过网桥对帧进行过滤，以便将发往特定地址的帧发送给

组的消息仅在需要到达该组成员的 LAN 段上进行转发。

为此，本文件对 ISO/IEC 10038: 1993 进行了一系列变更和补充，定义如下：

a) 桥接局域网中过滤服务的性质；
b) 流量类别的概念以及在网桥中支持多种流量类别对转发过程运行的影响；

c) 支持动态多播过滤服务所需的过滤数据库的结构；

d) 提供动态多播过滤服务所需的注册协议；
e) 支持动态多播过滤服务管理所需的管理服务和操作。

IEEE Std 802.1D 和 IEEE P802.1Q 之间的关系

正在开发的另一个 IEEE 标准 IEEE P802.1Q 扩展了过滤服务和 MAC 桥接的概念，以提供一组功能，使 MAC 桥接能够支持虚拟局域网 (VLAN) 的定义和管理。

IEEE P802.1Q 中定义的功能包括定义 VLAN 帧格式，该格式能够通过不具备发送优先级信息功能的 LAN 技术（如 CSMA/CD）传输 VLAN 标识和用户优先级信息。此信息在附加报头字段中传输，称为*标签标头*，它插入在原始帧的目标 MAC 地址和源 MAC 地址（以及路由信息字段，如果存在）之后。IEEE P802.1Q 扩展了此标准的优先级处理方面，以利用 VLAN 帧格式的能力，在任何一组连接的底层 MAC 上端到端地传输用户优先级信息。

IEEE 802.1Q 中包含的 VLAN 桥接规范独立于此标准，因为 IEEE 802.1Q 对 VLAN 桥接的一致性要求与此标准中定义的 MAC 桥接的一致性要求做出了独立且不同的陈述。但是，IEEE 802.1Q 使用了此标准中包含的规范的许多元素，特别是

a) 桥接架构；
b) 内部子层服务，以及 IEEE 802 LAN MAC 对其提供的规范；
c) 货运代理运作的主要特点；
d)生成树算法和协议；
e) 通用属性注册协议（GARP）；以及
f) GARP 多播注册协议 (GMRP)。

参與者

以下是 IEEE 802.1 工作组互通活动参与者的列表。发布时有投票权的成员标有星号 (*)。

威廉・P・利丁斯基，*椅子米克・西曼，*互通任务组主席**

托尼・杰弗里，*编辑*

史蒂夫・亚当斯*
史蒂芬・艾德斯
肯・奥朗格
弗洛伊德・巴克斯*
约翰・巴特利特*
莱斯・贝尔*
艾夫纳・本・多尔
迈克尔・伯杰*
詹姆斯・S・宾德*
大卫・布雷迪
马丁・布鲁尔
比尔・邦奇*
鲍勃・卡迪纳尔
保罗・卡罗尔*
杰弗里・卡特林*
丹尼斯・凯夫
艾伦・钱伯斯*
陈志伟
大卫・W・张*
肯・查普曼
韩华进*
迟冲
克里斯・克里斯特*
保罗・康登*
格伦・康纳利*
大卫・库勒罗特*
泰德・戴维斯*
安迪・戴维斯
大卫・德莱尼*
普拉卡什・德赛
杰弗里・迪茨*
库尔特・多宾斯
彼得・埃克莱辛*
JJ Ekstrom*
诺曼・W・芬恩*
伊沙伊・弗兰克尔
保罗・弗兰茨
拉尔斯・亨里克・弗雷德里克森* 阿努普・加瓦尼*
约翰格林汉姆
史蒂夫哈多克
沙拉姆・哈基米*
约翰・哈特*
斯科特・哈维尔
韦恩・海瑟薇
理查德・豪斯曼*
维克・海耶斯
戴维・海德*
盖比・赫克特
迪帕克・赫格德 *
艾瑞尔・亨德尔
约翰・希基
大卫・霍伦德
史蒂夫・霍洛维茨*
熊美玲
丽塔・亨特
戴维・胡萨克
阿尔塔夫・侯赛因*
Vipin K. Jain*
尼尔・贾维斯
托尼・杰弗里*
艾伦・凯西
加藤丰之*
哈尔・基恩*
凯文・凯彻姆*
基思・克拉姆*
布鲁斯・克林*
沃尔特・克尼特尔
丹・克伦特*
保罗・库默
保罗・拉沙佩尔*
比尔・莱恩
保罗・兰吉尔*
比尔・利丁斯基*
约翰・林德梅尔*
加里・利特尔顿
罗伯特・D・洛夫
安迪・卢克
彼得・马蒂尼
基思・麦克洛格里
马丁・麦克尼利斯
米兰・默哈尔*
约翰・梅森杰*
科林・米克
阿莫尔・米特拉
亚隆・纳赫曼*
克里希纳・纳拉亚纳斯瓦米*
保罗・尼科利奇
Lawrence Ng*
亨利・魏*
尤金・奥尼尔
小原聪*
大冈俊雄*
约尔格・奥滕斯迈尔*
吕克・帕里索*
约纳达夫・佩里
约翰・皮肯斯*
吉迪恩普拉特
科克・普莱斯
史蒂夫・兰伯格*
什洛莫・雷切斯 *
迪克・雷奥尔
詹姆斯・里士满*
阿尼尔・里辛加尼 (Anil Rijsinghani)*
道格・鲁比
雷・萨莫拉
艾曼・赛义德*
米克・希曼*
里奇・塞弗特
李・森德尔巴赫*
希曼舒・沙阿*
菲尔・西蒙斯*
K.卡尔・岛田
舒健
帕拉姆吉特・辛格*
罗斯玛丽・V・斯莱格*
亚历山大・史密斯*
安德鲁・史密斯*
拉里・斯蒂芬尼*
斯图尔特・索洛韦*
桑德尔・苏布拉马尼亚姆*
理查德・斯威特
罗宾・塔斯克*
福阿德・托巴吉
佃直树
达德苏古尔・瓦曼
史蒂夫・范塞特斯*
多诺・范・米罗普*
约翰・韦克利*
王彼得*
王飞利
王永昌*
特雷弗・沃里克*
鲍勃・沃森
艾伦・魏斯伯格
格伦・韦尼格
基思・威利特*
迈克尔・维特科夫斯基*
黄爱德华*
迈克尔・D・赖特*
米歇尔・赖特*
余志雄*
韦恩・扎科夫斯基*

IEEE Std 802.1D 投票委员会成员如下：

威廉・B・亚当斯
基特・阿苏尔
威廉・E・艾恩
托马斯・W・贝利
布拉德・J・布斯
彼得・K・坎贝尔
詹姆斯・T・卡洛
戴维・E・卡尔森
艾伦・M・钱伯斯
弗雷德里克・N・蔡斯
罗伯特・克劳德
托马斯・J・迪宁
彼得・艾克莱辛
约翰・W・芬德里希
迈克尔・A・菲舍尔
哈维・A・弗里曼
帕特里克・S・戈尼亚
胡里奥・冈萨雷斯・桑斯
罗伯特・M・格罗
克里斯・G・盖伊
斯蒂芬・R・哈多克
艾伦・W・哈撒韦
J. Scott Haugdahl
肯尼斯・赫克
亨利・霍伊特
拉杰・贾恩
尼尔・A・贾维斯
托尼・杰弗里
爱德华・R・凯利
彼得・M・凯利
金龙范
塔迪厄斯・科比拉兹
丹尼尔・R・克伦特
史蒂芬・巴顿・克鲁格
肯尼思・C・孔
威廉・G・莱恩
大卫・J・劳
兰斯・M・利奇
威廉・利丁斯基
兰道夫・S・利特尔
约瑟夫・G・马利
彼得・马蒂尼
克里斯・麦克唐纳
米兰・默哈尔
约翰・L・梅森杰
贝内特・迈耶
科林・K・米克
吉恩・E・米利根
戴维・S・米尔曼
沃伦・梦露
约翰・E・蒙塔古
韦恩・D・莫耶斯
西蒙・穆勒
保罗・尼科利奇
罗伯特・奥哈拉
查尔斯・奥斯特莱彻
约尔格・奥滕斯迈尔
罗杰・潘丹达
露西・W・佩森
约翰・R・皮肯斯
维克拉姆・普尼
安德里斯・普特宁斯
爱德华・Y・罗歇
詹姆斯・W・罗姆莱恩
弗洛伊德・E・罗斯
克里斯托弗・鲁兰德
诺曼・施奈德温德
米克・西曼
里奇・塞弗特
迈克尔・A・史密斯
威廉・R・史密斯
帕特里夏・泰勒
杰弗里・O・汤普森
马克・雷内・内田
约翰・维亚普拉纳
巴里・M・沃恩布鲁克
唐纳德・F・韦尔
厄尔・J・惠特克
杨千里
袁奥伦
乔纳森・M・茨威格

当 IEEE-SA 标准委员会于 1998 年 6 月 25 日批准 IEEE 标准 802.1D 时，其成员如下：

理查德・J・霍勒曼，*椅子*　　**唐纳德・N・海尔曼，***副主席*

朱迪思・戈尔曼，*秘书*

萨蒂什・K・阿加瓦尔
克莱德・R・坎普
詹姆斯・T・卡洛
加里・R・恩格曼
哈罗德・E・爱泼斯坦
杰伊・福斯特*
托马斯・F・加里蒂
鲁本・加尔松
詹姆斯・H・格尼
吉姆・D・伊萨克
洛厄尔・G・约翰逊
罗伯特肯尼利
EG “Al” Kiener
Joseph L. Koepfinger*
斯蒂芬・R・兰伯特
吉姆・洛戈塞蒂斯
唐纳德・C・洛夫里
L.布鲁斯・麦克朗
路易斯-弗朗索瓦・保罗
罗纳德・彼得森
杰拉尔德・H・彼得森
约翰・B・波西
加里・S・罗宾逊
汉斯・E・温里希
唐纳德・W・齐普斯

* 名誉会员

克里斯汀・M・迪特曼
IEEE 标准项目编辑

IEEE 标准 802.11c-1998

IEEE Std 802.11c-1998 添加了将 IEEE 802.11 MAC 参数映射到 ISO/IEC 15802-3（IEEE Std 802.1D）参数所需的信息。

參與者

在 IEEE 802.11c 标准草案被送交赞助商投票时，IEEE 802.11 工作组有以下投票成员：

维克多・海斯，*椅子*
维多利亚・M・庞奇尼，*任务组主席*

杰夫・阿布拉莫维茨
基思・B・阿蒙森
卡尔・F・安德伦
青柳一宏
大卫・巴格比
菲尔・贝朗格
约翰・比迪克
西蒙・布莱克
杨・波尔
罗纳德・布罗克曼
韦斯利・布罗德斯基
约翰・H・卡法雷拉
纳夫塔利・查亚特
肯・克莱门茨
维姆・迪普斯特拉滕
达罗尔・德雷珀
彼得・艾克莱辛
达尔文・恩格威尔
约翰・法卡特塞利斯
杰夫・菲舍尔
马修・菲舍尔
迈克尔・菲舍尔
乔治・菲舍尔
约翰・费舍尔
五地元博
蒂姆・戈弗雷
简・哈格
卡尔・汉内斯塔德
罗伯特・海勒
马丁・霍本
杜安・赫恩
池田昌之
理查德・贾
唐纳德・C・约翰逊
唐木伸夫
迪安・M・川口
斯图尔特・J・克里
正木勋
吉姆・麦克唐纳
吉恩・米勒
三浦彰
森雅治
森仓昌弘
拉维・P・纳拉马蒂
科林・奈勒
理查德・范尼
鲍勃・奥哈拉
大泽友树
冈上和弘
理查德・H・潘恩
艾尔・佩特里克
鲍勃・范
斯坦利・A・雷布尔
威廉・羅伯茨
肯特・G・罗林斯
奥伦・罗森菲尔德
迈克尔・罗森伯格
克莱门斯・CW・鲁佩尔
钱多斯・雷宾斯基
阿尼尔・K・桑沃卡
罗伊・西布林
麦克・希巴
托马斯・西普
唐纳德・I・斯隆
高梨均志
戸口悟
樱桃汤姆
迈克・特罗姆鲍尔
汤姆・索洛吉安尼斯
沙羅什・維蘇納
魏年昌
哈里・沃斯特尔
蒂莫西・齐默尔曼
约翰尼・茨威格
吉姆・泽伦

主要贡献来自 Henri Moelard。

802.11c 投票委员会成员如下：

基特・阿苏尔
托马斯・W・贝利
彼得・K・坎贝尔
詹姆斯・T・卡洛
戴维・E・卡尔森
布莱恩・J・凯西
纳夫塔利・查亚特
罗伯特・克劳德
维姆・迪普斯特拉滕
托马斯・J・迪宁
克里斯托斯・杜利格里斯
保罗・S・伊士曼
约翰・E・埃姆里希
菲利普・H・恩斯洛
范长新
约翰・W・芬德里希
迈克尔・A・菲舍尔
哈维・A・弗里曼
罗伯特・加利亚诺
高塔姆・加莱
帕特里克・S・戈尼亚
胡里奥・冈萨雷斯・桑斯
克里斯・G・盖伊
J. Scott Haugdahl
维克・海耶斯
唐纳德・N・海尔曼
亨利・霍伊特
拉杰・贾恩
托尼・杰弗里

A. 卡默曼
迪安・M・川口
爱德华・R・凯利
彼得・M・凯利
加里・凯斯勒
金龙范
史蒂芬・巴顿・克鲁格
约瑟夫库伯勒
简・库里斯
兰斯・M・利奇
李在镕
兰道夫・S・利特尔
罗纳德・马哈尼
彼得・马蒂尼
贝内特・迈耶
吉恩・E・米利根
戴维・S・米尔曼
沃伦・梦露
约翰・E・蒙塔古
韦恩・D・莫耶斯
西蒙・穆勒
长沼健
保罗・尼科利奇
罗伯特・奥哈拉
唐纳・奥马奥尼
查尔斯・奥斯特莱彻
杨噢
约翰・M・奥塞普丘克
罗杰・潘丹达

罗纳德・彼得森
约翰・R・皮肯斯
阿尔贝托・普罗富莫
维克拉姆・普尼
费尔南多・拉莫斯
詹姆斯・A・伦弗洛
艾弗里特・里格斯比三世
爱德华・Y・罗歇
詹姆斯・W・罗姆莱恩
弗洛伊德・E・罗斯
迈克尔・罗森伯格
克里斯托弗・鲁兰德
阿尼尔・K・桑沃卡
诺曼・施奈德温德
詹姆斯・E・舒斯勒
里奇・塞弗特
利奥・辛托宁
帕特里夏・泰勒
迈克・特罗姆鲍尔
马克・雷内・内田
萨罗什・N・维苏纳
约翰・维亚普拉纳
詹姆斯・沃希斯
巴里・M・沃恩布鲁克
杨千里
袁奥伦
克里斯・泽格林
乔纳森・M・茨威格

当 IEEE-SA 标准委员会于 1998 年 9 月 16 日批准 IEEE 标准 802.11c 时，其成员如下：

理查德・J・霍勒曼，*椅子*　　**唐纳德・N・海尔曼，***副主席*

朱迪思・戈尔曼，*秘书*

萨蒂什・K・阿加瓦尔
克莱德・R・坎普
詹姆斯・T・卡洛
加里・R・恩格曼
哈罗德・E・爱泼斯坦
杰伊・福斯特*
托马斯・F・加里蒂
鲁本・加尔松

詹姆斯・H・格尼
吉姆・D・伊萨克
洛厄尔・G・约翰逊
罗伯特肯尼利
EG “Al” Kiener
Joseph L. Koepfinger*
斯蒂芬・R・兰伯特
吉姆・洛戈塞蒂斯
唐纳德・C・洛夫里

L.布鲁斯・麦克朗
路易斯-弗朗索瓦・保罗
罗纳德・彼得森
杰拉尔德・H・彼得森
约翰・B・波西
加里・S・罗宾逊
汉斯・E・温里希
唐纳德・W・齐普斯

* 名誉会员

克里斯汀・M・迪特曼
IEEE 标准项目编辑

抽象的：此标准 1993 年版中引入的媒体访问控制 (MAC) 桥接概念已得到扩展，以定义桥接 LAN 中的附加功能，旨在提供加速流量功能，支持在 LAN 环境中传输时间关键型信息；并提供支持在 LAN 环境中动态使用组 MAC 地址的过滤服务。

关键词：局域网、MAC 桥接管理、MAC 桥接、媒体访问控制 (MAC) 桥接、多播地址过滤、流量类别加速

美国电气电子工程师学会，纽约州纽约市东 47 街 345 号，邮
编 10017-2394

打印： 国际标准书号 0-7381-0329-2 SH94651
PDF： ISBN 0-7381-1416-2 SS94651

内容

1. 概述 1

1.1 简介 1
1.2 范围 1

2. 参考文献 3

3. 定义 6

3.1 桥接局域网 6
3.2 加速交通 6
3.3 群组 6
3.4 IEEE 802 局域网（LAN） 6

4. 缩写 7

5. 一致性 8

5.1 静态一致性要求 8
5.2 选项 8
5.3 协议实施一致性声明（PICS）形式 9
5.4 建议 9
5.5 MAC 特定的桥接方法 9

6. MAC 服务支持 10

6.1 MAC 服务支持 10
6.2 MAC 服务的保留 10
6.3 服务质量维护 11
6.4 MAC 桥内提供的内部子层服务 15
6.5 特定MAC程序对内部子层服务的支持 17
6.6 桥接 LAN 中的过滤服务 25

7. 操作原理 29

7.1 桥梁操作 29
7.2 桥梁架构 30
7.3 运作模式 32
7.4 端口状态、活动端口和活动拓扑 35
7.5 帧接收 36
7.6 帧传输 37
7.7 转发过程 37
7.8 学习过程 42
7.9 过滤数据库 42
7.10 桥接协议实体和 GARP 协议实体 49
7.11 桥梁管理 50
7.12 寻址 50

8. 生成树算法和协议 58

8.1 算法需满足的要求 58
8.2 MAC 桥接器的要求 58

8.3 概述 59
8.4 港口国家 65
8.5 协议参数和定时器 67
8.6 程序要素 74
8.7 协议的操作 83
8.8 桥接协议实体的管理 85
8.9 程序模型 87
8.10 性能 107

9. 桥接协议数据单元 (BPDU) 的编码 110

9.1 结构 110
9.2 参数类型的编码 110
9.3 BPDU 格式及参数 111

10. GARP 多播注册协议 (GMRP) 114

10.1 目的 114
10.2 运作模式 114
10.3 GMRP 应用程序的定义 117
10.4 符合 GMRP 119

11. GMRP 的“C”代码实现示例 122

11.1 目的 122
11.2 GMRP 应用程序特定头文件 123
11.3 GMRP应用程序代码 128

12. 通用属性注册协议（GARP） 135

12.1 目的 135
12.2 GARP 操作概述 135
12.3 GARP 架构 137
12.4 GARP 需满足的要求 142
12.5 GARP 参与者之间的互操作性要求 142
12.6 与 GARP 应用程序的一致性 144
12.7 GARP 协议操作概述 144
12.8 状态机描述 150
12.9 行政控制 155
12.10 程序 157
12.11 GARP 协议数据单元的结构和编码 162
12.12 计时器值、粒度和关系 166
12.13 程序模型 167
12.14 互操作性考虑 167

13. GARP 的 C 代码实现示例 168

13.1 目的 168
13.2 GARP 应用程序独立头文件 169
13.3 GARP 应用独立代码 182

14. 桥梁管理 206

14.1 管理功能 206

14.2 管理对象 207
14.3 数据类型 207
14.4 桥梁管理实体 208
14.5 MAC 实体 211
14.6 转发流程 211
14.7 过滤数据库 214
14.8 桥接协议实体 218
14.9 GARP 实体 221

15. 管理协议 224

15.1 将操作映射到 LMMS 服务 224
15.2 管理对象包含结构 226
15.3 MAC 桥接 DLE 管理对象类定义 238
15.4 端口管理对象类定义 243
15.5 选择性翻译表条目管理对象类定义 249
15.6 永久数据库管理对象类定义 250
15.7 过滤数据库管理对象类定义 251
15.8 数据库条目管理对象类定义 252
15.9 GARP 应用管理对象类定义 254
15.10 GARP 属性类型管理对象类定义 255
15.11 GARP 属性管理对象类定义 256
15.12 GARP 定时器管理对象类定义 257
15.13 ASN.1 定义 259

16. 桥梁性能 262

16.1 保证端口过滤率 262
16.2 保证桥接中继速率 262

附件 A (规范性) PICS 形式 263

A.1 简介 263
A.2 缩写和特殊符号 263
A.3 PICS 表格填写说明 264
A.4 ISO/IEC 15802-3 的 PICS 形式 266
A.5 主要功能和选项 267
A.6 帧的中继和过滤 268
A.7 过滤数据库中过滤条目的维护 270
A.8 寻址 272
A.9 生成树算法 273
A.10 桥梁管理 276
A.11 性能 277
A.12 GARP 和 GMRP 278

附件 B （资料性附录） 计算生成树参数 280

B.1 概述 280
B.2 缩写和特殊符号 280
B.3 计算 280
B.4 参数范围的选择 285

附件C（规范性）源路由透明桥操作 288

C.1 概述 288
C.2 支持 MAC 服务 291
C.3 操作原理 295
C.4 桥梁管理 310
C.5 管理协议 315

附件D（规范性）源路由透明桥操作的PICS形式 316

D.1 简介 316
D.2 帧的中继和过滤 316
D.3 网桥编号和 LAN ID 317
D.4 桥梁管理 317

附件 E （规范性） 对象标识符值的分配 318

E.1 简介 318
E.2 分配表 318

附件 F （资料性） 目标拓扑、迁移和互操作性 324

F.1 目标拓扑 324
F.2 迁移注意事项 325
F.3 与高层多播协议的互操作性及相关问题 327

附件 G （资料性附录） 保护 MAC 桥接器中 FCS 字段的完整性 329

G.1 背景 329
G.2 CRC 和 FCS 背后的基本数学思想 329
G.3 检测无损电路方法 331
G.4 FCS 的算法修改 332
G.5 结论 335

附件 H （资料性附录）交通等级加速和动态设计考虑
多播过滤 336

H.1 通用属性注册协议 (GARP) 336
H.2 用户优先级和流量类别 352

数字

图 6-1 MAC子层的内部组织 10
图 7-1 桥接局域网 31
图 7-2 桥接端口 32
图 7-3 桥梁结构 32
图 7-4 中继 MAC 帧 33
图 7-5 观察网络流量 34
图 7-6 桥间协议的操作 34
图 7-7 GARP 协议的运行 35
图 7-8 转发流程详细操作说明 38
图 7-9 更高层实体使用的连接点与 MAC 中继实体 54
图 7-10 控制信息对转发路径的影响 56
图 7-11 每个端口的连接点 56
图 7-12 单点连接—允许中继 57
图 7-13 单点连接——不允许使用继电器 57
图 8-1 活动拓扑 60
图 8-2 生成树 61
图 8-3 港口国家 64
图 9-1 配置 BPDU 参数及格式 112
图 9-2 拓扑变化通知 BPDU 参数和格式 113
图 10-1 有向图示例 115
图 12-1 从一个工作站传播属性值的示例 136
图 12-2 来自两个站的属性值传播示例 137
图 12-3 活动拓扑子树的形成示例 138
图 12-4 GARP 架构 139
图 12-5 GID 架构 140
图 12-6 GARP PDU 主要组件的格式 163
图 12-7 GARP 定时关系 167
图 15-1 管理对象包含结构 237
图 C-1 SRT Bridge 操作逻辑 296
图 C-2 源路由桥的元素 298
图 C-3 路由信息字段的结构 298
图 C-4 路由信息字段 299
图 C-5 RC 字段第二个八位字节上的基本和扩展 LF 位 300
图 C-6 最大框架基值及其原理 300
图 C-7 最大帧值的范围 301
图 C-8 路线描述符字段 301
图 C-9 重复桥号测试图解 308
图 F-1 生成树拓扑示例 324
图 H-1 拓扑：叶 LAN 场景 338
图 H-2 拓扑：主干 LAN 场景 346
图 H-3 拓扑：共享媒体 LAN 段场景 349

表格

表 7-1 用户优先级再生 37
表 7-2 建议的用户优先级到流量类别映射 40
表 7-3 出站访问优先级 41
表 7-4 老化时间参数值 45
表 7-5 结合单个 MAC 地址的静态和动态过滤条目 48
表 7-6 结合静态过滤条目和组注册表条目“所有群组地址”和“所有未注册的群组地址” 48
表 7-7 转发或过滤特定组 MAC 地址 49
表 7-8 标准 LLC 地址分配 51
表 7-9 保留地址 52
表 7-10 解决桥梁管理问题 53
表 8-1 最大桥梁直径 108
表 8-2 传输和传输延迟 108
表 8-3 生成树算法计时器值 108
表 8-4 桥接和端口优先级参数值 109
表 8-5 路径成本参数值 109
表 12-1 GARP 应用程序地址 143
表 12-2 申请人：州摘要 148
表 12-3 申请人状态表 152
表 12-4 注册商状态表 153
表 12-5 保留所有状态表 153
表 12-6 联合申请人/注册处州 153
表 12-7 申请人/注册商组合状态表 154
表 12-8 仅限申请人状态机 155
表 12-9 简单申请人状态机 156
表 12-10 GARP 定时器参数值 166
表 15-1 桥梁管理实体操作到 LMMS 服务的映射 225
表 15-2 将转发流程操作映射到 LMMS 服务 225
表 15-3 将过滤数据库操作映射到 LMMS 服务 225
表 15-4 桥接协议实体操作到 LMMS 服务的映射 225
表 15-5 GARP 参与者操作与 LMMS 服务的映射 226
表 15-6 将选择性翻译表操作映射到 LMMS 服务 226
表 15-7 发现桥参数映射 (14.4.1.1) 227
表 15-8 读取桥接参数的映射 (14.4.1.2) 227
表 15-9 设置桥梁名称参数的映射 (14.4.1.3) 227
表 15-10 重置桥接参数映射（14.4.1.4） 228
表 15-11 读取端口参数映射 (14.4.2.1) 228
表 15-12 设置端口名称参数映射（14.4.2.2） 228
表 15-13 读取转发端口计数器参数映射 (14.6.1.1) 228
表 15-14 读取端口默认用户优先级参数映射 (14.6.2.1) 229
表 15-15 设置端口默认用户优先级参数映射（14.6.2.2） 229
表 15-16 读取端口用户优先级再生表参数映射（14.6.2.3） 229
表 15-17 设置端口用户优先级再生表参数映射（14.6.2.2） 229
表 15-18 读取端口流量类别表参数映射 (14.6.3.1) 230
表 15-19 设置流量类别表参数映射（14.6.3.2） 230
表 15-20 读取出站访问优先级表参数映射（14.6.3.3） 230
表 15-21 读取过滤数据库参数映射 (14.7.1.1) 231
表 15-22 设置过滤数据库老化时间参数映射（14.7.1.2） 231
表 15-23 读取永久数据库参数映射 (14.7.5.1) 231
表 15-24 创建过滤条目参数映射（14.7.6.1） 231
表 15-25 删除过滤条目参数映射（14.7.6.2） 232
表 15-26 读取过滤条目参数映射（14.7.6.3） 232

表 15-27 读取过滤条目范围参数映射（14.7.6.4） 232
表 15-28 读取桥接协议参数的映射（14.8.1.1） 233
表 15-29 设置桥接协议参数的映射（14.8.1.2） 233
表 15-30 读取端口参数映射（14.8.2.1） 233
表 15-31 强制端口状态参数映射（14.8.2.2） 234
表 15-32 设置端口参数的映射（14.8.2.3） 234
表 15-33 读取选择性翻译表条目范围参数映射
（ISO/IEC 11802-5） 234
表 15-34 读取 GARP 定时器参数映射（14.9.1.1） 234
表 15-35 设置 GARP 定时器参数映射（14.9.1.2） 235
表 15-36 读取 GARP 申请人控制参数的映射 (14.9.2.1) 235
表 15-37 设置 GARP 申请人控制参数的映射 (14.9.2.2) 235
表 15-38 读取 GARP 状态参数映射（14.9.3.1） 236
表 C-1 最大框架基值 300
表 C-2 最大帧扩展值 301
表 C-3 专门路由的帧转发状态表 303
表 C-4 所有路由资源管理器帧转发状态表 305
表 C-5 生成树资源管理器帧转发状态表 307
表 H-1 初始加入：无数据包丢失 338
表 H-2 初始加入：第一个加入 PDU 的数据包丢失 339
表 H-3 初始加入：第二次加入 PDU 时出现数据包丢失 340
表 H-4 最后离开：无数据包丢失 341
表 H-5 最后离开：第一个离开 PDU 的数据包丢失 342
表 H-6 单个成员全部离开/重新加入: 无数据包丢失 343
表 H-7 单个成员离开/重新加入：LeaveAll PDU 上的数据包丢失 344
表 H-8 单个成员离开/重新加入: 首次加入 PDU 时数据包丢失 345
表 H-9 主干网初始加入: 无数据包丢失 346
表 H-10 主干网初始加入：首次加入 PDU 时的数据包丢失 347
表 H-11 主干网初始加入：两个网桥同时加入，第一次加入时出现数据包丢失 348 表 H-12
共享媒体：第三方无需发送 GARP 消息即可加入/离开 350
表 H-13 共享媒体：为三位参与者保留顺序 351
表 H-14 流量类型到流量类别的映射 353
表 H-15 流量类型缩写 355
表 H-16 定义流量类型 355

信息技术 系统间电信和信息交换 局域网和城域网 通用规范

第 3 部分：媒体访问控制 (MAC) 桥接器

1. 概述

1.1 简介

所有类型的 IEEE 802 局域网（或 LAN；参见 3.4）都可以使用 MAC 桥连接在一起。每个单独的 LAN 都有自己独立的 MAC。创建的桥接 LAN 允许将连接到不同 LAN 的站互连，就像它们连接到单个 LAN 一样，尽管它们实际上连接到具有各自 MAC 的不同 LAN。MAC 桥在 MAC 服务边界以下运行，并且对于在逻辑链路控制 (LLC) 子层或网络层（ISO/IEC 7498-1：1994）中运行在此边界之上的协议是透明的[1]）一个或多个MAC桥接器的存在可能导致MAC子层提供的服务质量的差异；正是由于这样的差异，MAC桥接器的操作才不是完全透明的。

桥接 LAN 可以提供

a) 不同MAC类型LAN上的站之间的互连；
b)有效增加局域网的物理范围、允许的附件数量或总体性能；

c) 出于管理或维护原因对物理 LAN 进行分区。

注——与源路由透明桥接操作有关的范围、定义、参考和一致性要求可在附件 C.1 中找到。

1.2 范围

为了使使用 IEEE 802 MAC 服务的数据处理设备在使用不同或相同媒体访问控制方法的互连 IEEE 802 LAN（见 3.4）的支持下实现兼容互连，本标准规定了 MAC 桥接器操作的通用方法。为此，它

a) 将桥接功能定位在MAC子层的架构描述中。
b) 从MAC服务的支持和保存、服务质量的维护等方面定义MAC桥的运行原则。

c) 指定各个 LAN 向网桥中提供帧中继的媒体访问方法独立功能提供的 MAC 内部子层服务。

d) 识别桥接器要执行的功能，并根据提供这些功能的流程和实体提供桥接器内部操作的架构模型。

e) 建立桥接局域网中网桥之间的协议要求以配置网络，并指定生成树活动拓扑的分布式计算。

f) 指定网桥协议数据单元 (BPDU) 的编码。
g) 建立桥接局域网中桥接管理的要求，识别管理对象并定义管理操作。

[1]关于参考的信息可以参见第 2 条。

h) 指定如何使用 ISO/IEC 15802-2: 1995 提供的协议和架构描述使远程管理器能够执行管理操作。

i) 指定性能要求并推荐桥梁操作参数的默认值和适用范围。

j) 规定了符合本标准的设备必须满足的要求。
k) 指定使用 MAC 特定桥接方法的标准。

此标准规定了直接连接到 IEEE 802 LAN 的 MAC 桥接器的操作，如所实施的 MAC 技术或技术的相关 MAC 标准中所规定的那样。

远程网桥的规范超出了本标准的范围，远程网桥使用广域网 (WAN) 介质将 LAN 互连以便在网桥之间传输帧。

注：远程 MAC 桥接在 ISO/IEC 15802-5: 1997 [ANSI/IEEE Std 802.1G，1997 年版] 中指定。

2. 参考文献

以下标准包含的条款通过本文引用而成为 ISO/IEC 15802 本部分的条款。出版时，所示版本有效。所有标准都可能修订，鼓励根据 ISO/IEC 15802 本部分达成协议的各方调查应用下列所示标准的最新版本的可能性。ISO 和 IEC 成员维护当前有效的国际标准的登记册。

ANSI X3.159-1989，美国国家信息系统标准——编程语言——C。[2]

IEEE Std 802-1990，IEEE 局域网和城域网标准：概述和架构。[3]

IEEE Std 802.1F-1993，IEEE 局域网和城域网标准：IEEE 802 管理信息的通用定义和程序。

IEEE Std 802.3，1998 年版，信息技术 - 系统间电信和信息交换 - 局域网和城域网 - 具体要求 - 第 3 部分：带冲突检测的载波侦听多路访问 (CSMA/CD) 访问方法和物理层规范。

IEEE Std 802.9a-1995，IEEE 局域网和城域网标准：介质访问控制 (MAC) 层和物理 (PHY) 层的综合业务 (IS) LAN 接口补充：ISLAN 16-T 规范。

IEEE Std 802.11-1997，信息技术 - 系统间电信和信息交换 - 局域网和城域网 - 具体要求 - 第 11 部分：无线局域网介质访问控制 (MAC) 和物理层 (PHY) 规范。[4]

IETF INTERNET-DRAFT，互联网组管理协议 (IGMP)，第 2 版，1997 年 1 月 20 日[5]。

IETF RFC 1493，Decker、Langille、Rijsinghani 和 McCloughrie，《桥接管理对象的定义》，1993 年 7 月[6]。

ISO 6937-2: 1983，信息处理 – 文本通信用编码字符集 – 第 2 部分：拉丁字母和非字母图形字符。[7]

ISO/IEC 7498-1: 1994，信息处理系统 — 开放系统互连 — 基本参考模型 — 第 1 部分：基本模型。

---

[2]您可从美国国家标准协会销售部 (地址：美国纽约州纽约市西 42 街 11 号 13 楼，邮编 10036) 购买 ANSI 出版物。

[3]IEEE 出版物可从电气电子工程师协会 (445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331, USA) 获取。

[4]这项标准的国际标准草案 (ISO/IEC DIS 8802-11) 正在起草中。如需了解该 DIS 的状态，请联系 ISO 中央秘书处，地址：1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse；或美国国家标准协会销售部，地址：11 West 42nd Street, 13th Floor, New York, NY 10036, USA。

[5]互联网草案可以在 IETF 网站 http://www.ietf.cnri.reston.va.us/home.html 上查阅，或者拨打 InterNIC 电话 1-800-444-4345 了解有关通过邮件接收副本的信息。

[6]可以通过 FTP 在 ds.internic.net/rfc/rfcnnnn.txt 上检索 Internet RFC（其中 nnnn 是标准的出版号，例如 1493），或致电 InterNIC 电话 1-800-444-4345 以获取有关通过邮件接收副本的信息。

[7]ISO 和 ISO/IEC 文件可从以下地址获取：ISO 中央秘书处，地址为 1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse；或美国国家标准协会销售部，地址为 11 West 42nd Street, 13th Floor, NewYork, NY 10036, USA。

ISO/IEC 8802-2: 1998 [ANSI/IEEE Std 802.2, 1998 版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 具体要求 – 第 2 部分：逻辑链路控制。[8]

ISO/IEC 8802-4: 1990 [ANSI/IEEE Std 802.4-1990]，信息处理系统 – 局域网 – 第 4 部分：令牌传递总线访问方法和物理层规范。

ISO/IEC 8802-5: 1998 [ANSI/IEEE Std 802.5, 1998 版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 具体要求 – 第 5 部分：令牌环访问方法和物理层规范。

ISO/IEC 8802-6: 1994 [ANSI/IEEE Std 802.6, 1994 年版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 具体要求 – 第 6 部分：分布式队列双总线 (DQDB) 访问方法和物理层规范。

ISO/IEC 8802-9: 1996 [ANSI/IEEE Std 802.9, 1996 年版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 具体要求 – 第 9 部分：介质访问控制 (MAC) 层和物理 (PHY) 层的综合业务 (IS) LAN 接口。

ISO/IEC 8802-12: 1998 [ANSI/IEEE Std 802.12, 1998 版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 具体要求 – 第 12 部分：需求优先接入方法、物理层和中继器规范。

ISO/IEC 8824: 1990，信息技术 — 开放系统互连 — 抽象语法符号一（ASN.1）规范（暂时保留版）。

ISO/IEC 8825: 1990，信息技术 — 开放系统互连 — 抽象语法符号一（ASN.1）基本编码规则规范（暂时保留版）。

ISO/IEC 9595: 1998，信息技术 — 开放系统互连 — 通用管理信息服务。

ISO 9314-2: 1989，信息处理系统 — 光纤分布式数据接口 — 第 2 部分：FDDI 令牌环媒体访问控制 (MAC)。

ISO/IEC 9596-1: 1991，信息技术 — 开放系统互连 — 通用管理信息协议 — 第 1 部分：规范。

ISO/IEC TR 11802-2: 1997，信息技术 — 系统间电信和信息交换 — 局域网和城域网 — 技术报告和指南 — 第 2 部分：标准组 MAC 地址。

ISO/IEC 11802-5: 1997 [ANSI/IEEE Std 802.1H，1997 年版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 技术报告和指南 – 第 5 部分：局域网中的以太网 V2.0 媒体访问控制 (MAC) 桥接。

---

[8]ISO [IEEE] 和 ISO/IEC [IEEE] 文档可从以下地址获取：ISO 中央秘书处，地址为 1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse；或从电气电子工程师协会，地址为 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331, USA。

ISO/IEC 15802-1: 1995，信息技术 — 系统间电信和信息交换 — 局域网和城域网 — 通用规范 — 第 1 部分：介质访问控制 (MAC) 服务定义。

ISO/IEC 15802-2: 1995 [ANSI/IEEE Std 802.1B，1995 年版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 通用规范 – 第 2 部分：LAN/MAN 管理。

ISO/IEC 15802-5: 1998 [ANSI/IEEE Std 802.1G，1998 版]，信息技术 – 系统间电信和信息交换 – 局域网和城域网 – 通用规范 – 第 5 部分：远程介质访问控制 (MAC) 桥接。

3.定义

就本标准而言，适用以下术语和定义：

3.1 桥接局域网

通过 MAC 桥连接起来的单个 IEEE 802 LAN 的串联。

3.2 加速交通

由于抖动、延迟或吞吐量限制，或由于管理策略而需要优先处理的流量。

3.3 组

集团员工

a) 组 MAC 地址；以及

b) 定义会员特征的一组属性；以及

c) 一组属性，用于定义网桥针对发往该组 MAC 地址成员的帧的转发/过滤行为；

一组终端站都希望接收发往该组 MAC 地址的信息。此类终端站组的成员被称为*小组成员。*

据称，一个团体*存在*如果与该组关联的属性在网桥的过滤数据库的条目中可见，或者在描述该组状态的 GARP 状态机中可见，则称该组*有成员*如果组的属性表明可以通过桥的特定端口访问该组的成员。

注——组成员可能希望接收的信息的一个例子是多播视频数据流。

3.4 IEEE 802 局域网（LAN）

IEEE 802 LAN（本文中也简称为 LAN）是一种 LAN 技术，可提供与 ISO/IEC 15802-1 中定义的 MAC 服务等效的 MAC 服务。IEEE 802 LAN 包括 IEEE 标准 802.3（CSMA/CD）、ISO/IEC 8802-4（令牌总线）、ISO/IEC 8802-5（令牌环）、ISO/IEC 8802-6（DQDB）、ISO/IEC 8802-9（IS-LAN）、IEEE 标准 802.11（无线）、ISO/IEC 8802-12（需求优先级）和 ISO 9314-2（FDDI）LAN。

注：ISO/IEC 8802-6 的无连接服务部分提供了等效的 MAC 服务。本标准仅涵盖 48 位寻址 ISO/IEC 8802-6 PDU 的桥接。60 位寻址 ISO/IEC 8802-6 PDU 的桥接不在本标准的范围内。

4. 缩写

本标准中使用了下列缩写：

业务规则协议数据单元	桥接协议数据单元
CRC	循环冗余校验
胎儿干细胞	帧校验序列
区域空气质量评估计划	通用属性注册协议
协议数据单元	GARP 协议数据单元
性别识别号码	GARP信息声明
全球化知识产权组织	GARP 信息传播
通用航空采购计划	GARP 多播注册协议
因特网工程任务组	互联网工程任务组
组播管理协议	互联网组管理协议
苹果	媒体访问控制

5. 一致性

5.1 静态一致性要求

声称符合本标准的 MAC Bridge 应

a) 符合其端口实施的 MAC 技术的相关 MAC 标准，如 6.5 所述；

b) 符合 ISO/IEC 8802-2 标准，可实现 7.3 和 7.1.2 要求的支持 1 类操作的 LLC；

c) 如 7.1 中所述并在 7.5、7.6 和 7.7 中指定的中继和过滤帧，用于支持基本过滤服务，以及与每个出站端口相关联的单个流量类别；

d) 维护按照 7.1 所述以及 7.8 和 7.9 中指定的做出帧过滤决策所需的信息，以支持基本过滤服务。

e) 使用过滤数据库（7.9）中以下参数的规定值：

1) 过滤数据库大小，过滤数据库中可容纳的最大条目数。

2)永久数据库大小，永久数据库中可保存的最大条目数。

f) 符合7.12规定的寻址规定。

g) 提供

1) 如果未使用 48 位通用管理地址，则分配组 MAC 地址来识别桥接协议实体的方法；

2) 根据 8.2 中生成树算法和协议的运行要求，为桥接器的每个端口提供一个不同的端口标识符，如 8.5 中所述。

h) 按照8.7规定，实现第8条描述的生成树算法和协议。

i) 以下参数不得超过 8.10.2 中给出的值：

1）最大桥接传输延迟

2）Maximum Message Age 增量估计过高

3）BPDU最大传输延迟

j) 对以下参数使用表8-3中给出的值：

1）保持时间

k) 按照第 9 条规定对传输的 BPDU 进行编码，并验证接收的 BPDU。

l) 指定每个桥接端口的保证端口过滤率、桥接的保证桥接中继率以及相关的时间间隔*特遣队*和*贸易*如第 16 条所述。在规定参数范围内的桥接操作不得违反本标准的任何其他一致性规定。

5.2 选项

声称符合本标准的 MAC Bridge 可以

a) 提供控制转发帧优先级映射的能力，如6.4所述，7.5.1 和 7.7.3。

b) 提供读取和更新过滤数据库和永久数据库的能力，如7.9中所述。

c) 提供设置老化时间的能力，如 7.9 中所述。提供此功能的网桥应实现表 7-4 中规定的全部值范围。

d) 实现 7.12.4 中指定的桥接管理寻址可选规定、7.12.5 中指定的桥接地址与桥接端口关联可选规定、以及 7.9.6 中指定的永久数据库中预处理组地址可选规定。

e) 提供为以下参数分配值的能力，以允许配置生成树活动拓扑：

1）桥接优先级
2）端口优先级
3）每个端口的路径成本
提供此功能的桥接器应实现 8.10.2 和表 8-4 和表 8-5 中指定的值范围。

f) 提供设置生成树算法和协议的以下参数值的能力：

1）桥梁最大年龄
2）桥接问候时间
3）桥接转发延迟
提供此功能的桥接器应实现 8.10.2 和表 8-3 中指定的值范围。

g) 支持桥的管理。声称支持管理的桥应支持第 14 条中定义的所有管理对象和操作。

h) 支持使用远程管理协议。声称支持远程管理的网桥应

1) 说明支持哪些远程管理协议标准或规范。
2) 说明远程管理协议支持使用哪些管理对象定义和编码的标准或规范。

i) 支持禁用拓扑变化检测的能力，如8.5.5.10所述。
j) 提供与一个或多个出站端口相关的多个流量类别，如 7.7 所述；
k) 提供针对单个MAC地址的静态过滤条目的能力，以指定转发或过滤行为应基于动态过滤信息，如7.9.1中所述；
l) 按照 7.1 中所述并在 7.5、7.6、7.7 和 7.9 中指定的中继和过滤帧来支持扩展过滤服务，并实现 GARP 多播注册协议（GMRP），详见 10.4.1 中定义的一致性要求；

m) 提供管理中继帧优先级的能力，如 6.4、7.5.1 和 7.7.3 中所述。

5.3 协议实施一致性声明（PICS）形式

声称符合本标准的实施的供应商应填写附件 A 中提供的 PICS 形式表的副本，并提供识别供应商和实施所需的信息。

5.4 建议

5.4.1 管理

强烈建议支持第14章规定的管理对象和管理操作。

5.5 MAC 特定的桥接方法

可能存在特定于 MAC 的桥接方法。在同一 LAN 上使用特定于 MAC 的桥接方法和本标准中规定的方法应

a) 不妨碍桥接局域网中各站之间的通信。
b) 保留MAC服务。
c) 保留每种桥接方法在其自身领域内的特性。
d) 提供两种桥接技术在 LAN 上同时共存而不会产生不利影响的能力。

6. MAC 服务支持

MAC 桥接器通过在桥接 LAN 的各个 MAC 之间中继和过滤帧，将组成桥接 LAN 的各个 IEEE 802 LAN 互连。桥接功能在 MAC 子层中的位置如图 6-1 所示。

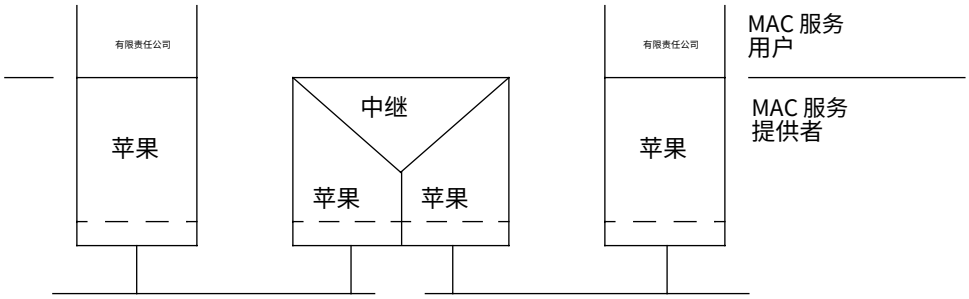


图 6-1—MAC 子层的内部组织

本节讨论桥接局域网中服务提供的以下方面：

a) 向终端站提供MAC服务；
b) 保留MAC服务；
c) 维护服务质量；
d) 提供MAC桥内的内部子层服务；
e) 通过特定的MAC程序支持内部子层服务；
f) 过滤服务。

6.1 支持MAC服务

为桥接 LAN 连接的终端站提供的 MAC 服务是 ISO/IEC 15802-1 中定义的（未确认的）无连接模式 MAC 服务。MAC 服务被定义为许多特定 MAC 服务共有的功能的抽象；它描述了通过在 MAC 服务接入点发出的 MA-UNITDATA 请求原语和相应的 MA-UNITDATA 指示原语在源和目标终端站之间传输用户数据。每个 MA-UNITDATA 请求和指示原语都有四个参数：目标地址、源地址、MAC 服务数据单元 (MSDU) 和优先级。

桥接操作方式可最大程度地提高 MAC 服务对终端站的可用性，并有助于维护桥接 LAN。因此，最好能够在桥接 LAN 中配置桥接：

a) 以便在终端站之间提供冗余路径，使得桥接 LAN 能够在发生组件故障（桥接或 LAN 的组件）时继续提供服务。
b) 鉴于桥接 LAN 组件的可用性，因此终端站之间支持的路径是可预测和可配置的。

6.2 MAC 服务的保存

由 MAC 桥接器互连的 LAN 组成的桥接 LAN 提供的 MAC 服务与单个 LAN 提供的 MAC 服务类似 (6.3)。因此，

a) 网桥不直接由通信终端站寻址，除非作为用于管理目的的终端站：在终端站之间传输的帧在其目标地址字段中携带对等终端站的 MAC 地址，而不是网桥的 MAC 地址（如果有）。

b) 所有 MAC 地址在桥接 LAN 内必须是唯一的。

c) 终端站的 MAC 地址不受桥接 LAN 的拓扑和配置的限制。

6.3 服务质量维护

MAC 子层为连接到 LAN 或桥接 LAN 的终端站提供 MAC 服务。桥接支持的 MAC 服务质量不应明显低于单个 LAN 提供的服务质量。要考虑的服务质量参数与以下内容相关：

a) 服务可用性
b) 帧丢失
c) 帧乱序
d) 帧复制
e）帧经历的传输延迟
f) 帧寿命
g) 未检测帧错误率
h) 支持的最大服务数据单元大小
i) 用户优先级
j) 吞吐量

6.3.1 服务可用性

服务可用性是按提供 MAC 服务的总时间比例来衡量的。网桥的运行可以提高或降低服务可用性。

通过自动重新配置桥接 LAN 可以提高服务可用性，从而避免在数据路径中使用故障组件（例如中继器、电缆或连接器）。桥接器本身发生故障、桥接器拒绝服务或桥接器进行帧过滤可能会降低服务可用性。

当自动重新配置发生时，桥接器可能会拒绝服务并丢弃帧（6.3.2），以保留 MAC 服务的其他方面（6.3.3 和 6.3.4）。可能会拒绝向未从重新配置中受益的终端站提供服务；因此，这些终端站的服务可用性会降低。桥接器可能会过滤帧，以便将桥接 LAN 中的流量本地化。如果终端站移动，则它可能无法从其他终端站接收帧，直到桥接器保存的过滤信息更新为止。

为了最大限度地提高服务可用性，网桥不应导致服务丢失或服务提供延迟，除非是由于桥接 LAN 组件发生故障、移除或插入，或由于终端站移动。这些被视为异常事件。因此，维持 MAC 服务质量所需的任何附加协议的运行仅限于桥接 LAN 的配置，并且与服务提供的单个实例无关。

注：这仅在不存在准入控制机制的情况下才适用，即网桥提供“尽力而为”服务。网桥中准入控制机制的规范和适用性不在本标准的范围内。

6.3.2 帧丢失

MAC 服务不保证服务数据单元的交付。源站传输的帧完好无损地到达目标站的概率很高。网桥的操作导致的额外帧丢失最少。

源站发送的帧可能无法到达目的站，原因是

a) 物理层传输或接收期间的帧损坏。
b) 桥接器丢弃帧，因为
    1) 它无法在某个最大时间段内传输该帧，因此必须丢弃该帧，以防止超过最大帧寿命（6.3.6）。

    2) 由于帧到达的速率不断超过其传输速率，因此内部缓冲容量耗尽，无法继续存储帧。

    3) 该帧携带的服务数据单元的大小超出了该帧将中继到的 LAN 上所采用的 MAC 程序所支持的最大值。
    4) 桥接 LAN 的连接拓扑结构的变化需要在有限的时间内丢弃帧以维持服务质量的其他方面。

6.3.3 帧乱序

对于给定的目标地址和源地址组合，MAC 服务不允许对具有给定用户优先级的帧进行重新排序。与 MA_UNITDATA.request 原语相对应的 MA_UNITDATA.indication 服务原语（具有相同的请求优先级和相同的目标地址和源地址组合）的接收顺序与请求原语的处理顺序相同。

注——网桥中的转发过程（7.7）的操作使得 MAC 服务的帧排序特性得以保留。

当桥接 LAN 中的桥接器能够以某种方式连接各个 MAC，使得任何源站 - 目标站对之间存在多条路径时，需要协议的操作来确保使用单一路径。

6.3.4 帧复制

MAC 服务不允许帧重复。网桥的运行不会引入用户数据帧的重复。

桥接 LAN 中帧重复的可能性源于以下原因：在桥接器内后续传输时，接收到的帧可能重复，或者源端站和目标端站之间可能存在多条路径。

桥接器不得复制用户数据帧。如果桥接 LAN 中的桥接器能够以任何源站-目标站对之间存在多条路径的方式连接各个 MAC，则需要协议的运行来确保使用单条路径。

6.3.5 运输延误

MAC 服务引入了帧传输延迟，该延迟取决于所采用的特定媒体和 MAC 方法。帧传输延迟是 MA_UNITDATA.request 原语和相应的 MA_UNITDATA.indication 原语之间的经过时间。经过时间值仅在成功传输的服务数据单元上计算。

由于 MAC 服务是在终端站内的抽象接口上提供的，因此无法精确指定总帧传输延迟。但是，可以测量与媒体访问以及传输和接收相关的延迟分量；并且可以测量由中间系统（在本例中为桥接器）引入的传输延迟。

桥接器引入的最小额外传输延迟是接收帧所花费的时间加上访问要中继该帧的介质所花费的时间。请注意，在中继帧之前，必须完全接收该帧，因为需要计算帧校验序列 (FCS)，如果帧有错误，则丢弃该帧。

6.3.6 帧寿命

MAC 服务确保特定通信实例的传输延迟存在上限。此最大帧寿命对于确保更高层协议的正确运行是必要的。桥接引入的额外传输延迟在 6.3.5 中讨论。

为了强制实施最大帧寿命，桥接器可能需要丢弃帧。由于 MAC 子层向桥接器提供的信息不包括任何特定帧已经经历的传输延迟，因此桥接器必须丢弃帧以强制实施每个桥接器的最大延迟。

最大桥接传输延迟值取决于桥接 LAN 中所有桥接器施加的最大延迟以及期望的最大帧寿命。表 8-2 中指定了建议值和绝对最大值。

6.3.7 未检测帧错误率

MAC 服务在传输帧中引入了非常低的未检测帧错误率。使用 FCS 可防止未检测错误，该 FCS 由源站的 MAC 子层在传输前附加到帧中，并由目标站在接收时进行检查。

为给定服务数据单元计算的 FCS 取决于所采用的 MAC 方法。因此，如果计算方法和/或 FCS 覆盖范围的差异，或 FCS 覆盖范围内的数据发生变化，导致两种 MAC 方法为服务数据单元计算出不同的 FCS，则有必要在提供不同类型 IEEE 802 MAC 之间中继功能的网桥内重新计算 FCS。这可能会导致网桥操作产生额外的未检测错误。对于在相同 MAC 类型的 LAN 之间中继的帧，网桥不得引入大于通过保留 FCS 实现的未检测帧错误率。

注：应用附件 G 中描述的技术，即使在 FCS 覆盖范围内的数据发生变化的情况下，实现未检测帧错误率的任意小的增加。作为本标准的维护活动，将启动对此要求措辞的修订，以期对符合要求的实现中可接受的未检测帧错误率的增加设置定量限制。

6.3.8 最大服务数据单元大小

IEEE 802 LAN 可支持的最大服务数据单元大小随 MAC 方法及其相关参数（速度、电气特性等）而变化。它可能受 LAN 所有者的限制。两个 LAN 之间的桥接器支持的最大服务数据单元大小是 LAN 支持的最大服务数据单元大小中的较小者。桥接器不会尝试将帧中继到不支持该帧所传达的服务数据单元大小的 LAN。

6.3.9 优先级

MAC 服务将用户优先级作为服务质量参数。具有高优先级的 MA_UNITDATA.request 原语可能优先于在同一站或连接到同一 LAN 的其他站发出的其他请求原语，并可能产生更早的 MA_UNITDATA.indication 原语。

MAC 子层将请求的用户优先级映射到单个 MAC 方法支持的访问优先级上。请求的用户优先级可以传送到目标站。

可以通过将用户优先级与帧关联来管理桥接器中帧所经历的传输延迟。

传输延迟包括

a) 按照 7.7.4 中描述的选择传输帧的程序，排队延迟直到该帧成为端口上第一个传输的帧；

b) 帧传输的访问延迟。

排队延迟可以通过 user_priority 来管理。在支持多个访问优先级的 MAC 方法中，访问延迟可以通过 user_priority 来管理。

与帧相关联的用户优先级可以通过某些 IEEE 802 LAN MAC 类型固有的优先级信号机制来发送信号。由于并非所有 IEEE 802 LAN MAC 类型都能够发送与帧相关联的用户优先级，因此 MAC 桥接器会根据信号信息和桥接器中保存的配置信息的组合来重新生成用户优先级。

桥接器将用户优先级映射到一个或多个流量类别；支持多个流量类别的桥接器能够支持加速流量类别。转发过程 7.7 描述了 MAC 桥接器中用户优先级和流量类别的使用。鉴于桥接器中对帧乱序的限制（如 6.3.3 中所述），优先级和流量类别的映射是静态的。

注 1—本标准中使用的术语“流量类别”仅在转发过程的优先级处理和排队功能的操作上下文中使用，如 7.7 中所述。其他上下文中赋予该术语的任何其他含义均不适用于本标准中该术语的使用。

在 IEEE 802 LAN 中发送 user_priority 信号的能力允许 user_priority 在桥接 LAN 中以端到端的方式传输。这与将 user_priority 映射到流量类别以及将 user_priority 映射到 access_priority 的一致方法相结合，允许根据传输路径中涉及的 Bridges 和 MAC 方法的功能一致地使用 user_priority 信息。

注 2—IEEE P802.1Q[8]定义帧格式和相关程序，可用于在无法发送用户优先级信号的 LAN MAC 类型之间传输用户优先级信息。使用 P802.1Q 帧格式允许保持 user_priority 的端到端重要性，而不管各个 LAN MAC 类型是否能够发送优先级信号。

在正常情况下，user_priority 在通过网桥的中继功能传输时不会被修改；但是，在某些情况下，出于管理目的，可能需要控制 user_priority 的传播方式。用户优先级再生表（表 7-1）提供了在管理控制下按端口映射传入 user_priority 值的能力。在默认状态下，此表

[8]IEEE P802.1Q/D11（1998 年 7 月 30 日）是经过授权的 IEEE 标准项目，但在本标准付印时尚未获得 IEEE-SA 的批准。有关获取草案的信息，请联系 IEEE。

提供从 user_priority 值到重新生成的 user_priority 值的标识映射；即，默认情况下，重新生成的 user_priority 与传入的 user_priority 相同。

6.3.10 吞吐量

桥接 LAN 提供的总吞吐量可能比等效单个 LAN 提供的总吞吐量大得多。网桥可以通过过滤帧来本地化桥接 LAN 内的流量。桥接 LAN 中可用的过滤服务在 6.6 中描述。

通过网桥进行通信的各个 LAN 上的终端站之间的吞吐量可能会因网桥中的帧丢弃而降低，因为在较长时间内无法在形成到目的地的路径的 LAN 上以所需的速率进行传输。

6.4 MAC 桥接器内提供的内部子层服务

MAC 实体向桥接器内的 MAC 中继实体提供的内部子层服务是由 LAN 端口的单个 MAC 提供的。这遵守其所连接的 LAN 的适当 MAC 程序和协议。除其来源 LAN 外，不会在任何 LAN 上转发控制帧（即不传送 MAC 用户数据的帧）。

内部子层服务源自 ISO/IEC 15802-1 定义的 MAC 服务，通过在该规范中添加执行中继功能所需的元素来扩展该规范。在连接的终端站内，这些附加元素可以被视为低于 MAC 服务边界，并且仅与服务提供商的操作有关；或不构成 MAC 服务对等性质的本地事务。在与 ISO/IEC 15802-1 定义的 MA_UNITDATA.request 和 MA_UNITDATA.indication 原语相关的参数列表中添加了三个参数。它们是 frame_type、MAC_action 和 frame_check_sequence。内部子层服务的定义不会向 LAN MAC 服务定义定义的服务原语添加任何新服务原语。

内部子层服务不包括 MAC 特定的功能和程序，这些功能和程序的操作仅限于各个 LAN。描述此服务的单元数据原语是

M_UNITDATA.指示 （
帧类型，
mac_action，
目标地址，
源地址，
mac_服务_数据_单元，
用户优先级，
帧校验序列
）

每个 M_UNITDATA 指示原语对应于从单个 LAN 接收无错误的 MAC 帧。

注——接收帧的错误情况的详细规范包含在相关的 MAC 标准中；例如，FCS 错误、长度错误、非整数八位字节数。

这**框架类型**参数表示帧的类别。该参数的值为 user_data_frame、mac_specific_frame 或 reserved_frame 之一。

这**mac_action**参数表示接收指示的 MAC 实体请求的操作。如果 frame_type 参数的值为 user_data_frame，则 mac_action 参数为以下之一：

request_with_response、request_with_no_response 或 response。对于 mac_specific_frames 和 reserved_frames，此参数不适用。

这**目标地址**参数是单个 MAC 实体的地址，也可以是一组 MAC 实体的地址。

这**源地址**参数是源MAC实体的单独地址。

这**mac_service_data_unit**参数是服务用户数据。

这**用户优先级**参数是发起服务用户请求的优先级。此参数的值范围为0至7。

注 1 — 默认 user_priority 值为 0。值 1 到 7 构成 user_priority 的有序序列，其中 1 为最低值，7 为最高值。有关 user_priority 值的使用及其如何映射到流量类别的进一步说明，请参阅 7.7.3 和 H.2。

网桥能够根据数据指示中包含的用户优先级信息和接收端口的用户优先级再生表重新生成用户优先级，如 7.5.1 中所述。user_priority 参数采用再生 user_priority 的值。对于无法发送优先级信号的 IEEE 802 LAN MAC 类型，user_priority 参数采用接收指示的端口的默认用户优先级的值。网桥所有端口的默认用户优先级的默认值均为 0；如果支持第 14 条中描述的管理功能，则可以通过此类管理功能修改给定端口的值。

这**帧校验序列**参数明确作为原语的参数提供，以便它可以用作相关请求原语的参数，而无需重新计算。

接收特定帧所来自的 LAN 的标识是本地事务，并不表示为服务原语的参数。

M_UNITDATA.请求 （
帧类型，
mac_动作，
目标地址，
源地址，
mac_服务_数据_单元，
用户优先级，
访问优先级，
帧校验序列
）

调用数据请求原语将帧传输到单个 LAN。

这**框架类型**参数表示帧的类别。

这**mac_action**参数表示请求目标MAC实体的动作。

这**目标地址**参数是单个 MAC 实体的地址，或者是一组 MAC 实体的地址。

这**源地址**参数是源MAC实体的单独地址。

这**mac_service_data_unit**参数是服务用户数据。

这**用户优先级**参数是发起服务用户请求的优先级。此参数的值范围为0至7。

注 2 — 默认 user_priority 值为 0。值 1 到 7 构成 user_priority 的有序序列，其中 1 为最低值，7 为最高值。有关 user_priority 值的使用及其如何映射到流量类别的进一步说明，请参阅 7.7.3 和 H.2。

这**访问优先级**参数是本地服务提供商用来传达请求的优先级。它可用于确定本地 MAC 实体排队的帧传输的优先级，无论是本地还是连接到同一 LAN 的其他站（如果 MAC 方法允许）。如果指定此参数，则其值在 0（最低）到 7（最高）范围内。

这**帧校验序列**参数明确作为原语的参数提供，以便无需重新计算即可使用。

帧要传输到的 LAN 的标识是本地事务，并不表示为服务原语的参数。

6.5 特定MAC程序对内部子层服务的支持

本小节规定了内部子层服务与每个单独的 IEEE 802 MAC 类型的 MAC 协议和程序之间的映射，以及 MAC 帧中服务参数的编码。映射是通过参考规定了各个 MAC 方法的 IEEE 802 标准来规定的。映射提请注意连接到该 MAC 类型的 LAN 的网桥的任何特殊职责。

6.5.1 IEEE 标准 802.3 (CSMA/CD) 支持

CSMA/CD 访问方法在 IEEE Std 802.3 中指定。该标准的第 3 条款指定了 MAC 帧结构，第 4 条款指定了 MAC 方法。

收到 M_UNITDATA.request 原语后，本地 MAC 实体执行传输数据封装，使用下面指定的参数组装帧。它会在帧交给 MAC 子层中的传输媒体访问管理组件进行传输之前添加前导码和起始帧分隔符（IEEE Std 802.3，4.2.3）。

在接收媒体访问管理接收到 MAC 帧后，该 MAC 帧将被传递给接收数据解封装，后者将验证 FCS 并按照下文所述将帧拆开为 M_UNITDATA.indication 原语提供的参数（IEEE Std 802.3，4.2.4）。

这**框架类型**参数仅采用值 user_data_frame 并且未在 MAC 帧中明确编码。

这**mac_action**参数仅采用值 request_with_no_response 并且未在 MAC 帧中明确编码。

这**目标地址**参数被编码在 MAC 帧的目标地址字段中（IEEE Std 802.3，3.2.3）。

source_address 参数编码在 MAC 帧的源地址字段中（IEEE Std 802.3，3.2.3）。

mac_service_data_unit 参数中的八位字节数编码在 MAC 帧的长度字段中（IEEE Std 802.3，3.2.6），数据的八位字节编码在数据字段中（IEEE Std 802.3，3.2.7）。

这**用户优先级**数据请求原语中提供的参数未编码在 MAC 帧中。数据指示原语中提供的 user_priority 参数采用接收 MAC 帧的端口的默认用户优先级参数的值（见 6.4）。

这**帧校验序列**参数编码在 MAC 帧的 FCS 字段中（IEEE Std 802.3，3.2.8）。FCS 是根据目标地址、源地址、长度、数据和 PAD 字段计算的。如果 M_UNITDATA.request 原语不附带此参数，则根据 IEEE Std 802.3，3.2.8 计算。

注 1 — 因为 PAD 字段（如果存在）对 FCS 有贡献，所以此参数需要至少包括 PAD 字段对 FCS 的贡献，以便保留原始 FCS（参见附件 G）。

除了为支持 LLC 使用 MAC 服务而指定的操作之外，不需要采取任何特殊操作来支持 CSMA/CD 访问方法的 MAC 内部子层服务。

注 2 — IEEE Std 802.3 的支持仅在桥接器操作方面进行描述，这些操作是在通过 802.3 MAC 使用 LLC 服务中继帧时进行的。ISO/IEC 11802-5 定义了桥接以太网 V2.0 帧的推荐做法。

注 3 — IEEE Std 802.3（1998 年版）描述了其帧格式中长度或以太网协议类型的使用；但本节的文字尚未修改以描述以太网协议类型的使用。

6.5.2 ISO/IEC 8802-4 支持（令牌传递总线）

令牌传递总线访问方法在 ISO/IEC 8802-4 中指定。在该标准中，第 4 节指定了帧格式，第 5 节讨论了访问协议的基本概念，第 7 节提供了令牌传递总线 MAC 操作的明确规范。

收到 M_UNITDATA.request 原语后，本地 MAC 实体接口机 (IFM) 将帧排队，以便由访问控制机 (ACM) 传输（ISO/IEC 8802-4，第 7 节）。在传输时，使用下面指定的参数设置帧字段，并添加前导码、起始分隔符和结束分隔符（ISO/IEC 8802-4，第 4 节）。

当接收机 (RxM) (ISO/IEC 8802-4, 7.1.5) 收到有效的 MAC 帧时，会生成 M_UNITDATA.indication 原语，其参数源自下文指定的帧字段。

这**框架类型**参数在帧控制 (FC) 字段的 FF 位（位位置 1 和 2）中编码（ISO/IEC 8802-4，4.1.3.1，4.1.3.2）。位模式 0 1 表示 user_data_frame，位模式 0 0 或 1 0 表示 mac_specific_frame，位模式 1 1 表示 reserved_frame。

这**mac_action**参数在 FC 字段的 MMM 位（位位置 3、4 和 5）中编码（ISO/IEC 8802-4，4.1.3.2）。对于 user_data_frames，这些值对于 request_with_no_response 取 0 0 0 的值，对于 request_with_response 取 0 0 1 的值，对于 response 取 0 1 0 的值。

这**目标地址**参数被编码在 MAC 帧的目标地址字段中（ISO/IEC 8802-4, 4.1.4.1）。

这**源地址**参数被编码在MAC帧的源地址字段中（ISO/IEC 8802-4，4.1.4.2）。

这**mac_service_data_unit**参数在 MAC 数据单元字段中编码（ISO/IEC 8802-4, 4.1.5）。

这**用户优先级**参数在 FC 字段的 PPP 位（位位置 6、7 和 8）中编码（ISO/IEC 8802-4、4.1.3.2；5.1.7）。

这**帧校验序列**参数编码在 MAC 帧的 FCS 字段中（ISO/IEC 8802-4，4.1.6）。FCS 是根据帧控制、目标地址、源地址和数据字段计算的。如果 M_UNITDATA.request 原语不附带此参数，则根据 ISO/IEC 8802-4，4.1.6 计算。

除了为支持 LLC 使用 MAC 服务而指定的操作之外，不需要采取任何特殊操作来支持令牌传递总线访问方法的 MAC 内部子层服务。

6.5.3 ISO/IEC 8802-5 支持（令牌传递环）

令牌传递环访问方法在 ISO/IEC 8802-5 中指定。该标准的第 3 条款指定了格式和设施，第 4 条款指定了令牌传递环协议。

收到 M_UNITDATA.request 原语后，本地 MAC 实体使用下面指定的参数组成一个帧，将帧控制、目标地址、源地址和 FCS 字段附加到用户数据，并将帧排队以进行传输。在传输时，将添加起始分隔符、访问控制字段、结束分隔符和帧状态字段。

在接收到不是由桥接端口的本地 MAC 实体传输的有效 MAC 帧（ISO/IEC 8802-5，4.1.4）时，将路由信息指示位（在源地址字段中占据与目标地址字段中的组地址位相同的位置）设置为零，并生成 M_UNITDATA.indication 原语，其参数来自帧字段，如下所示。

这**框架类型**参数在帧控制字段的 frame_type 位（FF 位）中编码（ISO/IEC 8802-5，3.2.3.1）。位模式 0 1 表示 user_data_frame，位模式 0 0 表示 mac_specific_frame，位模式 1 0 或 1 1 表示 reserved_frame。

这**mac_action**参数仅采用值 request_with_no_response 并且未在 MAC 帧中明确编码。

这**目标地址**参数被编码在 MAC 帧的目标地址字段中（ISO/IEC 8802-5，3.2.4.1）。

这**源地址**参数被编码在 MAC 帧的源地址字段中（ISO/IEC 8802-5，3.2.4.2）。

这**mac_service_data_unit**参数在信息字段中编码（ISO/IEC 8802-5,3.2.6）。

这**用户优先级**与 user_data_frames 相关的参数在帧控制字段的 YYY 位中编码（ISO/IEC 8802-5, 3.2.3）。

这**帧校验序列**参数编码在 MAC 帧的 FCS 字段中（ISO/IEC 8802-5，3.2.7）。FCS 是根据帧控制、目标地址、源地址和信息字段计算的。如果 M_UNITDATA.request 原语不附带此参数，则根据 ISO/IEC 8802-5，3.2.7 计算。

如果生成了 frame_type 和 mac_action 参数值分别为 user_data_frame 和 request_with_no_response 的 M_ UNITDATA.indication 原语，或者如果缓冲可用时生成了这样的指示，则可以将帧 ISO/IEC 8802-5，3.2.9) 的帧状态字段中的地址识别 (A) 位设置为 1；否则，除非 ISO/IEC 8802-5 有要求，否则不得设置 A 位。

如果 A 位设置为 1，则帧复制（C）位（ISO/IEC 8802-5, 3.2.9）可以设置为 1，以反映接收缓冲的可用性；否则不应设置 C 位。

为了支持 MAC 内部子层服务，令牌环桥必须能够识别和删除其自身传输的帧，即使它们携带的源地址与传输它们的桥端口的源地址不同。

6.5.4 光纤分布式数据接口（FDDI）支持

FDDI 访问方法在 ISO 9314-2 中指定。该标准的第 6 条指定了服务，第 7 条和第 8 条分别指定了设施和操作。

收到有效帧（ISO 9314-2，8.3.1）时，该帧不是由桥接端口的本地MAC实体传输的，并且源地址的第一位等于零，则生成M_UNITDATA.indication原语。与原语相关的参数来自下文中指定的帧字段。

除非 ISO 9314-2 另有要求，否则，不得设置接收帧的环上帧状态字段中的地址识别 (A) 指示符（ISO 9314-2，7.3.8）。如果生成了 M_UNITDATA.indication 原语（其 frame_type 和 mac_action 参数值分别为 user_data_frame 和 request_with_no_response），如果要转发帧，并且接收缓冲可用，则可以设置帧复制 (C) 指示符（ISO 9314-2，7.3.8）。否则，除非 ISO 9314-2 另有要求，否则，不得更改 C 指示符。

注：ISO 9314-2 MAC 桥接器对 C 指示器设置的规范是对 ISO 9314-2 中给出的规范的增强。ISO 9314-2 可以要求桥接器在接收到以 FDDI 终端站为地址的帧时，或接收到与 FDDI MAC 操作相关的帧时，改变 A 和/或 C 指示器

与接收到帧时生成的 M_UNITDATA.indication 相关的参数现在如下：

这**框架类型**参数在帧控制字段 (ISO 9314-2，7.3.3) 的帧格式位 (CL、FF 和 ZZZZ 位) 中编码。位模式 0L01rXXX 表示 user_data_frame (异步 LLC 帧，其中 L 表示地址长度，可以是 0 或 1，r 为保留，可以接收为 0 或 1，XXX 的范围为 000 至 111)。所有其他位模式产生的 frame_type 参数值为 not_user_data_frame。

这**mac_action**参数仅采用值 request_with_no_response 并且未在 MAC 帧中明确编码。

这**目标地址**参数包含在 MAC 帧的目标地址字段中（ISO 9314-2，7.3.4-7.3.4.1）。

这**源地址**参数被编码在MAC帧的源地址字段中（ISO 9314-2，7.3.4-7.3.4.2）。

这**mac_service_data_unit**参数在信息字段中编码（ISO 9314-2,7.3.5）。

这**用户优先级**当帧为异步传输的 LLC 帧（其帧控制字段值为 0L010PPP）时，与 user_data_frames 关联的参数被编码在帧控制字段的 PPP 位中（ISO 9314-2，7.3.3.4），其中 L 表示地址长度（ISO 9314-2，7.3.3.2）。

这**帧校验序列**参数被编码在 MAC 帧的帧校验序列字段中（ISO 9314-2，7.3.6）。frame_check_sequence 是根据帧控制、目标地址、源地址和信息字段计算得出的。

收到 M_UNITDATA.request 原语后，本地 MAC 实体使用上述提供的参数组成一个帧，将帧控制、目标地址、源地址和帧校验序列附加到用户数据，并在收到合适的令牌后将帧排入队列进行传输（ISO 9314-2，8.3.1）。在传输时，将添加前导码、起始分隔符、结束分隔符和帧状态字段。

如果 M_UNITDATA.request 原语没有附带帧校验序列，则根据 ISO 9314-2, 7.3.6 进行计算。

帧控制字段的位模式应为 0L01rPPP，表示异步 LLC 帧，其中 L 表示地址长度（ISO 9314-2，7.3.3.2），r 保留并设置为零，PPP 表示帧的优先级（ISO 9314-2，7.3.3.4）。

如果**用户优先级**参数值指定后，将以 PPP 位进行编码，访问优先级（ISO 9314-2，8.1.4）由令牌持有定时器（THT）或实现者选项得出。如果未指定 user_priority 参数值，则应将 PPP 位设置为零，访问优先级由 THT 得出。

为了支持 MAC 内部层服务，FDDI 桥接器会按照 ISO 9314-2 的要求删除其自身传输的帧，即使这些帧可以携带与传输它们的桥接端口不同的源地址。

6.5.5 ISO/IEC 8802-6 支持（分布式队列双总线）

ISO/IEC 8802-6 访问方法在 ISO/IEC 8802-6 中指定。该标准的 5.1 子条款指定了向 LLC 提供 MAC 服务，第 6 节指定了 DQDB 层协议数据单元 (PDU) 格式。

收到 M_UNITDATA.request 原语后，本地 MAC 实体（MAC 融合功能 [MCF]，ISO/IEC 8802-6，5.1.1）应使用提供的参数，通过编码和连接公共 PDU 报头（ISO/IEC 8802-6，6.5.1.1）、MAC 融合协议（MCP）报头（ISO/IEC 8802-6，6.5.1.2）、报头扩展（ISO/IEC 8802-6，6.5.1.3）、INFO 字段（ISO/IEC 8802-6，6.5.1.4）、PAD 字段（ISO/IEC 8802-6，6.5.1.5）、可选的 CRC32 字段（ISO/IEC 8802-6，6.5.1.6）和公共 PDU，组成初始 MAC 协议数据单元（IMPDU）。尾部（ISO/IEC 8802-6，6.5.1.7）。这些报头和尾部内设置的值均按规定执行，并符合 ISO/IEC 8802-6，5.1.1.1.1（IMPDU 的创建）。然后应将 IMPDU 提交给 IMPDU 分段机进行传输（ISO/IEC 8802-6，5.1.1.1.2）。

这**框架类型**参数应采用值 user_data_frame，并且未在 IMPDU 中明确编码。

这**mac_action**参数应采用值 request_with_no_response 并且未在 IMPDU 中明确编码。

通用 PDU 头应根据 ISO/IEC 8802-6, 6.5.1.1 进行编码。

这**目标地址**参数应在 MCP 头的目标地址 (DA) 子字段中进行右对齐编码，其中 Address_Type 子字段设置为 48_bit_address（二进制 1000），并用 12 个零填充以左填充 60 位地址字段（ISO/IEC 8802-6，6.5.1.2.1）。

这**源地址**参数应在 MCP 头的源地址 (SA) 子字段中进行右对齐编码，其中 Address_Type 子字段设置为 48_bit_address（二进制 1000），并用 12 个零填充以左填充 60 位地址字段（ISO/IEC 8802-6，6.5.1.2.1）。

MCP 头的协议标识符 (PI) 子字段应编码为十进制 1（以指示 LLC）（ISO/IEC 8802-6，6.5.1.2.4.1）。

MCP 头的 PAD 长度（PL）子字段应根据 ISO/IEC 8802-6, 6.5.1.2.4.2 进行编码。

MCP 头的桥接字段应设置为全零（ISO/IEC 8802-6，6.5.1.2.6）。此字段仅供 60 位 ISO/IEC 8802-6 PDU 桥接使用。

如果在相应的数据指示中存在报头扩展字段，则应使用收到的值对报头扩展字段进行编码。否则，应根据 ISO/IEC 8802-6, 5.1.1.1.1，步骤 11) 和 13) 对报头扩展字段进行编码。

注：ISO/IEC 8802-6 中的 Header Extension 字段是 MAC 专用功能，与 MAC Bridge 的操作无关；因此，其值未在内部子层服务中表示。但是，如果 Header Extension 存在于收到的 MAC 帧中，则其值将保留在相应的传输帧中。

这**mac_service_data_unit**参数应被编码到 IMPDU 的 INFO 字段中（ISO/IEC 8802-6, 6.5.1.4）。

这**用户优先级**参数应编码在 MCP 头的 QOS_DELAY 子字段中（ISO/IEC 8802-6，6.5.1.2.5.1）。MCP 头的 QOS_LOSS 子字段应编码为零（ISO/IEC 8802-6，6.5.1.2.5.2）。如果指定了 frame_check_sequence 参数的值（ISO/IEC 8802-6，6.5.1.2.5.3），或者未指定 frame_check_sequence 参数但计算了 FCS 并将其包含在 CRC32 字段中，则 MCP 头的 CRC32 指示位 (CIB) 子字段应设置为 1。header_extension 参数的长度应编码在 MCP 头的头扩展长度 (HEL) 子字段中（ISO/IEC 8802-6，6.5.1.2.5.4）。

PAD 字段应根据 ISO/IEC 8802-6, 6.5.1.5 进行编码。

这**帧校验序列**如果指定了 frame_check_sequence 参数，则应将其编码在 CRC32 字段中（ISO/IEC 8802-6，6.5.1.6）。如果未指定 frame_check_sequence 参数，则实现者可以选择

a) 计算并包括 CRC32 字段并将 CIB 设置为 1，或者
b) 不包括 CRC32 字段并将 CIB 设置为零。

通用 PDU 尾部应根据 ISO/IEC 8802-6, 6.5.1.1 进行编码。

当 MCF 接收块收到有效的 IMPDU（ISO/IEC 8802-6，5.1.1.2）时，如果通过检查目标和源地址字段中的 Address_Type 子字段（ISO/IEC 8802-6，6.5.1.2.1）验证其包含 48 位地址，则应生成 M_UNITDATA.indication 原语，其参数从 IMPDU 字段派生而来，如下所示。如果 Address_Type 未指示 48 位地址，则应丢弃 IMPDU，并且不会生成 M_UNITDATA.indication 原语。

这**框架类型**参数应采用值 user_data_frame，因为它没有在 IMPDU 中明确编码。

这**mac_action**参数应采用值请求with_no_response，因为它没有在IMPDU中明确编码。

这**目标地址**参数应赋予 MCP 头的目标地址（DA）子字段的 MSAP 地址的值（ISO/IEC 8802-6, 6.5.1.2.1）。

这**源地址**参数应赋予 MCP 头的源地址（SA）子字段的 MSAP 地址的值（ISO/IEC 8802-6, 6.5.1.2.1）。

这**mac_service_data_unit**参数应赋予 IMPDU 的 INFO 字段的值（ISO/IEC 8802-6, 6.5.1.4）。

这**用户优先级**参数应赋予 MCP 头的 QOS_DELAY 子字段的值（ISO/IEC 8802-6，6.5.1.2.5.1）。

这**帧校验序列**如果 CRC32 指示位 (CIB)（ISO/IEC 8802-6, 6.5.1.2.5.3）设置为 1，则参数应赋予 CRC 32 字段（ISO/IEC 8802-6, 6.5.1.6）的值；否则应不指定。

除了为支持 LLC 使用 MAC 服务而指定的操作之外，DQDB 访问方法对 MAC 内部子层服务的支持不需要任何特殊操作。

6.5.6 IEEE 标准 802.11 (无线局域网) 的支持

无线局域网接入方法在 IEEE Std 802.11 中指定。该标准的第 7 条款指定了帧格式，第 9 条款指定了 MAC 子层功能，第 11 条款指定了强制的 MAC 子层管理功能。

8802-11 LAN 的桥接器应连接到 8802-11 门户，后者又连接到 8802-11 分布系统。为了实现桥接，门户上显示的服务接口与 8802-11 MAC SAP 上显示的服务接口相同。8802-11 分布系统的实例可从 802 LAN 组件实现。8802-11 STA 通过 8802-11 接入点连接到分布系统。桥接器不应连接到 8802-11 独立 BSS。有关 8802-11 架构的描述，请参阅 IEEE Std 802.11 的第 5 条。

在接收到 M_UNITDATA.request 原语后，门户将构建一个 MAC 服务数据单元并将其传递给 MAC 数据服务进行传输，遵循 IEEE Std 802.11 第 6、7、9 条和附件 C 中指定的帧格式和程序，使用如下所示提供的参数。

在收到有效的 MAC 服务数据单元（参见 IEEE 标准 802.11 第 6、7、9 条和附件 C）后，门户会生成一个 M_UNITDATA.indication 原语，其参数值源自如下指定的帧字段。

frame_type参数只接受user_data_frame这个值，在处理MSDU_from_LLC时，user_data_frame的frame_type需要按照IEEE Std 802.11第7.1.3.1节规定的参数进行转换，并在MAC帧中明确编码。

mac_action 参数仅采用值 request_with_no_response 并且未明确编码。

destination_address 参数在 MAC 帧中编码为 IEEE Std 802.11 的 7.2.2 表 4 中所述的 DA。

source_address 参数在 MAC 帧中编码为 IEEE Std 802.11 的 7.2.2 表 4 中描述的 SA。

mac_service_data_unit 参数编码在 MAC 帧的 Frame Body 字段（IEEE Std 802.11，7.1.3.5）中。MSDU 的长度应为≤2304 个八位字节。长度未编码在 MAC 帧中，而是在 PHY 标头中传送。

user_priority 参数未编码在 MAC 帧中。M_UNITDATA.indication 原语中提供的 user_priority 参数应采用接收 MAC 服务数据单元的端口的 Default_User_Priority 参数的值（见 6.4）。

frame_check_sequence 参数根据 IEEE Std 802.11, 7.1.3.6 FCS 在 MAC 帧的帧校验序列 (FCS) 字段中编码。

access_priority 参数未在 MAC 帧中编码。

除了 IEEE Std 802.11 中指定的操作外，无线 LAN 访问方法对 MAC 内部子层服务的支持不需要任何特殊操作。

6.5.7 ISO/IEC 8802-12 的支持（需求优先级）

需求优先级接入方法在 ISO/IEC 8802-12 中规定。该标准的第 10 条规定了帧格式，第 11 条规定了 MAC 协议。规定了两种 MAC 帧格式，一种与 IEEE Std 802.3 帧格式兼容，一种与 ISO/IEC 8802-5 帧格式兼容（给定的需求优先级 LAN 始终只使用其中一种格式运行）。

收到 M_UNITDATA.request 原语后，本地 MAC 实体将构建并传输相应的 MAC 帧，如 ISO/IEC 8802-12、11.5.7（FUNCTION Build_Frame）和 11.6.6（MAC6_TRANSMIT_FRAME）中所述。

在接收到 MAC 帧（ISO/IEC 8802-12，11.6.5，MAC_READ_FRAME）时，本地 MAC 实体将生成 M_UNITDATA.indication 原语，如 ISO/IEC 8802-12，11.5.6（PROCEDURE Process_Received_MAC_Frame）中所述。

这**框架类型**参数仅采用值 user_data_frame 并且未在 MAC 帧中明确编码。

这**mac_action**参数仅采用值 request_with_no_response 并且未在 MAC 帧中明确编码。

这**目标地址**参数被编码在 MAC 帧的目标地址 (DA) 字段中 (ISO/IEC 8802-12、10.2.1 和 10.3.3)。

这**源地址**参数被编码在 MAC 帧的源地址（SA）字段中（ISO/IEC 8802-12、10.2.1 和 10.3.3）。

这**mac_service_data_unit**参数被编码在 MAC 帧的数据字段（IEEE Std 802.3 帧格式，ISO/IEC 8802-12，10.2.3）或信息字段（ISO/IEC 8802-5 帧格式，ISO/IEC 8802-12，10.3.5）中。

对于 IEEE Std 802.3 帧格式，**用户优先级**参数未在 MAC 帧中编码，但对应于 ISO/IEC 8802-12 优先级值“normal”或“high”。在帧接收时，值“normal”映射到 user_priority 0，值“high”映射到 user_priority 4。在帧传输时，user_priority 值 0 至 3 映射到“normal”，值 4 至 7 映射到“high”。

对于 ISO/IEC 8802-5 帧格式，**用户优先级**参数在帧控制字段的 YYY 位中编码（ISO/IEC 8802-12，10.3.2.2）。

这**访问优先级**M_UNITDATA.request 原语中的参数映射到 ISO/IEC 8802-12 优先级值“normal”或“high”：access_priority 值 0 到 3 映射到“normal”，access_priority 值 4 到 7 映射到“high”。

这**帧校验序列**参数在 MAC 帧的帧校验序列 (FCS) 字段中编码 (ISO/IEC 8802-12、10.2.4 和 10.3.6)。

除了 ISO/IEC 8802-12 中指定的操作外，需求优先访问方法对 MAC 内部子层服务的支持不需要任何特殊操作。

6.6 桥接 LAN 中的过滤服务

MAC 桥接器在桥接 LAN 中提供过滤服务，支持维护服务质量的某些方面；特别是传输延迟、优先级和吞吐量。此外，这些服务还对桥接 LAN 中特定 MAC 地址的传播提供一定程度的管理控制。

所描述的服务是最一般意义上的服务；即，它们是 MAC 服务用户或管理员可用的功能描述，用于控制和访问桥接 LAN 中的过滤功能。每项服务的描述均未假设服务如何实现。至少存在以下可能性：

a) 使用 IEEE 802 标准和其他地方定义的现有协议和机制；
b) 使用管理功能，无论是本地定义还是通过远程管理协议实现；

c) 其他方式，标准化或其他方式。

6.6.1 过滤服务提供的目的

桥接 LAN 中提供的过滤服务可用于以下小节中所述的目的。

6.6.1.1 行政控制

过滤服务提供对网络指定部分中特定源地址和目标地址使用的管理控制。此类控制允许网络管理员和管理员通过建立特定 MAC 地址不被转发的管理边界来限制使用个人和群组 MAC 地址的网络层和其他协议的运行范围。

6.6.1.2 吞吐量和终端站负荷

过滤服务可提高网络的整体吞吐量，并减少因接收发往其他终端站的帧而对终端站造成的负载。它们通过以下方式实现此目的

a) 将发往特定 MAC 地址的帧限制到网络的某些部分，这些部分很可能位于源 MAC 地址和目标 MAC 地址之间的路径上；
b) 将组寻址帧的范围缩小到包含该流量合法接收者的终端站的网络部分。

注：本标准中描述的过滤服务的某些方面依赖于终端站的积极参与。如果无法积极参与，则过滤服务的这些方面将不可用。

6.6.2 过滤服务提供的目标

桥接 LAN 中提供的过滤服务提供了一组功能，可用于

a) 允许 MAC 服务提供商动态了解发往各个 MAC 地址的帧的接收者位于何处；

b) 允许作为发往组 MAC 地址的 MAC 帧的潜在接收者的终端站向 MAC 服务提供商动态指示他们希望接收哪个目标 MAC 地址；

c) 对特定MAC地址的传播范围实施管理控制。

6.6.3 过滤服务的用户

桥接 LAN 中提供的过滤服务可供以下用户使用：

a) 网络管理和管理，用于实施管理控制。网络管理员和过滤服务提供商之间的交互可以通过本地方式或通过明确的管理机制实现；

b) 终端站，用于控制它们将接收的目标地址。终端站和过滤服务提供者之间的交互可能是隐式的，如学习过程 (7.8) 提供的过滤服务，或通过显式使用过滤服务原语。

6.6.4 服务基础

桥接 LAN 中的所有过滤服务都依赖于过滤规则的建立以及后续的过滤决策，这些决策基于在桥接 LAN 中传播的 MAC 帧中的源或目标 MAC 地址字段中包含的值。

注——本标准定义的过滤服务基于源地址学习和目标地址过滤。

6.6.5 服务类别

桥接 LAN 中的过滤服务分为以下几类：

一个)*基本过滤服务*. 这些服务由转发进程 (7.7) 和过滤数据库中的静态过滤条目 (7.9.1) 和动态过滤条目 (7.9.2) 支持。动态过滤条目中包含的信息通过学习进程 (7.8) 的运行进行维护。

b) *扩展过滤服务*。这些服务由转发进程 (7.7) 以及过滤数据库中的静态过滤条目 (7.9.1) 和组注册表项 (7.9.3) 支持。组注册表项中包含的信息通过 GMRP (10) 的操作进行维护。扩展过滤服务的类别如下：

1）支持动态组转发和过滤行为；
2) 各个 MAC 地址的静态过滤信息能够指定端口子集，并根据动态过滤信息做出转发或过滤决策。

注——本标准中定义的基本过滤服务与先前发布的 MAC 桥接标准 ISO/IEC 10038: 1993 [IEEE Std 802.1D，1993 版] 提供的过滤功能完全对应。

所有网桥都应支持基本过滤服务。网桥是否支持任一类别的扩展过滤服务是可选的。

6.6.6 服务配置

当过滤数据库中缺少明确信息时，转发过程对于转发或过滤发往组 MAC 地址的帧的行为取决于网桥支持的服务类别。

基本过滤服务支持桥接 LAN 区域所需的过滤行为，这些区域存在多播帧的潜在接收者，但接收者或桥接器无法支持这些组 MAC 地址的过滤信息的动态配置，或者接收者需要接收发往组 MAC 地址的所有流量。

扩展过滤服务支持桥接 LAN 区域所需的过滤行为，这些区域存在多播帧的潜在接收者，并且帧的潜在接收者和桥接器均能够支持组 MAC 地址的过滤信息的动态配置。为了将此扩展过滤行为与仅支持基本过滤服务的网络区域的需求相结合，可以静态和动态地配置支持扩展过滤服务的桥接器，以根据每个组 MAC 地址修改其过滤行为，并根据每个出站端口针对多播帧提供的整体过滤服务修改其过滤行为。后一种功能允许配置端口针对未配置特定静态或动态过滤信息的组 MAC 地址的默认转发或过滤行为。

服务配置提供了针对以下情况配置整体过滤的能力：

a) 仅实现基本过滤服务的桥梁；
b) 在异构环境中支持扩展过滤服务的网桥，在这种环境中，某些设备无法参与动态多播过滤，或者某些设备（例如路由器）有查看未过滤流量的特定需求；以及

c) 在同质环境中支持扩展过滤服务的网桥，其中所有设备都能够参与动态多播过滤。

6.6.7 扩展过滤服务的服务定义

过滤服务通过服务原语来描述，这些原语定义了 MAC 服务用户和 MAC 服务提供者之间跨 MAC 服务边界的特定交互类型。由于这些交互不是在对等实体之间定义的，因此它们只是根据 MA C 服务用户向 MAC 服务提供者发送的服务请求来描述。

6.6.7.1 动态注册和注销服务

这些服务允许 MAC 服务用户动态控制从 MAC 服务提供商处收到的目标组 MAC 地址集，方法是

a) 注册/取消注册与这些地址相关的特定群组的成员资格；
b) 针对群组的整体转发/过滤行为注册/取消注册其服务要求。

这些服务的提供是通过 GMRP 及其相关程序实现的，如第 10 条所述。

注意：这些服务旨在为 MAC 服务用户提供对多播数据流访问的动态控制，例如，由服务器提供的多个视频频道，每个频道使用不同的组 MAC 地址。注册和取消注册组成员身份的能力，加上与组成员身份相关的过滤操作，限制了此类服务对桥接 LAN 中可用带宽的影响。出于类似的原因，这些服务可用于控制其他类别的多播流量的接收。

注册组成员 (MAC 地址)

向 MAC 服务提供商表明 MAC 服务用户希望接收包含 MAC_ADDRESS 参数中指示的组 MAC 地址作为目标地址的帧。此参数可以携带的 MAC 地址不包括

a) 任何个人地址；
b) 表 7-9 中标识的任何保留地址；
c) 任何 GARP 应用程序地址，如表 12-1 所定义。

注销组成员 (MAC 地址)

向 MAC 服务提供商指示终端站不再希望接收包含 MAC_ADDRESS 参数中指示的组 MAC 地址作为目标地址的帧。

注册服务要求（要求规范）

向 MAC 服务提供商表明 MAC 服务用户需要任何支持扩展过滤服务的设备按照 REQUIREMENT_SPECIFICATION 参数定义的服务要求向 Mac 服务用户方向转发帧。此参数可以携带的值有

a) 转发所有群组；
b) 转发未注册的群组。

注销服务要求（要求说明）

向 MAC 服务提供商表明 MAC 服务用户不再需要任何支持扩展过滤服务的设备按照 REQUIREMENT_SPECIFICATION 参数定义的服务要求向 Mac 服务用户方向转发帧。此参数可以携带的值有

a) 转发所有群组；
b) 转发未注册的群组。

这些服务的使用可能导致组 MAC 地址和服务需求信息在生成树中传播，影响桥接 LAN 中的网桥和终端站中的组注册条目 (7.9.3) 的内容，从而影响网桥和终端站对于多播帧的帧转发行为。

7. 操作原理

本条款建立了桥梁运行的原则和模型如下：

a) 解释 Bridge 操作的主要元素并列出支持这些元素的功能。
b) 建立一个桥梁的架构模型来管理这些功能的提供。
c) 从支持功能的流程和实体的角度，提供了桥接器操作的模型。

d) 详细说明桥接局域网中的寻址要求并指定桥接中实体的寻址。

7.1 桥梁运行

桥梁运行的主要要素包括

a) 帧的中继和过滤。
b) 维护做出帧过滤和中继决策所需的信息。
c) 以上内容的管理。

7.1.1 中继

MAC 桥接器在与其端口相连的桥接 LAN 的独立 MAC 之间中继各个 MAC 用户数据帧。帧的顺序应按照 7.7.3 中的定义保留。

支持帧中继和维护网桥支持的服务质量的功能包括

a)帧接收。
b) 丢弃收到的错误帧（6.3.2）。
c) 如果 frame_type 不是 user_data_frame，或者其 mac_action 参数不是 request_with_no_response （6.4），则丢弃帧。
d) 如果需要，重新生成用户优先级（6.4）。
e) 应用过滤信息后丢弃帧。
f) 可传输服务数据单元大小超出范围时丢弃帧（6.3.8）。
g) 将接收到的帧转发到其他桥接端口。
h) 根据过滤信息选择流量类别。
i) 根据流量类别对帧进行排队。
j) 丢弃帧以确保不超过最大桥接传输延迟（6.3.6）。
k) 选择排队的帧进行传输。
l) 出站访问优先级选择（6.3.9）。
m) 如果需要，映射服务数据单元并重新计算帧校验序列（6.3.7、7.7.6）。
n)帧传输。

7.1.2 过滤和传递信息

桥接器会过滤帧，即不会将桥接器端口接收到的帧中继到该桥接器上的其他端口，以防止帧重复（6.3.4）。支持为此目的使用和维护信息的功能是

a) 桥接局域网拓扑的计算和配 置。

桥接还会过滤帧，以减少桥接 LAN 中不位于流量源和目标之间路径的部分流量。支持为此目的使用和维护信息的功能包括：

b) 保留地址的永久配置。
c) 静态过滤信息的显式配置。
d) 通过观察桥接 LAN 流量的源地址，自动学习单播目标地址的动态过滤信息。

e) 已经学习到的动态过滤信息的老化。
f) 根据GMRP协议交换的结果自动添加和删除动态过滤信息。

网桥将帧分为不同的流量类别，以便加快传输关键或时间敏感型服务生成的帧。支持为此目的使用和维护信息的功能是

g) 明确配置与桥接端口相关的流量类别信息。

7.1.3 桥梁管理

支持桥梁管理的功能控制并监控上述功能的提供。它们在第14条中进行了规定。

7.2 桥梁架构

7.2.1 桥梁的建筑模型

图 7-1 给出了桥接 LAN 的物理拓扑示例。组件 LAN 通过 MAC 桥接器互连；MAC 桥接器的每个端口都连接到单个 LAN。图 7-2 说明了具有两个端口的桥接器，图 7-3 说明了这种桥接器的架构。

桥梁模型由以下部分组成

a) 连接网桥端口的 MAC 中继实体；
b) 至少两个港口；
c) 更高层实体，至少包括一个桥接协议实体。

7.2.2 MAC中继实体

MAC 中继实体处理 MAC 方法独立功能，即在桥接端口之间中继帧、过滤帧和学习过滤信息。它使用由每个端口的单独 MAC 实体提供的内部子层服务。（内部子层服务及其支持在 6.4 和 6.5 中描述。）帧在连接到不同 LAN 的端口之间中继。

7.2.3 端口

每个桥接端口都会向其所连接的 LAN 发送和接收帧。与端口永久关联的单个 MAC 实体提供用于帧传输和接收的内部子层服务。MAC 实体处理所有与 MAC 方法相关的功能（MAC 协议和程序），如 IEEE 802 LAN MAC 技术的相关标准中所述。

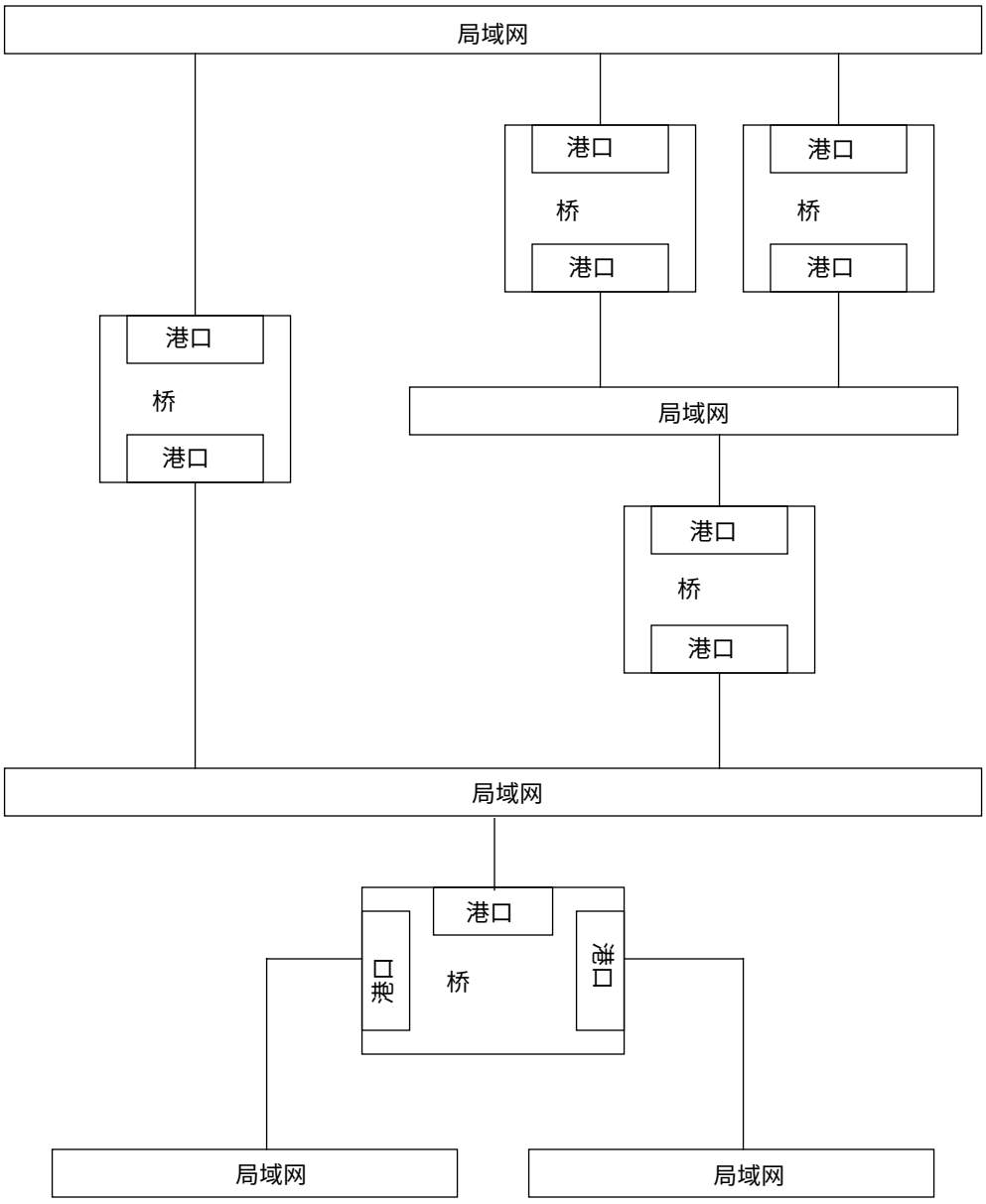


图 7-1—桥接局域网

7.2.4 高层实体

桥接协议实体负责桥接局域网拓扑的计算和配 置。

桥接协议实体和其他高层协议用户（例如桥接管理（7.1.3）和 GARP 应用实体（包括 GARP 参与者）（第 12 条））使用逻辑链路控制程序。这些程序为每个端口单独提供，并使用各个 MAC 实体提供的 MAC 服务。

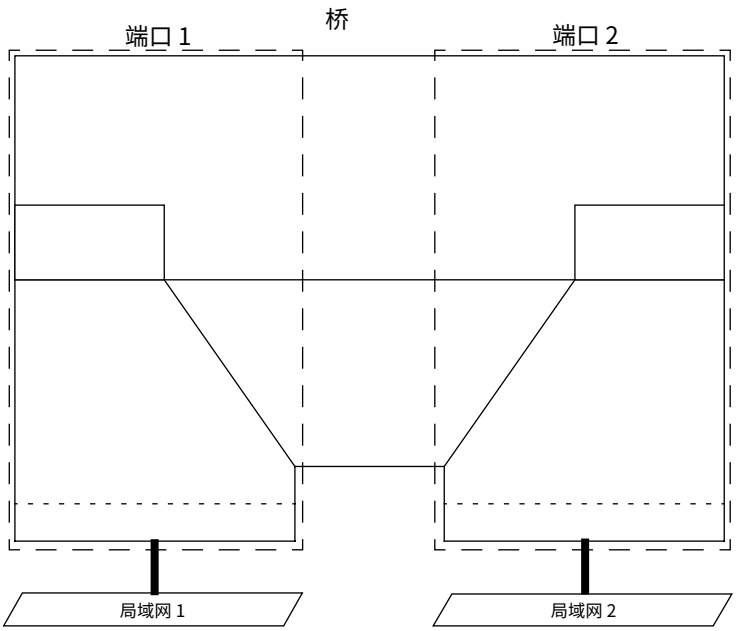


图 7-2 — 桥接端口

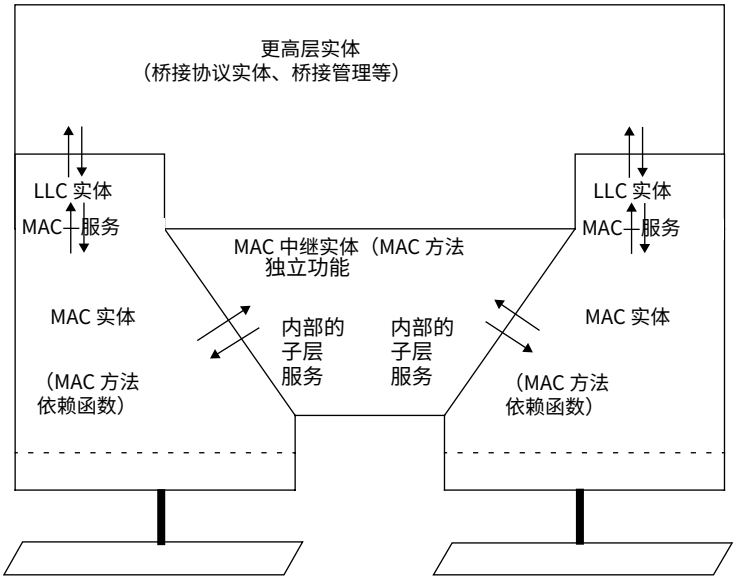


图 7-3 — 桥梁结构

7.3 运作模式

操作模型只是描述 MAC 桥接器功能的基础。它绝不旨在限制 MAC 桥接器的实际实现；这些实现可以采用与本标准指定的外部可见行为兼容的任何内部操作模型。设备是否符合本标准完全取决于可观察的协议。

子条款 7.5 和 7.6 规定了 MAC 中继实体对内部子层服务的使用。与每个端口相关的状态信息控制着端口对桥接 LAN 的参与。（端口状态在 8.4 中详细说明。）

帧被接受传输，并在接收时传送到和来自模拟网桥中 MAC 中继实体操作的进程和实体。这些是

a) 转发过程（7.7），它将要中继到其他桥接端口的接收到的帧转发出去，并根据过滤数据库（7.9）中包含的信息和桥接端口（7.4）的状态对帧进行过滤；

b) 学习过程（7.8）通过观察每个端口上接收到的帧的源地址，根据端口状态（7.4）更新过滤数据库（7.9）；
c) 过滤数据库（7.9），保存过滤信息并支持转发过程查询是否应将具有目标 MAC 地址字段的给定值的帧转发到给定端口。

每个桥接端口还可充当终端站，为 LLC 提供 MAC 服务，而 LLC 又支持桥接协议实体 (7.10) 以及 LLC 其他可能用户的操作，例如提供桥接管理 (7.11) 的协议。

每个桥接端口应支持 LLC 类型 1 程序的操作，以便支持桥接协议实体的操作。桥接端口可以支持其他类型的 LLC 程序，这些程序可能由其他协议使用。

图 7-4 说明了具有两个端口的桥接器的端口之间的帧中继的单个实例。

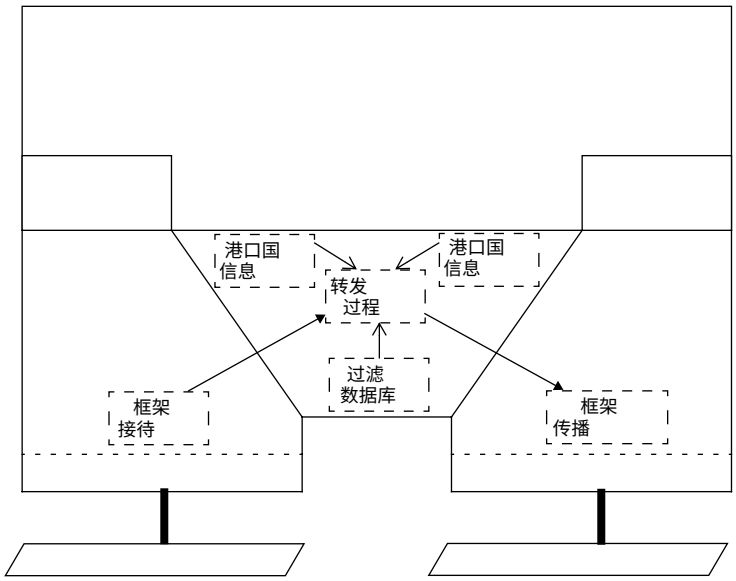


图 7-4 — 中继 MAC 帧

图 7-5 说明了在过滤数据库中包含由双端口桥接器的一个端口接收的单个帧所携带的信息。

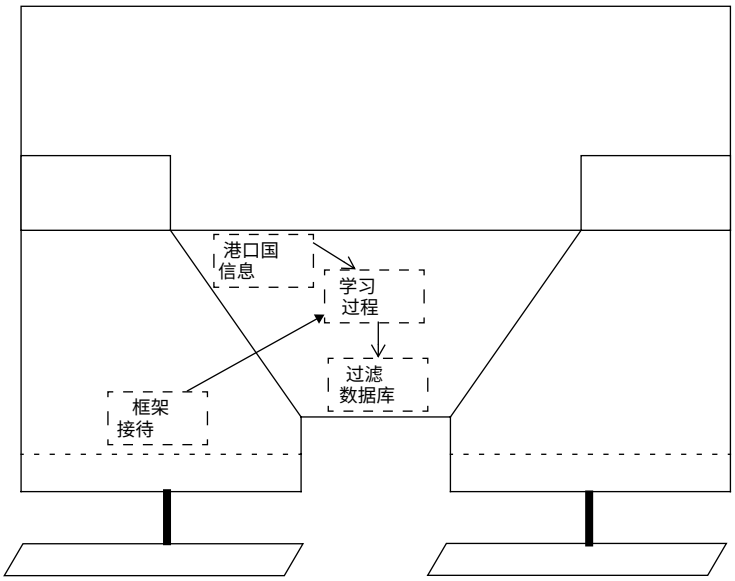


图 7-5—网络流量观察

图 7-6 说明了桥接协议实体对桥接协议数据单元的接收和传输。

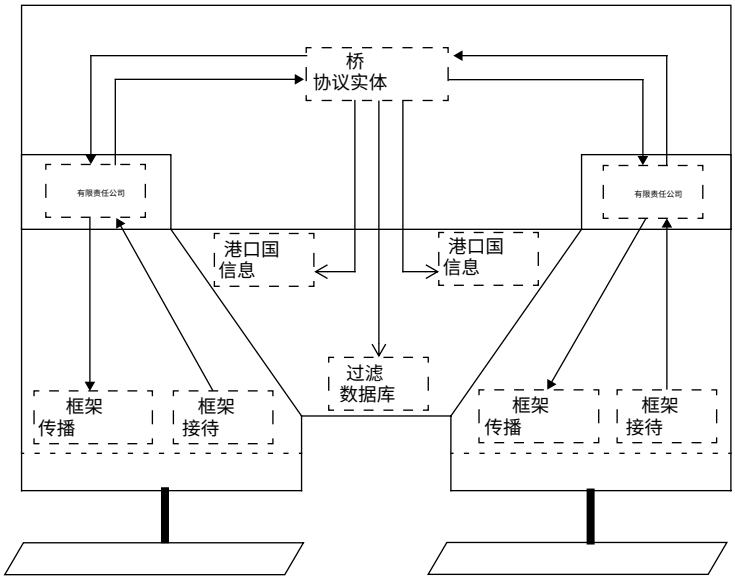


图 7-6—桥间协议的运行

图 7-7 说明了 GARP 协议实体（7.10）对 GARP 协议数据单元的接收和传输。

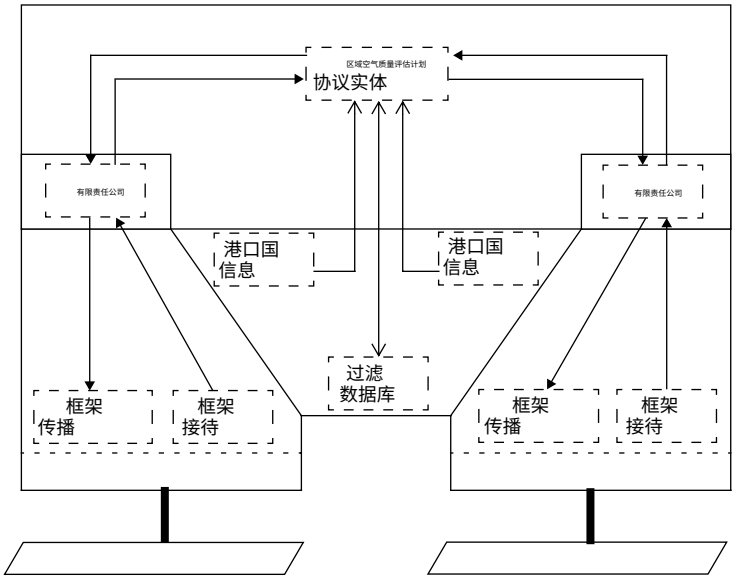


图 7-7—GARP 协议的运行

7.4 端口状态、活动端口和活动拓扑

与每个桥接端口相关的状态信息决定了它是否参与 MAC 帧的中继。端口可以通过管理禁用，在这种情况下，它不参与桥接 LAN 的运行；未被禁用的端口可以通过生成树算法的运行动态地排除参与帧中继。如果上述情况都不适用于端口，则描述为*转发*。

这*主动拓扑*桥接 LAN 的任意时刻的生成树是通过转发端口将 LAN 和网桥互连而形成的一组通信路径。分布式生成树算法（第 8 条）的功能是构建一个活动拓扑，该拓扑相对于用于寻址 LAN 上的终端站的任何给定 MAC 地址对之间的通信而言是简单连接的。

图 7-6 说明了桥接协议实体的操作，该实体操作生成树算法及其相关协议，并修改端口状态信息，作为确定桥接 LAN 的活动拓扑的一部分。与确定活动拓扑相关的端口状态在 8.4 中详细说明。

图 7-4 说明了转发过程对端口状态信息的使用：首先，对于接收帧的端口，以确定是否要通过任何其他端口中继接收的帧；其次，对于另一个端口，以确定是否要通过该特定端口转发中继的帧。

学习过程将终端站位置信息纳入过滤数据库还取决于活动拓扑。如果学习过程要将与端口上接收到的帧相关的信息纳入过滤数据库，则该端口被描述为处于学习状态；否则，它处于非学习状态。图 7-5 说明了学习过程如何使用接收帧的端口的端口状态信息来确定是否要将站点位置信息纳入过滤数据库。

7.5 帧接收

与每个桥接端口关联的单个 MAC 实体检查在其所连接的 LAN 上传输的所有帧。

所有无错误的接收帧都会产生 M_UNITDATA 指示原语，应按如下方式处理。

注——按照相关 MAC 规范的定义，错误的帧将被 MAC 实体丢弃，而不会产生任何 M_UNITDATA 指示；参见 6.4。

M_UNITDATA.indication 原语 frame_type 和 mac_action 参数值分别为 user_data_frame 和 request_with_no_response（6.4）的帧应提交给学习和转发过程。

具有其他 frame_type 和 mac_action 参数值的帧（例如 request_with_response 和 response 帧）不得提交给转发过程。它们可以提交给学习过程。

frame_type 为 user_data_frame 且以桥接端口作为终端站的帧应提交给 LLC。此类帧在目标地址字段中携带端口的单独 MAC 地址或与端口 (7.12) 关联的组地址。提交给 LLC 的帧也可以提交给学习和转发过程，如上所述。

以终端站为目的地发送到桥接端口的帧，以及通过转发过程从同一桥接器中的其他桥接端口中继到该桥接端口的帧也应提交给 LLC。

不得向 LLC 提交任何其他框架。

7.5.1 重新生成用户优先级

接收帧的 user_priority 是使用帧中包含的优先级信息和接收端口的用户优先级再生表来再生的。对于每个接收端口，用户优先级再生表有八个条目，对应于 user_priority 的八个可能值（0 到 7）。每个条目针对给定的接收 user_priority 值指定相应的再生 user_priority 值。

注 1—IEEE 802 LAN 技术最多可发送 8 个 user_priority 值。附件 H.2 进一步解释了 user_priority 值的使用及其如何映射到流量类别。

表 7-1 定义了数据指示中接收的 user_priority 参数的八个可能值的再生 user_priority 的默认值；这些值应用作每个端口的用户优先级再生表相应条目的初始值。

可选地，可以支持通过管理手段修改用户优先级再生表中的值的能力，如第 14 条所述。如果提供该能力，则可以为每个接收端口和每个接收到的 user_priority 值独立设置表条目的值，并且桥接器可以具有使用表 7-1 中指定的参数范围中的全部值的能力。

注 2 — 确保桥接器内用户优先级的再生和映射与桥接 LAN 中该用户优先级的端到端意义一致非常重要。在给定桥接器内，为给定端口选择的用户优先级再生表的值应与通过该端口在桥接 LAN 的其余部分接收的流量相关联的优先级一致，并且应为每种 MAC 方法生成适当的访问优先级值。接收时通过用户优先级再生表再生的用户优先级值用于：

— 通过流量类别表（7.7.3）确定给定出站端口的流量类别，以及
— 通过固定的、特定于 MAC 方法的映射（7.7.5）来确定用于给定出站 MAC 方法的访问优先级。

表 7-1 显示了用户优先级再生的默认值。表 7-2 显示了流量类别表的默认值，适用于所有可能支持的流量类别数量。表 7-3 显示了不同出站 MAC 方法所需的从用户优先级到访问优先级的固定映射。

表 7-1 — 用户优先级再生

用户 优先事项	默认再生 用户优先级	范围
0	0	0–7
1	1	0–7
2	2	0–7
3	3	0–7
4	4	0–7
5	5	0–7
6	6	0–7
7	7	0–7

7.6 帧传输

与每个桥接端口关联的单个 MAC 实体传输由 MAC 中继实体提交给它的帧。

中继帧由转发过程提交传输。与此类帧关联的 M_UNITDATA.request 原语传达在相应的 M_UNITDATA.indication 原语中收到的源地址和目标地址字段的值。

LLC 协议数据单元由 LLC 作为桥接端口提供的 MAC 服务的用户提交。传输此类协议数据单元的帧在源地址字段中携带端口的单独 MAC 地址。

每个帧的传输均遵循特定 IEEE 802 LAN 技术应遵守的 MAC 程序。相应 M_UNIT-DATA.request 原语的 frame_type 和 mac_action 参数的值应分别为 user_data_frame 和 request_with_no_response（6.5）。

根据桥接端口提供的 MAC 服务的 LLC 用户请求传输的帧也应提交给 MAC 中继实体。

7.7 转发过程

在任何给定桥接端口 (7.5) 收到帧后，提交给转发进程的帧将根据转发进程的组成功能通过其他桥接端口转发。这些功能强制执行拓扑限制 (7.7.1)、使用过滤数据库信息过滤帧 (7.7.2)、对帧进行排队 (7.7.3)、选择排队的帧进行传输 (7.7.4)、映射优先级 (7.7.5) 以及在需要时重新计算 FCS (7.7.6)。

7.7.1–7.7.6 中描述了转发过程功能，即针对在给定端口（称为“接收端口”）上接收到的给定帧采取的操作。该帧可转发以在某些端口（称为“传输端口”）上传输，而在其他端口上则被丢弃而不传输。

注——本标准中描述的转发过程操作模型仅限于 MAC 桥接器的中继功能操作，并未考虑在实际实现中将帧传递到 MAC 进行传输时可能发生的情况。在某些 MAC 实现中以及在某些流量条件下，将选定的帧传递到 MAC 进行传输的过程的模型描述与 LAN 介质本身上可见的实际帧序列之间可能会出现一定程度的不确定性。示例包括令牌传递总线 MAC 中 access_priority 的处理，或 FDDI LAN 中令牌保持时间的不同值的影响。当观察介质上的流量时，这种不确定性可能导致明显违反为转发过程描述的排队/出队和优先级规则。因此，在本标准的某些实现中，仅通过将观察到的 LAN 流量与所描述的转发过程模型相关联，可能无法测试是否符合标准；一致性测试也必须考虑MAC实现的（允许的）行为。

图 7-4 说明了双端口桥接器端口间帧中继的单个实例中的转发过程的操作。图 7-8 说明了转发过程的详细操作。

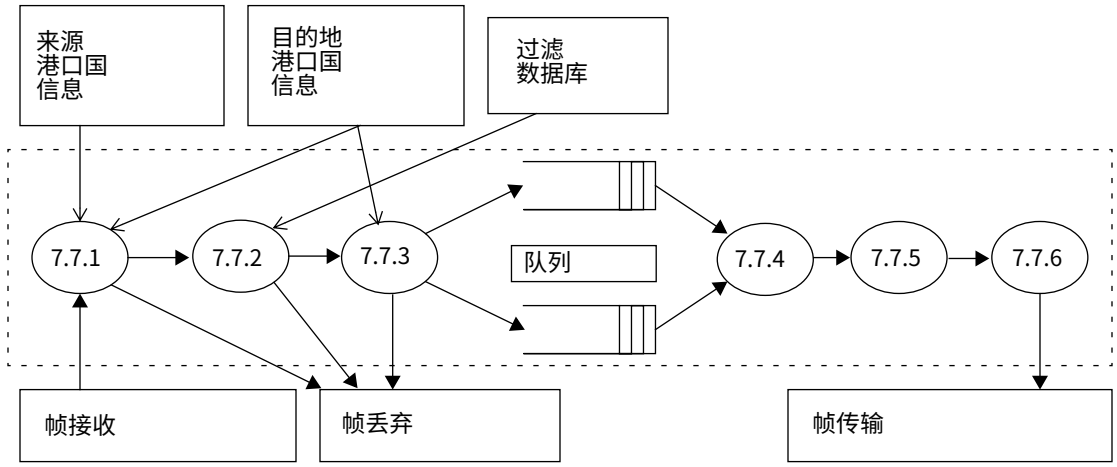


图 7-8—转发过程详细操作说明

7.7.1 强制拓扑限制

每个端口被选为潜在传输端口当且仅当

a) 接收帧的端口处于转发状态（8.4），并且
b）考虑传输的端口处于转发状态，并且
c) 考虑传输的端口与接收帧的端口不同，并且

d) 帧传达的 mac_service_data_unit 的大小不超过考虑传输的端口所连接的 LAN 支持的最大 mac_service_data_unit 大小。

对于每个未被选为潜在传输端口的端口，该帧应被丢弃。

7.7.2 过滤帧

转发过程根据以下情况做出过滤决定

a) 接收帧中携带的目标MAC地址；
b) 过滤数据库中包含的该MAC地址和接收端口的信息；
c) 潜在传输端口的默认组过滤行为（7.9.4）。

对于 7.7.1 中选择的每个潜在传输端口，应根据此信息转发或丢弃（即过滤）帧，并按照过滤数据库条目类型（7.9.1、7.9.2 和 7.9.3）的定义进行操作。所需的转发和过滤行为总结在 7.9.4、7.9.5、表 7-5、表 7-6 和表 7-7 中。

7.7.3 队列帧

转发过程为排队的帧提供存储，等待机会将这些帧提交给与每个桥接端口相关联的各个 MAC 实体进行传输。在同一桥接端口上接收的帧的顺序应保留

a) 对于给定的目的地址和源地址组合，具有给定用户优先级的单播帧；

b) 对于给定的目的地址，具有给定用户优先级的多播帧。

转发过程可能为给定的桥接端口提供多个传输队列。根据帧的用户优先级，使用流量类别表将帧分配到存储队列，该表是与每个端口相关联的状态信息的一部分。对于每个可能的 user_priority 值，该表指示应分配的流量类别的对应值。user_priority 的值范围从 0 到 7。队列与流量类别一一对应。

注 1—附件 H.2 进一步解释了 user_priority 值的使用以及它们如何映射到流量类别。

出于管理目的，流量类别表最多支持八个流量类别，以便为每个 user_priority 级别提供单独的队列。流量类别编号为 0 到 N-1，其中 N 是与给定出站端口关联的流量类别数。流量类别信息的管理是可选的。流量类别 0 对应于非加急流量；非零流量类别是加急流量类别。

注 2 — 在给定的桥接器中，允许为每个端口实现不同数量的流量类别。与支持单一传输优先级的 MAC 方法（如 CSMA/CD）关联的端口可以支持多个流量类别。

如果转发进程不支持给定端口的加速流量类别，换句话说，如果端口只有一个流量类别，则 user_priority 的所有值都将映射到流量类别 0。在支持加速流量的网桥中，对于实施的流量类别数量，建议将 user_priority 映射到流量类别，如表 7-2 所示。表主体中的每个条目都是分配给具有给定 user_priority 的流量的流量类别，对于给定数量的可用流量类别。

转发进程排队等待在端口上传输的帧在提交给该端口的单个 MAC 实体时应从该队列中删除。即使已知传输已失败，也不得再尝试在该端口上传输该帧。

如果已超出保证缓冲的时间，则转发过程排队等待在端口上传输的帧将从该队列中删除，并且不再进行传输。

如果有必要确保在随后传输帧时不会超过最大桥接传输延迟（6.3.6），则应将端口上排队等待传输的帧从该队列中删除。

表 7-2 - 建议的用户优先级到流量类别的映射

		可用流量类别的数量							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
用户优先级	**0（默认）**	0	0	0	1	1	1	1	2
	1	0	0	0	0	0	0	0	0
	2	0	0	0	0	0	0	0	1
	3	0	0	0	1	1	2	2	3
	4	0	1	1	2	2	3	3	4
	5	0	1	1	2	3	4	4	5
	6	0	1	2	3	4	5	5	6
	7	0	1	2	3	4	5	6	7

注：本表所示值选择背后的原理在 H.2 中进行了讨论。所示映射的结果是，在实施四个或更多流量类别的网桥中，携带默认用户优先级的帧将获得相对于用户优先级 1 和 2 的优先处理。

如果相关端口离开转发状态，则在端口上排队等待传输的帧将从该队列中删除。

从任何特定端口的队列中移除帧本身并不意味着该帧也应从任何其他端口的传输队列中移除。

7.7.4 选择要传输的帧

所有网桥都应支持以下算法作为选择传输帧的默认算法：

a) 对于每个端口，根据端口支持的流量类别选择帧进行传输。对于给定的流量类别支持值，仅当选择时端口支持的流量类别数值较高的所有队列都为空时，才会从相应的队列中选择帧进行传输；

b) 对于给定的队列，选择传输帧的顺序应保持 7.7.3 中规定的排序要求。

只要满足 7.7.3 的要求，可以通过管理手段选择的附加算法可以作为实施选项得到支持。

7.7.5 映射优先级

M_UNITDATA.request原语（6.4）中的user_priority参数应等于相应数据指示中的user_priority参数。

用户优先级到出站访问优先级的映射是通过固定的、特定于 MAC 方法的映射实现的。M_UNITDATA.request 原语 (6.4) 中的访问优先级参数应根据表 7-3 中所示的接收数据请求的 MAC 方法的值从用户优先级中确定。表 7-3 中所示的值不能通过管理或其他方式修改。

表 7-3 — 出站访问优先级

用户优先级	根据 MA C 方法的出站访问优先级								
	802.3	8802-4	8802-5（默认）	8802-5（备用）	8802-6	802.9a一个	8802.11	8802-12	光纤分布式数据接口
0	0	0	0	4	0	0	0	0	0
1	0	1	1	4	1	0	0	0	1
2	0	2	2	4	2	0	0	0	2
3	0	3	3	4	3	0	0	0	3
4	0	4	4	4	4	0	0	4	4
5	0	5	5	5	5	0	0	4	5
6	0	6	6	6	6	0	0	4	6
7	0	7	6	6	7	0	0	4	6

一个由于 IEEE Std 802.9a-1995 6.5 中没有支持的定义，因此假定对于此 MAC 方法，访问优先级 0 将映射到“低”。

表格中显示 8802-5 MAC 方法的两列。标有“8802-5（替代）”的列中包含映射，以便允许向后兼容按照 ISO/IEC 10038: 1993 制造的设备；但是，使用此映射会将可用的访问优先级值的数量减少到三个。因此，建议在向后兼容性不是问题的情况下，将标有“8802-5（默认）”的列作为默认映射。

7.7.6 重新计算FCS

当一个帧在同一 IEEE 802 LAN 类型的两个单独的 MAC 实体之间转发时，并且中继该帧不涉及对 FCS 覆盖范围内的数据进行任何更改，则 M_UNITDATA.indication 原语中收到的 FCS 可以在相应的 M_UNITDATA.request 原语中提供，并且不需要重新计算（6.3.7）。

当一个帧在两个不同类型的 MAC 实体之间转发时，如果 MAC 方法之间的差异使得根据目标 MAC 方法的 MAC 程序计算出的 FCS 与接收帧携带的 FCS 不同，或者如果中继该帧涉及更改 FCS 覆盖范围内的数据，则需要重新计算 FCS。必要时，将根据传输 MAC 实体的特定 MAC 程序重新计算 FCS。

注：有两种方法可以重新创建有效的 FCS。第一种是通过算法修改接收到的 FCS 来生成新的 FCS，这基于对 FCS 算法的了解以及帧在接收和传输之间经历的转换。第二种方法是依靠正常的 MAC 程序来重新计算

传出帧的 FCS。前一种方法可能更可取，因为它能够防止未检测到的帧错误水平增加。附件 G 更详细地讨论了这些可能性。内部子层服务 (6.4) 的 frame_check_sequence 参数能够发出 FCS 的有效性或其他信息；数据请求中此参数中未指定的值向传输 MAC 指示接收到的 FCS 不再有效，因此必须重新计算 FCS。

如果下列任一情况成立，则需要重新计算 FCS：

a) 两个MAC实体所使用的MAC方法之间确定FCS所用的算法不同；

b) 两个MAC实体所使用的MAC方法之间的FCS覆盖范围有所不同；
c) 在两个MAC实体之间中继帧涉及对FCS覆盖范围内的数据进行更改。

7.8 学习过程

学习过程观察每个端口上接收的帧的源地址，并根据接收端口的状态有条件地更新过滤数据库。

帧由与每个桥接端口相关联的各个 MAC 实体提交给学习过程，如 7.5 中所述。

学习过程可以通过检查收到的帧的源地址字段推断出通过桥接 LAN 到达特定终端站的路径。当且仅当满足以下条件时，它才应在过滤数据库中创建或更新动态过滤条目 (7.9、7.9.2)，将接收 帧的端口与帧源地址字段中的 MAC 地址相关联

a) 接收帧的端口处于允许学习状态（8.4），并且
b）该帧的源地址字段表示一个特定的终端站，即不是一个组地址，并且
c) 不存在与相关 MAC 地址对应的静态过滤条目 (7.9、7.9.1)，且端口映射中未指定该端口的转发或过滤，并且
d) 得到的条目数不会超出过滤数据库的容量。

如果过滤数据库已填满其容量，但可以创建新条目，则可以删除现有条目以便为新条目腾出空间。

图 7-5 说明了学习过程的操作，其中包括在桥接器的一个端口上接收的单个帧所携带的站点位置信息，并将其包含在过滤数据库中。

7.9 过滤数据库

过滤数据库支持转发进程查询转发进程从给定接收端口接收到的帧是否要通过给定的潜在传输端口转发（7.7.1、7.7.2）。它以过滤条目的形式包含过滤信息，这些条目可以是

a) 静态的，并通过管理操作明确配置；或
b) 动态的，通过桥接器的正常运行及其支持的协议自动输入到过滤数据库中。

静态过滤条目是一种单一条目类型，它代表过滤数据库中所有静态信息，包括单个 MAC 地址和组 MAC 地址。它允许管理控制

c) 转发具有特定目标地址的帧；以及
d) 在过滤数据库中纳入与扩展过滤服务相关的动态过滤信息，并使用该信息。

过滤数据库应包含静态过滤条目类型的条目。

静态过滤信息只能在明确的管理控制下添加、修改和从过滤数据库中删除。任何老化机制都不能自动删除静态过滤信息。静态过滤信息的管理可以通过使用桥梁管理 (7.11) 提供的远程管理功能，使用第 14 条中规定的操作进行。

两种条目类型用于表示动态过滤信息。动态过滤条目用于指定已学习单个地址的端口。它们由学习过程 (7.8) 创建和更新，并受过滤数据库的老化和删除。组注册条目支持组 MAC 地址的注册。它们由 GMRP 协议创建、更新和删除，以支持扩展过滤服务 (6.6.5、7.9.3 和第 10 条)。可以使用第 14 条中指定的操作，使用桥接管理 (7.11) 提供的远程管理功能来读取动态过滤信息。

静态和动态条目均包括

e) MAC 地址规范；
f) 端口映射，每个出站端口都有一个控制元素，用于指定 MAC 地址规范的过滤。

网桥支持的过滤服务（基本过滤服务与扩展过滤服务）决定了网桥在转发发往组 MAC 地址的帧方面的默认行为。在支持扩展过滤服务的网桥中，每个端口针对组 MAC 地址的默认转发行为可以通过静态过滤条目和/或组注册条目以静态和动态方式配置，这些条目可以承载以下 MAC 地址规范：

g) 所有组地址，不再存在更具体的静态过滤条目；
h) 所有未注册的组地址（即，所有不存在组注册条目的组 MAC 地址），并且不存在更具体的静态过滤条目。

注意：在静态过滤条目中使用“所有组地址”规范（上面的 g 项）并结合适当的控制规范，可以将支持扩展过滤服务的网桥配置为在其部分或全部端口上仅支持基本过滤服务的网桥。这样做的原因可能如下：

— 相关端口为希望接收多播流量但无法注册组成员身份的“传统”设备提供服务；

— 相关端口为需要接收所有多播流量的设备提供服务，例如路由器或诊断设备。

过滤数据库应支持学习过程 (7.8) 创建、更新和删除动态过滤条目。在支持扩展过滤服务的网桥中，过滤数据库应支持 GMRP 创建、更新和删除组注册条目（第 10 条）。

图 7-4 说明了在具有两个端口的桥接器的端口之间的单个帧中继实例中转发过程如何使用过滤数据库。

图 7-5 说明了学习过程在过滤数据库中创建或更新动态条目。

图 7-6 说明了桥接协议实体 (7.10) 的操作，该实体操作生成树算法和协议，并将其通过该协议发信号通知给过滤数据库的活动拓扑变化。

7.9.1 静态过滤条目

静态过滤条目指定

a) MAC地址规范，包括
   1) 单个 MAC 地址；或
   2) 组 MAC 地址；或
   3) 所有组地址，对于这些组地址，不再存在更具体的静态过滤条目；或者
   4) 所有未注册的组地址，对于这些组地址，不存在更具体的静态过滤条目。
b) 端口映射，包含每个出站端口的控制元素，指定符合此规范的目标 MAC 地址的帧将被

   1) 转发，独立于过滤数据库所保存的任何动态过滤信息；或

   2）经过过滤，独立于任何动态过滤信息；或
   3) 根据动态过滤信息进行转发或过滤，如果没有专门针对 MAC 地址的动态过滤信息，则根据出站端口的默认组过滤行为（7.9.4）进行转发或过滤。

所有网桥都应有能力支持 MAC 地址规范的前两个值，以及所有静态过滤条目的每个控制元素的前两个值（即，应有能力支持上面的 a1、a2、b1 和 b2）。

支持扩展过滤服务的网桥应具有支持 MAC 地址规范的所有四个值和指定组 MAC 地址的静态过滤条目的所有三个控制元素值的能力，并应具有支持指定单个 MAC 地址的静态过滤条目的所有三个控制元素值的能力（即，应具有支持 a1 到 a4 的能力，除了支持 b1 和 b2 之外，还应具有支持 b3 的能力）。

对于给定的 MAC 地址规范，可以为转发过程接收帧的每个入站端口创建具有不同端口映射的单独静态过滤条目。

除了控制帧的转发之外，组 MAC 地址的静态过滤条目还为 GMRP 协议 (10、12、12.9.1) 提供注册管理控制值。将特定组地址的帧转发到出站端口的静态配置表示注册已在该端口上固定：即使没有动态信息，也希望接收发往该组的帧。将原本可能发送到出站端口的帧进行静态过滤配置表示注册被禁止。组地址没有静态过滤条目，或者基于动态过滤信息进行转发或过滤的配置表示注册正常。

注意：配置多个静态过滤条目（每个条目用于不同的入站端口）可能会使注册控制变得复杂。组注册通过桥接传播到入站端口，如果传播信息的任何部分表明希望接收组的帧，则该端口将充当 GMRP 申请人。这些帧将从不希望接收它们的端口过滤掉，并保持正确的操作。

7.9.2 动态过滤条目

动态过滤条目指定

a) 单独的 MAC 地址；

b) 端口映射由控制元素组成，该控制元素指定将发往该 MAC 地址的帧转发到单个端口。

注 1—这相当于指定单个端口号；因此，此规范直接相当于 ISO/IEC 10038: 1993 中的动态条目规范。

动态过滤条目由学习过程 (7.8) 创建和更新。自条目创建或上次更新以来，经过指定的时间（老化时间）后，它们将被自动删除。对于给定的 MAC 地址，过滤数据库中最多只能创建一个动态过滤条目。

如果此 MAC 地址已经存在任何静态过滤条目，并且该条目具有控制元素规范，对于学习过程指定的出站端口，该条目指定转发或过滤而不考虑动态过滤信息，则学习过程不得创建或更新动态过滤条目。

注 2 — 对于不允许静态过滤条目根据动态过滤信息指定转发或过滤（见 7.9.1）的网桥，包括符合 ISO/IEC 10038: 1993 的网桥，这有效地阻止了在同一 MAC 地址存在静态过滤条目的情况下创建动态过滤条目。这反过来确保这些网桥继续符合其自己的规范，如果静态过滤条目已经存在，则禁止创建动态过滤条目。

对于允许静态过滤条目根据动态过滤信息指定转发或过滤功能的网桥，同一 MAC 地址可以同时存在动态和静态过滤条目，只要该地址不是在具有指定“独立于任何动态过滤信息进行转发或过滤”的静态过滤条目的端口上学习到的。

此更新规范提供的功能允许将源地址学习限制在端口的子集中。

管理层无法创建或更新动态过滤条目。

如果给定 MAC 地址存在动态过滤条目，则创建或更新同一地址的静态过滤条目会导致删除动态过滤条目中可能包含的任何冲突信息。如果删除此类冲突信息会导致端口映射未在任何端口上指定转发，则该动态过滤条目将从过滤数据库中删除。

动态过滤条目的老化可确保已移动到桥接 LAN 不同部分的终端站不会永久无法接收帧。它还考虑了桥接 LAN 的活动拓扑变化，这些变化可能导致终端站从桥接器的角度看起来似乎在移动；即，到这些终端站的路径随后通过不同的桥接端口。

老化时间可由管理层设置（第 14 条）。表 7-4 中指定了适用值的范围和建议的默认值；建议在大多数情况下无需明确配置。如果老化时间的值可由管理层设置，则网桥应能够使用指定范围内的值，粒度为 1 秒。

表 7-4—老化时间参数值

范围	受到推崇的 默认值	范围
老化时间	300.0 秒	10.0～1 000 000.0 秒

注 3—此处指定粒度是为了为第 14 条中定义的管理操作中表达的粒度建立通用基础，而不是为了限制符合要求的实现所支持的实际计时器的粒度。如果实现支持的粒度不是 1 秒，则管理在 Set 操作后读回的值可能与 Set 中表达的实际值不匹配。

第 8 条中规定的生成树算法和协议包括一个程序，用于通知桥接 LAN 中的所有网桥拓扑变化。它为老化计时器指定了一个短值，在任何拓扑变化后都会强制执行一段时间（参见 8.3.5）。在拓扑不变的情况下，此程序允许正常老化以适应较长的时间段，在此期间寻址终端站不会自行生成帧（可能是通过关闭电源），而不会牺牲桥接 LAN 在自动配置后继续提供服务的能力。

7.9.3 组注册表项

组注册条目指定

a) MAC地址规范，包括
    1) 组 MAC 地址；或
    2) 所有组地址，对于这些组地址，不再存在更具体的静态过滤条目；或者
    3) 所有未注册的组地址，对于这些组地址，不存在更具体的静态过滤条目。
b) 端口映射由每个出站端口的控制元素组成，用于指定发往 MAC 地址的帧的转发或过滤。

组注册条目由 GMRP 操作创建、修改和删除（第 10 条）。对于给定的 MAC 地址规范，过滤数据库中最多只能创建一个组注册条目。

注意：可以有一个静态过滤条目，该条目在某些或所有端口上具有转发或过滤值，以屏蔽相应组注册条目中保存的动态值。组注册条目中的值将继续由 GMRP 更新；因此，随后修改该条目以允许在一个或多个端口上使用动态过滤信息会立即激活迄今为止被静态信息屏蔽的真实 GMRP 注册状态。

7.9.4 默认组过滤行为

可以通过指定每个出站端口的默认值来管理组寻址帧的转发和过滤。这些默认值中的每一个的行为（由适用于给定帧的 MAC 地址、接收端口和出站端口的更明确的过滤数据库条目的控制元素修改）如下。

注 1—如 7.9.1 所述，网桥可以选择性地提供为每个接收端口创建具有不同端口映射的单独静态过滤条目的功能。如果不提供此功能，则给定的静态过滤条目将适用于所有接收端口。

一个)*转发所有群组*. 转发该帧，除非明确的静态过滤条目指定了独立于任何动态过滤信息的过滤。

b) *转发未注册的群组*. 帧被转发，除非
    1）显式静态过滤条目指定独立于任何动态过滤信息的过滤；或

    2）显式静态过滤条目根据动态过滤信息指定转发或过滤，并且存在适用的显式组注册条目指定过滤；或

    3) 不存在适用的显式静态过滤条目，但适用的组注册条目指定了过滤。

c) *过滤未注册的群组*. 除非

1）显式静态过滤条目指定独立于任何动态过滤信息的转发；或

2）显式静态过滤条目指定转发或根据动态过滤信息进行过滤，并且存在适用的显式组注册条目指定转发；或

3) 不存在适用的显式静态过滤条目，但适用的组注册条目指定转发。

在仅支持基本过滤服务的网桥中，默认的组过滤行为是针对网桥的所有端口转发所有组。

注 2 — 转发所有组直接对应于 ISO/IEC 10038: 1993 中指定的行为，即转发过滤数据库中不存在静态过滤信息的组 MAC 地址帧。转发所有组利用特定组 MAC 地址的静态过滤条目中包含的信息，但会覆盖组注册条目中包含的任何信息。转发未注册组类似于网桥针对单个 MAC 地址的转发行为；如果特定组 MAC 地址没有静态或动态信息，则转发该帧，否则，根据静态配置或动态学习的信息转发该帧。

在支持扩展过滤服务的网桥中，每个出站端口的默认组过滤行为由过滤数据库中包含的以下信息决定：

d) 适用于接收端口的任何静态过滤条目，其 MAC 地址规范为所有组地址或所有未注册的组地址；

e) 任何适用于接收端口的组注册条目，其 MAC 地址规范为所有组地址或所有未注册的组地址。

该信息确定默认组过滤行为的方法在 7.9.5、表 7-6 和表 7-7 中指定。

注 3 — 结果是，可以通过静态过滤条目为桥的每个端口配置默认组过滤行为，通过 GMRP 创建/更新的组注册条目动态确定（条款 10），或两者兼而有之。例如，如果过滤数据库中没有所有组地址或所有未注册组地址的任何静态或动态信息，则默认组过滤行为将是过滤所有端口上的未注册组。随后，在给定端口上为所有未注册组地址创建指示“已注册”的动态组注册条目将导致该端口表现出转发未注册组行为。同样，在给定端口上为所有组地址创建指示“注册已修复”的静态过滤条目将导致该端口表现出转发所有组行为。

因此，通过在静态过滤条目的端口映射中为所有组地址和所有未注册组地址规范使用“注册固定”、“注册禁止”和“正常注册”的适当组合，对于给定端口，可以

— 将默认的组过滤行为修复为上述三种行为之一；或
— 将行为选择限制为三者的子集，并允许 GMRP 注册（或缺失）决定最终选择；或者

— 根据通过 GMRP 收到的任何注册，允许采用三种行为中的任意一种。

7.9.5 查询过滤数据库

过滤数据库中的每个条目包括

a) MAC 地址规范；
b) 端口图，每个出站端口都有一个控制元素。

给定的单个 MAC 地址规范可以包含在过滤数据库中的静态过滤条目、动态过滤条目、两者或两者都不包含。表 7-5 结合了静态过滤条目和动态

过滤单个 MAC 地址的条目信息，以指定通过出站端口转发或过滤具有该目标地址的帧。

表 7-5 — 结合单个 MAC 地址的静态和动态过滤条目

过滤信息	此单个 MAC 地址和端口的静态过滤条目控制元素指定：				
	向前	筛选	使用动态过滤信息，或不存在静态过滤条目。此动态过滤条目控制元素 单独的 MAC 地址和端口指定：		
			向前	筛选	无动态过滤入场礼物
结果	向前	筛选	向前	筛选	向前

表 7-6 指定了对于“所有组地址”地址规范和“所有未注册的组地址”地址规范，将静态过滤条目与组注册条目组合的结果（已注册或未注册）。

表 7-6 — 结合静态过滤条目和组注册条目，用于“所有组地址”和“所有未注册的组地址”

过滤信息	该组 MAC 地址和端口的静态过滤条目控制元素指定：				
	登记固定的（向前）	登记禁止（筛选）	使用组注册信息，或不存在静态过滤条目。此组的组注册条目控制元素 组 MAC 地址和端口指定：		
			挂号的（向前）	不是挂号的（筛选）	没有群组登记入口展示
结果	挂号的	未注册	挂号的	未注册	未注册

表 7-7 将特定组 MAC 地址的静态过滤条目和组注册条目信息与表 7-6 中所有组地址和所有未注册的组地址的结果相结合，以指定通过出站端口转发或过滤具有该目标组 MAC 地址的帧。

7.9.6 永久数据库

永久数据库为许多静态过滤条目提供固定存储空间。过滤数据库应使用此固定数据存储中包含的过滤数据库条目进行初始化。

表 7-7 — 特定组 MAC 地址的转发或过滤

				静态过滤条目控制元素 组 MAC 地址和端口指定：				
				登记 固定的 （向前）	登记 禁止 （筛选）	使用组注册信息，或不存在静态过滤条目。 组注册条目控制元素 本组 MAC 地址和端口指定：		
						挂号的 （向前）	不是 挂号的 （筛选）	没有群组 登记 入场礼物
所有组地址控制元素 对于此端口指定（表7-6）：	未注册	所有未注册的组地址 此端口的控制元素 具体规定（表7-6）：	未注册	向前	筛选	向前	筛选	筛选 （筛选 未註冊 团体）
			挂号的	向前	筛选	向前	筛选	向前 （向前 未註冊 团体）
	挂号的			向前	筛选	向前 （转发全部 团体）	向前 （转发全部 团体）	向前 （转发全部 团体）

使用第 14 条中定义的管理功能，可以在明确的管理控制下向永久数据库添加和从永久数据库删除条目。永久数据库中静态过滤条目内容的更改不会影响转发过程所做的转发和过滤决策，直到过滤数据库重新初始化为止。

注意——永久数据库的这个方面可以被看作是为过滤数据库提供“启动映像”，在添加任何动态过滤信息之前定义所有初始条目的内容。

7.10 桥接协议实体和GARP协议实体

桥接协议实体操作生成树算法和协议。

连接到桥接 LAN 中给定单个 LAN 的桥接器的桥接协议实体通过交换桥接协议数据单元 (BPDU) 进行通信。

图 7-6 说明了桥接协议实体的操作，包括包含 BPDU 的帧的接收和传输、与各个桥接端口相关的状态信息的修改、以及向过滤数据库通知活动拓扑的变化。

GARP 协议实体操作与桥接器支持的 GARP 应用程序相关的算法和协议，并由这些 GARP 应用程序的 GARP 参与者集合组成（12.3，第 10 条）。

连接到桥接 LAN 中给定单个 LAN 的桥接器的 GARP 协议实体通过交换 GARP 协议数据单元 (GARP PDU) 进行通信。

图 7-7 说明了 GARP 协议实体的操作，包括包含 GARP PDU 的帧的接收和传输、过滤数据库中包含的控制信息的使用以及过滤数据库过滤信息变化的通知。

7.11 桥梁管理

远程管理设施可由桥接器提供。桥接器管理被建模为通过桥接器管理实体执行。桥接器管理提供的设施以及支持这些设施的操作在第 14 条中指定。

第 15 条指定了通过使用 ISO/IEC 15802-2 实现这些管理操作所使用的协议操作、标识符和值。

桥接管理协议使用 LLC 程序操作所提供的服务，而 LLC 程序操作又使用桥接 LAN 提供的 MAC 服务。

7.12 寻址

所有通过桥接 LAN 进行通信的 MAC 实体都应使用 48 位地址。这些地址可以是通用管理地址、本地管理地址，或两者的组合。

7.12.1 终端站

使用桥接 LAN 提供的 MAC 服务在终端站之间传输的帧分别在帧的源和目标地址字段中携带源和目标对等终端站的 MAC 地址。桥接 LAN 中用于帧中继的对等用户之间传输的帧不携带桥的地址或其他标识方式。

广播地址和其他组 MAC 地址适用于整个桥接 LAN 提供的 MAC 服务的使用。如果没有通过管理配置为静态过滤条目或通过 GMRP 配置为组注册条目（第 14 条、第 10 条、7.9 条）的明确过滤器，则具有此类目标地址的帧将在整个桥接 LAN 中中继。

7.12.2 桥接端口

与每个桥接端口关联的单个 MAC 实体应具有单独的 MAC 地址。此地址用于所采用的特定 MAC 方法所需的任何 MAC 程序。

从端口所连接的 LAN 接收到的帧，如果其目标地址字段中带有端口的 MAC 地址，则这些帧将完全按照终端站的方式提交给 MAC 服务用户 (LLC)。

7.12.3 桥接协议实体和GARP协议实体

桥接协议实体仅接收和传输 BPDU。这些仅从其他桥接协议实体接收和传输（或当两个桥接端口连接到同一个 LAN 时，它们彼此之间进行通信）。

GARP 协议实体仅接收和传输根据其支持的 GARP 应用程序的要求格式化的 GARP PDU (12.11)。这些仅从其他 GARP 协议实体接收和传输。

桥接协议实体或 GARP 协议实体使用与每个活动桥接端口相关联的各个 LLC 实体提供的 DL_UNITDATA.request 原语（参见 ISO/IEC 8802-2）来传输 BPDU 或 GARP PDU。每个 PDU 在一个选定的桥接端口上传输。PDU 通过相应的 DL_UNITDATA.indication 原语接收。DL_UNITDATA.request 原语的 source_address 和 destination_address 参数均应表示分配给桥接生成树协议的标准 LLC 地址。这可在 LLC 的其他用户中识别桥接协议实体和 GARP 协议实体。

每个 DL_UNITDATA.request 原语都会引起 LLC UI 命令 PDU 的传输，该 PDU 在其信息字段中传送 BPDU 或 GARP PDU。源和目标 LLC 地址字段设置为请求原语中提供的值。

表 7-8 给出了分配给桥接生成树协议 LLC 地址的值。[9]

表 7-8—标准 LLC 地址分配

任务	价值
桥接生成树协议	01000010

代码表示：显示值的最低有效位位于最右边。从右到左，位的有效位依次增加。应该注意的是，此处使用的代码表示是为了与本标准其他地方使用的表示保持一致而选择的；但是，它与 ISO/IEC TR 11802-1: 1997 中使用的表示不同。

本标准定义了协议标识符字段，该字段存在于所有 BPDU（第 9 条）和 GARP PDU（12.11 条）中，用于在 LLC 地址分配范围内识别桥接协议实体和 GARP 协议实体支持的不同协议。本标准指定了第 9 条中协议标识符的单个值，用于 BPDU。该值用于识别在桥接协议实体之间交换的 BPDU，这些实体运行第 8 条中指定的生成树算法和协议。该协议标识符的第二个值用于 GARP PDU，定义在 12.11 条中。该值用于识别在 GARP 参与者之间交换的 GARP PDU，这些参与者运行第 12 条中指定的 GARP 协议。该字段的其他值保留用于将来的标准化。

收到具有未知协议标识符的 BPDU 或 GARP PDU 的桥接协议实体或 GARP 协议实体应丢弃该 PDU。

运行第 8 条中规定的生成树算法和协议的桥接协议实体始终会将 BPDU 发送到与包含 BPDU 的帧所传输的 LAN 相连的所有其他桥接协议实体。应在目标地址字段中使用组地址来寻址该实体组。应在永久数据库 (7.12.6) 中配置此组地址，以便将 BPDU 限制在其所传输的单个 LAN 中。

[9]ISO/IEC TR 11802-1： 1997，信息技术 - 系统间电信和信息交换 - 局域网和城域网 - 技术报告和指南 - 第 1 部分：局域网中逻辑链路控制地址的结构和编码，包含标准 LLC 地址分配的完整列表，并记录了分配的标准。

为此，已分配了一个 48 位通用地址，称为桥组地址。其值在表 7-9 中指定。使用 48 位通用管理地址的桥应在所有传送 BPDU 的 MAC 帧的目标地址字段中使用此地址。

表 7-9—保留地址

任务	价值
桥组地址	01-80-C2-00-00-00
IEEE 标准 802.3x 全双工暂停操作	01-80-C2-00-00-01
为未来标准化保留	01-80-C2-00-00-02
为未来标准化保留	01-80-C2-00-00-03
为未来标准化保留	01-80-C2-00-00-04
为未来标准化保留	01-80-C2-00-00-05
为未来标准化保留	01-80-C2-00-00-06
为未来标准化保留	01-80-C2-00-00-07
为未来标准化保留	01-80-C2-00-00-08
为未来标准化保留	01-80-C2-00-00-09
为未来标准化保留	01-80-C2-00-00-0A
为未来标准化保留	01-80-C2-00-00-0B
为未来标准化保留	01-80-C2-00-00-0C
为未来标准化保留	01-80-C2-00-00-0D
为未来标准化保留	01-80-C2-00-00-0E
为未来标准化保留	01-80-C2-00-00-0F

GARP 协议实体

a) 运行第12条规定的GARP协议；并且
b) 支持给定的 GARP 应用程序，

始终向所有其他 GARP 协议实体发送 GARP PDU，这些实体

c) 实施相同的 GARP 应用程序；并且
d) 连接到传输包含 GARP PDU 的帧的 LAN。

特定于相关 GARP 应用程序的组 MAC 地址应用作目标 MAC 地址字段，以寻址该组 GARP 协议实体。为此目的，已分配了一组 48 位通用地址，称为 GARP 应用程序地址。GARP 应用程序地址的值在表 12-1 中定义。这些组 MAC 地址保留用于分配给标准协议，根据此类分配的标准（ISO/IEC TR 11802-2 第 5.5 条）。

在仅提供基本过滤服务的网桥中，不应在过滤数据库 (7.9) 或永久数据库 (7.9.6) 中配置 GARP 应用程序地址集。在提供

扩展过滤服务，GARP 应用程序地址集应配置为过滤数据库（7.9.1）和永久数据库（7.9.6）中的静态过滤条目，如下所示：

e) 应配置分配给网桥支持的 GARP 应用程序的 GARP 应用程序地址，以便将该 GARP 应用程序的 GARP PDU 限制在传输它们的单个 LAN 上；

f) 分配给网桥不支持的 GARP 应用程序的 GARP 应用程序地址不得在过滤数据库或永久数据库中配置。

管理层不得提供在永久数据库或过滤数据库中为任何 GARP 应用程序地址创建、删除或修改条目的功能。

传送 BPDU 或 GARP PDU 的 MAC 帧的源地址字段包含传输 PDU 的桥接端口的单独 MAC 地址 (7.12.2)。

7.12.4 桥梁管理实体

桥接管理实体使用与每个桥接端口相关联的各个 LLC 实体提供的服务来传输和接收协议数据单元。每个桥接管理实体又使用 MAC 服务，该服务由与该端口相关联的各个 MAC 实体提供，并由整个桥接 LAN 提供支持。

作为桥接 LAN 提供的 MAC 服务的用户，桥接管理实体可以连接到桥接 LAN 中的任何点。如果需要，桥接器会中继发送到桥接管理实体的帧，以到达其所连接的 LAN。

为了确保接收到的帧不重复，单个 LAN 或桥接 LAN 中的基本要求是必须满足每个连接点都关联一个唯一地址。

特定桥的桥管理实体由一个或多个单独的 MAC 地址以及高层协议标识符和寻址信息寻址。它可以与与其关联的桥的端口共享一个或多个桥接 LAN 连接点。建议它利用与每个桥端口关联的所有 MAC 实体提供的 MAC 服务，即，它可以通过每个桥端口使用在目标地址字段中携带该端口单独 MAC 地址的帧来访问。

此标准指定了一个供公众使用的标准组地址，用于将管理请求传达给与连接到桥接 LAN 的所有桥接端口相关联的桥接管理实体。在目标地址字段中携带此地址值的 MAC 帧中传达的管理请求通常会引发单个桥接的多个响应。此地址称为所有 LAN 桥接管理组地址，其值在表 7-10 中指定。

表 7-10 — 桥接管理地址

任务	价值
所有 LAN 桥接管理组地址	01-80-C2-00-00-10

7.12.5 桥梁的唯一标识

应为每个桥分配一个唯一的 48 位通用管理 MAC 地址，称为桥地址。桥地址可以是桥端口的单独 MAC 地址，在这种情况下建议使用编号最小的桥端口（端口 1）的地址。

注：生成树协议（第 8 条）要求每个网桥都关联一个唯一标识符。该标识符源自 8.5.1.3、8.5.3.7 和 9.2.5 中规定的网桥地址。

7.12.6 保留地址

目标地址字段中包含表 7-9 中指定的任何组 MAC 地址的帧不得由网桥中继。它们应在永久数据库中配置。管理不得提供从永久数据库或过滤数据库中修改或删除这些条目的功能。这些组 MAC 地址保留用于分配给标准协议，具体分配标准为此类分配标准（ISO/IEC TR 11802-2 第 5.5 条）。

7.12.7 高层实体的连接点和连接点

更高层实体（例如桥接协议实体和 GARP 协议实体 (7.10) 以及桥接管理 (7.11)）被建模为通过一个或多个连接点直接连接到桥接 LAN。从它们与桥接 LAN 的连接的角度来看，与桥接相关联的更高层实体可以被视为不同的终端站，直接连接到桥接端口所服务的一个或多个 LAN 段，就像任何其他终端站连接到桥接 LAN 一样。实际上，更高层实体在许多情况下将共享桥接中继功能所使用的相同物理连接点，如 7.12 中所述；但是，从这些功能传输和接收帧的角度来看，其行为与它们包含在逻辑上独立的终端站中相同，这些终端站的连接点位于它们所关联的端口“之外”。图 7-9 在功能上等同于图 7-3，但说明了高层实体使用的连接点和 MAC 中继实体使用的连接点之间的逻辑分离。

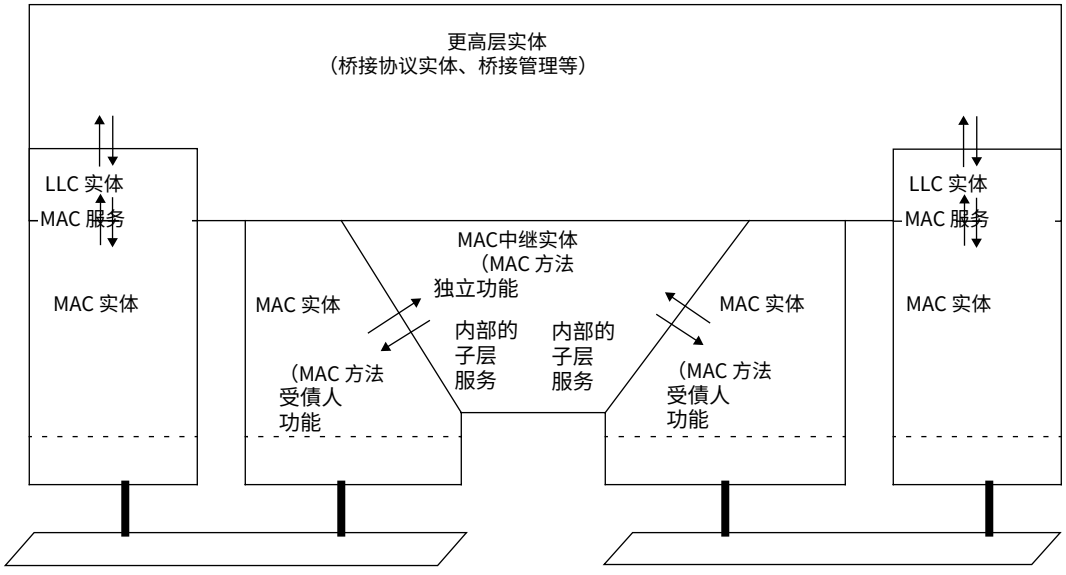


图 7-9—高层实体使用的连接点的逻辑分离和 MAC 中继实体

更高层实体分为两个不同的类别：

a) 那些只需要与桥接 LAN 有一个连接点的实体，例如桥接管理实体；

b) 那些需要每个桥接端口都有一个连接点的实体，例如桥接协议实体和 GARP 参与者。

这两个类别之间的根本区别在于，对于后者，对于相关实体的操作来说，至关重要的是它能够将收到的帧与 Bridge 最初看到这些帧的 LAN 段相关联，并且能够将帧传输到直接连接到该 LAN 段的对等实体。因此，至关重要的是

c) 它不会通过与一个端口关联的连接点接收已由桥接器从其他端口中继的帧；并且

d) 通过一个连接点传输的帧不会被桥接器中继到任何其他端口。

因此，用于到达此类实体的 MAC 地址会永久配置在过滤数据库中，以防止网桥将通过任何端口接收到的此类帧中继到网桥的任何其他端口，如 7.12.3 和 7.12.6 中所定义。

注意——用于寻址此类实体的 MAC 地址通常是组 MAC 地址。

MAC 中继实体桥接器通过桥接器的其他端口转发在一个端口上接收到的帧，但须遵守允许此类转发发生的以下控制信息：

— 与接收帧的端口相关的端口状态信息（7.4）；
— 过滤数据库中保存的信息（7.9）；
— 与可能传输帧的端口相关的端口状态信息（7.4）。

图 7-10 对此进行了说明，其中端口状态和过滤数据库信息所表示的控制信息表示为插入 MAC 中继实体提供的转发路径中的一系列开关（显示为打开、断开状态）。为了使网桥在两个端口之间转发给定帧，所有三个开关都必须处于关闭状态。该图还说明了转发路径中的控制对更高层实体通过与该段的连接点直接在给定 LAN 段上发送和接收帧的能力没有影响（例如，从实体 A 到段 A）；它们仅影响任何间接传输/接收所采用的路径（例如，从实体 A 到段 B）。

图 7-11 说明了针对需要每个端口连接点的更高层实体的帧的转发路径状态。所有网桥中的过滤数据库都永久配置为阻止中继发往这些实体的帧，这意味着它们只能通过其直接连接点（即从段 A 到实体 A，从段 B 到实体 B）接收帧，而不管端口状态如何。

图 7-12 说明了在端口状态和过滤数据库状态允许帧中继的情况下，针对仅需要单个连接点的高层实体的帧的转发路径状态。来自 LAN 段 B 的发往高层实体的帧由网桥中继，并由实体接收并在 LAN 段 A 上传输。

图 7-13 说明了转发路径的状态，对于仅需要单个连接点的高层实体，其中一个端口状态不允许中继。来自 LAN 段 A 的发往高层实体的帧由该实体接收；但是，来自 LAN 段 B 的帧不会被网桥中继，因此可以

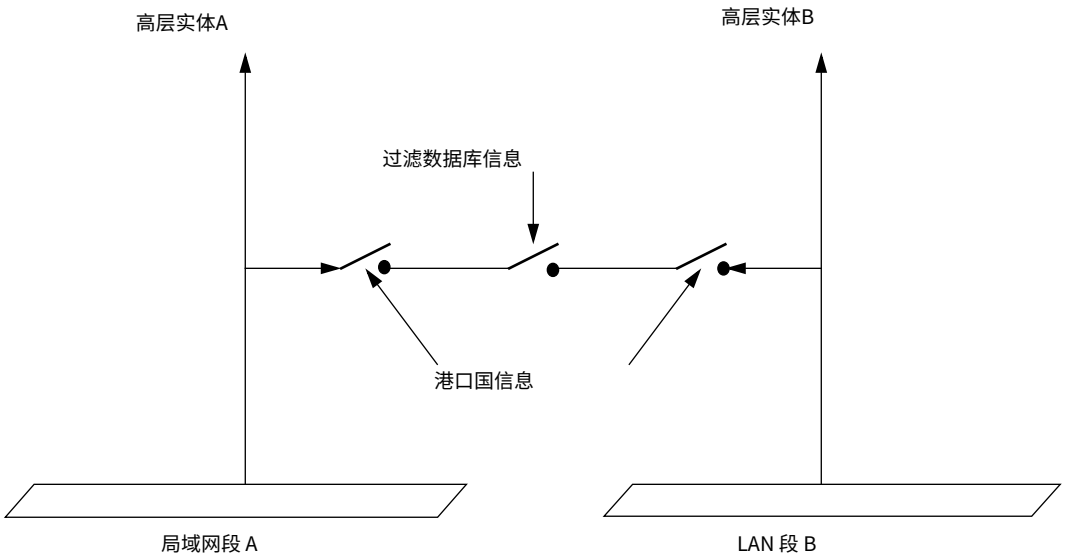


图 7-10—控制信息对转发路径的影响

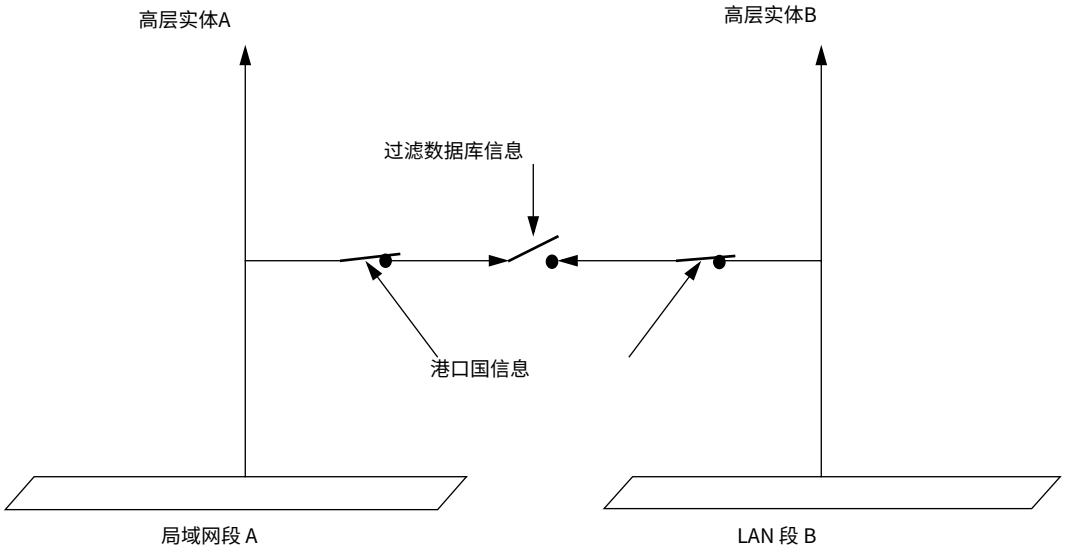


图 7-11—每个端口的连接点

仅当 A 段和 B 段之间的桥接 LAN 的其他组件提供了其他转发路径时，实体才会收到该转发路径。

注 — 如果图 7-13 中所示的端口状态是生成树正常运行的结果（而不是由于设备故障或对端口状态信息的管理控制而导致的），则将存在这样的路径，要么通过此网桥的另一个端口（图中未显示）连接到段 A，要么通过一个或多个在段 A 和段 B 之间提供路径的网桥。如果从段 B 到段 A 没有活动的生成树路径，则桥接 LAN 已划分为两个独立的桥接 LAN，一个位于此端口的两侧，并且所示的更高层实体只能通过段 A 访问。

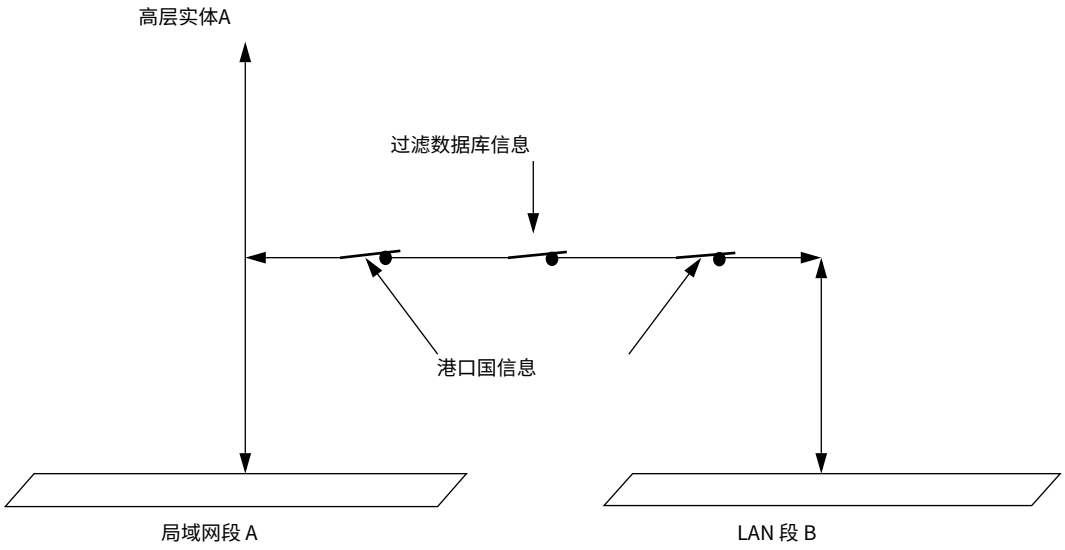


图 7-12 — 单点连接 — 允许中继

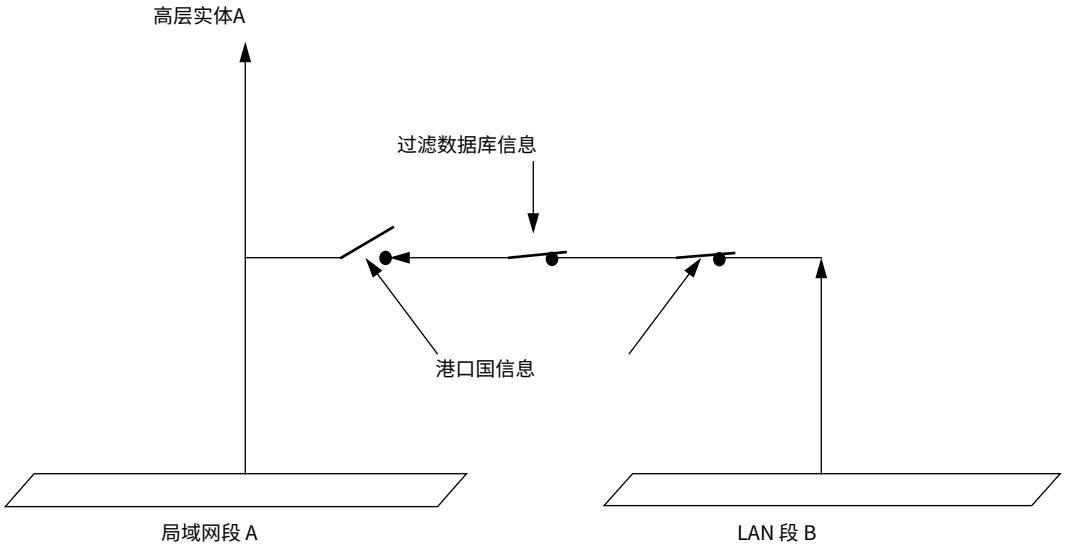


图 7-13 — 单点连接 — 不允许使用中继

8.生成树算法和协议

本节中描述的配置算法和协议将桥接 LAN 拓扑简化为单个生成树。

8.1 算法需要满足的要求

生成树算法及其相关的桥接协议用于支持、保存和维护第 6 条中讨论的 MAC 服务各方面的质量。为了执行此功能，该算法满足以下要求，每个要求都与该条款中的讨论相关：

a) 将任意拓扑的桥接局域网的活动拓扑配置为单生成树，使得任意两个终端站之间最多有一条数据路由，从而消除数据环路（6.3.3；6.3.4）。

b) 当桥接故障或数据路径中断时，它将通过在可用桥接 LAN 组件的范围内自动重新配置生成树拓扑来提供容错能力，并且能够自动适应添加到桥接 LAN 的任何桥接或桥接端口，而不会形成瞬态数据环路（6.1）。

c) 整个活动拓扑将在任何规模的桥接 LAN 中保持稳定。它将以高概率在短的、已知的有界间隔内保持稳定，以尽量减少任何一对终端站之间通信服务不可用的时间 (6.1)。

d) 活动拓扑将是可预测和可重复的，并可通过算法参数管理进行选择，从而允许在流量分析之后应用配置管理，以满足性能管理的目标（6.1；6.3.10）。

e) 它将对终端站透明地运行，使得终端站在使用 MAC 服务 (6.2) 时不知道它们连接到单个 LAN 还是桥接 LAN。

f) 网桥在建立和维护任何特定 LAN 上的生成树时所消耗的通信带宽将只占总可用带宽的一小部分，并且与桥接 LAN 支持的总流量无关，而不管网桥或 LAN 的总数是多少（6.3.10）。

此外，该算法和协议满足以下目标，从而限制了网桥及其配置的复杂性：

g) 与每个桥接端口相关的内存要求与桥接 LAN 中的桥接器和 LAN 的数量无关。

h) 除了通过正常程序分配 MAC 地址外，网桥在添加到桥接 LAN 之前不必进行单独配置。

8.2 MAC 桥接器的要求

为了使桥接协议正常运行，需要满足以下条件：

a) 一个唯一的 MAC 组地址，由桥接 LAN 内的所有桥接器识别，用于标识连接到单个 LAN 的所有桥接器的桥接协议实体。

b) 每个桥接器的标识符，在桥接 LAN 内是唯一的。

c) 每个桥接端口有一个不同的端口标识符，该标识符可以独立于其他桥接器中使用的值进行分配。

每个桥应提供这些参数的值或为它们分配值的机制。对于使用 48 位通用管理地址的 MAC 桥，标识桥协议实体的唯一 MAC 地址是桥组地址 (7.12.3)。

此外，为了管理生成树活动拓扑的配置，需要满足以下要求：

1) 为桥接 LAN 中的一组桥接器分配每个桥接器的相对优先级的一种方法。

2) 在单个桥接器的端口集内分配每个端口的相对优先级的方法。
3) 为每个端口分配路径成本组件的方法。

当支持桥接管理时，这些参数可以由管理层设置。

每个网桥的唯一标识符部分来自网桥地址 (7.12.5)，部分来自可管理优先级组件 (9.2.5)。网桥的相对优先级由唯一标识符的数值比较确定，数值越低，标识符的优先级越高。

每个端口的标识符的一部分是固定的，并且对于桥接器上的每个端口都是不同的，另一部分是可管理的优先级组件（9.2.7）。端口的相对优先级由唯一标识符的数值比较确定，数值越低，标识符的优先级越高。

与每个端口相关的路径成本可能是可管理的。此外，8.10.2 建议为连接到特定 MAC 类型和速度的 LAN 的端口提供默认值。

8.3 概述

8.3.1 主动拓扑及其计算

生成树算法和协议从桥接 LAN 的任意连接组件配置简单连接的主动拓扑。帧通过桥接 LAN 中的某些桥接端口转发，而不通过处于阻塞状态的其他端口转发。在任何时候，桥接器都只有效地连接处于转发状态的端口所连接的 LAN。帧通过处于转发状态的桥接端口在两个方向上转发。处于阻塞状态的端口不会在任何一个方向上转发帧，但可能包含在主动拓扑中，即，如果组件发生故障、被移除或被添加，则将进入转发状态。

图 7-1 显示了桥接 LAN 的示例。图 8-1 显示了按照以下配置的相同桥接 LAN 的活动拓扑，即逻辑连接。

其中一个桥接器称为桥接 LAN 中的根或根桥接器。每个单独的 LAN 都连接有一个桥接端口，用于将帧从该 LAN 转发到根，并将帧从根方向转发到该 LAN。此端口称为该 LAN 的指定端口，它所属的桥接器是该 LAN 的指定桥接器。根是它所连接的所有 LAN 的指定桥接器。每个桥接器上处于转发状态的端口是根端口（最靠近根的端口 - 见下文）和指定端口（如果有）。未禁用且既不是根端口也不是指定端口的端口不会将帧转发到它们所连接的 LAN；此类端口称为备用端口。

在图 8-1 中，网桥 1 已被选为根（尽管仅通过查看拓扑无法判断哪个网桥是根），并且是 LAN A 和 LAN B 的指定网桥。网桥 2 是 LAN C 和 LAN D 的指定网桥，网桥 4 是 LAN E 的指定网桥。图 8-2 显示了此桥接 LAN 配置的逻辑树形拓扑。

桥接 LAN 的稳定活动拓扑由以下因素决定：

a) 与每个桥梁相关的唯一桥梁标识符。

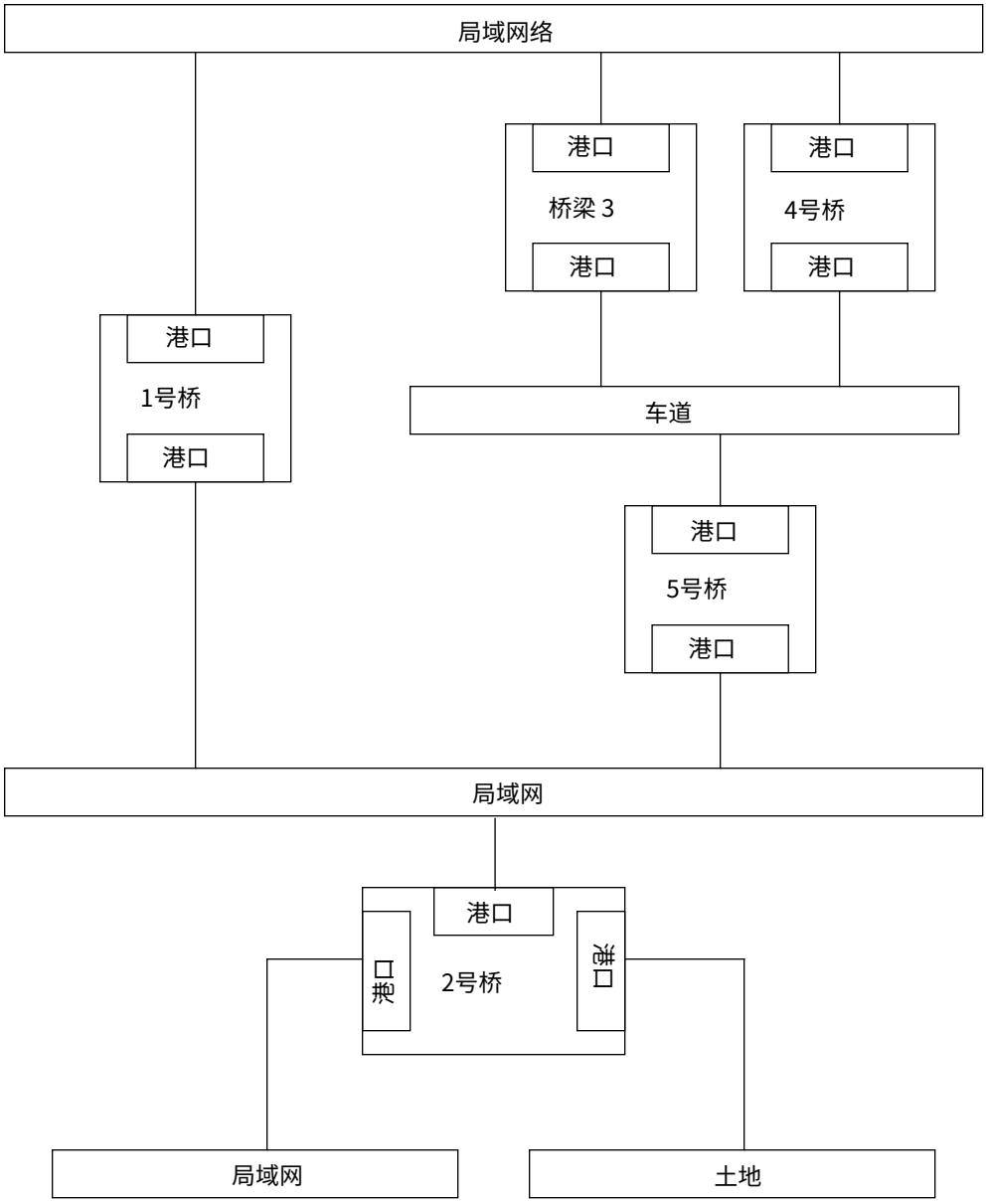


图 8-1 — 主动拓扑

b) 与每个桥接端口相关的路径成本。

c) 与每个桥接端口关联的端口标识符。

具有最高优先级桥接标识符的桥接器是根（为便于计算，此标识符为数值最低的标识符）。桥接 LAN 中的每个桥接端口都具有与之关联的根路径成本。这是每个桥接端口接收从根转发到桥接器的最低成本路径上的帧的路径成本之和。每个 LAN 的指定端口是根路径成本值最低的桥接端口：如果两个或多个端口具有相同的根路径成本值，则首先使用其桥接器的桥接标识符，然后使用其端口标识符作为决胜局。因此，为每个 LAN 选择一个桥接端口作为指定端口，通过相同的计算从桥接器自己的端口中选择桥接器的根端口，并完全确定桥接 LAN 的活动拓扑。

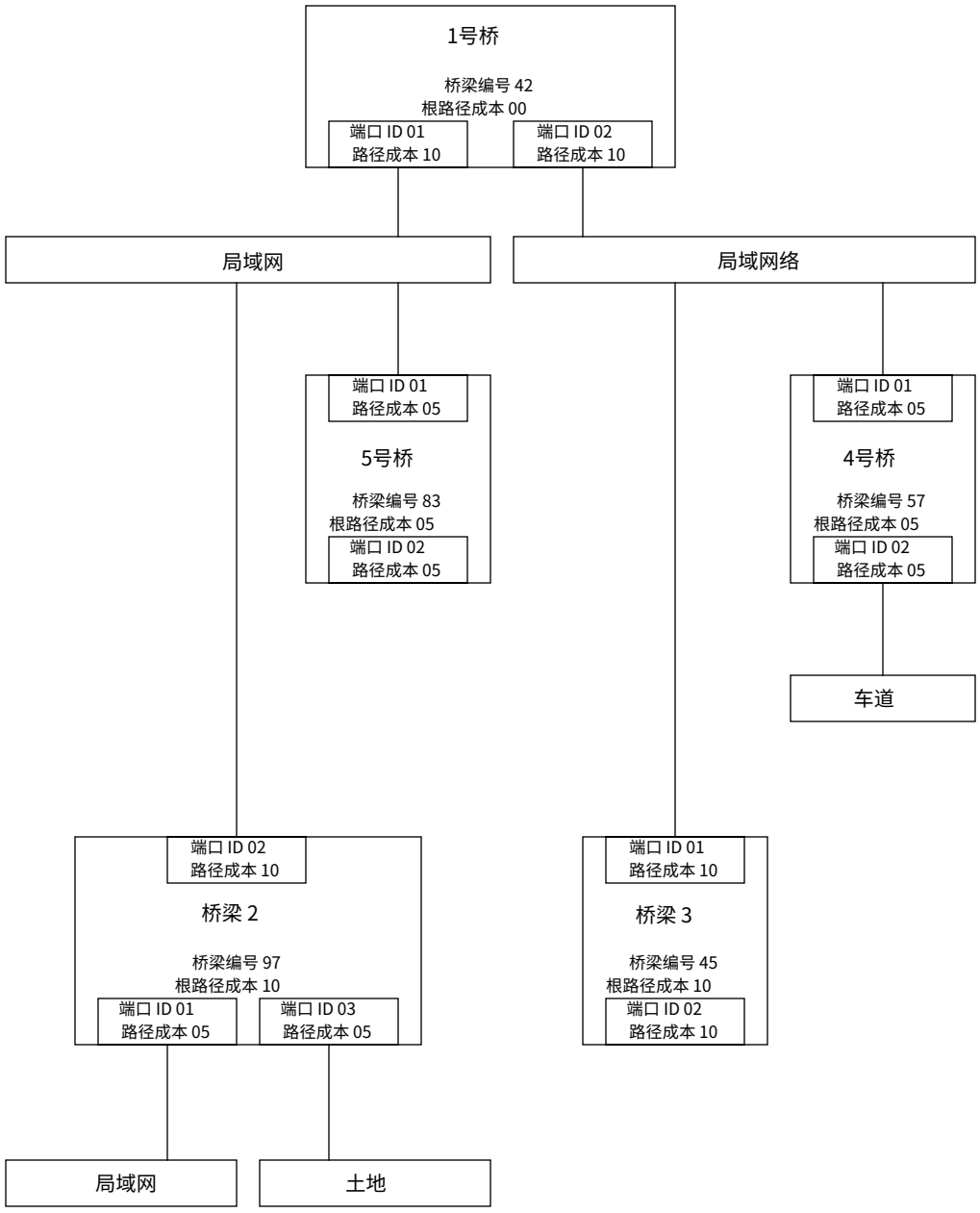


图 8-2 — 生成树

可以管理每个桥接器的桥接标识符的组件以及每个桥接器端口的路径成本和端口标识符，从而允许管理员选择桥接 LAN 的活动拓扑。

8.3.2 传播拓扑信息

网桥之间会相互发送一种称为配置 BPDU 的网桥协议数据单元，以便通信和计算上述信息。传送 BPDU 的 MAC 帧在目标地址字段中携带网桥组地址，并由连接到传输该帧的 LAN 的所有网桥接收。

桥接协议数据单元不直接由桥接器转发，但桥接器可以使用其中的信息来计算其自己的 BPDU 以进行传输，并可能刺激该传输。配置 BPDU 在连接到单个 LAN 的桥接端口之间传送，与配置消息的概念不同，配置消息表示在整个桥接 LAN 中承载的信息的传播。

每个配置 BPDU 包含传输网桥认为是根的网桥的唯一标识符、从传输端口到根的路径成本、传输网桥的标识符以及传输端口的标识符等参数。此信息足以让接收网桥确定传输端口是否比当前被认为是指定端口的端口更有资格成为接收配置 BPDU 的 LAN 上的指定端口，并确定接收端口是否应成为网桥的根端口（如果尚未成为根端口）。

通过三种基本机制，可以在整个桥接 LAN 中及时传播必要的信息，以允许所有桥接端口确定其状态（阻塞或转发）：

a) 认为自己是根的网桥（所有网桥都首先认为自己是根，直到发现并非如此）会定期在其所连接的所有 LAN 上发起配置消息（通过传输配置 BPDU）。

b) 收到配置 BPDU 的网桥将决定哪个是其根端口，并传达更好的信息（即，最高优先级根标识符、最低根路径成本、最高优先级传输网桥和端口），并将该信息传递给它认为自己是指定网桥的所有 LAN。

c) 接收到下级信息的网桥在其认为是其所连接的 LAN 上的指定端口的端口上，传输自己的信息作为答复，以便连接到该 LAN 的所有其他网桥都能听到。

因此，在整个桥接 LAN 中，可以快速获知到具有最高优先级根标识符的桥接器的生成树路径，而有关其他潜在根和路径的次要信息则受到矛盾。

8.3.3 重新配置

为了在移除组件或对确定拓扑的参数进行管理更改时重新配置桥接 LAN，在整个桥接 LAN 中传播的拓扑信息具有有限的生命周期。这是通过在每个配置 BPDU 中传输所传达信息的年龄（自配置消息源自根以来经过的时间）来实现的。每个网桥都会存储其端口所连接到的每个 LAN 上的指定端口的信息，并监控该信息的年龄。

在正常稳定的运行中，Root定期传输配置消息，保证拓扑信息不会超时。

如果网桥超时保存端口的信息，它将尝试成为该端口所连接的 LAN 的指定网桥，并将从其根端口上的根接收的协议信息传输到该 LAN。

如果网桥的根端口超时，则可选择另一个端口作为根端口。然后，将根据新根端口上收到的信息来计算在网桥作为指定网桥的 LAN 上传输的信息。

如果当前根节点的信息记录已不存在，则桥接器将通过声称自己是根节点来重新配置。如果根节点确实发生故障，其他桥接器也将超时协议信息；有关最佳继任者和新拓扑的信息将在整个桥接器中快速传播

LAN。还有一种可能是，到当前根的路径已经改变，可能是成本增加，而重新配置网桥已超时，因为它认为来自根的较新信息较低，因为它具有较高的根路径成本。在后一种情况下，相邻网桥将立即回复由目标根发送的 BPDU。

为了确保桥接 LAN 中的所有网桥都对旧信息何时超时达成共识，超时值会在来自根的所有配置消息中传输。此值考虑了桥接 LAN 中每个 LAN 上传输和接收 BPDU 的传播延迟，从而考虑了协议信息沿生成树传播的传播延迟。为了最大限度地降低因配置消息丢失而触发重新配置的可能性，它包括根传输这些消息的时间间隔的额外倍数。

8.3.4 改变端口状态

由于在桥接 LAN 中传递协议信息存在传播延迟，因此无法从一个活动拓扑急剧过渡到另一个活动拓扑。拓扑变化可能在不同时间发生在桥接 LAN 的不同部分，而将桥接端口直接从活动拓扑中的非参与状态移动到转发状态可能会产生临时数据循环以及帧重复和乱序的风险。在开始转发帧之前，最好让其他桥接有时间回复劣质协议信息。

因此，桥接端口必须等待新的拓扑信息在整个桥接 LAN 中传播，并且等待使用旧的活动拓扑转发的任何帧的帧生命周期到期，然后才能转发帧。

在此期间，还希望使过滤数据库中可能不再正确的站点位置信息超时，并在此间隔的后半部分学习新的站点位置信息，以尽量减少端口进入转发状态时帧的初始泛洪的影响。因此，当算法决定应将端口置于转发状态时，它首先被置于侦听状态，等待建议它返回阻塞状态的协议信息，并等待协议计时器到期以将其移至学习状态。在学习状态下，它仍然阻止帧的转发，但学习到的站点位置信息由学习过程包含在过滤数据库中。最后，协议计时器到期将其移至转发状态，其中转发中继帧和学习站点位置信息均已启用。

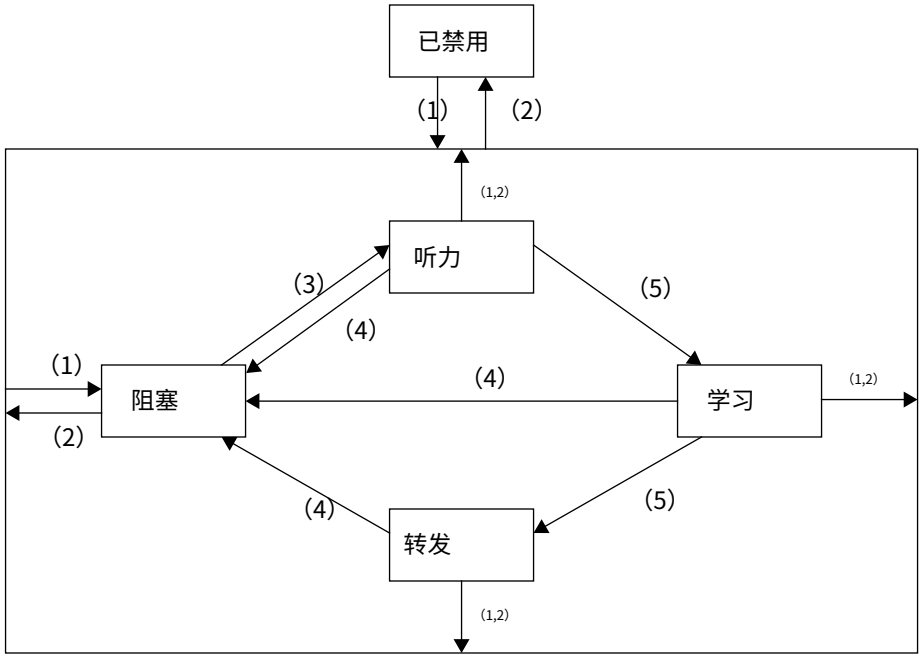
图 8-3 显示了端口状态之间的转换。

8.3.5 通知拓扑变化

在正常稳定运行中，过滤数据库中的站点位置信息仅因站点的物理重新定位而发生变化。因此，可能希望对过滤数据库中的条目采用较长的老化时间，尤其是当许多终端站在重新定位后通电后传输帧时，这将导致重新学习站点位置信息。

然而，当桥接 LAN 的活动拓扑重新配置时，从网络中的桥接器的角度来看，终端站可能会出现移动。即使该桥接器上的端口状态没有改变，情况也是如此。即使只有部分桥接 LAN 重新配置，也有必要在活动拓扑发生变化后重新学习站点位置。

生成树算法和协议为检测到活动拓扑变化的网桥提供了可靠地向根通知该变化的过程，并为根随后将该变化传达给所有网桥提供了过程。然后，网桥使用一个短值在一段时间内使过滤数据库中的动态条目过期。



1）通过管理或初始化启用端口
2）端口因管理或故障而被禁用
3）算法选择指定端口或根端口
4）算法选择作为备用端口
5）协议定时器到期（转发定时器）

图 8-3 — 端口状态

当非根桥更改桥接 LAN 的活动拓扑时，它会在其根端口所连接的 LAN 上传输拓扑更改通知 BPDU。此传输会重复进行，直到桥收到该 LAN 的指定桥的确认。确认包含在配置 BPDU 中，因此通知最终将得到确认或将进行进一步的重新配置。指定桥使用相同的程序将通知传递给根或向根传递。

如果根节点收到此类通知，或者更改拓扑结构本身，它将在一段时间内在所有传输的配置消息中设置拓扑更改标志。这段时间是这样的，所有网桥都将收到一个或多个配置消息，或者将进行进一步的重新配置。设置此标志后，网桥将使用转发延迟值（在每个侦听和学习状态下所花费的时间间隔）来使动态条目老化。当再次重置标志时，网桥将恢复使用过滤数据库老化时间。

可以通过启用变更检测参数 (8.5.5.10) 在每个端口上启用或禁用检测拓扑变化的功能。此功能旨在允许在已知连接单个终端站的端口上禁用拓扑变化检测，并且打开和关闭该终端站的电源会导致触发拓扑变化通知机制。是否支持将此参数设置为禁用状态的功能是可选的。

8.4 港口国家

单个桥接端口的操作是根据端口的状态和提供并支持桥接端口操作所需功能（7.1）的流程（7.3）来描述的。

每个端口的状态控制从与该端口关联的各个 MAC 实体接收的帧的处理 (7.5)、将帧提交给 MAC 实体进行传输 (7.6) 以及将端口纳入桥接 LAN 的活动拓扑的可能性。

生成树算法和协议的操作用于维护和更改每个端口的状态，以满足算法的要求（8.1）。可能的端口状态和与帧处理相关的规则特定于此算法和桥接协议。

下面针对端口可能处于的五种状态（阻塞、侦听、学习、转发或禁用）分别指定以下内容：

a) 国家的目的。
b) 转发过程（7.7）是否丢弃接收到的帧。
c) 转发过程（7.7）是否提交转发的帧进行传输。
d）学习过程（7.8）如何处理接收到的帧。
e) 桥接协议实体（7.10）是否将端口纳入其活动拓扑的计算中。

f) 港口在什么条件下进入和离开该州。

8.4.1 阻塞

处于此状态的端口不参与帧中继，从而可以防止通过桥接 LAN 的活动拓扑中存在的多条路径产生帧重复。

转发进程应丢弃收到的帧。它不应提交转发的帧进行传输。学习进程不应将站点位置信息添加到过滤数据库。

桥接协议实体应将端口纳入其活动拓扑的计算中。收到的 BPDU 应根据生成树算法和协议的要求进行处理。

此状态是在桥接器初始化后或通过管理操作启用端口时从禁用状态进入的。此状态可以通过生成树算法和协议的操作从侦听、学习或转发状态进入。端口进入阻塞状态是因为它已收到信息，即另一个桥接器是该端口所连接的 LAN 的指定桥接器。

协议计时器到期或在此端口或另一个端口上收到配置消息后，可以通过生成树算法和协议的操作退出此状态并进入侦听状态。可以通过管理操作退出此状态并进入禁用状态。

8.4.2 聆听

处于此状态的端口正在准备参与帧中继。帧中继暂时被禁用，以防止临时环路，因为桥接 LAN 的活动拓扑在此状态的生命周期内可能会发生变化。学习被禁用，因为当活动拓扑稳定时，活动拓扑的变化可能导致获取的信息不正确。

转发进程应丢弃收到的帧。它不应提交转发的帧进行传输。学习进程不应将站点位置信息添加到过滤数据库。

桥接协议实体应将端口纳入其活动拓扑的计算中。收到的 BPDU 应按照生成树算法和协议的要求进行处理。可以提交 BPDU 进行传输。

当生成树算法和协议的运行确定端口应该参与帧中继时，从阻塞状态进入此状态。

协议计时器到期后，可以通过生成树算法和协议的操作退出此状态，并进入学习状态。在此端口或另一个端口上收到桥接协议数据单元后，可以通过生成树算法和协议的操作退出此状态，并进入阻塞状态。通过管理操作，可以退出此状态，并进入禁用或阻塞状态。

8.4.3 学习

处于此状态的端口正在准备参与帧中继。帧中继暂时被禁用，以防止临时环路，这种环路可能在此状态的生命周期内随着桥接 LAN 的活动拓扑发生变化而出现在桥接 LAN 中。启用学习功能可在帧中继之前获取信息，从而减少不必要中继的帧数。

转发进程应丢弃收到的帧。它不应提交转发的帧进行传输。学习进程应将站点位置信息纳入过滤数据库。

桥接协议实体应将端口纳入其活动拓扑的计算中。收到的 BPDU 应按照生成树算法和协议的要求进行处理。可以提交 BPDU 进行传输。

当协议计时器到期时，通过生成树算法和协议的操作从监听状态进入此状态。

协议计时器到期后，可以通过生成树算法和协议的操作进入转发状态，从而退出此状态。在此端口或另一个端口上收到桥接协议数据单元后，可以通过生成树算法和协议的操作进入阻塞状态，从而退出此状态。可以通过管理操作退出此状态，并进入禁用或阻塞状态。

8.4.4 转发

处于此状态的端口正在参与帧中继。

转发进程可以转发接收到的帧。它可以提交转发的帧以供传输。学习进程应将站点位置信息纳入过滤数据库。

桥接协议实体应将端口纳入其活动拓扑的计算中。收到的 BPDU 应按照生成树算法和协议的要求进行处理。可以提交 BPDU 进行传输。

当协议计时器到期时，通过生成树算法和协议的操作从学习状态进入此状态。

通过生成树算法和协议的操作，在此端口或另一个端口上收到桥接协议数据单元后，可以离开此状态，并进入阻塞状态。通过管理操作，可以离开此状态，并进入禁用或阻塞状态。

8.4.5 已禁用

处于此状态的端口不参与帧中继或生成树算法和协议的操作。

转发进程应丢弃收到的帧。它不应提交转发的帧进行传输。学习进程不应将站点位置信息纳入过滤数据库。

桥接协议实体不应将端口纳入其活动拓扑计算中。 所接收的 BPDU 不应由生成树算法和协议处理。 不应提交 BPDU 进行传输。

通过管理操作可以从任何其他状态进入此状态。

当端口通过管理操作启用时，将离开此状态，并进入阻塞状态。

8.5 协议参数和计时器

通过交换桥接协议数据单元，信息在各个桥接器的协议实体之间传输。本节规定了在指定的两种类型的 BPDU 中传送的参数：配置 BPDU 和拓扑更改通知 BPDU。这些参数和其他信息元素的编码在第 9 节中规定。

每个桥接协议实体都维护着一些独立于各个端口的参数和计时器，以及每个端口的计时器和参数。本节规定了这些参数、它们的用途以及在什么条件下更新它们。

8.5.1 配置BPDU参数

8.5.1.1 根标识符

传输配置 BPDU 的网桥假定为根网桥的唯一网桥标识符。

传达此参数是为了让所有桥接器就根达成一致。

8.5.1.2 根路径成本

从传输桥到根桥的路径成本，由根标识符表示。

传达此参数是为了使网桥能够决定连接到已接收配置 BPDU 的 LAN 的哪个网桥为该 LAN 提供了到根的最低成本路径。

8.5.1.3 桥接标识符

传输配置 BPDU 的桥的唯一桥标识符。

传递此参数是为了使桥能够

a) 如果一个 LAN 连接着两个或多个网桥，并且这些网桥提供到根的等价路径，则决定应选择哪个网桥作为该 LAN 的指定网桥。

b) 检测同一桥上两个或多个端口连接到同一 LAN 的情况，即通过桥接 LAN 组件的路径进行直接通信，但这些组件均不运行生成树算法和协议。

8.5.1.4 端口标识符

传输桥上配置 BPDU 所通过的端口的端口标识符。此标识符唯一地标识该桥上的端口。

传达此参数是为了使网桥能够决定，在同一个网桥上的两个或多个端口连接到一个 LAN 的情况下，哪些端口是如此连接的。

8.5.1.5 消息年龄

配置消息的年龄，即自根生成配置 BPDU 以来的时间，该配置 BPDU 促使该配置 BPDU 的生成。

传达此参数是为了使桥接器能够丢弃年龄超过最大年龄的信息（见下文）。

8.5.1.6 最大年龄

桥接 LAN 中的所有桥接器使用的超时值。Max Age 的值由 Root 设置。

传达此参数是为了确保桥接 LAN 中的每个桥接器都具有一致的值，以便测试存储的配置信息的使用期限。

8.5.1.7 你好时间

Root 生成配置 BPDU 的时间间隔。

此参数不直接由生成树算法使用，而是在配置 BPDU 中传送，以便于管理功能监控协议性能。

8.5.1.8 转发延迟

桥接 LAN 中的所有桥接器使用的超时值。转发延迟的值由根设置。

传递此参数是为了确保桥接 LAN 中的每个桥接器在将端口状态转换为转发状态时使用一致的转发延迟计时器值。此参数还用作活动拓扑发生变化后过滤数据库动态条目老化的超时值。

8.5.1.9 拓扑改变确认

为响应指定端口上收到的拓扑更改通知而发送的配置消息中设置的标志。传送此参数是为了允许可靠的确认协议运行，以通知根活动拓扑的更改。

8.5.1.10 拓扑变化

在通知或检测到拓扑变化之后的一段时间内传输的所有配置 BPDU 中由根设置的标志。

传达此参数是为了通知整个桥接 LAN 中的网桥，桥接 LAN 部分的活动拓扑已发生变化，并且过滤数据库应更快地淘汰条目，以限制由于使用过滤数据库中的错误信息而导致的连接到桥接 LAN 的终端系统的临时隔离的影响。

应用于过滤数据库中动态条目的老化时间值等于为网桥保存的转发延迟时间参数的值；即，在从根接收的所有配置消息中设置拓扑更改标志后，转发延迟时间过去之后，过滤数据库中剩余的动态条目就是在此期间创建或更新的条目。

8.5.2 拓扑变化通知 BPDU 参数

拓扑改变通知 BPDU 中不传达任何参数。

8.5.3 桥梁参数

8.5.3.1 指定根

被视为根的桥的唯一桥标识符。

此参数用作网桥传输的所有配置 BPDU 中的根标识符参数的值。

8.5.3.2 根路径成本

从此桥到根的路径成本。当桥为根时，此参数的值为零。否则，它等于根端口的指定成本和路径成本参数值的总和。此参数用于测试在接收的配置消息信息中传达的根路径成本参数的值，以及作为传输的配置消息信息中的根路径成本参数的值。

8.5.3.3 根端口

提供到根的最低成本路径的端口的端口标识符，即，该端口的指定成本和路径成本参数的值的总和最低的端口。

如果两个或多个端口提供到根的相等的最低成本路径，则选择根端口作为具有最高优先级桥标识符的端口作为该端口的指定桥参数。

如果两个或多个端口提供到根的相等的最低成本路径，并持有相同的指定桥参数值，则选择根端口为该端口持有的最高优先级指定端口。

最后，如果两个或多个端口提供相同的最低成本路径到根，并具有相同的指定桥和指定端口参数值，则根端口将被选为具有最高优先级端口标识符的端口。同一桥上不同端口的端口标识符保证不同，从而强制执行决胜局。

此参数用于标识建立到根的路径的端口。当桥为根时，此参数不重要，设置为零。

8.5.3.4 最大年龄

收到的协议信息在被丢弃之前的最大生存期。

8.5.3.5 问候时间

尝试成为根或已经是根的网桥传输配置 BPDU 之间的时间间隔。

8.5.3.6 转发延迟

端口在进入学习或转发状态之前分别处于侦听状态和学习状态所花费的时间。它也是过滤数据库中动态条目的老化时间，而收到的配置消息则表示拓扑发生变化。

8.5.3.7 桥接标识符

桥的唯一桥标识符。此参数用作

a) 网桥传输的所有配置 BPDU 中的网桥标识符参数。
b) 当网桥为根或试图成为根时、在有关当前根的所有信息过期之后、或在管理操作之后，网桥的指定根。

此参数由两部分组成，一部分来自唯一的桥接地址 (7.12.5)，确保桥接 LAN 中桥接标识符的唯一性，另一部分允许调整桥接标识符的优先级，并在优先级比较中被视为更重要的部分。当支持桥接管理时，此参数的优先级部分可以通过管理操作进行更新。

8.5.3.8 桥最大年龄

当网桥为根或试图成为根时，Max Age 参数的值。

当支持桥接管理时，此参数可以通过管理操作更新。

8.5.3.9 桥接问候时间

当网桥为根或试图成为根时，Hello Time 参数的值。

此参数是当网桥尝试通知其根端口所连接的 LAN 上的指定网桥拓扑变化时，向根传输拓扑变化通知 BPDU 的间隔。

当支持桥接管理时，此参数可以通过管理操作更新。

8.5.3.10 桥接转发延迟

当网桥为根或尝试成为根时，转发延迟参数的值。

当支持桥接管理时，此参数可以通过管理操作更新。

8.5.3.11 检测到拓扑变化

布尔参数设置为 True 以记录拓扑更改已被 Bridge 检测到或已通知给 Bridge。设置为 True 时，此参数用于在 Bridge 本身不是 Root 时刺激向 Root 传输拓扑更改通知；如果 Bridge 是 Root 或成为 Root，则将 Bridge 的拓扑更改参数值设置为 True。传输受可靠

确认机制，如 8.3.5 和 8.5.1.9 所述；拓扑变化通知 BPDU 按照桥接问候时间的固定间隔进行传输，直到得到确认。

8.5.3.12 拓扑变化

设置为记录的布尔参数

a) 对于非根桥，最近接受的配置消息是否指示活动拓扑的变化；或者

b) 对于根，在前一个拓扑变化时间段内是否检测到拓扑变化。

此参数用于传播传输的配置消息中的拓扑变化指示，并确定是否对过滤数据库中的动态条目使用短（转发延迟）或长（老化时间）超时值。

8.5.3.13 拓扑变化时间

当网桥为根时，网桥在检测到拓扑变化后发出指示拓扑变化的配置消息的时间段。此参数的值等于网桥的网桥最大年龄和网桥转发延迟参数值的总和。

8.5.3.14 保持时间

通过给定 LAN 端口传输配置 BPDU 的最小时间间隔：在任何保持时间间隔内最多只能传输一个配置 BPDU。此参数是固定参数，其值如表 8-3 中所示。

8.5.4 桥接定时器

8.5.4.1 你好定时器

此计时器用于确保网桥在成为根或试图成为根时定期传输配置 BPDU。

计时器的超时值为 Bridge 的 Bridge Hello Time 参数的超时值。

8.5.4.2 拓扑变化通知定时器

该计时器的作用是确保将任何检测到的拓扑变化通知给网桥根端口所连接的 LAN 上的指定网桥。

计时器的超时值为 Bridge 的 Bridge Hello Time 参数的超时值。

8.5.4.3 拓扑变化定时器

此计时器用于确定在检测到拓扑变化后，当网桥作为根时，传输带有由网桥设置的拓扑变化标志的配置 BPDU 的时间段。

计时器的超时值为网桥的拓扑改变时间参数的超时值。

8.5.5 端口参数

8.5.5.1 端口标识符

端口标识符，在该桥的端口中是唯一的。此参数用作通过端口传输的所有配置消息的端口标识符参数的值。

该参数由两部分组成。一部分与实际设备对端口的物理或逻辑支持具有固定关系；这部分确保端口标识符在单个桥接器的各个端口之间是唯一的，并且是分配在从 1 向上的范围内的小整数。该参数的另一部分允许调整端口的优先级，并在优先级比较中被视为更重要的部分。当支持桥接器管理时，此参数的优先级部分可以由管理更新。

8.5.5.2 状态

端口的当前状态（即禁用、监听、学习、转发或阻塞）。

此参数用于控制转发和学习过程对来自与端口关联的 MAC 实体的帧的接受、转发过程对帧的转发以及 BPDU 的发送和接收 (8.4)。

该参数由协议的动作更新。

当支持桥接管理时，该参数也可以通过管理操作来更新。

8.5.5.3 路径成本

当该端口为根端口时，通过此端口的路径对此桥到根的路径总成本的贡献。

当网桥不是根时，此参数被用作网桥传输的所有配置 BPDU 中提供的根路径成本参数的值，并将其添加到根端口的指定成本参数的值中。

当支持桥接管理时，该参数可以通过管理操作来更新。

8.5.5.4 指定根

在指定桥为端口所连接的 LAN 传输的配置 BPDU 的根标识符参数中，记录为根的桥的唯一桥标识符。

此参数用于测试接收到的配置 BPDU 中传达的根标识符参数的值。

8.5.5.5 指定成本

a) 对于指定端口，为该端口所连接的 LAN 提供的路径成本（等于桥的根路径成本）；否则，

b) 此端口所连接的 LAN 上的指定端口所提供的到根路径的成本。

此参数用于测试接收到的配置 BPDU 中传达的根路径成本参数的值。

8.5.5.6 指定桥

唯一的桥梁标识符（8.5.3.7）

a) 如果是指定港口，则为该港口所属的桥梁；否则，
b) 该桥接器被认为是此端口所连接的 LAN 的指定桥接器。

此参数用于

c) 连同端口的指定端口和端口标识符参数来确定此端口是否应该是其所连接的 LAN 的指定端口。

d) 测试收到的配置 BPDU 中传送的桥标识符参数的值。

8.5.5.7 指定端口

指定桥（8.5.5.6）上的桥端口的端口标识符（8.5.5.1），指定桥通过该端口传输此端口存储的配置消息信息。

此参数用于

a) 连同端口的指定桥和端口标识符参数来确定此端口是否应该是其所连接的 LAN 的指定端口。

b) 通过管理来确定桥接局域网的拓扑结构。

8.5.5.8 拓扑改变确认

在关联端口上要传输的下一个配置 BPDU 中的拓扑改变确认标志的值。

此参数用于记录是否需要设置拓扑改变确认标志来回复收到的拓扑改变通知 BPDU。

8.5.5.9 配置待处理

布尔参数集用于记录应在相关端口的保持计时器到期时传输配置 BPDU。

此参数与端口的保持计时器一起使用，以确保配置 BPDU 不会传输过于频繁，但会传输最新的信息。

8.5.5.10 启用变更检测

布尔参数集，用于记录是否为相关端口启用拓扑变化检测。如果禁用变化检测，则从将拓扑变化传达给根的角度来看，由此端口检测到的或通过此端口通知给桥的拓扑变化将被忽略。

此参数的默认状态应为桥接器所有端口的启用状态。是否支持将此参数设置为禁用状态是可选的。

8.5.6 端口计时器

8.5.6.1 消息年龄计时器

该计时器用于测量端口记录的接收协议信息的年龄，并确保当其年龄超过网桥记录的最大年龄参数值时，丢弃该信息。

计时器的超时值为 Bridge 的 Max Age 参数的超时值。

8.5.6.2 转发延迟定时器

此计时器确定端口处于侦听和学习状态的时间。计时器的超时值是网桥的转发延迟参数。

8.5.6.3 保持定时器

该计时器用于确保配置 BPDU 不会通过任何桥接端口过于频繁地传输。

计时器的超时值是桥的保持时间的超时值。

8.6 程序要素

8.6.1 发送配置 BPDU

8.6.1.1 目的

将指定根、根路径成本、指定桥、指定端口以及协议计时器的值的知识传达给连接到与传输配置 BPDU 的端口相同的 LAN 的其他桥端口。

8.6.1.2 使用

a) 情况 1. 作为配置 BPDU 生成过程的一部分（8.6.4）。
b) 情况 2. 作为回复配置 BPDU 程序的一部分 (8.6.5)。
c) 情况 3. 由于先前调用了该程序，当端口的配置待处理标志参数被设置时，端口的保持计时器（8.7.8）到期之后。
d）情况4.作为确认拓扑改变程序的一部分（8.6.16）。

8.6.1.3 程序

8.6.1.3.1 步骤 1

如果端口的保持定时器处于活动状态，则应设置端口的配置待定标志参数。这样就完成了该过程。否则，如果保持定时器不处于活动状态，则调用以下步骤。

8.6.1.3.2 第 2 步

如果端口的保持定时器未处于活动状态，则准备传输配置 BPDU。配置 BPDU 应具有以下参数设置：

a) 根标识符应设置为桥接器持有的指定根参数的值。
b) 根路径成本应设置为桥接器持有的根路径成本参数的值。
c) 桥接标识符应设置为桥接器持有的桥接标识符参数的值。
d) 端口标识符应设置为传输配置 BPDU 的桥接端口的端口标识符参数的值。

e) 如果该网桥已被选为根，即，如果该网桥持有的指定根和网桥标识符参数的值相同，则消息年龄应设置为零。
f) 否则，应设置消息年龄的值，以使传输的配置 BPDU 不会低估在根端口上收到的协议消息的年龄；即，传输的值不得小于该端口的消息年龄计时器记录的值，且应大于收到的值，并将包含任何传输延迟。该参数的值不得超过其真实值，且不得超过 8.10.2 中规定的最大消息年龄增量高估值。

g) 最大年龄、Hello Time 和转发延迟应设置为网桥保存的最大年龄、Hello Time 和转发延迟参数的值。
h) 拓扑改变确认标志应设置为端口的拓扑改变确认标志参数的值。

i) 拓扑变化标志应设置为桥接器的拓扑变化标志参数的值。

8.6.1.3.3 步骤 3

如果配置 BPDU 中的消息年龄参数的值小于最大年龄参数的值，则

a) 端口的拓扑改变确认标志参数被重置。
b) 端口的配置待处理标志参数被重置。
c) BPDU 应在最大 BPDU 传输延迟时间内通过端口传输（如 8.10.2 中所述）。

d) 端口的保持计时器已启动。

8.6.2 记录配置信息

8.6.2.1 目的

对于某个端口，记录该端口接收到的配置 BPDU 传达的协议参数。

8.6.2.2 使用

在收到配置 BPDU 后，该配置 BPDU 传达了取代已持有的协议信息，即，如果

a) 情况 1. 根标识符表示的桥的优先级高于记录为指定根的桥，或者

b) 情况 2. 根标识符与指定根相同，并且根路径成本低于记录为端口的指定成本，或者

c) 情况 3. 根标识符和根路径成本与端口记录的一致，并且桥标识符表示的桥的优先级高于记录为端口的指定桥，或者
d）情况 4. 根标识符和根路径成本与端口记录的一致，并且桥标识符与端口记录的指定桥相同，并且

1) 接收 BPDU 的网桥不是该端口的指定网桥，或者
2） 端口标识符表示优先级不低于指定端口的端口。

8.6.2.3 程序

8.6.2.3.1 步骤 1

为端口保存的指定根、指定成本、指定桥和指定端口参数设置为接收到的配置 BPDU 中传达的根标识符、根路径成本、桥标识符和端口标识符参数的值。

8.6.2.3.2 第 2 步

端口的消息年龄计时器启动，从收到的配置 BPDU 中传达的消息年龄参数的值开始运行。

8.6.3 记录配置超时值

8.6.3.1 目的

将 Max Age、Hello Time、Forward Delay 和 Topology Change 标志参数更新为从 Root 接收的最新值。

8.6.3.2 使用

在根端口上收到配置 BPDU 后，将调用记录配置信息程序 (8.6.2.2)。

8.6.3.3 程序

网桥持有的最大年龄、Hello Time、转发延迟和拓扑变化参数被设置为接收到的配置 BPDU 中传达的值。

8.6.4 配置BPDU的生成

8.6.4.1 目的

向连接到每个 LAN 的桥接器（对于该桥接器而言，该桥接器是指定桥接器）传达指定根、根路径成本、指定桥接器、指定端口以及协议计时器的值的知识。

8.6.4.2 使用

a) 情况 1. 在根端口上收到配置 BPDU 后，将调用记录配置信息程序（8.6.2.2）。

b) 情况 2. Hello Timer 到期后。
c) 情况 3. 当桥接端口的消息年龄计时器到期时，通过配置更新过程选择该桥接器作为指定根。

d) 案例 4. 通过管理措施选择该桥作为指定根。

8.6.4.3 程序

对于每个作为其所连接的 LAN 的指定端口的端口（即，为该端口保存的指定桥接器和指定端口参数的值分别与该端口的桥接器标识符和端口标识符的值相同，且不处于禁用状态），使用传输配置 BPDU 过程（8.6.1）。

8.6.5 回复配置BPDU

8.6.5.1 目的

当另一个桥接端口已在该 LAN 上传输配置 BPDU 时，为该 LAN 建立指定桥接和指定端口。如果传输桥接尚未收到来自当前根的配置消息（原因是该根是新建立的，或者由于 BPDU 丢失以及随后的消息年龄计时器到期），就会发生这种情况。

8.6.5.2 使用

在所连接的 LAN 的指定端口上接收到配置 BPDU 后，不会更新该端口保存的信息，即，不满足使用记录配置信息过程的条件（8.6.2.2）。

8.6.5.3 程序

传输配置 BPDU 过程 (8.6.1) 用于接收配置 BPDU 的端口。

8.6.6 发送拓扑改变通知 BPDU

8.6.6.1 目的

通知通向根的路径上的桥接器，传输桥接器已检测到拓扑的扩展。最终，这将导致根收到拓扑更改通知。

8.6.6.2 使用

a) 情况 1. 非根桥检测或收到拓扑变化通知后。

b) 情况 2. 拓扑改变通知计时器到期后。

8.6.6.3 程序

拓扑改变通知 BPDU 应在最大 BPDU 传输延迟时间内通过根端口传输（8.10.2）。

8.6.7 配置更新

8.6.7.1 目的

更新桥接器和桥接器端口所持有的配置信息。

8.6.7.2 使用

a) 情况 1. 收到配置 BPDU 后，将调用记录配置信息程序（8.6.2.2）。

b) 情况 2. 当某个端口的消息年龄计时器到期时，该端口将成为其所连接的 LAN 的指定端口。

c) 情况 3. 因管理行动导致港口国发生变化。

8.6.7.3 程序

8.6.7.3.1 步骤 1

应使用根选择程序（8.6.8）来选择指定根和根端口，并计算此桥的根路径成本。

8.6.7.3.2 第 2 步

应使用指定端口选择程序（8.6.9）来确定每个端口是否应成为其所连接的 LAN 的指定端口。

8.6.8 根选择

8.6.8.1 目的

选择指定根和根端口，并计算该桥的根路径成本。

8.6.8.2 使用

此过程由配置更新过程（8.6.7）使用。

8.6.8.3 程序

8.6.8.3.1 步骤 1

根端口被设置为在那些不是其所连接的 LAN 的指定端口中标识那些没有被禁用、并且具有比桥的桥标识符具有更高优先级的指定根参数的端口；

a) 具有与之关联的最高优先级根，即记录为该端口的指定根。
b) 在两个或多个具有最高优先级指定根参数的端口中，具有与之关联的最低根路径成本，即任何端口的指定成本与路径成本参数之和最低；或

c) 在两个或多个具有最高优先级指定根参数和最低相关根路径成本值的端口中，具有最高优先级桥标识符，记录为该端口所连接的 LAN 的指定桥；或者

d) 在两个或多个具有最高优先级指定根参数、相关根路径成本最低的值和最高优先级指定桥的端口中，将最高优先级端口标识符记录为该端口所连接的 LAN 的指定端口；或者

e) 在两个或多个具有最高优先级指定根参数、相关根路径成本最低的值以及最高优先级指定桥和指定端口的端口中，具有最高优先级端口标识符。

8.6.8.3.2 第 2 步

如果不存在这样的端口，则根端口参数的值设置为零，并且

a) 桥接器持有的指定根参数设置为桥接器持有的桥接器标识符参数，并且

b) Bridge 持有的 Root Path Cost 参数值被设置为零。

8.6.8.3.3 步骤 3

否则，即如果其中一个桥端口已被确定为根端口，则

a) 桥接器持有的指定根参数设置为根端口持有的指定根参数，并且

b) Bridge 持有的 Root Path Cost 参数值被设置为与 Root Port 关联的 Root Path Cost 参数值，也就是为 Root Port 记录的 Path Cost 和 Designated Cost 参数值之和。

8.6.9 指定端口选择

8.6.9.1 目的

对于每个端口，确定该端口是否应该是其所连接的 LAN 的指定端口。

8.6.9.2 使用

作为配置更新程序的一部分（8.6.7）。

8.6.9.3 程序

应为每个港口启动成为指定港口的程序（8.6.10）：

a) 已被选为其所连接的 LAN 的指定端口，即，该端口保存的指定桥接器和指定端口参数的值分别与该端口的桥接器标识符和端口标识符的值相同，或者

b) 桥接器记录的指定根参数与端口记录的不同（请注意，此过程遵循根选择），或

c) 网桥为端口所连接的 LAN 提供到根的低成本路径，即网桥记录的根路径成本小于为端口记录的指定成本，或者
d) 该桥向根提供一条同等成本的路径，并且该桥的桥标识符表示的桥的优先级高于该端口记录的指定桥，或者
e) 该桥向根提供一条等价路径，并且该桥是该端口所连局域网的指定桥，该端口的端口标识符比记录为指定端口的端口标识符具有更高的优先级。

8.6.10 成为指定端口

8.6.10.1 目的

假设某个端口是其所连接的 LAN 上的指定端口，则需要为那些确定桥接 LAN 的活动拓扑的端口参数分配适当的值。

8.6.10.2 使用

a) 情况 1. 端口的消息年龄计时器到期后。
b) 情况 2. 作为配置更新过程（8.6.7）的一部分，指定端口选择过程（8.6.9）将该端口选为其所连接的 LAN 的指定端口。

c) 情况 3. 港口国因管理行动而发生变化后（8.8.1、8.8.2、8.8.3 和 8.8.5）。

8.6.10.3 程序

a) 将为Port持有的Designated Root参数设置为Bridge持有的Designated Root参数的值；

b) 将端口持有的指定成本参数设置为桥持有的根路径成本的值；

c) 为端口保存的指定桥参数设置为桥的桥标识符；
d) 为端口保存的指定端口参数设置为端口的端口标识符。

8.6.11 港口国选择

8.6.11.1 目的

根据更新的配置信息选择桥接端口的状态，该配置信息指示每个端口在桥接 LAN 的活动拓扑中的角色，即它是否应该

a) 作为桥的根端口。
b) 成为指定港口。
c) 作为备用端口，即，充当冗余连接的桥接 LAN 中的备份端口。

8.6.11.2 使用

使用配置更新程序后

a) 情况 1. 收到一个配置 BPDU，该配置 BPDU 传达的信息取代了端口记录的信息。

b) 情况 2. 端口的消息年龄计时器到期，导致该端口成为其所连接的 LAN 的指定端口。

c) 情况 3. 因管理行动导致港口状态发生变化。

8.6.11.3 程序

对于每个桥的端口：

a) 如果该端口是桥的根端口，则
    1) 重置端口的Configuration Pending标志参数和Topology Change Acknowledge标志参数。

    2） 对于端口，使用 Make Forwarding 规程（8.6.12）。
b) 否则，如果端口是其所连接的 LAN 的指定端口，即端口的指定桥接器参数与桥接器持有的桥接器标识符参数相同，并且端口持有的指定端口和端口标识符参数相同，且端口不处于禁用状态，则

    1) 如果端口的消息年龄计时器正在运行，则将其停止。
    2） 对于端口，使用 Make Forwarding 规程（8.6.12）。
c) 否则，如果该端口是备用端口，即既不是根端口也不是指定端口，则

    1) 重置端口的Configuration Pending标志参数和Topology Change Acknowledge标志参数。

    2) 使用进行阻塞的程序（8.6.13）。

8.6.12 进行转发

8.6.12.1 目的

允许端口参与帧中继，按照适当的间隔来确保桥接 LAN 中的临时环路不会导致帧重复。

8.6.12.2 使用

作为港口国选择程序的一部分（8.6.11）。

8.6.12.3 程序

如果港口国是阻塞的，那么

a) 端口状态设置为监听，并且
b) 端口的转发延迟计时器已启动。

8.6.13 进行阻塞

8.6.13.1 目的

终止端口参与帧中继。

8.6.13.2 使用

作为港口国选择程序的一部分（8.6.11）。

8.6.13.3 程序

如果端口未处于禁用或阻塞状态，则

a) 如果端口处于转发或学习状态，且设置了端口的变化检测启用参数，则调用拓扑变化检测程序（8.6.14）；
b) 该端口的端口状态设置为Blocking；
c) 端口的转发延迟计时器停止。

8.6.14 拓扑变化检测

8.6.14.1 目的

记录 Bridge 检测到的或通知给 Bridge 的拓扑变化，并采取行动将检测到拓扑变化的事实传达给 Root。

8.6.14.2 使用

8.6.14.2.1 情况 1

在所连接的 LAN 的指定端口上接收到拓扑改变通知 BPDU。

8.6.14.2.2 情况 2

当桥接端口在该端口的转发延迟计时器到期后进入转发状态时，前提是该端口的“启用变化检测”参数已设置，并且桥接是其端口所连接到的至少一个 LAN 的指定桥接。

8.6.14.2.3 情况 3

当处于转发状态或学习状态的桥接端口进入阻塞状态时，前提是该端口的“更改检测启用”参数已设置。

8.6.14.2.4 情况 4

当桥成为根时。

8.6.14.3 程序

a) 如果该网桥已被选为根，即该网桥的指定根和网桥标识符参数相同，则

    1) 设置网桥保持的拓扑改变标志参数。
    2) 网桥的拓扑改变定时器启动。

b) 如果该网桥尚未被选为根，并且该网桥的“检测到拓扑变化”标志参数尚未设置，则

    1) 调用传输拓扑改变通知BPDU程序（8.6.6）。
    2) 启动拓扑改变通知定时器。

c) 设置网桥的“拓扑变化检测”标志参数。

8.6.15 拓扑变化已确认

8.6.15.1 目的

终止拓扑改变通知 BPDU 的传输。

8.6.15.2 使用

从根端口所连接的 LAN 的指定桥接收到带有拓扑改变确认标志参数设置的配置 BPDU 之后。

8.6.15.3 程序

a) 重置桥接器保存的拓扑变化检测标志参数；
b) 拓扑改变通知计时器停止。

8.6.16 确认拓扑变化

8.6.16.1 目的

确认另一个网桥检测到的拓扑变化的通知。

8.6.16.2 使用

在所连接的 LAN 的指定端口上收到拓扑改变通知 BPDU 后。

8.6.16.3 程序

a) 设置该端口的拓扑改变确认标志参数；
b) 端口使用传输配置 BPDU 程序（8.6.1）。

8.7 协议的运行

桥接协议实体应

a) 通过传输桥接协议数据单元与其他桥接中的对等实体进行通信；

b) 更新存储的协议变量和计时器；
c) 改变桥端口的状态；

下列的

— 桥接协议数据单元的接收;
— 桥梁和港口计时器到期；

按照以下规范的要求（8.7.1、8.7.2、8.7.3、8.7.4、8.7.5、8.7.6、8.7.7 和 8.7.8）。如有歧义，应参考程序模型（8.9），该模型构成了协议操作的最终描述。

本规范使用 8.6 中描述的协议程序元素，这些元素与本子条款以及 8.5 中描述的协议参数和计时器一起，提供了生成树算法和协议的抽象描述。通过维护协议参数和计时器以及传输 BPDU（如所述），可实现对本规范的符合性。实施不受其他限制；特别是，不存在对单个程序元素的符合性。

8.7.1 收到的配置BPDU

8.7.1.1 案例 1

如果收到的配置 BPDU 传达了取代 8.6.2.2 中指定的端口已保存的协议信息，则使用以下步骤序列：

a)步骤1.记录配置信息程序（8.6.2）。
b)步骤2.配置更新程序（8.6.7）。
c) 步骤3.港口国选择程序（8.6.11）。
d) 步骤 4. 如果在配置更新之前，该网桥被选为根，但不再是根，则 Hello Timer (8.5.4.1) 将停止。

e) 步骤 5. 如果在配置更新之前，该网桥被选为根，但是现在不再是根，并且设置了“检测到拓扑变化”标志参数，则停止拓扑变化计时器，使用传输拓扑变化通知 BPDU 程序（8.6.6），并且启动拓扑变化通知计时器。

f) 步骤 6. 如果在根端口（即，由配置更新过程选择为根端口的端口）上接收到配置 BPDU，则使用记录配置超时值（8.6.3）和配置 BPDU 生成（8.6.4）过程。

g) 步骤 7. 如果在根端口上收到了配置 BPDU，并且设置了拓扑改变确认标志参数，则使用拓扑改变确认程序（8.6.15）。

8.7.1.2 案例 2

如果收到的配置 BPDU 没有传达取代端口已保存的信息，并且该端口是其所连接的 LAN 的指定端口，即，端口保存的指定桥和指定端口参数的值与桥的桥标识符和该端口的端口标识符的值相同，则使用回复配置 BPDU 程序 (8.6.5)。

8.7.2 收到拓扑变化通知 BPDU

如果收到拓扑变化通知 BPDU 的端口是其所连接的 LAN 的指定端口，则

a)步骤1.使用拓扑变化检测程序（8.6.14）。
b) 步骤2.使用确认拓扑变化程序（8.6.16）。

8.7.3 Hello 定时器到期

使用配置 BPDU 生成过程 (8.6.4) 并启动 Hello Timer (8.5.4.1)。

8.7.4 消息年龄定时器到期

a) 步骤 1. 对消息年龄计时器已过期的端口使用成为指定端口的程序（8.6.10）。

b)步骤2.使用配置更新程序（8.6.7）。
c) 步骤3.使用港口国选择程序（8.6.11）。
d) 步骤 4. 如果在配置更新后，该网桥被选为根，则
1) 步骤1. 将Bridge持有的Max Age、Hello Time、Forward Delay参数设置为Bridge Max Age、Bridge Hello Time、Bridge Forward Delay参数的值。
2)步骤2.使用拓扑变化检测程序(8.6.14)。
3)步骤3.拓扑改变通知计时器(8.5.4.2)停止。
4) 步骤4. 使用配置BPDU生成过程（8.6.4）并启动Hello Timer。

8.7.5 转发延迟定时器到期

a) 步骤 1. 如果转发延迟计时器（8.5.6.2）已到期的端口状态为监听，则

1）步骤1.将端口状态设置为学习状态，并且
2)步骤2，转发延迟定时器重新启动。
b) 步骤 2. 否则，如果转发延迟计时器（8.5.6.2）已到期的端口状态为学习，则

1）步骤1. 端口状态设置为转发，并且
2) 步骤2.如果该网桥是其端口所连接的至少一个 LAN 的指定网桥，并且设置了该端口的“更改检测启用”参数，则调用拓扑更改检测程序（8.6.14）。

8.7.6 拓扑变化通知定时器到期

a) 步骤1.使用传输拓扑改变通知BPDU程序（8.6.6）。
b) 步骤2. 拓扑改变通知计时器（8.5.4.2）重新启动。

8.7.7 拓扑变化定时器

a) 步骤1. 重置网桥持有的“拓扑变化检测”标志参数。
b) 步骤2. 重置网桥持有的拓扑改变标志参数。

8.7.8 保持定时器到期

如果设置了保持计时器 (8.5.6.3) 已过期的端口的配置待处理标志参数，则会为该端口调用传输配置 BPDU 过程 (8.6.1)。

8.8 桥接协议实体的管理

可以对桥接协议实体进行管理控制，该实体操作生成树算法和协议，以便

— 满足任何本地信息和配置服务的要求。
— 支持管理运营。

本节规定了下列管理操作与生成树算法和协议的参数和过程的交互：

a）初始化；
b) 启用单个端口；
c) 禁用单个端口；
d) 改变桥标识符的优先级部分；
e) 改变端口标识符的优先级部分；
f) 更改与单个端口相关的路径成本；
g) 启用或禁用与单个端口相关的拓扑变化检测。

这些操作应修改协议参数和计时器并传输 BPDU，如下所述（8.8.1、8.8.2、8.8.3、8.8.4、8.8.5、8.8.6、8.8.7 和 8.8.8）。实施不受其他限制；特别是，程序的各个元素没有一致性。如有歧义，应参考程序模型（8.9），该模型构成了这些操作的最终描述。

本款未指定哪些操作可供远程管理站使用，也未指定这些操作如何组合和传达。远程管理可提供的操作和设施在第 14 款中详细说明。同样，本款未指定本地信息和配置程序的可用性。

8.8.1 初始化

a) 步骤1. 为网桥保存的指定根参数设置为等于网桥标识符的值，并且为网桥保存的根路径成本和根端口参数的值设置为零。

b) 步骤2. 将Bridge 持有的Max Age、Hello Time、Forward Delay 参数设置为Bridge Max Age、Bridge Hello Time、Bridge Forward Delay 参数的值。
c) 步骤3. 重置网桥的“检测到拓扑变化”和“拓扑变化标志”参数，以及“拓扑变化通知计时器”（8.5.4.2）和“拓扑变化计时器”（8.5.4.3）
如果正在运行则停止。
d) 步骤 4. 对于每个桥接端口
1)步骤1.成为指定端口程序（8.6.10）用于为端口的指定根、指定成本、指定桥和指定端口参数分配值。

2)步骤2.如果端口在初始化之后要被启用，则将端口状态设置为Blocking；否则，将端口状态设置为Disabled。
3)步骤3，重置拓扑改变确认标志参数。
4)步骤4.重置Configuration Pending标志参数。
5)步骤5.消息年龄计时器(8.5.6.1)如果正在运行则停止。
6)步骤6.如果转发延迟定时器（8.5.6.2）正在运行，则停止。
7)步骤7.如果保持计时器（8.5.6.3）正在运行，则停止。
8) 步骤8. 设置变化检测启用标志（8.5.5.10）。

e) 步骤5. 端口状态选择程序（8.6.11）用于选择每个桥端口的状态。

f) 步骤 6. 调用配置 BPDU 生成程序 (8.6.4)，并启动 Hello Timer（8.5.4.1）已开始。

8.8.2 启用端口

a) 步骤1.成为指定端口程序（8.6.10）用于为端口的指定根、指定成本、指定桥和指定端口参数分配值。
b) 步骤2. 端口状态设置为Blocking。
c)步骤3，重置拓扑改变确认标志参数。
d) 步骤4.重置配置挂起标志参数。
e) 步骤5. 如果消息年龄计时器（8.5.6.1）正在运行，则停止。
f) 步骤 6. 如果转发延迟计时器 (8.5.6.2) 正在运行，则停止。
g) 步骤 7. 如果保持计时器 (8.5.6.3) 正在运行，则停止。
h) 步骤8.使用港口国选择程序（8.6.11）。

8.8.3 禁用端口

a) 步骤1.成为指定端口程序（8.6.10）用于为端口的指定根、指定成本、指定桥和指定端口参数分配值。
b) 步骤2. 端口状态设置为禁用。
c)步骤3，重置拓扑改变确认标志参数。
d) 步骤4.重置配置挂起标志参数。
e) 步骤5. 如果消息年龄计时器（8.5.6.1）正在运行，则停止。
f) 步骤 6. 如果转发延迟计时器 (8.5.6.2) 正在运行，则停止。
g)步骤7.使用配置更新程序（8.6.7）。
h) 步骤8.使用港口国选择程序（8.6.11）。
i) 步骤 9. 如果在配置更新后，该网桥被选为根，则
1) 步骤1. 将Bridge持有的Max Age、Hello Time、Forward Delay参数设置为Bridge Max Age、Bridge Hello Time、Bridge Forward Delay参数的值。
2)步骤2.使用拓扑变化检测程序(8.6.14)。
3)步骤3.拓扑改变通知计时器(8.5.4.2)停止。
4) 步骤4. 使用配置BPDU生成过程（8.6.4）并启动Hello Timer。

8.8.4 设置桥优先级

a) 步骤1.计算桥标识符的新值。
b) 步骤 2. 对于每个已被选为其所连接的 LAN 的指定端口的端口，将保存的指定桥接器参数的值（即，指定桥接器和指定端口参数的值分别与该端口的桥接器标识符和端口标识符的值相同；）设置为桥接器标识符的新值。

c) 步骤3.桥接器持有的桥接器标识符参数被设置为新值。
d)步骤4.使用配置更新程序（8.6.7）。
e) 步骤5.使用港口国选择程序（8.6.11）。

f) 步骤 6. 如果在配置更新后，该网桥已被选为根，则
1) 步骤1. 将Bridge持有的Max Age、Hello Time、Forward Delay参数设置为Bridge Max Age、Bridge Hello Time、Bridge Forward Delay参数的值。
2)步骤2.使用拓扑变化检测程序(8.6.14)。
3)步骤3.拓扑改变通知计时器(8.5.4.2)停止。
4) 步骤4. 使用配置BPDU生成过程（8.6.4）并启动Hello Timer。

8.8.5 设置端口优先级

a)步骤1.计算端口标识符的新值。
b) 步骤2. 如果该端口已被选为其所连接的 LAN 的指定端口（即，指定桥接器和指定端口参数的值分别与桥接器标识符和端口标识符的值相同），则为该端口保存的指定端口参数将设置为端口标识符的新值。

c) 步骤3. 将为端口保存的端口标识符参数设置为新值。
d) 步骤 4. 如果为端口保存的指定桥参数的值等于桥的桥标识符的值，并且端口标识符的新值的优先级高于记录为指定端口的值，则

1)步骤1.成为指定端口程序（8.6.10）用于为端口的指定根、指定成本、指定桥和指定端口参数分配值。
2)步骤2.使用港口国选择程序（8.6.11）。

8.8.6 设置路径成本

a) 步骤1.端口的路径成本参数设置为新值。
b)步骤2.使用配置更新程序（8.6.7）。
c) 步骤3.使用港口国选择程序。

8.8.7 启用变化检测

已设置端口（8.5.5.10）的变化检测启用标志。

8.8.8 禁用变更检测

端口（8.5.5.10）的变化检测启用标志已重置。

8.9 程序模型

本小节构成了生成树算法和协议操作的权威描述。本标准 8.6、8.7 和 8.8 中的自然语言文本旨在非正式地呈现此处指定的操作语义。如果该文本与本程序模型之间存在解释差异，则以后者为准。

8.9.1 概述

协议的参数、计时器、程序元素和操作下面以计算机语言 C（ANSI X3.159）的可编译程序形式呈现。

介绍此程序的目的是准确、明确地说明算法和协议的操作。用计算机语言描述协议的操作绝不是为了限制协议的实现；实际实现可以采用任何适当的技术。

设备是否符合本标准完全取决于可观察协议。本小节中的程序包含与实现相关的建模细节；这些细节并不存在一致性。

8.6、8.7 和 8.8 中的自然语言文本遵循本小节中的计算机语言文本。为了保持程序文本的紧凑性，所有注释均参考自然语言描述，格式为 4.nnn。当程序语句调用过程元素时，将进一步引用该过程所列的特定条件，这些条件导致调用该过程。

在程序模型的计算机语言描述中，应该注意的是，BPDU 以解包形式显示，即解包为各个参数。跨 LLC 服务边界传递的 BPDU 的实际格式如第 9 条所示。

```
/**************************************************** ************************
*
* 生成树算法和协议
*
*****************************************************************************/

/**************************************************** ************************
* 定义常数
*****************************************************************************/

# 定义    零              0
# 定义    一              1

# 定义    错误的          0
# 定义    真的            1

/** 端口状态。**/

# 定义    已禁用              0                                   /*  (8.4.5)             * /
# 定义    听力                1                                   /*  (8.4.2)             * /
# 定义    学习                2                                   /*  (8.4.3)             * /
# 定义    转发                3                                   /*  (8.4.4)             * /
# 定义    阻塞                4                                   /*  (8.4.1)             * /

/** BPDU 类型常量 **/

# 定义    配置bpdu类型                0
# 定义    Tcn_bpdu_类型             128

/** 伪实现常量。**/

# 定义    端口数量 2
          /* 任意选择，以允许下面的代码编译 */

# 定义    所有端口 端口数+1
          /* 端口从 1 开始，C 中的数组从 0 开始 */

# 定义    默认路径成本 10
          /* 随意的 */

# 定义    消息年龄增量 1
          /* 尽可能减少年龄的估计，允许
               BPDU 传输时间 */

# 定义    无端口 0
          /* Bridge 根端口参数的保留值，表示无
               根端口，当 Bridge 为根时使用 */
```

```
/************************************************************ ************************
* 类型定义、结构和联合声明
*****************************************************************************/

/** 基本类型。**/

类型定义 int Int;                    /* 与此处使用的约定保持一致
                                        大小写。类型和定义的常量的首字母大写。*/


类型定义 Int Boolean;                /* :（真，假）*/

类型定义 Int State;                  /* :（残疾人、听力、学习、
                                              转发、阻止）*/


/** 第 9 章“桥接协议数据单元的编码”中定义的 BPDU 编码类型为:

协议版本                    (9.2.2)

Bpdu 类型                   (9.2.3)

旗帜                        (9.2.4)

标识符                      (9.2.5)

成本                        (9.2.6)

端口号                      (9.2.7)

时间                        (9.2.8)

* * /

# 包括“types.c”                    /* 定义 BPDU 编码类型 */

/** 配置 BPDU 参数（8.5.1）**/

类型定义结构
{
    Bpdu_type类型;

    标识符      根ID;                                    /*  (8.5.1.1)      * /

    成本        根路径成本;                              /*  (8.5.1.2)      * /

    标识符      桥梁ID;                                  /*  (8.5.1.3)      * /

    端口号      端口ID;                                  /*  (8.5.1.4)      * /

    时间        消息年龄;                                /*  (8.5.1.5)      * /

    时间        最大年龄;                                /*  (8.5.1.6)      * /

    时间        你好时间;                                /*  (8.5.1.7)      * /

    时间        转发延迟;                                /*  (8.5.1.8)      * /

    旗帜        拓扑变化确认;                            /*  (8.5.1.9)      * /

    旗帜        拓扑变化;                                /*  (8.5.1.10)     * /

}  配置_bpdu;
```

```
/** 拓扑变化通知 BPDU 参数（8.5.2）**/

类型定义结构
{
      Bpdu_type类型;

} Tcn_bpdu;

/** 桥接参数 (8.5.3) **/

类型定义结构
{
      标识符designated_root;                     /*（8.5.3.1）    * /

      成本          根路径成本;                   /*  (8.5.3.2)    * /

      整数          根端口;                       /*  (8.5.3.3)    * /

      时间          最大年龄;                     /*  (8.5.3.4)    * /

      时间          你好时间;                     /*  (8.5.3.5)    * /

      时间          转发延迟;                     /*  (8.5.3.6)    * /

      标识符        桥梁ID;                       /*  (8.5.3.7)    * /

      时间          桥梁最大年龄;                 /*  (8.5.3.8)    * /

      时间          桥接器你好时间;               /*  (8.5.3.9)    * /

      时间          桥接转发延迟;                 /*  (8.5.3.10)   * /

      布尔值        检测到拓扑变化;               /*  (8.5.3.11)   * /

      布尔值        拓扑变化;                     /*  (8.5.3.12)   * /

      时间          拓扑变化时间;                 /*  (8.5.3.13)   * /

      时间          保持时间;                     /*  (8.5.3.14)   * /

}  桥接数据;

/** 端口参数（8.5.5）**/

类型定义结构
{
      端口号        端口ID;                       /*（8.5.5.1）    * /

      状态          状态;                         /*  (8.5.5.2)    * /

      整数          路径成本;                     /*  (8.5.5.3)    * /

      标识符        指定根;                       /*  (8.5.5.4)    * /

      整数          指定成本;                     /*  (8.5.5.5)    * /

      标识符        指定桥梁;                     /*  (8.5.5.6)    * /

      端口号        指定端口;                     /*  (8.5.5.7)    * /

      布尔值        拓扑变化确认;                 /*  (8.5.5.8)    * /

      布尔值        配置待定;                     /*  (8.5.5.9)    * /
```



```
    布尔值        改变检测已启用;                              /*（8.5.5.10）          * /

} 端口数据;

/** 类型来支持此伪实现的计时器。**/

类型定义结构
{
    布尔值       积极的;          /* 计时器正在使用。 */

    时间         价值;            /* 计时器的当前值，计数完毕。 */

} 计时器;


/*********************************************************** **************************
* 静态存储分配
***********************************************************************************/


Bridge_data       桥梁信息;                                  /*   (8.5.3)           * /

端口数据          端口信息[所有端口];                        /*   (8.5.5)           * /

配置bpdu          config_bpdu[所有端口];

端口映射表        tcn_bpdu[所有端口];

计时器            你好_计时器;                               /*   (8.5.4.1)         * /

计时器            tcn_计时器;                                /*   (8.5.4.2)         * /

计时器            拓扑变化计时器;                            /*   (8.5.4.3)         * /

计时器            消息年龄计时器[所有端口];                  /*   (8.5.6.1)         * /

计时器            转发延迟定时器[所有端口];                  /*   (8.5.6.2)         * /

计时器            保持定时器[所有端口];                      /*   (8.5.6.3)         * /

/*********************************************************** **************************
* 代码
***********************************************************************************/

/** 程序元素 (8.6) **/

传输配置（端口号） Int                                       /*   (8.6.1)           * /
            端口号;
{
    如果（hold_timer[port_no].active）                       /*   (8.6.1.3.1)       * /
    {
        port_info[port_no].config_pending        =  真的;   /*   (8.6.1.3.1)       * /
    }
    别的                                                     /*   (8.6.1.3.2)       * /
    {
        config_bpdu[端口号].类型          =配置_bpdu_类型;

        config_bpdu[port_no].root_id = bridge_info.designated_root;
                                                             /*（8.6.1.3.2（a））*/
        config_bpdu[port_no].根路径成本=桥接信息.根路径成本;
                                                             /*   (8.6.1.3.2（b））*/
        config_bpdu[端口号].bridge_id          =桥梁信息.桥梁ID;
                                                             /*   (8.6.1.3.2（c））*/
        config_bpdu[端口号].端口id =            端口信息[端口号].端口ID;
```

```
                                                                    /*   (8.6.1.3.2（d））*/
        如果（根桥（））
        {
            config_bpdu[port_no].message_age = 零；                  /*   (8.6.1.3.2（e））*/
        }
        别的
        {
            config_bpdu[端口号].message_age
               =   message_age_timer[bridge_info.root_port].值
                   +消息年龄增加；                                   /*   (8.6.1.3.2（f））*/
        }
        config_bpdu[端口号].max_age config_bpdu[端口号].最大年龄;    /*   (8.6.1.3.2（g））*/
        config_bpdu[端口号].forward_delay config_bpdu[端口号].
        号].topology_change_acknowledgment          =桥接信息.转发延迟；

           =端口信息[端口号].拓扑变化确认；
                                                                    /*  (8.6.1.3.2(h))          * /
        config_bpdu[端口号].拓扑变化
           =桥梁信息.拓扑变化；                                     /*  (8.6.1.3.2(—))          * /

        如果 (config_bpdu[port_no].message_age < bridge_info.max_age) {

            端口信息[端口号].拓扑变化确认 = False;
                                                                        /*  (8.6.1.3.3)         * /
            port_info[port_no].config_pending = False;                  /*   (8.6.1.3.3）*/
            send_config_bpdu(port_no，&config_bpdu[port_no]);
            start_hold_timer(port_no);                                  /*   (8.6.3.3（b））*/
         }
    }

    }
        /* 在哪里

        send_config_bpdu(port_no,bpdu) Int
                        端口号；
        配置bpdu        *bpdu；

        是一个伪实现的特定例程，它在指定的时间内在指定的端口上传输 bpdu。


        * /

        /*  和 */

        布尔根桥()
        {
            返回（bridge_info.designated_root          ==  桥梁信息.桥梁ID);
        }
布尔值    supersedes_port_info（端口号，          配置）          /*  (8.6.2.2)               * /
int 端口号；
配置bpdu        * 配置；
{
    返回       （
     （ 配置->root_id
        <   port_info[port_no].designated_root                      /*  (8.6.2.2      一个）  * /
    )
    ||
     （  （ 配置->root_id
            ==  port_info[port_no].designated_root
        )
        &&
         （  （ 配置->根路径成本
                < 港口信息[港口编号].指定成本                       /*  (8.6.2.2      b)        * /
            )
            ||
```

```
            ((      配置->根路径成本
                    ==  port_info[港口编号].designated_cost
                )
                &&
                 (   (  配置->bridge_id
                        <   port_info[port_no].指定桥梁                    /*  (8.6.2.2      c)      */
                    )
                    ||
                     (   (  配置->bridge_id
                            ==  port_info[port_no].指定桥梁
                        )                                                  /*  (8.6.2.2      d)      */
                        &&
                         (   (  配置->bridge_id != bridge_info.bridge_id
                            )                                              /*（8.6.2.2 d1）*/
                            ||
                             (  配置->端口号
                                <= port_info[port_no].指定端口
                            )                                              /*（8.6.2.2 d2）*/
    )))))
             ) ;
}

记录配置信息（端口号，配置）Int 端口号；                                    /*  (8.6.2)             */

配置_bpdu *配置；
{
    端口信息[端口号].指定根端口信息[端口号].指定成本=  配置->root_id；          /*  (8.6.2.3.1)         */
    端口信息[端口号].指定桥端口信息[端口号].指定端口=  配置->根路径成本；=配置->桥接
    =                                                  ID；
                                                       配置->端口ID；

    启动消息定时器（port_no，config->message_age）；                        /*（8.6.2.3.2）       */
}

record_config_timeout_values(配置)                                          /*  (8.6.3)             */
Config_bpdu *config;
{
    bridge_info.max_age =        配置->最大年龄；                           /*  (8.6.3.3)           */
    bridge_info.hello_time        =  配置->hello_time；
    bridge_info.forward_delay          =  配置->转发延迟；
    bridge_info.topology_change          =配置->拓扑_改变；
}

配置bpdu_generation（）{                                                    /*（8.6.4）           */

    int 端口号；

    对于（port_no = 一；port_no <= 端口数；port_no++）{                     /*（8.6.4.3）         */

        如果（指定端口（端口号）                                            /*  (8.6.4.3)           */
                &&
                 （port_info[port_no].state != 已禁用）
            )
        {
            传输配置（端口号）；                                            /*  (8.6.4.3)           */
        }                                                                   /*  (8.6.1.2)           */
    }
}

        /* 在哪里 */

        Boolean specified_port(port_no) Int
        port_no;
        {
            返回（（port_info[port_no].designated_bridge
```

```
                                == bridge_info.bridge_id
                            )
                            &&
                             (  port_info[port_no].指定端口
                                  == 端口信息[端口号].端口 ID
                            )
                        ) ;
            }
回复（端口号）                                                          /*  (8.6.5)             * /
int 端口号;
{
    传输配置（端口号）;                                                 /*  (8.6.5.3)           * /
}


发送_tcn()                                                              /*（8.6.6）           * /
{
    int 端口号;

    port_no = bridge_info.root_port;
    tcn_bpdu[port_no].type = Tcn_bpdu_type;

    send_tcn_bpdu(port_no，&tcn_bpdu[bridge_info.root_port]);/* (8.6.6.3)                      * /

}

        /* 在哪里

        send_tcn_bpdu(port_no,bpdu) Int
                        端口号;
        端口映射表      *bpdu;

        是一个伪实现特定例程，用于在指定的时间内在指定的端口上传输 bpdu。


        * /

配置更新（）                                                            /*  (8.6.7)             * /
{
    根选择（）;                                                         /*    (8.6.7.3.1)       * /
                                                                        /*  (8.6.8.2)           * /

    指定端口选择（）;                                                   /*    (8.6.7.3.2)       * /
                                                                        /*  (8.6.9.2)           * /
}

根选择（）                                                               /*（8.6.8）           * /
{
    整数 根端口;
    整数 端口号;

    根端口 = 无端口;

    对于（port_no = 一；port_no <= 端口数；port_no++）{                    /*（8.6.8.3.1）*/

        如果（（    （! 指定端口（端口号））&&

                    （port_info[port_no].state != 已禁用）&&

                    （端口信息[端口号].designated_root < 桥接信息.桥接ID）
                )
                &&
                 (  （根端口 == 无端口）
                    ||
                    （port_info[port_no].designated_root
```

```
                    <   port_info[根端口].designated_root/* (8.6.8.3.1(a))                    * /
                )
                ||
                 (   (  port_info[port_no].designated_root ==
                        port_info[root_port].designated_root
                    )
                    &&
                    ( ( ( port_info[port_no].designated_cost
                                + 端口信息[端口号].路径成本
                            )
                            <
                             (  port_info[根端口].designated_cost
                                + 端口信息[根端口].路径成本
                            )                                                /* （8.6.8.3.1 （b））    * /
                        )
                        ||
                        ( ( ( port_info[port_no].designated_cost
                                    + 端口信息[端口号].路径成本
                                )
                                ==
                                 (  port_info[根端口].designated_cost
                                    + 端口信息[根端口].路径成本
                                )
                            )
                            &&
                            ( (       港口信息[港口编号].指定桥梁<
                                    port_info[根端口].designated_bridge
                                )                                            /*（8.6.8.3.1（c））     * /
                                ||
                                 (   (  港口信息[港口编号].指定桥梁==
                                        port_info[根端口].designated_bridge
                                    )
                                    &&
                                     (   (  port_info[port_no].指定端口 <
                                            port_info[根端口].指定端口
                                        )                                    /*（8.6.8.3.1（d））     * /
                                        ||
                                         (   (  port_info[port_no].designated_port ==
                                                port_info[root_port].designated_port
                                            )
                                            &&
                                             (  port_info[port_no].port_id <
                                                port_info[root_port].port_id
                                            )                                /*（8.6.8.3.1（e））*/
        ) ) ) ) ) ) ) )
    {
        根端口 = 端口号;
    }
}

bridge_info.root_port = 根端口;                                              /*（8.6.8.3.1）          * /

如果 (root_port == No_port) {                                                /*（8.6.8.3.2）          * /


    bridge_info.指定根                  =  桥梁信息.桥梁ID;
                                                                             /*   （8.6.8.3.2（a））  * /
    bridge_info.根路径成本              =  零;                               /*   （8.6.8.3.2（b））  * /
}
别的                                                                         /*  (8.6.8.3.3)         * /
{
    bridge_info.指定根                  =  port_info[根端口].designated_root;
                                                                             /*（8.6.8.3.3 （—））  * /
    bridge_info.根路径成本              =   (port_info[root_port].designated_cost
                                            + 端口信息[根端口].路径成本
```

```
                                        ）；                    /*  （8.6.8.3.3(b)）     * /
    }
}
指定端口选择() {                                                /*  (8.6.9)             * /

    int 端口号;

    对于（port_no = 一；port_no <= 端口数；port_no++）{          /*（8.6.9.3)            * /

        如果（指定端口（端口号）                                 /*  (8.6.9.3    一个）  * /
                ||
                (
                    端口信息[端口号].指定根 != 桥接信息.指定根
                                                                /*  (8.6.9.3    b)      * /
                )
                ||
                (   bridge_info.根路径成本 <
                        port_info[港口编号].designated_cost
                )                                               /*  (8.6.9.3    c)      * /
                ||
                (   (   bridge_info.root_path_cost ==
                            port_info[港口编号].designated_cost
                    )
                    &&
                    (   (   bridge_info.bridge_id
                            <   port_info[port_no].指定桥梁
                        )                                       /*  (8.6.9.3    d)      * /
                        ||
                        (   (   bridge_info.bridge_id
                                    == 港口信息[港口编号].指定桥梁
                            )
                            &&
                            (   端口信息[端口号].端口ID
                                <= port_info[port_no].指定端口
                            )                                   /*（8.6.9.3）   e)      * /
            ））   ）  ）
        {
            成为指定端口（端口号）；                             /*（8.6.10.2一）*/
        }
    }
}

成为指定端口(端口号) int 端口号;                                /*（8.6.10）            * /

{
    port_info[port_no].designated_root      =  桥梁信息.指定根；
                                                                /*（8.6.10.3）   一个）* /

    端口信息[端口号].designated_cost = 桥接信息.根路径成本;
                                                                /* (8.6.10.3 b)         * /

    港口信息[港口编号].指定桥 = 桥信息.桥id;
                                                                /* (8.6.10.3 c)         * /

    端口信息[端口号].指定端口 = 端口信息[端口号].端口id;
                                                                /*（8.6.10.3 d）        * /
}

端口状态选择()                                                  /*（8.6.11）            * /
{
    int 端口号;

    对于（port_no = 一；port_no <= 端口数；port_no++）{

        如果 (port_no == bridge_info.root_port)                 /* (8.6.11.3一)         * /
```

```
        {
            port_info[port_no].config_pending = False;                  /*（8.6.11.3）    a1）*/
            port_info[port_no].topology_change_acknowledge          =  错误的;

            进行转发（端口号）；                                          /*  (8.6.11.3     a2）*/
        }
        否则，如果（指定端口（端口号））{                                    /*  (8.6.11.3     b)     * /

            停止消息定时器（端口号）；                                    /*  (8.6.11.3     b1）*/

            进行转发（端口号）；                                          /*  (8.6.11.3     b2）*/
        }
        别的                                                            /*  (8.6.11.3     c)     * /
        {
            port_info[port_no].config_pending           =  错误的;      /*  (8.6.11.3     c1）*/
            port_info[port_no].topology_change_acknowledge          =  错误的;

            使阻塞（端口号）；                                            /*  (8.6.11.3     c2）*/
        }
    }
}
make_forwarding(port_no) Int                                            /*  (8.6.12)            * /
port_no;
{
    如果 (port_info[port_no].state == 阻塞) {                            /*（8.6.12.3）          * /

        设置端口状态（端口号，正在监听）；                                /*（8.6.12.3）    a）* /

        启动转发延迟定时器（端口号）；                                    /* (8.6.12.3 b)         * /
    }
}



make_blocking（端口号）                                                  /*  (8.6.13)            * /
int 端口号;
{
    如果 （（端口信息[端口号].状态&&          !=  已禁用）

          （端口信息[端口号].状态             !=  阻塞）
        )                                                               /*  (8.6.13.3)          * /
    {
        如果（（port_info[port_no].state            ==  转发)
                ||
              （端口信息[端口号].状态               ==  学习）
            )
        {
            如果 (port_info[port_no].change_detection_enabled == True)
                                                                        /*  (8.5.5.10)          * /
            {
                  拓扑变化检测（）；                                    /*  (8.6.13.3     一个）* /
            }                                                           /*  (8.6.14.2.3)        * /
        }

        设置端口状态（端口号，阻塞）；                                   /* (8.6.13.3 b)         * /

        停止转发延迟定时器（端口号）；                                   /* (8.6.13.3 c)         * /
    }
}

        /* 在哪里 */

        set_port_state(port_no，state) Int
        port_no;
        國家國家;
```

```
            {
                端口信息[端口号].状态                = 状态;
            }

拓扑变化检测() {                                                    /*  (8.6.14)            * /

    如果 (root_bridge())                                            /*  (8.6.14.3      一个）* /
    {
        bridge_info.topology_change          = 真的;                /*  (8.6.14.3      a1）*/

        启动拓扑变化计时器（）；                                    /*（8.6.14.3 a2）*/
    }

    否则，如果 (bridge_info.topology_change_detected == False) /* (8.6.14.3 b) {          * /

        传输_tcn（）；                                              /*（8.6.14.3 b1）*/

        启动_tcn_timer（）；                                        /*（8.6.14.3 b2）*/
    }

    bridge_info.拓扑变化_检测到=真；                                /* (8.6.14.3 c)        * /
}

拓扑变化已确认()                                                    /*  (8.6.15)            * /

    bridge_info.topology_change_detected          = 错误的;         /*  (8.6.15.3      一个)* /

    停止_tcn_定时器（）；                                           /* (8.6.15.3 b) */
}

确认拓扑变化(端口号) Int 端口号；                                   /*  (8.6.16)            * /

{
    port_info[port_no].topology_change_acknowledge          = 真的;  /*  (8.6.16.3      一个)* /

    传输配置（端口号）；                                           /* (8.6.16.3 b) */
}

/** 协议的操作（8.7）**/

received_config_bpdu(port_no， Int      配置）                      /*（8.7.1）           * /
port_no;
配置bpdu        * 配置;
{
    布尔值     根;

    根 = 根桥（）；

    如果（port_info [port_no] .state！=已禁用）{

        如果（supersedes_port_info（port_no，config））{            /*  (8.7.1.1)           * /
                                                                    /*  (8.6.2.2)           * /
            记录配置信息（端口号，配置）；                          /*  (8.7.1.1      一个）* /
                                                                    /*  (8.6.2.2)           * /
            配置_更新（）；                                         /*  (8.7.1.1      b)    * /
                                                                    /*  (8.6.7.2      一个）* /
            端口状态选择（）；                                      /*  (8.7.1.1      c)    * /
                                                                    /*  (8.6.11.2      一个）* /

            如果 ((!root_bridge()) && 根) {                         /* (8.7.1.1 d)         * /

                停止_hello_timer（）；

                如果（bridge_info.topology_change_detected）{       /* (8.7.1.1 e)         * /
```

```
                停止拓扑变化计时器（）；

                传输_tcn（）；                                  /*（8.6.6.1）              * /

                启动_tcn_timer（）；
            }
        }

        如果 (port_no == bridge_info.root_port) {

            记录配置超时值（配置）；                            /*  (8.7.1.1      e)       * /
                                                                /*  (8.6.3.2)              * /
            配置_bpdu_生成（）；                                /*  (8.6.4.2      一个）   * /

            如果（配置->拓扑变化确认）{                         /*   （8.7.1.1克）          * /

                拓扑变化确认（）；                              /*   （8.6.15.2）           * /
            }
        }
    }

    否则，如果（指定端口（端口号））{                           /*  (8.7.1.2)              * /

        回复（port_no）；                                       /*  (8.7.1.2)            * /
                                                                /*  (8.6.5.2)              * /
    }
  }
}

received_tcn_bpdu(port_no，tcn) int                             /*  (8.7.2)                * /
port_no;
Tcn_bpdu *tcn;
{
    如果（port_info [port_no] .state！=已禁用）{

        如果（指定端口（端口号））{

            拓扑变化检测（）；                                  /*  (8.7.2      一个）     * /
                                                                /*  (8.6.14.2.1)           * /
            确认拓扑变化（端口号）；                            /*  (8.7.2 b)              * /
        }                                                       /*  (8.6.16.2)             * /
    }
}

hello_timer_expiry()                                            /*  (8.7.3)                * /
{
    配置_bpdu_生成（）；                                        /*  (8.6.4.2      b)       * /

    启动_hello_timer（）；
}

message_age_timer_expiry(port_no) Int                           /*（8.7.4）                * /
port_no;
{
    布尔根；

    根 = 根桥（）；

    成为指定端口（端口号）；                                    /*  (8.7.4一)              * /
                                                                /*  (8.6.10.2      b)      * /
    配置_更新（）；                                             /*  (8.7.4 b)              * /
                                                                /*  (8.6.7.2       b)      * /
    端口状态选择（）；                                          /*  (8.7.4三)              * /
                                                                /*  (8.6.11.2      b)      * /
```

```
    如果 ((root_bridge()) && (!root)) {                                    /*（8.7.4 d）          * /

        bridge_info.max_age =       桥梁信息.桥梁最大年龄；                 /*（8.7.4 d1）         * /
        bridge_info.hello_time        = 桥梁信息.桥梁问候时间；
        bridge_info.forward_delay          =桥接信息.桥接转发延迟；

        拓扑变化检测（）；                                                 /*  (8.7.4 d2)         * /
                                                                           /*  (8.6.14.2.4)       * /
        停止_tcn_定时器（）；                                              /*  (8.7.4 d3)         * /

        配置_bpdu_生成（）；                                               /*（8.7.4 d4）         * /

        启动_hello_timer（）；
    }
}

forward_delay_timer_expiry(port_no) int                                    /*（8.7.5）            * /
port_no;
{
    如果（port_info [port_no] .state == 正在监听）{                        /* (8.7.5一)           * /

        设置端口状态（port_no，学习）；                                    /*（8.7.5 a1）         * /

        启动转发延迟定时器（端口号）；                                     /* (8.7.5 a2)          * /
    }

    否则，如果（port_info [port_no] .state == 学习）{                      /* (8.7.5 b)           * /

        设置端口状态（端口号，转发）；                                     /*（8.7.5 b1）         * /

        如果（designated_for_some_port（））{                              /* (8.7.5      b2）    * /

            如果 (port_info[port_no].change_detection_enabled == True)
                                                                           /*  (8.5.5.10)         * /
            {
                拓扑变化检测（）；                                         /*  (8.6.14.2.2)       * /
            }
        }
    }
}

        /* 在哪里 */

        布尔值    指定_for_some_port（）
        {
            整数 端口号；

            对于（port_no = 一；port_no <= 端口数；port_no++）{

                如果（port_info[port_no].designated_bridge
                    == bridge_info.bridge_id
                )
                {
                    返回（真）；
                }
            }

            返回（假）；
        }

tcn_timer_expiry()                                                         /*  (8.7.6)            * /
{
        传输_tcn（）；                                                     /*  (8.7.6     一个）  * /

        启动_tcn_timer（）；                                               /* (8.7.6 b)           * /
```

```
}

拓扑变化定时器到期()                                                    /*  (8.7.7)              * /

        bridge_info.topology_change_detected          =  错误的;        /*  (8.7.7     一个）    * /

        bridge_info.拓扑变化=False;                                     /* (8.7.7 b)             * /
}

hold_timer_expiry(port_no) int                                          /*  (8.7.8)              * /
port_no;
{
    如果（port_info [port_no] .config_pending）{

        传输配置（端口号）；                                            /*  (8.6.1.2)            * /
    }
}
/** 桥接协议实体的管理 (8.8) **/ 初始化()
                                                                        /*  (8.8.1)              * /
{
    int 端口号；

    bridge_info.designated_root          = bridge_info.bridge_id; = 零；  /* (8.8.1一)           * /
    bridge_info.root_path_cost
    bridge_info.root_port = No_port;

    bridge_info.max_age =          桥梁信息.桥梁最大年龄；              /* (8.8.1 b)            * /
    bridge_info.hello_time          =  桥梁信息.桥梁问候时间；
    bridge_info.forward_delay           =桥接信息.桥接转发延迟；

    bridge_info.topology_change_detected            =假；               /* (8.8.1 三)           * /
    bridge_info.topology_change = False;
    stop_tcn_timer();
    停止拓扑变化计时器（）；

    对于（port_no = One; port_no <= No_of_ports; port_no++）/*（8.8.1 d）{                    * /


        初始化端口（端口号）；

    }

    端口状态选择（）；                                                  /* (8.8.1 e)            * /

    配置_bpdu_generation（）；启动                                      /* (8.8.1 f)            * /
    _hello_timer（）；
}
                                                                                               /*

                                                                                               * /
    初始化端口（端口号） Int
    port_no；
    {

        成为指定端口（端口号）；                                        /*  (8.8.1     d1）      * /

        设置端口状态（端口号，阻塞）；                                  /*  (8.8.1     d2）      * /

        端口信息[端口号].拓扑变化确认 = False;
                                                                        /*  (8.8.1 d3)           * /

        端口信息[端口号].config_pending = False;                        /*  (8.8.1     d4）      * /

        port_info[port_no].change_detection_enabled          =  真的;   /*  (8.8.1     d8)       * /
```

```
        停止消息定时器（端口号）；                                  /*  (8.8.1    d5)     * /

        停止转发延迟定时器（端口号）；                              /*  (8.8.1    d6)     * /

        停止保持定时器（端口号）；                                  /* (8.8.1 d7)        * /
    }

启用端口（端口号）                                                  /*（8.8.2）           * /
int 端口号；
{

    初始化端口（端口号）；

    端口状态选择（）；                                              /* (8.8.2克)           * /
}
                                                                    /*

                                                                    * /

禁用端口（端口号）                                                  /*（8.8.3）           * /
int 端口号；
{
    布尔根；

    根 = 根桥（）；

    成为指定端口（端口号）；                                        /*  (8.8.3    一个）    * /

    设置端口状态（端口号，已禁用）；                                /*  (8.8.3    b)        * /

    port_info[port_no].拓扑变化确认 = False; /* (8.8.3 c)                                  * /

    端口信息[端口号].config_pending = False;                        /*  (8.8.3    d)        * /

    停止消息定时器（端口号）；                                      /*  (8.8.3    e)        * /

    停止转发延迟定时器（端口号）；                                  /*  (8.8.3    f)        * /

    配置_更新（）；                                                 /*  (8.8.3    七）      * /

    端口状态选择（）；                                              /*  (8.8.3    h)        * /

    如果 ((root_bridge()) && (!root)) {                             /*（8.8.3 我）          * /

        bridge_info.max_age =          桥梁信息.桥梁最大年龄；      /*（8.8.3 i1）          * /
        bridge_info.hello_time           =  桥梁信息.桥梁问候时间；
        bridge_info.forward_delay              =桥接信息.桥接转发延迟；

        拓扑变化检测（）；                                          /*  (8.8.3    i2）      * /

        停止_tcn_定时器（）；                                       /*  (8.8.3    i3）      * /

        配置_bpdu_生成（）；                                        /*  (8.8.3    i4）      * /

        启动_hello_timer（）；
    }
}

set_bridge_priority(new_bridge_id) 标识符                           /*  (8.8.4)             * /
                新桥ID；                                            /*  (8.8.4    一个）    * /
{
    布尔值     根;
    int 端口号；

    根 = 根桥（）；
```

```
    对于（port_no = One; port_no <= No_of_ports; port_no++）/*（8.8.4 b）{                          * /

        如果（指定端口（端口号））{

            港口信息[港口编号].指定桥 = 新桥 ID;
        }
    }

    bridge_info.bridge_id =          新桥ID;                        /*  (8.8.4   c)       * /

    配置_更新（）;                                                  /*  (8.8.4   d)       * /

    端口状态选择（）;                                               /*  (8.8.4   e)       * /

    如果 ((root_bridge()) && (!root)) {                             /* (8.8.4 f)          * /

        bridge_info.max_age =        桥梁信息.桥梁最大年龄;         /* （8.8.4 f1)       * /
        bridge_info.hello_time        =  桥梁信息.桥梁问候时间;
        bridge_info.forward_delay        =桥接信息.桥接转发延迟;

        拓扑变化检测（）;                                           /*  (8.8.4   f2)     * /

        停止_tcn_定时器（）;                                        /*  (8.8.4   f3)     * /

        配置_bpdu_生成（）;                                         /*  (8.8.4   f4)     * /

        启动_hello_timer（）;
    }
}

设置端口优先级（端口号，新端口号） int 端口号;                      /*  (8.8.5)           * /

端口ID新端口ID;                                                     /*  (8.8.5   一个)    * /
{
    如果（指定端口（端口号））{                                      /*  (8.8.5   b)       * /

        port_info[port_no].指定端口                =  新的端口号;
    }

    端口信息[端口号].端口id = 新端口id;                              /* (8.8.5 c)          * /

    如果（（   bridge_info.bridge_id                                 /*  (8.8.5   d)       * /
               == 港口信息[港口编号].指定桥梁
          )
          &&
          (  端口信息[端口号].端口ID
             < 港口信息[港口编号].指定港口
          )
        )
    {
        成为指定端口（端口号）;                                      /*  (8.8.5   d1)    * /

        端口状态选择（）;                                           /*（8.8.5 d2)        * /
    }
}

                                                                                      /*

                                                                                      * /
设置路径开销（端口号，         路径成本)                             /*  (8.8.6)           * /
整数  端口号;
成本    路径成本;
{
    端口信息[端口号].路径成本              =  路径成本;              /*  (8.8.6   一个)    * /
```

```
    配置_更新（）；                                    /* (8.8.6 b)          * /

    端口状态选择（）；                                  /* (8.8.6 c)          * /
}

                                                                              /*

                                                                              * /
enable_change_detection(port_no) Int                                          /*  (8.8.7)           * /
port_no;
{
    port_info[port_no].change_detection_enabled          = 真的;
}
                                                                              /*

                                                                              * /
disable_change_detection(port_no) Int                                         /* (8.8.8)            * /
port_no;
{
    port_info[port_no].change_detection_enabled          = 错误的;
}
/** 伪实现特定的计时器运行支持 **/

打钩（）
{
    整数 端口号；

    如果（hello_timer_expired（））
    {    hello_timer_expiry（）；
    }

    如果（tcn_timer_expired（））
    {    tcn_timer_expiry();
    }

    如果（拓扑变化定时器_过期（））拓扑变化定时器_
    {    过期（）；
    }

    对于（port_no = 一；port_no <= 端口数；port_no++）{

        如果 (message_age_timer_expired(port_no)) {

            消息年龄定时器到期（端口号）；
        }
    }

    对于（port_no = 一；port_no <= 端口数；port_no++）{

        如果（转发延迟定时器过期（端口号））
        {
            转发延迟定时器到期（端口号）；
        }
        如果（hold_timer_expired（端口号））
        {
            保持定时器到期（端口号）；
        }
    }
}

/* 在哪里 */

启动_hello_timer（）
{ hello_timer.value = (时间) 零;
```

```
    hello_timer.active = True;
}

停止_hello_timer（）
{   hello_timer.active          =  错误的;
}
布尔值      hello_timer_expired()
{   如果 (hello_timer.active && (++hello_timer.value >= bridge_info.hello_time)) {
        hello_timer.active = False; 返回
        (True);
    }
    返回（假）；
}

启动_tcn_timer()
{   tcn_timer.value = (时间) 零；
    tcn_timer.active = 真；
}

停止定时器()
{   tcn_timer.active          =  错误的;
}

布尔值      tcn_timer_expired()
{   如果 (tcn_timer.active && (++tcn_timer.value >= bridge_info.bridge_hello_time)) {
        tcn_timer.active = False; 返回
        （True）；
    }
    返回（假）；
}

启动拓扑变化定时器()
    topology_change_timer.value = (时间) 零;
    topology_change_timer.active = True;
}

停止拓扑变化定时器()
    topology_change_timer.active            =  错误的;
}

布尔值      topology_change_timer_expired()
{ 如果 （   topology_change_timer.active && (
                ++拓扑变化计时器.值
                > = bridge_info.topology_change_time
            )
        )
    {       topology_change_timer.active 返回      =假；
        （True）；
    }
    返回（假）；
}

启动消息定时器（port_no，Int                    信息年龄)
        端口号；
时间   消息年龄；
{   message_age_timer[port_no].value =              消息年龄；
    message_age_timer[port_no].active =               真的;
}
stop_message_age_timer(port_no) Int
        端口号；
{   message_age_timer[port_no].active             =  错误的;
}

Boolean message_age_timer_expired(port_no) Int
port_no;
```

```
{ 如果 （消息定时器[端口号].活动&&
          （++message_age_timer[port_no].value >= bridge_info.max_age）
    {  message_age_timer[port_no].active 返回（True假；

    }
      返回（假）；
}
start_forward_delay_timer(port_no) Int
    端口号；
{  转发延迟定时器[端口号].值 转发延迟定时器[端口 = 零;
   号].活动 = 真的;
}

stop_forward_delay_timer(port_no) Int
    端口号；
{  转发延迟定时器[端口号].活跃 = 错误的;
}

布尔值  forward_delay_timer_expired(端口号) 端口号；
整数
{  如果（forward_delay_timer[port_no].active &&
          （++forward_delay_timer[port_no].值 >= bridge_info.forward_delay）
    {  forward_delay_timer[port_no].active = 返回 错误的;
          （True）；
    }
    返回（假）；
}

start_hold_timer(port_no) Int
    端口号；
{  hold_timer[port_no].值 = 零;
   hold_timer[port_no].active = 真的;
}

stop_hold_timer(port_no) Int
    端口号；
{  hold_timer[port_no].active = 错误的;
}

布尔值  hold_timer_expired(port_no) 端口
整数    号；
{  如果（hold_timer[port_no].active &&
          （++hold_timer[port_no].值 >= bridge_info.hold_time）
    {  hold_timer[port_no].active = 返回 错误的;
          （True）；
    }
    返回（假）；
}


/** 伪实现特定传输例程 **/

#包括“transmit.c”
```

8.10 性能

本小节对桥接 LAN 中的网桥性能以及生成树算法和协议参数的设置提出了要求。这些对于确保算法和协议正确运行是必要的。

它建议使用性能参数的默认操作值。指定这些值是为了避免在操作之前设置值，并且选择这些值是为了最大限度地提高桥接 LAN 组件互操作的便利性。

它规定了性能参数的绝对最大值。规定了适用值的范围，以协助选择操作值并为实施者提供指导。

8.10.1 要求

为了正确运行，桥接 LAN 中的桥接器的参数和配置应确保

— 如果不需要，网桥不会启动重新配置。这意味着除非发生故障，否则网桥协议消息在其后继者到达之前不会超时。
— 重新配置后，新的活动拓扑上不会转发帧，而最初在前一个活动拓扑上转发的帧仍在桥接 LAN 中。这确保了帧不会重复。

通过限制

- 这**最大桥梁直径**桥接 LAN 数： 任意两个终端站连接点之间的桥接器的最大数量。

- 这**最大桥接传输延迟**：网桥接收和传输转发帧之间的最大时间，超过此限制的帧将被丢弃。

- 这**最大BPDU传输延迟**：在需要传输此类 BPDU 之后，传输桥接协议数据单元之前的最大延迟，如 8.7 中所述。
- 这**最大消息年龄增量估计过高**这可能与传输的 BPDU 中的消息年龄参数的值或存储的桥接协议消息信息的年龄有关。

— 价值观**桥接问候时间**，**桥梁最大年龄**，**桥接转发延迟**， 和**保持时间** 参数。

此外，桥梁不得

— 低估传输的 BPDU 中的消息年龄参数的增量。
— 低估前向延迟。
— 作为 Root 时，高估 Hello Time 间隔。

8.10.2 参数值

表 8-1 至表 8-5 指定了建议的默认值、绝对最大值和参数范围。

桥架的绝对最大值不得超过表 8-2 中规定的值。**最大桥接传输延迟**，**最大BPDU传输延迟**， 和**最大消息年龄增量估计过高**。

如果值**桥接问候时间**，**桥梁最大年龄**， 和**桥接转发延迟**可由管理层设置，桥接器应能够使用表 8-3 中指定的参数范围的全部值，粒度为 1 秒。

表 8-1—最大桥梁直径

范围	受到推崇的 价值
最大桥梁直径	7

表 8-2——中转和传输延迟

范围	受到推崇的 价值	绝对 最大限度
最大桥接传输延迟	1.0	4.0
最大BPDU传输延迟	1.0	4.0
最大消息年龄增量估计过高	1.0	4.0

所有时间均以秒为单位。

桥应使用**保持时间**如表8-3所示。

桥接器应强制执行以下关系：

2× (*桥接转发延迟*–1.0*秒*) >=*桥梁最大年龄*

桥梁最大年龄>=2× (*Bridge_Hello_ Time*+1.0*秒*)

建议使用默认值**路径成本**每个桥接端口的参数基于表 8-5 所示的值，这些值根据每个端口所连接的 LAN 段的速度进行选择。

注意：表 8-5 中所示的值适用于全双工和半双工操作。所示推荐值和范围旨在尽量减少需要管理路径成本的桥接器数量，以便控制桥接 LAN 的拓扑结构。

表 8-3—生成树算法计时器值

范围	推荐或 默认值	固定值	范围
桥接问候时间	2.0	—	1.0–10.0
桥梁最大年龄	20.0	—	6.0–40.0
桥接转发延迟	15.0	—	4.0–30.0
保持时间	—	1.0	—

所有时间均以秒为单位。
— 不适用。
子条款 8.10.2 限制了桥接最大年龄和桥接转发延迟之间的关系。

如果**桥梁优先权**和**端口优先级**对于每个端口，管理都可以设置，桥接器应能够使用表 8-4 中指定的参数范围的所有值，粒度为 1。

表 8-4 — 桥接和端口优先级参数值

范围	推荐或 默认值	范围
桥梁优先权	32,768	0–65 535
端口优先级	128	0–255

网桥不得使用比 8-5 中指定的绝对最小值更低的与任何端口相关的路径成本参数值。

如果值**路径成本**可由管理层设置，桥接器应能够使用表 8-5 中指定的参数范围的全部值，粒度为 1。

表 8-5 — 路径成本参数值

范围	链接速度	受到推崇的 价值	受到推崇的 范围	范围
路径成本	4兆位/秒	250	100–1000	1–65 535
路径成本	10 Mb/秒	100	50–600	1–65 535
路径成本	16 Mb/秒	62	40–400	1–65 535
路径成本	100 Mb/s	19	10–60	1–65 535
路径成本	1 Gb/秒	4	3–10	1–65 535
路径成本	10 Gb/s	2	1–5	1–65 535

9. 桥接协议数据单元（BPDU）的编码

本节指定了桥接协议实体之间交换的生成树协议的 BPDU 的结构和编码。

9.1 结构

9.1.1 八位字节的传输和表示

所有 BPDU 都应包含整数个八位字节。BPDU 中的八位字节从 1 开始编号，并按照它们放入数据链路服务数据单元 (DLSDU) 的顺序递增。八位字节中的位从 1 到 8 编号，其中 1 是低位。

当使用连续的八位字节来表示二进制数时，较低的八位字节数具有最高有效值。

所有桥接协议实体都遵守这些位和八位字节排序约定，从而允许进行通信。

9.1.2 组件

协议标识符编码在所有 BPDU 的初始八位字节中。本标准保留单个协议标识符值。本标准不对其他标准协议中具有不同协议标识符字段值的 BPDU 的结构、编码或使用（如果存在）做出进一步限制。

运行第 8 条中指定的生成树算法和协议的桥接协议实体所使用的 BPDU 使用保留的协议标识符值，并具有以下结构。

每个 BPDU 包含固定数量的参数，这些参数对于协议的运行必不可少。每个参数都以一个或多个八位字节进行编码，长度固定。BPDU 中的参数顺序是固定的。

每个 BPDU 的参数由 BPDU 类型决定；所有相同类型的 BPDU 应包含相同的参数、相同的顺序和以相同的方式编码。

9.2 参数类型的编码

9.2.1 协议标识符的编码

协议标识符应以两个八位字节进行编码。

9.2.2 协议版本标识符的编码

协议版本标识符应编码为一个八位字节。如果两个协议版本标识符被解释为无符号二进制数，则较大的数字将与最近定义的协议版本相关联。

9.2.3 BPDU 类型的编码

BPDU 的类型应编码为单个八位字节。八位字节中包含的位模式仅用于区分类型；不同类型的 BPDU 之间不存在任何顺序关系。

9.2.4 标志的编码

标志应编码为单个八位字节中的位。因此，可以在单个八位字节中编码多个标志。如果八位字节中的相应位取值为 1，则设置标志。八位字节中与为给定类型的 BPDU 定义的标志不对应的位位置将被重置，即应取值为 0。不会为给定协议版本和类型的 BPDU 定义其他标志。

9.2.5 桥标识符的编码

桥接标识符应编码为八个八位字节，表示无符号二进制数。两个桥接标识符可以进行数字比较，数字较小的表示优先级较高的桥接。

桥接标识符的两个最高有效八位字节组成一个可设置的优先级组件，允许管理桥接的相对优先级（8.5.3.7 和第 14 条）。六个最低有效八位字节确保桥接标识符的唯一性；它们应根据以下程序从全局唯一桥接地址（7.12.5）派生而来。

第三最高有效八位字节来自 MAC 地址的初始八位字节；八位字节的最低有效位（位 1）被分配桥接地址第一位的值，下一个最高有效位被分配桥接地址第二位的值，依此类推。在使用 48 位 MAC 地址的桥接 LAN 中，第四到第八个八位字节同样被分配桥接地址第二到第六个八位字节的值。

9.2.6 根路径成本的编码

根路径成本应编码为四个八位字节，表示无符号二进制数，任意成本单位的倍数。第 8.10.2 节包含有关根路径成本增量的建议，以便可以在此参数上放置一些通用值，而无需对桥接 LAN 中的桥进行管理安装实践。

9.2.7 端口标识符的编码

端口标识符应编码为两个八位字节，表示无符号二进制数。如果对两个端口标识符进行数字比较，则较小的数字应表示优先级较高的端口。端口标识符中较高的八位字节是可设置的优先级组件，允许管理同一桥上端口的相对优先级（8.5.5 和第 14 条）。较低的八位字节是端口号，以无符号二进制数表示。值 0 不用作端口号。

9.2.8 计时器值的编码

计时器值应编码为两个八位字节，表示无符号二进制数乘以 1/256 秒的时间单位。这样可以表示 0 到 256 秒（但不包括）范围内的时间。

9.3 BPDU格式及参数

9.3.1 配置BPDU

配置 BPDU 的格式如图 9-1 所示。每个传输的配置 BPDU 应包含以下参数（8.5.1），不得包含其他参数：

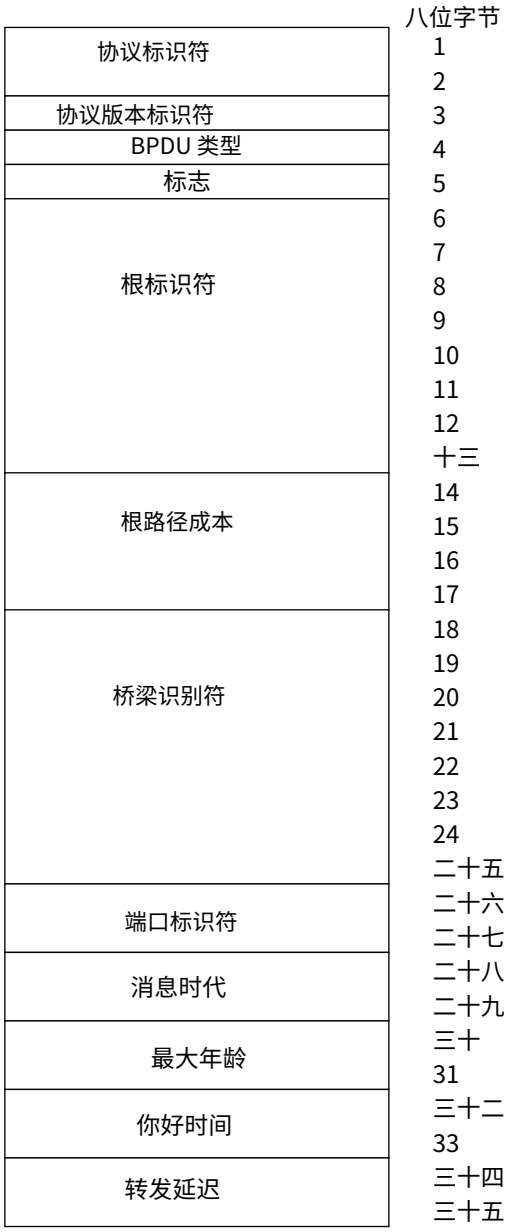


**图 9-1 — 配置 BPDU
参数及格式**

a) 协议标识符编码在 BPDU 的 1 和 2 个八位字节中。其值为 0000 0000 0000 0000，用于标识第 8 条中指定的生成树算法和协议。
b) 协议版本标识符编码在 BPDU 的八位字节 3 中。其值为 0000 0000。
c) BPDU 类型编码在 BPDU 的八位字节 4 中。此字段应取值 0000 0000。这表示配置 BPDU。

d) 拓扑改变确认标志被编码在 BPDU 的第 5 个八位字节的第 8 位中。
e) 拓扑改变标志被编码在 BPDU 的第 5 个八位字节的第 1 位中。
f) 根标识符编码在 BPDU 的第 6 个八位字节至第 13 个八位字节中。
g) 根路径成本在 BPDU 的第 14 到 17 个八位字节中编码。
h) 桥接标识符被编码在 BPDU 的第 18 个八位字节到第 25 个八位字节中。

i） 端口标识符编码在 BPDU 的第 26 和 27 个八位字节中。

j) 消息年龄计时器值编码在 BPDU 的第 28 和 29 个八位字节中。

k) 最大年龄计时器的值被编码在 BPDU 的第 30 个八位字节和 31 个八位字节中。

l) Hello Time 计时器值编码在 BPDU 的第 32 个八位字节和 33 个八位字节中。

m) 转发延迟计时器值编码在 BPDU 的第 34 个八位字节和 35 个八位字节中。

消息年龄（八位字节 28 和 29）应小于最大年龄（八位字节 30 和 31）。

9.3.2 拓扑变化通知 BPDU

拓扑改变通知 BPDU 的格式如图 9-2 所示。每个传输的拓扑改变通知 BPDU 应包含以下参数（8.5.2），不得包含其他参数：

a) 协议标识符编码在BPDU的第1和第2个八位字节中。其值为0000 0000 0000 0000，用于标识本标准第8条规定的生成树算法和协议。

b) 协议版本标识符编码在 BPDU 的八位字节 3 中。其值为 0000 0000。

c) BPDU 类型编码在 BPDU 的八位字节 4 中。此字段应取值 1000 0000（其中第 8 位显示在序列的左侧）。这表示拓扑更改通知 BPDU。

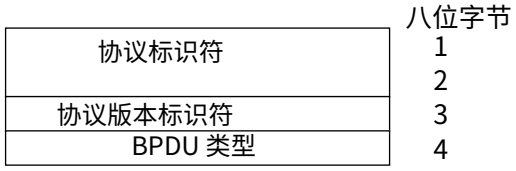


图 9-2—拓扑变化通知 BPDU 参数和格式

9.3.3 收到的 BPDU 的验证

当且仅当 BPDU 至少包含四个八位字节，且协议标识符具有为 BPDU 指定的值（9.3.2）时，桥接协议实体才应按照 8.7 中的规定处理接收到的 BPDU，并且

a) BPDU 类型表示配置 BPDU，且该 BPDU 至少包含 35 个八位字节，并且 BPDU 消息年龄参数的值小于其最大年龄参数的值；或者

b) BPDU 类型表示拓扑改变通知 BPDU。

在情况 a) 中，就根据本标准进行处理而言，超出八位字节 35 的任何八位字节都将被忽略。同样，在情况 b) 中，超出八位字节 4 的任何八位字节都将被忽略。

注——为了将来有可能对生成树协议进行扩展规范，并且通过协议版本标识符的不同值来标识新版本，因此在接收时不检查协议版本标识符。

10. GARP 多播注册协议（GMRP）

10.1 目的

GARP（通用属性注册协议）多播注册协议 (GMRP) 提供了一种机制，允许网桥和终端站动态地向连接到同一 LAN 段的 MAC 网桥注册（随后取消注册）组成员身份信息，并将该信息传播到桥接 LAN 中支持扩展过滤服务的所有网桥。GMRP 的运行依赖于第 12 条中定义的 GARP 提供的服务。

通过GMRP注册、注销和传播的信息形式如下：

一个)*群组成员信息*。这表示存在一个或多个属于特定组（或多个组）的 GMRP 参与者，并携带与该组关联的组 MAC 地址。特定组成员信息的交换可能导致在过滤数据库中创建或更新组注册条目，以指示已注册组成员的端口。这些条目的结构在 7.9.3 中描述；

b) *团体服务需求信息*. 这表示一个或多个 GMRP 参与者要求转发所有组或转发未注册组作为默认组过滤行为（参见 6.6.7 和 7.9.4）。

组成员信息注册可使桥接 LAN 中的网桥知道，发往相关组 MAC 地址的帧只能向已注册的组成员转发。因此，发往与该组关联的地址的帧只能在已收到此类成员注册的端口上转发。

通过注册组服务要求信息，桥接 LAN 中的网桥可以意识到，它们的任何可以按照收到组服务要求信息的方向转发帧的端口都应该根据所表达的组服务要求修改它们的默认组转发行为。

注意 — 这种动态操纵默认组转发行为的能力允许桥接端口考虑桥接 LAN 中不了解 GMRP 的设备对转发发往未注册组 MAC 地址的帧的需求。

GMRP 的运行可能导致

c) 在桥接 LAN 上传播组成员信息和组服务要求信息，并随之创建、更新或删除桥接 LAN 中所有支持扩展过滤服务的桥接器的过滤数据库中的组注册表项；

d) 此类网桥的群组过滤行为随之发生改变。

10.2 运作模式

GMRP 定义了一个*GARP 应用程序*（12.3、12.3.1 和 10.3）提供 6.6.5 和 6.6.7 中定义的扩展过滤服务。为此，GMRP 利用

a) 提供的声明和传播服务*GARP 信息分发*（GID；12.3 和 12.3.2）和*GARP 信息传播*（GIP；12.3 和 12.3.3）在桥接 LAN 内声明和传播组成员信息和组服务需求信息；

b) GID（12.3 和 12.3.2）提供的注册服务允许在桥接 LAN 中声明的组成员身份信息和组服务要求信息控制参与设备针对发往组 MAC 地址的帧的帧过滤行为。

任何在特定网桥上注册（或取消注册）的新信息均由 GARP 通过桥接 LAN 传播到可能存在的其他网桥，方式如 12.2 中所述。此信息用于更新参与终端站和网桥的过滤数据库中的组注册表项 (7.9.3)。

10.2.1 群组成员信息的传播

转发过程使用过滤数据库中的组注册表项来确保帧仅通过必要的桥接端口传输，以便到达组成员所连接的 LAN。图 10-1 说明了 GMRP 为单个组创建的组注册表项。

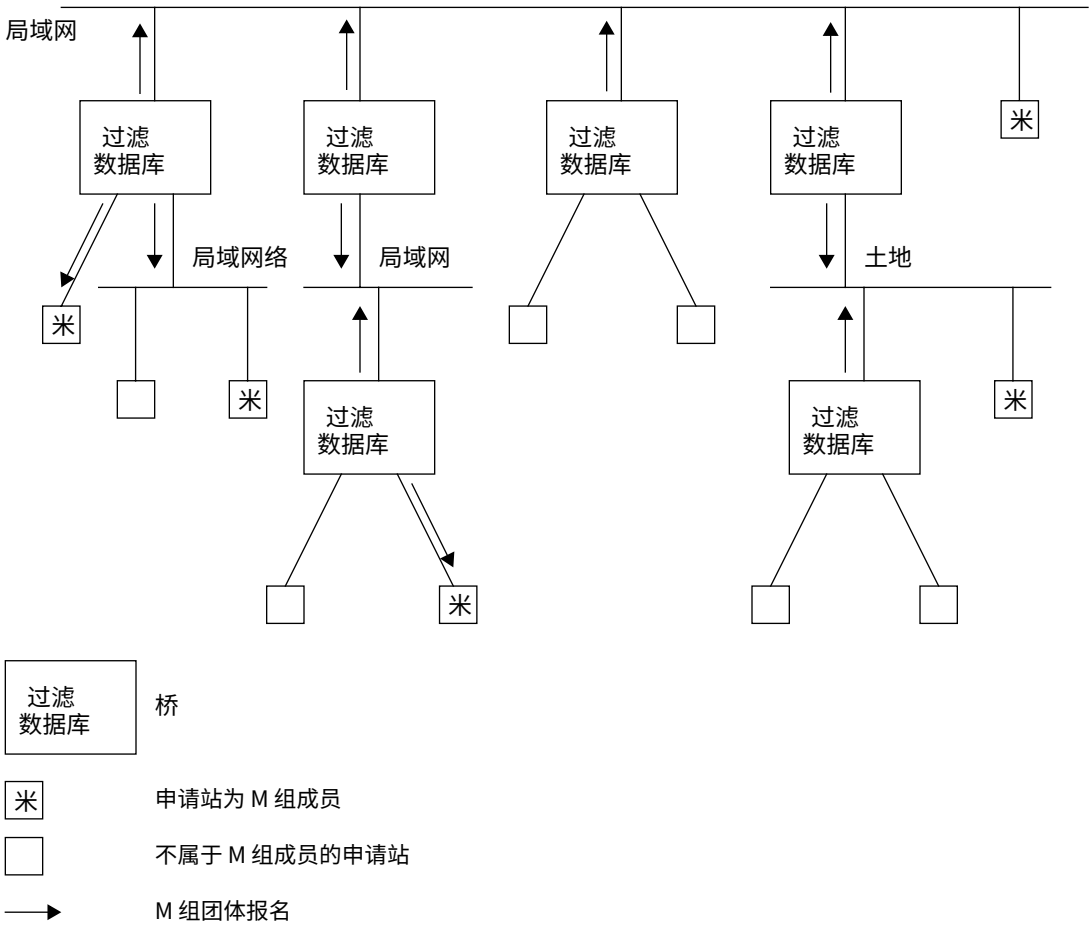


图 10-1—有向图示例

通过从所有端口接收帧并仅通过 GMRP 已创建组注册条目的端口转发，桥接有助于实现基于开放主机组概念的组分发机制。任何希望接收传输到特定组的帧的 GMRP 参与者 (12.3) 都通过请求相关组的成员身份来注册其这样做的意图。任何希望将帧发送到特定组的 MAC 服务用户都可以从桥接 LAN 的任何连接点执行此操作。这些帧可以在已注册的所有 LAN 段上接收

GMRP 参与者已连接，但桥接器对尚未由 GMRP 创建组注册条目的端口应用的过滤可确保帧不会在既未连接注册的 GMRP 参与者也不在帧源和注册组成员之间的活动拓扑路径中的 LAN 段上传输。因此，GMRP 和组注册条目会将帧限制在由生成树协议定义的整体无环路活动拓扑的修剪子集中，其中每个修剪子集仅包含从给定源到达组成员所需的 LAN 段。

注意 — 此处使用的术语“开放主机组”源自 IETF 定义的互联网组成员协议 (IGMP) 定义中引入的术语。

作为发往该组的 MAC 帧的来源的 MAC 服务用户不必自己注册为该组的成员，除非他们还希望接收由其他来源传输到该组地址的帧。

10.2.2 团体服务需求信息传播

GMRP 传播组服务需求信息的方式与传播组注册信息的方式相同。如果给定桥接器中的任何端口具有“所有组”或“所有未注册组”的注册组服务需求（以静态过滤条目和/或动态过滤条目中的控制信息表示，其中 MAC 地址规范为“所有组”或“所有未注册组”），则此事实将传播到桥接器的所有其他端口，从而导致该信息在相邻桥接器的端口上注册。作为该注册的结果，这些端口的默认组过滤行为可能会发生变化，以保持与注册信息所表达的服务需求的兼容性，如 7.9.4 中定义。这确保了在桥接 LAN 的不同区域的服务需求不同的 LAN 中可以保持连接。

注意：在桥接 LAN 中，所有“边缘”端口的默认组过滤行为并不相同，服务需求传播将倾向于导致所有“骨干”端口切换到桥接 LAN 中使用的最高优先级组过滤行为。优先级规则在 7.9.4 中定义。

10.2.3 源修剪

如 10.2.1 所述，GMRP 的操作定义了生成树的子树，这是在桥接 LAN 中网桥的过滤数据库中创建组注册表项的结果。桥接 LAN 中的终端站还能够利用通过 GMRP 注册的组成员身份信息，以便跟踪当前存在活跃成员的组集以及上游设备的服务要求。如果终端站的注册组成员身份和组服务要求信息表明，通过它们所连接的 LAN 段没有可到达的帧的有效接收者，则作为帧发往某个组的源的终端站可以抑制此类帧的传输。

注意：实际上，就帧传输而言，终端站可视为单端口桥，具有自己的默认组过滤行为和通过 GMRP 更新的“过滤数据库”条目，这些条目会告知终端站是否应将其生成的多播帧转发到连接的 LAN。为了实现这一点，终端站必须实现 GARP 的注册器和申请人功能，如 12.7.3、12.8.1 和 12.8.2 中所述。12.7.7 和 12.7.8 中所述的仅申请人和简单申请人参与者不包含源修剪所需的注册器功能。

这种终端系统行为被称为*源修剪。S*源修剪允许 MAC 服务用户（这些用户的 MAC 帧目的地为多个组，例如服务器站或路由器）避免在桥接 LAN 中没有当前组成员希望接收此类流量的情况下在其本地 LAN 段上出现不必要的流量泛滥。

10.2.4 终端站使用组服务要求注册

本标准主要将传播组服务要求信息的能力描述为传播网桥本身要求的一种方式。但是，这种机制也可以由具有涉及混杂接收某些方面的要求的终端站使用，例如路由器或网络监视器。需要能够接收所有多播流量的 GMRP 感知终端站可以通过声明其所连接的 LAN 段上所有组的成员身份来实现此目的；同样，希望接收未注册多播流量的终端站可以通过声明所有未注册组的成员身份来实现此目的。这些设施可能在哪些情况下使用将在 F.3 中进一步讨论。

10.3 GMRP 应用程序的定义

10.3.1 GARP 协议元素的定义

10.3.1.1 GMRP 对 GIP 上下文的使用

本标准中定义的 GMRP 在 12.3.4 中定义的基础生成树上下文中运行。应使用 GIP 上下文标识符值 0 来标识此上下文。在任何其他 GIP 上下文中使用 GMRP 不在本标准的范围内。

10.3.1.2 GMRP应用地址

发往 GMRP 参与者的 GARP PDU 的目标地址所使用的组 MAC 地址应为表 12-1 中标识的 GMRP 地址。

10.3.1.3 GMRP属性类型的编码

GMRP 的操作定义了两种在 GARP 协议交换中携带的属性类型（12.11.2.2），如下所示：

a) 组属性类型；
b) 服务需求属性类型。

组属性类型用于标识组 MAC 地址的值。GARP PDU（12.11.2.2）中携带的组属性类型的值应为 1。

服务需求属性类型用于标识组服务需求的值。GARP PDU（12.11.2.2）中携带的服务需求属性类型的值应为 2。

10.3.1.4 GMRP属性值的编码

组属性类型实例的值应编码为 GARP PDU（12.11.2.6）中的属性值，为六个八位字节，每个字节代表一个无符号二进制数。八位字节源自 48 位 MAC 地址的十六进制表示（定义在 IEEE Std 802 中），如下所示：

a) 十六进制表示法中的每个两位十六进制数字以正常方式表示一个无符号的十六进制值，即每个数字的最右边的数字代表该值的最低有效位，最左边的数字代表最高有效位。

b) 属性值编码的第一个八位字节来自 MAC 地址十六进制表示中最左边的十六进制值。八位字节的最低有效位（位 1）被分配为十六进制值的最低有效位，下一个最高有效位被分配为十六进制值的第二个有效位的值，依此类推。

c) 编码的第二到第六个八位字节同样被分配MAC地址的十六进制表示中的第二到第六个十六进制值。

此属性类型的值不得包括单独的 MAC 地址。

服务需求属性类型实例的值应编码为 GARP PDU（12.11.2.6）中的属性值，作为单个八位字节，用于表示无符号二进制数。此类型仅定义两个值：

a) 所有组均应编码为值0；
b) 所有未注册的组应编码为值1。

其余可能的值（2 到 255）未定义。

10.3.2 提供和支持扩展过滤服务

10.3.2.1 端系统注册和注销

GARP 参与者的 GMRP 应用元素提供 6.6.7.1 中定义的动态注册和注销服务，如下所示：

收到 REGISTER_GROUP_MEMBER 服务原语后，GMRP 参与者发出 GID_Join.request 服务原语。请求的 attribute_type 参数携带组属性类型 (10.3.1.3) 的值，attribute_value 参数携带 REGISTER_GROUP_MEMBER 原语中携带的 MAC_ADDRESS 参数的值。

收到 DEREGISTER_GROUP_MEMBER 服务原语后，GMRP 参与者发出 GID_Leave.request 服务原语。请求的 attribute_type 参数携带组属性类型 (10.3.1.3) 的值，attribute_value 参数携带 DEREGISTER_GROUP_MEMBER 原语中携带的 MAC_ADDRESS 参数的值。

收到 REGISTER_SERVICE_REQUIREMENT 服务原语后，GMRP 参与者发出 GID_Join.request 服务原语。请求的 attribute_type 参数携带服务需求属性类型 (10.3.1.3) 的值，attribute_value 参数携带 REGISTER_SERVICE_REQUIREMENT 原语中携带的 REQUIREMENT_SPECIFICATION 参数的值。

收到 DEREGISTER_SERVICE_REQUIREMENT 服务原语后，GMRP 参与者发出 GID_Leave.request 服务原语。请求的 attribute_type 参数携带服务需求属性类型 (10.3.1.3) 的值，attribute_value 参数携带 DEREGISTER_SERVICE_REQUIREMENT 原语中携带的 REQUIREMENT_SPECIFICATION 参数的值。

10.3.2.2 注册和注销事件

GARP 参与者的 GMRP 应用程序元素对 GID 发出的注册和注销事件做出如下响应：

收到 GID_Join.indication 后，其 attribute_type 等于组属性类型或服务需求属性类型 (10.3.1.3) 的值，GMRP 应用程序元素将更新过滤数据库中任何组注册条目 (7.9.3)，以获取 attribute_value 参数中携带的 MAC 地址规范，以指示相关 MAC 地址规范已在与 GMRP 参与者关联的端口上注册。这是通过在 GMRP 参与者中将相关端口指定为转发来实现的

条目的端口映射字段。如果过滤数据库中不存在这样的组注册条目，则创建一个新的组注册条目。

注意：组成员信息和组服务需求信息均通过组注册表项记录在过滤数据库中（参见 7.9.3）。对于组成员信息，组注册条目中的 MAC 地址规范为组 MAC 地址。对于组服务需求信息，MAC 地址规范为“所有组地址”或“所有未注册组地址”。

收到 GID_Leave.indication 后，其 attribute_type 等于组属性类型或服务需求属性类型 (10.3.1.3) 的值，GMRP 应用程序元素会更新过滤数据库中任何组注册条目 (7.9.3)，以了解 attribute_value 参数中携带的 MAC 地址规范，以指示相关 MAC 地址规范不再在与 GMRP 参与者关联的端口上注册。这可以通过在条目的端口映射字段中将相关端口指定为过滤来实现。如果过滤数据库中不存在此类过滤数据库条目，则将忽略该指示。如果将该端口设置为过滤导致端口映射中没有指定为转发的端口（即所有组成员都已取消注册），则将从过滤数据库中删除该组注册条目。

10.3.2.3 行政控制

通过与 GARP 操作 (12.9.1) 相关的注册管理控制参数，可以对与 GMRP 应用程序相关的状态机的注册状态进行静态控制。这些参数在过滤数据库中由静态过滤条目 (7.9.1) 中保存的信息表示。如果给定 MAC 地址规范不存在静态过滤条目，则相应属性值的注册管理控制参数的值为桥接器所有端口的正常注册。

永久数据库的初始状态（即，尚未通过管理操作配置的网桥中的永久数据库的状态）包括一个静态过滤条目，其 MAC 地址规范为所有组，其中端口映射指示注册已修复。此静态过滤条目将确定网桥所有端口的默认组过滤行为为转发所有组。永久数据库中的此条目可能会被管理操作删除或更新。

注意——初始静态过滤条目的规范意味着使用转发未注册组或过滤未注册组的操作需要网络管理员或管理员有意识地采取行动。

在实施管理能力的地方，可以通过 14.7 中定义的管理功能来应用和修改注册服务机构管理控制参数。

通过与 GARP 操作 (12.9.2) 相关的申请人管理控制参数，可以对申请人状态机参与协议交换的能力进行静态控制。在实现管理功能的地方，可以通过 14.9 中定义的管理功能应用和修改申请人管理控制参数。

10.3.3 程序模型

第 11 和 13 节包含 GMRP 和 GARP 的示例“C”代码描述。

10.4 符合 GMRP

本小节定义了声称符合 GMRP 的实现的一致性要求。涵盖两种情况：在 MAC 桥接器中实现 GMRP 和在终端站中实现 GMRP。虽然本标准主要涉及定义 MAC 桥接器的要求，但包括了终端站实现 GMRP 的一致性要求，以便提供有用的

此类实施的指导。附件 A 中定义的 PICS Proforma 仅与 MAC 桥接器的一致性声明有关。

10.4.1 MAC 桥接器对 GMRP 的符合性

声称符合 GMRP 的 MAC 桥接器应

a) 符合12.8.1、12.8.2、12.8.3定义的GARP申请者和注册者状态机的操作，以及LeaveAll生成机制；

b) 根据这些状态机的要求交换 GARP PDU，其格式符合 12.11 中描述的通用 PDU 格式，并且能够携带 10.3.1 中定义的应用特定信息，使用表 12-1 中定义的 GMRP 应用地址；

c) 按照 12.3.3 和 12.3.4 中定义的基础生成树上下文的 GIP 操作传播注册信息；

d) 实现10.3中定义的GMRP应用组件；
e) 按照7.12.3的要求转发、过滤或丢弃携带任何GARP应用程序地址作为目标MAC地址的MAC帧。

10.4.2 终端站符合 GMRP

声称符合 GMRP 的终端站应

a) 符合以下任一操作
   1) 申请人状态机，如12.8.1所定义；或
   2) 仅限申请人状态机，如 12.8.5 中定义；或
   3) 简单申请人状态机，定义见12.8.6；
b) 按照所实施的 GARP 状态机的要求交换 GARP PDU，按照 12.11 中描述的通用 PDU 格式进行格式化，并能够携带 10.3.1 中定义的应用特定信息，使用表 12-1 中定义的 GMRP 应用地址；

c) 支持提供10.3.2.1定义的终端系统注册和注销功能；
d) 按照7.12.3的要求丢弃携带任何GARP应用程序地址作为目标MAC地址的MAC帧。

声称符合申请人状态机（12.8.1）操作的终端站还应

e)符合12.8.2和12.8.3中定义的GARP Registrar状态机的操作和LeaveAll生成机制；

f) 支持提供10.3.2.2定义的组和组服务要求注册和注销服务；以及

g) 按照注册的组和组服务要求信息，过滤发往组 MAC 地址的传出帧，方式与 7.7.2 中描述的转发过程的过滤功能的操作一致。

建议只有那些需要具有执行源修剪（10.2.3）能力的终端站才符合申请人状态机（12.8.1）的操作。

由于 12.7.9 中所述的原因，建议不需要执行源修剪能力的终端站优先实现仅申请人状态机（12.8.5），而不是简单申请人状态机（12.8.6）。

注：仅实施 a) 2) 和 b) 至 d) 的终端站相当于仅申请人参与者 (12.7.7) 的描述；仅实施 a) 3) 和 b) 至 d) 的终端站相当于简单申请人参与者 (12.7.8) 的描述。此类终端站仅需要能够注册一个或多个组的成员资格，

并在稍后的某个时间点撤销该成员资格；因此，不需要支持注册器或离开所有状态机的操作。

实施 a) 1) 和 b) 至 g) 的终端站能够执行 10.2.3 中所述的“源修剪”，即抑制发送至当前无成员组的帧。因此，此类终端站需要支持完整的申请人状态机，以及注册器和全部离开状态机。

11. GMRP 的“C”代码实现示例

11.1 目的

本节包含 GMRP 应用程序的示例实现（第 10 条）。示例实现利用了 GARP 协议第 13 条中所示的示例实现和支持 GMRP 和其他使用 GARP 的应用程序所需的状态机制（第 12 条）。包含此“C”代码描述是为了演示 GMRP 的结构，并表明可以构建一个开销合理的低实现。实现的设计旨在最大限度地提高清晰度和通用性，而不是为了紧凑性。

示例实现如下节所示：

a) GMRP应用程序的头文件（11.2）；
b) GMRP应用程序代码（11.3）。

实现中应用程序相关（本条款）和应用程序无关（条款 13）方面的分离清楚地说明了使用相同的基本 GARP 状态机实现其他应用程序所涉及的内容。代码旨在通过内联注释实现自我文档化。

注——为清楚起见，示例实现假设对于给定的 MAC 地址，过滤数据库中只能存在一个静态过滤条目（参见 7.9.1）。

11.2 GMRP 应用程序特定的头文件

11.2.1 gmr.h

```
/* gmr.h       * /
# 如果定义   gmr_h__
# 定义       gmr_h__

# 包括         “garp.h”
# 包括         “gid.h”
# 包括         “gip.h”

/********************************************************** ****************************
 * GMR：GARP 多播注册应用程序：GARP 属性
 ********************************************************************************************
 * /

typedef enum {All_attributes，Legacy_attribute，Multicast_attribute}
                    属性类型；

typedef enum {Forward_all，Forward_unregistered} Legacy_control；

typedef enum {Number_of_legacy_controls = 1};

/********************************************************** ****************************
 * GMR：GARP 多播注册应用程序：创建、销毁
 ********************************************************************************************
 * /

外部布尔 gmr_create_gmr（int process_id，无符号 vlan_id，
                                        无效** gmr）；
    /*
     * 创建一个新的 GMR 实例，分配并初始化一个控制
     * 块，如果创建成功则返回 True 和指向此实例的指针。
     * 还创建 MCD (MultiCast 注册数据库) 实例和
     * GIP（控制信息传播）。
     *
     * 端口由系统创建并单独添加到 GMR（参见
     * 下面的 gmr_added_port() 和 gmr_removed_port()）。
     *
     * 操作系统提供的 process_id 将在后续调用中使用
     * 操作系统服务。系统本身确保时间
     * process_id 的范围，防止销毁的计时器尚未到期
     * 进程等。尽管许多进程会隐式地提供 process_id
     * 如果不是大多数系统，则在此实现中明确说明
     *清晰度。
     *
     * vlan_id 为 GMR 的这个实例提供了上下文。
     * vlan_id 0 表示基本 LAN，即
     * VLAN 发明之前的 802.1D。此假设可能受
     * 进一步讨论 802.1。
     * /

外部无效gmr_destroy_gmr（void *gmr）；
    /*
     * 销毁 GMR 实例，销毁并释放相关的
     * MCD 和 GIP 的实例，以及任何剩余的 GID 实例。
     * /

外部void gmr_added_port（void *my_gmr，int port_no）；
    /*
     * 系统已为该应用程序创建了新端 口，并将其添加到
     * GID 端口环。此功能应提供任何管理
     * 需要对传统控制或多播端口进行初始化
```

```
     * 过滤属性，例如可能存储在永久数据库中的属性
     * 专门用于端口或作为模板的一部分。
     *
     * 新创建的端口是“连接”的，用于 GARP 信息
     * 使用单独的函数 gip_connect_port() 进行传播。之前
     * 执行此操作后，系统应使用以下命令初始化新创建的端口
     * 任何针对特定多播值的永久数据库控制。
     *
     * 假设新端口在
     * 应用程序继续。连接规则未编码在
     * GMR。它们依赖于整个系统的中继连接，
     *可总结如下：
     *
     * 如果 (my_gmr->vlan_id == Lan)
     * {
     *          如果 stp_forwarding(port_no) gmr_connect_port(port_no);
     * }
     * 否则，如果 vlan_forwarding(vlan_id，port_no)
     * {
     *          gmr_connect_port（端口号）；
     * }
     *
     * 当系统继续运行时，它应该调用 gmr_disconnect_port()
     * 和 gmr_connect_port() 根据需要维持所需的连接。
     * /

外部无效gmr_removed_port（void *my_gmr，int port_no）；
    /*
     * 系统已删除并销毁 GID 端口。该函数应该
     * 提供所需的任何特定于应用程序的清理。
     * /

/************************************************************ ****************************
 * GMR：GARP 多播注册申请：加入、离开指示
 ********************************************************************************************
 * /

外部无效 gmr_join_indication（无效* my_gmr，无效* my_port，
                                        无符号加入_gid_index）；
    /*
     * 处理传统属性和多播属性的连接。
     * 前者由前几个 GID 索引表示，并产生
     * 分为三种情况：
     *
     * 1. 目前未设置“转发所有”或“转发未注册用户”（即
     * 已注册此端口），因此过滤数据库位于
     * “filter_by_default” 模式，并且只有正在被
     * 通过此端口转发（出）的是那些“在此注册”的。
     * 此外，过滤数据库中可能还有其他条目
     * 其对该端口的影响目前被
     * “filter_by_default”：如果存在任何多播条目
     * 端口，过滤数据库的模型要求其过滤器或
     * 转发行为明确表示在所有其他端口上
     * - 没有每个端口设置，这意味着“表现为默认”模式。
     *
     * 2. 目前已设置“转发未注册用户”，但未设置“转发全部用户”。
     * 过滤数据库处于“默认转发”模式。如果过滤
     * 已经为任何端口的多播创建了数据库条目，它指定
     * 如果多播未在此处注册，则过滤此端口，但
     * 已为该端口所连接的端口注册（在 GIP
     * 意义），否则转发（即在此注册的多播或
     * 未注册此端口所连接的任何端口）。
     *
     * 3. 当前已设置全部转发，且优先转发
     * 未注册可能设置也可能不设置。过滤数据库位于
```

```
     *        “forward_by_default” 模式。如果已为任何端口创建过滤数据库条目，则它指定此端口的转
     *       发。并非所有为此端口注册的多播都已输入到过滤数据库中。
     *
     *
     *
     * 如果对给定端口进行“fdb_filter”或“fdb_forward”调用，并且
     * 多播地址，导致创建一个过滤数据库条目，
     * 其他端口设置为根据以下条件过滤或转发该地址
     * 当“forward_by_default”或“filter_by_default”设置为当前时
     * 调用成功。此行为可避免临时过滤
     * 故障。
     *
     * 此函数 gmr_join_indication() 可更改过滤数据库条目
     * 仅用于引起该指示的端口。如果另一个端口
     * 处于传统模式 B（该端口的 Forward_unregistered 设置（已注册），
     * 但不是 Forward_all)，则在此注册多播地址
     * 端口会导致它在另一个端口上被过滤。这由
     * gmr_join_propagated() 适用于可能受影响的其他端口。它将
     * 被称为新注册的 GIP 传播的结果
     * 属性（多播地址）。
     * /

外部无效 gmr_join_propagated（无效* my_gmr，无效* my_port，
                                        无符号加入_gid_index）；
    /*
     *
     * /

外部无效 gmr_leave_indication（无效* my_gmr，无效* my_port，
                                         无符号离开_gid_index）；
    /*
     *
     * /

外部无效 gmr_leave_propagated（无效* my_gmr，无效* my_port，
                                         无符号离开_gid_index）；
    /*
     *
     * /

/********************************************************** ****************************
 * GMR：GARP 多播注册应用程序：协议和管理事件
 ********************************************************************************
 * /

外部void gmr_rcv（void *my_gmr，void *my_port，void *pdu）；
    /*
     * 处理此 GMR 实例的整个接收 pdu。
     * /

外部void gmr_tx（void *my_gmr，void *my_port）；
    /*
     * 为 GMR 的这个实例传输一个 pdu。
     * /

# 结束语/* gmr_h__ */
```

11.2.2 gmd.h

```
/* gmd.h     * /
# 如果定义   gmd_h__
＃ 定义     gmd_h__
```

```
#包括“sys.h”

/********************************************************** ****************************
 * GMD： GARP 多播注册应用数据库
 ******************************************************************************************
 * /

外部布尔gmd_create_gmd（无符号max_multicasts，void**gmd）；
    /*
     * 创建一个新的 GMD 实例，分配最多 max_multicasts 的空间
     * MAC 地址。
     *
     * 如果创建成功则返回 True，同时返回指向
     * GMD 信息。
     * /

外部无效gmd_destroy_gmd（void *gmd）；
    /*
     * 销毁 gmd 实例，释放先前分配的数据库和
     * 控制空间。
     * /


外部布尔 gmd_find_entry（void *my_gmd，Mac_address 键，
                                        无符号*在索引处找到）；

外部布尔 gmd_create_entry(void *my_gmd, Mac_address 键，
                                        无符号*created_at_index);

外部布尔 gmd_delete_entry（void *my_gmd，
                                        无符号 delete_at_index);

外部布尔gmd_get_key（                    void *my_gmd，无符号索引，Mac_address *key);

# 结束语/* gmd_h__ */
```

11.2.3 gmf.h

```
/* gmf.h      * /
# 如果定义   gmf_h__
# 定义       gmf_h__

# 包括       “系统.h”
# 包括       “prw.h”
# 包括       “GMR.h”

/********************************************************** ****************************
 * GMF：GARP 多播注册应用程序 PDU 格式
 ******************************************************************************************
 * /

类型定义结构
{/*
     * 此数据结构保存了解析 GMR 所需的临时状态
     * 特别是 PDU。GPDU 为 GARP 应用提供了通用基础
     * 格式化程序；可以根据 GMF 的要求在这里添加附加状态。
     * /

     GPDU GPDU;

} GMF;
```

```
typedef struct /* Gmf_msg_data */ {
    Attribute_type属性;
    事件            事件;
    Mac 地址        键1;
    Mac 地址        键2;
    Legacy_control  遗留控制;
} Gmf_消息;
外部无效          gmf_rdmsg_init(Pdu *pdu, Gmf **gmf);
外部无效          gmf_wrmsg_init(Gmf *gmf, Pdu *pdu,          整数 虚拟局域网标识）;
外部布尔 gmf_rdmsg(                Gmf *gmf，Gmf_msg *msg）;
外部布尔 gmf_wrmsg(                Gmf *gmf，Gmf_msg *msg）;
# 结束语/* gmf_h__ */
```

11.2.4 fdb.h

```
/* fdb.h       * /
# 如果定义   fdb_h__
＃ 定义     fdb_h__

#包括“sys.h”

/******************************************************* ****************************
 * FDB：过滤数据库接口
 ********************************************************************************************
 * /

外部无效fdb_filter（无符号vlan_id， int port_no， Mac_address 地址）;

外部无效fdb_forward（无符号vlan_id， int port_no， Mac_address 地址）;

外部无效fdb_filter_by_default（无符号vlan_id，int port_no）;

外部无效fdb_forward_by_default（无符号vlan_id，int port_no）;

# 结束语 /* fdb_h__ */
```

11.3 GMRP应用程序代码

11.3.1 gmr.c

```
/* gmr.c */

# 包括 “gmr.h”
# 包括 “gid.h”
# 包括 “gip.h”
# 包括 “garp.h”
# 包括 “gmd.h”
# 包括 “gmf.h”
# 包括 “fdb.h”


/************************************************** ****************************
 * GMR：GARP 多播注册应用程序：实施规模
 ********************************************************************************
 * /

枚举{Max_multicasts = 100};

枚举{Number_of_gid_machines = Number_of_legacy_controls + Max_multicasts};

枚举 {Unused_index = Number_of_gid_machines};

/************************************************** ****************************
 * GMR：GARP 多播注册应用程序：创建、销毁
 ********************************************************************************
 * /

typedef struct /* Gmr */ {

    卡普 克；

    未签名 虚拟局域网标识；

    空白 *国土安全部；

    未签名 gmd 条目的数量；

    未签名 最后一个使用的gmd+1；

} 巨人；


布尔值gmr_create_gmr（int process_id，无符号vlan_id，void **gmr）{/*

    * /
    Gmr *我的_gmr；

    如果（！ sysmalloc（sizeof（Gmr），＆my_gmr））
        转到gmr_creation_failure；

    my_gmr->g.进程ID = 进程ID；
    my_gmr->g.gid = 无效的;

    如果（！ gip_create_gip（Number_of_gid_machines，＆my_gmr->g.gip））
        转到gip_creation_failure；

    my_gmr->g.max_gid_index = Number_of_gid_machines - 1; my_gmr-
    >g.last_gid_used = Number_of_legacy_controls - 1;

    my_gmr->g.join_indication_fn = gmr_join_indication;
```

```
        my_gmr->g.leave_indication_fn       = gmr_leave_indication;
        my_gmr->g.join_propagated_fn        = gmr_join_propagated;
        my_gmr->g.leave_propagated_fn       = gmr_leave_propagated;
        my_gmr->g.transmit_fn               = gmr_tx;
        my_gmr->g.added_port_fn             = gmr_添加的端口;
        my_gmr->g.removed_port_fn           = gmr_removed_port;

        my_gmr->vlan_id = vlan_id;

        如果（! gmd_create_gmd（Max_multicasts，&my_gmr->gmd））
            转到gmd_creation_failure;

        my_gmr->number_of_gmd_entries       = 最大多播数;
        my_gmr->last_gmd_used_plus1         = 0;

        *gmr = 我的_gmr;        返回（真）;
gmd_creation_failure:           gip_destroy_gip(my_gmr->g.gip);
gip_creation_failure（创建失败）:  sysfree(my_gmr);
gmr_creation_failure（创建失败）:  返回（假）;
}


void gmr_destroy_gmr(Gmr *my_gmr) {

        Gid *我的端口;

        gmd_destroy_gmd(my_gmr->gmd);
        gip_destroy_gip(my_gmr->g.gip);

        当（（my_port = my_gmr->g.gid）！ = NULL）
            gid_destroy_port(&my_gmr->g，my_port->port_no);
}


void gmr_added_port(Gmr *my_gmr，int port_no){/*

        * 提供任何传统控制或多播的管理初始化
        * 此处模板中的属性适用于新端口。
        * /
}


void gmr_removed_port(Gmr *my_gmr，int port_no){/*

        * 提供任何 GMR 特定的清理或管理警报功能
        * 删除端口。
        * /
}


/****************************************************** ****************************
  * GMR：GARP 多播注册申请：加入、离开指示
  ********************************************************************************
  * /

void gmr_join_indication(Gmr *my_gmr，Gid *my_port，unsigned join_gid_index) {/*

        * gmr_join_indication() 的实现尊重这三种情况
        * 在头文件中描述了过滤数据库的状态和
        * 注册的旧版控件（转发所有、转发未注册的）和
        * 多播。它做了一些优化尝试，但并不完美。
        * 当一个传统模式转换为
        * 其他。
        *
        * /
```

```
    未签名              gmd_指数；
    未签名              gid_索引；
    Mac 地址            钥匙;

    如果（！gid_registered_here（my_port，Forward_all））{

        如果 （   （加入_gid_index == 转发_全部）
            ||（joining_gid_index == Forward_unregistered））
        {
            gmd_index = 0; gid_index = gmd_index + Number_of_legacy_controls; 当 (gmd_index <
            my_gmr->last_gmd_used_plus1)
            {
                如果（！gid_registered_here（my_port，gid_index））{

                    如果 (joining_gid_index == Forward_all) {

                        gmd_get_key(my_gmr->gmd，gmd_index，&key);
                        fdb_forward(my_gmr->vlan_id，my_port->port_no，key);
                    }
                    否则，如果（！gip_propagates_to（my_port，gid_index））{
                        /*(joining_gid_index == Forward_unregistered) */
                        gmd_get_key(my_gmr->gmd，gmd_index，&key);
                        fdb_forward(my_gmr->vlan_id，my_port->port_no，key);
                    }
                }
                gmd_索引++；gid_索引++;
            }

            fdb_forward_by_default（my_gmr->vlan_id，my_port->port_no）；
        }
        else /* 多播属性 */ {

            gmd_index = join_gid_index-Number_of_legacy_controls;
            gmd_get_key(my_gmr->gmd，gmd_index，&key);

            fdb_forward(my_gmr->vlan_id，my_port->port_no，键);
}   }   }


void gmr_join_propagated(Gmr *my_gmr，Gid *my_port，无符号joining_gid_index){/*

    *
    * /
    未签名              gmd_指数；
    Mac 地址            钥匙;

    if (joining_gid_index >= Number_of_legacy_controls) {/* 多播属性 */


        如果 （   （！gid_registered_here（我的端口，       Forward_all））
            &&   （gid_registered_here（我的端口，（！   Forward_unregistered））
            && gid_registered_here（我的端口，            加入_gid_index））））
        {
            gmd_index = join_gid_index-Number_of_legacy_controls;
            gmd_get_key(my_gmr->gmd，gmd_index，&key);

            fdb_filter（my_gmr->vlan_id，my_port->port_no，键）；
}   }   }


void gmr_leave_indication(Gmr *my_gmr，Gid *my_port，无符号leaving_gid_index){/*

    *
    * /
    未签名              gmd_指数；
```

```
    未签名              gid_索引；
    布尔值      模式a；
    布尔值              模式_c；
    Mac 地址            钥匙;

    gid_registered_here(my_port，Forward_all); mode_c = !
    gid_registered_here(my_port，Forward_unregistered);

    如果 （           （leaving_gid_index == Forward_all） （!
        ||     （     mode_a）
               && (leaving_gid_index == Forward_unregistered)))
    {
        gmd_index = 0; gid_index = gmd_index + Number_of_legacy_controls; 当 (gmd_index <
        my_gmr->last_gmd_used_plus1)
        {
            如果（! gid_registered_here（my_port，gid_index））
            {
                如果 (mode_c || gip_propagates_to(my_port，gid_index)) {

                    gmd_get_key(my_gmr->gmd，gmd_index，&key); fdb_filter(my_gmr-
                    >vlan_id，my_port->port_no，key);
            }     }

            gmd_索引++；gid_索引++；
        }

        如果（mode_c）
            fdb_filter_by_default（my_gmr->vlan_id，              我的端口->端口号）；
    }
    否则，如果（! mode_a）
    {
        if (mode_c || gip_propagates_to(my_port, gid_index)) { /* 多播属性 */

            gmd_index = 离开_gid_index - 遗留控制数量；

            gmd_get_key(my_gmr->gmd，gmd_index，&key); fdb_filter(my_gmr-
            >vlan_id，my_port->port_no，key);
}     }     }


void gmr_leave_propagated(Gmr *my_gmr，Gid *my_port，unsigned leaves_gid_index) { /*

    *
    * /
    未签名              gmd_指数；
    Mac 地址            钥匙;

    if (leaving_gid_index >= Number_of_legacy_controls) {/* 多播属性 */


        如果 （    （! gid_registered_here（我的端口，        Forward_all) )
            &&    （gid_registered_here（我的端口，（!    Forward_unregistered))
            && gid_registered_here（我的端口，              离开_gid_index））)
        {
            gmd_index = 离开_gid_index - Number_of_legacy_controls;
            gmd_get_key(my_gmr->gmd，gmd_index，&key);

            fdb_forward(my_gmr->vlan_id，my_port->port_no，键);
}     }     }



/****************************************************** ****************************
 * GMR：GARP 多播注册应用程序：接收消息处理
```

```
  ********************************************************************************
  * /

静态 void gmr_db_full(Gmr *my_gmr，Gid *my_port) { /*

      * 如果希望能够正确操作尺寸较小的
      * 数据库，在此处添加代码。最好的方法似乎是使用 GID
      * 管理控件配置传统模式的属性
      * 控制 Forward_all 在所有加入的端口上进行固定注册
      * 消息已被丢弃，因为它们的密钥不在数据库中。
      * 然后启动重试计时器，尝试从
      * 稍后再尝试将数据从数据库中删除，如果成功，则等待几个 LeaveAll
      * 在将 Forward_all 切换回 Normal_registration 之前次数。
      * /
}

静态 void gmr_rcv_msg(Gmr *my_gmr，Gid *my_port，Gmf_msg *msg) { /*

      * 处理一条收到的消息。
      *
      * 根据消息事件和属性类型发送消息（传统模式
      * 控制或多播地址）除 LeaveAll 的情况外
      * 消息事件，它同样适用于所有属性。
      *
      * LeaveAll 消息不会引起任何指示（直接加入或离开）。
      * 即使对于点对点链路协议增强（其中
      * 普通 Leave 也是如此。这里不需要做进一步的工作。
      *
      * LeaveAllRange 消息目前被视为 LeaveAll
      * （即，该范围被忽略）。
      *
      *所有剩余的消息都引用单个属性（即单个
      * 注册组地址）。尝试在 MCD 中查找匹配的条目
      * 数据库。如果找到，则将消息发送到将
      * 处理本地 GID 效果和 GIP 传播到其他端口。
      *
      * 如果没有找到条目，则可以丢弃 Leave 和 Empty 消息，但
      * JoinIn 和 JoinEmpty 消息需要进一步处理。首先，尝试
      * 使用可用空间创建新条目（在数据库中，
      * 对应于空闲的 GID 机器集）。如果失败，可以尝试
      * 用于从未使用或处于较低水平的机器组中恢复空间
      * 重要状态。最后，数据库被认为是满的，并且收到的
      * 消息被丢弃。
      *
      * 一旦（如果）找到条目，则 Leave、Empty、JoinIn 和 JoinEmpty
      * 全部提交给 GID (gid_rcv_msg())。
      *
      * JoinIn 和 JoinEmpty 可能会引发 Join 指示，然后这些指示会被传播
      * 由 GIP 提供。
      *
      * 在共享介质上，Leave 和 Empty 不会引起指示
      * 立即。然而，这个例程确实测试并传播
      * 保留指示，以便可以不加改变地使用点对点
      * 协议增强。
      *
      * /

      无符号 gmd_index = Unused_index; 无符号
      gid_index = Unused_index;

      如果 （  （msg->event == Gid_rcv_leaveall）||（msg->event
            == Gid_rcv_leaveall_range））
      {
            gid_rcv_leaveall（我的端口）；
      }
```

```
    别的
    {
        如果 (msg->attribute == Legacy_attribute) {

            gid_index = msg->legacy_control;
        }
        否则，如果 (!gmd_find_entry(my_gmr->gmd，msg->key1，&gmd_index)) {/* &&
        (msg->attribute == Multicast_attribute) */

            如果（    （msg->event == Gid_rcv_joinin）||（msg-
                >event == Gid_rcv_joinempty））
            {
                如果 (!gmd_create_entry(my_gmr->gmd，msg->key1，&gmd_index)) {

                    如果（gid_find_unused（＆my_gmr-> g，
                                        Number_of_legacy_controls，＆gid_index）
                    {
                        gmd_index = gid_index - Number_of_legacy_controls;
                        gmd_delete_entry(my_gmr->gmd，gmd_index); (void)
                        gmd_create_entry(my_gmr->gmd，msg->key1，
                                                &gmd_索引）；
                    }
                    别的
                        gmr_db_full（my_gmr，my_port）；
        }   }   }

        如果（gmd_index！=未使用索引）
            gid_index = gmd_index + Number_of_legacy_controls；

        如果 (gid_index != Unused_index)
            gid_rcv_msg（my_port，gid_index，msg->event）；
}   }


void gmr_rcv(Gmr *my_gmr，Gid *my_port，Pdu *pdu) {/*

    * 处理此 GMR 实例的整个接收 pdu：初始化
    * Gmf pdu 解析程序，并在消息持续期间读取和处理
    * 一次一个。
    * /
    基因转移   通用基因改造
    Gmf_msg   味精；

    gmf_rdmsg_init（＆gmf，pdu）；

    虽然（gmf_rdmsg（＆gmf，   &消息）
        gmr_rcv_msg（我的gmr，   我的端口，&msg);
}


/******************************************************* ****************************
 * GMR：GARP 多播注册应用程序：传输处理
 ********************************************************************************
 * /

静态 void gmr_tx_msg(Gmr *my_gmr，无符号 gid_index，Gmf_msg *msg) {

    无符号gmd_index；

    如果 (msg->event == Gid_tx_leaveall) {

        消息->属性                    =所有属性；
    }
    否则，如果（gid_index == Forward_all）{
```

```
            消息->属性                  = 遗留属性；
            msg->legacy_control         = 全部转发；
        }
        else /* Multicast_attribute 的索引 */ {

            msg->属性 = Multicast_attribute;

            gmd_index = gid_index - Number_of_legacy_controls;
            gmd_get_key(my_gmr->gmd，gmd_index，&msg->key1);
}       }


void gmr_tx(Gmr *my_gmr，Gid *my_port) {/*

        * 获取并准备一个用于传输的 pdu（如果没有），
        * 只需返回；如果还有更多内容需要传输，GID 将重新安排呼叫
        * 到这个函数。
        *
        * 从 GID 获取要传输的消息，并使用 Gmf 将它们打包到 pdu 中
        *（多播          pdu 格式化程序)。
        * /
        普杜            *协议数据单元;
        基因转移          通用基因改造
        Gmf_msg         味精；
        事件              tx_事件；
        未签名            gid_索引；


        如果 ((tx_event = gid_next_tx(my_port，&gid_index)) != Gid_null) {

            如果（syspdu_alloc（&pdu））{

                gmf_wrmsg_init(&gmf, pdu, my_gmr->vlan_id);

                做
                {
                    消息事件      = tx_事件；

                    gmr_tx_msg（my_gmr，gid_index，&msg）；

                    如果 (!gmf_wrmsg(&gmf，&msg)) {

                        gid_untx（我的端口）；
                        休息;
                    }
                } 当（（tx_event = gid_next_tx（my_port，          &gid_index))
                                ！=Gid_null）；

            syspdu_tx(pdu，my_port->port_no);}
}       }
```

12. 通用属性注册协议（GARP）

12.1 目的

GARP 提供通用属性传播功能，GARP 应用程序参与者（GARP 参与者）可使用该功能向桥接 LAN 内的其他 GARP 参与者注册和取消注册属性值。属性类型的定义、属性可携带的值以及注册时与这些值关联的语义特定于相关 GARP 应用程序的操作。

注：第 10 条定义了一个这样的应用程序，即 GMRP，它利用 GARP 允许注册两种属性类型：组 MAC 地址和组服务要求。这些属性类型的值用于动态控制桥接 LAN 内 GMRP 参与者的过滤行为。

12.2 GARP 操作概述

GARP 的操作允许特定 GARP 应用程序的参与者进行*声明，*或撤回声明，相对于*属性*价值观，以及这些声明（或撤回）是否导致*登记*（或者*注销登记*) 与桥接 LAN 中该应用程序的其他 GARP 参与者共享这些参数值。

对于 GARP 参与者已声明（或撤回声明）某一给定属性值的事实，将通过与该属性值的申请者状态机相关联的状态变量进行记录，该状态变量适用于做出声明的端口。

仅当端口接收到包含声明或撤销的 GARP PDU 时才会进行注册。端口上属性值的当前注册状态通过与该属性值的注册器状态机相关联的状态变量来记录。

仅当所有连接到与端口相同的 LAN 段的 GARP 参与者（与端口本身关联的参与者除外）撤回声明时，才会取消注册给定端口上的给定属性值。

在桥接端口上注册的参数值是*主动拓扑*（7.4）对于*GIP 背景*（12.3.4) 会作为声明通过属于活动拓扑的所有其他桥接端口进行传播。因此，给定的声明将传播到桥接 LAN 中存在相关应用程序的 GARP 参与者的所有设备，并且在每个桥接中，属性值将在活动拓扑中距离声明源“最近”的那些端口上注册。

注 1 — 注册可在任何端口上发生，无论端口的生成树状态如何；但是，注册信息的传播遵循活动拓扑。

注 2 — 除非另有说明，后面对 GARP 的描述均假设在 GIP 上下文 0 内运行，生成树由第 8 章定义的协议和程序的运行建立和维护。

图 12-1 说明了属性值在桥接 LAN 组件之间的注册和传播，这是由单个终端站声明该属性值而产生的。该图显示了哪些桥接端口也声明了属性值以传播该值。图中所示的桥接的所有端口都假定处于转发状态。可以看出，传播弧导致属性值传播到所有 LAN 段；但是，传播的方向性导致属性值仅在接收（而不是传输）传播信息的桥接端口上注册。

图 12-2 说明了属性值在桥接 LAN 组件之间的注册和传播，这是由于两个终端站在不同的 LAN 段上声明了相同的属性值。

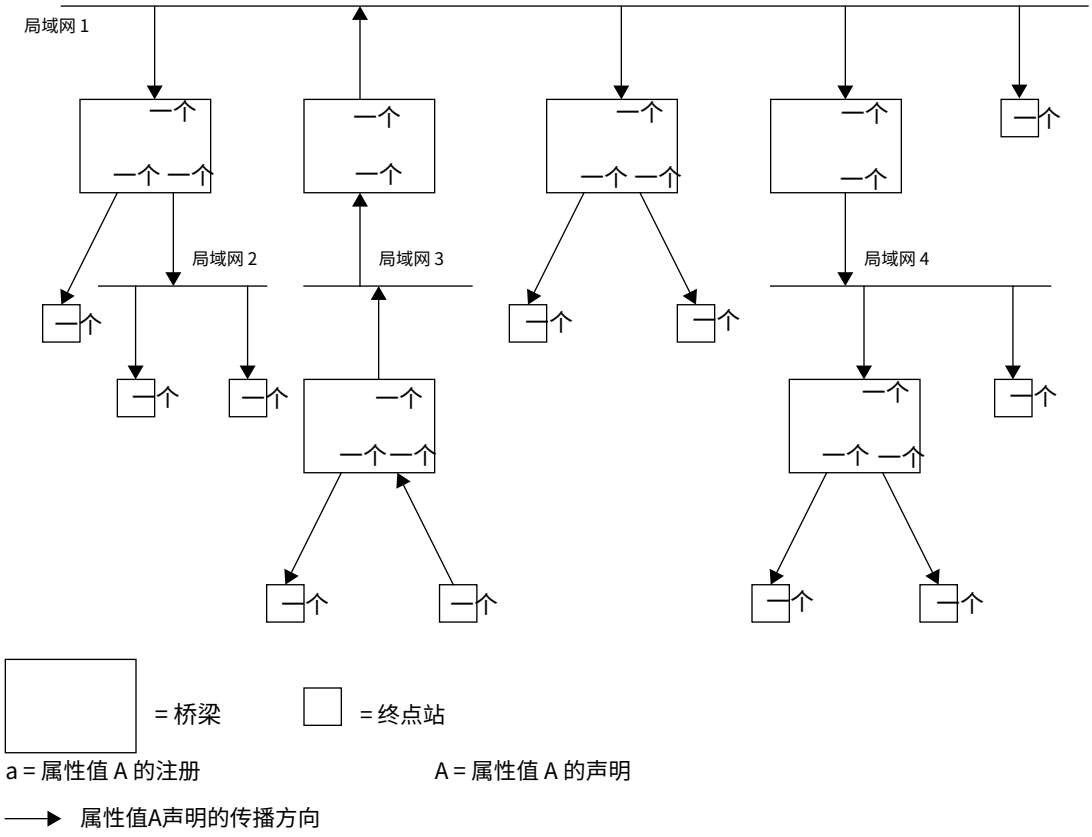


图 12-1 - 从一个站传播属性值的示例

从该图可以看出，增加第二个声明源会导致所有终端站都注册该属性值，并且一些网桥会在多个端口上注册该值。

这些属性值在桥接端口上的注册本质上是不对称的。在任何桥接中，

a) 已注册给定的属性值，并且
b）是主动拓扑的一部分，

完全定义活动生成树拓扑的子集，该子集包含已声明该属性值的所有 GARP 参与者。同样，对于任何终端站，已注册属性值的存在表明终端站所连接的生成树拓扑中的一个或多个 GARP 参与者已声明该值。

因此，已注册的属性值可视为从 GARP 参与者指向活动拓扑子树的指针，该子树包含一个或多个已声明该属性值的 GARP 参与者。因此，给定参与者中的此类注册集可表示活动拓扑子树，该子树包含所有已声明该属性值的 GARP 参与者。

因此，桥接 LAN 中给定属性值的注册集可视为形成一组子树，每个子树从给定的 GARP 参与者指示活动拓扑的子集，其中可以找到已声明该属性值的所有 GARP 参与者。在图 12-3 中，基于图 12-2 中显示的注册，形成此子树集的端口显示为箭头的原点。

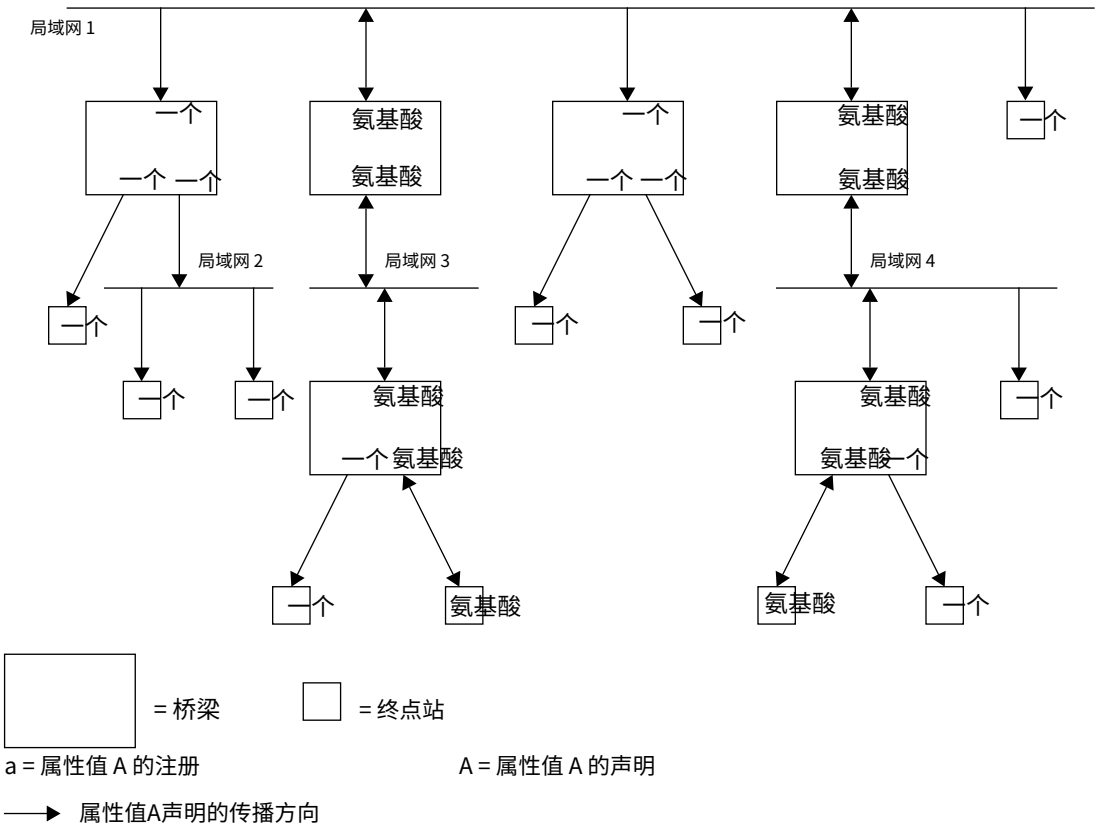


图 12-2——两个站的属性值传播示例

GARP 的这一属性表明了 GARP 最适合于哪些类型的应用。需要形成“可达性”树（构成包含所有已注册参与者的活动拓扑子集）的应用通常适合使用 GARP。例如，如果图 12-3 中的属性值 A 是 MAC 地址，其语义为“我希望收到超级碗决赛得分的详细信息”，并且认为最好将这些结果仅发送到包含已声明属性 A 的终端站的活动拓扑子集，则具有这些结果的终端站可以使用其端口上是否存在 A 注册来指示是否在其所连接的 LAN 段上发送结果，并且任何接收结果的网桥都可以确定应在哪些端口上转发结果。

在 MAC 桥接器中，GARP 仅在桥接器过滤模式设置为扩展过滤模式时运行。无法在扩展过滤模式下运行或已设置为在基本过滤模式下运行的桥接器对于 GARP 协议交换是透明的，并在所有处于转发状态的端口上转发 GARP PDU。同样，未实现特定 GARP 应用程序的桥接器对于发往该应用程序的 GARP 协议交换是透明的。

12.3 GARP 体系结构

图 12-4 说明了双端口桥接器和终端站中的 GARP 参与者的组件。

一个*GARP 参与者*在桥梁或终端站中，由*GARP 应用程序*组件，以及*GARP 信息声明 (GID)*与桥接器的每个端口相关联的组件。每个端口、每个 GARP 应用程序都有一个这样的 GARP 参与者。GARP 参与者之间的信息传播

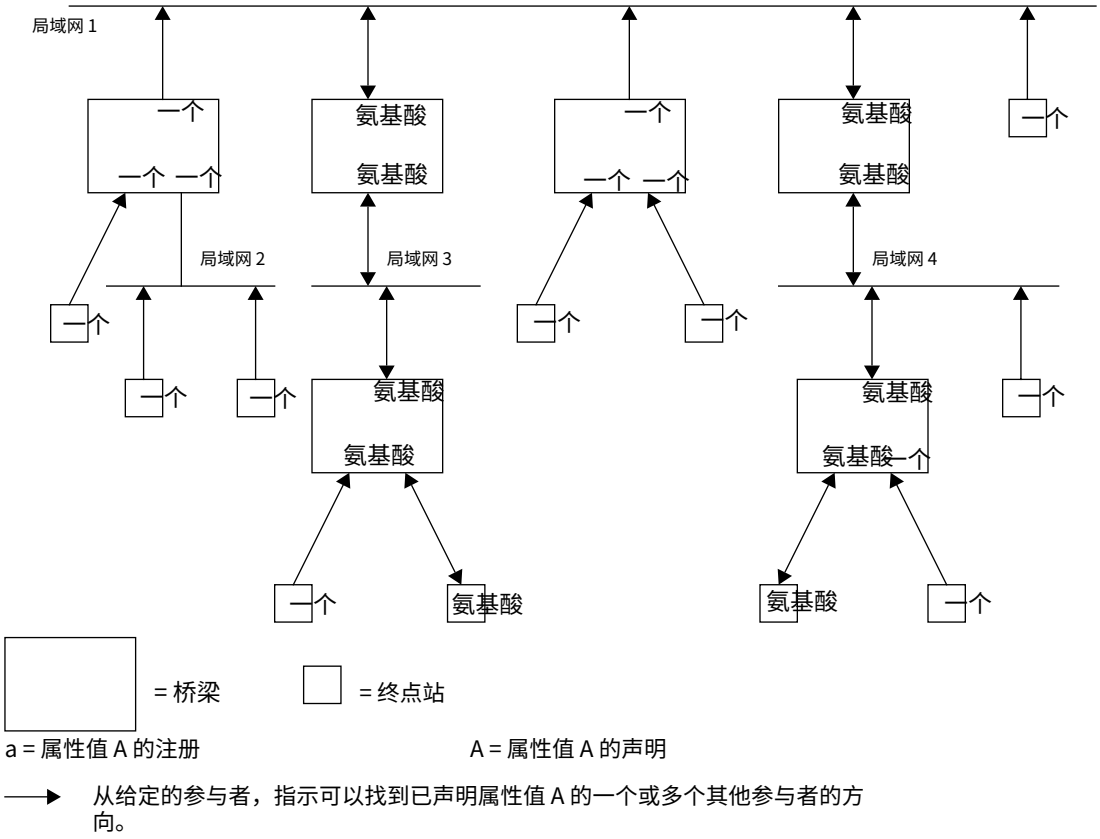


图 12-3—活动拓扑子树的形成示例

桥中的相同应用程序由*GARP 信息传播 (GIP)*组件。GARP 参与者之间通过 LLC 类型 1 服务进行协议交换，使用为相关 GARP 应用程序定义的组 MAC 地址和 PDU 格式。

注 — 本节定义了用于 GARP 协议交换（12.11）的通用 PDU 结构；但是，每个 GARP 应用程序都定义了该通用结构的进一步规范，以便携带该应用程序运行所需的应用程序标识和属性值。

12.3.1 GARP 应用程序组件

对于每个 GARP 应用程序，定义如下：

a) 应用程序使用的一个或多个属性类型的集合；
b) 每种属性类型可以携带的值的集合；
c) 与每个属性类型和值相关的语义；
d) 用于在该应用程序的 GARP 参与者之间交换 GARP PDU 的组 MAC 地址；

e) GARP PDU中属性类型和值的结构和编码;
f) 对支持该应用的终端站和网桥的 GARP 状态机的支持要求。

GARP 参与者的 GARP 应用组件负责定义与 GARP PDU 中接收的参数值和运算符相关的语义，并生成用于传输的 GARP PDU。应用组件利用 GID 组件以及与 GARP 参与者相关的状态机。

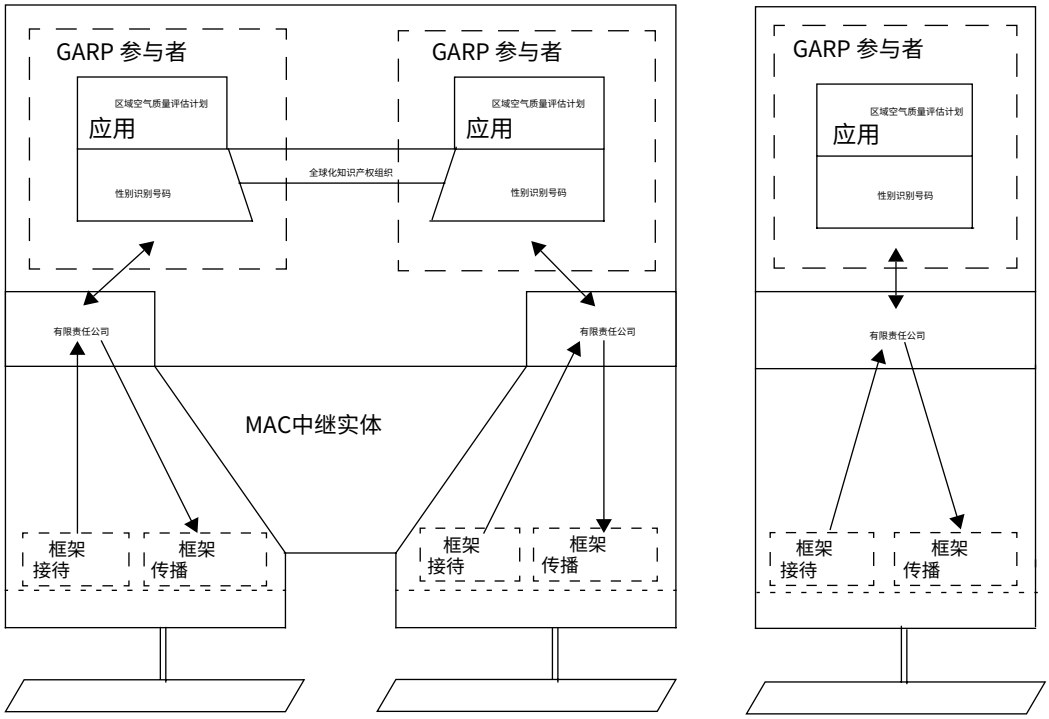


图 12-4 — GARP 架构

与 GID 的操作相关联，以控制其协议交互。GID 组件向应用程序组件提供的服务由 12.3.2 中描述的一组原语定义。

12.3.2 全局标识符

GID 实例由一组状态机组成，这些状态机定义与 GARP 参与者（GID 实例是其组成部分）相关联的所有属性值的当前注册和声明状态。图 12-5 说明了与 GID 实例相关联的一组状态机。

GID 的操作定义为

a）申请人状态转换表（表12-3）；
b) 注册服务商状态转换表（表12-4）；
c) 表示与 GARP 参与者相关的每个属性值的当前声明状态（申请人状态机）和注册状态（注册商状态机）的状态机；

d) GID 用户可用的服务原语。

12.3.2.1 声明

为了允许 GID 服务的用户通过给定的端口请求 GID 做出（加入）或撤销（离开）属性声明，定义了两个原语，如下所示：

GID_Join.请求（属性类型，属性值）

GID_Leave.request（属性类型，属性值）

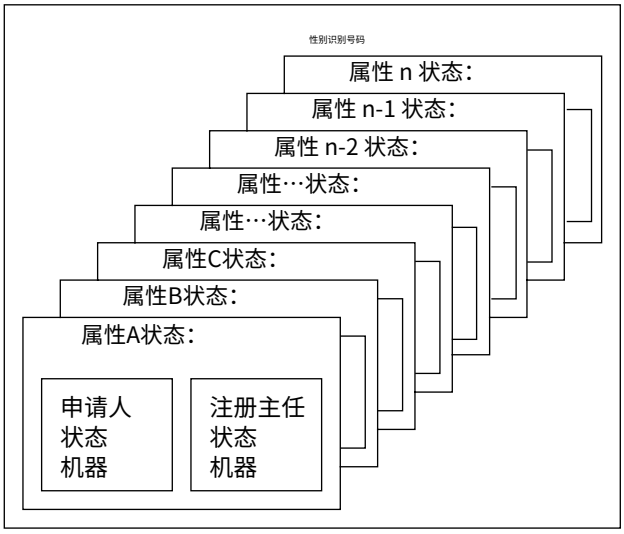


图 12-5 — GID 架构

GID 根据相关属性的当前申请者状态以及申请者状态转换表（表 12-3）中为该状态和事件定义的操作，确定收到这些原语后要采取的操作。如此确定的新状态记录在该属性的申请者状态变量中。

GID 请求可由 GARP 应用程序组件和 GARP 信息传播功能生成。

12.3.2.2 注册

定义了两个原语，以允许 GID 服务向其用户指示给定属性值已在给定端口上注册（加入）或取消注册（离开），这是由于端口所连接的 LAN 段上的协议活动导致的，如下所示：

GID_Join.indication（属性类型，属性值）

GID_Leave.indication（属性类型，属性值）

GID 根据相关属性的当前注册器状态以及注册器状态转换表（表 12-4）中为该状态和事件定义的操作，确定在收到 GARP PDU 中包含的注册和注销信息时要采取的操作。如此确定的新状态记录在该属性的注册器状态变量中。

GARP 应用程序和 GARP 信息传播功能均接收 GID 指示。

12.3.3 全球知识产权局

GARP 信息传播 (GIP) 功能对于所有 GARP 应用程序以相同的方式运行，并使桥接端口上注册的属性信息能够通过桥接 LAN 传播到其他 GARP 参与者。

对于给定的 GARP 应用程序和 GIP 上下文（12.3.4），以及对于由该 GIP 上下文定义的处于转发状态的端口集，GIP 的操作如下：

a) GIP 从集合中的给定端口接收到的任何 GID_Join.indication 都会作为 GID_Join.request 传播到与集合中的任何其他端口关联的 GID 实例；
b) 当且仅当集合中除端口 P 之外的任何其他端口上都不存在该属性的注册时，GIP 从集合中的给定端口接收到的任何 GID_Leave.indication 都将作为 GID_Leave.request 传播到与集合中的任何其他端口（比如端口 P）相关联的 GID 实例。

这些传播规则的效果是，对于 GIP 上下文定义的端口集，如果任何其他端口已看到有关属性的注册，则注册将在给定端口上传播，并且如果所有其他端口现在都已取消有关属性的注册，则取消注册将在给定端口上传播。

由于给定 GIP 上下文中处于转发状态的端口集可以动态变化（例如由于生成树重新配置而变化），因此 GIP 在检测到转发端口集的变化时会按如下方式运行：

c) 如果将某个端口添加到处于转发状态的端口集合中，并且该端口先前已注册过某个属性（对于该属性，GID_Join.indication 发生得比任何 GID_Leave.indication 都晚），则该属性的 GID_Join.requests 会传播到与集合中任何其他端口关联的 GID 实例；

d) 如果某个端口从处于转发状态的端口集中移除，并且该端口之前已注册过某个属性（该属性的 GID_Join.indication 发生得比任何 GID_Leave.indication 都更近），则该属性的 GID_Leave.requests 会传播到实例
与集合中任何其他端口关联的 GID。

12.3.4 GARP 信息传播上下文

对于 GARP 感知桥的给定端口和该桥支持的 GARP 应用程序，每个端口都可以存在一个 GARP 参与者实例*GARP 信息传播上下文*（GIP 上下文 (GIP Context) 是该 Bridge 所理解的。GIP 上下文定义了 Bridge 端口的子集，这些端口构成了该上下文中适用的活动拓扑 (7.4)。GIP 上下文是相对于某个其他活动拓扑定义的，并定义了该活动拓扑的子集。

GIP 上下文的一个示例是由生成树算法和协议操作形成的活动拓扑定义的上下文（第 8 条）。对于此 GIP 上下文，此上下文定义的给定桥内的活动拓扑与生成 树操作形成的活动拓扑完全相同。此 GIP 上下文称为*基本生成树上下文*。

注意：本标准仅使用基本生成树上下文来定义 GMRP 的操作；但是，GARP 的定义方式允许使用其他 GIP 上下文，如果这对于其他 GARP 应用程序或对现有 GMRP 功能的扩展是必要的。特别是，这种灵活性是为了允许在基于使用多个生成树和/或虚拟 LAN (VLAN) 技术的网络架构中使用 GARP 应用程序，其中活动拓扑将由生成树实例或定义上下文的 VLAN 定义。VLAN 是 IEEE P802.1Q 项目下进一步开发的主题。

GARP 协议交换可以在桥接的所有端口上进行；但是，对于给定的 GARP 应用程序，由 GIP（12.3.3）操作控制的 GARP 注册信息在桥接 LAN 上的传播仅发生在由 GIP 上下文定义的活动拓扑的端口集之间。GIP 上下文的定义使得为传播注册信息而选择的活动拓扑允许将 GARP 注册传播到桥接 LAN 的所有区域，这些区域（可能）包含该 GARP 应用程序和 GIP 上下文的 GARP 参与者。每个 GARP 应用程序都定义了一组可以在其中运行的 GIP 上下文，定义了一组转发端口的规则

为每个上下文选择 GIP 上下文，并为每个上下文分配 GIP 上下文标识符，以便与 GARP 的操作及其管理控制 (12.9) 结合使用。

对于由第 8 条中定义的生成树算法和协议的操作所定义的 GIP 上下文，应当使用 0 值作为 GIP 上下文标识符。当在任何 GIP 上下文中使用除 0 以外的 GARP 时，GARP 应用程序的定义应当指定如何识别 GARP 参与者之间交换的 PDU 属于相关的 GIP 上下文。

12.4 GARP 需要满足的要求

GARP 及其相关算法用于在桥接 LAN 内建立、维护和解除属性注册，并在连接到 LAN 的 GARP 参与者之间传播注册信息。为了执行此功能，协议及其相关算法需要满足以下要求，这些要求在所示子条款中进行了说明。这些要求包括申请人和注册商行为的要求、故障条件下的错误恢复、性能、可扩展性、与不支持 GARP 的设备的向后兼容性，以及对桥接器、终端站和网络施加的负载：

a) 它将允许连接到桥接 LAN 的 GARP 参与者发布与 GARP 应用程序相关的属性值的声明（12.3.2、12.8.1 和 12.10）；
b) 它将允许连接到桥接 LAN 的 GARP 参与者撤销与 GARP 应用程序相关的属性值的声明（12.3.2、12.8.1 和 12.10）；
c) 它将把桥接器收到的声明传播给可通过该桥接器所连接的 LAN 段访问的 GARP 参与者（12.3.3）；

d) 它将允许 GARP 参与者维护状态信息，该信息指示参与者设备的每个端口上属性的声明和注册的当前状态（12.8.1 和 12.8.2）；
e) 它将允许 GARP 参与者删除与桥接 LAN 的部分或全部内不再活动的属性相关的状态信息，例如由于参与者发生故障而导致的属性状态信息（12.8.2 和 12.8.3）；

f) 建立或撤销属性注册信息以及通过桥接 LAN 传播该信息所涉及的等待时间将很小（即与桥接 LAN 上的帧传播延迟相当），并将作为桥接 LAN 直径的函数线性增加（12.8.1、12.10、12.8.2 和 12.8.3）；

g) 当 GARP 参与者发生故障时，它将以有弹性的方式运行（H.1.4）；
h) 它将以一种在单个数据包丢失的情况下具有弹性的方式运行；
i）它将正常运行
    1) 在同构桥接局域网中，即所有桥接器都支持基本过滤服务和扩展过滤服务（12.8.1、12.10、12.8.2 和 12.8.3）的桥接局域网中；
    2) 在异构桥接局域网中，即桥接局域网中有些网桥仅支持基本过滤服务，有些网桥同时支持基本过滤服务和扩展过滤服务（12.5、12.8.1、12.10、12.8.2、12.8.3 和 F.2）。

j) 申请人和注册商在与协议相关的 PDU 交换中在任何特定 LAN 段上消耗的通信带宽将只占总可用带宽的一小部分，并且与桥接 LAN 支持的总流量无关。所消耗的带宽将取决于该 LAN 上的申请人希望保持成员资格的组数。

12.5 GARP 参与者之间的互操作性要求

为了保证GARP的互操作性，需要做到以下几点：

a) 对于每个定义的 GARP 应用程序，应使用一个唯一的组 MAC 地址（称为 GARP 应用程序地址）作为 GARP 协议交换的目标 MAC 地址

该应用程序的 GARP 参与者。表 12-1 定义了已分配给现有 GARP 应用程序的组 MAC 地址集；该表中未分配的值已保留供未来标准化 GARP 应用程序使用。实施与该表中给定条目对应的 GARP 应用程序的网桥不得转发发往该 MAC 地址的帧；未实施与该表中给定条目对应的 GARP 应用程序的网桥必须将活动拓扑中的任何端口上接收到的发往该 MAC 地址的帧转发到活动拓扑中的所有其他端口；

b) GARP 参与者之间的 GARP PDU 传输和接收应通过 LLC 类型 1 程序实现，格式按照相关 GARP 应用程序的定义，使用 12.11 中定义的通用 PDU 格式。应使用表 7-8 中定义的生成树协议的标准 LLC 地址分配作为源和目标 LLC 地址。

图 12-4 显示了 LLC 类型 1 程序的使用；

c) GARP 参与者收到的 PDU [即，具有 a) 和 b) 中指定的目标 MAC 和 LLC 地址的 PDU]，如果格式不正确（即，不是按照 12.11 中定义的那样构造和编码，并且不具有按照相关应用程序定义的那样编码的属性类型和值），则应被丢弃；

d) GARP参与者的协议行为应符合12.8和12.10中定义的状态机描述和程序。

表 12-1 — GARP 应用程序地址

任务	价值
GMRP 地址（参见第 10 条）	01-80-C2-00-00-20
预订的	01-80-C2-00-00-21
预订的	01-80-C2-00-00-22
预订的	01-80-C2-00-00-23
预订的	01-80-C2-00-00-24
预订的	01-80-C2-00-00-25
预订的	01-80-C2-00-00-26
预订的	01-80-C2-00-00-27
预订的	01-80-C2-00-00-28
预订的	01-80-C2-00-00-29
预订的	01-80-C2-00-00-2A
预订的	01-80-C2-00-00-2B
预订的	01-80-C2-00-00-2C
预订的	01-80-C2-00-00-2D
预订的	01-80-C2-00-00-2E
预订的	01-80-C2-00-00-2F

注 1 — 表 12-1 中的第二个条目在 IEEE P802.1Q 中分配供 GARP VLAN 注册协议 (GVRP) 使用。

注 2 — GARP 使用与生成树协议相同的 LLC 地址；但是，GARP 应用程序和生成树使用不同的 MAC 地址和协议标识符可确保将适当的 PDU 传送到适当的协议实体。桥接协议实体接收目标 MAC 地址等于表 7-9 中标识的桥接组地址、LLC 地址等于表 7-8 中指定的值并且带有第 9 条中定义的生成树协议标识符的 PDU（参见 7.12.3）。目标 MAC 地址等于表 12-1 中标识的任何 GARP 应用程序地址、LLC 地址等于表 7-8 中指定的值并且带有第 12.11 条中定义的 GARP 协议标识符的 PDU，其处理方式如上文 a) 至 d) 所述。

12.6 符合 GARP 应用程序

本节中描述的 GARP 规范定义了 GARP 应用程序通用的行为方面。对 GARP 的符合性是针对特定 GARP 应用程序定义的。每个 GARP 应用程序都定义了声明符合该应用程序所需的 GARP 功能方面，如 12.3.1 中所述。因此，符合 GARP 的实现是至少支持声明符合该实现所支持的 GARP 应用程序集所需的 GARP 功能的实现。

12.7 GARP 协议操作概述

本概述仅作为对 GARP 协议操作的非正式介绍。 最终描述包含在状态机描述 (12.8)、过程 (12.10) 和 GARP 协议数据单元的编码 (12.11) 中。

在下面的描述中，系统端口之间的信息传播遵循 GIP 上下文的活动拓扑。

12.7.1 基本概念

GARP 运作背后的基本概念是

a) 可以实现简单、完全分布式的多对多协议。无需额外的选举协议即可改变问题以允许多对一设计；

b) 该协议应该能够抵御一组相关消息中单个消息的丢失，但不需要更强大；

c) 希望作出声明的 GARP 参与者（申请人）发送加入消息；

d) 如果申请者看到其他参与者发送了两个加入消息，则它自己不需要发送加入消息就可以参与有关声明；

e) 想要撤回声明的申请人只需发送一条“离开”消息即可；然后就可以忘记相关注册。无需确认已注销注册，因为其他参与者可能希望自己保留注册。

f) 因多条消息丢失而导致的丢失或虚假继续注册可通过定期发送 LeaveAll 消息的机制进行清除。LeaveAll 消息声明所有注册将很快终止，除非一个或多个参与者通过发出进一步的加入来声明对特定注册的持续兴趣。

为了防止组成员错过 Leave 消息，从而导致另一个参与者的注册器（记录组成员身份的组件）认为没有成员，需要一种额外的机制。如果参与者收到 Leave 消息，并且没有后续加入，它会在终止注册之前发送另一条消息以提示重新加入。

12.7.2 GARP 消息

到目前为止的描述介绍了 GARP 中使用的三种基本消息类型：Join、Leave 和 LeaveAll。但是，仅使用这三种消息类型会增加协议的最终复杂性：

— 考虑两个 GARP 参与者位于点对点链路两端的情况。其中一个参与者（比如说 Andy）向另一个参与者（Bill）发送两条加入消息，但这并不说明 Andy 知道 Bill 希望做出该声明。上面对基本 GARP 概念的复述掩盖了这样一个事实：潜在申请人可能还需要知道是否有其他申请人做出相同的声明，例如桥接端口的要求。尝试通过设置加入计时器来解决这个问题，使得在两个或更多参与者可能发送消息的间隔内，没有一个参与者可以发送两条消息，这会导致计时器正确性依赖性，在确定协议操作的正确性方面，这种依赖性非常危险。

— 考虑注册器发送第二个 Leave 以提示重新加入。如果刚刚发送 Join，则协议现在取决于第二个 Join 不会丢失。协议取决于注册器的 Leave 计时器和其他参与者的 Join 计时器的相对值。

因此，该协议基于一般设计原则，即协议参与者应传达其当前状态，而不是发送指示。使用四种特定于属性的消息类型：

a) 空：我不想声明这个属性值。我没有注册过这个属性值，但我关心是否有任何参与者希望声明它。

b) JoinEmpty：我希望声明此属性值。我尚未注册此属性值，但我关心是否有任何参与者希望声明它。
c) JoinIn：我希望声明此属性值。我要么已经注册了此属性值，要么我不在乎是否有其他参与者希望声明它（我将表现得好像有其他参与者一样）。
d) 离开：我已经注册了此属性值，但是现在正在取消注册它。

和之前一样，还有垃圾收集消息

e) LeaveAll：所有注册将很快被取消；如果任何参与者对任何注册有持续的兴趣，他们需要重新加入以维持注册。

理论上 Leave 消息可能有 LeaveIn 和 LeaveEmpty 变体；这些变体在 GARP PDU 中编码，以最大程度地了解实现正在做什么，并避免丢失或非法代码。但是，可以看出，状态机对这两个消息变体的处理方式相同。

没有理由发送一个简单的 In 消息，即“我不想做出这个声明，但已经代表其他参与者注册了属性值（或者表现得好像有其他参与者已经做出了声明）”。

该协议充分利用了 JoinEmpty 和 JoinIn 消息以及 Leave 和 Empty 之间的区别。

JoinIn 消息满足 Join 消息抑制的要求。如果申请人看到 JoinIn 消息，它可以避免自己发送 Join 来声明该声明，因为它知道 JoinIn 的接收者和发送者都认为有参与者已经做出了声明。JoinIn 不被视为确认，因为在多路访问段上，可能有许多参与者需要注册属性值。此外，不关心是否有其他参与者对该注册感兴趣的参与者可以始终发送 JoinIn 而不是 JoinEmptys。但是，假设只有一条 JoinIn 消息丢失，两条消息足以确保所有注册者都已注册该组，而且可能性很高。

Leave 消息将导致其接收者取消成员资格注册，而 JoinEmpty 和 Empty 消息只会提示他们重新加入，因此 JoinEmpty 和 Empty 消息可随时用于提示重新加入，而无需再次将最近加入的成员剔除出去。

12.7.3 申请人和注册官

每个 GARP 参与者都维护一个 Leave All 协议组件 (12.7.6)。它还为每个感兴趣的属性维护两个协议组件：申请人和注册商。

注册器的作用是记录该网段上其他参与者声明的属性注册。它不发送任何协议消息。

申请人的工作有两方面：

a) 确保该参与者的声明已由其他参与者的注册机构注册——如果它想要保留这些注册。

b) 确保在任何人撤回声明（离开）后，其他参与者有机会重新声明（重新加入）——如果有任何参与者想要维持注册。

注——上述第 b) 项仅适用于完整申请人状态机（12.7.5）的行为；仅申请人状态机和简单申请人状态机（12.7.7 和 12.7.8）仅涉及第 a) 项。

因此，申请人要照顾所有潜在参与者的利益。这使得注册程序变得非常简单。

12.7.4 注册商行为

注册器有一个计时器，即离开计时器和三种状态：

a) IN：我已经注册了这个属性值已经在这个段上声明的事实。
b) MT：（空）此段上该属性值的所有声明均已撤回。
c) LV：我已注册此属性值，但现在注册已超时（使用离开计时器）。如果我在离开计时器到期之前未看到此属性的声明，我将成为 MT。

注册服务商对收到的消息做出如下反应：

d) 加入消息（JoinIn 或 JoinEmpty）使注册器变为 IN（我已注册该属性）。

e) 如果注册器处于 IN 状态，则 Leave 或 LeaveAll 会导致其变为 LV（注册超时）并启动 Leave 计时器。否则（LV 或 MT）则无效果。
f) 空消息（其他人没有注册该属性）无效[见12.7.2 a）]。

Registrar 虽然不发送消息，但是会影响 Applicant 发送的 Join 消息类型。如果 Registrar 为 IN，则发送 JoinIn；否则发送 JoinEmpty。

12.7.5 申请人行为

在这个简单的注册器的背景下，接下来要考虑的是希望作出声明的申请人的行为，从其从未看到或发送任何消息开始。

如果没有丢失任何消息，申请者可以发送 Join 或接收 JoinIn，然后满足于所有注册商都已注册其声明。在单条消息丢失假设下，它需要发送两个 Join，或接收两个 JoinIn，或发送一个 Join 并接收一个 JoinIn（以任意顺序）。其状态的这一部分可以记录在一个简单的计数器中：

my_membership_msgs = 0、1 或 2

每次发送 Join 或接收 JoinIn 时，计数器都会递增。如果有机会传输 PDU 时计数器值为 0 或 1，则会发送 Join 消息并递增计数器。

注 1—为了成功注册，计数器值不需要大于 2。
注 2 — 随机加入计时器已设置为运行，以确保安排这样的机会。整个参与者只需要运行一个加入计时器，而不是每个属性运行一个 — 假设与最大数量的属性相关的消息可以打包到单个 PDU 中。

如果收到 JoinEmpty、Empty、Leave 或 LeaveAll 消息，则计数器重置为 0。

当申请人离开该组时，它会发送一条离开消息。

12.7.5.1 焦虑的申请人

如果主要目标是实现全面分析和最大程度提高实现灵活性，那么用计数器和标志变量来表达协议行为并不总是最佳方法。从此时起，分配给加入消息计数的值将获得状态名称前缀：

a) V 或非常焦虑相当于 my_join_msgs = 0。自申请人启动以来，没有发送过任何加入消息，也没有收到任何加入消息，或者收到过离开或空白消息。申请人没有理由相信其他注册商已经注册了相关的属性值。
b) A 或 Anxious 等同于 my_join_msgs = 1。如果没有丢失消息，其他注册商将注册此属性值。

c) Q 或 Quiet 等同于 my_join_msgs = 2。申请人认为没有必要发送进一步的消息。

12.7.5.2 成员和观察员

到目前为止描述的申请人除非试图做出声明，否则不需要存在。积极使用已注册属性值的桥接端口和终端站（例如，在第 10 条中定义的 GMRP 应用程序的情况下，为传输实施源修剪的桥接器和终端站）需要维护其 GARP 机器，即使它们不想做出（或刚刚撤回）声明。（术语 GARP 机器是指参与者中申请人和注册商为给定属性值维护的总状态。）

在申请人状态机的上下文中，成员是试图对给定属性值做出或维持声明的参与者，或者尚未发送 Leave 消息以允许其成为观察者。观察者会跟踪属性状态，但不希望做出声明。

12.7.5.3 主动和被动成员

引入主动和被动成员的概念，以便当多个参与者针对同一注册主动加入和离开时，允许发送最少数量的消息。

由于成员可能在未发送加入消息的情况下变为静默状态，因此应允许其再次成为观察员，而无需发送离开消息。所有观察员都是被动的，因此存在三种潜在（子）状态，由以下状态名称后缀区分：

a) A，或活跃会员。
b）P，即被动成员。
c）O，即观察员。

如果观察员需要成为成员，它首先成为被动成员。如果它是安静观察员（即其加入消息数已为 2，因此它对其他注册员已注册声明感到满意），则它无需发送加入消息，并成为被动和安静。否则，即其加入消息数少于 2，它会请求最早的消息传输机会以发送加入消息。

如果被动成员发送加入消息，它将成为主动成员。

如果主动成员收到 Leave 或 LeaveAll 消息，它将成为被动成员。

12.7.5.4 请假

当申请人是会员并希望继续成为会员时，它会收到离开消息，它会变得非常焦虑。除非它收到来自另一个会员的加入消息，否则它会自己发送 JoinEmpty。这对其他会员有以下影响。首先，它将导致他们注册该属性。其次，如果他们希望继续成为会员并传输 JoinIn，它将导致他们自己变得非常焦虑。

后者的效果是保护任何作为成员的参与者不会因为离开后的一个数据包丢失而意外取消其他成员的注册。

作为观察员的申请人必须提示其他成员重新加入，以防他们错过离开。将另一个（子）状态添加到非常焦虑、焦虑、安静的集合中，其状态名称前缀为：

— L 或离开，记录下一次传输机会时需要发送消息的情况。观察者将发送一条空消息，然后变得非常焦虑。

12.7.5.5 离开

活跃成员必须发送 Leave 消息才能撤回声明。Leaving 子状态用于记录该事实。

12.7.5.6 申请国概况

以下矩阵总结了申请人状态及其简称：VA 代表非常焦虑的活跃成员，QO 代表安静的观察者，等等。

表 12-2 - 申请人：各州摘要

	非常焦虑	焦虑的	安静的	离开
活跃会员	弗吉尼亚州	AA	质量保证	洛杉矶
被动会员	副总裁	美联社	量子点	
观察者	画外音	AO	质量保证	洛

请注意，不存在 LP（离开被动成员）状态，因为被动成员在希望撤回声明时可以直接转换为观察员状态。

12.7.6 Leave All 协议组件

Leave All 协议组件负责启动参与者的垃圾收集。这是通过 LeaveAll 状态机定期生成 LeaveAll 消息来实现的，如 12.8.3 中定义。

Leave All 状态机的操作会导致参与者在 leaveAllTimer 到期时生成 LeaveAll 消息。从另一个参与者收到 Leave All 消息会导致计时器重新启动而不生成消息，从而确保当多个参与者连接到同一个 LAN 段时，多个 LeaveAll 消息被抑制。

收到 LeaveAll 消息会导致所有申请人变得非常焦虑，并且所有注册商都会进入离开 (LV) 状态。这样做的效果是迫使任何仍处于活动状态的申请人重新加入；如果注册商状态机在其离开计时器到期之前未看到任何加入消息，它将取消注册该属性。这实际上会导致参与者删除所有不再存在活动申请人的注册。

12.7.7 仅限申请人的参与者

在参与者仅希望做出声明的情况下，可以简化 GARP 参与者；例如，终端站可能需要通过 GMRP（第 10 条）实现注册以接收组寻址帧的功能，但终端站不作为此类帧的源，因此无需实现源修剪。这样的参与者无需注意其他参与者做出的声明（即，它不需要实现注册器状态机），不会发送 Leave All 或 Join Empty 消息，也不会提供额外的管理控制。

这可能导致此类参与者需要支持的国家机器简化，如下所示：

a) 不需要支持LeaveAll定时器，或者LeaveAll PDU的生成；
b) 无需支持注册商状态机的操作。注册商的“IN”状态是为申请人状态机的操作而假设的，因此 Join 消息始终以 JoinIn 的形式发送；
c) 不需要状态机支持LO状态或生成Empty消息；
d) 无需支持 12.9 中定义的管理控制。

仅申请人状态机的操作在 12.8.5 和表 12-8 中描述。

12.7.8 简单申请参与者

简单申请人参与者的操作是对仅申请人状态机（表 12-8）的进一步简化，通过删除被动成员和观察者状态，将申请人简化为最简单的形式。因此，简单申请人参与者不会试图抑制其初始加入和最终离开消息。结果是最简单的申请人状态机，与完整 GARP 参与者的操作兼容。

简单申请人状态机的操作在12.8.6和表12-9中描述。

12.7.9 选择仅申请人参与者或简单申请人参与者

仅申请人参与者保留了完整申请人的被动成员和观察员状态，这意味着它保留了完整参与者在不需要时抑制加入或离开消息的能力（参见 12.7.5.3）；相反，简单申请人参与者无法执行此类抑制。如果有可能在同一 LAN 段上出现多个简单申请人参与者，则可能导致大量额外的、不必要的加入和离开流量。因此，建议在无需执行注册的设备中优先实施仅申请人参与者，而不是简单申请人参与者。

12.8 状态机描述

本款中使用的缩写采用以下约定：

編輯 接收 PDU XXX
韓國 发送 PDU XXX
请求XXX GID服务请求XXX GID服务指
印地安纳州 示XXX

状态机描述中使用了以下缩写。有关其含义的正式定义，请参阅 12.10：

加入 接收“加入”消息 接
rJoinEmpty 收“加入”空消息 接
空 收“空”消息
离开 接收 Leave In 消息 接收 Leave
rLeaveEmpty Empty 消息 接收或发送 LeaveAll
全部离开 消息
sJ[E,I] 如果注册器状态 = IN，则发送加入消息；否则，发送加入空消息 发送加入消息
sJ[我]
东南 发送空消息
系统性红斑狼疮 发送 留下空白信息 发送 留
全部离开 下全部信息
请求加入 GID 服务请求声明属性值
要求休假 GID 服务请求撤回属性值声明
连接 发出 GID 服务指示，表示属性值已注册 发出 GID 服务指示，表示属性值已注销 传
独立脱欧 输 GARP PDU 的机会已经出现。此类事件在间隔 0 - JoinTime 中随机选择的时间
传输PDU！ 间隔内发生。JoinTime 的定义如下

表 12-10

离开计时器 休假时间计时器
离开计时器！ 休假时间已到
离开所有计时器 全部离开周期计时器。全
离开所有时间！ 部离开计时器已到期。
- x- 不适用的事件/状态组合。没有发生任何动作或状态转换。

状态机描述中使用计时器是为了在定义的时间段过去后采取行动。状态机描述中使用以下术语来定义计时器状态以及可对其执行的操作：

a) 计时器被称为*跑步*如果最近对其执行的操作是*开始*或 *重新启动。*

b）正在运行的计时器被认为具有*已到期*自最近一次启动或重启操作发生以来，与计时器关联的时间段已经过去。

c）计时器被称为*停止*如果它已过期，或者最近对其执行的操作是*停止*行动。

d）A*开始*操作将已停止的计时器设置为运行状态，并将时间段与计时器关联。此时间段将取代先前启动事件可能与计时器关联的任何时间段。

e）A*重启*操作会停止正在运行的计时器，然后对其执行启动操作。

f) A*停止*动作将计时器设置为停止状态。

状态表和状态图中的状态名称使用以下缩写：

注册处状态

路威酩轩 离开
在 在
公吨 空的

申请国

弗吉尼亚州 非常焦虑，活跃成员 焦虑，
AA 活跃成员
质量保证 安静、活跃的成员
洛杉矶 离开，活跃成员
副总裁 非常焦虑，被动成员 焦虑，
美联社 被动成员 安静，被动成员
量子点
画外音 观察者非常焦虑
AO 焦虑的观察者
质量保证 安静，观察者
洛 离开，观察者

简单申请人状态

五 很焦虑
一个 焦虑的
问 安静的

12.8.1 申请人状态机

对于参与者需要维护状态信息的每个属性值，完整 GARP 参与者都会维护此状态机的单个实例。

注 — 从概念上讲，将针对为给定应用程序定义的所有属性类型的所有可能值维护状态信息；但是，在 GARP 的实际实施中，某些应用程序中可能的属性值范围可能会排除这种情况，并且实施将限制状态为参与者作为成员或可能的未来成员有直接兴趣的属性值。

表 12-3 描述了此状态机的详细操作。所示的状态转换处理接收 Leave In 或 Leave Empty 消息的可能性；但是，状态机只生成 Leave Empty 消息。发送 Leave Empty 消息还会导致针对 Registrar 状态机的事件，从而导致从 IN 到 LV 的转换。

12.8.2 注册器状态机

对于当前已注册的每个属性值，或者注册器状态机正在取消注册的过程中，完整的 GARP 参与者都会维护此状态机的单个实例。

表 12-3—申请人州表

		状态										
		弗吉尼亚州	AA	质量保证	洛杉矶	副总裁	美联社	量子点	画外音	AO	问哦	大号哦
事件	**传输PDU!**	sJ[E,I] AA	sJ[E,I] 质量保证	- x-	系统性红斑狼疮 画外音	sJ[E,I] AA	sJ[E,I] 质量保证	- x-	- x-	- x-	- x-	东南 画外音
	加入	AA	质量保证	质量保证	洛杉矶	美联社	量子点	量子点	AO	质量保证	质量保证	AO
	rJoinEmpty	弗吉尼亚州	弗吉尼亚州	弗吉尼亚州	画外音	副总裁	副总裁	副总裁	画外音	画外音	画外音	画外音
	空	弗吉尼亚州	弗吉尼亚州	弗吉尼亚州	洛杉矶	副总裁	副总裁	副总裁	画外音	画外音	画外音	画外音
	离开	弗吉尼亚州	弗吉尼亚州	弗吉尼亚州	洛杉矶	副总裁	副总裁	副总裁	洛	洛	洛	画外音
	rLeaveEmpty	副总裁	副总裁	副总裁	画外音	副总裁	副总裁	副总裁	洛	洛	洛	画外音
	全部离开	副总裁	副总裁	副总裁	画外音	副总裁	副总裁	副总裁	洛	洛	洛	画外音
	请求加入	- x-	- x-	- x-	弗吉尼亚州	- x-	- x-	- x-	副总裁	美联社	量子点	副总裁
	要求休假	洛杉矶	洛杉矶	洛杉矶	- x-	画外音	AO	质量保证	- x-	- x-	- x-	- x-

注意：与申请人一样，状态信息在概念上是针对给定应用程序定义的所有属性类型的所有可能值进行维护的；但是，在 GARP 的实际实施中，某些应用程序中可能的属性值范围可能会排除这种情况，并且实施将状态限制为参与者有直接兴趣的属性值。对于对其他参与者注册的内容不感兴趣的简单设备，该设备可能完全忽略注册商操作是合适的。

该状态机的详细操作如表12-4所示。

12.8.3 离开全部状态机

每个完整 GARP 参与者都存在一个 Leave All 状态机。此状态机生成的 Leave All 消息还会针对与该参与者和端口相关的所有申请人和注册商状态机生成 LeaveAll 事件；因此，这些状态机处理 LeaveAll 生成的方式与接收来自外部源的 LeaveAll 消息的方式相同。

该状态机的详细操作如表12-5所示。

12.8.4 申请人/注册商组合状态机

表 12-6 显示了所有可达状态，其中单元格包含联合状态名称、申请人.注册商，不可达状态标记为 ---。MT 和 LV 注册商状态被归为一组，因为区分这两个状态的唯一事件是离开计时器到期，这不会影响任何其他状态。总共有 24 个可达状态。

组合状态机如表 12-7 所示。为了简洁起见，省略了动作（传输什么消息、实现应在何时检查或启动计时器、何时向上层用户指示加入和离开）。

表 12-4 — 注册商状态表

		状态		
		在	**路威酩轩**	**公吨**
事件	**加入**	在	停止离开定时器 连接 在	连接 在
	rJoinEmpty	在	停止离开定时器 连接 在	连接 在
	空	在	路威酩轩	公吨
	离开	启动离开定时器 路威酩轩	路威酩轩	公吨
	rLeaveEmpty	启动离开定时器 路威酩轩	路威酩轩	公吨
	全部离开	启动离开定时器 路威酩轩	路威酩轩	公吨
	离开计时器!	- x-	独立脱欧 公吨	- x-

表 12-5 — 全部离开状态表

		状态	
		积极的	**被动的**
事件	**传输PDU!**	全部离开 被动的	- x-
	全部离开	启动 leavealltimer 被动的	启动 leavealltimer 被动的
	离开所有时间!	启动 leavealltimer 积极的	启动 leavealltimer 积极的
	（所有其他活动）	- x-	- x-

表 12-6 — 申请人/注册商联合州

	非常焦虑		**焦虑的**		**安静的**		**离开**	
活跃会员	VA VA LV	弗吉尼亚州	AA—MT AA.LV	AA.印度	量子点 质量保证	质量保证	洛杉矶 洛杉矶	拉美
被动会员	副总裁 LV副总裁	副总裁	--- ---	美联社	---	韓國		
观察者	VO 左心室射血分数	配音	--- ---	AO印度	---	质量保证	低密度脂蛋白 低电压	---

表 12-7 — 申请人/注册商合并状态表

状态		事件								
		离开计时器！	传输PDU！	加入	rJoinEmpty	空	离开	rLeaveEmpty，全部离开	请求加入	要求休假
	VA	- x-	AA—MT	AA.印度	弗吉尼亚州	VA	副总裁	副总裁	- x-	洛杉矶
	VA LV	VA	AA.LV	AA.印度	弗吉尼亚州	VA LV	LV副总裁	LV副总裁	- x-	洛杉矶
	弗吉尼亚州	- x-	AA.印度	AA.印度	弗吉尼亚州	弗吉尼亚州	LV副总裁	LV副总裁	- x-	拉美
	AA—MT	- x-	量子点	质量保证	弗吉尼亚州	VA	副总裁	副总裁	- x-	洛杉矶
	AA.LV	AA—MT	质量保证	质量保证	弗吉尼亚州	VA LV	LV副总裁	LV副总裁	- x-	洛杉矶
	AA.印度	- x-	质量保证	质量保证	弗吉尼亚州	弗吉尼亚州	LV副总裁	LV副总裁	- x-	拉美
	量子点	- x-	—	质量保证	弗吉尼亚州	VA	副总裁	副总裁	- x-	洛杉矶
	质量保证	量子点	—	质量保证	弗吉尼亚州	VA LV	LV副总裁	LV副总裁	- x-	洛杉矶
	质量保证	- x-	—	质量保证	弗吉尼亚州	弗吉尼亚州	LV副总裁	LV副总裁	- x-	拉美
	洛杉矶	- x-	VO	拉美	配音	洛杉矶	洛杉矶	VO	VA	- x-
	洛杉矶	洛杉矶	左心室射血分数	拉美	配音	洛杉矶	洛杉矶	左心室射血分数	VA LV	- x-
	拉美	- x-	左心室射血分数	拉美	配音	拉美	洛杉矶	左心室射血分数	弗吉尼亚州	- x-
	副总裁	- x-	AA—MT	美联社	副总裁	副总裁	副总裁	副总裁	- x-	VO
	LV副总裁	副总裁	AA.LV	美联社	副总裁	LV副总裁	LV副总裁	LV副总裁	- x-	左心室射血分数
	副总裁	- x-	AA.印度	美联社	副总裁	副总裁	LV副总裁	LV副总裁	- x-	配音
	美联社	- x-	质量保证	韓國	副总裁	副总裁	LV副总裁	LV副总裁	- x-	AO印度
	韓國	- x-	—	韓國	副总裁	副总裁	LV副总裁	LV副总裁	- x-	质量保证
	VO	- x-	—	AO印度	配音	VO	低密度脂蛋白	低密度脂蛋白	副总裁	- x-
	左心室射血分数	VO	—	AO印度	配音	左心室射血分数	低电压	低电压	LV副总裁	- x-
	配音	- x-	—	AO印度	配音	配音	低电压	低电压	副总裁	- x-
	AO印度	- x-	—	质量保证	配音	配音	低电压	低电压	美联社	- x-
	质量保证	- x-	—	质量保证	配音	配音	低电压	低电压	韓國	- x-
	低密度脂蛋白	- x-	VO	AO印度	配音	VO	低密度脂蛋白	低密度脂蛋白	副总裁	- x-
	低电压	低密度脂蛋白	左心室射血分数	AO印度	配音	左心室射血分数	低电压	低电压	LV副总裁	- x-

12.8.5 仅限申请人的 GARP 参与者

仅限申请人的 GARP 参与者针对参与者需要维护状态信息的每个属性值维护仅限申请人状态机的单个实例。

表 12-8 描述了此状态机的详细操作。所示的状态转换处理接收 Leave In 或 Leave Empty 消息的可能性；但是，状态机只生成 Leave Empty 消息。

表 12-8 — 仅限申请人状态机

		状态									
		弗吉尼亚州	AA	质量保证	洛杉矶	副总裁	美联社	量子点	画外音	AO	问哦
输入	发送-和平单位！	SJ[我] AA	SJ[我] 质量保证	- x-	系统性红斑狼疮 画外音	SJ[我] AA	SJ[我] 质量保证	- x-	- x-	- x-	- x-
	加入	AA	质量保证	质量保证	洛杉矶	美联社	量子点	量子点	AO	质量保证	质量保证
	rJoinEmpty	弗吉尼亚州	弗吉尼亚州	弗吉尼亚州	画外音	副总裁	副总裁	副总裁	画外音	画外音	画外音
	空	弗吉尼亚州	弗吉尼亚州	弗吉尼亚州	洛杉矶	副总裁	副总裁	副总裁	画外音	画外音	画外音
	离开	弗吉尼亚州	弗吉尼亚州	弗吉尼亚州	洛杉矶	副总裁	副总裁	副总裁	画外音	画外音	画外音
	r离开-空的	副总裁	副总裁	副总裁	画外音	副总裁	副总裁	副总裁	画外音	画外音	画外音
	全部离开	副总裁	副总裁	副总裁	画外音	副总裁	副总裁	副总裁	画外音	画外音	画外音
	请求加入	- x-	- x-	- x-	弗吉尼亚州	- x-	- x-	- x-	副总裁	美联社	量子点
	要求休假	洛杉矶	洛杉矶	洛杉矶	- x-	画外音	AO	质量保证	- x-	- x-	- x-

12.8.6 简单申请参与者

对于参与者需要维护状态信息的每个属性值，简单申请人参与者都会维护一个简单申请人状态机的单个实例。

表 12-9 描述了此状态机的详细操作。所示的状态转换处理接收 Leave In 或 Leave Empty 消息的可能性；但是，状态机只生成 Leave Empty 消息。

12.9 行政控制

与每个注册器状态机实例相关的是*注册商行政控制*参数。这些参数允许对每个属性值的注册状态进行管理控制，因此，通过 GIP 提供的传播机制，可以对声明的传播进行控制。

每个申请人状态机的总体控制参数，*申请人行政控制，*确定申请人状态机是否参与 GARP 协议交换。

表 12-9—简单申请人状态机

		状态		
		五	**一个**	**问**
事件	**传输PDU!**	sJ[我] 一个	sJ[我] 问	- x-
	加入	一个	问	问
	rJoinEmpty	五	五	五
	空	五	五	五
	离开	五	五	五
	rLeaveEmpty	五	五	五
	全部离开	五	五	五
	请求加入	- x-	- x-	- x-
	要求休假	系统性红斑狼疮 哦	系统性红斑狼疮 哦	系统性红斑狼疮 哦

这些参数可以设置为 12.9.1 和 12.9.2 中定义的值。

12.9.1 注册服务机构管理控制值

一个)*正常注册。*注册器正常响应传入的 GARP 消息。

b) *注册已修复。*注册器忽略所有 GARP 消息，并保持 IN（已注册）状态。

c) *禁止注册。*注册器忽略所有 GARP 消息，并保持 EMPTY（未注册）状态。

该参数的默认值为正常注册。

或者，实现可以支持针对每个注册器状态机记录导致该状态机最近状态改变的 GARP PDU 发起者的 MAC 地址的能力。

注：注册器管理控制是通过过滤数据库中所有 GARP 应用程序的静态条目的端口映射参数内容来实现的。对于 GMRP，相关的静态条目是静态过滤条目（7.9.1 和 10.3.2.3）。静态条目中的端口映射参数内容可以通过 14.7 中定义的管理操作进行修改。如果给定属性没有此类控制信息，则假定默认值为“正常注册”。

12.9.2 申请人行政控制值

一个)*普通参加者。*状态机正常参与GARP协议交换。

b) *非参加者。*状态机不发送任何 GARP 消息。

该参数的默认值为“普通参会者”。

注：任何 GARP 申请的申请人管理控制参数均可通过 14.9 中定义的管理操作进行修改。如果给定属性没有此类信息，则假定默认值为“普通参与者”。

12.10 程序

以下小节定义了 12.8 节中状态机描述中确定的协议动作和程序。

12.10.1 丢弃格式错误的 GARP PDU

如果以下任一情况属实，则收到 GARP PDU 的 GARP 参与者应丢弃该 PDU：

a) PDU携带未知的协议标识符；
b) 该PDU的格式不符合12.11节定义的GARP PDU格式。

GARP PDU 中包含的不能被 GARP 应用程序理解的信息项应按照 12.11.3 中的规定丢弃。

12.10.2 协议参数和计时器

12.10.2.1 联合计时器

加入周期定时器 jointimer 控制应用于申请者状态机的 transmissionPDU! 事件之间的间隔。每个端口、每个 GARP 参与者都需要此定时器的一个实例。transmitPDU! 事件之间的最大时间间隔由 JoinTime 定义，如表 12-10 中定义。

12.10.2.2 离开定时器

离开周期定时器 leavetimer 控制注册器状态机在转换到 MT 状态之前在 LV 状态中等待的时间。每个处于 LV 状态的状态机都需要一个定时器实例。离开周期定时器在启动或重新启动时设置为值 LeaveTime；LeaveTime 的定义见表 12-10。

12.10.2.3 离开定时器

Leave All 周期定时器 leavealltimer 控制 Leave All 状态机生成 LeaveAll PDU 的频率。每个端口、每个 GARP 参与者都需要该定时器。在启动或重新启动时，Leave All 周期定时器设置为 LeaveAllTime < T < 1.5*LeaveAllTime 范围内的随机值 T。LeaveAllTime 的定义见表 12-10。

12.10.3 协议事件定义

除非这些事件定义另有说明，否则网桥中的 GARP PDU 接收可以通过网桥的所有端口进行，并且由于此类接收而生成的事件仅影响与接收 PDU 的端口相关联的状态机。

12.10.3.1 请求加入

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果 GID 服务用户针对与该状态机关联的属性实例发出 GID_Join.request 服务原语，则认为已经发生 ReqJoin 事件。

12.10.3.2 请求离开

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果 GID 服务用户针对与该状态机关联的属性实例发出 GID_Leave.request 服务原语，则认为已经发生 ReqLeave 事件。

12.10.3.3 rJoinIn

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果收到 GARP PDU（12.11），且以下条件为真，则认为发生了 rJoinIn 事件：

a) 该 PDU 寻址到与状态机关联的 GARP 应用程序的 GARP 应用程序地址（表 12-1）；

b) PDU 包含一个消息（12.11.1），其中的属性类型是与状态机相关的类型；

c) 消息包含一个属性，其中属性事件（12.11.2.4）指定JoinIn事件，属性值（12.11.2.6）等于与状态机关联的值。

12.10.3.4 rJoinEmpty

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果收到 GARP PDU（12.11），并且满足以下条件，则认为发生了 rJoinEmpty 事件：

a) 该 PDU 寻址到与状态机关联的 GARP 应用程序的 GARP 应用程序地址（表 12-1）；

b) PDU 包含一个消息（12.11.1），其中的属性类型是与状态机相关的类型；

c) 消息包含一个属性，其中属性事件（12.11.2.4）指定JoinEmpty事件，并且属性值（12.11.2.6）等于与状态机关联的值。

12.10.3.5 rEmpty

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果收到 GARP PDU（12.11），并且以下条件为真，则认为发生了 rEmpty 事件：

a) 该 PDU 寻址到与状态机关联的 GARP 应用程序的 GARP 应用程序地址（表 12-1）；

b) PDU 包含一个消息（12.11.1），其中的属性类型是与状态机相关的类型；

c) 消息包含一个属性，其中属性事件（12.11.2.4）指定空事件，并且属性值（12.11.2.6）等于与状态机关联的值。

12.10.3.6 rLeaveIn

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果收到 GARP PDU（12.11），且以下条件为真，则认为发生了 rLeaveIn 事件：

a) 该 PDU 寻址到与状态机关联的 GARP 应用程序的 GARP 应用程序地址（表 12-1）；

b) PDU 包含一个消息（12.11.1），其中的属性类型是与状态机相关的类型；

c) 消息包含一个属性，其中属性事件（12.11.2.4）指定LeaveIn事件，并且属性值（12.11.2.6）等于与状态机关联的值。

12.10.3.7 rLeaveEmpty

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果收到 GARP PDU（12.11），且以下条件为真，则认为发生了 rLeaveEmpty 事件：

a) 该 PDU 寻址到与状态机关联的 GARP 应用程序的 GARP 应用程序地址（表 12-1）；

b) PDU 包含一个消息（12.11.1），其中的属性类型是与状态机相关的类型；

c) 消息包含一个属性，其中属性事件（12.11.2.4）指定Leave-Empty事件，并且属性值（12.11.2.6）等于与状态机关联的值。

12.10.3.8 离开全部

对于申请人/注册商组合状态机、仅申请人状态机、简单申请人状态机或全部离开状态机的实例，如果满足以下条件，则认为全部离开事件已经发生：

a) 收到 GARP PDU（12.11），并且以下条件均成立：

1） 该PDU 寻址到与状态机关联的 GARP 应用程序的 GARP 应用程序地址（表 12-1）；

2） PDU 包含一个消息（12.11.1），其中的属性类型是与状态机关联的类型；

3) 该消息包含一个LeaveAll属性，其中存在LeaveAll事件（12.11.2.5）。

对于申请人/注册商组合状态机、仅申请人状态机或简单申请人状态机的实例，如果满足以下条件，则认为 LeaveAll 事件已经发生：

b）与该状态机关联的 Leave All 状态机执行 sLeaveAll 动作（12.10.4.5）。

注意：LeaveAll 状态机以每个应用程序（而不是每个属性类型）为基础运行，但 LeaveAll 消息以每个属性类型为基础运行。因此，当 LeaveAll 状态机发出 LeaveAll 时，它必须为相关应用程序支持的每个属性类型生成一个 LeaveAll 属性。

12.10.3.9 离开计时器!

对于申请人/注册商组合状态机的实例，当与该状态机关联的离开计时器到期时，离开计时器！事件被视为已经发生。

12.10.3.10 离开所有计时器!

对于申请人/注册商组合状态机、仅申请人状态机、简单申请人状态机或 LeaveAll 状态机的实例，当与该状态机关联的 leavealltimer 到期时，即认为 leavealltimer！事件已经发生。

12.10.3.11 发送PDU！

对于组合申请人/注册商状态机、仅申请人状态机或简单申请人状态机的实例，当与该状态机关联的联合计时器到期时，transmitPDU！事件被视为已发生。

对于 LeaveAll 状态机的实例，当状态机有机会传输 LeaveAll 消息时，认为 transmitPDU！事件已经发生。

12.10.4 动作定义

除非这些操作定义中另有说明，否则作为网桥中状态机操作结果的 GARP PDU 传输仅通过与该状态机关联的端口进行，并且仅当该端口处于转发状态时才进行。

12.10.4.1 -x-

未采取任何措施。

12.10.4.2 sJ[E, I]，sJ[I]

传输一个 GARP PDU，其格式如 12.11.1 中定义。PDU 的格式应为

a) PDU 包含一个消息（12.11.1.2），该消息携带一个属性类型（12.11.2.2），该属性类型指定与状态机相关的类型；
b) 该消息包含一个属性 (12.11.1.2)，该属性指定一个属性事件 (12.11.2.4)，等于 JoinIn (如果注册器处于 IN 状态，或者没有实现注册器功能) 或 JoinEmpty (如果注册器处于 LV 或 MT 状态)，以及一个等于与状态机关联的值的属性值。

应使用与状态机关联的 GARP 应用程序的 GARP 应用程序地址作为目标 MAC 地址来传输 PDU。

12.10.4.3 西欧

传输一个 GARP PDU，其格式如 12.11.1 中定义。PDU 的格式应为

a) PDU 包含一个消息（12.11.1.2），该消息携带一个属性类型（12.11.2.2），该属性类型指定与状态机相关的类型；
b) 消息包含一个属性（12.11.1.2），该属性指定一个等于空的属性事件（12.11.2.4），以及一个等于与状态机关联的值的属性值。

应使用与状态机关联的 GARP 应用程序的 GARP 应用程序地址作为目标 MAC 地址来传输 PDU。

12.10.4.4 系统性红斑狼疮

传输一个 GARP PDU，其格式如 12.11.1 中定义。PDU 的格式应为

a) PDU 包含一个消息（12.11.1.2），该消息携带一个属性类型（12.11.2.2），该属性类型指定与状态机相关的类型；
b) 该消息包含一个属性（12.11.1.2），该属性指定一个等于 LeaveEmpty 的属性事件（12.11.2.4），以及一个等于与状态机关联的值的属性值。

应使用与状态机关联的 GARP 应用程序的 GARP 应用程序地址作为目标 MAC 地址来传输 PDU。

12.10.4.5 sLeaveAll

传输一个 GARP PDU，其格式如 12.11.1 中定义。PDU 的格式应使得对于与 GARP 应用相关的每个属性类型

a) PDU 包含一个消息（12.11.1.2），该消息携带一个指定相关属性类型的属性类型（12.11.2.2）；

b) 该消息包含 LeaveAll 属性（12.11.1.2）。

应使用与状态机关联的 GARP 应用程序的 GARP 应用程序地址作为目标 MAC 地址来传输 PDU。

sLeaveAll 操作还会针对与 GARP 应用程序关联的申请人/注册商组合状态机、仅申请人状态机和简单申请人状态机的所有实例引发 LeaveAll 事件。

12.10.4.6 启动离开定时器

促使 leavetimer 启动，符合 12.10.2.2 中定时器的定义。

12.10.4.7 停止离开定时器

导致 leavetimer 停止。

12.10.4.8 启动leavealltimer

导致leavealltimer启动，符合12.10.2.3中定时器的定义。

12.10.4.9 IndJoin

当注册器状态机的实例从 LV 或 MT 状态转换到 IN 状态时，IndJoin 操作会导致向 GID 服务用户发出 GID_Join.indication 原语，指示与相关状态机相对应的属性实例。

12.10.4.10 独立休假

当注册器状态机的实例从 LV 状态转换到 MT 状态时，IndLeave 操作会导致向 GID 服务用户发出 GID_Leave.indication 原语，指示与相关状态机相对应的属性实例。

12.10.4.11 未登记

每个 GARP 参与者都会记录收到注册请求但由于过滤数据库中没有足够的空间记录注册而无法注册相关属性的次数。此计数的值可能会被管理层检查。

注——针对此类事件采取的进一步行动取决于实施选择。

12.11 GARP 协议数据单元的结构和编码

本小节描述了所有 GARP 参与者之间交换的 GARP 协议数据单元 (GARP PDU) 的通用结构和编码。GARP 应用程序操作所特有的元素的结构和编码由应用程序本身定义。

每个 GARP PDU 都标识了生成它的 GARP 应用程序以及要将其传输到的 GARP 应用程序。如果网桥接收到的 GARP PDU 被标识为属于其不支持的 GARP 应用程序，则应在处于转发状态的所有其他端口上转发此类 PDU。

注 1 — 如果使用 GARP 支持可以在任何 GIP 上下文（非 0，即基本生成树）中运行的应用程序，则在协议交换中识别该上下文的方法由相关的 GARP 应用程序定义。

每个 GARP PDU 都携带一个或多个 GARP 消息，每个消息都标识一个 GARP 事件（例如 Join、Leave、LeaveAll）以及该事件适用的属性类和值。给定的 GARP 参与者应按照接收 GARP PDU 的顺序处理它们，并且在给定的 GARP PDU 中，应按照 GARP 消息放入数据链路服务数据单元 (DLSDU) 的顺序处理它们。

注 2 — 作为对 sLeaveAll 操作及其相关 LeaveAll 事件的状态机响应的结果而生成的任何消息都将被放入 LeaveAll 消息之后的 DLSDU 中，或放入更靠后的 DLSDU 中。

12.11.1 结构

12.11.1.1 八位字节的传输和表示

所有 GARP PDU 均由整数个字组成，从 1 开始编号，并按照放入数据链路服务数据单元 (DLSDU) 的顺序递增。每个八位字节中的位编号从 1 到 8，其中 1 是低位。

当使用连续的八位字节来表示二进制数时，较低的八位字节数具有最高有效值。

当使用本节中的图表表示 GARP PDU（的元素）的编码时，使用以下表示法：

a) 八位字节 1 显示在页面顶部，编号较高的八位字节显示在页面底部；
b) 当某一行出现多个八位字节时，八位字节按以下方式显示：编号最小的八位字节在左侧，编号较高的八位字节在右侧；
c) 在八位字节内，位显示为左侧为位 8，右侧为位 1。

12.11.1.2 结构定义

协议标识符应编码在所有 GARP PDU 的初始八位字节中。本标准保留一个协议标识符值来标识通用属性注册协议。运行本条款中指定的协议的 GARP PDU 携带此保留的协议标识符值，并应具有以下结构：

a) 前两个八位字节包含*协议标识符*价值。
b) 协议标识符后面是一个或多个*消息*. PDU 中的最后一个元素是*结束标记*。

c) 每条消息由一个*属性类型*和一个*属性列表*，按此顺序。

d) 属性列表由一个或多个*属性*s. 属性列表中的最后一个元素是结束标记。

e) 属性由*属性长度*，一个*属性事件*，以及*属性值*，按此顺序。如果属性事件为“LeaveAll”，则省略属性值。

以下 BNF 产生式给出了 GARP PDU 结构的正式描述：

GARP PDU ::= 协议 ID，消息 {，消息}，结束标记协议 ID
SHORT ::= 1
消息 ::= 属性类型，属性列表
属性类型 BYTE ::=*由特定 GARP 应用程序定义* 属性列表 ::= 属性 {,
属性}, 结束标记属性 ::= 普通属性 | LeaveAll 属性

普通属性 ::= 属性长度，属性事件，属性值 LeaveAll 属性 ::= 属性长
度，LeaveAll 事件
属性长度 BYTE ::= 2-255
属性事件 BYTE ::= JoinEmpty | JoinIn | LeaveEmpty | LeaveIn | Empty
LeaveAll 事件 BYTE ::= LeaveAll
属性值 ::=*由特定 GARP 应用程序定义* 结束标记 ::= 0x00 |
PDU 结束 全部离开 ::= 0

加入空 ::= 1
加入 ::= 2
留空 ::= 3
离开 ::= 4
空 ::= 5

如本结构定义所标识的 GARP PDU 中携带的参数应按照 12.11.2 中规定的方式进行编码。

图 12-6 说明了 GARP PDU 及其组件的结构。

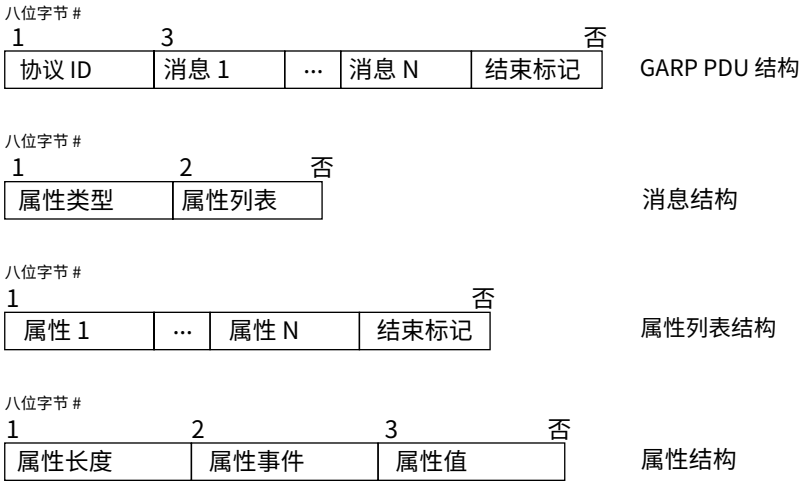


图 12-6 - GARP PDU 主要组件的格式

12.11.2 GARP PDU 参数的编码

12.11.2.1 协议标识符的编码

协议标识符应编码为两个八位字节，表示无符号二进制数。它采用十六进制值 0x0001，用于标识本节定义的 GARP 协议。

12.11.2.2 属性类型的编码

属性类型应编码为单个八位字节，用于表示无符号二进制数。属性类型标识消息适用的属性类型。属性类型可采用的值范围以及这些值的含义由相关应用程序定义。值 0 是保留的，任何 GARP 应用程序都不应将其用作属性类型。否则，GARP 应用程序可能会为 1 到 255 范围内的任何一组属性类型值分配含义。

12.11.2.3 属性长度的编码

属性长度应编码为单个八位字节，表示为无符号二进制数，等于属性占用的八位字节数（包括属性长度字段）。属性长度的有效值范围为 2 至 255。

属性长度的其它值被保留，不得使用。

12.11.2.4 属性事件的编码

属性事件应编码为单个八位字节，表示无符号二进制数。属性事件的允许值和含义如下：

1： JoinEmpty 运算符
2： JoinIn 运算符
3： LeaveEmpty 运算符
4： LeaveIn 运算符
5： 空运算符

属性事件的其它值被保留。

属性事件在接收时被解释为 GID 事件，并应用于由属性类型和属性值字段定义的属性的状态机。

12.11.2.5 LeaveAll 事件的编码

LeaveAll 事件应编码为单个八位字节，表示无符号二进制数。LeaveAll 事件的允许值和含义如下：

0： LeaveAll 运算符

LeaveAll 事件的其它值被保留。

在接收时，LeaveAll 事件被解释为 GID Leave All 事件，该事件将应用于由属性类型字段定义类型的所有属性的状态机。

12.11.2.6 属性值的编码

属性值以 N 个八位字节进行编码，符合相关 GARP 应用程序定义的属性类型规范。

12.11.2.7 结束标记的编码

结束标记应编码为单个八位字节，用于表示无符号二进制数。其值为 0。

结束标记的其它值被保留，不得使用。

注 — 如 12.11.1 中的 GARP PDU 结构定义所定义，如果遇到 GARP PDU 的末尾，则从处理 PDU 内容的角度来看，这被视为结束标记。

12.11.3 打包和解析GARP PDU

使用结束标记（12.11.2.7）来表示属性列表的结束和 GARP PDU 的结束，以及将 PDU 的（物理）结束解释为结束标记的事实，简化了将信息打包到 GARP PDU 中以及在接收时正确解释该信息的要求。

12.11.3.1 包装

连续的消息被打包到 GARP PDU 中，并且在每个消息中，连续的属性被打包到每个消息中，直到遇到 PDU 的末尾或当时没有更多属性可打包。可能会发生以下情况：

a) PDU 有足够的空间来打包所有需要当时传输的属性。在这种情况下，PDU 被传输，当有其他属性需要传输时，才会传输后续的 PDU；

b) PDU 有足够的空间来打包当时需要传输的前 N 个属性。在这种情况下，PDU 被传输，而接下来的 N 个属性被编码在后续的 PDU 中；

c) PDU 有足够的空间来打包当时需要传输的前 N 个属性，但 PDU 中的剩余空间对于属性 N+1 来说太小，因此 PDU 的最后几个八位字节携带属性 N+1 的部分编码。在这种情况下，PDU 可以按原样传输，属性 N+1 及其后继者将在后续 PDU 中完整编码。

12.11.3.2 解析

连续的消息以及每个消息中的连续属性从 PDU 中解包。如果此过程因到达 PDU 末尾而终止，则 PDU 的末尾将用于发出信号，表示当前属性列表和整个 PDU 都已终止。可能发生两种情况：

a) 最后一个要解包的属性已完成。在这种情况下，该属性将正常处理，并且 PDU 的处理终止；

b) 最后一个要解包的属性不完整。在这种情况下，部分属性将被丢弃，并且 PDU 的处理终止。

12.11.3.3 丢弃无法识别的信息

为了能够与给定 GARP 应用程序的先前版本保持向后兼容，在收到的 GARP PDU 中遇到无法识别的元素时，将采用以下程序：

a) 如果遇到无法识别属性类型的消息，则丢弃该消息。这是通过丢弃属性列表中的连续属性来实现的，直到到达结束标记或 PDU 的末尾。如果到达结束标记，则继续处理下一条消息。

b) 如果遇到一个属性，其中的属性事件对于相关的属性类型来说是无法识别的，那么该属性将被丢弃，并且如果尚未到达 PDU 的末尾，则继续处理下一个属性或消息。

12.12 计时器值、粒度和关系

12.12.1 计时器值

GARP 协议中使用的默认计时器值在表 12-10 中定义。GARP 计时器使用的值可通过第 14 条中定义的管理功能逐个端口进行修改。

表 12-10 — GARP 计时器参数值

范围	值（厘秒）
加入时间	20
离开时间	60
全部离开	1000

注意：GARP 计时器的默认值与媒体访问方法或数据速率无关。这是经过深思熟虑的选择，旨在最大限度地发挥协议的“即插即用”特性。

12.12.2 定时器分辨率

GARP 计时器的实现应基于 5 厘秒或更小的计时器分辨率。

12.12.3 时间关系

GARP 协议*正确性*并不严重依赖于时间关系；但是，如果在同一 LAN 段上交换 GARP PDU 的状态机中运行的协议计时器之间保持以下关系，则协议运行得更有效率，并且发生不必要的取消注册的可能性更小：

a) 应选择 JoinTime，使得在 LAN 段上使用的 LeaveTime 值内至少可以出现两个 JoinTime。这确保在发出 Leave 或 LeaveAll 消息后，申请人可以在发出另一条 Leave 消息之前重新加入；

b) LeaveAllTime 应大于 LAN 段上使用的 LeaveTime 值。为了尽量减少 Leave All 之后产生的重新加入流量，为 LeaveAllTime 选择的值应大于 LeaveTime。

这些关系如图 12-7 所示。标为 A（离开时间减去两个加入时间）和 B（离开所有时间减去离开时间）的时间间隔应全部为正值且非零，以确保 GARP 有效运行。表 12-10 中指定的时间参数值已被选定，以确保维持这些计时器关系。

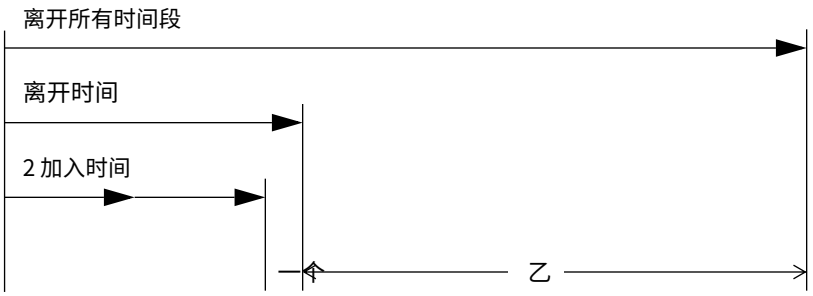


图 12-7 — GARP 时序关系

12.13 程序模型

第 13 节包含 GARP 的示例“C”代码描述。

12.14 互操作性考虑

对于给定的 GARP 应用程序，GARP 协议的正确运行要求给定的通信 GARP 参与者集之间的协议交换保持顺序性；即，参与者 A 不能在参与者 A 收到 GARP PDU A 之前收到 GARP PDU B（参与者 B 收到 GARP PDU A 后生成）。在相关参与者都连接到同一 LAN 段的情况下，可以确保这种顺序性。但是，如果一组 GARP 参与者通过未实现该 GARP 应用程序（或根本不实现 GARP）的中间桥进行通信，则 7.7.3 中表达的顺序性约束不足以保证 GARP 协议的正确运行。为了通过这样的桥保持 PDU 的正确排序，必须满足以下约束：

如果 GARP PDU A 在端口 X 上收到，并且将在端口 Y 和 Z 上转发，并且在 Y 上转发之后，GARP PDU B 在端口 Y 上收到并将在端口 Z 上转发，则 B 的转发不能先于 A 在端口 Z 上进行。

注——这表示对多播帧的排序约束比 7.7.3 中规定的更强，但比符合 ISO/IEC 10038: 1993 的要求的约束更弱。

不满足此约束的后果是，特定 GARP 应用程序的用户可能会遇到注册丢失的概率增加。因此，如果中间网桥不满足上述约束，则不建议构建涉及通过中间网桥转发 GARP PDU 的 LAN 配置。

13. GARP 的 C 代码实现示例

13.1 目的

本节包含 GARP 协议和状态机制的示例实现，这些实现和状态机制是支持 GMRP（条款 10）和其他使用 GARP 的应用程序（条款 12）所必需的。包含此“C”代码描述是为了演示 GARP 及其组件的结构，并表明可以构建一个成本合理的实现。实现的设计目的是最大限度地提高清晰度和通用性，而不是为了紧凑性。

示例实现如下子条款所示：

a) GARP 应用程序独立代码的头文件（13.2）；
b) GARP 应用独立代码（13.3）。

实现中应用程序相关（第 11 条）和应用程序无关（本条）方面的区分清楚地说明了使用相同的基本 GARP 状态机实现其他应用程序所涉及的内容。代码旨在通过内联注释实现自我文档化。

注意——所显示的代码基于为完整 GARP 参与者定义的功能和状态机；仅申请人参与者和简单参与者的代码将涉及所示功能的简化，如 12.7.7 和 12.7.8 中所述。

13.2 GARP 应用程序独立头文件

13.2.1 garp.h

```
/* 加普.h * /
# 如果定义 garp_h__
＃ 定义 garp_h__

#包括“sys.h”

/********************************************************** ****************************
 * GARP：通用属性注册协议：通用应用元素
 ******************************************************************************
 * /

typedef struct /* Garp */ { /*

    * 每个 GARP，即使用 GARP 的应用程序的每个实例
    * 协议，表示为具有共同初始值的结构或控制块
    * 字段。这些字段包括指向应用程序特定函数的指针，这些函数
    * 由 GID 和 GIP 组件向
    * 应用程序，以及所有应用程序共有的其他控件。指针
    * 包含指向应用程序的 GID 实例（每个端口一个）的指针，
    * 和 GIP（每个应用程序一个）。信令功能包括
    * 添加和删除端口，应用程序应该使用这些端口来
    * 使用所需的任何管理状态初始化端口属性。
    * /

    整数          进程ID；

    空白          * 群组识别码；

    未签名        * 吉普；

    未签名          最大gid索引；

    未签名          上次使用的gid；

    void（*join_indication_fn）（void *，void *my_port，无符号joining_gid_index）；void
     （*leave_indication_fn）（void *，void *gid，
                                                  未签名       离开_gid_index）；
    void（*join_propagated_fn）（void *，void *gid，
                                                  未签名       加入_gid_index);
    void（*leave_propagated_fn）（void *，void *gid，
                                                  未签名       离开_gid_index）；

    无效（*transmit_fn）（              无效*，无效* gid）；
    无效（*接收函数）（                    无效*，无效*gid，Pdu*pdu）；

    空白          (*已添加端口函数）（          void *， int port_no); void
    空白          (*删除了端口fn）（            *， int port_no);

} 卡普；

# 结束语 /* garp_h__ */
```

13.2.2 gid.h

```
/*gid.h * /
# 如果定义 gid_h__
＃ 定义 gid_h__
```

```
# 包括 “系统.h”
# 包括 “garp.h”


/**************************************************************** ****************************
 * GID：GARP 信息分发协议：概述
 ********************************************************************************************
 *
 * 在此参考实现中，有一个 GID 实例，
 * 每个应用程序实例的 GARP 信息分发协议
 * 系统中的物理或逻辑端口[基本生成树运行
 * 通过物理端口；VLAN 注册通过物理端口进行
 * 每个 VLAN 的每个物理端口提供一个（或零个）逻辑端口；一个实例
 * 多播注册可以通过物理端口或
 * VLAN 的逻辑端口，取决于注册是否在
 * 是否属于 VLAN 的范围。
 *
 * 这个单一的 GID 实例运行着多个 GID 机器，每个 GID 机器对应一个
 * 应用程序实例感兴趣的属性（属性是
 * GARP 可以注册的最小单位，例如单个多播
 * 地址——加入和离开指示针对单个属性发生）。
 * GID 对单个属性的语义一无所知——它只是
 * 对每个属性的 GID 机器的状态以及
 * GID 事件会改变该状态并引发进一步的协议
 * 事件。每个属性都由其 GID 索引标识，在本例中
 * 实现是 GID 机器数组的简单索引。
 * 属性的 GID 索引对于属于应用程序的每个端口都是相同的
 * 实例。
 *
 * 操作单个 GID 实例而不是完全独立的实例
 * machines 允许每个端口有一组 GID 计时器，并且
 * 方便将单个属性的消息打包成
 * 单个 PDU。
 * /

/**************************************************************** ****************************
 * GID：GARP 信息分发协议：GID 机器
 ********************************************************************************************
 * /

typedef struct /* Gid_machine */ {

    /* 每个端口上每个属性的 GID 状态保存在 GID“机器”中
     * 包括港口的申请人国和注册国，
     * 包括各州的控制修饰符。
     *
     * GID 机器及其 GID 状态的内部表示不是
     * 直接访问：此结构在此处定义以允许 GID
     * 控制块（可从外部访问）定义如下。
     * /

     未签名的申请人：5;                /*：申请人状态*/

     未签名的注册员：5;                /*：注册商状态*/

}  Gid_机器;

/**************************************************************** ****************************
 * GID：GARP 信息分发协议：管理状态
 ********************************************************************************************
 *
 * 实现独立于单个 GID 状态的表示
 * 属性包括管理控制，遵循标准状态
 * 机器规格如下：
 *
```

```
 * 申请人： 主要州：                                    非常焦虑、焦虑、安静、离开
 *
 *                    当前参与：主动、被动
 *
 *                     管理控制                    ：    正常，无协议
 *
 * 注册处： 主要州：                                    进入、离开、清空
 *
 *                     管理控制                    ：    正常注册，
 *                                                   注册已修复，
 *                                                   禁止注册
 *
 * 定义一个结构体来返回当前状态（调用者提供
 * 参考）。
 * /

typedef enum {Very_anxious, Anxious, Quiet, Leaving} Gid_applicant_state; typedef enum {Normal,
No_protocol} Gid_applicant_mgt;
typedef enum {In，Leave，Empty}                                          请参阅Gid_registrar_state;
typedef enum {Normal_registration，Registration_fixed，Registration_forbidden
                  }                                                      Gid_registrar_mgt;

typedef struct /* Gid_states */ {

    未签名的申请人状态：2;

    未签名的申请人_mgt              ：1;

    无符号registrar_state: 2;

    未签名的 registrar_mgt          ：2;

}  Gid_状态;

/********************************************************* ****************************
 * GID：GARP 信息分发协议：协议事件
 ******************************************************************************************
 *
 * 指定了完整的事件集，以允许事件传递
 * 无需重写。另一个极端是
 * 接收消息的事件集、接收指示集、
 * 一组用户请求等，以及每个请求的单独的转换表。
 *
 * 事件定义是有序的，以避免浪费空间
 * 转换表并将开关盒打包在一起。
 * /
typedef enum/*Gid_event*/{


    Gid_null、Gid_rcv_leaveempty、Gid_rcv_leavein、Gid_rcv_empty、
    Gid_rcv_joinempty、Gid_rcv_joinin、
    Gid_join，Gid_leave，
    Gid_normal_operation，Gid_no_protocol，
    Gid_normal_registration、Gid_fix_registration、Gid_forbid_registration、Gid_rcv_leaveall、
    Gid_rcv_leaveall_range、
    Gid_tx_leaveempty、Gid_tx_leavein、Gid_tx_empty、Gid_tx_joinempty、Gid_tx_joinin、
    Gid_tx_leaveall、Gid_tx_leaveall_range
}  Gid_事件;

枚举 {Number_of_gid_rcv_events          = (Gid_rcv_joinin + 1), =
        Gid_req_events 数量                 2、
        Gid_amgt_events 数量            =  2、
        Gid_rmgt_events 数量            =  3、
        Gid_tx_events 数量              =  7};
```

```
/**************************************************** ****************************
 * GID：GARP 信息分发协议：协议计时器值
 ********************************************************************************
 *
 * 描述计时器的目标、亚秒级响应时间等。
 * /
typedef int 毫秒；

枚举 {Gid_default_join_time 枚举          =    200}; /* 毫秒 */ 600}; /* 毫秒 */
{Gid_default_leave_time 枚举              =    100}; /* 毫秒 */
{Gid_default_hold_time 枚举               =
{Gid_leaveall_count                       =      4}；
枚举{Gid_default_leaveall_time = 10000}; /* 毫秒 */

/**************************************************** ****************************
 * GID：GARP 信息分发协议：协议实例
 ********************************************************************************
 * /

typedef struct /* Gid */ {

    /* 每个 GID 实例都由其中一个控制块表示。
     * 每个 GARP 应用程序的每个端口都有一个 GID 实例。
     * 控制块通过 next_in_port_ring 链接在一起
     * 指针形成一个完整的环。
     *
     * 每个控制块包含一个指向 GARP 控制块的指针
     * 表示指定应用程序的应用程序实例
     * 加入、离开、传输和接收函数及其进程标识符
     *（用于向系统其余部分标识应用程序实例
     * 包括计时器功能）。
     *
     * 指定与此 GID 实例关联的端口号。
     * 应用程序实例的所有端口的 GID 控制块
     * 通过 next_in_port_ring 指针链接在一起形成一个
     * 完整的环。“连接”的端口，例如，都在一个生成环中
     * 树的转发状态，通过next_in_connected_ring链接起来
     * 指针。这些端口还设置了 is_connected 标志，并且它
     * 处理单个连接端口的情况 [is_connected 为 true (设置)]
     * 仅当 is_enabled 也为真时。GID 控制块定义是
     * 共享以允许 GIP 遍历这些字段。
     *
     * 可以启用或禁用整个端口的 GID 处理：
     * 当前状态记录在这里，以防收到 PDU 或其他事件
     * 仍在系统中待处理。
     *
     * 标准 GID 的运行方式就像 GID 实体连接到一个共享
     * 中等。如果链路
     * 是点对点的。特别是收到的 Leave 消息
     * 可以立即发出离开指示，而无需
     * 进一步征求共享网络中其他潜在成员的加入
     * 中等的。
     *
     * 控制块提供了一个用于记录操作的“暂存器”
     * 在本次 GID 调用期间，由 GID 机器处理引起
     *（“调用”表示 GARP 运行时，例如，当收到一个数据包时
     * 正在处理）。每次调用 gid_do_actions() 后都会被调用
     * 要立即安排传输，请启动加入计时器（如果
     * 尚未启动），或者启动离开计时器（再次，如果尚未启动
     * 已启动）取决于 cschedule_tx_now 的设置，
     * cstart_join_timer 和 cstart_leave_timer，以及这些计时器
     * 已开始（或请求立即安排）
     * 记录在 tx_now_scheduled、join_timer_running 和
     * leave_timer_running. 如果 hold_tx 为真，则调度并启动计时器
     * 将被保留，直至保持计时器到期。
```

```
 *
 * join、leave、hold 和 leaveall 计时器的超时值为
 * 记录在这里以便根据媒体类型进行管理
 * 和速度，以及端口是否连接到点对点交换机到
 * 交换机链接至共享媒体，或通过非 GARP 感知交换机至
 * 可能具有不同链接速度的交换机。
 *
 * 单个 GID 机器的休假时间为三到四
 * leave_timeout_4 的到期时间。这提供了足够的粒度
 * 为休假定时器提供保护，以防止过早到期（而
 * 另一个 GID 参与者可能正准备发送加入消息）而不
 * 最坏情况下的离开时间过长。它也足够粗糙
 * 允许轻松存储每个条目的离开计时器状态
 * 注册处状态。
 *
 * leaveall 超时实现为 leaveall_countdown 过期
 * leaveall_timeout_n。这支持抑制 Leaveall 生成
 * 当收到 Leaveall 时，由这台机器执行，无需
 * 操作系统支持取消或重新启动计时器。
 * 当 leaveall_countdown 达到零时，启动加入计时器（如果没有
 * 已在运行）。每当加入计时器到期时，应用程序的
 * 调用传输函数，这将导致调用 gid_next_tx，
 * 反过来又会返回 Gid_tx_leaveall。这确保 Leavealls
 * 在包含结果的 PDU 开头传输
 * 本地 Leaveall 处理 - 例如立即重新加入。这是一个
 * 在接收方数据包丢失的情况下，实现协议稳健性的好方法
 * - 仅当 Leaveall 本身丢失时，重新加入才会丢失 - 而且它
 * 最小化发送的 PDU 数量。
 *
 * 此 GID 控制块指向已分配的 GID 机器的数组
 * 当创建此 GID 实例时。它支持更多 GID 属性
 * 可以通过应用程序格式化程序打包成单个 PDU
 *（大多数 4096 个 VLAN 的简单编码都需要）。tx_pending
 * 标志表示 last_transmitted 之间的一些 GID 机器
 * 和 last_to_transmit 索引可能有要发送的消息。
 * last_transmitted 消息生成之前的应用程序状态
 * 机器存储在由 untransmit_machine 索引的 GID 机器中 -
 * 该空间保留在 GID 机器数组的最末端，
 * 这比存储所需的大一个
 * 在创建 GID 时指定的最大属性数。这允许
 * 实现简单的取消传输功能，类似 C
 * 库函数 ungetc()。
 * /

卡普       * 应用;

整数              端口号;

空白             *下一个端口环;

空白             *下一个_在_连接_环中;

未签名           已启用                  : 1;

未签名           已连接                  : 1;

未签名           是点到点                : 1;

未签名           cschedule_tx_now        : 1;

未签名           启动加入定时器          : 1;

未签名           启动离开定时器          : 1;

未签名           tx_now_scheduled        : 1;
```

```
        未签名          join_timer_running      ： 1；
        未签名          离开计时器运行          ： 1；
        未签名          hold_tx                 ： 1；
        未签名          tx_pending              ： 1；
        整数            加入超时；
        整数            离开超时4；
        整数            保持超时；
        整数            离开所有_倒计时；
        整数            离开所有超时时间；
        Gid_machine     *机器；
        未签名          最后传输；
        未签名          最后发送；
        未签名          取消传输机器；
} 吉德；

/************************************************************ ****************************
 * GID：GARP信息分发协议：创建、销毁等。
 ********************************************************************************************
 * /

外部布尔 gid_create_port(Garp *application，int port_no)；
    /*
     * 创建一个新的 GID 实例，为 GID 机器分配空间
     * 应用程序需要，将端口添加到端口环中
     *，并向应用程序发出信号，告知新端口
     * 已创建。
     *
     * 创建时，每个 GID 机器设置为正常运行或
     * 无协议！！
     *
     * 端口在创建时启用，但未连接（参见 GIP）。
     * /

外部无效 gid_destroy_port（Garp *应用程序，int port_no）；
    /*
     * 销毁 GID 实例，断开端口（如果仍然
     * 连接（使叶子按要求繁殖），然后导致
     * 按要求留下此端口的指示，最后释放所有
     * 分配空间，向应用程序发出信号，表示端口已经
     * 已删除。
     * /

/************************************************************ ****************************
 * GID：GARP 信息分发协议：有用的功能
 ********************************************************************************************
 * /

外部布尔 gid_find_port(Gid *first_port，int port_no，void **gid)；
    /*
     * 查找端口号 port_no 的 GID 实例。
     * /
```

```
外部 Gid *gid_next_port(Gid *this_port);
    /*
     * 在端口环中查找此应用程序的下一个端口。
     * /

外部布尔 gid_find_unused(Garp *应用程序，无符号 from_index，
                                                    无符号*found_index);
    /*
     * 查找未使用的 GID 机器（即，具有空注册器和
     * 非常焦虑的观察者申请人）从 GID 索引开始搜索
     * from_index，并搜索到gid_last_used。
     * /


/************************************************************ ******************************
 * GID：GARP 信息分发协议：MGT
 ********************************************************************************************
 * /

外部 void gid_read_attribute_state(Gid *my_port，无符号索引，
                                        Gid_states *状态);
    /*
     *
     * /

外部 void gid_manage_attribute(Gid *my_port，无符号索引，
                                    Gid_event 指令）；
    /*
     * 更改 my_port 上属性的管理状态。该指令
     * 可以是 Gid_normal_operation、Gid_no_protocol、Gid_normal_registration，
     * Gid_fix_registration 或 Gid_forbid_registration。如果
     * 管理状态导致离开指示，发送给用户
     *并传播到其他端口。
     * /

/*
 * 进一步的管理功能，包括禁用和启用 GID 端口，
 * 必须提供。本参考文献的目的很大一部分
 * 实施的目的是为了促进此类
 * 管理操作。
 * /

/************************************************************ ******************************
 * GID：GARP 信息分发协议：事件处理
 ********************************************************************************************
 * /

外部无效 gid_rcv_msg(Gid *my_port，无符号 gid_index，Gid_event msg);
    /*
     * 仅适用于 Gid_rcv_leave、Gid_rcv_empty、Gid_rcv_joinempty、Gid_rcv_joinin。
     * 请参阅 gid_rcv_leaveall 了解 Gid_rcv_leaveall、Gid_rcv_leaveall_range。
     *
     * Joinin 和 JoinEmpty 可能导致 Join 指示；此函数调用
     * GIP 来传播这些。
     *
     * 在共享介质上，Leave 和 Empty 不会引起指示
     * 立即。然而，这个例程确实测试并传播
     * 保留指示，以便可以不加改变地使用点对点
     * 协议增强。
     * /

外部无效 gid_join_request（Gid *my_port，无符号 gid_index）；
    /*
     * 可以多次调用而不会产生任何不良影响。
```

```
      *
      * /

外部无效 gid_leave_request（Gid *my_port，无符号 gid_index）；
     /*
      * 可以多次调用而不会产生任何不良影响。
      *
      * /

外部无效 gid_rcv_leaveall（Gid *my_port）；
     /*
      * /

外部布尔 gid_registered_here(Gid *my_port，无符号 gid_index);
     /*
       * 如果注册器不为空或者注册已固定，则返回 True。
      * /

/*********************************************************** ****************************
 * GID：GARP 信息分发协议：计时器处理
 ********************************************************************************************
 * /

无效 gid_do_actions（Gid *my_port）；
     /*
      * 执行本次 GID 调用中积累的“暂存器”操作，
      * 和未完成的‘立即’传输和加入计时器启动
      * 已被保持计时器的操作延迟。
      * /

/*
 * 由 GID 启动的定时器的定时器到期例程，主要由
 * gid_do_actions()。join_timer_expired() 立即由
 * gid_do_actions()。
 * /
外部无效 gid_leave_timer_expired（ 外部无效          void *应用程序，int port_no); void *应用程
gid_join_timer_expired（                              序，int port_no);
外部无效 gid_leaveall_timer_expired（void *application，int port_no）；外部无效
gid_hold_timer_expired（void *application，int port_no）；


/*********************************************************** ****************************
 * GID：GARP 信息分发协议：传输处理
 ********************************************************************************************
 * /

外部Gid_event gid_next_tx(Gid *my_port，无符号*gid_index);
     /*
      * 扫描 GID 机器以查找需要传输的消息。
      * 如果没有消息需要传输，则返回Gid_null；否则，
      * 将消息作为 Gid_event 返回。
      *
      * 如果消息传输当前处于暂停状态（暂停状态即将到期
      * 计时器并调用 gid_release_tx()]，该函数将返回 Gid_null
      * 因此如果方便的话可以安全地调用它。
      *
      * 支持可以生成比
      * 可以适合单个应用程序 PDU。这允许此实现
      * 例如，支持所有 4096 个 VLAN 标识符的 GARP 注册。
      *
      * 支持应用程序将消息打包成单个 PDU
      * 无需详细了解帧格式和消息编码
      * 需要判断消息是否合适，并避免应用
      * 必须对每条消息进行两次 GID 调用 - 一次用于获取
      * 消息和另一条消息表明它已被占用 - 此例程支持
```

```
      * gid_untx() 函数。这在概念上类似于 C 库
      * ungetc() 函数。它恢复上一台机器的申请人状态
      * 从中获取了一条消息，并将该机器确立为下一个
      * 其中一个应该被接受的信息 - 有效地推动
      * 消息返回到 GID 机器集。它只应该用于
      * 在一次 GID 调用中；否则，中间事件及其
      * 随之而来的状态改变可能会丢失，并产生不明结果。
      * /

外部无效 gid_untx (Gid *my_port);
     /*
      * 参见上文描述。
      * /

# 结束语/*gid_h__*/
```

13.2.3 gidtt.h

```
/*gidtt.h          * /
# 如果定义   gidtt_h__
＃ 定义     gidtt_h__

# 包括“gid.h”


/****************************************************** ****************************
 * GIDTT：GARP 信息分发协议：转换表
 ********************************************************************************
 * /

外部 Gid_event gidtt_event(Gid                          *我的端口，
                           Gid_machine    * 机器，
                           事件                事件）；

外部 Gid_event gidtt_tx(Gid                          *我的端口，
                        Gid_machine    * 机器）；

外部 Gid_event gidtt_leave_timer_expiry(Gid                          *我的端口，
                                        Gid_machine    * 机器）；

外部布尔 gidtt_in (Gid_machine *machine);
     /*
      * 如果注册员在场或者注册已修复，则返回 True。
      * /

外部布尔 gidtt_machine_active(Gid_machine *machine);
     /*
      * 当且仅当注册处为正常注册、为空且
      * 申请为普通会员，非常焦虑的观察者。
      * /

外部无效 gidtt_states（Gid_machine *machine，
                       Gid_states *状态);
     /*
      * 报告 GID 机器状态：Gid_applicant_state，
      *Gid_applicant_mgt，Gid_registrar_state，Gid_registrar_mgt。
      * /

# 结束语/*gidtt_h__*/
```

13.2.4 gip.h

```
/* gip.h * /
# 如果定义 gip_h__
# 定义 gip_h__

# 包括“gid.h”

/********************************************************** ****************************
 * GIP：GARP信息传播：创造，毁灭
 ********************************************************************************************
 * /

外部布尔 gip_create_gip (无符号 max_attributes，无符号 **gip)；
    /*
     * 创建一个新的 GIP 实例，为传播计数分配空间
     * 最多可达 max_attributes。
     *
     * 如果创建成功则返回 True，同时返回指向
     * GIP 信息。此指针传递给端口的 gid_create_port()
     * 使用此 GIP 实例并与 GID 信息一起保存。
     * /

外部void gip_destroy_gip（void *gip）；
    /*
     * 销毁 GIP 实例，释放先前分配的空间。
     * /

/********************************************************** ****************************
 * GIP：GARP 信息传播：传播函数
 ********************************************************************************************
 * /

外部无效 gip_connect_port（Garp *应用程序，int port_no）；
    /*
     * 找到端口，检查它是否已经连接，然后连接
     * 将其放入使用 GID 控制中的 GIP 字段的 GIP 传播环中
     * 块将源端口链接到信息所在的端口
     * 得以传播。
     *
     * 从其他已连接的端口传播连接，如下所示
     * 必要的。
     * /

外部无效 gip_disconnect_port（Garp *应用程序，int port_no）；
    /*
     * 检查端口是否已连接，然后断开与
     * GIP 传播环。将叶子传播到其他端口，
     * 留在环中并在必要时导致离开 my_port。
     * /

外部无效 gip_propagate_join（Gid *my_port，无符号索引）；
    /*
     * 传播单个属性的连接指示（已识别
     * 通过其属性类和索引的组合）从 my_port 到其他
     * 端口，如果需要，会向其他端口发出加入请求。
     *
     * GIP 维护每个已连接端口的加入成员计数
     * 属性（在给定上下文中），以便在连接时不会导致叶子
     * 来自其他港口的人员将保留会员资格。
     *
     * 因为这个计数是通过“航位推算”来维持的，所以它很重要
     * 仅当有变化迹象时才调用此函数
     * 源端口和索引。
```

```
     * /

外部无效 gip_propagate_leave（Gid *my_port，无符号索引）；
    /*
     * 传播单个属性的离开指示，导致离开
     * 如果需要，向其他端口发出请求。
     *
     * 在进一步阅读之前，请参阅 gip_propagate_join() 的注释。
     * 此功能减少“航位推算”成员数量。
     * /

外部布尔 gip_propagates_to(Gid *my_port，无符号索引);
    /*
     * 如果任何其他端口正在传播与索引关联的属性，则为 True
     * 到我的端口。
     * /

外部无效 gip_do_actions（Gid *my_port）；
    /*
     * 调用 GID 执行此过程中积累的 GID“暂存器”操作
     * 对 GIP 传播列表中的所有端口调用 GARP，
     * 包括源端口。
     * /

# 结束语/* gip_h__ */
```

13.2.5 prw.h

```
/* prw.h      * /
# 如果定义   prw_h__
＃ 定义     prw_h__

#包括“sys.h”

/************************************************************ ****************************
 * PRW：PDU 读写访问
 ********************************************************************************************
 * /

类型定义  结构
{
     普杜 *协议数据单元;

     整数        记录ID;

     整数        记录长度;

}  GPDU;

外部无效              prw_rdrec_init(Pdu *pdu, Gpdu *gpdu);

外部布尔值 prw_rdrec（Gpdu gpdu）；

外部布尔值 prw_skiprec(Gpdu gpdu);


#定义 Garp_terminating_record_id 0x0000

# 万一 /* prw_h__ */
```

13.2.6 系统.h

```
/* 系统.h     * /
# 如果定义   系统状态变量
＃ 定义     系统状态变量

# 包括 <stddef.h>

/************************************************************ ****************************
 * SYS：系统提供的例程和原语
 ********************************************************************************************
 *
 * 此头文件表示系统提供的例程和原语分组
 * 进入     五       类别：
 *
 * 系统          ： 系统约定。
 *
 * 系统内存      ：一般内存分配。
 *
 * 系统协议数据单元      ：协议缓冲区分配、访问、传输和接收。
 *
 * SYSTIME：调度程序：立即、在固定（大约）时间内、在
 *                    随机时间段。
 *
 * 系统错误      ：检测到程序逻辑中存在严重错误的系统错误例程
 *                    在执行过程中，在检测点没有合理的行动方针。
 *
 * /

/************************************************************ ****************************
 * SYS：系统约定
 ********************************************************************************************
 * /

typedef enum {False = 0，True = 1} 布尔值；

枚举{零 = 0，一 = 1，二 = 2}；

类型定义  未签名       字符       八位组；

类型定义  未签名       短的       Int16；

类型定义  未签名       字符     * Mac地址；

/************************************************************ ****************************
 * SYS：系统提供的内存分配例程
 ********************************************************************************************
 * /

外部布尔 sysmalloc(int size，void **allocated)；

外部无效                sysfree（void *分配）；


/************************************************************ ****************************
 * SYSPDU：系统提供的 PDU 访问原语
 ********************************************************************************************
 * /

类型定义 void          普渡；
```

```
外部布尔 syspdu_alloc(Pdu **pdu);

外部      空白      syspdu_free（   普杜   *协议数据单元）；

外部      布尔值    rdcheck（       普杜   *协议数据单元，   整数 剩余八位字节数）；

外部      布尔值    rdoctet（       普杜   *协议数据单元，   八位字节 值);

外部      布尔值    rdint16(        普杜   *协议数据单元，   Int16    值);

外部      布尔值    rdskip(         普杜   *协议数据单元，   整数 要跳过的数字）；

外部      空白      syspdu_tx（     普杜   *协议数据单元，   整数 端口号）；

/********************************************************** ****************************
 * SYSTIME：系统提供的调度功能
 ********************************************************************************************
 * /

外部无效 systime_start_random_timer（int process_id，
                        void (*expiry_fn)(void *，int instance_id)，
                        整数      实例ID，
                        整数      暂停）；

外部无效systime_start_timer（int                    进程ID，
                                    void (*expiry_fn) (void *，int instance_id)，
                                    整数      实例ID，
                                    整数      暂停）；

外部无效 systime_schedule（int                  进程ID，
                                  void (*expiry_fn) (void *，int instance_id)，
                                  int 实例 ID）；

/********************************************************** ****************************
 * SYSERR：严重错误处理
 ********************************************************************************************
 * /

外部 syserr_panic();

# 结束语 /* sys_h__ */
```

13.3 GARP 应用独立代码

13.3.1 gid.c

```
/*gid.c*/

# 包括 “系统.h”
# 包括 “gid.h”
# 包括 “gidtt.h”
# 包括 “gip.h”
# 包括 “garp.h”

/********************************************************** ****************************
 * GID：GARP信息分发协议：创建、销毁
 ********************************************************************************************
 * /

静态布尔 gid_create_gid(Garp *application、int port_no、void **gid) { /*

    * 创建一个新的 GID 实例。
    * /

    Gid *我的端口；

    如果 (!sysmalloc(sizeof(Gid), &my_port))
        转到gid_creation_failure；

    my_port->应用程序                          = 应用;
    我的端口->端口号                           = 端口号；
    我的端口->下一个端口环 我的端口->下一个连  = 我的端口；
    接环                                       = 我的端口；

    my_port->已启用                            = 错误的;
    my_port->已连接                            = 错误的;
    my_port->is_point_to_point                 = 真的;

    my_port->cschedule_tx_now                  = 错误的;
    my_port->cstart_join_timer my_port-        = 错误的;
    >cstart_leave_timer my_port-               = 错误的;
    >tx_now_scheduled my_port-                 = 错误的;
    >join_timer_running my_port-               = 错误的;
    >leave_timer_running my_port-              = 错误的;
    >hold_tx                                   = 错误的;

    my_port->加入超时                          = Gid_默认_加入时间；
    my_port->leave_timeout_4                   = Gid_默认_离开时间/4；Gid_默认_
    my_port->hold_timeout                      = 保持时间；

    如果 (!sysmalloc(sizeof(Gid_machine)*(应用程序->max_gid_index + 2),
                        &my_port->machines）
        转到gid_mcreation_failure；

    my_port->leaveall_countdown                = gid_leaveall_count；
    my_port->leaveall_timeout_n                = Gid_default_leaveall_time/
                                               gid_leaveall_count；
    systime_start_timer（my_port->应用程序->进程ID，
                        gid_leaveall_timer_expired，
                        my_port->port_no，
                        我的端口->leaveall_timeout_n）；


    my_port->tx_pending                        = 错误的;
    my_port->上次传输 my_port->上次            =  应用程序->last_gid_used；应用程序-
    传输                                       =  >last_gid_used；
```

```
        my_port->untransmit_machine                    =应用程序->last_gid_used + 1;

        *gid = 我的端口；              返回（真）；
gid_mcreation_failure（创建失败）：   sysfree（我的端口）；
gid_creation_failure（创建失败）：    返回（假）；
}


静态 void gid_destroy_gid(Gid *gid) { /*

        * 销毁GID实例，释放先前分配的空间。
        * 向已注册的应用程序发送休假指示
        * 属性。
        * /
        无符号 gid_index；

        对于（gid_index = 0; gid_index <= gid->application->last_gid_used;
                gid_index++）
        {
                如果 (gid_registered_here(gid，gid_index))
                        gid->应用程序->leave_indication_fn（gid->应用程序，
                                                                gid，gid_index）；
        }

        sysfree(gid->机器); sysfree(gid);

}

静态 Gid *gid_add_port(Gid *existing_ports，Gid *new_port) { /*

        * 将 new_port 添加到端口环。
        * /

        吉德 * 事先的;
        吉德 * 下一个;
        整数   新端口号；

        如果 (现有端口 != NULL) {

                新端口号 = 新端口->端口号；

                下一个=现有端口；
                为了（;;）
                {
                        先前 = 下一个；下一个 = 先前->next_in_port_ring；

                        如果 (prior->port_no <= new_port_no) {

                                如果 （     （下一个->port_no   <=  先前->port_no)
                                        ||    （下一个->port_no   >    新端口号)
                                    ） 休息;
                        }
                        否则/*如 果（prior_>port_no>new_port_no）*/{

                                如果 （     （下一个->port_no   <=  先前->port_no)
                                        &&  （下一个->port_no   >    新端口号)
                                    ） 休息;
                }     }

                如果 (prior->port_no == new_port_no) syserr_panic();

                先前->下一个端口环 新端口->下一个端   =  新港；
                口环                                  =  下一个;
        }
```

```
        new_port->is_enabled = True; 返回
         （new_port）；
}


静态 Gid *gid_remove_port(Gid *my_port) {

        吉德 * 事先的;
        吉德 * 下一个;

        先前=我的端口；
        当（（next = previous->next_in_port_ring）！ = my_port）
              前一个=下一个；

        先前->下一个端口环 = 我的端口->下一个端口环；

        如果 (prior == my_port) 返回 (NULL); 否则
                                          返回（之前）；
}


布尔值      gid_create_port（Garp *应用程序，int port_no）
{
         吉德 *我的_端口；

         如果 (!gid_find_port(application->gid、port_no、&my_port)) {

              如果 (gid_create_gid (应用程序，port_no，&my_port)) {

                   应用程序->gid = gid_add_port(应用程序->gid，my_port); 应用程序-
                   >added_port_fn(应用程序，port_no); 返回(True);

         } }

        返回（假）；
  }


void gid_destroy_port(Garp *application, int port_no) {

         Gid *我的端口；

         如果 (gid_find_port(application->gid、port_no 和 my_port)) {

              gip_disconnect_port（应用程序，port_no）；

              应用程序->gid = gid_remove_port(my_port);

              gid_destroy_gid（我的端口）；

              应用程序->removed_port_fn（应用程序，port_no）；
  } }


/****************************************************** ****************************
  * GID：GARP 信息分发协议：有用的功能
  ******************************************************************************
  * /

布尔 gid_find_port(Gid *first_port，int port_no，void **gid) {

        吉德     * 下一个端口                    =第一个端口；
        当（next_port->port_no != port_no）{

              如果 ((next_port = next_port->next_in_port_ring) == first_port)
```

```
                返回（假）；
        }

        * gid = next_port; 返回（True）；
}


Gid *gid_next_port(Gid *this_port) {

        返回（this_port->next_in_port_ring）；
}


/******************************************************** ****************************
 * GID：GARP 信息分发协议：MGT
 ******************************************************************************************
 * /

void gid_read_attribute_state(Gid *my_port，无符号索引，Gid_states *state) { /*

        * /
        gidtt_states(&my_port->machines[index]，状态);
}


void gid_manage_attribute(Gid *my_port，无符号索引，Gid_event 指令) { /*

        *
        * /

        Gid_machine     * 机器;
        事件                  事件;

        机器          =  &my_port->机器[索引];
        事件          =    gidtt_event（my_port，机器，              指示）；
        如果 (事件 == Gid_join) {

                my_port->应用程序->join_indication_fn（my_port->应用程序，
                                                                        我的端口，索引）；
                gip_propagate_join（my_port，索引）；
        }
        否则，如果（事件 == Gid_leave）{

                my_port->应用程序->leave_indication_fn（my_port->应用程序，
                                                                        我的端口，索引）；
                gip_propagate_leave（my_port，索引）；
}       }


Boolean gid_find_unused(Garp *application, unsigned from_index,
                                                        无符号*found_index)
{
        未签名          gid_索引;
        吉德           *检查端口;

        gid_index = from_index; check_port = application->gid; 对于（;;）

        {
                如果 (gidtt_machine_active(&check_port->machines[gid_index])) {

                        如果（gid_index++>应用程序->last_gid_used）
                                返回（假）；
                        check_port = 应用程序->gid;
                }
                否则，如果（（check_port = check_port->next_in_port_ring）
```

```
                                        == 应用程序->gid)
            {
                  * found_index = gid_index; 返回
                    （True）；
}     }     }


/******************************************************** ****************************
  * GID：GARP 信息分发协议：事件处理
  ********************************************************************************************
  * /

静态 void gid_leaveall(Gid *my_port) { /*

      * 目前仅适用于共享媒体
      * /
      未签名          我;
      卡普          * 应用;

      应用程序=my_port->应用程序；
      对于（i = 0；i<=应用程序->last_gid_used；i++）
            (void) gidtt_event(my_port，&my_port->machines[i]，Gid_rcv_leaveempty);
}


void gid_rcv_leaveall(Gid *my_port) {

      my_port->leaveall_countdown = Gid_leaveall_count;
      gid_leaveall(my_port);
}


void gid_rcv_msg(Gid *my_port，无符号索引，Gid_event msg) {

      Gid_machine     * 机器;
      事件                事件;

      机器          =  &my_port->机器[索引];
      事件          =    gidtt_event（my_port，机器，              信息);
      如果 (事件 == Gid_join) {

            my_port->应用程序->join_indication_fn（my_port->应用程序，
                                                      我的端口，索引）；
            gip_propagate_join（my_port，索引）；
      }
      否则，如果（事件 == Gid_leave）{

            my_port->应用程序->leave_indication_fn（my_port->应用程序，
                                                      我的端口，索引）；
            gip_propagate_leave（my_port，索引）；
}     }


void gid_join_request(Gid *my_port，无符号gid_index) {

      （无效）（gidtt_event（my_port，&my_port->machines [gid_index]，            gid_join）；
}


void gid_leave_request(Gid *my_port，无符号 gid_index) {

      （无效）（gidtt_event（my_port，&my_port->machines [gid_index]，            离开）；
}
```

```
布尔 gid_registrar_in (Gid_machine *machine) {

      返回（gidtt_in（机器））；
}

/********************************************************** ****************************
 * GID：GARP 信息分发协议：接收处理
 ********************************************************************************************
 * /

void gid_rcv_pdu(Garp *application, int port_no, void *pdu) { /*

      * 如果找到此应用程序和端口号的 GID 实例，并且
      * 启用后，将 PDU 传递给应用程序，应用程序将对其进行解析（使用
      * 应用程序自己的 PDU 格式约定）并调用 gid_rcv_msg()
      * 从 PDU 中读取每个概念 GID 消息组件。一旦
      * 应用程序完成 PDU 后，调用 gip_do_actions() 来启动
      * 定时器记录在此端口的 GID 暂存器中，以及任何要
      * 它可能已经传播了连接或离开。
      *
      * 最后释放接收到的pdu。
      * /
      吉德         *我的_端口；

      如果 (gid_find_port(application->gid、port_no 和 my_port)) {

            如果（my_port->is_enabled）{

                  应用程序->receive_fn（应用程序，my_port，pdu）；

                  gip_do_actions（我的端口）；
      }     }
/* gid_rlse_rcv_pdu：插入所需的任何系统特定操作。 */}


/********************************************************** ****************************
 * GID：GARP 信息分发协议：传输处理
 ********************************************************************************************
 * /

Gid_event gid_next_tx(Gid *my_port，无符号*index){/*

      * 检查是否应该发送 leaveall；如果是，则返回 Gid_tx_leaveall；
      * 否则，扫描 GID 机器以查找需要传输的消息。
      *
      * 将从以下位置开始检查机器是否存在潜在传播
      * 机器遵循 last_transmitted 并一直到 last_to_transmit。
      * 如果所有机器都已传输，则 last_transmitted 等于 last_to_transmit
      * 并且 tx_pending 为 False （在这种情况下，tx_pending 区分了
      * 情况“所有机器都还未进行传输检查”来自“所有机器都已
      * 已检查。”）
      *
      * 如果 tx_pending 为 True 且所有机器尚未检查，则传输
      * 将从 GID 索引为 0 的机器启动，而不是立即启动
      * 下列的            最后传输。
      * /
      未签名              检查索引；
      未签名              停止后；
      事件                味精；

      如果（my_port->hold_tx）返回（Gid_null）；

      如果 (my_port->leaveall_countdown == 0) {

            my_port->leaveall_countdown = Gid_leaveall_count;
```

```
            systime_start_timer（my_port->应用程序->进程ID，
                                    gid_leaveall_timer_expired，
                                    my_port->port_no，
                                    我的端口->leaveall_timeout_n）；
            返回（Gid_tx_leaveall）；
      }

      如果（！my_port->tx_pending）返回（Gid_null）；

      检查索引                = my_port->last_transmitted + 1; = my_port-
      停止后                  >last_to_transmit;
      如果（停止后<检查索引）
            stop_after = my_port->应用程序->last_gid_used；

      对于（;; check_index++）
      {
            如果（检查索引 > 停止后）{

                  如果 (stop_after == my_port->last_to_transmit) {

                        my_port->tx_pending = False; 返回
                        （Gid_null）；
                  }
                  否则，如果（stop_after == my_port->application->last_gid_used）{

                        检查索引          = 0；
                        停止后            =我的端口->最后传输；
            }     }

            如果 ((msg = gidtt_tx(my_port，&my_port->machines[check_index]))
                        != Gid_null)
            {
                  * 索引 = my_port->last_transmitted = check_index;

                  my_port->机器[my_port->untransmit_machine].申请人 = my_port->机器
                  [check_index].申请人;

                  my_port->tx_pending = (check_index == stop_after);

                  返回（消息）；
}     }     } /* 结束（;;） */


void gid_untx(Gid *my_port) {

      my_port->机器[my_port->last_transmitted].申请人 = my_port->机器[my_port-
      >untransmit_machine].申请人；

      如果 (my_port->last_transmitted == 0)
            my_port->last_transmitted = my_port->应用程序->last_gid_used;
      别的
            我的端口->最后传输--；

      my_port->tx_pending = True;
}

/********************************************************* ****************************
 * GID：GARP 信息分发协议：计时器处理
 ********************************************************************************************
 * /

void gid_do_actions(Gid *my_port) {/*

      * 执行本次 GID 调用中积累的“暂存器”操作，
      * 和未完成的‘立即’传输和加入计时器启动
```

```
        * 已被保持定时器的操作延迟。注意
        * 保持定时器在此工作。可以指定它只是为了
        * 对传输施加最小间距 - 并与传输并行运行
        * 加入计时器 - 保持计时器和实际时间越长，效果越好
        * 连接计时器的值将决定实际的传输时间。
        * 未采用这种方法，因为它可能导致聚集
        * 保持时间内的传输。
        *
        * 如果仍有传输，则该程序将重新启动加入计时器
        * 待处理（如果 leaveall_countdown 为零，则将发送 Leaveall；如果
        * tx_pending 为真，个别机器可能有消息要发送。）
        * /

        int my_port_no = my_port->port_no;

        如果（my_port->cstart_join_timer）{

            my_port->last_to_transmit       =  my_port->last_transmitted; 真；
            my_port->tx_pending             =
            my_port->cstart_join_timer      =  错误的;
        }

        如果（！ my_port->hold_tx）
        {
            如果（my_port->cschedule_tx_now）{

                如果（！ my_port->tx_now_scheduled）
                    systime_schedule（my_port->应用程序->进程ID，
                                     gid_join_timer_expired，我的端
                                     口->端口号）；

                my_port->cschedule_tx_now       =假；
            }
            否则，如果（      （my_port->tx_pending || （my_port->leaveall_countdown == 0））（！ my_port-
                        &&    >join_timer_running））
            {
                systime_start_random_timer（my_port->应用程序->进程ID，
                                            gid_join_timer_expired，我的端
                                            口->端口号，
                                            我的端口->加入超时）；

                my_port->join_timer_running = True;
        }     }

        如果 (my_port->cstart_leave_timer && (!my_port->leave_timer_running)) {

            systime_start_timer（my_port->应用程序->进程ID，
                                 gid_leave_timer_expired，我的
                                 端口->端口号，
                                 my_port->leave_timeout_4);

        }
        my_port->cstart_leave_timer = False;
    }


    void gid_leave_timer_expired(Garp *application，int port_no) {

        吉德        *我的_端口；
        未签名        gid_索引；

        如果 (gid_find_port(application->gid、port_no 和 my_port)) {

            对于（gid_index = 0；gid_index<my_port->application->last_gid_used；
                gid_index++）
```

```
            {
                如果（    gidtt_leave_timer_expiry(my_port,&my_port->machines[gid_index]) Gid_leave)
                    ==
                {
                    my_port->应用程序->leave_indication_fn（my_port->应用程序，
                                                                    我的端口，gi d索引）；
                    gip_propagate_leave（my_port，gid_index）；
}     }     }     }


void gid_leaveall_timer_expired(Garp *application，int port_no) {

      Gid *我的端口；

      如果 (gid_find_port(application->gid、port_no 和 my_port)) {

            如果 (my_port->leaveall_countdown > 1)
                  my_port->leaveall_countdown--；
            别的
            {
                  gid_leaveall（我的端口）；

                  my_port->leaveall_countdown         = 0；
                  my_port->cstart_join_timer          = 真的;

                  如果（（! my_port->join_timer_running）&&（! my_port->hold_tx））

                  systime_start_random_timer（my_port->应用程序->进程ID，
                                              gid_join_timer_expired，我的端
                                              口->端口号，
                                              我的端口->加入超时）；

                  my_port->cstart_join_timer = my_port- 错误的;
                  >join_timer_running =                 真的;
}     }     }


void gid_join_timer_expired(Garp *application，int port_no) {

      Gid *我的端口；

      如果 (gid_find_port(application->gid、port_no 和 my_port)) {

            如果（my_port->is_enabled）{

                  应用程序->transmit_fn（应用程序，my_port）；

                  systime_start_timer（my_port->应用程序->进程ID，
                                       gid_hold_timer_expired，我的
                                       端口->端口号，
                                       我的端口->保持超时）；
}     }     }


void gid_hold_timer_expired(Garp *application，int port_no) {

      Gid *我的端口；

      如果 (gid_find_port(application->gid、port_no 和 my_port)) {

            my_port->hold_tx = False;

            gid_do_actions（我的端口）；
}     }
```

13.3.2 gidtt.c

```
/*gidtt.c*/

# 包括       “gidtt.h”
# 包括       “gid.h”


/********************************************************** ****************************
 * GIDTT：GID 协议转换表：实施概述
 ********************************************************************************************
 * /
/* 此 GID 实现直接使用转换表。这使得
 * 实现起来显得很庞大，但对网络代码大小的影响可能很小
 * 少于基于算法的实现所需的代码页，
 * 取决于处理器。基于处理的实现计划如下
 * 两种选择都可能很有趣。
 *
 * 申请人和注册处使用单独的转换表。两者都使用
 * 一个通用的转换表来处理大多数事件，并分离较小的事件
 * 表格来确定传输机会发生时的行为（两者
 * 申请人和注册员），以及休假计时器到期时（仅限注册员）。
 *
 * 存储的状态直接支持管理控制 - 这导致了
 * 总共有 14 个申请国和 18 个登记国（登记国
 * 还包含休假计时器支持):
 *
 *       申请人可能处于下列管理状态之一：
 *
 *         1.正常
 *
 *         2. 没有礼仪
 *
 *              协议机保持安静，不会发送任何消息，甚至不会提示同一 LAN 上的另一个 GID 参与者做出响应。
 *              在此状态下，申请人会在媒体上发送消息，因此可以在正常和无协议之间切换，同时最大程度地
 *              减少干扰。
 *
 *
 *
 *       注册处可能处于下列管理状态之一：
 *
 *         1.正常注册。
 *
 *         2. 注册已修复。
 *
 *              注册器始终将 LAN 上发生的任何事情报告给应用程序和 GIP。
 *
 *
 *         3. 禁止注册。
 *
 *              注册官始终向申请者和 GIP 报告“空”。
 *
 * 一组小表格用于报告管理状况的各个方面
 * 申请人和注册员。
 *
 * 主申请人过渡表 (applicant_tt) 由当前
 * 申请人状态和 GID 事件，并返回
 *
 *         1. 新申请国。
 *
 *         2. 需要时，启动加入计时器指令。
 *
 *
 *
```

```
 * 主注册转换表 (registrar_tt) 由当前
 * 注册器状态和 GID 事件，并返回
 *
 *          1. 新的登记国。
 *
 *          2. 必要时，加入指示或离开指示。
 *
 *          3. 需要时，启动离开计时器指令。
 *
 * 这两个表的唯一用户界面是通过公共
 * 函数 gidtt_event()，接受并返回 Gid_events（用于报告
 * 加入或离开指示），并将计时器启动请求写入
 * 直接使用 GID 暂存器。
 *
 * 申请人传输转换表（applicant_txtt）返回新的
 * 申请者状态，需要传输的消息，以及加入定时器是否
 * 应重新启动以传输进一步的消息。修饰符
 * 确定是否存在连接（由申请人表选择进行传输）
 * 应该作为 JoinIn 或 JoinEmpty 传输，取自
 * 注册商状态报告表。注册商状态永远不会被修改
 * 传播。
 * /


/*********************************************************** ******************************
 * GIDTT：GID 协议转换表：表条目定义
 *********************************************************************************************
 * /
枚举申请人状态
{
    弗吉尼亚州，  /* 非常焦虑，活跃 /*              * /
    啊，          焦虑的，          积极的      * /
    问：    /*    安静的，          积极的      * /
    啦，    /*    离开，            积极的      * /
    副总裁，/* 非常焦虑，被动 /*                * /
    美联社，      焦虑的，          被动的      * /
    Qp，    /*    安静的，          被动的      * /
    沃，    /* 非常焦虑的观察者 /*              * /
    敖，          焦虑的观察者                  * /
    问：    /*    安静的观察者                  * /
    瞧，    /*    离开观察员                    * /

    Von, /* 非常焦虑的观察者，非参与者 Aon, /* 焦虑的观察者，        * /
                                     非参与者                    * /
    Qon/* Quiet_observer，           非参与者                    * /
};

枚举 Registrar_states
{ /* 在，离开，空，但离开状态实现倒计时
    * 离开计时器。
    * /

    客栈，
    等级,L3,L2,L1,
    公吨，

    印度卢比，             /* 在，注册已修复              * /
    Lvr,L3r,L2r,L1r,
    地铁，

    嗯，                   /* 禁止注册 */
    Lvf,L3f,L2f,L1f,
    移动流量表
};
```



```
enum {Number_of_applicant_states = Qon + 1}; /* 用于数组大小 */ enum
{Number_of_registrar_states = Mtf + 1}; /* 用于数组大小 */

枚举计时器
{
    Nt = 0，      /* 没有计时器动作 /*            * /
    犁狮= 1、          启动加入定时器              * /
    左  =  1     /*    启动离开定时器              * /
};

枚举申请人信息
{
    Nm = 0, /* 没有消息需要传输 */ Jm,
                  /* 传输 Join 消息 /* 传输               * /
    嗯，          Leave 消息 /* 传输 Empty 消            * /
    埃姆          息                                     * /
};

枚举 Registrar_indications {

    镍  =  0，
    李  =  1、
    吉  =  2
};

/********************************************************** ******************************
 * GIDTT：GID 协议转换表：转换表结构
 ***********************************************************************************
 * /

类型定义结构                                /*    申请人_tt_entry           * /
{
    未签名的 new_app_state              :  5;    /*   {申请人州}              * /

    无符号cstart_join_timer: 1;

} 申请人_tt_entry;

类型定义结构                                /* 注册器_tt_entry */
{
    无符号 new_reg_state                :  5;

    未签名的指示                        :  2;

    无符号cstart_leave_timer : 1;

}   注册商_tt_entry;

类型定义结构                                /*申请人_txtt_entry*/
{
    未签名的 new_app_state              :  5;

    未签名的要发送的消息                : 2; /* 申请人信息 */

    无符号 cstart_join_timer            :  1;

}   申请人_txtt_entry;

类型定义结构                                /* 注册器离开定时器条目                * /
{
    无符号 new_reg_state                : 5; /* 注册状态                          * /

    无符号离开指示                      :  1;
```

```
        无符号cstart_leave_timer : 1;

} 注册商离开定时器入口；


/************************************************************ ****************************
  * GIDTT：GID 协议：主申请人过渡表
  ********************************************************************************************
  * /

静态申请人_tt_entry
            申请人_tt[gid_rcv_events 数量                          + 请求事件数量 请求事件数量]          +
                                Number_of_gid_amgt_events [申请人
                               状态数]=
{/*
        * 一般申请人过渡表。请参阅上文说明。
        * /
        {       /*Gid_null*/
                /*Va */{Va, Nt},/*Aa */{Aa, Nt},/*Qa */{Qa, Nt},/*La */{La, Nt}, /*Vp */{ Vp, Nt},/*Ap */{Ap,
                Nt},/*Vp */{Vp, Nt},
                /*Vo */{Vo, Nt},/*Ao */{Ao, Nt},/*Qo */{Qo, Nt},/*Lo */{Lo, Nt},

                /*Von*/{Von,Nt},/*Aon*/{Aon,Nt},/*Qon*/{Qon,Nt}
        },
        {       /*Gid_rcv_leaveempty*/
                /*Va */{Vp, Nt},/*Aa */{Vp, Nt},/*Qa */{Vp, Jt},/*La */{Vo, Nt}, /*Vp */{ Vp, Nt},/*Ap */{Vp,
                Nt},/*Qp */{Vp, Jt},
                /*Vo */{Lo, Nt},/*Ao */{Lo, Nt},/*Qo */{Lo, Jt},/*Lo */{Vo, Nt},

                /*Von*/{Von,Nt},/*Aon*/{Von,Nt},/*Qon*/{Von,Nt}
        },
        {       /*Gid_rcv_leavein*/
                /*Va */{Va, Nt},/*Aa */{Va, Nt},/*Qa */{Vp, Jt},/*La */{La, Nt}, /*Vp */{ Vp, Nt},/*Ap */{Vp,
                Nt},/*Qp */{Vp, Jt},
                /*Vo */{Lo, Nt},/*Ao */{Lo, Nt},/*Qo */{Lo, Jt},/*Lo */{Vo, Nt},

                /*Von*/{Von,Nt},/*Aon*/{Von,Nt},/*Qon*/{Von,Nt}
        },
        {       /*Gid_rcv_empty*/
                /*Va */{Va, Nt},/*Aa */{Va, Nt},/*Qa */{Va, Jt},/*La */{La, Nt}, /*Vp */{ Vp, Nt},/*Ap */{Vp,
                Nt},/*Qp */{Vp, Jt},
                /*Vo */{Vo, Nt},/*Ao */{Vo, Nt},/*Qo */{Vo, Nt},/*Lo */{Vo, Nt},

                /*Von*/{Von,Nt},/*Aon*/{Von,Nt},/*Qon*/{Von,Nt}
        },
        {       /*Gid_rcv_joinempty*/
                /*Va */{Va, Nt},/*Aa */{Va, Nt},/*Qa */{Va, Jt},/*La */{Vo, Nt}, /*Vp */{ Vp, Nt},/*Ap */{Vp,
                Nt},/*Qp */{Vp, Jt},
                /*Vo */{Vo, Nt},/*Ao */{Vo, Nt},/*Qo */{Vo, Jt},/*Lo */{Vo, Nt},

                /*Von*/{Von,Nt},/*Aon*/{Von,Nt},/*Qon*/{Von,Jt}
        },
        {       /*Gid_rcv_joinin*/
                /*Va */{Aa, Nt},/*Aa */{Qa, Nt},/*Qa */{Qa, Nt},/*La */{La, Nt}, /*Vp */{ Ap, Nt},/*Ap */{Qp,
                Nt},/*Qp */{Qp, Nt},
                /*Vo */{Ao, Nt},/*Ao */{Qo, Nt},/*Qo */{Qo, Nt},/*Lo */{Ao, Nt},

                /*Von*/{Aon,Nt},/*Aon*/{Qon,Nt},/*Qon*/{Qon,Nt}
        },
        {       /* Gid_join，加入请求。处理重复加入，即加入
                 * 已经存在的状态。不提供加入的反馈
                 * 管理控制禁止的；期望是
                 * 新管理层不会直接使用该表
                 * 请求。
                 * /
```

```
            /*Va */{Va, Nt},/*Aa */{Aa, Nt},/*Qa */{Qa, Nt},/*La */{Va, Nt}, /*Vp */{ Vp, Nt},/*Ap */{Ap,
            Nt},/*Qp */{Qp, Nt},
            /*Vo */{Vp, Jt},/*Ao */{Ap, Jt},/*Qo */{Qp, Nt},/*Lo */{Vp, Nt},

            /*Von*/{Von,Nt},/*Aon*/{Aon,Nt},/*Qon*/{Qon,Nt}
      },
      {     /*Gid_leave，离开请求。请参阅上面加入请求的注释。*/ /*Va */{La, Nt},/*Aa */{La, Nt},/*Qa */{La,
            Jt},/*La */{La, Nt},/*Vp */{Vo, Nt},/*Ap */{Ao, Nt},/*Qp */{Qo, Nt},

            /*Vo */{Vo, Nt},/*Ao */{Ao, Nt},/*Qo */{Qo, Nt},/*Lo */{Lo, Nt},

            /*Von*/{Von,Nt},/*Aon*/{Aon,Nt},/*Qon*/{Qon,Nt}
      },
      {     /*Gid_正常操作*/
            /*Va */{Vp, Nt},/*Aa */{Vp, Nt},/*Qa */{Vp, Jt},/*La */{La, Nt}, /*Vp */{ Vp, Nt},/*Ap */{Vp,
            Nt},/*Qp */{Vp, Jt},
            /*Vo */{Va, Nt},/*Ao */{Va, Nt},/*Qo */{Va, Jt},/*Lo */{Lo, Nt},

            /*Von*/{Va, Nt},/*Aon*/{Va, Nt},/*Qon*/{Va, Jt}
      },
      {     /* Gid_no_协议 */
            /*Va */{Von,Nt},/*Aa */{Aon,Nt},/*Qa */{Qon,Nt},/*La */{Von,Nt}, /*Vp */{ Von,Nt},/*Ap */
            {Aon,Nt},/*Qp */{Qon,Nt},
            /*Vo */{Von,Nt},/*Ao */{Aon,Nt},/*Qo */{Qon,Nt},/*Lo */{Von,Nt},

            /*Von*/{Von,Nt},/*Aon*/{Aon,Nt},/*Qon*/{Qon,Nt}
      },
      {     /*Gid_normal_registration，与申请人的 Gid_null 相同 */ /*Va */{Va, Nt},/*Aa */{Aa, Nt},/
            *Qa */{Qa, Nt},/*La */{La, Nt},/*Vp */{Vp, Nt},/*Ap */{Ap, Nt},/*Vp */{Vp, Nt}，

            /*Vo */{Vo, Nt},/*Ao */{Ao, Nt},/*Qo */{Qo, Nt},/*Lo */{Lo, Nt},

            /*Von*/{Von,Nt},/*Aon*/{Aon,Nt},/*Qon*/{Qon,Nt}
      },
      {     /*Gid_fix_registration，与申请人的Gid_null相同*//*Va*/{Va，Nt}，/*Aa*/{Aa，Nt}，/
            *Qa*/{Qa，Nt}，/*La*/{La，Nt}，/*Vp*/{Vp，Nt}，/*Ap*/{Ap，Nt}，/*Vp*/{Vp，Nt}，

            /*Vo */{Vo, Nt},/*Ao */{Ao, Nt},/*Qo */{Qo, Nt},/*Lo */{Lo, Nt},

            /*Von*/{Von,Nt},/*Aon*/{Aon,Nt},/*Qon*/{Qon,Nt}
      },
      {     /*Gid_forbid_registration，与申请人的Gid_null相同*//*Va*/{Va，Nt}，/*Aa*/{Aa，Nt}，/
            *Qa*/{Qa，Nt}，/*La*/{La，Nt}，/*Vp*/{Vp，Nt}，/*Ap*/{Ap，Nt}，/*Vp*/{Vp，Nt}，

            /*Vo */{Vo, Nt},/*Ao */{Ao, Nt},/*Qo */{Qo, Nt},/*Lo */{Lo, Nt},

            /*Von*/{Von,Nt},/*Aon*/{Aon,Nt},/*Qon*/{Qon,Nt}
      }
};


/************************************************** ****************************
 * GIDTT：GID 协议：主注册器转换表
 ********************************************************************************
 * /

静态 Registrar_tt_entry
            registrar_tt[gid_rcv_events 的数量                    +GID_REQ_事件数GID_RMGT_事件数]+
                        Number_of_gid_amgt_events [注册器~
                        状态数量]
{

      {     /* Gid_null              * /
            /*在    * /{Inn,Ni,Nt}，
            /*等级 * /{Lv，Ni，Nt}，
```

```
        /*L3 */{L3, Ni,Nt},/*L2 */{L2, Ni,Nt},/*L1 */{L1, Ni,Nt}, /*Mt */{Mt, Ni,Nt },


        /*Inr*/{Inr,Ni,Nt},
        /*Lvr*/{Lvr,Ni,Nt},
        /*L3r*/{L3r,Ni,Nt},/*L2r*/{L2r,Ni,Nt},/*L1r*/{L1r,Ni,Nt}, /*Mtr*/{Mtr,Ni,Nt },


        /*Inf*/{Inf,Ni,Nt},
        /*Lvf*/{Lvf,Ni,Nt},
        /*L3f*/{L3f,Ni,Nt},/*L2f*/{L2f,Ni,Nt},/*L1f*/{L1f,Ni,Nt}, /*Mtf*/{Mtf,Ni,Nt }

    },
    {   /* Gid_rcv_离开            * /
        /*旅馆*/{Lv，    Ni,Lt},
        /*等级*/{等级，   Ni,Nt},
        /*L3 */{L3, Ni,Nt},/*L2 */{L2, Ni,Nt},/*L1 */{L1, Ni,Nt}, /*Mt */{Mt, Ni,Nt },


        /*Inr*/{Lvr,Ni,Lt},
        /*Lvr*/{Lvr,Ni,Nt},
        /*L3r*/{L3r,Ni,Nt},/*L2r*/{L2r,Ni,Nt},/*L1r*/{L1r,Ni,Nt}, /*Mtr*/{Mtr,Ni,Nt },


        /*Inf*/{Lvf,Ni,Lt},
        /*Lvf*/{Lvf,Ni,Nt},
        /*L3f*/{L3f,Ni,Nt},/*L2f*/{L2f,Ni,Nt},/*L1f*/{L1f,Ni,Nt}, /*Mtf*/{Mtf,Ni,Nt }

    },
    {   /*Gid_rcv_empty*/
        /*Inn*/{Inn,Ni,Nt},
        /*Lv*/{Lv，Ni，Nt},
        /*L3 */{L3, Ni,Nt},/*L2 */{L2, Ni,Nt},/*L1 */{L1, Ni,Nt}, /*Mt */{Mt, Ni,Nt },


        /*Inr*/{Inr,Ni,Nt},
        /*Lvr*/{Lvr,Ni,Nt},
        /*L3r*/{L3r,Ni,Nt},/*L2r*/{L2r,Ni,Nt},/*L1r*/{L1r,Ni,Nt}, /*Mtr*/{Mtr,Ni,Nt },


        /*Inf*/{Inf,Ni,Nt},
        /*Lvf*/{Lvf,Ni,Nt},
        /*L3f*/{L3f,Ni,Nt},/*L2f*/{L2f,Ni,Nt},/*L1f*/{L1f,Ni,Nt}, /*Mtf*/{Mtf,Ni,Nt }


    },
    {   /* Gid_rcv_joinempty          * /
        /*Inn*/{Inn,Ni,Nt},
        /*Lv*/{Inn,Ni,Nt},
        /*L3 */{Inn,Ni,Nt},/*L2 */{Inn,Ni,Nt},/*L1 */{Inn,Ni,Nt}, /*Mt */{Inn,Ji,Nt },


        /*Inr*/{Inr,Ni,Nt},
        /*Lvr*/{Inr,Ni,Nt},
        /*L3r*/{Inr,Ni,Nt},/*L2r*/{Inr,Ni,Nt},/*L1r*/{Inr,Ni,Nt}, /*Mtr*/{Inr,Ni,Nt },


        /*Inf*/{Inf,Ni,Nt},
        /*Lvf*/{Inf,Ni,Nt},
        /*L3f*/{Inf,Ni,Nt},/*L2f*/{Inf,Ni,Nt},/*L1f*/{Inf,Ni,Nt}, /*Mtf*/{Inf,Ni,Nt }


    },
    {   /*Gid_rcv_joinin*/
        /*Inn*/{Inn,Ni,Nt},
```

```
        /*Lv*/{Inn,Ni,Nt},
        /*L3 */{Inn,Ni,Nt},/*L2 */{Inn,Ni,Nt},/*L1 */{Inn,Ni,Nt}, /*Mt */{Inn,Ji,Nt },


        /*Inr*/{Inr,Ni,Nt},
        /*Lvr*/{Inr,Ni,Nt},
        /*L3r*/{Inr,Ni,Nt},/*L2r*/{Inr,Ni,Nt},/*L1r*/{Inr,Ni,Nt}, /*Mtr*/{Inr,Ni,Nt },


        /*Inf*/{Inf,Ni,Nt},
        /*Lvf*/{Inf,Ni,Nt},
        /*L3f*/{Inf,Ni,Nt},/*L2f*/{Inf,Ni,Nt},/*L1f*/{Inf,Ni,Nt}, /*Mtf*/{Inf,Ni,Nt }

    },
    {   /*Gid_normal_operation，与注册器的Gid_null相同*//*在
                * /{Inn,Ni,Nt},
        /*等级 * /{Lv，Ni，Nt},
        /*L3 */{L3, Ni,Nt},/*L2 */{L2, Ni,Nt},/*L1 */{L1, Ni,Nt}, /*Mt */{Mt, Ni,Nt },


        /*Inr*/{Inr,Ni,Nt},
        /*Lvr*/{Lvr,Ni,Nt},
        /*L3r*/{L3r,Ni,Nt},/*L2r*/{L2r,Ni,Nt},/*L1r*/{L1r,Ni,Nt}, /*Mtr*/{Mtr,Ni,Nt },


        /*Inf*/{Inf,Ni,Nt},
        /*Lvf*/{Lvf,Ni,Nt},
        /*L3f*/{L3f,Ni,Nt},/*L2f*/{L2f,Ni,Nt},/*L1f*/{L1f,Ni,Nt}, /*Mtf*/{Mtf,Ni,Nt }

    },
    {   /*Gid_no_protocol，与注册器的Gid_null相同*//*在
                * /{Inn,Ni,Nt},
        /*等级 * /{Lv，Ni，Nt},
        /*L3 */{L3, Ni,Nt},/*L2 */{L2, Ni,Nt},/*L1 */{L1, Ni,Nt}, /*Mt */{Mt, Ni,Nt },


        /*Inr*/{Inr,Ni,Nt},
        /*Lvr*/{Lvr,Ni,Nt},
        /*L3r*/{L3r,Ni,Nt},/*L2r*/{L2r,Ni,Nt},/*L1r*/{L1r,Ni,Nt}, /*Mtr*/{Mtr,Ni,Nt },


        /*Inf*/{Inf,Ni,Nt},
        /*Lvf*/{Lvf,Ni,Nt},
        /*L3f*/{L3f,Ni,Nt},/*L2f*/{L2f,Ni,Nt},/*L1f*/{L1f,Ni,Nt}, /*Mtf*/{Mtf,Ni,Nt }

    },
    {   /* Gid_normal_registration */ /*Inn*/
        {Inn,Ni,Nt},
        /*Lv*/{Lv，Ni，Nt},
        /*L3 */{L3, Ni,Nt},/*L2 */{L2, Ni,Nt},/*L1 */{L1, Ni,Nt}, /*Mt */{Mt, Ni,Nt },


        /*Inr*/{Inn,Ni,Nt},
        /*Lvr*/{Lv，Ni，Nt},
        /*L3r*/{L3, Ni,Nt},/*L2r*/{L2, Ni,Nt},/*L1r*/{L1, Ni,Nt}, /*Mtr*/{Mt, Li,Nt },


        /*Inf*/{Inn,Ji,Nt},
        /*Lvf*/{Lv，Ji，Nt},
        /*L3f*/{L3, Ji,Nt},/*L2f*/{L2, Ji,Nt},/*L1f*/{L1, Ji,Nt}, /*Mtf*/{Mt, Ni,Nt }


    },
    {   /* Gid_fix_registration /*Inn*/     * /
        {Inr,Ni,Nt},
```

```
            /*Lv*/{Lvr，Ni，Nt}，
            /*L3 */{L3r,Ni,Nt},/*L2 */{L2r,Ni,Nt},/*L1 */{L1r,Ni,Nt}, /*Mt */{Mtr,Ji,Nt },


            /*Inr*/{Inr,Ni,Nt}，
            /*Lvr*/{Lvr,Ni,Nt}，
            /*L3r*/{L3r,Ni,Nt},/*L2r*/{L2r,Ni,Nt},/*L1r*/{L1r,Ni,Nt}, /*Mtr*/{Mtr,Ni,Nt },


            /*Inf*/{Inr,Ji,Nt}，
            /*Lvf*/{Lvr,Ji,Nt}，
            /*L3f*/{L3r,Ji,Nt},/*L2f*/{L2r,Ji,Nt},/*L1f*/{L1r,Ji,Nt}, /*Mtf*/{Mtr,Ji,Nt }


      },
      {     /*Gid_forbid_registration/*Inn*/      * /
            {Inf,Li,Nt}，
            /*Lv*/{Lvf,Li,Nt}，
            /*L3 */{L3f,Li,Nt},/*L2 */{L2f,Li,Nt},/*L1 */{L1f,Li,Nt}, /*Mt */{Mtf,Ni,Nt },


            /*Inr*/{Inr,Li,Nt}，
            /*Lvr*/{Lvr,Li,Nt}，
            /*L3r*/{L3r,Li,Nt},/*L2r*/{L2r,Li,Nt},/*L1r*/{L1r,Li,Nt}, /*Mtr*/{Mtr,Li,Nt },


            /*Inf*/{Inf,Ni,Nt}，
            /*Lvf*/{Lvf,Ni,Nt}，
            /*L3f*/{L3f,Ni,Nt},/*L2f*/{L2f,Ni,Nt},/*L1f*/{L1f,Ni,Nt}, /*Mtf*/{Mtf,Ni,Nt }

      }
};


/************************************************************ ******************************
 * GIDTT：GID 协议：申请人传输表
 ********************************************************************************************
 * /

静态申请人_txtt_entry
          申请人txtt[申请人所在州数量] =
{
 /*Va */{Aa, Jm,Jt},/*Aa */{Qa, Jm,Nt},/*Qa */{Qa, Nm,Nt},/*La */{Vo, Lm,Nt }, /*Vp */{Aa, Jm,Jt},/*Ap */{Qa,
 Jm,Nt},/*Qp */{Qp, Nm,Nt},
 /*Vo*/{Vo，Nm，Nt}，/*Ao*/{Ao，Nm，Nt}，/*Qo*/{Qo，Nm，Nt}，/*Lo*/{Vo，Nm，Nt}，

 /*Von*/{Von,Nm,Nt},/*Aon*/{Aon,Nm,Nt},/*Qon*/{Qon,Nm,Nt} };



/************************************************************ ******************************
 * GIDTT：GID 协议：注册离开计时器表
 ********************************************************************************************
 * /

静态Registrar_leave_timer_entry
          registrar_leave_timer_table[注册器状态数量]                                    =
{
 /*Inn*/{Inn,Ni,Nt}，
 /*Lv */{L3, Ni,Lt},/*L3 */{L2, Ni,Lt},/*L2 */{L1, Ni,Lt},/*L1 */{Mt, Li,Nt }, /*Mt */{Mt, Ni,Nt},


 /*Inr*/{Inr,Ni,Nt}，
 /*Lvr*/{L3r,Ni,Lt},/*L3r*/{L2r,Ni,Lt},/*L2r*/{L1r,Ni,Lt},/*L1r*/{Mtr,Ni,Nt }, /*Mtr*/{Mtr,Ni,Nt},
```

```
  /*Inf*/{Inf,Ni,Nt},
  /*Lvf*/{L3f,Ni,Lt},/*L3f*/{L2f,Ni,Lt},/*L2f*/{L1f,Ni,Lt},/*L1f*/{Mtf,Ni,Nt }, /*Mtf*/{Mtf,Ni,Nt}

};

/************************************************************ ******************************
  * GIDTT：GID 协议：国家报告表
  ********************************************************************************************
  * /

静态 Gid_applicant_state 申请人状态表[申请人状态数] = {

      /*Va */{非常焦虑},/*Aa */{焦虑},/*Qa */{安静},/*La */{离开},/*Vp */{非常焦虑},/*Ap */{焦虑},/*Qp */{安
      静},
      /*Vo */{非常焦虑},/*Ao */{焦虑},/*Qo */{安静},/*Lo */{离开},

      /*Von*/{非常焦虑},/*Aon*/{焦虑},/*Qon*/{安静}
};

静态 Gid_applicant_mgt 申请人_mgt_表[申请人状态数量] = {

      /*Va */{正常},/*Aa */{正常},/*Qa */{正常}, /*La */{正常},

      /*Vp */{正常},/*Ap */{正常},/*Qp */{正常}, /*Vo */{正常},/*Ao */{正
      常},/*Qo */ {正常}，/*Lo */{正常}，


      /*Von*/{No_protocol},/*Aon*/{No_protocol},/*Qon*/{No_protocol}
};

静态 Gid_registrar_state 注册器状态表[注册器状态数] = {

      /*旅馆*/{In}，
      /*Lv */{离开},/*L3 */{离开},/*L2 */{离开},/*L1 */{离开},/*Mt */{空},

      /*Inr*/{In}，
      /*Lvr*/{离开},/*L3r*/{离开},/*L2r*/{离开},/*L1r*/{离开},/*Mtr*/{空},

      /*Inf*/{In}，
      /*Lvf*/{离开},/*L3f*/{离开},/*L2f*/{离开},/*L1f*/{离开},/*Mtf*/{空}
};

静态 Gid_registrar_mgt 注册器_mgt_表[注册器状态数] = {

      /*旅馆*/{Normal_registration}，/*Lv
            */{正常注册}，/*L3                    * /{正常注册}，
      /*L2    */{正常注册}，/*L1                    * /{正常注册}，
      /*公吨 * /{正常注册}，

      /*Inr*/{Registration_fixed}, /*Lvr*/{Registration_fixed},/*L3r*/
      {Registration_fixed}, /*L2r*/{Registration_fixed},/*L1r*/{Registration_fixed}, /
      *Mtr*/{Registration_fixed},


      /*Inf*/{Registration_forbidden}、/*Lvf*/{Registration_forbidden}、/*L3f*/
      {Registration_forbidden}、/*L2f*/{Registration_forbidden}、/*L1f*/
      {Registration_forbidden}、/*Mtf*/{Registration_forbidden}

};

静态布尔值 registrar_in_table[Number_of_registrar_states] = {

      /*Inn*/{True},/*Lv*/{True},/*L3*/{True},/*L2*/{True},/*L1*/{True},/*Mt*/{False},
```

```
        /*Inr*/{真},/*Lvr*/{真},/*L3r*/{真},/*L2r*/{真},/*L1r*/{真},/*Mtr*/{真},

        /*Inf*/{False},/*Lvf*/{False},/*L3f*/{False},/*L2f*/{False},/*L1f*/{False}, /*Mtf*/ {错误的}
};

/*********************************************************** ****************************
  * GIDTT：GID 协议：接收事件、用户请求和管理处理
  ********************************************************************************************
  * /

Gid_event gidtt_event（Gid *my_port，Gid_machine *machine，Gid_event事件）{/*

        * 处理接收事件并加入或离开请求。
        * /

        申请人_tt_entry             * 过渡;
        注册商_tt_entry             * 转换;

        过渡              =  &applicant_tt[事件][机器->申请人]; ®istrar_tt[事件][机器->
        过渡              =  注册商];

        机器->申请人                =  转换到新状态：
        机器->注册器                =

        如果 ((event == Gid_join) && (atransition->cstart_join_timer)) my_port-
        >cschedule_tx_now = True;

        my_port->cstart_join_timer =          my_port->cstart_join_timer atransition-
                                      ||    >cstart_join_timer; my_port-
        my_port->cstart_leave_timer =         >cstart_leave_timer rtransition-
                                      ||    >cstart_leave_timer;

        切换（rtransition->指示）{

        案例 Ji：返回（Gid_join）；
              休息;
        案例 Li: 返回（Gid_leave）；
              休息;
        案例Ni：
        默认：        返回（Gid_null）；
}       }


布尔 gidtt_in(Gid_machine *machine) { /*

        *
        * /
        返回（registrar_in_table[machine->registrar]）；
}



/*********************************************************** ****************************
  * GIDTT：GID 协议转换表：传输消息
  ********************************************************************************************
  * /

Gid_event gidtt_tx(Gid                    *我的端口，
                   Gid_machine    * 机器）
{/*
        *
        * /
```

```
    未签名      味精；
    未签名      凛；

    如果 ((msg = 申请人_txtt[机器->申请人].msg_to_transmit) != Nm)
        rin = registrar_state_table[机器->注册器];

        my_port->cstart_join_timer = my_port->cstart_join_timer
            || 申请人_txtt[机器->申请人].cstart_join_timer；开关（消息）

    {
    案例 Jm：返回（rin！= Empty？Gid_tx_joinin：Gid_tx_joinempty）；
        休息;
    案例 Lm：返回（rin！= Empty？Gid_tx_leavein：Gid_tx_leaveempty）；
        休息;
    案例 Em：返回（Gid_tx_empty）；
        休息;
    案例数量：
    默认：      返回（Gid_null）；
    }
}


/*********************************************************** ****************************
 * GIDTT：GID 协议转换表：离开计时器处理
 ******************************************************************************************
 * /

Gid_event gidtt_leave_timer_expiry(Gid                          *我的端口，
                                   Gid_machine    * 机器）
{/*
    *
    * /
    Registrar_leave_timer_entry *rtransition；

    rtransition = ®istrar_leave_timer_table[machine->registrar];

    机器->注册器 = rtransition->new_reg_state;

    my_port->cstart_leave_timer =          my_port->cstart_leave_timer rtransition-
                                  ||    >cstart_leave_timer;

    返回（（rtransition->leave_indication == Li）？ Gid_leave：Gid_null）；
}


/*********************************************************** ****************************
 * GIDTT：GID 协议转换表：国家报告
 ******************************************************************************************
 * /

布尔 gidtt_machine_active (Gid_machine *machine) {

    如果（（机器->申请人 == Vo）&&（机器->注册商 == Mt））
        返回（假）；
    别的
        返回（真）；
}

void gidtt_states(Gid_machine *machine，Gid_states *state) { /*

    *
    * /

    状态->申请人状态 = 申请人状态表[机器->申请人];
```

```
        州->申请人管理                  =  申请人管理表[机器->申请人];

        状态->注册器状态                =  注册器状态表[机器->注册器];

        状态->registrar_mgt             = registrar_mgt_table[机器->注册器];
}
```

13.3.3 gip.c

```
/* gip.c */

# 包括       “gid.h”
# 包括       “gip.h”


/************************************************************ ******************************
 * GIP：GARP信息传播：创造，毁灭
 ********************************************************************************************
 * /

布尔 gip_create_gip (无符号 max_attributes，无符号 **gip) { /*

        * GIP 维护一组最多包含最大属性的传播计数。
        * 目前它没有维护任何附加信息，因此 GIP 实例
        * 直接由指向传播计数数组的指针表示。
        * /
        无符号*my_gip；

        如果 （！ sysmalloc（sizeof（unsigned）*max_attributes，＆my_gip））
              转到gip_creation_failure；

        *gip = 我的_gip；          返回（真）；
gip_creation_failure（创建失败）：   返回（假）；
}

void gip_destroy_gip(void *gip){/*

        *
        * /

        系统释放 (gip);
}

/************************************************************ ******************************
 * GIP：GARP 信息传播：连接、断开端口
 ********************************************************************************************
 * /

静态 void gip_connect_into_ring(Gid *my_port) {

        Gid *第一个连接，*最后一个连接；

        my_port->已连接                 =  真的;
        my_port->下一个连接环           =  我的端口；

        第一个连接=我的端口；

        做          第一个连接 = 第一个连接->下一个端口环； (!第一个连接->已连接);
        尽管

        my_port->next_in_connected_ring = first_connected;

        最后连接=第一次连接；
```

```
        当（last_connected->next_in_connected_ring！=first_connected）
                最后一个连接 = 最后一个连接->下一个连接环；

        最后一个连接的->下一个连接的环 = 我的端口；
}


静态 void gip_disconnect_from_ring(Gid *my_port) {

        Gid *第一个连接，*最后一个连接；

        首次连接                        =我的端口->下一个连接环；
        my_port->next_in_connected_ring          =  我的端口；
        my_port->已连接                          =  错误的；

        最后连接=第一次连接；

        当（last_connected->next_in_connected_ring！=my_port）
                最后一个连接 = 最后一个连接->下一个连接环；

        最后一个连接->下一个连接环 = 第一个连接；
}


void gip_connect_port(Garp *application, int port_no) { /*

        * 如果找到该应用程序和端口号的 GID 实例，则
        * 已启用，并且尚未连接，则将该端口连接到
        * GIP 传播环。
        *
        * 传播所有已注册的属性（即注册器
        * 似乎不为空）在任何其他连接的端口上，并且
        * 结果非零传播计数，到这个端口，生成一个连接
        * 要求。
        *
        * 传播已在此端口上注册的每个属性，并且不会
        * 任何其他（在连接此节点之前传播计数为零）
        * 端口）到所有连接的端口，更新传播计数。
        *
        * 执行任何所需的计时器。将端口标记为已连接。
        *
        * /
        吉德          *我的_端口；
        未签名         gid_索引；

        如果 (gid_find_port(application->gid、port_no 和 my_port)) {

                如果（（！my_port->is_enabled）||（my_port->is_connected））返回；

                gip_connect_into_ring（我的端口）；

                对于（gid_index = 0; gid_index <= 应用程序->last_gid_used;
                        gid_index++）
                {
                        如果（gip_propagates_to（my_port，gid_index））
                                gid_join_request（我的端口，gi d_index）；

                        如果（gid_registered_here（my_port，gid_index））
                                gip_propagate_join（my_port，gid_index）；
                }

                gip_do_actions（my_port）；my_port-
                >is_connected = True；
}       }
```

```
void gip_disconnect_port(Garp *application, int port_no) { /*
      * 撤销 gip_connect_port() 执行的操作。
      * /
      吉德          *我的_端口；
      未签名         gid_索引；

      如果 (gid_find_port(application->gid、port_no 和 my_port)) {

            如果（（！ my_port->is_enabled）||（！ my_port->is_connected））返回；

            对于（gid_index = 0; gid_index <= 应用程序->last_gid_used;
                  gid_index++）
            {
                  如果（gip_propagates_to（my_port，       gid_index）
                        gid_leave_request（我的端口，         gid_索引）；

                  如果（gid_registered_here（my_port，         gid_index）
                        gip_propagate_leave（我的端口，         gid_索引）；
            }

            gip_do_actions（my_port）；
            gip_disconnect_from_ring（my_port）；
            my_port->is_connected = False；
}     }


/******************************************************** ****************************
 * GIP：GARP 信息传播：传播单一属性
 ********************************************************************************************
 * /

void gip_propagate_join(Gid *my_port，无符号gid_index){/*
      * 传播加入指示，向其他端口发出加入请求
      * 如果需要。
      *
      * 如果 (a) 这是第一个端口，则需要传播连接
      * 连接组注册成员资格，或 (b) 有另一个端口
      * 在组中注册成员身份，但没有进一步的端口，这会导致
      * 向该端口发出加入请求。
      *
      * /
      未签名          加入会员；
      吉德          * 到达港口；

      如果（my_port->is_connected）{

            加入成员 =（my_port->应用程序->gip[gid_index] += 1）；

            如果 (加入成员 <= 2) {

                  到_端口=我的_端口；
                  当（（to_port = to_port->next_in_connected_ring）！ = my_port）{

                        如果 （  （加入成员==1）
                              ||    （gid_registered_here（to_port，gid_index））
                        {
                              gid_join_request（to_port，to_port->应用程序，
                              >join_propagated_fn（
                                                                        my_port->应用程序，
                                                                      我的端口，gi d索引）；
}     }     }     }     }
```

```
void gip_propagate_leave(Gid *my_port，无符号gid_index) {

    /* 传播单个属性的离开指示，导致离开
     * 如果需要，向其他端口发出请求。
     *
     * 在进一步阅读之前，请参阅 gip_propagate_join() 的注释。
     * 此功能减少“航位推算”成员数量。
     *
     * 第一步是检查此端口是否连接到其他端口；如果
     * 不，离开指示不应该被传播，加入的指示也不应该被传播
     * 会员资格减少。否则，休假将需要传播
     * 如果这是 (a) 连接组中最后一个要注册的端口
     * 成员资格，或 (b) 该组中有另一个端口正在注册
     * 会员资格，在这种情况下，休假申请需要发送给该
     * 单独端口。
     * /
    未签名         剩余成员；
    吉德          * 到达港口；

    如果（my_port->is_connected）{

        剩余成员 = （my_port->application->gip[gid_index] -= 1）；

        如果 (剩余成员 <= 1) {

            到_端口=我的_端口；
            当（（to_port = to_port->next_in_connected_ring）！ = my_port）{

                如果 （  （剩余成员 == 0）（gid_registered_here（to_port，
                    ||   gid_index）））
                {
                    gid_leave_request（to_port，to_port->gid_应用程序，
                    >leave_propagated_fn（
                                                                    my_port->应用程序，
                                                                    我的端口，gi d索引）；
}   }   }   }   }


布尔 gip_propagates_to (Gid *my_port，无符号 gid_index) {

    如果 （                (my_port->is_connected) (my_port->application-
        && （              >gip[gid_index] (my_port->application->gip[gid_index-1] (2)
            ||  （         gid_registered_here(my_port, gid_index)))))          ==  1）
                &&
        返回（真）；
    否则返回（False）；
}

/******************************************************* ****************************
 * GIP：GARP 信息传播：行动计时器
 ********************************************************************************
 * /

void gip_do_actions（Gid *my_port）{/*

    * 调用 GID 执行此过程中积累的 GID“暂存器”操作
    * 对 GIP 环中的所有端口（包括我的端口）调用 GARP。
    * /
    Gid *this_port = my_port;

    执行 gid_do_actions(this_port);
    当（（this_port = this_port->next_in_connected_ring）！ = my_port）；
}
```

14. 桥梁管理

MAC 桥接器根据 OSI 管理框架的原理和概念提供管理设施。

本条款

a) 介绍OSI管理的功能领域，以协助确定对网桥支持管理设施的要求。

b) 建立用于模拟桥接操作的流程之间的对应关系
（7.3）以及Bridge的管理对象。
c) 指定每个管理对象支持的管理操作。

14.1 管理职能

管理功能与用户对支持通信资源的规划、组织、监督、控制、保护和安全以及说明其使用的设施的需求有关。这些设施可归类为支持配置、故障、性能、安全和会计管理的功能领域。14.1.1 至 14.1.5 中总结了这些设施，以及管理通信资源通常所需的设施以及桥梁管理在该功能领域提供的特定设施。

14.1.1 配置管理

配置管理提供通信资源的识别、初始化、重置和关闭、操作参数的提供以及资源之间关系的建立和发现。桥梁管理在此功能区提供的设施包括

a) 识别组成桥接 LAN 的所有桥接器及其各自的位置，并根据该识别结果，识别特定终端站到特定单个 LAN 的位置。

b) 远程重置（即重新初始化）指定桥梁的能力。
c) 控制桥接端口传输帧的优先级的能力。
d) 强制特定生成树配置的能力。
e) 能够控制具有特定组 MAC 地址的帧传播到已配置的桥接 LAN 的某些部分。

14.1.2 故障管理

故障管理提供故障预防、检测、诊断和纠正。桥梁管理在此功能区提供的设施包括

a) 识别和纠正桥梁故障的能力，包括错误记录和报告。

14.1.3 绩效管理

绩效管理用于评估通信资源的行为和通信活动的有效性。桥梁管理在此功能领域提供的设施包括

a) 收集与性能和流量分析相关的统计数据的能力。具体指标包括网桥内各个端口的网络利用率、帧转发和帧丢弃计数。

14.1.4 安全管理

安全管理负责保护资源。桥梁管理不提供此功能区的任何特定设施。

14.1.5 会计管理

会计管理用于确定和分配成本以及设置费用。桥梁管理不提供此功能区的任何特定设施。

14.2 管理对象

管理对象为管理操作的语义建模。对对象的操作提供有关该对象相关流程或实体的信息，或促进对该流程或实体的控制。

MAC 桥接器的管理资源是 7.3 和 12.2 中建立的进程和实体的资源。具体来说

a)桥梁管理实体（14.4和7.11）。
b) 与每个桥接端口相关的单独 MAC 实体 (14.5、7.2、7.5 和 7.6)。
c) MAC中继实体的转发过程（14.6、7.2和7.7）。
d) MAC中继实体的过滤数据库（14.7和7.9）。
e) 桥接协议实体（14.8、7.10 和第 8 条）。
f) GARP 参与者（第 12 条）。

下面根据管理对象和操作来描述每种资源的管理。

注：本条款中指定的值作为管理操作的输入和输出，是抽象的信息元素。格式或编码问题属于传达或以其他方式表示此信息的特定协议的问题。本标准在第 15 条款中规定了一种这样的协议编码（用于可选的远程管理）。

14.3 数据类型

本小节规定了操作的语义，与管理协议中的编码无关。操作参数的数据类型仅按该规范的要求进行定义。

使用以下数据类型：

a）布尔值；
b) 枚举，用于命名值的集合；
c) 无符号，对于所有指定为某个量的“数量”的参数，以及用于数字比较的值。在对生成树优先级值进行数字比较的情况下，数字越小，优先级值越高；

d) MAC 地址；
e) Latin1 字符串，由 ANSI X3.159-1989 定义，适用于所有文本字符串；
f）时间间隔，一个无符号值，代表所有生成树协议超时参数的正整数秒数；

g) 计数器，所有参数均指定为某个数量的“计数”。计数器以 2 的 64 次方为模数递增和回绕；

h) GARP 时间间隔，一个无符号值，代表所有 GARP 协议超时参数的正整数厘秒数。

14.4 桥梁管理实体

桥梁管理实体在 7.11 中描述。

组成此管理资源的对象是

a) 桥梁配置。
b) 每个端口的端口配置。

14.4.1 桥接配置

Bridge 配置对象模拟了修改或查询 Bridge 资源配置的操作。每个 Bridge 都有一个 Bridge 配置对象。

可以在 Bridge Configuration 上执行的管理操作有 Discover Bridge、Read Bridge、Set Bridge Name 和 Reset Bridge。

14.4.1.1 发现桥

14.4.1.1.1 目的

请求有关桥接 LAN 中的桥接器的配置信息。

14.4.1.1.2 输入

a) 包含范围，一组有序的特定 MAC 地址对。每对指定一个 MAC 地址范围。当且仅当以下情况时，网桥才应响应
   1) 对于其中一对，其桥接地址与该对中每个 MAC 地址的数值比较表明它大于或等于第一个，并且
   2）小于或等于秒，并且
   3)其桥地址没有出现在下面的排除列表参数中。

为了进行此操作，一个 MAC 地址与另一个 MAC 地址之间的数字比较是通过按照以下步骤从 MAC 地址中得出一个数字来实现的。MAC 地址的连续八位字节被视为二进制数；当 MAC 地址用于 MAC 帧的源或目标字段时，在 LAN 介质上传输的第一个八位字节具有最高有效值，下一个八位字节具有下一个最高有效值。在每个八位字节中，每个八位字节的第一位是最低有效位。

b) 排除列表，特定 MAC 地址的列表。

14.4.1.1.3 输出

a) 桥接地址——桥接器的 MAC 地址，生成树算法和协议使用的桥接器标识符由此地址派生。

b) 桥梁名称——最多 32 个字符的文本字符串，具有本地确定的意义。
c) 端口数——桥接端口（MAC 实体）的数量。
d) 端口地址——每个端口指定以下内容的列表：
   1) 端口号——桥接端口的号码。
   2) 端口地址——与端口关联的单个 MAC 实体的特定 MAC 地址。

e) 正常运行时间——自 Bridge 上次重置或初始化以来经过的时间（以秒为单位）。

14.4.1.2 读桥

14.4.1.2.1 目的

获取有关大桥的一般信息。

14.4.1.2.2 输入

没有任何。

14.4.1.2.3 输出

a) 桥接地址——桥接器的 MAC 地址，生成树算法和协议使用的桥接器标识符由此地址派生。

b) 桥梁名称——最多 32 个字符的文本字符串，具有本地确定的意义。
c) 端口数——桥接端口（MAC 实体）的数量。
d) 端口地址——每个端口指定以下内容的列表：
    1)端口号。
    2) 端口地址——与端口关联的单个 MAC 实体的特定 MAC 地址。

e) 正常运行时间——自 Bridge 上次重置或初始化以来经过的时间（以秒为单位）。

14.4.1.3 设置桥名称

14.4.1.3.1 目的

将读取桥操作可读取的文本字符串与桥关联。

14.4.1.3.2 输入

a) 桥梁名称——最多 32 个字符的文本字符串。

14.4.1.3.3 输出

没有任何。

14.4.1.4 重置桥

14.4.1.4.1 目的

重置指定的桥接器。清除过滤数据库并用永久数据库中指定的条目进行初始化，并初始化桥接协议实体（8.8.1）。

14.4.1.4.2 输入

没有任何。

14.4.1.4.3 输出

没有任何。

14.4.2 端口配置

端口配置对象模拟了修改或查询桥接端口配置的操作。每个桥接都有一组固定的桥接端口（每个 MAC 接口一个），每个端口都由永久分配的端口号标识。

分配的端口号不要求连续。此外，一些端口号可能是虚拟条目，没有实际的 LAN 端口（例如，允许将来通过添加更多 MAC 接口来扩展网桥）。此类虚拟端口应以与永久禁用端口一致的方式支持端口配置管理操作和其他与端口相关的管理操作。

端口配置提供的信息包括指示其名称和类型的摘要数据。转发过程资源维护与转发、过滤和错误数据包数量相关的特定计数器信息。桥接协议实体支持的管理操作允许控制每个端口的状态。

可以对端口配置执行的管理操作有读取端口和设置端口名称。

14.4.2.1 读端口

14.4.2.1.1 目的

获取有关特定桥接端口的一般信息。

14.4.2.1.2 输入

a) 端口号——桥接端口的编号。

14.4.2.1.3 输出

a) 端口名称——最多 32 个字符的文本字符串，具有本地确定的意义。
b) 端口类型——端口的 MAC 实体类型（IEEE 标准 802.3；ISO/IEC 8802-4；ISO/IEC 8802-5；ISO/IEC 8802-6；ISO/IEC 8802-9；IEEE 802.9a；ISO/IEC 8802-12（IEEE 标准 802.3 格式）；ISO/IEC 8802-12（ISO/IEC 8802-5 格式）；IEEE 标准 802.11；ISO 9314-2；其他）。

14.4.2.2 设置端口名称

14.4.2.2.1 目的

将读取端口操作可读的文本字符串与桥接端口关联。

14.4.2.2.2 输入

a) 端口号。
b) 端口名称——最多 32 个字符的文本字符串。

14.4.2.2.3 输出

没有任何。

14.5 MAC实体

MAC 实体提供的管理操作和设施是各个 MAC 的层管理标准中指定的。每个桥接端口都与一个 MAC 实体相关联。

14.6 转发过程

转发进程包含与帧转发相关的信息。计数器会提供转发、过滤和由于错误而丢弃的帧数信息。配置数据（定义如何处理帧优先级）由转发进程维护。

组成此管理资源的对象是

a) 港口柜台；
b) 每个端口的优先级处理对象；
c) 每个端口的流量类别表。

14.6.1 端口计数器

端口计数器对象模拟了可以在转发过程资源的端口计数器上执行的操作。每个网桥有多个端口计数器对象实例（每个 MAC 实体一个）。

可以对端口计数器执行的管理操作是读取转发端口计数器。

14.6.1.1 读取转发端口计数器

14.6.1.1.1 目的

读取与特定桥接端口相关的转发计数器。

14.6.1.1.2 输入

a) 端口号。

14.6.1.1.3 输出

a) 已接收帧数——已接收的所有有效帧的数量（包括 BPDU、以终端站形式发送到网桥的帧以及提交至转发过程的帧）。
b) 丢弃入站——转发过程丢弃的已接收有效帧的数量。
c) 转发出站——转发到相关 MAC 实体的帧数。
d) 丢弃缺少缓冲区——要通过相关端口传输但由于缺少缓冲区而被丢弃的帧的数量。

e) 丢弃传输延迟超出——由于超出最大桥接传输延迟（可能有缓冲可用）而被丢弃的待传输帧的数量。
f) 错误时丢弃——在相关 MAC 上应转发但无法传输的帧数（例如，帧太大）。

g) 错误时丢弃详细信息——包含 16 个元素的列表，每个元素包含帧的源地址以及丢弃该帧的原因（帧太大）。该列表作为循环缓冲区进行维护。
目前，错误丢弃的唯一原因是可传输服务数据单元大小超出。

14.6.2 优先级处理

优先级处理对象模拟了可以对每个端口的默认用户优先级参数、用户优先级再生表参数和出站访问优先级表参数执行或查询的操作。可以对此对象执行的操作包括读取端口默认用户优先级、设置端口默认用户优先级、读取端口用户优先级再生表、设置端口用户优先级再生表和读取出站访问优先级表。

14.6.2.1 读取端口默认用户优先级

14.6.2.1.1 目的

读取特定桥接端口的默认用户优先级参数 (6.4) 的当前状态。

14.6.2.1.2 输入

a) 端口号。

14.6.2.1.3 输出

a) 默认用户优先级值 - 0-7 范围内的整数。

14.6.2.2 设置端口默认用户优先级

14.6.2.2.1 目的

设置特定桥接端口的默认用户优先级参数 (6.4) 的当前状态。

14.6.2.2.2 输入

a) 端口号；
b) 默认用户优先级值 - 0-7 范围内的整数。

14.6.2.2.3 输出

没有任何。

14.6.2.3 读取端口用户优先级再生表

14.6.2.3.1 目的

读取特定桥接端口的用户优先级再生表参数 (7.5.1) 的当前状态。

14.6.2.3.2 输入

a) 端口号。

14.6.2.3.3 输出

a) 接收用户优先级的重新生成的用户优先级值 0—0-7 范围内的整数。
b) 接收用户优先级 1 的重新生成的用户优先级值 - 0-7 范围内的整数。
c) 为接收的用户优先级 2 重新生成的用户优先级值 - 范围为 0-7 的整数。
d) 接收用户优先级 3 的重新生成的用户优先级值 - 0-7 范围内的整数。
e) 接收用户优先级 4 的重新生成的用户优先级值 - 0-7 范围内的整数。

f) 接收用户优先级 5 的重新生成的用户优先级值 - 0-7 范围内的整数。
g) 接收用户优先级 6 的重新生成的用户优先级值 - 0-7 范围内的整数。
h) 接收用户优先级 7 的重新生成的用户优先级值 - 0-7 范围内的整数。

14.6.2.4 设置端口用户优先级再生表

14.6.2.4.1 目的

设置特定桥接端口的用户优先级再生表参数 (7.5.1) 的当前状态。

14.6.2.4.2 输入

a) 端口号；
b) 接收用户优先级的重新生成的用户优先级值 0—0-7 范围内的整数。
c) 接收用户优先级 1 的重新生成的用户优先级值 - 0-7 范围内的整数。
d) 接收用户优先级 2 的重新生成的用户优先级值 - 0-7 范围内的整数。
e) 接收用户优先级 3 的重新生成的用户优先级值 - 0-7 范围内的整数。
f) 接收用户优先级 4 的重新生成的用户优先级值 - 0-7 范围内的整数。
g) 接收用户优先级 5 的重新生成的用户优先级值 - 0-7 范围内的整数。
h) 接收用户优先级 6 的重新生成的用户优先级值 - 0-7 范围内的整数。
i) 为接收的用户优先级 7 重新生成的用户优先级值 - 范围为 0-7 的整数。

14.6.2.4.3 输出

没有任何。

14.6.3 流量类别表

流量类别表对象模拟了可以对给定端口的流量类别表的当前内容执行的操作或查询操作。可以对此对象执行的操作包括读取端口流量类别表和设置端口流量类别表。

14.6.3.1 读取端口流量类别表

14.6.3.1.1 目的

读取给定端口的流量类别表 (7.7.3) 的内容。

14.6.3.1.2 输入

a) 端口号。

14.6.3.1.3 输出

a) 端口支持的流量类别数，范围为 1 至 8；
b) 对于端口上支持的每个流量类别值，流量类别的值在 0 到 7 范围内，以及分配给该流量类别的 user_priority 值集合。

14.6.3.2 设置端口流量类别表

14.6.3.2.1 目的

设置给定端口的流量类别表 (7.7.3) 的内容。

14.6.3.2.2 输入

a) 端口号；
b) 对于端口上支持的每个流量类别值，流量类别的值在 0 到 7 范围内，以及分配给该流量类别的 user_priority 值集合。

注意 — 如果指定的流量类别值大于端口上可用的最大流量类别，则应用于流量类别表的值是最大的可用流量类别。

14.6.3.2.3 输出

没有任何。

14.6.3.3 读取出站访问优先级表

14.6.3.3.1 目的

读取特定桥接端口的出站访问优先级表参数（表 7-3）的状态。

14.6.3.3.2 输入

a) 端口号。

14.6.3.3.3 输出

a) 用户优先级 0 的访问优先级值 - 0-7 范围内的整数。
b) 用户优先级 1 的访问优先级值 - 0-7 范围内的整数。
c) 用户优先级 2 的访问优先级值 - 0-7 范围内的整数。
d) 用户优先级 3 的访问优先级值 - 0-7 范围内的整数。
e) 用户优先级 4 的访问优先级值 - 0-7 范围内的整数。
f) 用户优先级 5 的访问优先级值 - 0-7 范围内的整数。
g) 用户优先级 6 的访问优先级值 - 0-7 范围内的整数。
h) 用户优先级 7 的访问优先级值 - 0-7 范围内的整数。

14.7 过滤数据库

过滤数据库在 7.9 中描述。它包含转发过程 (7.7) 使用的过滤信息，用于决定应通过桥接的哪个端口转发帧。

组成此管理资源的对象是

a) 过滤数据库；
b) 静态过滤条目；
c)动态过滤条目；
d) 团体登记条目；
e) 永久数据库。

14.7.1 过滤数据库

过滤数据库对象模拟了可对整个过滤数据库执行或影响的操作。每个桥接器都有一个过滤数据库对象。

可以对数据库执行的管理操作有读取过滤数据库、设置过滤数据库老化时间以及14.7.6中定义的创建过滤条目、删除过滤条目、读取过滤条目和读取过滤条目范围操作。

14.7.1.1 读取过滤数据库

14.7.1.1.1 目的

获取有关 Bridge 过滤数据库的一般信息。

14.7.1.1.2 输入

没有任何。

14.7.1.1.3 输出

a) 过滤数据库大小——过滤数据库中可保存的最大条目数。
b) 静态过滤条目的数量——当前过滤数据库中的静态过滤条目的数量；

c) 动态过滤条目的数量——当前过滤数据库中的动态过滤条目的数量；

d) 老化时间——当与条目关联的端口处于转发状态时，用于老化动态过滤条目；

e) 如果支持扩展过滤服务，组注册条目数——当前在过滤数据库中的组注册条目数。

14.7.1.2 设置过滤数据库老化时间

14.7.1.2.1 目的

设置动态过滤条目的老化时间。

14.7.1.2.2 输入

a) 老化时间。

14.7.1.2.3 输出

没有任何。

14.7.2 静态过滤条目

静态过滤条目对象模拟了可以在过滤数据库中的单个静态过滤条目上执行的操作。过滤数据库中的静态过滤条目对象集仅在管理控制下才会更改。

静态过滤条目对象支持14.7.6中定义的创建过滤条目、删除过滤条目、读取过滤条目和读取过滤条目范围操作。

14.7.3 动态过滤条目

动态过滤条目对象模拟了可以在过滤数据库中的单个动态过滤条目（即，由学习过程根据对网络流量的观察而创建的条目）上执行的操作。

动态过滤条目对象支持14.7.6中定义的删除过滤条目、读取过滤条目和读取过滤条目范围操作。

14.7.4 组注册条目

组注册条目对象模拟了可以在过滤数据库中的单个组注册条目上执行的操作。过滤数据库中的组注册条目对象集只会因 GARP 协议交换而发生变化。

组注册条目对象支持 14.7.6 中定义的读取过滤条目和读取过滤条目范围操作。

14.7.5 永久数据库

永久数据库对象模拟了可对永久数据库执行或影响永久数据库的操作。每个过滤数据库都有一个永久数据库。

永久数据库上可以执行的管理操作有读取永久数据库，以及14.7.6中定义的创建过滤条目、删除过滤条目、读取过滤条目和读取过滤条目范围操作。

14.7.5.1 读取永久数据库

14.7.5.1.1 目的

获取有关永久数据库的一般信息。

14.7.5.1.2 输入

没有任何。

14.7.5.1.3 输出

a) 永久数据库大小——永久数据库中可保存的最大条目数。
b) 静态过滤条目的数量——当前位于永久数据库中的静态过滤条目的数量。

14.7.6 常规过滤数据库操作

14.7.6.1 创建过滤条目

14.7.6.1.1 目的

在过滤数据库或永久数据库中创建或更新过滤条目。在过滤数据库或永久数据库中只能创建静态过滤条目。

14.7.6.1.2 输入

a) 标识符——过滤数据库或永久数据库。
b) 地址——条目的 MAC 地址。
c) 入站端口 — 操作适用的入站端口。此参数指定
    1）所有入站端口，或
    2）端口号。
d) 端口映射——一组控制指示符，每个端口一个，如 7.9.1 中所述。

如果实现不支持为指定的地址创建多个静态过滤条目，则假定入站端口参数的值指定所有入站端口。

如果实现不支持静态过滤条目指定使用动态过滤信息（7.9.1）的能力，则使用此操作在过滤数据库中创建具有与现有动态过滤条目相同 MAC 地址的静态过滤条目将导致现有条目被（新的）静态过滤条目替换。

如果实现支持为同一 MAC 地址 (7.9.1) 创建多个静态过滤条目，则创建新的静态过滤条目将导致同一入站端口和 MAC 地址的任何现有静态过滤条目被（新）静态过滤条目替换。为所有入站端口创建静态过滤条目将导致同一 MAC 地址的所有现有静态条目被（新）静态过滤条目替换。

14.7.6.1.3 输出

没有任何。

14.7.6.2 删除过滤条目

14.7.6.2.1 目的

从过滤数据库或永久数据库中删除过滤条目。

14.7.6.2.2 输入

a) 标识符——过滤数据库或永久数据库。
b) 地址——所需条目的 MAC 地址。
c) 入站端口 — 操作适用的入站端口。此参数指定
    1）所有入站端口，或
    2）端口号。

如果实现不支持为指定的地址创建多个静态过滤条目，则假定入站端口参数的值指定为所有入站端口。

当实施支持为指定地址创建多个静态过滤条目时，所有入站端口的入站端口参数值将导致删除指定 MAC 地址的所有静态过滤条目。

14.7.6.2.3 输出

没有任何。

14.7.6.3 读取过滤条目

14.7.6.3.1 目的

从过滤或永久数据库中读取过滤条目和组注册条目信息。此操作返回给定数据库中保存的给定 MAC 地址和入站端口规范的静态和动态信息。

14.7.6.3.2 输入

a) 标识符——过滤数据库或永久数据库。

b) 地址——所需信息的 MAC 地址。
c) 类型——静态或动态条目。
d) 如果类型 = 静态条目，则入站端口 — 操作适用的入站端口。此参数指定

1）所有入站端口，或
2）端口号。

如果实现不支持为指定的地址创建多个静态过滤条目，则假定入站端口参数的值指定为所有入站端口。

注意——动态条目类型包括动态过滤条目和组注册条目。

14.7.6.3.3 输出

a) 地址——所需条目的 MAC 地址。
b) 类型——静态或动态条目。
c) 端口映射——一组适合于条目类型的控制指示符，如 7.9.1 至 7.9.3.

14.7.6.4 读取过滤条目范围

14.7.6.4.1 目的

从过滤或永久数据库中读取一系列过滤条目和/或组注册条目。

由于请求范围内要返回的值的数量可能超出了传达管理响应的服务数据单元的容量，因此确定了返回的条目范围。定义范围的索引取值范围从零到过滤数据库大小减一。

14.7.6.4.2 输入

a) 标识符——过滤数据库或永久数据库。
b) 起始索引——所需输入范围的包含起始索引。
c) 停止索引——所需范围的包含结束索引。

14.7.6.4.3 输出

a) 起始索引——返回条目范围的包含起始索引。
b) 停止索引——返回的条目范围的包含结束索引。
c) 对于返回的条目范围的每个索引，返回以下内容：
1) 地址——所需条目的 MAC 地址。
2) 类型——静态或动态条目。
3) 端口映射——一组适合条目类型的控制指示符，如 7.9.1 至 7.9.3 中所述。

14.8 桥接协议实体

桥接协议实体在 7.10 和第 8 条中描述。组成此管理资源的对象是

a) 协议实体本身和
b) 其控制下的港口。

14.8.1 协议实体

协议实体对象模拟了可以执行的操作，或查询生成树算法和协议的操作。每个网桥都有一个协议实体；因此，它可以被识别为协议实体资源的单个固定组件。

协议实体上可以执行的管理操作有读取桥接协议参数和设置桥接协议参数。

14.8.1.1 读取桥接协议参数

14.8.1.1.1 目的

获取有关桥接协议实体的信息。

14.8.1.1.2 输入

没有任何。

14.8.1.1.3 输出

a) 桥梁标识符——如 8.5.3.7 所定义。
b) 自拓扑变化以来的时间——自桥接器（8.5.3.12）的拓扑变化标志参数上次为 True 以来经过的时间（以秒为单位）。
c) 拓扑变化计数——自桥接器通电或初始化以来，桥接器拓扑变化标志参数被设置的次数（即，从 False 变为 True）。
d)拓扑改变（8.5.3.12）。
e) 指定根（8.5.3.1）。
f) 根路径成本（8.5.3.2）。
g) 根端口（8.5.3.3）。
h) 最大年龄（8.5.3.4）。
i）你好时间（8.5.3.5）。
j)转发延迟（8.5.3.6）。
k) 桥梁最大年龄（8.5.3.7）。
l) 桥接问候时间 (8.5.3.9)。
m)桥接转发延迟（8.5.3.10）。
n)保持时间（8.5.3.14）。

14.8.1.2 设置桥接协议参数

14.8.1.2.1 目的

修改桥接协议实体中的参数，以强制配置生成树和/或调整重新配置时间以适应特定的拓扑。

14.8.1.2.2 输入

a) 桥梁最大年龄——新值（8.5.3.8）。
b) 桥接问候时间——新值（8.5.3.9）。
c) 桥接转发延迟——新值（8.5.3.10）。
d) 桥接优先级——桥接标识符优先级部分的新值（8.5.3.7）。

14.8.1.2.3 输出

没有任何。

14.8.1.2.4 程序

检查输入参数值是否符合 8.10.2。如果不符合，或者 Bridge Max Age 或 Bridge Forward Delay 的值小于表 8-3 中规定的范围的下限，则不对任何提供的参数采取任何措施。如果 Bridge Max Age、Bridge Forward Delay 或 Bridge Hello Time 中的任何一个值超出表 8-3 中规定的范围，则 Bridge 无需采取行动。

否则，将桥的桥最大年龄、桥问候时间和桥转发延迟参数设置为提供的值。设置桥优先级程序 (8.8.4) 用于将桥标识符的优先级部分设置为提供的值。

14.8.2 桥接端口

桥接端口对象根据生成树算法和协议的操作对与单个桥接端口相关的操作进行建模。每个桥接都有一组固定的桥接端口；因此，每个端口都可以通过永久分配的端口号进行标识，作为协议实体资源的固定组件。可以在桥接端口上执行的管理操作包括读取端口参数、强制端口状态和设置端口参数。

14.8.2.1 读取端口参数

14.8.2.1.1 目的

获取有关桥接协议实体内特定端口的信息。

14.8.2.1.2 输入

a) 端口号——桥接端口的编号。

14.8.2.1.3 输出

a) 正常运行时间——自端口上次重置或初始化以来经过的时间（以秒为单位）。
b) 状态——端口的当前状态（即禁用、监听、学习、转发或阻塞）（8.4 和 8.5.5.2）。

c) 端口标识符——唯一的端口标识符，由两部分组成：端口号和端口优先级字段（8.5.5.1）。

d)路径成本（8.5.5.3）。
e) 指定根（8.5.5.4）。
f) 指定费用（8.5.5.5）。
g) 指定桥梁（8.5.5.6）。
h)指定端口（8.5.5.7）。
i)拓扑改变确认（8.5.5.8）。

14.8.2.2 强制端口状态

14.8.2.2.1 目的

强制将指定端口禁用或阻塞。

14.8.2.2.2 输入

a) 端口号——桥接端口的编号。
b) 状态——禁用或阻塞（8.4 和 8.5.5.2）。

14.8.2.2.3 输出

没有任何。

14.8.2.2.4 程序

如果选定的状态为禁用，则对指定端口使用禁用端口程序 (8.8.3)。如果选定的状态为阻塞，则使用启用端口程序 (8.8.2)。

14.8.2.3 设置端口参数

14.8.2.3.1 目的

修改桥接协议实体中端口的参数，以强制配置生成树。

14.8.2.3.2 输入

a) 端口号——桥接端口的编号。
b) 路径成本——新值（8.5.5.3）。
c) 端口优先级——端口标识符优先级字段的新值（8.5.5.1）。

14.8.2.3.3 输出

没有任何。

14.8.2.3.4 程序

设置路径成本过程 (8.8.6) 用于设置指定端口的路径成本参数。设置端口优先级过程 (8.8.5) 用于将端口标识符 (8.5.5.1) 的优先级部分设置为提供的值。

14.9 GARP 实体

GARP 的操作在第 12 章中描述。组成此管理资源的对象包括

a) GARP 计时器对象；
b) GARP 属性类型对象；
c) GARP 状态机对象。

14.9.1 GARP 计时器对象

GARP 计时器对象模拟了可对给定端口上 GARP 协议使用的计时器的当前设置执行的操作或查询这些设置。可对 GARP 计时器对象执行的管理操作包括读取 GARP 计时器和设置 GARP 计时器。

14.9.1.1 读取 GARP 计时器

14.9.1.1.1 目的

读取给定端口的 GARP 计时器的当前值。

14.9.1.1.2 输入

a) 端口标识符。

14.9.1.1.3 输出

a) JoinTime 的当前值—厘秒；
b) LeaveTime的当前值——厘秒；
c) LeaveAllTime 的当前值——厘秒。

14.9.1.2 设置 GARP 定时器

14.9.1.2.1 目的

为给定端口的 GARP 计时器设置新值。

14.9.1.2.2 输入

a) 端口标识符；
b) JoinTime 的新值—厘秒；
c) LeaveTime的新值—厘秒；
d) LeaveAllTime 的新值—厘秒。

14.9.1.2.3 输出

没有任何。

14.9.2 GARP 属性类型对象

GARP 属性类型对象模拟了可以对给定属性类型执行或查询 GARP 操作的操作。可以对 GARP 属性类型执行的管理操作包括读取 GARP 申请人控制和设置 GARP 申请人控制。

14.9.2.1 读取 GARP 申请人控制

14.9.2.1.1 目的

读取与给定端口、GARP 应用程序和属性类型的所有 GARP 参与者关联的 GARP 申请人管理参数 (12.9.2) 的当前值。

14.9.2.1.2 输入

a) 端口标识符；
b) GARP应用地址（表12-1）；
c) 属性类型（12.11.2.2）。

14.9.2.1.3 输出

a) 当前申请人的行政控制值（12.9.2）；
b) 注册失败 - 由于过滤数据库空间不足，此 GARP 应用程序无法注册此类型属性的次数。

14.9.2.2 设置 GARP 申请人控制

14.9.2.2.1 目的

为与给定端口、GARP 应用程序和属性类型的所有 GARP 参与者关联的 GARP 申请人管理控制参数 (12.9.2) 设置新值。

14.9.2.2.2 输入

a) 端口标识符；
b) GARP应用地址（表12-1）；
c) 属性类型（12.11.2.2）；
d) 申请人所需的行政控制值（12.9.2）。

14.9.2.2.3 输出

没有任何。

14.9.3 GARP 状态机对象

GARP 状态机对象模拟了可以对给定状态机执行或查询 GARP 操作的操作。可以在 GARP 状态机上执行的管理操作是读取 GARP 状态。

14.9.3.1 读取GARP状态

14.9.3.1.1 目的

读取 GARP 状态机实例的当前值。

14.9.3.1.2 输入

a) 端口标识符；
b) GARP应用地址（表12-1）；
c) GIP 背景（12.3.4）；
d) 与状态机相关的属性类型（12.11.2.2）；
e) 与状态机相关的属性值（12.11.2.6）。

14.9.3.1.3 输出

a) 申请人和注册商组合状态机的属性当前值（表 12-
6);
b) 可选，发起者地址 - 导致此状态机状态改变的最新 GARP PDU 发起者的 MAC 地址 (12.9.1)。

15. 管理规程

本节规定了第 14 章中规定的 MAC 桥提供的管理功能是如何通过使用

a) 通过 LAN/MAN 管理协议（LMMP）和融合协议实体（CPE）提供的 LAN/MAN 管理服务（LMMS），如 ISO/IEC 15802- 2 所规定；或

b) 通用管理信息服务（CMIS），通过通用管理信息协议（CMIP）和表示 P-DATA 服务提供，如 ISO/IEC 9595 和 ISO/IEC 9596-1 所规定；或

c) 能够传达等效信息的其他管理协议。

注 1—除了关联控制服务支持之外，LMMS 和 CMIS 的服务原语定义相同，LMMP 和 CMIP 规定的 PDU 格式和程序也相同。因此，这两个选项之间的区别在于支持 LMMP（CPE 加上 LLC、MAC 和 PHY）或 CMIP（通过“完整 OSI 堆栈”承载的演示服务）所需的协议堆栈的复杂性，而不是服务定义或它们所传达的信息。因此，为简单起见，本节的其余部分将仅涉及 LMMS 和 LMMP。

注 2 — IETF 已开发出等效协议规范 RFC 1493，该规范定义了用于简单网络管理协议 (SNMP) 的桥接 MIB。RFC 1493 定义了与 ISO/IEC 10038：1993 版 MAC 桥接标准中定义的管理功能相对应的 MIB；目前正在对其进行修订，以反映此版 MAC 桥接标准中定义的管理功能。

它指定

a) 第14条定义的对象的管理操作与LMMP提供的LMMS服务之间的映射；

b) 为提供第 14 条中定义的管理对象而实例化的管理对象类（及其组件）。这些管理对象类定义使用 ISO/IEC 10165-4 中指定的模板符号进行记录；记录的其他方面符合 IEEE Std. 802.1F-1993 提供的进一步指导。

c) 名称绑定是必要的，用于指定如何命名管理对象类，以及管理对象类之间允许的包含关系。还提供名称绑定，以便正确定位 MAC 桥接对象包含在 ISO/IEC 10742 中为管理数据链路层定义的包含层次结构中。

15.1 将操作映射到 LMMS 服务

第14条定义的对象的操作通过以下LMMS服务进行：

a) M-GET
b) M-设置
c) M-ACTION
d) M-创造
e) M-删除
f) M-取消-获取

表 15-1 至表 15-4 显示了第 14 条中定义的管理操作如何与 LMMS 服务原语（在 ISO/IEC 15802-2 中定义）和管理对象类（在 15.3 至 15.8 中定义）相对应。表 15-6 显示了可用于操作选择性转换表的操作，如 ISO/IEC 11802-5 中定义。

表15-7至表15-33指定了管理操作参数到LMMS服务原语参数的详细映射。

表 15-1—桥接管理实体操作与 LMMS 服务的映射

管理运营	LMMS 服务元素	管理对象类
发现桥（14.4.1.1）	麦格特	MAC 桥接器 DLE (15.3)
读桥（14.4.1.2）	麦格特	MAC 桥接器 DLE (15.3)
设置桥名称（14.4.1.3）	设定	MAC 桥接器 DLE (15.3)
重置桥接器（14.4.1.4）	M-动作	MAC 桥接器 DLE (15.3)
读取端口（14.4.2.1）	麦格特	端口 (15.4)
设置端口名称（14.4.2.2）	设定	端口 (15.4)

表 15-2—转发流程操作到 LMMS 服务的映射

管理运营	LMMS 服务元素	管理对象类
读取转发端口计数器（14.6.1.1）	麦格特	端口 (15.4)
读取端口默认用户优先级（14.6.2.1）	麦格特	端口 (15.4)
设置端口默认用户优先级（14.6.2.2）	设定	端口 (15.4)
读取端口用户优先级再生表（14.6.2.3）	麦格特	端口 (15.4)
设置端口用户优先级再生表（14.6.2.4）	设定	端口 (15.4)
读取端口流量类别表（14.6.3.1）	麦格特	端口 (15.4)
设置端口流量类别表（14.6.3.2）	设定	端口 (15.4)
读取出站访问优先级表（14.6.3.3）	麦格特	端口 (15.4)

表 15-3—过滤数据库操作与 LMMS 服务的映射

管理运营	LMMS 服务元素	管理对象类
读取过滤数据库（14.7.1.1）	麦格特	过滤数据库 (15.7)
设置过滤数据库老化时间（14.7.1.2）	设定	过滤数据库 (15.7)
读取永久数据库（14.7.5.1）	麦格特	永久数据库（15.7）
创建过滤条目（14.7.6.1）	M-创造	数据库条目（15.8）
删除过滤条目（14.7.6.2）	删除	数据库条目（15.8）
读取过滤条目（14.7.6.3）	麦格特	数据库条目（15.8）
读取过滤条目范围（14.7.6.4）	M-获取，M-取消获取	数据库条目（15.8）

表 15-4—桥接协议实体操作到 LMMS 服务的映射

管理运营	LMMS 服务元素	管理对象类
读取桥接协议参数（14.8.1.1）	麦格特	MAC 桥接器 DLE (15.3)
设置桥接协议参数（14.8.1.2）	设定	MAC 桥接器 DLE (15.3)
读取端口参数（14.8.2.1）	麦格特	端口 (15.4)
强制港口状态（14.8.2.2）	M-动作	端口 (15.4)
设置端口参数（14.8.2.3）	设定	端口 (15.4)

表 15-1 至表 15-6 中标识的管理操作集以及表 15-7 至表 15-38 中的映射定义了可在管理器和代理站之间协商的功能单元包，如 ISO/IEC 10040 的附件 A.3.2 中所述。

表 15-5 - GARP 参与者操作与 LMMS 服务的映射

管理运营	LMMS 服务元素	管理对象类
读取 GARP 计时器（14.9.1.1）	麦格特	GARP 计时器（15.12）
设置 GARP 计时器（14.9.1.2）	设定	GARP 计时器（15.12）
读取 GARP 申请人控制 (14.9.2.1)	麦格特	GARP 属性类型 (15.10)
设置 GARP 申请人控制 (14.9.2.2)	设定	GARP 属性类型 (15.10)
读取 GARP 状态（14.9.3.1）	麦格特	GARP 属性 (15.11)

表 15-6—选择性翻译表操作到 LMMS 服务的映射

管理运营	LMMS 服务元素	管理对象类
读取选择性翻译表条目 范围（参见 ISO/IEC 11802-5）	M-获取，M-取消获取	选择性翻译表条目（15.5）

本标准分配以下对象标识符值：

{iso(1) 成员机构(2) us(840) ieee802dot1D(10009) functionalUnitPackage(1)
bridgeManagementFunctionalUnit(1)}

作为 ISO/IEC 10040 中定义的 ASN.1 类型 FunctionalUnitPackage 的 functionalUnitPackageId 字段的值，用于协商此组管理操作的使用。

桥接功能单元包由三个功能单元组成，它们在 ASN.1 类型 FunctionalUnitPackage 的 manager-RoleFunctionalUnit 和 agentRoleFunctionalUnit 字段中标识：

位 0：标识支持（值 1）或不支持（值 0）定义的管理操作集
表 15-1 至表 15-5，以及它们到 LMMS 的映射，如表 15-7 至表 15-32 所述；

位 1：标识支持（值 1）或不支持（值 0）表中定义的管理操作
15-6，及其到LMMS的映射如表15-33所述；
位 2：标识支持（值 1）或不支持（值 0）表中定义的管理操作
15-5，以及它们到LMMS的映射，如表15-34至表15-38所述。

注：当使用表 15-7 至表 15-38 中描述的服务映射以及 ISO/IEC 15802-2 中定义的 LAN/MAN 管理协议时，管理操作可在单个网桥上执行，方法是使用其单独的 MAC 地址对网桥进行寻址，或使用组 MAC 地址对一组网桥进行寻址。特别是，表 7-5 中标识的所有 LAN 网桥管理组地址可用于将管理操作寻址到桥接 LAN 中的所有网桥。当在完整 OSI 堆栈上使用 ISO/IEC 9596-1 (CMIP) 时，无法使用组寻址。

表 15-7 至表 15-38 仅显示了本标准要求以特定方式分配值的 LMMS 服务参数。任何未明确标识的 LMMS 参数都是存在的、不存在的、可选的或有条件的，如 LMMS 服务原语本身的定义中所述。

15.2 管理对象包含结构

包含结构的顶点是 MAC 桥接 DLE（数据链路实体）管理对象；数据链路子系统中可能有零个或多个 MAC 桥接 DLE 管理对象。MAC 桥接 DLE 管理对象包含在数据链路子系统管理对象类的实例中，如 ISO/IEC 10742 中定义。

每个 MAC Bridge DLE 管理对象包含一个或多个端口管理对象。端口管理对象模拟 MAC Bridge 单个端口的可管理属性。这些条目在初始化时实例化，不会动态创建或删除。

表 15-7—Discover Bridge 参数映射（14.4.1.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	数据链路子系统（参见 ISO/IEC 10742）	—
基础对象实例	数据链路子系统名称 管理对象	—
范围	一级下属	—
筛选	管理对象类 = MAC Bridge DLE **和** Bridge Address 属性值在 包含范围 **和** 桥地址属性值不在 排除列表	—
属性标识符列表	读取或发现 Bridge 属性 组名（15.3.22）	—
管理对象类	—	MAC 桥接器
管理对象实例	—	MAC Bridge DLE 管理对象的名称
属性列表	—	所有属于以下成员的属性的名称/值 读取或发现桥属性组（15.3.22）

表 15-8—读取桥参数映射（14.4.1.2）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	MAC 桥接器 DLE (15.3)	—
基础对象实例	MAC 桥 DLE 的名称 管理对象	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取或发现 Bridge 属性组 名称（15.3.22）	—
管理对象类	—	MAC 桥接器 DLE (15.3)
管理对象实例	—	MAC Bridge DLE 管理对象的名称
属性列表	—	所有属于以下成员的属性的名称/值 读取或发现桥属性组（15.3.22）

表 15-9 — 设置桥名称参数的映射（14.4.1.3）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	MAC 桥接器 DLE (15.3)
基础对象实例	MAC Bridge DLE 管理对象的名称
范围	单独的基础对象
筛选	未使用
修改列表	桥梁名称属性（15.3.3）名称和所需替换值

每个端口管理对象中可能包含零个或多个选择性转换表条目管理对象。这些表条 目模拟了特定端口的选择性转换表中包含的信息，如 ISO/IEC 11802-5 的 5.2 中所述。这些条目可能根据远程管理请求动态创建或删除。

表 15-10—重置桥参数映射（14.4.1.4）

M-ACTION 参数名称	Req/Ind（操作输入）
基对象类	MAC 桥接器 DLE (15.3)
基础对象实例	MAC Bridge DLE 管理对象的名称
范围	单独的基础对象
筛选	不要求
动作类型	重置桥
行动信息	未使用
动作回复	未使用

表 15-11—读取端口参数映射（14.4.2.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取端口属性组名称 (15.4.26)	—
管理对象类	—	端口 (15.4)
管理对象实例	—	端口管理对象名称
属性列表	—	所有属于以下成员的属性的名称/值 读取端口属性组（15.4.26）

表 15-12—设置端口名称参数的映射（14.4.2.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	端口 (15.4)
基础对象实例	端口管理对象名称
范围	单独的基础对象
筛选	未使用
修改列表	端口名称属性（15.4.3）名称和所需的替换值

表 15-13—读取转发端口计数器参数的映射（14.6.1.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取转发端口计数器属性组名称 (15.4.27)	—
管理对象类	—	端口 (15.4)
管理对象实例	—	端口管理对象名称
属性列表	—	所有属于 Read For 成员的属性的名称/值 守卫端口计数器属性组（15.4.27）

表 15-14—读取端口默认用户优先级参数的映射（14.6.2.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	默认用户优先级属性名称 (15.4.12)	—
管理对象类	—	端口 (15.4)
管理对象实例	—	端口管理对象名称
属性列表	—	默认用户的名称/值 优先级属性（15.4.12）

表 15-15 — 设置端口默认用户优先级参数的映射（14.6.2.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	端口 (15.4)
基础对象实例	端口管理对象名称
范围	单独的基础对象
筛选	未使用
修改列表	默认用户优先级属性名称和所需替换值（15.4.12）

表 15-16—读取端口用户优先级再生表参数映射（14.6.2.3）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	用户优先级再生表属性名称（15.4.13）	—
管理对象类	—	港口
管理对象实例	—	端口管理对象名称
属性列表	—	用户优先级再生表属性的名称/值（15.4.13）

表 15-17—设置端口用户优先级再生表参数的映射（14.6.2.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	端口 (15.4)
基础对象实例	端口管理对象名称
范围	单独的基础对象
筛选	未使用
修改列表	用户优先级再生表属性名称和所需替换值（15.4.13）

每个端口管理对象中可以包含零个或一个 GARP 计时器管理对象。此对象模拟了给定端口上所有 GARP 应用程序实例（参见第 12 条）使用的计时器值。

表 15-18—读取端口流量类别表参数的映射（14.6.3.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	流量类别表属性名称（15.4.14）	—
管理对象类	—	端口 (15.4)
管理对象实例	—	端口管理对象名称
属性列表	—	流量类别表属性的名称/值（15.4.14）

表 15-19 — 设置流量类别表参数的映射（14.6.3.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	端口 (15.4)
基础对象实例	端口管理对象名称
范围	单独的基础对象
筛选	未使用
修改列表	流量类别表属性名称和所需替换值（15.4.14）

表 15-20—读取出站访问优先级表参数的映射（14.6.3.3）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	出站访问优先级表 属性名称（15.4.15）	—
管理对象类	—	端口 (15.4)
管理对象实例	—	端口管理对象名称
属性列表	—	出站访问的名称/值 优先级表属性（15.4.15）

如果相关端口支持 GARP，则 GARP 计时器管理对象存在；否则不存在。

每个端口管理对象中可以包含零个或多个 GARP 应用程序管理对象。每个 GARP 应用程序管理对象都对 GARP 应用程序的通用属性进行建模（参见第 12 条）。对于相关 GARP 应用程序支持的每个属性类型，每个 GARP 应用程序管理对象中都存在一个 GARP 属性类型管理对象。在每个 GARP 属性类型管理对象中，对于该应用程序当前维护给定 GIP 上下文的状态信息的属性类型的每个值，都存在一个 GARP 属性管理对象。GARP 属性管理对象根据相关 GARP 应用程序的操作进行创建、更新和删除。GARP 属性管理对象对可在单个 GARP 状态机上执行的操作进行建模。

表 15-21—读取过滤数据库参数的映射（14.7.1.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	过滤数据库 (15.7)	—
基础对象实例	过滤数据库的名称 管理对象	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取数据库属性组名称（15.6.5）	—
管理对象类	—	过滤数据库 (15.7)
管理对象实例	—	过滤数据库管理对象的名称
属性列表	—	所有成员属性的名称/值 读取数据库属性组 (15.6.5)，由过滤数据库管理对象扩展 类定义（15.7）

表 15-22—设置过滤数据库老化时间参数的映射（14.7.1.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	过滤数据库 (15.7)
基础对象实例	过滤数据库管理对象的名称
范围	单独的基础对象
筛选	未使用
修改列表	老化时间属性名称和所需替换值（15.7.3）

表 15-23—读取永久数据库参数的映射（14.7.5.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	永久数据库（15.6）	—
基础对象实例	永久数据库名称 管理对象	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取数据库属性组名称（15.6.5）	—
管理对象类	—	永久数据库（15.6）
管理对象实例	—	永久数据库管理对象的名称
属性列表	—	所有成员属性的名称/值 读取数据库属性组 (15.6.5)

表 15-24—创建过滤条目参数的映射（14.7.6.1）

M-CREATE 参数名称	Req/Ind（操作输入）
基对象类	数据库条目（15.8）
基础对象实例	新数据库条目管理对象的名称
上级对象实例	过滤数据库或永久数据库管理对象的名称
引用对象实例	未使用
属性列表	端口映射属性的名称和期望的初始值（15.8.3）

表 15-25—删除过滤条目参数的映射（14.7.6.2）

M-DELETE 参数名称	Req/Ind（操作输入）
基对象类	数据库条目（15.8）
基础对象实例	要删除的数据库条目管理对象的名称
范围	未使用
筛选	未使用

表 15-26—读取过滤条目参数映射（14.7.6.3）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	数据库条目（15.8）	—
基础对象实例	过滤数据库或永久数据库中的数据库条目管理对象的名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取数据库条目属性组名（15.8.6）	—
管理对象类	—	数据库条目（15.8）
管理对象实例	—	数据库条目管理对象的名称
属性列表	—	所有成员属性的名称/值 读取数据库条目属性组 (15.8.6)

表 15-27—读取过滤条目范围参数的映射（14.7.6.4）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	过滤数据库 (15.7) 或永久数据库 (15.6)	—
基础对象实例	过滤数据库或永久数据库的名称 数据库管理对象	—
范围	一级下属	—
筛选	条目索引属性值（15.8.5）在指定范围内	—
属性标识符列表	读取数据库条目属性组名称（15.8.6）	—
管理对象类	—	数据库条目
管理对象实例	—	数据库条目管理对象的名称
属性列表	—	所有属于以下成员的属性的名称/值 读取数据库条目属性组 (15.8.6)

每个 MAC Bridge DLE 管理对象内都包含一个过滤数据库管理对象和一个永久数据库管理对象。这些数据库管理对象模拟了每个数据库的共同属性，例如大小、条目数和老化时间。它们还构成了过滤和永久数据库中过滤条目管理对象的容器对象。

每个数据库管理对象中可以包含零个或多个数据库条目管理对象。每个数据库条目管理对象在过滤或永久数据库中模拟单个条目（动态或静态）。静态条目（静态过滤条目）可以通过远程管理请求创建和删除；动态条目由学习过程（动态过滤条目）的操作或 GMRP（组注册条目）的操作创建，并可以通过远程管理请求删除。

表 15-28—读取桥接协议参数的映射（14.8.1.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	MAC 桥接器 DLE (15.3)	—
基础对象实例	MAC 桥 DLE 的名称 管理对象	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取桥参数 属性组名称（15.3.23）	—
管理对象类	—	MAC 桥接器 DLE (15.3)
管理对象实例	—	MAC Bridge DLE 管理对象的名称
属性列表	—	属于读取的所有属性的名称/值 桥接协议参数属性组（15.3.23）

表 15-29—设置桥接协议参数的映射（14.8.1.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	MAC 桥接器 DLE (15.3)
基础对象实例	MAC Bridge DLE 管理对象的名称
范围	单独的基础对象
筛选	未使用
修改列表	桥接最大年龄 (15.3.17)、桥接问候时间 (15.3.18)、桥接转发延迟 (15.3.19) 和 桥梁优先级（15.3.21）属性名称和所需替换值

表 15-30—读取端口参数映射（14.8.2.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	读取端口参数属性 组名（15.4.28）	—
管理对象类	—	端口 (15.4)
管理对象实例	—	端口管理对象名称
属性列表	—	读取端口的所有属性的名称/值 参数属性组（15.4.28）

图 15-1 显示了 MAC 桥接器的管理对象包含结构；该图中用虚线标记的元素显示了 ISO/IEC 10742 中定义的数据链路层包含结构。

表 15-31—强制端口状态参数映射（14.8.2.2）

M-ACTION 参数名称	Req/Ind（操作输入）
基对象类	端口 (15.4)
基础对象实例	端口管理对象名称
范围	单独的基础对象
筛选	未使用
动作类型	强制港口状态 (15.4.29)
行动信息	所需状态（禁用或阻止）
动作回复	未使用

表 15-32—设置端口参数的映射（14.8.2.3）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	端口 (15.4)
基础对象实例	端口管理对象名称
范围	单独的基础对象
筛选	未使用
修改列表	路径成本（15.4.20）和端口优先级（15.4.19）属性名称和所需的替换值

表 15-33—读取选择性转换表条目范围参数的映射（ISO/IEC 11802-5）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	端口 (15.4)	—
基础对象实例	端口管理对象名称	—
范围	一级下属	—
筛选	类型 值 属性值 (15.5.2) 在指定范围内	—
属性标识符列表	空的	—
管理对象类	—	选择性翻译表条目（15.5）
管理对象实例	—	选择性翻译表条目管理对象的名称

表 15-34 — 读取 GARP 计时器参数的映射（14.9.1.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	GARP 计时器（15.12）	—
基础对象实例	GARP 定时器的名称 管理对象	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	入职时间 (15.12.3)、离职时间 (15.12.4)、和“一直保留”(15.12.5) 属性	—
管理对象类	—	GARP 计时器（15.12）
管理对象实例	—	GARP 定时器的名称 管理对象
属性列表	—	加入时间 (15.12.3)、离开时间 (15.12.4) 和全部离开的名称/值 时间（15.12.5）属性

表 15-35 — 设置 GARP 计时器参数的映射（14.9.1.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	GARP 计时器（15.12）
基础对象实例	GARP 定时器管理对象的名称
范围	单独的基础对象
筛选	未使用
修改列表	加入时间（15.12.3）、离开时间的名称/替换值（15.12.4）和“一直离开”（15.12.5）属性

表 15-36 — 读取 GARP 申请人控制参数的映射（14.9.2.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	GARP 属性类型 (15.10)	—
基础对象实例	GARP 属性的名称管理对象	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	名称申请人行政控制（15.10.3）和失败注册 (15.10.4) 属性	—
管理对象类	—	GARP 属性类型 (15.10)
管理对象实例	—	GARP 属性的名称管理对象
属性列表	—	名称/值协议管理控制 (15.10.3) 和失败注册 (15.10.4) 属性

表 15-37 — 设置 GARP 申请人控制参数的映射（14.9.2.2）

M-SET 参数名称	Req/Ind（操作输入）
基对象类	GARP 属性类型 (15.10)
基础对象实例	GARP 属性管理对象的名称
范围	单独的基础对象
筛选	未使用
修改列表	申请人管理控制 (15.10.3) 属性的名称/替换值

表 15-38 — 读取 GARP 状态参数的映射（14.9.3.1）

M-GET 参数名称	Req/Ind（操作输入）	Rsp/Conf（操作输出）
基对象类	GARP 属性 (15.11)	—
基础对象实例	GARP 属性的名称 管理对象	—
范围	单独的基础对象	—
筛选	不要求	—
属性标识符列表	状态值的名称 (15.11.3) 和（可选） 最后一个 PDU 的发起者 (15.11.4) 属性	—
管理对象类	—	GARP 属性 (15.11)
管理对象实例	—	GARP 属性的名称 管理对象
属性列表	—	状态值的名称/值（15.11.3）和（可选）最后一个 PDU 的发起者（15.11.4） 属性

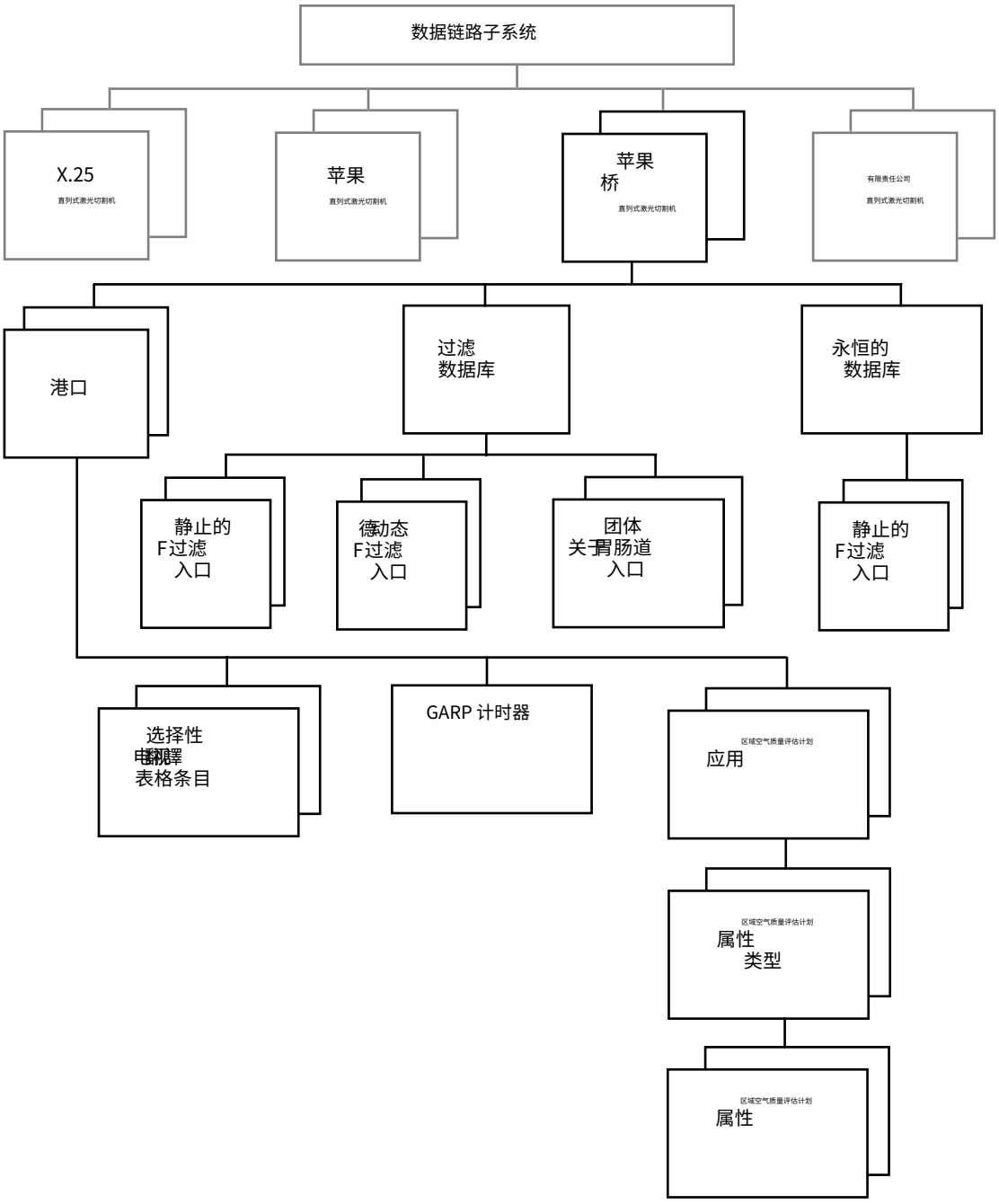


图 15-1—管理对象包含结构

15.3 MAC桥DLE管理对象类定义

```
oMACBridgeDLE 管理对象类
   源自“ISO/IEC 10742”：datalinkEntity；特点是

      pMACBridgeDLE 软件包
         行为
            桥接MAC地址        行为
               定义为                  !MAC Bridge DLE 管理对象类结合了 Bridge Configuration 对象和 Protocol
                                       Entity 对象的特征，定义如下：

                                       14.4.1 和 14.8.1.!
            ;
         ;
         属性                    桥地址                                  GET, -- 命名属性 GET-REPLACE,
                                 桥梁名称
                                 桥接端口号                              得到，
                                 桥接端口地址                            得到，
                                 正常运行时间                            得到，
                                 桥梁标识符                              得到，
                                 自拓扑变化以来的时间                    得到，
                                                                         得到，
                                 拓扑变化                                得到，
                                 指定根                                  得到，
                                 根路径成本                              得到，
                                 根端口                                  得到，
                                 最大年龄                                得到，
                                 一个HelloTime                           得到，
                                 转发延迟                                得到，
                                 桥最大年龄                              获取-替换，
                                 桥接HelloTime                           获取-替换，
                                 桥接转发延迟                            获取-替换，
                                 保持时间                                得到，
                                 桥接优先级                              获取替换
         ;
         属性组                  agReadOrDiscoverBridge，
                                 agReadBridgeProtocolParameters
         ;
         行动                    ac重置桥；
      ;
   ;
注册为 {Bridge.moClass macBridgeDLE(0)};
```

15.3.1 MAC Bridge DLE 的名称绑定

```
nbMACBridgeDLE-数据链路子系统名称绑定
   从属对象类                                    oMACBridgeDLE 和子类；
   由高级对象类命名                              “ISO/IEC 10742”:datalink子系统和子类;

   带属性                                        桥地址；
注册为 {Bridge.nameBinding macBridgeDLE-datalinkSubsystem(0)};
```

15.3.2 桥地址属性定义

```
aBridgeAddress 属性
   WITH 属性语法           桥.桥地址；
   匹配                    平等、秩序；
   行为
      桥地址          行为
         定义为            !参见 14.4.1.1.3 a)、14.4.1.2.3 a)、8.5.3.7、7.12.5.!
      ;
   ;
注册为 {Bridge.attribute bridgeAddress(0)};
```

15.3.3 桥名称属性定义

```
aBridgeName 属性
   WITH 属性语法           桥梁.桥梁名称；
```

```
   匹配                                平等；
   行为
      桥接名称          行为
         定义为                        !参见 14.4.1.1.3 b)、14.4.1.2.3 b)。!
      ;
   ;
注册为 {Bridge.attribute bridgeName(1)};
```

15.3.4 桥接端口数属性定义

```
桥接端口号              属性
   带属性                句法          桥接端口号                    ;
   匹配                                平等、秩序;
   行为
      桥接端口号              行为
         定义为                        !参见 14.4.1.1.3 c)、14.4.1.2.3 c)。、14.4.2.!
      ;
   ;
注册为 {Bridge.attribute bridgeNumPorts(2)};
```

15.3.5 桥接端口地址属性定义

```
aBridgePortAddresses 属性
   WITH 属性语法                       桥接.桥接端口地址相等;                ;
   匹配
   行为
      桥接端口地址                   行为
         定义为                        !参见 14.4.1.1.3 d)、14.4.1.2.3 d)。!
      ;
   ;
注册为 {Bridge.attribute bridgePortAddresses(3)};
```

15.3.6 正常运行时间属性定义

```
aUptime 属性
   衍生的      从                      “ISO/IEC    10165-5”:nonWrapping64BitCounter          ;
   匹配        为了                    平等，          订购;
   行为
      正常运行时间  行为
         定义为                        !参见 14.4.1.1.3 e)、14.4.1.2.3 e)。!
      ;
   ;
注册为 {Bridge.attribute uptime(4)};
```

15.3.7 桥接标识符属性定义

```
aBridgeIdentifier 属性
   WITH 属性语法                       桥梁.桥梁标识符相等;                ;
   匹配
   行为
      桥接标识符              行为
         定义为                        见 8.5.3.7.!
      ;
   ;
注册为 {Bridge.attribute bridgeIdentifier(5)};
```

15.3.8 拓扑改变以来的时间属性定义

```
自拓扑变化以来的时间            属性
   衍生的      从                      “ISO/IEC    10165-5”:nonWrapping64BitCounter          ;
   匹配        为了                    平等，          订购;
   行为
      自拓扑变化以来的时间          行为
         定义为                        ！看    14.8.1.1.3        b).!
      ;
   ;
注册为 {Bridge.attribute timeSinceTopologyChange(6)};
```

15.3.9 拓扑变化计数属性定义

```
拓扑变化计数                    属性
   衍生的     从                    “ISO/IEC    10165-5”:nonWrapping64BitCounter          ;
   匹配       为了                  平等，          订购;
   行为
      拓扑变化计数                 行为
         定义为                    参见 14.8.1.1.3          !
      ;
   ;
注册为 {Bridge.attribute topologyChangeCount(7)};
```

15.3.10 拓扑变化属性定义

```
拓扑变化              属性
   带属性              句法        桥梁.拓扑变化                    ;
   匹配                            平等;
   行为
      拓扑变化            行为
         定义为                    !见 8.5.3.12.!
      ;
   ;
注册为 {Bridge.attribute topologyChange(8)};
```

15.3.11 指定根属性定义

```
指定根                属性
   带属性              句法        桥梁.桥梁标识符相等;              ;
   匹配
   行为
      指定根              行为
         定义为                    !见 8.5.3.1.!
      ;
   ;
注册为 {Bridge.attribute specifiedRoot(9)};
```

15.3.12 根路径代价属性定义

```
aRootPathCost 属性
   带属性              句法        桥.根路径成本                ;
   匹配                            平等、秩序                ;
   行为
      根路径成本          行为
         定义为                    见 8.5.3.2.!
      ;
   ;
注册为 {Bridge.attribute rootPathCost(10)};
```

15.3.13 根端口属性定义

```
aRootPort 属性
   带属性              句法        桥接根端口              ;
   匹配                            平等;
   行为
      根端口       行为
         定义      作为            见 8.5.3.3.!
      ;
   ;
注册为 {Bridge.attribute rootPort(11)};
```

15.3.14 最大年龄属性定义

```
aMaxAge 属性
   WITH 属性语法                   桥梁.最大年龄;
   匹配                            平等、秩序                ;
   行为
```

```
      bMaxAge 行为
         定义为                    见 8.5.3.4.!
      ;
   ;
注册为 {Bridge.attribute maxAge(12)};
```

15.3.15 Hello Time属性定义

```
aHelloTime 属性
   WITH 属性语法                   Bridge.HelloTime      ;
   匹配                            平等、秩序             ;
   行为
      bHelloTime 行为
         定义为                    见 8.5.3.5.!
      ;
   ;
注册为 {Bridge.attribute helloTime(13)};
```

15.3.16 转发延迟属性定义

```
转发延迟属性
   带属性              句法        桥接转发延迟              ;
   匹配                            平等、秩序             ;
   行为
      转发延迟          行为
         定义为                    见 8.5.3.6.!
      ;
   ;
注册为 {Bridge.attribute forwardDelay(14)};
```

15.3.17 桥梁最大年龄属性定义

```
aBridgeMaxAge 属性
   带属性              句法        Bridge.BridgeMaxAge       ;
   匹配                            平等、秩序             ;
   行为
      桥最大年龄        行为
         定义为                    见 8.5.3.8.!
      ;
   ;
注册为 {Bridge.attribute bridgeMaxAge(15)};
```

15.3.18 桥接Hello Time属性定义

```
桥接HelloTime        属性
   带属性              句法        Bridge.BridgeHelloTime        ;
   匹配                            平等、秩序;
   行为
      桥接HelloTime        行为
         定义为                    见 8.5.3.9.!
      ;
   ;
注册为 {Bridge.attribute bridgeHelloTime(16)};
```

15.3.19 桥接转发延迟属性定义

```
aBridgeForwardDelay 属性
   WITH 属性语法                   Bridge.BridgeForwardDelay 平等，   ;
   匹配                            排序;
   行为
      桥接转发延迟              行为
         定义为                    !见 8.5.3.10.!
      ;
   ;
注册为 {Bridge.attribute bridgeForwardDelay(17)};
```

15.3.20 保持时间属性定义

```
aHoldTime 属性
   带属性          句法      桥接.保持时间;
   匹配                      平等、秩序          ;
   行为
      保持时间    行为
         定义     作为       !见 8.5.3.14.!
      ;
   ;
注册为 {Bridge.attribute holdTime(18)};
```

15.3.21 桥优先级属性定义

```
桥接优先级          属性
   带属性          句法      桥梁.桥梁优先级          ;
   匹配                      平等、秩序;
   行为
      桥接优先级          行为
         定义为             见 8.5.3.7.!
      ;
   ;
注册为 {Bridge.attribute bridgePriority(19)};
```

15.3.22 读取或发现桥接属性组定义

```
读取或发现桥              属性组
   团体     元素          桥梁名称，
                          aBridgeNumPorts，
                          桥接端口地址，
                          正常运行时间
   ;
   固定的 ;
   描述                   !此属性组用于读取桥接器和发现桥接器操作的映射，如表 15-7 和表 15-8 所定
                          义。!;

注册为 {Bridge.attributeGroup readOrDiscoverBridge(0)};
```

15.3.23 读取桥接协议参数属性组定义

```
读取桥接协议参数                  属性组
   团体     元素          桥梁标识符，
                          自拓扑变化以来的时间、拓扑变化计
                          数、
                          拓扑变化，
                          指定根，
                          根路径成本，
                          根端口，
                          最大年龄，
                          你好，
                          转发延迟，
                          aBridgeMaxAge，
                          aBridgeHelloTime，
                          桥接转发延迟，
                          保持时间
   ;
   固定的 ;
   描述                   !此属性组用于读取桥接协议参数操作的映射，如表 15-28 所定义。!;


注册为 {Bridge.attributeGroup readBridgeProtocolParameters(1)};
```

15.3.24 重置桥接动作定义

```
acResetBridge 动作
   行为
      bResetBridge 行为
```

```
        定义为                          !见 14.4.1.4.!
    ;
  ;
注册为 {Bridge.action resetBridge(0)};
```

15.4 端口管理对象类定义

```
oPort 管理对象类
  源自                                  “ISO/IEC   10165-2”:顶部;
  特点是
    端口包
      行为
        bPort 行为
          定义为                        !端口管理对象类结合了端口配置对象、端口计数器对象、传输优先级对象和桥
                                        接端口对象的特性，定义如下：


                                        14.4.2、14.6.1、14.6.2 和 14.8.2。!
        ;
      ;
      属性
                                        端口号                                  GET, -- 命名属性 GET-REPLACE,
                                        端口名称
                                        端口类型                                得到，
                                        已接收帧                                得到，
                                        丢弃入站                                得到，
                                        aForwardOutbound                        得到，
                                        丢弃缺少缓冲区                          得到，
                                        丢弃传输延迟超出范围
                                                                                得到，
                                        错误丢弃                                得到，
                                        丢弃错误详细信息                        得到，
                                        默认用户优先级                          获取-替换，
                                        用户优先级再生表
                                                                                获取-替换，
                                        流量类别表                              获取-替换，
                                        出站访问优先级表
                                                                                得到，
                                        端口正常运行时间                        得到，
                                        一个国家                                得到，
                                        端口标识符                              得到，
                                        端口优先级                              获取-替换，
                                        路径成本                                获取-替换，
                                        指定根端口                              得到，
                                        指定成本                                得到，
                                        指定桥梁                                得到，
                                        指定端口                                得到，
                                        拓扑改变确认                            得到，
                                        “ISO/IEC 10742”:提供者实体名称
                                                                                得到
      ;
      属性组                            agReadPort，
                                        agReadForwardingPortCounters，
                                        agReadPortParameters
      ;
      行动                              acForcePortState;
    ;
  ;
注册为 {Bridge.moClass port(6)};
```

15.4.1 端口的名称绑定

```
nbPort-MACBridgeDLE       名称绑定
  从属对象                    班级                        oPort 和子类;
  由具有属性的上级对象类命名                              oMACBridgeDLE 和子类;
                                                          aPortNumber;
注册为 {Bridge.nameBinding port-MACBridgeDLE(1)};
```

15.4.2 端口号属性定义

```
aPortNumber 属性
   WITH 属性语法                    桥接端口号                ;
   匹配                             平等、秩序                ;
   行为
      bPortNumber 行为
         定义为                     !参见 8.5.5.1 和 14.4.2。!
      ;
   ;
注册为 {Bridge.attribute portNumber(20)};
```

15.4.3 端口名称属性定义

```
aPortName 属性
   带属性              句法        桥接端口名称          ;
   匹配                            平等;
   行为
      端口名称      行为
         定义       作为          !参见 14.4.2.1.3      一个）!
      ;
   ;
注册为 {Bridge.attribute portName(21)};
```

15.4.4 端口类型属性定义

```
aPortType 属性
   WITH 属性语法                    桥接.端口类型;
   匹配                             平等、秩序                ;
   行为
      bPortType 行为
         定义为                     !参见 14.4.2.1.3 b)。          此属性包含一个对象
                                    指示 MAC 实体类型的标识符值。值可能是用于标识定义 MAC 实体类型的标准的注册
                                    值，也可能是独立于 MAC 标准注册的值，用于标识特定的 MAC 实体类型。!
      ;
   ;
注册为 {Bridge.attribute portType(22)};
```

15.4.5 接收帧属性定义

```
已接收帧            属性
   衍生的    从                    “ISO/IEC   10165-5”:nonWrapping64BitCounter      ;
   匹配      为了                  平等，      订购;
   行为
      已接收帧数          行为
         定义为                     !参见 14.6.1.1.3      一个）!
      ;
   ;
注册为 {Bridge.attribute framesReceived(23)};
```

15.4.6 丢弃入站属性定义

```
丢弃入站            属性
   衍生的    从                    “ISO/IEC   10165-5”:nonWrapping64BitCounter      ;
   匹配      为了                  平等，      订购;
   行为
      丢弃入站          行为
         定义为                     !参见 14.6.1.1.3      b).!
      ;
   ;
注册为 {Bridge.attribute discardInbound(24)};
```

15.4.7 转发出站属性定义

```
aForwardOutbound     属性
   衍生的      从                    “ISO/IEC    10165-5”:nonWrapping64BitCounter          ;
   匹配        为了                  平等，          订购;
   行为
      转发出站                 行为
         定义为                    !参见 14.6.1.1.3          !
      ;
   ;
注册为 {Bridge.attribute forwardOutbound(25)};
```

15.4.8 丢弃缺少缓冲区属性定义

```
丢弃缺少缓冲区                  属性
   衍生的      从                    “ISO/IEC    10165-5”:nonWrapping64BitCounter          ;
   匹配        为了                  平等，          订购;
   行为
      丢弃缺少缓冲区              行为
         定义为                    !参见 14.6.1.1.3          d)!
      ;
   ;
注册为 {Bridge.attribute discardLackOfBuffers(26)};
```

15.4.9 丢弃传输延迟超出属性定义

```
丢弃传输延迟超出范围                    属性
   衍生的      从                    “ISO/IEC    10165-5”:nonWrapping64BitCounter          ;
   匹配        为了                  平等，          订购;
   行为
      丢弃传输延迟超出                        行为
         定义为                    !参见 14.6.1.1.3          e）！
      ;
   ;
注册为 {Bridge.attribute discardTransitDelayExceeded(27)};
```

15.4.10 错误时丢弃属性定义

```
错误丢弃              属性
   衍生的      从                    “ISO/IEC    10165-5”:nonWrapping64BitCounter          ;
   匹配        为了                  平等，          订购;
   行为
      错误时丢弃           行为
         定义为                    !参见 14.6.1.1.3          !
      ;
   ;
注册为 {Bridge.attribute discardOnError(28)};
```

15.4.11 错误时丢弃详细属性定义

```
aDiscardOnErrorDetail 属性
   WITH 属性语法               Bridge.DiscardOnErrorDetail 平等，       ;
   匹配                        排序;
   行为
      错误详细信息             行为
         定义为                !参见 14.6.1.1.3 g)。!
      ;
   ;
注册为 {Bridge.attribute discardOnErrorDetail(29)};
```

15.4.12 默认用户优先级属性定义

```
aDefaultUserPriority 属性
   WITH 属性语法               Bridge.用户优先级          ;
   匹配                        平等、秩序                 ;
   行为
```

b默认用户优先级 行为
定义为 !参见 14.6.2.1.3 a) 和 14.6.2.2.2 b)。!
;
;
注册为 {Bridge.attribute defaultUserPriority(52)};

15.4.13 用户优先级再生表属性定义

用户优先级再生表 属性
WITH 属性语法 Bridge.UserPriorityRegenerationTable 平等 ;
匹配 ;
行为
b用户优先级再生表 行为
定义为 !参见 14.6.2.3.3 和 14.6.2.4.2.!
;
;
注册为 {Bridge.attribute userPriorityRegenerationTable(53)};

15.4.14 流量类别表属性定义

aTrafficClassTable 属性
WITH 属性语法 桥接.交通类别表平等; ;
匹配
行为
流量类别表 行为
定义为 !参见 14.6.3.1.3 a) 和 14.6.3.2.2 b)。!
;
;
注册为 {Bridge.attribute TrafficClassTable(54)};

15.4.15 出站访问优先级表属性定义

出站访问优先级表 属性
WITH 属性语法 Bridge.OutboundAccessPriorityTable 平等，排 ;
匹配 序;
行为
出站访问优先级 行为
定义为 ！看 14.6.3.3.3.!
;
;
注册为 {Bridge.attribute outboundAccessPriorityTable(55)};

15.4.16 端口正常运行时间属性定义

aPortUptime 属性
WITH 属性语法 桥梁.正常运行时间;
匹配 平等、秩序 ;
行为
bPortUptime 行为
定义为 !参见 14.8.2.1.3 a)。!
;
;
注册为 {Bridge.attribute portUptime(32)};

15.4.17 状态属性定义

一个国家 属性
和 属性 句法 桥梁.状态;
匹配 为了 平等、秩序 ;
行为
bState 行为
定义为 !参见 14.8.2.1.3 b) 和 8.5.5.2.!
;
;
注册为 {Bridge.attribute state(33)};

15.4.18 端口标识符属性定义

```
端口标识符              属性
   带属性               句法        桥接端口标识符                ;
   匹配                             平等;
   行为
      端口标识符            行为
         定义为                     !参见 14.8.2.1.3 c) 和 8.5.5.1.!
      ;
   ;
注册为 {Bridge.attribute portIdentifier(34)};
```

15.4.19 端口优先级属性定义

```
aPortPriority 属性
   带属性               句法        桥接端口优先级              ;
   匹配                             平等、秩序              ;
   行为
      端口优先级          行为
         定义为                     !参见 14.8.2.3.2 c)。!
      ;
   ;
注册为 {Bridge.attribute portPriority(35)};
```

15.4.20 路径成本属性定义

```
aPathCost 属性
   带属性               句法        桥梁.路径成本;
   匹配                             平等、秩序              ;
   行为
      路径成本      行为
         定义       作为            见 8.5.5.3.!
      ;
   ;
注册为 {Bridge.attribute pathCost(36)};
```

15.4.21 端口指定根属性定义

```
aPortDesignatedRoot 属性
   WITH 属性语法                    桥梁.指定根                ;
   匹配                             平等;
   行为
      指定端口根                行为
         定义为                     见 8.5.5.4.!
      ;
   ;
注册为 {Bridge.attribute portDesignatedRoot(37)};
```

15.4.22 指定成本属性定义

```
指定成本                属性
   带属性               句法        桥梁指定成本                ;
   匹配                             平等、秩序;
   行为
      指定成本              行为
         定义为                     见 8.5.5.5.!
      ;
   ;
注册为 {Bridge.attribute specifiedCost(38)};
```

15.4.23 指定桥属性定义

```
aDesignatedBridge 属性
   WITH 属性语法                    桥梁.指定桥梁平等;                ;
   匹配
   行为
```

```
      指定桥                    行为
        定义为                  见 8.5.5.6.!
      ;
   ;
注册为 {Bridge.attribute specifiedBridge(39)};
```

15.4.24 指定端口属性定义

```
指定端口              属性
   带属性              句法        桥接指定端口              ;
   匹配                            平等;
   行为
      指定端口            行为
        定义为                  见 8.5.5.7.!
      ;
   ;
注册为 {Bridge.attribute specifiedPort(40)};
```

15.4.25 拓扑ChangeAck属性定义

```
aTopologyChangeAck 属性
   WITH 属性语法               Bridge.TopologyChangeAck 平      ;
   匹配                        等，排序;
   行为
      拓扑改变确认          行为
        定义为                  见 8.5.5.8.!
      ;
   ;
注册为 {Bridge.attribute topologyChangeAck(41)};
```

15.4.26 读取端口属性组定义

```
读取端口      属性组
   团体    元素                 端口名称，
                                端口类型
   ;
   固定的 ;
   描述                         !该属性组用于读取端口操作的映射，如表 15-11 所定义。!;

注册为 {Bridge.attributeGroup readPort(2)};
```

15.4.27 读取转发端口计数器属性组定义

```
读取转发端口计数器                  属性组
   团体    元素                 已接收帧，
                                aDiscardInbound，
                                aForwardOutbound，
                                丢弃缺少的缓冲区，
                                丢弃传输延迟超出，丢弃错误，

                                丢弃错误详细信息
   ;
   固定的 ;
   描述                         !此属性组用于读取转发端口计数器操作的映射，定义在

                                表 15-13.!;
注册为 {Bridge.attributeGroup readForwardingPortCounters(3)};
```

15.4.28 读取端口参数属性组定义

```
读取端口参数              属性组
   集团元素                 aPortUptime，
                            一个州，
                            端口标识符，
                            路径成本，
                            指定根端口，
                            指定成本，
```

```
                                    指定桥梁，
                                    指定端口，
                                    拓扑改变确认
  ;
  固定的 ;
  描述                              !此属性组用于读取端口参数操作的映射，如表 15-30 所定义。!;

注册为 {Bridge.attributeGroup readPortParameters(5)};
```

15.4.29 强制端口状态行动定义

```
acForcePortState 动作
  行为
    强制端口状态          行为
      定义为                        !模拟强制港口国家操作，如 14.8.2.2 所述。!

    ;
  ;
  提供信息              句法                        桥接.强制端口状态;
注册为 {Bridge.action forcePortState(1)};
```

15.5 选择性翻译表条目管理对象类定义

```
oSelectiveTranslationTableEntry 管理对象类
  源自                              “ISO/IEC   10165-2”:顶部;
  特征                  经过
    选择性翻译表条目                        包裹
      行为
        选择性翻译表条目                        行为
          定义为                !选择性翻译表条目管理对象类对选择性翻译表中单个条目的属性进行建模，
                                如 ISO/IEC 11802-5 的 5.2 中所述!


        ;
      ;
      属性                      类型值              GET——命名属性
      ;
    ;
  ;
注册为 {Bridge.moClass selectedTranslationTableEntry(2)};
```

15.5.1 选择性翻译表条目的名称绑定

```
nbSelectiveTranslationTableEntry-Port          名称绑定
  从属对象类                                    oSelectiveTranslationTableEntry 和子类;

  由具有属性的上级对象类命名                    oPort 和子类;
                                                类型值;
  创造;
  删除;
注册为 {Bridge.nameBinding selectedTranslationTableEntry-Port(2)};
```

15.5.2 类型值属性定义

```
aTypeValue 属性
  WITH 属性语法                 Bridge.Type值          ;
  平等比赛；行为

    bTypeValue 行为
      定义为                    !表示已注册以太网协议类型的值的整数值。类型值作为整数的编码以及在以太网类
                                型字段中编码时对应的位/八位字节传输顺序如 ISO/IEC 11802-5 中定义。!



  ;
注册为 {Bridge.attribute typeValue(42)};
```

15.6 永久数据库管理对象类定义

```
oPermanentDatabase 管理对象类
    源自                                “ISO/IEC    10165-2”：顶部    ;
    特征              经过
        永久数据库                包裹
            行为
                永久数据库                 行为
                    定义为
                                    !永久数据库管理对象类的一个实例是组成永久数据库的数据库条目管理对象的容
                                    器管理对象。永久数据库的功能如 14.7.5 中所述；永久数据库管理对象仅对支持
                                    读取永久数据库操作所必需的永久数据库方面进行建模，如 14.7.5.1 中所述。!
                ;
            ;
            属性                    aDatabaseName 初始值 Bridge.permanentDatabaseName
                                                                得到，  - - 命名属性
                                    数据库大小                   得到，
                                    条目数                       得到
            ;
            属性组                  读取数据库
            ;
        ;
    ;
注册为 {Bridge.moClass permanentDatabase(3)};
```

15.6.1 永久数据库的名称绑定

```
nbPermanentDatabase-MACBridgeDLE 名称绑定
    从属对象类                                  o永久数据库及其子类;
    由高级对象类命名                            oMACBridgeDLE 和子类;
    带属性                                      数据库名称;
注册为 {Bridge.nameBinding permanentDatabase-MACBridgeDLE(3)};
```

15.6.2 数据库名称属性定义

```
数据库名称属性
    带属性            句法          桥接.数据库名称              ;
    匹配                            平等、秩序                 ;
    行为
        数据库名称        行为
            定义为                  !数据库的命名属性!
        ;
    ;
注册为 {Bridge.attribute databaseName(43)};
```

15.6.3 数据库大小属性定义

```
数据库大小属性
    带属性            句法          Bridge.数据库大小            ;
    匹配                            平等、秩序                 ;
    行为
        数据库大小        行为
            定义为                  !参见 14.7.1.1.3 a) 和 14.7.5.1.3 a)。!
        ;
    ;
注册为 {Bridge.attribute databaseSize(44)};
```

15.6.4 条目数属性定义

```
条目数            属性
    带属性            句法          桥接条目数                      ;
    匹配                            平等、秩序;
```

```
   行为
      条目数                    行为
         定义为                       !参见 14.7.1.1.3 a) 和 14.7.5.1.3 a)。!
      ;
   ;
注册为 {Bridge.attribute numberOfEntries(45)};
```

15.6.5 读取数据库属性组定义

```
agReadDatabase 属性              团体
   集团元素                      数据库大小，
                                 条目数
   ;
   描述                          !此属性组用于读取过滤数据库和读取永久数据库操作的映射，如表 15-21 和


                                 表 15-23.!;
注册为 {Bridge.attributeGroup readDatabase(6)};
```

15.7 过滤数据库管理对象类定义

```
oFilteringDatabase 管理对象类
   源自                          “ISO/IEC   10165-2”：顶部   ;
   特征                 经过
      过滤数据库                 包裹
         行为
            过滤数据库           行为
               定义为            !过滤数据库管理对象类的一个实例是组成过滤数据库的数据库条目管理对象的容器管
                                 理对象。过滤数据库的功能如 14.7.1 中所述；过滤数据库管理对象仅对过滤数据库的
                                 那些方面进行建模，这些方面对于支持读取过滤数据库和设置过滤数据库老化时间是
                                 必要的




                                 操作，如 14.7.1.1 和 14.7.1.2 所述。!
            ;
         ;
         属性                    aDatabaseName 初始值          Bridge.filteringDatabaseName 获取，
                                                                  - - 命名属性
                                 数据库大小                    得到，
                                 条目数                        得到，
                                 动态条目数老化时间            得到，
                                                               获取替换
         ;
         属性组                  读取数据库                    动态条目数老化时间

                                                               - - 添加到组定义
         ;
      ;
   ;
   有条件套餐
      韓國支持           包裹
         属性                    组注册条目数                        得到
         ;
         属性组                  读取数据库
                                          组注册条目数
                                          - - 添加到组定义
         ;
      注册为 {Bridge.package fdGroupSupport (0)};
   若此端口支持扩展过滤服务（6.6）则显示！；已注册为{Bridge.moClassfilteringDatabase(7)};
```

15.7.1 用于过滤数据库的名称绑定

```
nbFilteringDatabase-MACBridgeDLE 名称绑定
   从属对象类                                oFilteringDatabase 和子类;
   由高级对象类命名                          oMACBridgeDLE 和子类;
```

带属性 数据库名称；
注册为 {Bridge.nameBinding filterDatabase-MACBridgeDLE(4)}；

15.7.2 动态条目数属性定义

动态条目数 属性
WITH 属性语法 Bridge.NumberOfDynamicEntries 相 ；
匹配 等，排序；
行为
动态条目数 行为
定义为 ！看 14.7.1.1.3c).！
；
；
注册为 {Bridge.attribute numberOfDynamicEntries(46)}；

15.7.3 老化时间属性定义

老化时间属性
WITH 属性语法 桥接老化时间 ；
匹配 平等、秩序 ；
行为
bAgeingTime 行为
定义为 !参见 14.7.1.1.3 d) 和 7.9.2。！
；
；
注册为 {Bridge.attribute ageingTime(47)}；

15.7.4 组注册表项数属性定义

组注册条目数 属性
WITH 属性语法 Bridge.NumberOfGroupRegistrationEntries 相等，排 ；
匹配 序；
行为
b组注册条目数 行为
定义为 !参见 14.7.1.1.3 !
；
；
注册为 {Bridge.attribute numberOfGroupRegistrationEntries(56)}；

15.8 数据库条目管理对象类定义

oDatabaseEntry 管理对象类
源自 “ISO/IEC 10165-2”：顶部 ；
特征 经过
数据库入口点 包裹
行为
数据库入口 行为
定义为 !数据库条目管理对象类为过滤和永久数据库中的动态条目（动态过滤条目和组注册条目）和静态条目（静态过滤条目）建模，如 14.7.1 和 14.7.5 中所述。数据库条目管理对象类提供 14.7.6 中所述的功能。！

；
；
属性 地址 GET, -- 命名属性 GET-REPLACE,
端口映射
条目类型 得到，
条目索引 得到
；
属性组 读取数据库条目
；
；
；
注册为 {Bridge.moClass databaseEntry(5)}；

15.8.1 数据库条目的名称绑定

nbPermanentEntry-永久数据库名称绑定
从属对象类 oDatabaseEntry 和子类；
由高级对象类命名 o永久数据库及其子类；
带属性 地址；
创造；
删除；
注册为 {Bridge.nameBinding permanentEntry-PermanentDatabase(5)}；

nbStaticEntry-FilteringDatabase 名称绑定
从属对象类 oDatabaseEntry 和子类；
由高级对象类命名 oFilteringDatabase 和子类；
带属性 地址；
创造；
删除；
注册为 {Bridge.nameBinding staticEntry-FilteringDatabase(6)}；

nbDynamicEntry-FilteringDatabase 名称绑定
从属对象类 oDatabaseEntry 和子类；
由高级对象类命名 oFilteringDatabase 和子类；
带属性 地址；
删除；
注册为 {Bridge.nameBinding dynamicEntry-FilteringDatabase(7)}；

nbGroupEntry-过滤数据库名称 绑定
由具有属性的上级对象类命名的下级对象类 oDatabaseEntry 和子类；oFilteringDatabase 和子类；aAddress；

注册为 {Bridge.nameBinding groupEntry-FilteringDatabase(8)}；

15.8.2 地址属性定义

地址属性
WITH 属性语法 桥.地址；
匹配 平等、秩序 ；
行为
bAddress 行为
定义为 !参见 14.7.6.1.2 b)、14.7.6.2.2 b)、14.7.6.3.2 b)、14.7.6.3.3 a) 和 14.7.6.4.3 c)1)。该属性可以是 MAC 地址（动态和组条目）或由端口号和 MAC 地址组成的序列（静态条目），其中端口号表示入站

端口。在后一种情况下，使用端口号 256 来表示“所有端口”。!

；
；
注册为 {Bridge.attribute 地址 (57)}；

15.8.3 端口映射属性定义

aPortMap 属性
WITH 属性语法 桥梁.端口映射 ；
匹配 平等；
行为
bPortMap 行为
定义为 参见 14.7.6.3.3 c) 和 14.7.6.4.3 c) 3)。
这**端口号**语法用于动态过滤条目管理对象类的实例中，以指示指定转发的端口。

这**端口映射**语法用于组注册表项和无法根据动态过滤信息的使用指示转发/过滤的静态过滤条目（见 7.9.1）。BITSTRING 用于表示出站端口转发/过滤指示符。BIT-STRING 中给定位的位数对应于它所指端口的端口号；位中的值为 1 表示转发，0 表示过滤。

这**端口扩展映射**语法用于静态过滤条目，这些条目能够根据动态过滤信息的使用指示转发/过滤（见 7.9.1）。BITSTRING 中的位对用于表示出站端口转发/过滤指示符。BITSTRING 中的位 0 和 1 对应端口号 0，位 2 和 3 对应端口号 1；依此类推。在每对位中，较低位中的值为 1 表示基于动态过滤信息的转发或过滤。较低位中的值为 0 表示基于较高位的值的转发或过滤；较高位中的值为 1 表示转发，

0 表示过滤。！

;
;
注册为 {Bridge.attribute portMap(58)};

15.8.4 条目类型属性定义

aEntryType 属性
WITH 属性语法 桥接入口类型 ;
匹配 平等;
行为
bEntryType 行为
定义为 !参见 14.7.6.3.3 b)、14.7.6.4.3 c) 2)。!
;
;
注册为 {Bridge.attribute entryType(50)};

15.8.5 条目索引属性定义

aEntryIndex 属性
WITH 属性语法 桥接入口索引 ;
匹配 平等、秩序 ;
行为
bEntryIndex 行为
定义为 !给定数据库（过滤或永久）中数据库条目的索引号。其值范围从 0 到（数据库大小）-1。在给定数据库中，所有数据库条目都应分配唯一的索引号。！

;
;
注册为 {Bridge.attribute entryIndex(51)};

15.8.6 读取数据库条目属性组定义

读取数据库条目 属性组
团体 元素 端口图，
条目类型，
条目索引
;
固定的 ;
描述 !此属性组用于读取过滤条目和读取过滤条目范围操作的映射，如表 15-26 和

表 15-27.!;
注册为 {Bridge.attributeGroup readDatabaseEntry(7)};

15.9 GARP 应用管理对象类定义

oGARP应用程序管理对象类
源自 “ISO/IEC 10165-2”：顶部 ;
特征 经过
pGARP应用 包裹
行为
bGARP应用 行为

定义为 !GARP 应用程序管理对象类充当与特定 GARP 应用程序相关的 GARP 属性类型对象 (14.9.2) 的容器对象。!

;
;
属性 应用程序名称 GET——命名属性
;
;
;
注册为 {Bridge.moClass garpApplication(8)};

15.9.1 GARP 应用程序的名称绑定

nbGARP应用程序端口名称绑定
由具有属性的上级对象类命名的下级对象类 oGARP应用和 子类;
oPort 和子类;
应用程序名称;
注册为 {Bridge.nameBinding garpApplication-Port(9)};

15.9.2 应用程序名称属性定义

应用程序名称 属性
带属性 句法 桥接器应用程序名称 ;
匹配 平等;
行为
应用程序名称 行为
定义为 !GARP 应用程序的名称，由相关应用程序的 GARP 应用程序地址组成（参见 12.3.1）。!

;
;
注册为 {Bridge.attribute applicationName(59)};

15.10 GARP 属性类型管理对象类定义

oGARPAttributeType 管理对象类
源自 “ISO/IEC 10165-2”：顶部 ;
特征 经过
属性类型 包裹
行为
博格ARP属性类型 行为
定义为 !GARP 属性类型管理对象类是 GARP 属性管理对象类的管理对象的容器对象。该类的实例包含在 GARP 应用程序管理对象中，该对象针对该应用程序支持的每个属性类型都包含该实例。该对象还模拟了修改属性类型的申请人管理控制值以及读取该属性类型的失败注册计数的能力（参见 14.9.2）。!

;
;
属性 一个GARP属性类型 GET,--命名属性
申请人行政控制
获取-替换，
注册失败 得到
;
;
;
注册为 {Bridge.moClass garpAttributeType(9)};

15.10.1 GARP 属性类型的名称绑定

nbGARPAttributeType-GARPParticipant 名称绑定
从属对象类 oGARPAttributeType和子类;
由高级对象类命名 oGARPParticipant 和子类;
带属性 aGARP属性类型;
注册为 {Bridge.nameBinding garpAttributeType-GARPParticipant(10)};

15.10.2 GARP 属性类型属性定义

```
aGARPAttributeType 属性
   WITH 属性语法                    桥接.GARP属性类型相等;          ;
   匹配
   行为
      bGARP属性类型          行为
         定义为                    !此属性带有给定 GARP 应用程序的 GARP 属性类型的值（参见
                                   12.11.2.2）。!
      ;
   ;
注册为 {Bridge.attribute garpAttributeType(64)};
```

15.10.3 申请人行政控制属性定义

```
申请人行政控制                          属性
   WITH 属性语法                    桥梁.申请人行政控制平等                          ;
   匹配                                ;
   行为
      b申请人行政控制                       行为
         定义为                    !此属性携带申请人行政控制参数的值

                                   GARP 属性类型（参见 12.9.2）!
      ;
   ;
注册为 {Bridge.attribute 申请人AdministrativeControl (66)};
```

15.10.4 失败注册属性定义

```
注册失败                  属性
   衍生的     从              “ISO/IEC   10165-5”:nonWrapping64BitCounter        ;
   匹配       为了             平等，        订购;
   行为
      注册失败                行为
         定义为                !为 GARP 属性类型维护的失败注册计数器的当前值（参见 14.9.2）。!

      ;
   ;
注册为 {Bridge.attribute failedRegistrations(63)};
```

15.11 GARP 属性管理对象类定义

```
oGARPAttribute 管理对象类
   源自                          “ISO/IEC   10165-2”：顶部   ;
   特征              经过
      属性              包裹
         行为
            boGARP属性          行为
               定义为             !GARP 属性管理对象类为 GARP 属性实例在状态机上可执行的操作建模。此类的实例包含
                                  在 GARP 属性类型管理对象中，该对象包含相关应用程序支持的任何 GIP 上下文中存在的
                                  属性类型每个实例。!


            ;
         ;
         属性                    aGARP属性                          GET, -- 命名属性 GET
                                 状态值
         ;
      ;
   ;
   有条件套餐
      pGARPOriginatorSupport     包裹
         属性                    最后一个 PDU 的发起者       得到
         ;
      注册为 {Bridge.package garpOriginatorSupport (1)};
```

```
   如果                                !GARP (14.9.2.1) 支持记录最后一个 GARP PDU 发起者地址。!;

注册为 {Bridge.moClass garpAttribute (10)};
```

15.11.1 GARP 属性的名称绑定

```
nbGARPAttribute-GARPAttributeType 名称绑定
   从属对象类                                          oGARPAttribute 和子类;
   由高级对象类命名                                    oGARPAttributeType 和子类;
   带属性                                              aGARP属性;
注册为 {Bridge.nameBinding garpAttribute-GARPAttributeType(11)};
```

15.11.2 GARP 属性 属性定义

```
aGARPAttribute 属性
   WITH 属性语法                    Bridge.GARP属性          ;
   匹配                             平等;
   行为
      bGARP属性          行为
         定义为                     !这是 GARP 属性管理对象类的命名属性。它由 GIP 上下文标识符值 (12.3.3) 和 GARP 属性值
                                    (12.11.2.6) 连接而成。!

      ;
   ;
注册为 {Bridge.attribute garpAttribute (65)};
```

15.11.3 状态值属性定义

```
aStateValue 属性
   WITH 属性语法                    桥梁状态值               ;
   匹配                             平等;
   行为
      bStateValue 行为
         定义为                     !此属性携带 GARP 属性实例的申请人和注册商组合状态的值（见
                                    12.8.4）。!

      ;
   ;
注册为 {Bridge.attribute stateValue (67)};
```

15.11.4 OriginatorOfLastPDU 属性定义

```
aOriginatorOfLastPDU 属性
   WITH 属性语法                    桥接器.OriginatorOfLastPDU 相等;   ;
   匹配
   行为
      最后一个 PDU 的发起者      行为
         定义为                     !此属性携带最后一个 GARP PDU 发起者的源 MAC 地址值，该 PDU 导致 GARP 状
                                    态机的状态发生变化，适用于 GARP 属性管理对象类的此实例


                                    （见 14.9.2.1.3）!
      ;
   ;
注册为 {Bridge.attribute originatorOfLastPDU (68)};
```

15.12 GARP 定时器管理对象类定义

```
oGARPTimers 管理对象类
   源自                             “ISO/IEC   10165-2”：顶部   ;
   特征                 经过
      pGARP定时器       包裹
         行为
            boGARP定时器     行为
               定义为                !GARP 定时器管理对象类模型
                                     可以对所有使用的计时器值执行的操作
```

与给定端口关联的 GARP 应用程序。此管理对象类的实例包含在支持任何 GARP 应用程序的任何端口的端口管理对象中。！

;
;
属性 aGARP定时器 GET, -- 命名属性 GET-REPLACE,
加入时间
离开时间 获取-替换，
离开所有时间 获取替换
;
;
;
注册为 {Bridge.moClass garpTimers (11)};

15.12.1 GARP 计时器的名称绑定

nbGARPTimers 端口名称绑定
由具有属性的上级对象类命名的下级对象类 oGARPTimers 和子类；oPort 和子类；

aGARP计时器；
注册为 {Bridge.nameBinding garpTimers-Port(12)};

15.12.2 GARP 定时器属性定义

aGARPTimers 属性
WITH 属性语法 Bridge.GARP定时器 ;
匹配 平等；
行为
bGARPTimers 行为
定义为 !这是 GARP 计时器对象的命名属性。！
;
;
注册为 {Bridge.attribute garpTimers (69)};

15.12.3 加入时间属性定义

aJoinTime 属性
WITH 属性语法 桥接.连接时间；
匹配 平等、秩序 ;
行为
bJoinTime 行为
定义为 !GARP 参与者当前使用的连接时间值（参见 12.12.1）。！

;
;
注册为 {Bridge.attribute joinTime(60)};

15.12.4 请假时间属性定义

aLeaveTime 属性
WITH 属性语法 桥.离开时间；
匹配 平等、秩序 ;
行为
bLeaveTime 行为
定义为 !GARP 参与者当前使用的休假时间值（参见 12.12.1）。！

;
;
注册为 {Bridge.attribute leaveTime(61)};

15.12.5 全部离开 属性定义

aLeaveAllTime 属性
带属性 句法 Bridge.LeaveAllTime ;
匹配 平等、秩序 ;
行为
离开所有时间 行为

定义为 !GARP 参与者当前使用的“全部离开时间”值（参见 12.12.1）。!

;
;
注册为 {Bridge.attribute leaveAllTime(62)};

15.13 ASN.1 定义

以下 ASN.1 模块定义了完成 MAC Bridge 的管理对象类定义所需的数据类型和数据值。

```
桥 {iso(1) member-body(2) us(840) ieee802dot1D(10009) asn1Module(2) bridgeDefinitions(0) version2(1)}
定义 ::= 开始

进口
- - 从 IEEE Std. 802.1F MACAddress 导入
MACAddress
摘自 IEEE802CommonDefinitions {iso(1) 成员机构(2) us(840) ieee802-1F(10011)
                               asn1module(2) commondefinitions(0) version1(0) }
；——进口结束

- - 为 OID 前缀桥定义短格式标识符
                对象标识符 ::={iso(1) member-body(2) us(840) ieee802dot1D(10009)} 对象标
行动            识符 ::={bridge action(9)}
asn1模块        对象标识符 ::={bridge asn1Module(2)} 对象标识符 ::={bridge
属性            attribute(7)} 对象标识符 ::={bridge attributeGroup(8)} 对象标识
属性组          符 ::={bridge standardSpecificExtensions(0)} 对象标识
扩展            符 ::={bridge managedObjectClass(3)} 对象标识符 ::={bridge
莫班            nameBinding(6)}
名称绑定
通知            对象标识符 ::={桥接通知(10)} 对象标识符 ::={桥接包
包裹            (4)} 对象标识符 ::={桥接参数(5)}
范围

- - 定义 GDMO 定义 AccessPriority 使用的其他类型和值
                                ::= 整数                - - 范围 0 至 7
地址                            ::= 选择                {[0] MAC地址，                - - 动态条目
                                                         [1] 序列                      {PortNumber，
                                                                                        MAC地址}
                                                        }                             - - 静态条目
老化时间                        ::= 整数                - - 以 1/256 秒为单位
申请人行政控制                  ::= 布尔值              {已禁用 FALSE，
                                                        已启用 TRUE}
应用程序名称                    ::= GARP应用程序地址 ::=
桥地址                          MAC地址
桥接转发延迟                    ::=转发延迟             - - 范围 4.0 - 30.0 秒
桥HelloTime                     ::= 你好时间            - - 范围 1.0 - 10.0 秒
桥梁识别符                      ::= 序列                {桥梁优先级，
                                                        桥地址 }
BridgeMaxAge                    ::= 最大年龄            - - 范围 6.0 - 40.0 秒
桥名                            ::= 图形字符串          - - 最多 32 个字符
桥接端口数                      ::= 整数                - - 范围 1 - 255
桥端口地址                      ::= 序列集              {端口号，
                                                        端口地址 }
桥接优先级                      ::= 整数                - - 范围 0 - 65535
数据库名称                      ::= 图形字符串
数据库大小                      ::= 整数
指定桥梁                        ::= 桥梁标识符
指定成本                        ::= 整数
指定端口                        ::= 端口标识符
指定根                          ::= 桥梁标识符
错误详细信息丢弃                ::= 序列集              {源地址                MAC地址，
                                                        原因                   丢弃原因 }
```

```
                                                    - - SET 中有 16 个元素
丢弃原因                    ::= 整数                {框架太大(1)
                                                    - - 目前没有其他原因--}
条目索引                    ::= 整数
条目类型                    ::= 整数                {静态（1），
                                                    动态（0） }
过滤数据库名称 GraphicString ::= “fdb”
ForwardDelay                ::= 整数                - - 以 1/256 秒为单位
GARP应用地址                ::= MAC地址
GARP属性                    ::= 序列                {GIPContextID，
                                                    attributeValue OCTETSTRING}
- - 属性值字段携带一个以八位字节序列编码的 nAttribute 值，如 GARP 应用程序所定义
- - 属性的使用范围。八位字节串的编码和解释与此类属性的定义相同
- - 在 GARP PDU (12.11) 中以八位字节序列形式携带时的值。
GARPAttributeType           ::= 整数
GARP定时器                  ::= GraphicString
网关IP上下文ID              (“GT”) ::= INTEGER
你好时间                    ::= 整数                - - 以 1/256 秒为单位
保持时间                    ::= 整数                - - 以 1/256 秒为单位
加入时间                    ::= 整数                - - 单位为厘秒
离开时间                    ::= 整数                - - 单位为厘秒
全部离开                    ::= 整数                - - 单位为厘秒
最大年龄                    ::= 整数                - - 以 1/256 秒为单位
条目数                      ::= 整数
动态条目数                  ::= 整数
组注册条目数最后一个 PDU 的发起者 ::= 整数
                            ::= MAC地址
出站访问优先级表            ::= 序列                {访问优先级，        - - 用户优先级 = 0
                                                    访问优先级，        - - 用户优先级 = 1
                                                    访问优先级，        - - 用户优先级 = 2
                                                    访问优先级，        - - 用户优先级 = 3
                                                    访问优先级，        - - 用户优先级 = 4
                                                    访问优先级，        - - 用户优先级 = 5
                                                    访问优先级，        - - 用户优先级 = 6
                                                    访问优先级}         - - 用户优先级 = 7
永久数据库名称 GraphicString := “pdb” 端口地
址                          ::= MAC地址
端口位图                    ::= 位串
端口扩展映射                ::= 位串
端口标识符                  ::= 序列                {端口优先级，
                                                    端口号 }
端口名称                    ::= 图形字符串                - - 最多 32 个字符
端口号                      ::= 整数
端口映射                    ::= 选择                {[0] 端口号，
                                                     [1] 端口位图
                                                     [2] 端口扩展映射
端口优先级                  ::= 整数                      - - 范围 0 - 255
路径成本                    ::= 整数                      - - 范围 1 - 65535
端口类型                    ::= 对象标识符
根路径成本                  ::= 整数
根端口                      ::= 端口标识符
强制端口状态                ::= 状态 — 新状态仅限于禁用或阻止 ::= INTEGER
状态                                                {已禁用 (0)，        - - 港口国
                                                    聆听（1）
                                                    学习（2），
                                                    转发（3），
                                                    阻塞（4）}
状态值                      ::= 整数                {va-mt（0），
                                                    va-lv (1),
                                                    副总裁-MT (2),
                                                    副总统-LV (3),
                                                    vo-mt（4），
                                                    vo-lv (5),
                                                    va-in（6），
                                                    副总统（7）
```

```
                                                    声音输入 (8),
                                                    AA-MT（9），
                                                    aa-lv (10),
                                                    aa-in（11），
                                                    ap-in（12），
                                                    ao-in（13）
                                                    qa-mt（14），
                                                    qa-lv（15），
                                                    卡因（16），
                                                    qp-in（17），
                                                    qo-in（18），
                                                    拉山（19），
                                                    la-lv (20),
                                                    洛山（21）
                                                    lo-lv（22），
                                                    拉-在（23）}
拓扑变化                        ::= 布尔值
拓扑改变确认                    ::= 布尔值
流量分类表                      ::= 序列 {
                                        不支持的类                  INTEGER，--范围 0 到 7
                                        TrafficClassTableEntry 的序列
- - SEQUENCE OF 中的元素数量恰好等于流量数量
- - 端口上支持的类别。
TrafficClassTableEntry          ::= 序列        {交通类别           INTEGER，--范围 0 到 7 用户优
                                                优先事项            先级设置}
- - 每个可能的用户优先级值在流量类别表中恰好出现一次；即，
- - 不允许创建从用户优先级到流量的映射表
- - 类不明确或未定义。类型值
                                ::= 整数                - - 范围 0 - 65535
正常运行时间                    ::= 整数                - - 单位为 1/256 秒
用户优先级                      ::= 整数                - - 范围 0 至 7
用户优先级再生表                ::= 序列        {用户优先级，       - - 接收优先级 = 0
                                                用户优先级，        - - 接收优先级 = 1
                                                用户优先级，        - - 接收优先级 = 2
                                                用户优先级，        - - 接收优先级 = 3
                                                用户优先级，        - - 接收优先级 = 4
                                                用户优先级，        - - 接收优先级 = 5
                                                用户优先级，        - - 接收优先级 = 6
                                                用户优先级}         - - 接收优先级 = 7
结尾
```

16. 桥梁性能

本条款规定了一组表示 Bridge 性能的参数。选择这些参数是为了在 Bridge 中建立基本的置信度，用于初步确定其是否适用于给定的应用。它们不能被视为对 Bridge 性能的详尽描述。建议提供并寻求有关 Bridge 实现适用性的进一步性能信息。

定义了以下一组性能参数：

a) 保证端口过滤率，以及相关的时间间隔*电视*$_F$它们共同表征了可以保证过滤的流量。

b) 保证桥接中继速率，以及相关的时间间隔*电视*$_R$。

16.1 保证端口过滤率

对于特定的桥接端口，有效的**保证端口过滤率**以每秒帧数为单位，是指给定特定桥接端口的任何一组帧在任何时间段内被过滤的值。*电视*$_F$间隔，只要以下所有条件都为真，转发进程就会过滤所有集合：

a) 集合中的帧数不超过特定 Bridge Port 的**保证端口过滤率**乘以*电视*$_F$。

b)**保证端口过滤率**每个其他桥接端口的容量均未超出。
c)**保证桥接中继速率**没有超出。
d) 中继帧不会因输出拥塞而被丢弃（7.7.3）。
e) 在时间间隔开始之前，已在过滤数据库中配置了过滤决策所依据的信息*电视*$_F$。

16.2 保证桥接中继速率

对于 Bridge 来说，有效的**保证桥接中继速率**以每秒帧数为单位，是给定特定桥接端口的任何一组帧在任何*电视*$_R$间隔内，只要以下所有条件都成立，转发过程就应中继所有集合：

a) 套装中的框架数量不超过 Bridge 的**保证桥接中继速率** 乘以*电视*$_R$。

b)**保证端口过滤率**每个桥端口的容量不得超过。
c) 中继帧不会因为输出拥塞而被丢弃（7.7.3）。
d) 在时间间隔开始之前，转发决策所依据的信息已在过滤数据库中配置*电视*$_R$。

附件 A

（规范性）

PICS 形式[1]

A.1 简介

声称符合本标准的协议实施的供应商应填写以下协议实施一致性声明 (PICS) 形式。如果声称支持源路由透明桥接操作，则供应商还应填写附件 D 中描述的协议实施一致性声明 (PICS) 形式。

完整的 PICS 形式是所讨论的实施的 PICS。PICS 是一份关于已实施协议的功能和选项的声明。PICS 有多种用途，包括使用

a) 由协议实施者提供一份清单，以减少因监督而无法符合标准的风险；

b) 由实施的供应商和获取者（或潜在获取者）提供，作为实施能力的详细说明，相对于标准 PICS 形式提供的理解的共同基础进行说明；

c) 由该实现的用户（或潜在用户）作为初步检查与另一实现互通可能性的基础（注意，虽然互通永远无法得到保证，但通常可以从不兼容的 PICS 中预测到互通失败）；

d) 由协议测试人员作为选择适当测试的基础，用以评估实施的一致性声明。

A.2 缩写和特殊符号

A.2.1 地位象征

米 强制的
哦 选修的
在 可选，但至少支持一组用相同数字标记的选项*n*是必须的

十 禁止
预测：条件项符号，包括谓词标识：参见 A.3.4 逻辑否定，应用于条件项
¬ 的谓词

A.2.2 通用缩写

不适用不适用
前额叶皮质 协议实施一致性声明

[1]*PICS 形式的版权发布：* 本标准的用户可以自由复制本附件中的 PICS 形式，以便将其用于预期目的，并可进一步发布完整的 PICS。

A.3 PICS 表格填写说明

A.3.1 PICS 形式的总体结构

PICS 形式的第一部分，即实施标识和协议摘要，应按要求填写，并提供充分识别供应商和实施所需的信息。

PICS 表格的主要部分是一份固定格式的问卷，分为几个小节，每个小节包含多个单独的项目。问卷项目的答案应填写在最右边的栏中，要么简单地标记一个答案以表示一个受限选项（通常是“是”或“否”），要么输入一个值或一组值或一个值的范围。（请注意，有些项目可以从一组可能的答案中选出两个或多个选项；所有相关选项都应标记。）

每个项目在第一列中通过项目参考标识。第二列包含要回答的问题；第三列记录项目的状态 — 支持是强制性的、可选的还是有条件的；另请参阅下面的 A.3.4。第四列包含对本标准正文中指定该项目的材料的参考，第五列提供答案的空间。

供应商还可以提供（或被要求提供）进一步的信息，这些信息被归类为附加信息或例外信息。如果存在，则每种进一步的信息都将在标有*人工智能*或者*习，*分别用于交叉引用目的，其中*我*是项目的任何明确标识（例如，只是一个数字）。其格式和表示没有其他限制。

完整的 PICS 形式（包括任何附加信息和例外信息）是相关实施的协议实施一致性声明。

注：如果实现能够以多种方式配置，则单个 PICS 可能能够描述所有此类配置。但是，供应商可以选择提供多个 PICS，每个 PICS 涵盖实现配置功能的某些子集，以便更轻松、更清晰地呈现信息。

A.3.2 附加信息

附加信息项允许供应商提供旨在帮助解释 PICS 的更多信息。我们并不打算或期望供应商提供大量信息，即使没有此类信息，PICS 也可以被视为完整。示例可能是概述如何设置（单个）实现以在各种环境和配置中运行，或者有关实现方面的信息，这些信息不在本标准的范围内，但与某些项目的答案有关。

对附加信息项目的引用可以输入在问卷中任何答案的旁边，也可以包含在例外信息项目中。

A.3.3 异常信息

有时，供应商可能会希望以与指示的要求相冲突的方式回答具有强制性状态的项目（在应用任何条件之后）。 对于这种情况，支持栏中没有预先打印的答案：相反，供应商应将缺失的答案连同*席*引用例外信息项，并应在例外项本身中提供适当的理由。

以这种方式需要异常项的实现不符合此标准。

注——上述情况的一个可能原因是，该标准中有一个缺陷被报告，预计该缺陷的修正将改变实施中未满足的要求。

A.3.4 条件状态

A.3.4.1 条件项目

PICS 表格包含许多条件性项目。这些项目本身的适用性以及其适用状态（强制或可选）取决于是否支持某些其他项目。

如果一组项目受相同适用条件的约束，则该组开头会出现一个关于该条件的单独初步问题，并指示如果选择“不适用”答案，则跳至问卷的后面部分。否则，单个条件项目将通过“状态”栏中的条件符号表示。

条件符号的形式为“**预测：**S” 其中**预测**是如下文 A.3.4.2 中描述的谓词，S 是状态符号、M 或 0。

如果谓词值为真（见 A.3.4.2），则条件项适用，其状态由谓词后的状态符号表示：答案栏应按通常方式标记。如果谓词值为假，则应标记“不适用”（N/A）答案。

A.3.4.2 谓词

谓词是下列之一：

a) PICS 形式中项目的项目引用：如果该项目被标记为支持，则谓词的值为真，否则为假；

b) 谓词名称，对于定义为布尔表达式的谓词，该谓词通过使用布尔运算符 OR 组合项引用而构造：如果一个或多个项被标记为支持，则谓词的值为真；

c) 在项引用或谓词名称前面加上逻辑否定符号“¬”：如果省略“¬”符号而形成的谓词的值为假，则该谓词的值为真，反之亦然。

在谓语或谓语定义中，或在分组条件项的初步问题中使用引用的每个项目都以“项目”列中的星号表示。

A.4 IEEE 标准 802.1D-1998 的 PICS 形式

A.4.1 实施标识

供应商	
有关 PICS 的疑问联系点	
实施名称和版本	
全面识别所需的其他信息——例如机器名称和版本以及/或操作系统名称	
注 1—所有实施方案仅需要前三项；其他信息可根据需要填写，以满足完整识别的要求。 注 2—术语“名称”和“版本”应进行适当解释，以与供应商的术语相对应（例如，类型、系列、型号）。	

A.4.2 协议摘要，IEEE 标准 802.1D-1998

协议规范识别	IEEE Std 802.1D-1998，信息技术 - 系统间电信和信息交换 - 局域网和城域网 - 通用规范 - 媒体访问控制 (MAC) 桥接器
识别已作为 PICS 一部分完成的 PICS 表格的修正和更正	补充。 ： 纠正。 ： 补充。 ： 纠正。 ：
是否需要任何例外项目？（参见 A.3.3：答案是“是”表示实施不符合 IEEE Std 802.1D-1998。）	不 [ ] 是的 [ ]

声明日期	

A.5 主要功能和选项

物品	特征	地位	参考	支持
（1a）	通讯支持 桥接端口支持哪些符合相关 MAC 标准的媒体访问控制类型？		6.5	
(1a.1) *	CSMA/CD、IEEE 标准 802.3	O.1		是的 [ ] 不 [ ]
(1a.2) *	令牌总线、ISO/IEC 8802-4 令	O.1		是的 [ ] 不 [ ]
(1a.3) *	牌环、ISO/IEC 8802-5 FDDI、	O.1		是的 [ ] 不 [ ]
(1a.4) *	ISO 9314-2	O.1		是的 [ ] 不 [ ]
(1a.5) *	DQDB，ISO/IEC 8802-6	O.1		是的 [ ] 不 [ ]
(1a.6) *	冰岛，ISO/IEC 8802-9	O.1		是的 [ ] 不 [ ]
(1a.7) *	ISLAN 16-T、IEEE Std 802.9a 需求优先	O.1		是的 [ ] 不 [ ]
(1a.8) *	级、ISO/IEC 8802-12（IEEE Std 802.3 格式）	O.1		是的 [ ] 不 [ ]
(1a.9) *	需求优先级，ISO/IEC 8802-12（ISO/IEC 8802-5格式）	O.1		是的 [ ] 不 [ ]
(1a.11) *	无线局域网，ISO/IEC DIS 8802-11	O.1		是的 [ ] 不 [ ]
（1b）	符合 ISO/IEC 8802-2 的所有桥接端口是否都支持 LLC 类型 1?	米	7.2, 7.3, 7.12, ISO/IEC 8802-2	是的 [ ]
(1c)	任何桥接端口是否支持源路由透明桥接操作？（如果声称支持，还应填写附件 D 中详述的 PICS 形式表格。）	哦	附件 C	是的 [ ] 不 [ ]
（2）	帧的中继和过滤（A.6）	米	7.5, 7.6, 7.7	是的 [ ]
（2a）	Bridge 是否支持基本过滤服务?	米	6.6.5, 7.7.2	是的 [ ]
(2b) *	Bridge 是否支持扩展过滤服务?	哦	6.6.5, 7.7.2	是的 [ ] 不 [ ]
	如果不支持第 (2b) 项，则标记“N/A”并继续第 (2e) 项。			不适用[ ]
(2c) *	网桥是否支持动态组转发和过滤行为?	2b：M	6.6.5	是的 [ ] 不 [ ]
(2d) *	网桥是否支持针对单个 MAC 地址的静态过滤信息来指定端口子集的功能，以便根据动态过滤信息做出转发或过滤决策?	2b：哦	6.6.5	是的 [ ] 不 [ ]
(2e)	桥接器是否支持其任何端口上的加速流量类别?	哦	7.1.2, 7.7.3	是的 [ ] 不 [ ]
（4） *	Bridge 是否支持中继帧的优先级管理?	哦	6.5、7.5.1、7.7.3, 7.7.5, 表 7-1，表 7-2、表 7-3	是的 [ ] 不 [ ]
（5）	过滤信息的维护（A.7）	米	7.8, 7.9	是的 [ ]
(7a)	管理层可以读取过滤数据库吗?	哦	7.9	是的 [ ] 不 [ ]

A.5 主要功能和选项 *（持续）*

物品	特征	地位	参考	支持
(7c) *	静态过滤条目可以创建和删除吗？	哦	7.9.1	是的 [ ] 不 [ ]
（7克）	可以在永久数据库中创建和删除静态过滤条目吗？	哦	7.9.6	是的 [ ] 不 [ ]
(7小时)	是否可以为给定的 MAC 地址规范创建静态过滤条目，并为每个入站端口提供不同的端口映射？	哦	7.9.1	是的 [ ] 不 [ ]
(7—)	GMRP 可以动态创建、更新和删除组注册条目吗？	2c:M	7.9.3, 10	是的 [ ] 不适用 [ ]
（10）	寻址（A.8）	米	7.12	是的 [ ]
(9a) *	可以将网桥配置为使用 48 位通用地址吗？	O.3	7.12	是的 [ ] 不 [ ]
(9b) *	可以将网桥配置为使用 48 位本地地址吗？	O.3	7.12	是的 [ ] 不 [ ]
(13) *	生成树算法和协议（A.9）	米	8、9	是的 [ ]
(16) *	网桥是否支持生成树拓扑的管理？	哦	8.2	是的 [ ] 不 [ ]
(17) *	Bridge 是否支持协议计时器的管理？	哦	8.10	是的 [ ] 不 [ ]
(19) *	桥梁管理运营	哦	14	是的 [ ] 不 [ ]
(20a) *	桥梁管理操作是否通过远程管理协议支持？	19：O.4	5	是的 [ ] 不 [ ] 不适用 [ ]
(20b) *	是否通过本地管理接口支持桥梁管理操作？	19：O.4	5	是的 [ ] 不 [ ] 不适用 [ ]

A.6 帧的中继和过滤

物品	特征	地位	参考	支持
（2f）	接收到的带有媒体访问方法错误的帧会被丢弃吗？	米	6.4, 7.5	是的 [ ]
（2克）	正确接收的帧是否提交至学习过程？	米	7.5	是的 [ ]
(2小时)	用户数据帧是唯一经过中继的帧类型吗？	米	7.5	是的 [ ]
(2—)	没有响应的请求帧是唯一中继的帧吗？	米	7.5	是的 [ ]
（2j）	所有发往桥接协议实体的帧是否都提交给了它？	米	7.5	是的 [ ]
（2千）	用户数据帧是唯一传输的帧类型吗？	米	7.6	是的 [ ]

A.6 帧的中继和过滤 *（持续）*

物品	特征	地位	参考	支持
（2升）	是否只传输没有响应帧的请求帧?	米	7.6	是的 [ ]
（2分钟）	中继帧是否仅在 7.7.3 中的条件下排队传输?	米	7.7.3, 8.4	是的 [ ]
（2n）	中继帧的顺序是否按照转发过程的要求保留?	米	7.7.3, 7.1.1	是的 [ ]
(2o)	中继帧是否仅提交给 MAC 实体进行一次传输?	米	7.7.4, 6.3.4	是的 [ ]
（2 页）	是否对中继帧强制实施最大桥接传输延迟?	米	7.7.3	是的 [ ]
（2q）	如果端口离开转发状态，排队的帧会被丢弃吗?	米	7.7.3	是的 [ ]
（2 号）	中继帧的用户优先级是否尽可能保留?	米	6.4	是的 [ ]
（2秒）	否则，用户优先级是否设置为接收端口的默认用户优先级?	米	6.4	是的 [ ]
（2吨）	是否通过用户优先级再生表来再生用户优先级?	米	7.5.1，表 7-1	是的 [ ]
（2单位）	再生用户优先级到流量类别的映射是否通过流量类别表执行?	米	7.7.3，表 7-2	是的 [ ]
（2v）	访问优先级是否源自网桥支持的每种出站媒体访问方法的表 7-3 中的值定义的再生用户优先级?	米	7.7.5，表 7-3	是的 [ ]
（2周）	该实现是否会导致未检测到的帧错误率高于通过保留 FCS 可实现的帧错误率?	十	7.7.6, 6.3.7	不 [ ]
（2倍）	在同一 MAC 类型的端口之间中继的帧的 FCS 是否被保留?	哦	7.7.6	是的 [ ] 不 [ ]
（2岁）	在接收到桥接端口的本地 MAC 实体传输的有效帧时，桥接器是否会生成 M_UNITDATA.indication 原语?	MS1：X	6.5.4，ISO 9314-2	不 [ ] 不适用 [ ]
（2z）	是否仅使用异步服务?	MS1：M	ISO 9314-2 第 8.1.4 条	是的 [ ] 不适用 [ ]
（2AA）	在从 FDDI 环接收到要转发的帧时，网桥是否设置 C 指示器?	MS1：哦	6.5.4，ISO 9314-2 第 7.3.8 条	是的 [ ] 不 [ ] 不适用 [ ]
（2ab）	当从 FDDI 环接收到一个帧并进行转发时，网桥是否保持 C 指示器不变?	MS1：哦	6.5.4，ISO 9314-2 第 7.3.8 条	是的 [ ] 不 [ ] 不适用 [ ]
	如果第 4 项不受支持，则标记“N/A”并继续第 (4d) 项。			不适用 [ ]

A.6 帧的中继和过滤 *（持续）*

物品	特征	地位	参考	支持
(4a) *	每个端口的默认用户优先级参数是否可以设置为 0 至 7 范围内的任意值?	4：O.5	6.4	是的 [ ] 不 [ ]
(4b) *	每个端口的用户优先级再生表中的条目是否可以设置为表 7-1 所示的全部值范围?	4：O.5	7.5.1，表 7-1	是的 [ ] 不 [ ]
(4c)	每个端口的流量类别表中的条目是否可以设置为表 7-2 所示的所有值?	MS2：哦	7.7.3，表 7-2	是的 [ ] 不 [ ] 不适用 [ ]
	如果支持第 4 项，则标记“N/A”并继续第 (4g) 项。			不适用 [ ]
(4日)	网桥是否支持每个端口的默认用户优先级参数的推荐默认值?	¬ 4:M	6.4	是的 [ ]
(4e)	网桥是否支持表 7-1 中定义的每个端口接收用户优先级和再生用户优先级之间的推荐默认映射?	¬ 4:M	7.5.1，表 7-1	是的 [ ]
（4f）	网桥是否支持表 7-2 中所示的每个端口的推荐默认 user_priority 到流量类别映射?	三叉戟	7.7.3，表 7-2	是的 [ ] 不适用 [ ]
（4克）	在确定所示媒体访问方法的访问优先级时，网桥是否可以使用表 7-3 中所示值以外的任何值?	十	7.7.5，表 7-3	不 [ ]

谓词：
MS1 = 1a.4 与非 (1a.1 或 1a.2 或 1a.3 或 1a.5 或 1a.6 或 1a.7 或 1a.8 或 1a.9) MS2 = 2d 与 4
MS3 = 2d 且不为 4

A.7 过滤数据库中过滤条目的维护

物品	特征	地位	参考	支持
(5a)	当且仅当港口国允许时，才会创建和更新动态过滤条目吗?	米	7.8、7.9.2、8.4	是的 [ ]
(5b)	在收到具有组源地址的帧时是否会创建动态过滤条目?	十	7.8, 7.9.2	不 [ ]
(5c)	过滤数据库是否支持静态过滤条目?	米	7.9.1	是的 [ ]
(5日)	是否可以创建与现有静态过滤条目冲突的动态过滤条目?	十	7.8, 7.9, 7.9.1, 7.9.2	不 [ ]

A.7 过滤数据库中过滤条目的维护 *（持续）*

物品	特征	地位	参考	支持
(5e)	过滤数据库是否支持动态过滤条目?	米	7.9.2	是的 [ ]
（5f）	创建静态过滤条目是否会删除动态过滤条目中针对同一地址的任何冲突信息?	米	7.9.1, 7.9.2	是的 [ ]
（5克）	每个静态过滤条目是否指定一个 MAC 地址规范和一个端口映射?	米	7.9.1	是的 [ ]
(5小时)	如果在老化时间内没有更新，动态过滤条目是否会从过滤数据库中删除?	米	7.9.2	是的 [ ]
(5一)	每个动态过滤条目是否指定一个 MAC 地址规范和一个端口映射?	米	7.9.2	是的 [ ]
(5j)	过滤数据库是否使用永久数据库中包含的条目进行初始化?	米	7.9.6	是的 [ ]
	如果不支持第 (2c) 项，则标记 N/A 并继续第 (6a) 项。			不适用 [ ]
（5千）	每个组注册条目是否指定一个 MAC 地址规范和端口映射?	2c:M	7.9.3	是的 [ ]
（5升）	组注册表项中的 MAC 地址规范可以代表所有组、所有未注册的组还是特定组的 MAC 地址?	2c:M	7.9.3	是的 [ ]
(5分钟)	组注册条目是否按照 GMRP 的规范在过滤数据库中创建、更新和删除?	2c:M	7.9.3, 10	是的 [ ]
（5n）	除了通过 GMRP 操作之外，是否还可以通过其他方式在过滤数据库中创建、更新和删除组注册条目?	2c:X	7.9.3, 10	不 [ ]
(6a)	说明过滤数据库的大小。	米	7.9	_____ 条目
(6b)	说明永久数据库的大小。	米	7.9	_____ 条目
	如果不支持第（7c）项，则标记 N/A 并继续第（8a）项。			不适用 [ ]
(7日)	可以为单个 MAC 地址创建静态过滤条目吗?	**7c：**米	7.9.1	是的 [ ]
(7e)	可以为组 MAC 地址创建静态过滤条目吗?	**7c：**米	7.9.1	是的 [ ]

A.7 过滤数据库中过滤条目的维护 *（持续）*

物品	特征	地位	参考	支持
（7f）	可以为广播 MAC 地址创建静态过滤条目吗？	**7c：**米	7.9.1	是的 [ ]
(8a)	是否可以将网桥配置为使用表 7-4 中推荐的老化时间默认值？	哦	7.9.2，表 7-4	是的 [ ] 不 [ ]
(8b)	网桥是否可以配置为使用表 7-4 中指定的老化时间值范围中的任意一个？	哦	7.9.2，表 7-4	是的 [ ] 不 [ ]

A.8 寻址

物品	特征	地位	参考	支持
(10a)	每个端口都有单独的 MAC 地址吗？	米	7.12.2	是的 [ ]
(10b)	所有 BPDU 是否都传输到同一个组地址？	米	7.12.3, 8.2	是的 [ ]
	如果不支持第 (9a) 项，则标记 N/A 并继续第 (10d1) 项。			不适用 [ ]
(10分)	当使用通用地址时，所有 BPDU 是否都会传输到桥接协议组地址？	**9a：**米	7.12.3, 8.2	是的 [ ]
(10天)	BPDU 的源地址是发送端口的地址吗？	**9a：**米	7.12.3	是的 [ ]
(10d1)	BPDU 的 LLC 地址是否是生成树协议识别的标准 LLC 地址？	米	7.12.3，表 7-8	是的 [ ]
(10e)	桥接地址是通用地址吗？	米	7.12.5, 8.2	是的 [ ]
(10f)	网桥中继的帧是否发往任何保留地址？	十	7.12.6	不 [ ]
	如果第（13）项不受支持，则标记 N/A 并继续第（11c）项。			不适用 [ ]
(11a)	是否可以使用端口的 MAC 地址和分配的 LSAP 通过每个端口进行桥接管理？	**13：**哦	7.12.4	是的 [ ] 不 [ ]
(11b)	是否可以使用所有 LAN 桥接管理组地址通过所有端口访问桥接管理？	13：哦	7.12.4	是的 [ ] 不 [ ]
(11c)	桥地址是端口1的地址吗？	**9a：**哦	7.12.5	是的 [ ] 不 [ ] 不适用 [ ]
(11d)	永久数据库中预先配置的保留地址之外是否有组地址？	哦	7.12.6	是的 [ ] 不 [ ]

A.8 寻址 *（持续）*

物品	特征	地位	参考	支持
	如果第（11d）项不受支持，则标记 N/A 并继续第（12a）项。			不适用 [ ]
(11e)	可以删除过滤数据库中的附加预配置条目吗？	**11天：** 哦	7.12.6	是的 [ ] 不 [ ]
(12a)	是否可以分配一组 MAC 地址来识别桥接协议实体？	**9b：** 米	8.2	是的 [ ] 不适用 [ ]
(12c)	每个桥梁端口是否都有一个独特的标识符？	米	8.2、8.5.5.1	是的 [ ]

A.9 生成树算法

物品	特征	地位	参考	支持
(13a)	以下所有桥梁参数是否均得到维护？	米	8.5.3	是的 [ ]
	指定根		8.5.3.1	
	根成本		8.5.3.2	
	根端口		8.5.3.3	
	最大年龄		8.5.3.4	
	你好时间		8.5.3.5	
	转发延迟		8.5.3.6	
	桥梁识别符		8.5.3.7	
	桥梁最大年龄		8.5.3.8	
	桥接问候时间		8.5.3.9	
	桥接转发延迟		8.5.3.10	
	检测到拓扑变化		8.5.3.11	
	拓扑变化		8.5.3.12	
	拓扑变化时间		8.5.3.13	
	保持时间		8.5.3.14	
(13b)	以下所有桥接计时器是否均已维护？	米	8.5.4	是的 [ ]
	你好定时器		8.5.4.1	
	拓扑改变通知定时器		8.5.4.2	
	拓扑改变定时器		8.5.4.3	
(13c)	每个端口是否都维护以下所有端口参数？	米	8.5.5	是的 [ ]
	端口标识符		8.5.5.1	
	状态		8.5.5.2, 8.4	

A.9 生成树算法 *（持续）*

物品	特征	地位	参考	支持
	路径成本		8.5.5.3	
	指定根		8.5.5.4	
	指定费用		8.5.5.5	
	指定桥梁		8.5.5.6	
	指定端口		8.5.5.7	
	拓扑改变确认		8.5.5.8	
	配置待定		8.5.5.9	
	已启用变更检测		8.5.5.10	
(13d)	是否为每个端口维护以下所有计时器?	米	8.5.6	是的 [ ]
	消息年龄定时器		8.5.6.1	
	转发延迟定时器		8.5.6.2	
	保持定时器		8.5.6.3	
（十三）	是否按照下列每个事件的要求维护协议参数和计时器并传输 BPDU?	米	8.7, 8.9, 8.5.3, 8.5.4, 8.5.5, 8.5.6	是的 [ ]
	收到的配置 BPDU		8.7.1	
	收到拓扑变化通知 BPDU		8.7.2	
	Hello 定时器到期		8.7.3	
	消息年龄定时器到期		8.7.4	
	转发延迟定时器到期		8.7.5	
	拓扑变化通知定时器到期		8.7.6	
	拓扑变化计时器到期		8.7.7	
	保持定时器到期		8.7.8	
(13f)	以下操作是否会修改协议参数和计时器，并按要求传输 BPDU?	米	8.8, 8.9, 8.5.3, 8.5.4, 8.5.5, 8.5.6	是的 [ ]
	初始化		8.8.1	
	启用端口		8.8.2	
	禁用端口		8.8.3	
	设置桥接优先级		8.8.4	
	设置端口优先级		8.8.5	
	设置路径成本		8.8.6	
(13克)	实现是否支持将“更改检测已启用”参数的值设置为“已禁用”？	哦	8.5.5.10	是的 [ ] 不 [ ]

A.9 生成树算法 *（持续）*

物品	特征	地位	参考	支持
(14a)	网桥是否低估了传输的 BPDU 中的消息年龄参数的增量?	十	8.10.1	不 [ ]
(14b)	桥接器是否低估了前向延迟?	十	8.10.1	不 [ ]
(14c)	Bridge 是否高估了 Hello Time 间隔?	十	8.10.1	不 [ ]
(15a)	桥接器是否使用指定的保持时间值?	米	8.10.2，表 8-3	是的 [ ]
	如果第 (16) 项不支持，请标记 N/A 并继续第 (17a) 项			不适用 [ ]
(16a)	可以设置 Bridge 的相对优先级吗?	**16：**米	8.2、8.5.3.7、8.8.4	是的 [ ]
(16b)	可以设置端口的相对优先级吗?	**16：**米	8.2、8.5.5.1、8.8.5	是的 [ ]
(16c)	可以设置每个端口的路径成本吗?	**16：**米	8.2、8.5.5.3、8.8.6	是的 [ ]
	如果第 (17) 项不受支持，请标记 N/A 并继续第 (18a) 项。			不适用 [ ]
(17a)	Bridge Max Age 是否可以设置为指定范围内的任意值?	**17：**米	8.10.2、8.5.3.8、表 8-3	是的 [ ]
(17b)	桥接问候时间 (Bridge Hello Time) 是否可以设置为指定范围内的任意值?	**17：**米	8.10.2、8.5.3.9、表 8-3	是的 [ ]
(17c)	桥接转发延迟是否可以设置为指定范围内的任意值?	**17：**米	8.10.2，8.5.3.10，表 8-3	是的 [ ]
(18a)	所有 BPDU 都包含整数个八位字节吗?	米	9.1.1	是的 [ ]
(18b)	以下所有 BPDU 参数类型是否均按规定进行编码?	米	9.1.1, 9.2	是的 [ ]
	协议标识符		9.2.1	
	协议版本标识符		9.2.2	
	BPDU 类型		9.2.3	
	标志		9.2.4	
	桥梁标识符		9.2.5	
	根路径成本		9.2.6	
	端口标识符		9.2.7	
	计时器值		9.2.8	
(18c)	配置 BPDU 是否具有指定的格式和参数?	米	9.3.1	是的 [ ]
(18d)	拓扑改变通知 BPDU 是否具有指定的格式和参数?	米	9.3.2	是的 [ ]
(18e)	收到的 BPDU 是否按照规定进行验证?	米	9.3.3	是的 [ ]

A.10 桥梁管理

物品	特征	地位	参考	支持
	如果项目（19）没有得到支持，则标记N/A并继续（20c）。			不适用 [ ]
(19a)	探索桥	**19：** 米	14.4.1.1	是的 [ ]
(19b)	阅读桥	**19：** 米	14.4.1.2	是的 [ ]
(19c)	设置桥名称	**19：** 米	14.4.1.3	是的 [ ]
(19d)	重置桥	**19：** 米	14.4.1.4	是的 [ ]
(19e)	读端口	**19：** 米	14.4.2.1	是的 [ ]
(19f)	设置端口名称	**19：** 米	14.4.2.2	是的 [ ]
(19克)	读取转发端口计数器	**19：** 米	14.6.1.1	是的 [ ]
(19时)	读取过滤数据库	**19：** 米	14.7.1.1	是的 [ ]
(19一)	设置过滤数据库老化时间	**19：** 米	14.7.1.2	是的 [ ]
(19j)	读取永久数据库	**19：** 米	14.7.5.1	是的 [ ]
(1.9万)	创建过滤条目	**19：** 米	14.7.6.1	是的 [ ]
(19升)	删除过滤条目	**19：** 米	14.7.6.2	是的 [ ]
(19分钟)	讀取過濾項目	**19：** 米	14.7.6.3	是的 [ ]
(19n)	读取过滤条目范围	**19：** 米	14.7.6.4	是的 [ ]
(19度)	读取桥接协议参数	**19：** 米	14.8.1.1	是的 [ ]
(19页)	设置桥接协议参数	**19：** 米	14.8.1.2	是的 [ ]
(19q)	读取端口参数	**19：** 米	14.8.2.1	是的 [ ]
(19右)	强制港口状态	**19：** 米	14.8.2.2	是的 [ ]
(19秒)	设置端口参数	**19：** 米	14.8.2.3	是的 [ ]
(19吨)	读取端口默认用户优先级	第四章：M	14.6.2.1	是的 [ ] 不适用 [ ]
(19单位)	设置端口默认用户优先级	第四章：M	14.6.2.2	是的 [ ] 不适用 [ ]
(19伏)	读取端口用户优先级再生表	5. MS5：M	14.6.2.3	是的 [ ] 不适用 [ ]
(19 周)	设置端口用户优先级再生表	5. MS5：M	14.6.2.4	是的 [ ] 不适用 [ ]
(19倍)	读取端口流量类别表	第七章：M	14.6.3.1	是的 [ ] 不适用 [ ]
(19岁)	设置端口流量类别表	第七章：M	14.6.3.2	是的 [ ] 不适用 [ ]
(19z)	读取出站访问优先级表	第六章：M	14.6.3.3	是的 [ ] 不适用 [ ]
(19AA)	读取 GARP 计时器	第八章：M	14.9.1.1	是的 [ ] 不适用 [ ]
(19ab)	设置 GARP 计时器	第八章：M	14.9.1.2	是的 [ ] 不适用 [ ]
(19ac)	阅读 GARP 申请人控制	第八章：M	14.9.2.1	是的 [ ] 不适用 [ ]
（19）	设置 GARP 申请人控制	第八章：M	14.9.2.2	是的 [ ] 不适用 [ ]
(19ae)	读取 GARP 状态	第八章：M	14.9.3.1	是的 [ ] 不适用 [ ]

A.10 桥梁管理 *（持续）*

物品	特征	地位	参考	支持
	如果不支持第（20a）项，则标记 N/A 并继续第（20e）项。			不适用 [ ]
(20条)	支持哪些管理协议标准或规范?	**20A:** 米	5.	
(20天)	支持哪些管理对象和编码的标准或规范?	**20A:** 米	5.	
	如果不支持第（20b）项，则标记 N/A 并继续 A.11。			不适用 [ ]
(20e)	支持什么规格的本地管理接口?	**20b:** 米	5.	

谓词:
MS4=19 且 4a
MS5=19 且 4b
MS6=19 且 4
MS7=19 且 4c
MS8=19 且 2b

A.11 性能

物品	特征	地位	参考	支持
(21a)	指定保证端口过滤率以及相关的测量间隔*特遣队*，针对每个桥接端口，采用下面指定的格式。	米	16.1	
(21b)	指定保证桥接中继速率以及相关的测量间隔*贸易*，采用下面指定的格式。 补充信息应清楚地标明港口。	米	16.2	

保证桥接中继速率	贸易
每秒 _ _ _ _ _ _ _ _ _ _ 帧	_ _ _ _ _ 秒

A.11 性能 *（持续）*

端口号或 其他身份证明	保证端口过滤率 （指定所有端口）	*电视F* （指定所有端口
	每秒 ___________ 帧	_____ 秒
	每秒 ___________ 帧	_____ 秒
	每秒 ___________ 帧	_____ 秒
	每秒 ___________ 帧	_____ 秒
	每秒 ___________ 帧	_____ 秒
	每秒 ___________ 帧	_____ 秒
	每秒 ___________ 帧	_____ 秒
	每秒 ___________ 帧	_____ 秒

A.12 GARP 和 GMRP

物品	特征	地位	参考	支持
	如果不支持第 2b 项，则标记 N/A 并继续第 (22i) 项。			不适用 [ ]
(22a)	GMRP 应用程序地址是否在所有 GMRP 协议交换中用作目标 MAC 地址?	2b：M	10.4.1， 表 12-1	是的 [ ]
(22b)	GMRP 协议交换是否通过 LLC 类型 1 程序实现，并使用生成树协议的 LLC 地址?	2b：M	12.4、12.5、 表 7-8	是的 [ ]
(22c)	GMRP 协议交换是否使用 GARP PDU 格式实现，以及为 GMRP 定义的属性类型和值编码的定义?	2b：M	10.3.1、12.4、 12.5、12.11	是的 [ ]
(22日)	该实现是否支持申请人、注册商和全部离开状态机的操作?	2b：M	12.8, 13	是的 [ ]
(22e)	网桥是否仅在属于基本生成树上下文活动拓扑的端口上传播 GMRP 注册信息?	2b：M	12.3.3, 12.3.4, 十三	是的 [ ]
(22f)	处于转发状态的端口上接收到的 GARP PDU 是否根据处理 GARP 应用程序地址的要求进行转发、过滤或丢弃?	2b：M	7.12.3, 12.5	是的 [ ]

A.12 GARP 和 GMRP *（持续）*

物品	特征	地位	参考	支持
(22克)	GMRP 应用程序是否按照第 10 条的定义运行?	2b：M	10、10.3	是的 [ ]
(22小时)	收到的、对于所支持的 GARP 应用程序来说格式不正确的 GARP PDU 是否会被丢弃?	2b：M	10.3.1、12.4、12.5、12.10、12.11	是的 [ ]
(22一)	所有 GARP PDU 是否 （a）在处于转发状态的端口上接收，并且 （b）用于 Bridge 不支持的 GARP 应用程序， 在所有其他处于转发状态的端口上进行转发?	米	7.12.3, 12.5	是的 [ ]
(22j)	是否有任何 GARP PDU (a) 在 任何端口上接收，并且 （b）用于 Bridge 不支持的 GARP 应用程序， 提交给任何 GARP 参与者?	十	7.12.3, 12.5	不 [ ]
(22千)	是否有任何 GARP PDU （a）在任何未处于转发状态的端口上接收，并且 （b）用于 Bridge 不支持的 GARP 应用程序， 是否转发至该桥的任何其他端口?	十	7.12.3, 12.5	不 [ ]
(22升)	是否有任何 GARP PDU （a）在处于转发状态的任何端口上接收，并且 （b）用于 Bridge 支持的 GARP 应用程序， 是否转发至该桥的任何其他端口?	十	7.12.3, 12.5	不 [ ]
(22 分钟)	所有 GARP PDU 是否：(a) 在任何端口上接收，并且 （b）用于 Bridge 支持的 GARP 应用程序， 提交给适当的 GARP 参与者?	米	7.12.3, 12.5	是的 [ ]

附件 B

（信息性）

计算生成树参数

本附件描述了计算基本生成树算法性能参数的推荐值和操作范围的方法和原理。

B.1 概述

计算过程分为几个步骤。每个步骤都会确定一些参数的值，这些参数的值将作为后续步骤的基础。

给出的描述和公式适用于同构桥接 LAN，即其中所有单个 LAN 和桥接器的类型和速度都相同。很容易将其扩展到异构桥接 LAN。

解释以 IEEE 802 LAN 的推荐值为例。所有时间均以秒为单位。

B.2 缩写和特殊符号

直径 **最大桥梁直径** 最大限度**帧寿**
生活 **命**
时间 平均帧**运输延误** 平均的**介质访问**
疯狂的 **延迟 最大介质访问延迟 最大桥**
mma_d **接传输延迟** 决议**消息时代**
BT_D
时间单位
msg_aio **最大消息年龄增量高估最大消息年龄高估**
消息
普杜 **最大BPDU传输延迟**
丢失的消息 重新配置前允许的最大丢失桥接协议消息数 **最大桥接协议消息传播时间 Hello Time**
消息属性
你好
hold_t **保持时间**
最大年龄 **最大年龄**
转发延迟 **转发延迟**

B.3 计算

B.3.1 寿命、直径和传输延迟

B.3.1.1 步骤

选择桥接 LAN 的最大桥直径和最大桥传输延迟。请注意，如果各个 LAN 支持一系列传输优先级，则桥传输延迟可能会根据优先级而变化。

B.3.1.2 选择依据

帧的生存期等于最大桥直径乘以最大桥传输延迟加上初始传输的最大介质访问延迟，即

生活= (*直径*× *BT_D*) +*mma_d* (1)

平均**帧传输延迟**桥接 LAN 中端系统之间的通信量比单个 LAN 中端系统之间的通信量大，其平均值之和**转发延迟**和**帧传输延迟**端系统之间的路径上的桥梁。这些桥梁的顺序如下：**介质访问延迟**对于负载较轻的 LAN。因此，对于桥接 LAN 两端的系统，将有以下情况：

时间>=(*直径*× *疯狂的*) +*疯狂的* (2)

这限制了任何坚持低**桥梁通行最大延误**和**最大桥梁直径高**。

B.3.1.3 IEEE 802 桥接局域网的推荐值

mma_d>=0.5
生活<=7.5
直径=7
BT_D=1.0

B.3.2 BPDU 的传输

B.3.2.1 步骤

选择 BPDU 的传输优先级以及**最大BPDU传输延迟**。

B.3.2.2 选择依据

一般情况下，会选择较高的传输优先级，因为桥接 LAN 的持续运行取决于 BPDU 的成功传输和接收。在某些情况下，单个 LAN 的其他本地流量可能更重要。

可以选择的最低值**最大BPDU传输延迟**那么**最大介质访问延迟**对于该优先级的框架。考虑到在尝试实现该数字时可能出现的实施困难，选择等于该值似乎更合理**最大桥接传输延迟**对于以该优先级传输的帧。

普杜=*BT_D* (3)

B.3.2.3 IEEE 802 桥接局域网的推荐值

并非所有 IEEE 802 MAC 方法都支持优先级传输。因此，选择了以下方法：

普杜=*BT_D*
= 1.0

B.3.3 消息年龄的准确性

B.3.3.1 步骤

为**最大消息年龄增量估计过高**。

这是对传输的桥接协议数据单元中消息年龄参数值的增量的最大高估。此参数允许接收协议消息的桥接在消息太旧时丢弃其中的信息。传输桥接不应被允许低估此字段的值。

计算**最大消息年龄估计过高**，这是任何桥接器对收到的桥接协议消息信息的年龄做出的最大高估。

B.3.3.2 选择依据

选择**最大消息年龄增量估计过高**受以下条款管辖：

一个)*时间单位*—在配置消息中携带消息年龄参数的分辨率。

b) Bridge 中计时器的粒度和准确性。
c)**最大BPDU传输延迟**。

假设网桥定时器不一定与收到的 BPDU 同步，假设它们是准确的，并且它们的粒度为*时间单位*，我们将尽最大努力做到以下几点：

msg_aio=普杜+时间单位 (4)

注：由于time_unit较小，在B.3.3.3所示的msg_aio和msg_ao的推荐值中，公式（4）中的该项已近似为零。

该值应四舍五入为最接近的倍数*时间单位*这里值得注意的是，任何 Bridge 都会始终将值增加至少一个单位。

对于桥接器接收和存储桥接协议消息信息的计时器，**最大消息年龄估计过高**将等于**最大消息年龄增量估计过高**乘以**最大桥梁直径**减一：

消息=msg_aio× (*直径*–1) (5)

B.3.3.3 IEEE 802 桥接局域网的推荐值

msg_aio = 1.0
消息 = 6.0

B.3.4 你好时间

B.3.4.1 步骤

暂时选择一个 Hello Time 的值。

B.3.4.2 选择依据

选择 Hello Time 是基于其对最大桥接协议消息传播时间的贡献（参见下一步）。

以高于**最大BPDU传输延迟**在最坏的情况下，当试图保证正确操作时，这些消息就会相互碰撞。

临时价值为**最大BPDU传输延迟**建议。

你好=2×*普杜* (6)

B.3.4.3 IEEE 802 桥接局域网的推荐值

你好=2.0

B.3.5 桥接协议消息传播时间

B.3.5.1 步骤

计算**最大桥接协议消息传播时间**。

B.3.5.2 选择依据

这**最大桥接协议消息传播时间**是桥接协议消息信息从桥接到桥接穿越桥接 LAN 所花费的最大时间。它由以下部分组成：

a) 单个桥接协议消息穿过桥接 LAN 的最大传播时间，
IE，**最大BPDU传输延迟**乘以**最大桥梁直径**减一。
b) 津贴**你好时间**可容忍的连续丢失桥接协议消息的最大数量的倍数（请注意，即使丢失一条消息也是很少见的）。
c) 进一步允许**你好时间**，因为我们不应该假设与根桥同步，并且我们可能需要等待很长时间才能传输下一个 BPDU。

消息属性=（（*丢失的消息*+1）×*你好*）+*普杜*×（直径– 1） (7)

B.3.5.3 IEEE 802 桥接局域网的推荐值

假设*丢失的消息*=3、

消息属性=14.0

B.3.6 保持时间

B.3.6.1 步骤

选择一个值**保持时间**。

B.3.6.2 选择依据

如果**保持时间**大于**最大BPDU传输延迟**，然后**最大桥接协议消息传播时间**在最坏的情况下，将延迟**保持时间**每座桥的延误**最大BPDU传输延迟**。这将使上文 B.3.5 中的结论无效。因此，选择以下内容：

$$hold\_t = 普杜 \quad (8)$$

B.3.6.3 IEEE 802 桥接局域网的推荐值

hold_t=1.0

B.3.7 最大年龄

B.3.7.1 步骤

计算下限**最大年龄**用于桥接 LAN。

B.3.7.2 选择依据

在稳定条件下（即没有发生故障、移除或插入网桥和其他 LAN 组件），桥接 LAN 外围的网桥不得使根超时。否则将导致暂时的本地拒绝服务。

这意味着**最大年龄**必须足以应对最坏情况的传播延迟和协议消息年龄不准确性，如下所示。

如果在任何时候，桥接器依赖于协议消息信息，而该信息的年龄被最大程度地高估，那么

a) 从根节点接收到的下一个协议消息的传输与当前正在使用的协议消息信息的原始传输之间的间隔，

b）高估当前信息的年龄，以及
c) 下一个要接收的协议消息的传播时间

必须小于最大年龄，否则网桥将使协议消息信息超时并尝试自己成为根。

$$\begin{aligned} 最大年龄 &= （（丢失的消息+1）\times 你好）+消息+（普杜\times（直径-1）） \\ &= 消息+消息属性 \end{aligned}$$

B.3.7.3 IEEE 802 桥接局域网的推荐值

最大年龄=20.0

B.3.8 转发延迟

B.3.8.1 步骤

计算**转发延迟**。

B.3.8.2 选择依据

当端口的转发延迟计时器到期并且网桥开始转发该端口上收到的帧时，必须确定系统中不再有任何在先前的活动拓扑上转发的帧。如果有，则存在重复帧的风险，或者，如果在建立新拓扑时旧活动拓扑的残余仍然存在，则存在创建数据循环的风险。

因此，前向延迟计时器运行的监听和学习周期必须涵盖以下连续周期：

a) 从第一个进入监听状态的桥接端口（并在后续的重新配置中保持该状态）到桥接 LAN 中的最后一个桥听到状态变化。**主动拓扑**。
b) 让最后一个网桥停止转发在前一个拓扑上接收到的帧，并让最后一个转发的帧消失。

在上述 a) 中，可能存在高达**最大消息年龄估计过高**在 Bridges 超时旧 Root 信息并准备成为或听取新 Root 的时间。在此之后，它可能需要**最大桥接协议消息传播时间**将新拓扑的消息从新的根传播到所有网桥。

对于上述 b)，停止转发的时间将是**最大传输停止延迟**，该延迟受最大桥接传输延迟（对于所有优先级）的限制 ； 随后，该帧将在帧生命周期内消失。

因此有以下内容：

$$2\times \mathit{转发} >= \mathit{消息} + \mathit{消息属性} + \mathit{BT\_D} + \mathit{生活} \quad (9)$$

B.3.8.3 IEEE 802 桥接局域网的推荐值 *转发*=15.0

B.4 参数范围的选择

B.4.1 绝对最大值

可能需要配置具有更大**最大介质访问延迟**比上述推荐值计算中假设的要多。这可能是 LAN 承载的流量类型或 Bridge 实现的特定方面（例如，旨在最大化吞吐量）的结果。但是，非常希望**最大桥接传输延迟**，**最大BPDU传输延迟**， 和**最大消息年龄增量估计过高**为了实现互操作性，必须遵守本标准。

使用这些参数的绝对最大值运行的网桥应能够使用桥接 LAN 中的推荐值进行配置**桥梁直径**至少为 3。以下情况满足此标准：

$$\mathit{BT\_D} <= 2.0$$
$$\mathit{普杜} <= 2.0$$
$$\mathit{msg\_aio} <= 2.0$$

这些限制被认为涵盖了比 B.3 中推荐值更大的参数值的要求。

B.4.2 保持时间

减少**保持时间**低于建议值**最大BPDU传输延迟**. 也不会在减少 Bridge 的带宽使用或处理能力方面达到任何目的，通过增加**保持时间**。因此，将该参数的值固定为常数是适当的：

hold_t=1.0

B.4.3 问候时间的范围

不要求**你好时间**小于**保持时间**. 同样，设置**你好时间**超过绝对最大值的两倍**最大BPDU传输延迟**。因此，我们选择了以下内容：

1.0 <=*你好*<=4.0

B.4.4 最大年龄和转发延迟的最大要求值

最大要求值**最大年龄**和**转发延迟**使用 B.3 中的方程式计算，参数值如下：

直径= 7
mma_d<=2.0
BT_D= 2.0
普杜= 2.0
msg_aio= 2.0
你好= 4.0
丢失的消息= 3

其结果如下：

最大年龄= 40.0
转发延迟= 30.0

尽管这些被认为是所需的最大值，但并不希望阻止使用更大的值。

B.4.5 最大年龄和转发延迟的最小值

最低现实价值**最大年龄**和**转发延迟**使用 B.3 中的方程式计算，参数值如下：

直径= 2
mma_d<= 0.5
BT_D= 0.5
普杜= 0.5
hold_t= 1.0

msg_aio= 1.0
你好= 1.0
丢失的消息= 3

其结果如下：

最大年龄<= 6.0
转发延迟= 4.0

建议 Bridge 实现不允许较低的**最大年龄**和**转发延迟**为了防止出现荒唐的设置。

B.4.6 最大年龄和转发延迟之间的关系

为了进一步防止参数设置不当影响生成树算法和协议的正确性，建议网桥加强以下关系：**最大年龄**和**转发延迟**按照 B.3.8 给出的方法确保

2× (*转发延迟*–1.0）> =*最大年龄*

附件 C

（规范性）

源路由透明桥接操作

C.1 概述

源路由透明 (SRT) 桥接器是一种 MAC 桥接器，当收到带有路由信息 (RII=1) 的帧时，它会执行源路由；当收到不带路由信息 (RII=0) 的帧时，它会执行透明桥接。本附件规定了 SRT 桥接器中源路由操作的协议。源路由帧格式和桥接器操作将促进终端站对帧的源路由。终端站的源路由操作在 ISO/IEC 8802-2 中规定。源路由提供以下内容：

a) 通过桥接 LAN 提供多条路由，从而提高资源利用率。
b) 通过桥接 LAN 对帧进行更好的单独控制。

C.1.1 范围

为了在桥接 LAN 上使用源路由实现数据处理设备的兼容互连，本 IEEE 标准规定了支持源路由的 MAC 桥接器的操作。为此，它

a) 定义了MAC桥在提供源路由方面的操作原则，包括对MAC服务的支持和保存以及服务质量的维护。

b) 指定与 SRT 桥相关的增强型 MAC 内部子层服务。
c) 增强桥梁内部操作的架构模型。
d) 建立桥接局域网中源路由功能的桥接管理要求，确定基本的管理对象和管理操作。

e) 规定了符合本标准的设备必须满足的要求。

未来研究的内容如下：

— 如何使用 ISO/IEC 15802-2 提供的协议和架构描述使远程管理器能够执行管理操作。

— 桥梁运行参数的性能要求、建议默认值和适用范围。

此外，SRT 操作仅针对已定义路由信息字段的 MAC 进行定义。截至本文发布时，它们包括以下内容：

— 令牌环（ISO/IEC 8802-5）。
— FDDI MAC（ISO 9314-2）。

与 MAC 相关的特定问题在 C.2.5 中讨论。

C.1.2 定义

C.1.2.1 ARE 路限值

此限制表示“所有路由探索器”框架中允许的最大路由描述符数量。此限制是在 Bridges 中实现的，以限制在路由确定过程中“所有路由探索器”框架在终端站之间穿越的 Bridges 数量。

C.1.2.2 资源管理器框架

SRT Bridge 在转发帧时会向其添加路由信息的帧。探索器帧用于发现路由。有一个“所有路由探索器”，旨在由每个端口转发；还有一个“生成树探索器”帧，仅由生成树协议指定转发它们的端口转发。

C.1.2.3 LAN 输入 ID (LIN)

SRT Bridge 从中接收帧的 LAN 的标识符。

C.1.2.4 LAN 输出 ID（LOUT）

SRT Bridge 将帧传输到的 LAN 的标识符。

C.1.2.5 并行桥

连接相同 LAN 的两个或多个网桥。

C.1.2.6 路线

通过一系列 LAN 和 SRT 桥的路径。

C.1.2.7 路线控制

源路由帧的路由信息 字段中的第一个 2 个八位字节字段。此字段指示要进行源路由的帧的类型和特性。

C.1.2.8 路由描述符

路由信息字段中的两个八位字节字段，用于指定 LAN ID 和桥接号。

C.1.2.9 路由发现

获取到达目的站的路线的过程。

C.1.2.10 路由信息

路由控制字段和由 LAN ID 和桥号组成的路由描述符序列，指示两个站之间通过网络的路由。

C.1.2.11 源路由

通过在帧中指定要遍历的路由来通过多 LAN 网络路由帧的桥接机制。

C.1.2.12 生成树

网桥拓扑，使得任意两个终端站之间有且仅有一条数据路由。

C.1.2.13 STE 路限制

此限制表示生成树探索器帧中允许的最大路由描述符数量。此限制是在网桥中实现的，以限制生成树探索器帧遍历的网桥数量。

C.1.2.14 透明桥接

一种对终端站透明的桥接机制，通过参与生成树算法来转发帧的网桥将 LAN 段互连。

C.1.2.15 缩写

在 IEEE 802 标准系列中，以下缩写是该标准所特有的：

是	所有路线探索器：RII=1，RT=10x
平均边际成本	ARE 路线描述符限制
国阵	桥梁号码
布隆迪	桥接端口对
林	LAN 输入 ID
输出端	LAN 输出 ID
肝硬化	长度字段
管理服务数据单元	MAC 服务数据单元非源路由：RII=0，
噪声反射率	无 RI 字段路由控制
钢筋混凝土	
研发	路线描述符
罗德岛州	路由信息
放射免疫分析	路由信息指示路由类型
逆转录	
弹性模量	特定路由帧：RII=1，RT=0xx 源路由透明
固体废物管理	（桥接）生成树资源管理器：RII=1，
科技教育	RT=11x STE 路由描述符限制
斯特林	

C.1.3 一致性

声称符合本附件的 MAC 桥接器

a) 应符合本标准的主体，并由本附件扩充，并代表本标准主体中描述的具有路由信息的帧的功能的超集。

b) 应符合C.1.3.1和C.1.3.2给出的静态和动态要求。

c) 此外，声称符合本标准的实施的供应商应填写附件 D 中提供的 PICS 形式表的副本，并提供识别供应商和实施所需的信息。

C.1.3.1 静态一致性要求

声称符合此标准的实现

a) 应符合其互连的局域网的MAC子层和物理层。
b) 应符合C.3中描述的SRT桥的静态特性。

C.1.3.2 动态一致性要求

声称符合此标准的实现

a) 应表现出与处理源路由帧的桥接操作相对应的外部行为，如C.3中所述。

b) 应按照C.3所述报告错误。

C.2 MAC服务支持

C.2.1 对MAC服务的支持

桥接 LAN 中的网桥支持为连接到桥接 LAN 的终端站提供的 MAC 服务，如 6.2 所述。

C.2.2 MAC 服务的保留

SRT 桥接器保留了 6.2 中描述的 MAC 服务。

C.2.3 服务质量维护

SRT Bridge 支持的 MAC 服务质量与单个 LAN 提供的服务质量相比并不差很多。6.3 中包含了完整的服务质量参数集。受源路由功能影响的参数如下：

a) 帧顺序错误
b) 帧复制
c) 未检测到的帧错误率
d) 最大服务数据单元大小

没有路由信息的帧称为非源路由帧 (NSR)。源路由引入了三种新类型的帧。具体来说，源路由帧 (SRF) 是通过源路由网络传输数据的主要机制。生成树探索器帧 (STE) 在由 SRT 桥互连的生成树部分上发送；它们在传输过程中收集生成树路由信息。STE 帧可用于管理和路由确定，但不是数据传输所必需的。所有路由探索器帧 (ARE) 都在路由发现期间明确使用。

C.2.3.1 帧乱序

源路由允许在给定的一对站点之间同时存在和使用多条路由。如果帧在不同的路由上交替发送，它们可能会以不同的顺序到达。但是，每种类型的任何特定路由都不会发生错误排序。接收相同类型帧（例如，SRF）的网桥将以先进先出 (FIFO) 的方式传输它们。关于特定类型的帧，

a) NSR 帧不会乱序（参见本标准的主体），因为它们遵循生成树的唯一路径。

b) STE 帧不会乱序（参见本标准的主体），因为它们遵循生成树的唯一路径。

c) ARE 帧用作控制帧。乱序不适用于这些帧，因为根据定义，它们通过多条路由发送。

d) 只要发送方使用一个且仅一个源路由来传递帧序列，SRF 就不会出现乱序；在同一对端口之间使用不同路由的帧容易出现帧乱序。

C.2.3.2 帧复制

a) NSR和STE帧不能重复（参见本标准正文）。
b) ARE 帧用作控制帧；接收端必须接受多个 ARE 帧（每个路由一个）。

c) SRF 不能被复制，因为帧仅按照路由信息字段中指定的 Bridge 顺序转发。此外，Bridge 可防止路由顺序中 LAN 号重复的帧被转发。

C.2.3.3 未检测的帧错误率

MAC 子层提供的服务在传输帧中引入了非常低的未检测帧错误率，参见 6.3.7。特别要注意以下几点：

a) NSR 和 SRF 的未检测错误率保持不变。
b) ARE 和 STE 帧在到达目的地时收集路由信息；因此，每个中间桥都会重新计算 FCS。这些帧可能具有更高的未检测帧错误率，但它们不适用于数据传输。

C.2.3.4 支持的最大服务数据单元大小

有关最大服务数据单元的完整描述，请参阅 6.3.8。SRT 桥接器满足与最大服务数据单元大小相关的所有要求。两个 LAN 之间的桥接器支持的最大服务数据单元大小是 LAN 所支持的最大服务数据单元大小中的较小者。桥接器不会尝试将帧中继到不支持该帧所传达的服务数据单元大小的 LAN。

源路由减少了网桥接收不可接受大小的帧，因为数据发送方可以在路由发现期间确定所选路由上数据单元的最大大小。

C.2.4 内部子层服务

除 6.4 节中指定的内部子层服务外，MAC 实体还通过一组新的原语提供额外的增强服务。

内部子层服务不包括操作仅限于单个 LAN 的程序。描述此服务的 unitdata 原语如下。除添加以下参数外，参数及其含义与 6.4 中的相同：

路由信息

C.2.4.1 交互

为MAC中继定义下列原语来请求MAC传输PDU。

M_UNITDATA.请求
M_UNITDATA.指示

C.2.4.2 详细服务规范

所有原语仅作为示例指定。每个服务都指定了特定的原语以及在 MAC 和 MAC 中继功能之间传递的所需信息。这里仅定义 6.4 中未包括的原语。

C.2.4.2.1 M_UNITDATA.指示

调用的每个 MAC 数据指示原语对应于从单个 LAN 接收一个帧。

服务原语的语义

M_UNITDATA.指示 （
帧类型，
mac_action，
目标地址，
源地址，
路由信息，
mac_服务_数据_单元，
用户优先级，
帧校验序列
）

路由信息。routing_information 参数指定接收帧的路由信息 字段。

生成时间。此原语由 MAC 实体生成，并发送给 MAC 中继功能，以指示要转发的帧已到达。只有当这些帧有效形成时（由相应的 MAC 定义），才会报告这些帧。

收讫效力。收到此原语后，MAC 中继功能将根据 3.7 中规定的转发逻辑来处理该帧。

C.2.4.2.2 M_UNITDATA.请求

调用数据请求原语将帧传输到单个 LAN。

服务原语的语义

M_UNITDATA.请求 （
帧类型，
mac_动作，
目标地址，
源地址，
路由信息，
mac_服务_数据_单元，

用户优先级，
访问优先级，
帧校验序列

路由信息。routing_information 参数指定由 Bridge 功能修改的要转发的帧的路由信息 字段。

注——如果提供了帧校验序列（FCS），则可以转发它而无需重新计算；否则，应该计算它。

生成时间。当MAC中继将帧传递给MAC实体进行转发时，MAC中继功能向MAC实体生成此原语。

收讫效力。收到此原语后，MAC 实体会附加所有适当的字段，并将正确格式的帧传递到协议的较低层，以便传输到对等的 MAC 实体。

C.2.5 内部子层服务支持

本节规定了内部子层服务到MAC协议和程序的映射，以及MAC帧中服务参数的编码。

C.2.5.1 令牌环的支持

令牌环访问方法在 ISO/IEC 8802-5 中指定。该标准的第 3 条指定了格式和设施，第 4 条指定了令牌环协议。

收到 M_UNITDATA.request 原语后，本地 MAC 实体使用下面指定的参数组成一个帧，将帧控制、目标地址、源地址、路由信息和帧校验序列字段附加到用户数据。路由信息指示位设置为 1，以指示路由信息字段的存在。帧被排队等待传输。在传输时，将添加起始分隔符、访问控制字段、结束分隔符和帧状态字段。

在收到有效帧时，其 frame_type 和 mac_action 参数值分别为 user_data_frame 和 request_with_no_response，且路由信息指示位设置为 1，则会生成 M_UNITDATA.indication 原语，其参数源自下文指定的帧字段。

注——在接收到不是由桥接端口的本地 MAC 实体传输的、且路由信息指示位设置为零的有效 MAC 帧时，将按照 6.5.3 的要求生成 M_UNITDATA.indication 原语，并由透明桥接功能处理该帧。

frame_type、mac_action 参数、destination_address、source_address、mac_service_data_unit、user_priority 和 frame_check_sequence 参数按照 6.5.3 中规定的方式进行编码。

路由信息指示位编码在源地址字段中。它在源地址字段中占据的位置与组地址指示位在目标地址字段中占据的位置相同。

如果 M_UNITDATA.request 原语没有附带 frame_check_sequence 参数，则按照 ISO/IEC 8802-5 的 3.2.7 进行计算。

A、C位的设置请参考6.5.3。

为了支持 MAC 内部子层服务，令牌环桥必须能够识别和删除其自身传输的帧，即使它们携带的源地址与传输它们的桥端口的源地址不同。

C.2.5.2 支持 FDDI

收到有效帧（ISO 9314-2）后，如果该帧不是由桥接端口的本地MAC实体传输的，则生成M_UNITDATA.indication原语。与该原语相关的参数来自6.5.4中规定的帧字段。

在接收到 M_UNITDATA.request 原语后，本地 MAC 实体使用 6.5.4 中指定提供的参数组成一个帧。

C.3 操作原理

本节使用 SRT Bridge 的原理和操作模型。它规定了以下内容：

a) 源路由桥操作的主要元素的解释和支持功能的列表

b) 管理这些功能提供的架构模型
c) SRT Bridge 运行模型，包括支持功能的流程和实体的运行

d) 有关附加寻址要求的详细信息

C.3.1 源路由桥操作

源路由操作的主要元素如下：

a) 源路由数据帧的中继
b) 中继和处理帧以支持路由信息的传输和获取
c) 上述事项的管理

C.3.1.1 数据帧的中继

MAC 桥接器在连接到其端口的桥接 LAN 的独立 MAC 之间中继各个 MAC 用户数据帧。在一个端口上接收并在另一个端口上传输的具有给定 user_priority 和 frame_type 的帧的顺序将保持不变。

支持源路由数据帧中继和维护Bridge支持的服务质量的功能如下：

a) 帧接收
b) 丢弃收到的错误帧
c) 源路由数据帧的选择
d) 如果可传输服务数据单元大小超出范围，则丢弃帧（C.2.3.4）
e) 路由信息的修改
f) 当超过最大桥接传输延迟时丢弃帧
g) 出站访问优先级选择
h) 帧传输

C.3.1.2 路由信息的传播

支持帧的中继和处理以支持路由信息的传输和获取的功能如下：

a) 处理所有路由探索器 (ARE) 帧
b) 处理生成树探索器 (STE) 帧

C.3.1.3 桥梁管理

支持桥梁管理控制和监控上述子条款中功能提供的功能在 C.4 中指定。

C.3.2 桥梁结构

每个桥接端口使用与该端口关联的单个 MAC 实体提供的服务接收和传输帧，如 7.2 中所述。MAC 中继实体处理桥接端口之间中继帧的 MAC 独立功能。如果接收的帧不是源路由 (RII = 0)，则使用第 7 条 (透明桥接逻辑) 中描述的逻辑转发或丢弃桥接帧。如果接收的帧是源路由 (RII = 1)，则使用 C.3.3 中描述的逻辑处理该帧（参见图 C-1）。

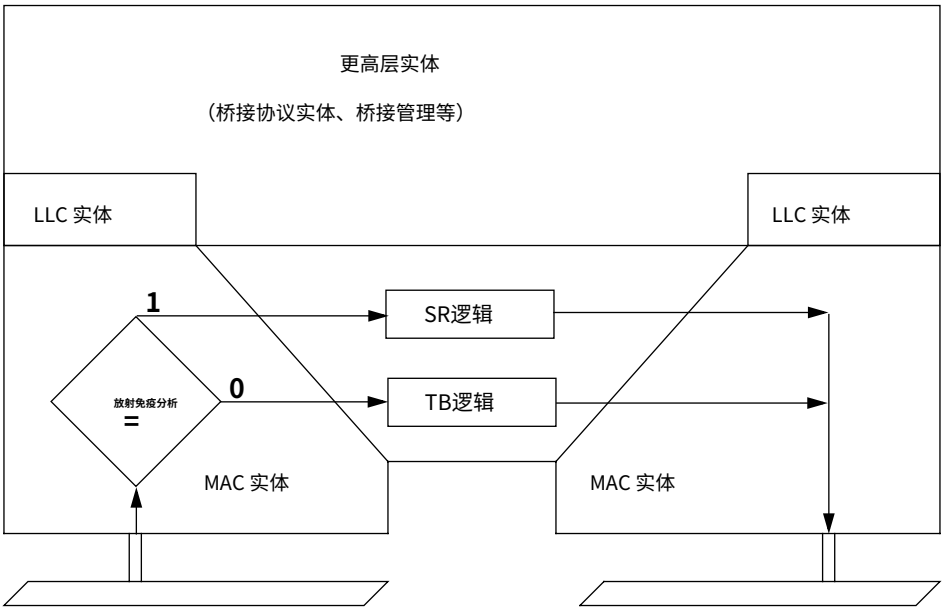


图 C-1—SRT Bridge 运行逻辑

在 SRT 桥接中，端口服务于特定 LAN，并且该 LAN 被分配一个唯一的 LAN_ID。由于 SRT 允许两个 LAN 之间存在多个桥接路径，因此每个桥接路径都分配有一个桥接编号 (BN)，因此相同两个 LAN 之间的任何两个桥接路径都必须具有不同的桥接编号。根据此分配，LAN 之间的每条路径都可以通过 LAN_ID 和桥接编号唯一标识。通过桥接的路径在 RI 字段中通过 LIN、BN、LOUT 指定，其中 LIN 是接收端口的 LAN_ID，LOUT 是转发端口的 LAN_ID。

因此，桥接中每对唯一端口之间存在桥接关系（桥接路径）。术语桥接端口对 (BPP) 将用于表示形成桥接路径的端口配对，并且每个

BPP 将由两个端口号及其各自的 BN 指定。现在可以通过分配给该端口对的 LAN_ID 和 BN 集来指定桥接路径。

每个端口均具有以下属性：

a) 端口号
b) LAN_ID
c) 最大MSDU
d) SRT 端口类型
e) RD 限制

采用这种桥接架构方法，只需按照 C.3.7 中的方法，在桥接端口对的基础上描述桥接帧处理即可。

C.3.3 桥梁运行

MAC 中继实体对与每个桥接端口相关联的各个 MAC 实体提供的内部子层服务的使用在 C.3.5 和 C.3.6 中指定。对没有路由信息 (RII=0) 的帧的桥接操作在 7.3 中描述。桥接协议实体保持不变。桥接管理实体得到增强，包括 C.4 中描述的 SR 管理对象。MAC 中继实体得到增强，可以处理源路由帧和非源路由帧。源路由转发过程根据帧中包含的信息和桥接端口的状态将接收到的源路由帧转发到其他桥接端口 (C.3.7)。透明桥接功能不处理源路由帧 (RII=1) 的源地址。这是因为无法将生成树端口与源路由帧上的地址关联，因为源路由路径并不总是与生成树路径一致。

C.3.3.1 源路由功能概述

对于源路由帧，转发决策基于帧主体中的路由信息 字段中包含的信息（如果存在该字段）。如果不存在，则转发决策基于帧的目标地址字段（由透明桥接逻辑决定）。

确定路由后，发送帧的站点通过在传输帧的路由信息（RI) 字段中嵌入路由描述来指定帧将要经过的路由。如果 MAC 类型相同，网桥会将这些帧从一个 LAN 段复制并转发到另一个 LAN 段，而不会更改帧的内容。如果 MAC 类型不同，则将更改帧的内容并重新生成 FCS。如果在路由信息被修改的帧中发送数据（例如，所有路由探索器帧），则将重新计算 FCS，并且用户数据完整性可能会受到损害。源路由提供了包含用户数据的帧像桥接 LAN 一样遍历的能力，而无需重新生成 FCS。

车站最初通过称为*路由发现*。此过程允许站点根据需要动态发现到目标站点的路由。转发路由发现（探索器）帧的每个网桥在 RI 字段中指示网桥的配置（LIN、BN、LOUT）和网桥可以支持的最大帧大小（通过特定的 LOUT），如果它小于已指定的大小，则重新生成 FCS。这样，当探索器帧从一个 LAN 段转发到另一个 LAN 段时，网桥就会构建路由信息。当帧到达目标站点时，RI 字段指示帧从源到目标的路径以及整个路径上支持的最大帧大小。

图 C-2 描述了源路由功能的元素。

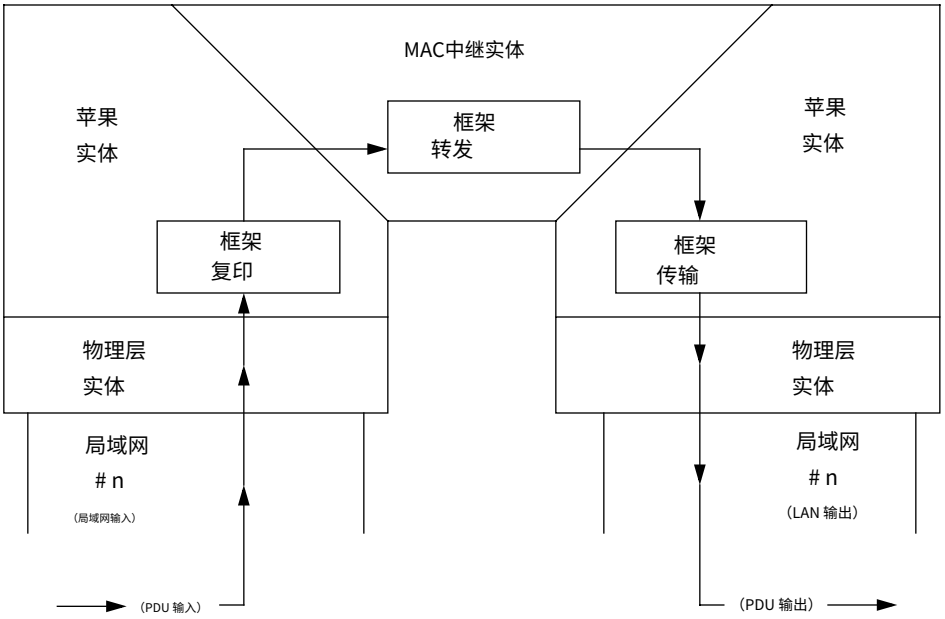
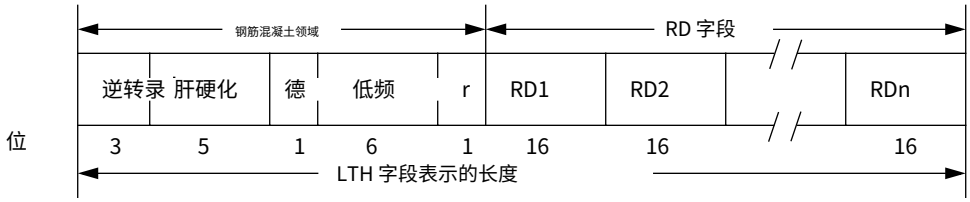


图 C-2 — 源路由桥的元素

C.3.3.2 源路由信息字段

路由信息字段的结构定义如图C-3所示。



RC = 路由控制（16 位） RDn = 路由描述符 n（16 位） RT = 路由类型（3 位）
r = 保留（1 位）

LTH = 长度（5 位） D = 方向位（1 位） LF = 最大帧（6 位）

图 C-3—路由信息字段的结构

RI 字段按照每个 MAC 处理数据的惯例在介质上传输。为了从时间上理解 RI 字段，图 C-4 相当于图 C-3 中的定义。

路由类型（RT）字段。此字段指示帧是沿着单条路由在网络中转发，还是通过多个互连的 LAN 的多条路由在网络中转发。

特定路由帧（RT=0XX）。如果最高有效 RT 位设置为 0，则 RD 字段包含通过网络的特定路由。在整个桥接 LAN 传输过程中，XX 位都会保留。

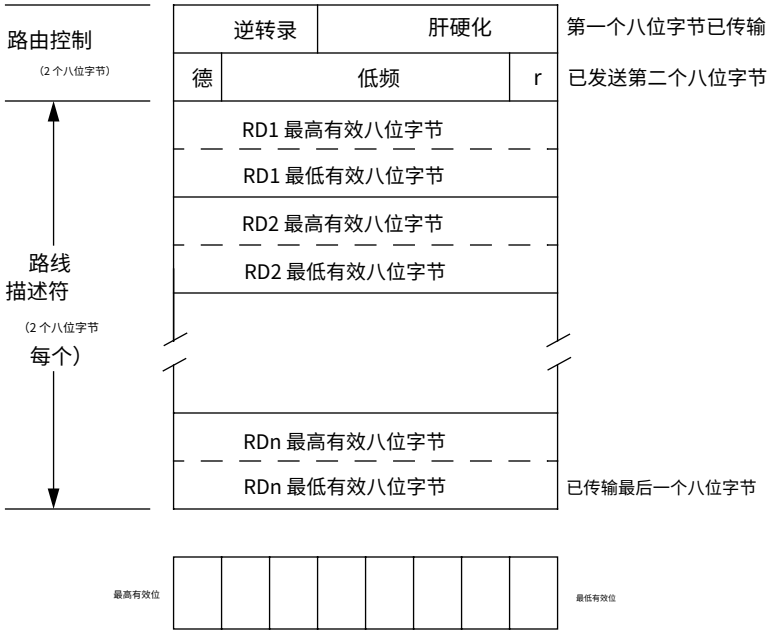


图 C-4 — 路由信息字段

所有路线探索框架（RT=10X）。如果 RT 位设置为 10X，表示所有路由探索器帧，则该帧将沿着 ARE 转发决策允许的网络中的每个路由进行路由。此类型将导致到达目标站的帧数量与从源到该站的不同路由数量一样多。它由没有路由描述符的终端站发起。SRT 桥接器在转发帧时将路由描述符添加到帧中。X 位在整个桥接 LAN 的传输中保留。

生成树探索器框架（RT=11X）。如果 RT 位设置为 11X，表示生成树资源管理器帧，则只有端口处于透明桥接转发状态的 SRT 桥接器才会将帧从一个 LAN 中继到另一个 LAN，结果该帧将沿着桥接 LAN 生成树转发，并且在网络中的每个 LAN 上出现不超过一次。它由没有路由描述符的终端站发起。SRT 桥接器在转发帧时将路由描述符添加到帧中。X 位在整个桥接 LAN 的传输中保留。

长度位 (LTH)。这五位表示 RI 字段的长度（以八位字节为单位）。长度字段值将是 2 到 30 之间的偶数值（含）。

方向位（D）。如果 D 位为 0，则帧将按照路由信息字段中指定的顺序遍历 LAN（RD1 到 RD2 到...到 RDn）。相反，如果 D 位设置为 1，则帧将以相反的顺序遍历 LAN（RDn 到 RDn-1 到...到 RD1）。D 位仅对 SRF 有意义。对于 STE 和 ARE 帧，D 位应为 0。

最大帧位 (LF)LF 位表示在特定路由上两个通信站之间传输的 MAC 服务数据单元 (MAC 信息字段) 的最大大小。LF 位仅对 STE 和 ARE 帧有意义。对于 SRF，LF 位将被忽略，并且网桥不得更改。发起探索者帧的站将 LF 位设置为其可以处理的最大帧大小。转发网桥应将 LF 位设置为指示不超过最小值的最大值

a) 收到的LF位指示值

b) Bridge 支持的最大 MSDU 大小
c) 接收帧的端口支持的最大 MSDU 大小
d) 传输帧的端口支持的最大 MSDU 大小

目标站可能会进一步降低 LF 值以指示其最大帧容量。基本 LF 位编码值和原理如图 C-6 所示。

LF 位编码由 3 位基本编码和 3 位扩展编码组成（见图 C-5）。SRT 桥应包含一个 LF 模式指示器，以允许选择基本或扩展 LF 位。当 LF 模式指示器设置为*基本模式*，桥接器应根据最大帧基值设置探索器帧中的 LF 位（见表 C-1）。当 LF 模式指示器设置为*扩展模式*，桥接器应根据最大帧扩展值设置探索器帧中的 LF 位（表 C-2）。

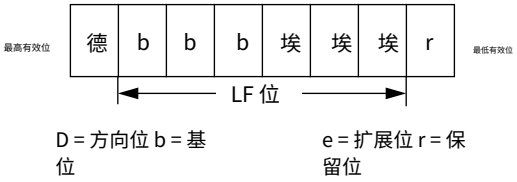


图 C-5 — RC 字段第二个八位字节上的基数和扩展 LF 位

000： 516 个八位字节（ISO 8473，无连接网络协议，加上 LLC）1 470 个
001： 八位字节（ISO/IEC 8802-3，CSMA/CD LAN）
010： 2 052 个八位字节（80 x 24 字符屏幕，带控件）
011： 4 399 个八位字节 (ISO/IEC 8802-5、FDDI、4 Mb/s 令牌环、ISO 9314-2) 8 130
100： 个八位字节 (ISO/IEC 8802-4 令牌总线 LAN)
101： 11 407 个八位字节（ISO/IEC 8802-5 4 位突发错误未受保护） 17
110： 749 个八位字节（ISO/IEC 8802-5 16 Mb/s 令牌环 LAN）
111： 41 600 个八位字节（扩展至 65 535 个八位字节的基础）

图 C-6—最大框架基值及其原理

表 C-1—最大框架基值

根据	价值	根据	价值
000	516 个八位字节	100	8 130 个八位字节
001	1 470 个八位字节	101	11 407 个八位字节
010	2 052 个八位字节	110	17 749 个八位字节
011	4 399 个八位字节	111	> 17 749 个八位字节

LF 值 = (blfv + elf 中的最大整数× (nlfv - blfv）/8）

在哪里

噴陣 = 低序 3 位表示的基本 LF 值 下一个基本 LF 值（111 后
北伐 = 的下一个值为 65 535） 扩展 LF 位编码（0 至 7）
精灵 =

表 C-2 — 最大帧扩展值

	扩大								
乙一个代埃		000	001	010	011	100	101	110	111
	000	516	635	754	873	993	1 112	1,231	1,350
	001	1,470	1,542	1,615	1,688	1,761	1,833	1,906	1 979
	010	2 052	2,345	2,638	2 932	3 225	3 518	3,812	4 105
	011	4,399	4,865	5,331	5,798	6 264	6 730	7 197	7,663
	100	8 130	8,539	8 949	9,358	9,768	10178	10,587	10,997
	101	11,407	12199	12,992	13,785	14,578	15,370	16 163	16,956
	110	17,749	20730	23 711	26,693	29,674	32,655	35,637	38,618
	111	41,600	44 591	47,583	50,575	53,567	56,559	59,551	> 59551

注意：桥接器不需要支持上述列表中的每个值。此外，具有特定最大帧大小的 MAC 上的源路由端系统不应发送 MAC 头和尾、RI 字段和信息超出该 MAC 的最大帧大小的帧。

一般而言，选择八位字节中的基值是为了允许通过各种 LAN MAC 方法传输最佳大小的帧。这些大小是通过从最大物理帧大小中减去最大大小的 MAC 报头和最大大小的路由信息 字段（30 个八位字节）来计算的（参见图 C-7）。与位编码相关的值表示 MAC 信息字段的最大长度（参见图 C-7）。扩展位用于提供每组基值之间的七个中间值。计算这些值的公式和实际计算值包含在表 C-2 中。

图 C-7—最大帧值的范围

路由描述符 (RD) 字段。图 C-8 描述了该字段。

图 C-8 — 路由描述符字段

RD 字段序列定义了通过网络的特定路由。每个 RD 包含一个网络唯一的 12 位 LAN ID 和一个 4 位 Bridge 编号，用于在两个或多个 Bridge 连接相同的两个 LAN（并行 Bridge）时区分它们。路由信息字段中的最后一个 Bridge 编号被保留并设置为零。

保留（r）位。所有保留（r）位在接收时都将被忽略并按接收原样传输。

C.3.3.3 源路由帧类型

网桥根据帧的路由类型 (RT) 来处理帧。要处理的帧的 RT 在帧的 RI 字段的 RT 子字段中指示。以下子条款定义了每种路由类型。

C.3.3.3.1 特定路由帧类型（RT=0XX）

RD 字段包含通过网络的特定路由。此类型用于数据帧从源到目的地的正常路由。

C.3.3.3.2 所有路由探索器帧类型（RT=10X）

此帧沿着网络中的所有非重复路由进行路由。这种类型的帧称为“所有路由探索者”帧，它将导致到达目标站的帧数量与到达该站的允许路由数量相同。网桥在转发帧时会将路由信息添加到帧中。

C.3.3.3.3 生成树探索器帧类型（RT=11X）

生成树探索器帧仅通过处于转发状态的桥接端口（即沿生成树）转发（由源路由逻辑）。结果是该帧在目标站只出现一次。桥接器将路由描述符添加到此帧，并且不会对目标地址执行过滤。这些帧遍历由 SRT 桥接器连接的生成树的相邻部分，在转发时获取路由。

C.3.3.4 源路由帧的桥接处理

桥接器的处理基于一个两阶段过程：帧接收和帧转发。这些的显式函数定义分别包含在 C.3.5 和 C.3.7 中。帧接收过程是指 MAC 实体向 MAC 中继实体指示帧。帧转发过程是指对已复制的帧进行内部检查、可能添加路由信息以及执行将帧转发到另一个 LAN 段的操作。

C.3.4 端口状态信息

端口状态在 8.4 中描述。

C.3.5 帧接收

如果帧没有路由信息（RII=0），那么桥接器应按照第 7 条所述处理该帧。

如果帧有路由信息（RII=1），则该帧应按C.3.7中指示的方式处理。此类帧的user_priority应按7.5.1中规定的方式重新生成。

C.3.6 帧传输

帧的传输如 7.6 中所述。

C.3.7 帧转发

转发帧的决定取决于 RI 字段的内容和桥接器的内部状态。给定 user_priority 和 frame_type 的帧应按照从 LAN 段复制的顺序转发。转发过程会转发要中继到其他桥接器端口的接收帧。非源路由帧 (RII=0) 应按照 7.7 中的规定转发。源路由帧 (RII=1) 的处理包含在以下子条款中。以下子条款描述了 SRT 桥接器对单个启用的桥接器端口对 (LIN 和 LOUT) 的操作。多端口桥接器包括多个桥接器端口对，并应按照本条款中描述的方式对每个端口对进行操作。不得通过禁用的 BPP 转发任何帧。

C.3.7.1 特定路由的数据帧（RT=0XX）

发现 RI 字段到桥接配置（LIN、BN、LOUT 组合）匹配的桥接器应转发此类帧而不改变 RI，前提是两个 MAC 实体使用相同的帧格式（例如，桥接器从令牌环到令牌环，ISO/IEC 8802-5）。如果帧的路由信息字段中出现多次 LOUT，则桥接器应丢弃该帧，并且如果已实现，则增加 DupLout 计数器。请注意，无论目标是个人地址还是组地址，SRF 都是根据状态表转发或丢弃的。表 C-3 解释如下：

（SRF1）。如果 SRT 桥接收到特定路由的帧，并且在路由信息字段中未找到 LIN-BN-LOUT 匹配，则该帧不得在 LOUT 指示的端口上转发。

（SRF2）。如果 SRT 桥接收到特定路由的帧并且 LIN-BN-LOUT 匹配，且路由信息字段中没有多次出现 LOUT，且长度字段为偶数，则应转发该帧，并且（如果已实施）SRFsForwarded 计数器应递增。

（SRF3）。如果 SRT 桥接收到特定路由的帧，并且 LIN-BN-LOUT 匹配但长度字段为零或奇数，则应丢弃该帧，并且如果实施，则应增加无效 RI 计数器。

（SRF4）。如果 SRT 桥接收到特定路由的帧并且 LIN-BN-LOUT 匹配但路由信息字段中出现多次 LOUT，则应丢弃该帧，并且如果实施，则应增加 DupLout 计数器。

表 C-3—特定路由帧转发状态表

参考。	健康）状况	行动
1型	LIN-BN-LOUT不匹配	不转发帧
超音速发动机	(LIN-BN-LOUT 匹配) & (LOUT 出现次数 = 1) & (RI_LTH = 偶数)	LOUT 增量上的前向帧 (SRFsForwarded)
3型	(LIN-BN-LOUT 匹配) & ((RI_LTH = 奇数) | (RI_LTH = 0))	丢弃帧 增量 (InvalidRi)
4号	(LIN-BN-LOUT 匹配) & (LOUT 出现次数 > 1)	丢弃帧 增量 (DupLout)

在上述所有情况下：RII = 1、RT = 0XX 且 BPP 状态 = 启用

定义：丢弃 = 不转发任何 LAN 输出 转发 = 传输 LAN
输出的帧

C.3.7.2 所有路线探索器帧（RT=10X）

为了限制不必要的流量，SRT 桥可以过滤多余或冗余的 ARE 帧。多端口 SRT 桥可以丢弃（过滤）以下 ARE 帧：

— 之前已经通过 Bridge。这可以通过在接收帧的 RI 字段中查找 LIN-BN-LOUT 组合来确定。

— 之前曾通过过连接到 Bridge 的任何 LAN。此条件比 a) 中描述的条件更严格。这可以通过检查接收帧的 RI 字段来确定，以确定 LAN ID 是否与连接到 Bridge 的任何 LAN（LIN 除外）匹配。

除以下任何一种情况外，所有 SRT 桥都应转发每个 LOUT 上的所有其他帧：

a) LOUT 已经存在于 RI 字段中。
b) RI 字段中的最后一个 LAN ID 与 LIN 不匹配。
c) 路线描述符的数量达到或超过了ARERdLimit。
d) RI字段长度为奇数。
e) RI 字段长度为零。
f) RI字段长度为4（可选——见表C-4）。
g) 方向位不为 0。

如果满足上述任一条件，则应丢弃该帧。如果满足条件 2，则 LanIdMismatch 计数器将递增。如果满足条件 4、5 或 7，则 Invalid RI 计数器将递增。

如果这些条件都不成立，那么

如果 LTH = 2，桥接器会将带有 LIN 的 RD 添加到 RI 字段，并且，
— Bridge 将其 Bridge 编号和 LOUT 编号添加到帧中。

它还会为添加的每个路由描述符将长度字段增加 2。在转发帧之前，如果桥端口对的传输容量小于收到的 LF 指示的大小，则桥会调整最大帧 (LF) 字段以反映桥的传输容量。桥的传输容量是以下两者中最小的

— LIN 的最大帧大小，
— LOUT 的最大帧大小，以及
— Bridge 本身可以处理的最大帧大小。

最后，重新计算FCS。ARE帧的处理如表C-4所示。

表C-4解释如下：

（ARE1）。如果 SRT 桥接收到所有路由探索器帧，并且其 RI 字段中已经有 LAN-out ID，或者桥已过滤该帧，则不得在 LAN-out ID 指示的 LAN-out 上转发该帧。

（ARE2）。如果 SRT Bridge 收到所有路由探索器帧，并且其最后一个 LAN ID 与 LIN 不匹配，则应丢弃该帧，并且（如果已实施）应增加 LanIdMismatch 计数器。

（ARE3）。如果 SRT 桥接收到所有路由探索器帧，并且其 RD 数量达到或超过 LOUT 指示的端口的 AreRdLimit，则不得转发该帧，并且（如果已实施）应增加 LOUT 指示的端口的 AreRdLmtExceeded 计数器。

表 C-4 - 所有路由资源管理器帧转发状态表

参考。	健康）状况	行动
ARE1	（LOUT 已在 RI）|（桥已过滤）	请勿在 LOUT 上转发
ARE2	RI 中的最后一个 LAN ID = LIN	丢弃帧 增加（LanIdMismatch）
ARE3	# RD >= AreRdLimit	请勿转发 增量（ARERdLmtExceeded）
ARE4	(RI_LTH = 奇数)	(RI_LTH = 0)	(D ¬ = 0)	丢弃帧 增量（InvalidRi）
ARE5	(RI_LTH = 2) & (桥未过滤) & (AreRdLimit > 1)	将 LIN、BN、LOUT 添加到 RI 字段 SET RI_LTH = 6 SET LF=MIN(LF,LIN,BR,LOUT) 重新计算FCS 前向帧 增量（AREFramesForwarded）
ARE6	(RI_LTH = 4) &（桥接未过滤）&（RI 中的最后一个 LAN ID = LIN）&（AreRdLimit > 1）	丢弃帧 增量（InvalidRi） 或（可选） 将 BN,LOUT 添加到 RI 字段 RI_LTH = RI_LTH + 2 SET LF = MIN(LF,LIN,BR,LOUT) 重新计算FCS 前向帧 增量（AREFramesFowarded）
ARE7	(RI_LTH = 6 至 28 之间的偶数（含）) & (LOUT 尚未在 RI 中) &（桥接器未过滤）&（RI 中的最后一个 LAN ID = LIN）&（RD 数量 < AreRdLimit）	将 BN,LOUT 添加到 RI 字段 RI_LTH = RI_LTH + 2 SET LF = MIN(LF,LIN,BR,LOUT 重新计算FCS 前向帧 增量（AREFramesFowarded）

在上述所有情况下：RII = 1、RT = 10X 且 BPP 状态 = 启用

定义：丢弃 = 不转发任何 LAN 输出 转发 = 传输 LAN 输出的帧

（ARE4）。如果 SRT 桥接收到所有路由探索器帧，并且其长度字段为零或奇数，或者方向位不为 0，则应丢弃该帧，并且（如果已实施）应增加 InvalidRi 计数器。

（ARE5）。如果 SRT Bridge 收到有效的 All Routes Explorer，其 Length 字段等于 2，并且 AreRdLimit 大于 1，则 Bridge 应修改正在转发的帧，方法是添加其 LAN-in ID（LIN）、其 Bridge 编号（BN）、其 LAN-out ID（LOUT）和 4 位为 0，将 LTH 字段设置为 6，将 RI 的 LF 字段设置为接收到的 LF 和 Bridge 的传输容量中的最小值，并重新计算 FCS，然后将帧排队以在其 LAN-out 上传输，并且（如果实施）AREFramesForwarded 计数器应递增。

（ARE6）。如果 SRT Bridge 收到有效的 All Routes Explorer 帧（RI 长度 = 4），并且 AreRd-Limit 大于 1，则它应该

丢弃该帧并且（如果已实施）增加 InvalidRi 计数器；或者
通过将前一个 RD 的最后 4 位设置为其桥编号 (BN) 来修改正在转发的帧，
添加下一个由LAN-out（LOUT）和剩余4位组成的两个八位字节RD为0，增加

将长度乘以 2，将最大帧字段设置为接收到的 LF 与桥接器的传输容量中的最小值，重新计算 FCS，将帧排队以在其 LAN-out 上传输，并且（如果实施）增加 AREFramesForwarded 计数器。

（ARE7）。如果 SRT 桥接器收到有效的所有路由探索器，其长度字段为 6 到 28（含，仅偶数），并且 RD 的数量小于 AreRdLimit，则桥接器应通过将前一个 RD 的最后 4 位设置为其桥接器编号 (BN)、添加由 LAN-out（LOUT）和剩余 4 位组成的下一个两个八位字节 RD 为 0、将长度增加 2、将最大帧字段设置为接收到的 LF 和桥接器的传输容量中的最小值、重新计算 FCS、将帧排队以在其 LAN-out 上传输，并且（如果实施）增加 ARE-FramesForwarded 计数器。

C.3.7.3 生成树探索器（STE）帧

转发 STE 帧的条件与 ARE 帧的条件相同，但有以下例外：

a) 由于生成树，不需要探索过滤。
b) 转发还取决于 LAN 输出端口的状态和生成树的状态。

不应过滤 STE 帧的目标地址。这些帧应遍历整个生成树。STE 帧的处理方式如表 C-5 所示。

表C-5解释如下：

（STE1）。如果 SRT 桥接收到生成树探测器帧，并且 LAN 输入端口和 LAN 输出端口处于转发状态，并且 LOUT 已在 RI 字段中，则应丢弃该帧，并且如果已实施，桥应增加 DuplicateLanIdOrTreeError 计数器。

（STE2）。如果 SRT 桥接收到生成树探测器帧，且 LAN 输入端口处于转发状态，且最后一个 LAN ID 与 LIN 不匹配，则应丢弃该帧，并且如果已实施，则应增加 LanId-Mismatch 计数器。

（STE3）。如果 SRT 桥接收到生成树探测器帧，且 LAN 输入端口和 LAN 输出端口处于转发状态，并且 RD 的数量达到或超过 LOUT 指示的端口的 SteRdLimit，则该帧不得在 LOUT 上转发，并且（如果已实施）LOUT 指示的端口的 SteRdLimit Exceeded 计数器应递增。

（STE4）。如果 SRT 桥接收到生成树探测器帧，且 LAN-in 端口处于转发状态，且长度字段为零或奇数或方向位不为 0，则应丢弃该帧，并且（如果已实施）应增加 InvalidRi 计数器。

（STE5）。如果具有处于转发状态的 LAN 输入和 LAN 输出端口的 SRT 桥接器收到长度字段等于 2 的有效生成树资源管理器，并且 SteRdLimit 大于 1，则 SRT 桥接器应通过添加其 LAN 输入 ID (LIN)、其桥接器编号 (BN)、其 LAN 输出 ID (LOUT) 和 4 位作为 0 来修改正在转发的帧。SRT 桥接器还应将长度字段设置为 6，将 RI 的最大帧字段设置为接收到的 LF 和桥接器的传输容量中的最小值，并重新计算 FCS。然后，SRT 桥接器应将帧排队以在其 LAN 输出上传输，并且（如果已实施）增加 STEFramesForwarded 计数器。

（STE6）。如果具有处于转发状态的 LAN 输入和 LAN 输出端口的 SRT 网桥收到长度字段为 4 的有效生成树资源管理器，并且 SteRdLimit 大于 1，则网桥应

a) 丢弃该帧，并且如果已实施，则增加 InvalidRi 计数器；或者

表 C-5 — 生成树浏览器帧转发状态表

参考。	港口状态 林露		健康）状况	行动
STE1	F	F	（LOUT 已进入罗德岛州）	丢弃帧， 增量（DupLanIdOrTreeError）
STE2	F	—	RI 中的最后一个 LAN ID = LIN	丢弃帧 增加（LanIdMismatch）
STE3	F	F	# RD >= SteRdLimit	丢弃帧增量增量 （SteRdLmtExceeded）
STE4	F	—	（D ¬ = 0）｜（RI_LTH = 0）｜（RI_LTH=奇数）	请勿转发 增量（InvalidRi）
STE5	F	F	（RI_LTH = 2）& （SteRdLimit > 1）	将 LIN、BN、LOUT 添加到 RI 字 段 SET RI_LTH = 6 SET LF = MIN_(LF,LIN,BR,LOUT) 重 新计算FCS 前向帧 增量（STEFramesForwarded）
STE6	F	F	（RI_LTH = 4）& （RI 中的最后一个 LAN ID = LIN）& （SteRdLimit > 1）	丢弃帧 增量 (InvalidRI) 或（可选） 将 BN、LOUT 添加到 RI 字段 RI_LTH = RI_LTH + 2 SET LF = MIN(LF,LIN,BR,LOUT) 重 新计算FCS 前向帧 增量（STEFramesFowarded）
STE7	F	F	（RI_LTH=6 至 28 之间的偶数（含）） & （LOUT 尚未在 RI 中）&（RI 中的最后一个 LAN ID = LIN）& （RD 数量 < SteRdLimit）	将 BN,LOUT 添加到 RI 字段 RI_LTH = RI_LTH + 2 SET LF = MIN(LF,LIN,BR,LOUT) 重 新计算FCS 前向帧 增量（STEFramesFowarded）
STE8	¬F	—		请勿在 LOUT 上转发
STE9	F	¬F		请勿在 LOUT 上转发

在上述所有情况下：RII = 1、RT = 11X 且 BPP 状态 = 启用
定义：
丢弃 = 不通过任何 LAN-out 转发 转发 = 通
过 LAN-out 传输帧 F
= 转发端口状态 = 不转发端
¬F 口状态
— = 任何港口国

b) 通过将前一个 RD 的最后 4 位设置为其桥号 (BN)，添加由 LAN-out (LOUT) 和剩余 4 位组成的下一个两八位字节 RD（0）来修改正在转发的帧。
SRT 桥还应将长度增加 2，将最大帧字段设置为收到的 LF 和桥的传输容量中的最小值，并重新计算 FCS。然后，SRT 桥应将帧排队以在其 LAN 输出上传输，并（如果已实施）增加 STE-FramesForwarded 计数器。

（STE7）。如果具有处于转发状态的 LAN 输入和 LAN 输出端口的 SRT 桥接器收到长度字段为 6 到 28（含，仅偶数）的有效生成树探测器，并且 RD 的数量小于 SteRdLimit，则 SRT 桥接器应通过设置最后 4 位来修改正在转发的帧

将前一个 RD 添加到其桥号 (BN)，添加由 LAN-out (LOUT) 和剩余 4 位组成的下一个两个八位字节 RD，并将其余 4 位设为 0。SRT 桥还应将长度增加 2，将最大帧字段设置为接收到的 LF 和桥的传输容量中的最小值，并重新计算 FCS。然后，SRT 桥应将帧排队以在其 LAN-out 上传输，并（如果已实施）增加 STEFramesForwarded 计数器。

（STE8）。如果端口处于转发状态，SRT 桥接器将仅转发在端口上接收的生成树探索器帧。

（STE9）。如果端口处于转发状态，SRT 桥接器将仅转发端口上的生成树探索器帧。

C.3.7.4 重复桥号测试

桥接管理可以使用下面的方法检查分配了与其桥接号相同的编号的并行桥接。

对于每两个由 Bridge 互连的 LAN，Bridge 管理都会启动 LLC TEST PDU 的传输，其目标地址设置为连接到一个 LAN 的端口的 MAC 地址，源地址设置为连接到另一个 LAN 的端口的 MAC 地址。TEST PDU 应包含由相对于源地址的 LAN-in、Bridge 编号和相对于源地址的 LAN-out 组成的路由。如果 Bridge 管理从此 TEST 收到多个响应，则存在一个被分配相同 Bridge 编号的并行 Bridge，并且应通知网络管理。参见图 C-9。

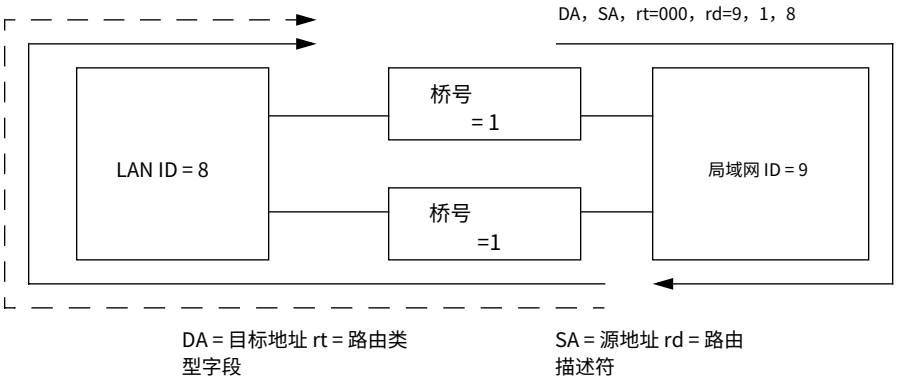


图 C-9—重复桥号测试图示

C.3.7.5 排队帧

转发过程为排队的帧提供存储，等待机会将这些帧提交给与每个桥接端口相关联的各个 MAC 实体进行传输。在同一桥接端口上接收的帧的顺序应保留

a) 对于给定的目的地址和源地址组合，具有给定用户优先级的单播帧；

b) 对于给定的目的地址，具有给定用户优先级的多播帧。

转发过程可能为给定的桥接端口提供多个传输队列。使用流量类别表（该表是与每个端口相关联的状态信息的一部分），根据帧的用户优先级将帧分配到存储队列。该表指示，对于每个可能的 user_priority 值，

应分配的流量类别的对应值。优先级值 0 表示 user_priority 的最低值，7 表示最高值。队列与流量类别一一对应。

出于管理目的，流量类别表最多支持八个流量类别，以便为每个 user_priority 级别提供单独的队列。流量类别编号为 0 到 N-1，其中 N 是与给定出站端口关联的流量类别数。流量类别信息的管理是可选的。流量类别 0 对应于非加速流量；非零流量类别是加速流量类别。

注意：在给定的桥接器中，允许为每个端口实现不同数量的流量类别。与支持单一传输优先级的 MAC 方法（如 CSMA/CD）关联的端口可以支持多个流量类别。

当转发进程不支持给定端口的加速流量类别时，换句话说，当端口只有一个流量类别时，所有 user_priority 值都会映射到流量类别 0。在支持加速流量的网桥中，对于已实现的流量类别数量，user_priority 到流量类别的默认映射如表 7-2 所示。

在提交给该端口的单个 MAC 实体时，转发过程排队等待在端口上传输的帧应从该队列中删除；即使已知传输失败，也不应再尝试在该端口上传输该帧。

如果已超出保证缓冲的时间，则转发过程排队等待在端口上传输的帧将从该队列中删除，并且不再进行传输。

如果有必要确保在随后传输该帧时不会超过最大桥接传输延迟（参见 6.3.6），则应从端口上排队等待传输的帧，并且随后不再提交给该端口的单个 MAC 实体。

如果相关端口离开转发状态，则在端口上排队等待传输的 STE 帧将从该队列中删除。

从任何特定端口的传输队列中移除一个帧本身并不意味着也应从任何其他端口的传输队列中移除该帧。

C.3.7.6 选择要传输的帧

应按照7.7.4的规定选择帧进行传输。

C.3.7.7 优先级映射

源路由转发过程根据 7.7.5 确定用于中继帧的 user_priority 和 access_priority 参数的值。

C.3.8 寻址

在源路由桥接 LAN 中，每个 LAN 和每个桥接器应分配一个编号，如下列小节所示。

C.3.8.1 局域网标识

多 LAN 网络中的每个单独 LAN 都应分配一个唯一的 12 位非零 LAN ID，所有连接到给定 LAN 的网桥都应一致了解该 ID。

C.3.8.2 桥梁编号

每个桥接器应有一个桥接器编号 (BN)。路由描述符序列 (LIN-BN-LOUT) 对于桥接器连接的两个 LAN 之间的每条路径应是唯一的。

C.3.8.3 路由描述符

路由描述符是一个两个八位字节的字段（16 位）。前 12 位表示 LAN ID，后面 4 位表示桥接编号（有关帧格式的详细信息，请参阅 C.3.3.2）。路由描述符序列定义了通过桥接 LAN 的唯一路由。由于路由的末端是 LAN 而不是桥接，因此路由信息字段中最后一个路由描述符的单个桥接部分被保留，并应设置为零。C.3.7 描述了使用源路由创建、构建和转发帧的方法。

C.4 桥梁管理

以下子条款对第 14 章中定义的管理功能进行了增强。这些增强包括通过桥接 LAN 控制源路由路径的能力。

C.4.1 桥梁管理实体

对 14.4 中定义的桥梁管理实体进行了以下增强。

C.4.1.1 桥接配置

C.4.1.1.1 发现桥

为了使 14.4.1.1 中定义的 Discover Bridge 管理操作能够报告 Bridge 中的 SRT 功能，后面的参数被添加到操作的输出中。

C.4.1.1.1.1 目的

表明桥梁的SRT传输能力。

C.4.1.1.1.2 附加输入

没有任何。

C.4.1.1.1.3 附加输出（14.4.1.1.3）

— SRT 传输容量 — 一个值字段，指示 SRT 桥 MAC 中继实体可以处理的帧的最大 MSDU 字段。如果桥不支持 SRT，则应将该值报告为零。

C.4.1.1.2 读桥

为了使 14.4.1.2 中定义的读取桥管理操作能够报告桥中的 SRT 功能，后面的参数被添加到操作的输出中。

C.4.1.1.2.1 目的

表明桥梁的SRT传输能力。

C.4.1.1.2.2 附加输入

没有任何。

C.4.1.1.2.3 附加输出（14.4.1.2.3）

— SRT 传输容量 — 一个值字段，指示 SRT 桥 MAC 中继实体可以处理的帧的最大 MSDU 字段。如果桥不支持 SRT，则应将该值报告为零。

C.4.2 转发过程

对 14.6 中定义的转发过程进行了以下增强。

C.4.2.1 端口计数器

参考 14.6.1。

C.4.2.1.1 读取转发端口计数器

C.4.2.1.1.1 目的

SRT 桥的端口计数器对象由 14.6.1.1 中定义的计数器以及以下子条款中的 SRT 特定计数器组成。

C.4.2.1.1.2 附加输入

没有任何。

C.4.2.1.1.3 附加输出

下列端口计数器被添加到14.6.1.1.3定义的输出列表中。如果计数器不受支持，则应将值报告为零。

h) InvalidRI——由于格式错误（即奇数 RI 长度或 0 RI 长度）而丢弃的帧数。

i) DupLout——由于 SRF 上的重复 LOUT 而丢弃的帧数。

j) LanIdMismatch——由于路由信息字段中的最后一个 LAN ID 不等于 LAN-in ID 而被丢弃的 ARE 和 STE 帧的数量。

k) DupLanIdOrTreeError——由于 LAN-out ID 已经存在于路由信息字段中而被丢弃的 STE 帧的数量。

l) STERDlimitExceeded——由于超出 STERD 限制而丢弃的 STE 帧的数量。

m) ARERDlimitExceeded——由于超出 ARERD 限制而丢弃的 ARE 帧的数量。

n) 转发的 SRF——从此端口转发到桥上另一个端口的 SRF 入站数量。

o) STEFramesForwarded——从此端口转发到桥接器上另一个端口的 STE 帧的入站数量。

p) AREFramesForwarded——从此端口转发到桥上另一个端口的 ARE 帧的入站数量。

C.4.3 SRT桥管理实体

C.4.3.1 SRT 桥接配置

SRT 桥管理实体的 SRT 桥配置操作允许管理读取桥 RD 限制、设置桥 RD 限制、读取桥 LF 模式和设置桥 LF 模式。

C.4.3.1.1 读取 LF 模式

C.4.3.1.1.1 目的

报告桥接器在 ARE 和 STE 帧中设置 LF 位所使用的模式。

C.4.3.1.1.2 输入

没有任何。

C.4.3.1.1.3 输出

a) LF 模式——一个值字段，指示桥接器是否使用基本模式还是扩展模式来设置 LF 位。

C.4.3.1.2 设置 LF 模式

C.4.3.1.2.1 目的

设置桥接器将用于设置 ARE 和 STE 帧中的 LF 位的模式。

C.4.3.1.2.2 输入

a) LF 模式——一个值字段，指示 Bridge 将使用基本模式还是扩展模式进行 LF 编码

C.4.3.1.2.3 输出

没有任何。

C.4.3.2 SRT 端口配置

SRT 桥管理实体的 SRT 端口配置操作允许管理读取和设置每个端口的 SRT 属性（LAN ID 和最大 MSDU 大小）。与 14.8.2 中定义的操作类似，以下子条款中的操作可在 SRT 端口上执行。

C.4.3.2.1 读取端口参数

C.4.3.2.1.1 目的

获取有关 SRT 桥接协议实体内特定端口的信息。

C.4.3.2.1.2 输入

a) 端口号

C.4.3.2.1.3 输出

a) SRT 端口类型——表示可用端口类型的值字段。类型包括 TB 和 SRT。
b) LAN ID——一个值字段，指示与端口所连接的 LAN 对应的 LAN ID。

c) 最大 MSDU 大小——一个值字段，指示此端口可以处理的帧的最大 MSDU。

d) ARE RD 限制——一个值字段，指示桥转发的 ARE 帧允许包含的路由描述符 (RD) 的最大数量。

e) STE RD 限制——一个值字段，指示桥转发的 STE 帧允许包含的路由描述符 (RD) 的最大数量。

C.4.3.2.2 设置 LAN ID

C.4.3.2.2.1 目的

将 LAN ID 与特定的桥接端口关联。

C.4.3.2.2.2 输入

a) 端口号——桥接端口的编号。
b) Lan ID——该Port所插入的LAN对应的Lan ID，由12位组成。

C.4.3.2.2.3 输出

没有任何。

C.4.3.2.3 设置最大 MSDU 大小

C.4.3.2.3.1 目的

将最大的 MSDU 大小与桥接端口关联。

C.4.3.2.3.2 输入

a) 端口号——桥接端口的编号。
b) 最大 MSDU 大小——指定此端口可处理的最大 MSDU 大小的值字段。

C.4.3.2.3.3 输出

没有任何。

C.4.3.2.4 设置 RD 限制

C.4.3.2.4.1 目的

将可由读取 RD 限制操作读取的路由描述符限制与桥转发的 ARE 和 STE 帧关联起来。

C.4.3.2.4.2 输入

a) 端口号——桥接端口的编号。

b) RD 限制类型——一个值字段，指示限制输入是否与 ARE 或 STE 帧相关联。

c) RD 限制值 — 一个值字段，用于指定 RD 限制类型所指定的帧类型的 RI 字段中可以包含的最大 RD 数量。此字段的取值范围为 0 至 14。

C.4.3.2.4.3 输出

没有任何。

C.4.4 SRT 桥接端口对数据库

以下操作可以在 SRT Bridge 上执行。

C.4.4.1 SRT 桥接端口对配置

C.4.4.1.1 读取 SRT 桥接端口对数据库大小

C.4.4.1.1.1 目的

读取桥接端口对数据库的大小。

C.4.4.1.1.2 输入

没有任何。

C.4.4.1.1.3 输出

a) 数据库大小——指示桥接端口对数据库中条目数量的值字段。

C.4.4.1.2 读取 SRT 桥端口对数据库条目

C.4.4.1.2.1 目的

读取 SRT 桥接端口对数据库实体的属性。

C.4.4.1.2.2 输入

a) 低端口号
b) 高端口号

C.4.4.1.2.3 输出

a) 源路由桥号
b) 桥接状态（启用或禁用）

C.4.4.1.3 设置 SRT 桥接端口对数据库条目

C.4.4.1.3.1 目的

设置 SRT 端口对数据库条目的属性。

C.4.4.1.3.2 输入

a) 低端口号
b) 高端口号
c) 源路由桥号
d) 桥接状态（启用或禁用）

C.4.4.1.3.3 输出

没有任何。

C.5 管理协议

此事有待进一步研究和解决。

附件 D

（规范性）

源路由透明桥操作的 PICS 形式[1]

D.1 简介

声称符合本标准附件 C 的协议实现的供应商除了应完成附件 A 中定义的 PICS 形式外，还应完成以下 PICS 形式。A.1 至 A.3 的规定适用于本附件中定义的 PICS 形式。

D.2 帧的中继和过滤

物品	特征	地位	参考	支持
（2a）	给定用户优先级的中继帧的顺序是否保留?	米	C.2.3.1	是的 [ ]
（2b）	STE、ARE 和 SRF 帧是否只提交给 MAC 实体进行传输一次?	米	C.2.3.2	是的 [ ]
(2c)	桥接器设置的最大帧字段是否如 C.3.3.2 中所述?	米	C.3.3.2	是的 [ ]
(2天)	接收到的没有路由信息 (NSR) 的帧是否按照 IEEE 标准 802.1D-1998 中所述进行处理?	米	C.3.5	是的 [ ]
(2e)	收到的特定路由帧 (SRF) 是否按照 C.3.7.1 中描述的方式进行处理?	米	C.3.7.1	是的 [ ]
（2f）	收到的所有路由探索器（ARE）帧是否按照 C.3.7.2 中描述的方式进行处理?	米	C.3.7.2	是的 [ ]
（2克）	该实现是否支持选项 1，允许过滤之前已经通过 Bridge 的帧?	哦	C.3.7.2	是的 [ ] 不 [ ]
(2小时)	该实现是否支持选项 2，允许过滤之前已经通过 Bridge 的帧?	哦	C.3.7.2	是的 [ ] 不 [ ]
(2一)	收到的生成树探测器 (STE) 帧是否按照 C.3.7.3 中描述的方式进行处理?	米	C.3.7.3	是的 [ ]
（2j）	实现是否按照 C.3.7.2 和 C.3.7.3 中描述的方式转发 LTH = 4 的探索器帧（STE、ARE）？	哦	C.3.7.2, C.3.7.3	是的 [ ] 不 [ ]

[1] *PICS 形式的版权发布：* 本标准的用户可以自由复制本附件中的 PICS 形式，以便将其用于预期目的，并可进一步发布完整的 PICS。

D.3 桥号和 LAN ID

物品	特征	地位	参考	支持
（3a）	是否可以在 Bridge 中配置所连接 LAN 的 LAN ID?	米	C.3.8.1	是的 [ ]
（3b）	可以为桥梁分配桥梁编号吗?	米	C.3.8.2	是的 [ ]
(3c)	桥梁管理是否可以使用重复桥梁号码测试来检查分配了相同号码的并行桥梁?	哦	C.3.7.4	是的 [ ] 不 [ ]

D.4 桥梁管理

此事还有待进一步研究和解决。

物品	特征	地位	参考	支持
(4a)	桥梁管理运营	哦	C.4	是的 [ ] 不 [ ]
(4b)	读取 LF 模式。	米	C.4.3.1.1	是的 [ ]
(4c)	设置 LF 模式。	米	C.4.3.1.2	是的 [ ]
(4日)	读取端口参数。	米	C.4.3.2.1	是的 [ ]
(4e)	设置 ARE RD 限制。	米	C.4.3.2.4	是的 [ ]
（4f）	指定此实现中 ARE RD Limit 的最大值。	米	C.4.3.2.4	一个___
（4克）	设置 STE RD 限制。	米	C.4.3.2.4	是的 [ ]
(4小时)	指定此实现中 STE RD Limit 的最大值。	米	C.4.3.2.4	一个___
(4一)	设置 LAN Id。	米	C.4.3.2.2	是的 [ ]
（4日）	设置最大 MSDU 大小。	米	C.4.3.2.3	是的 [ ]
（4千）	读取 SRT 桥接端口对数据库大小。	米	C.4.4.1.1	是的 [ ]
（4升）	读取 SRT 桥端口对数据库实体。	米	C.4.4.1.2	是的 [ ]
(4分钟)	设置 SRT 桥接端口对数据库实体。	米	C.4.4.1.3	是的 [ ]

附件 E

（规范性）

对象标识符值的分配

E.1 简介

本附件包含此标准在本次修订和之前修订中分配的所有对象标识符值的摘要。

每个表都显示与特定类别的信息对象相关的分配。表的标题标识信息对象的类别，并显示分配给表中条目的对象标识符值的不变部分。标记为 Arc 的列显示分配给不变部分之后的弧的值，该值完成了分配的对象标识符值。标记为 Purpose 的列包含信息对象的文本描述，并且，如果是当前分配，则包含对标准中信息对象定义位置的引用。标记为 Status 的列使用以下约定显示分配值的状态：

R *预订的。*对象标识符值保留供本标准将来使用。

碳 *当前的。*对象标识符值已分配给标准当前修订版中定义的信息对象。

德 *已弃用。*对象标识符值已分配给在标准先前修订版中定义的信息对象，并且该对象的使用现在已弃用。

E.2 分配表

标准特定扩展的分配 对象标识符值的不变部分 = {iso(1) 成员机构(2) us(840) ieee802dot1D(10009)标准特定扩展(0)}		
弧	**目的**	**地位**
不适用	不适用	不适用

功能单元包的分配 对象标识符值的不变部分 = {iso(1) 成员机构(2) us(840) ieee802dot1D(10009) functionalUnitPackage(1)}		
弧	**目的**	**地位**
桥梁管理功能单元(0)	桥接功能单元包标识符（IEEE Std 802.1j-1996 的 7.1）	德
桥梁管理功能单元(1)	桥接功能单元包标识符（15.1）	碳

ASN.1 模块标识符的分配 对象标识符值的
不变部分 =
{iso(1) 成员机构(2) us(840) ieee802dot1D(10009) asn1Module(2)}

弧	目的	地位
bridgeDefinitions(0) version1(0)	Bridge ASN.1 定义模块版本 1 的标识符	德
bridge定义(0) version2(1)	Bridge ASN.1 定义模块第 2 版的标识符	碳

管理对象类的分配 对象标识符值的不变
部分 =
{iso(1) 成员机构(2) us(840) ieee802dot1D(10009) managedObjectClass(3)}

弧	目的	地位
macBridgeDLE(0)	MAC Bridge DLE 管理对象类名称 (15.3)	碳
端口（1）	端口管理对象类名称（IEEE Std 802.1j-1996 的 7.4）	德
选择性翻译表条目(2)	选择性翻译表条目管理对象类名（15.5）	碳
永久数据库（3）	永久数据库管理对象类名（15.6）	碳
过滤数据库(4)	过滤数据库管理对象类名（IEEE Std 802.1j-1996 的 7.7）	德
数据库条目(5)	数据库条目管理对象类名（15.8）	碳
端口（6）	端口管理对象类名（15.4）	碳
过滤数据库（7）	过滤数据库管理对象类名（15.7）	碳
garp应用(8)	GARP 应用程序管理对象类名称（15.9）	碳
garpAttributeType(9)	GARP 属性类型管理对象类名（15.10）	碳
garp属性(10)	GARP 属性管理对象类名（15.11）	碳
garp计时器 (11)	GARP 计时器管理对象类名称（15.12）	碳

软件包分配
对象标识符值的不变部分 =
{iso(1) 成员主体(2) us(840) ieee802dot1D(10009) 软件包(4)}

弧	目的	地位
fdGroupSupport (0)	通过过滤数据库管理对象类 (15.7) 支持组注册条目计数器的条件包名称	碳
garpOriginatorSupport (1)	支持 GARP PDU 发起者地址记录的条件包名称（15.11）	碳

参数分配 对象标识符值的不变部分 =

{iso(1) 成员主体(2) us(840) ieee802dot1D(10009) 参数(5)}

弧	目的	统计我们
不适用	不适用	不适用

名称绑定标识符的分配 对象标识符值的不 变部分 = {iso(1) 成员机构(2) us(840) ieee802dot1D(10009) nameBinding(6)}		
弧	**目的**	**地位**
macBridgeDLE-数据链路子系统(0)	当包含在数据链路子系统中时，MAC Bridge DLE 的名称绑定标识符 (15.3.1)	碳
端口-MACBridgeDLE(1)	包含在 MAC Bridge DLE 中的端口名称绑定标识符 (15.4.1)	碳
选择性翻译表入口端口(2)	包含在端口中的选择性转换表条目的名称绑定标识符（15.5.1）	碳
permanentDatabase-MACBridgeDLE(3)	包含在 MAC Bridge DLE 中的永久数据库的名称绑定标识符 (15.6.1)	碳
过滤数据库-MACBridgeDLE(4)	当包含在 MAC Bridge DLE (15.7.1) 中时，用于过滤数据库的名称绑定标识符	碳
permanentEntry-永久数据库(5)	静态过滤条目包含在永久数据库中时的名称绑定标识符 (15.8.1)	碳
静态条目过滤数据库(6)	包含在过滤数据库中的静态过滤条目的名称绑定标识符 (15.8.1)	碳
动态条目过滤数据库(7)	包含在过滤数据库中的动态过滤条目的名称绑定标识符 (15.8.1)	碳
groupEntry-FilteringDatabase(8)	包含在过滤数据库中时组注册条目的名称绑定标识符 (15.8.1)	碳
garp应用程序端口(9)	包含在端口中的 GARP 应用程序的名称绑定标识符 (15.9.1)	碳
garpAttributeType-GARPApplication(10)	当包含在 GARP 应用程序中时，GARP 属性类型的名称绑定标识符 (15.10.1)	碳
garpAttribute-GARPAttributeType(11)	当 GARP 属性包含在 GARP 属性类型中时，其名称绑定标识符 (15.11.1)	碳
garp定时器端口(12)	包含在端口 (15.12.1) 中的 GARP 计时器的名称绑定标识符	碳

属性标识符的分配 对象标识符值的不变 部分 = {iso(1) 成员主体(2) us(840) ieee802dot1D(10009) 属性(7)}		
弧	**目的**	**地位**
桥地址(0)	桥接地址属性类型名称（15.3.2）	碳
桥梁名称(1)	桥名称属性类型名称（15.3.3）	碳
桥接端口号(2)	桥接端口数属性类型名称（15.3.4）	碳
桥接端口地址(3)	桥接端口地址属性类型名称（15.3.5）	碳
正常运行时间(4)	正常运行时间属性类型名称（15.3.6）	碳
桥接标识符(5)	桥接标识符属性类型名称（15.3.7）	碳
自拓扑变化以来的时间(6)	自拓扑改变以来的时间属性类型名称（15.3.8）	碳
拓扑变化计数(7)	拓扑变化计数属性类型名称（15.3.9）	碳
拓扑改变(8)	拓扑变化属性类型名称（15.3.10）	碳
指定根(9)	指定根属性类型名称（15.3.11）	碳
根路径成本(10)	根路径成本属性类型名称（15.3.12）	碳
根端口(11)	根端口属性类型名称（15.3.13）	碳
最大年龄(12)	最大年龄属性类型名称（15.3.14）	碳
你好时间（13）	Hello Time 属性类型名称（15.3.15）	碳

属性标识符的分配 ***（持续）*** 对象标识符值的不变部 分 = {iso(1) 成员主体(2) us(840) ieee802dot1D(10009) 属性(7)}		
弧	**目的**	**地位**
转发延迟（14）	转发延迟属性类型名称（15.3.16）	碳
bridgeMaxAge(15)	Bridge Max Age 属性类型名称（15.3.17）	碳
bridgeHelloTime(16)	桥接问候时间属性类型名称（15.3.18）	碳
桥接转发延迟(17)	桥接转发延迟属性类型名称（15.3.19）	碳
保持时间（18）	保持时间属性类型名称（15.3.20）	碳
桥优先级(19)	桥接优先级属性类型名称（15.3.21）	碳
端口号（20）	端口号属性类型名称（15.4.2）	碳
端口名称(21)	端口名称属性类型名称（15.4.3）	碳
端口类型(22)	端口类型属性类型名称（15.4.4）	碳
已接收帧数(23)	已接收帧属性类型名称（15.4.5）	碳
丢弃入站（24）	丢弃入站属性类型名称（15.4.6）	碳
转发出站(25)	转发出站属性类型名称（15.4.7）	碳
丢弃缺少的缓冲区（26）	丢弃缺少缓冲区的属性类型名称（15.4.8）	碳
丢弃TransitDelayExceeded（27）	丢弃传输延迟超出属性类型名称（15.4.9）	碳
错误时丢弃(28)	丢弃错误属性类型名称（15.4.10）	碳
丢弃错误详细信息（29）	丢弃错误详细信息属性类型名称（15.4.11）	碳
出站用户优先级(30)	出站用户优先级属性类型名称（IEEE Std 802.1j-1996 的 7.4.11）	德
出站访问优先级(31)	出站访问优先级（IEEE 标准 802.1j-1996 的 7.4.12）	德
端口正常运行时间(32)	端口正常运行时间属性类型名称（15.4.16）	碳
州(33)	状态属性类型名称（15.4.17）	碳
端口标识符（34）	端口标识符属性类型名称（15.4.18）	碳
端口优先级（35）	端口优先级属性类型名称（15.4.19）	碳
路径成本（36）	路径成本属性类型名称（15.4.20）	碳
指定根端口(37)	端口指定根属性类型名称（15.4.21）	碳
指定成本(38)	指定成本属性类型名称（15.4.22）	碳
指定桥梁(39)	指定桥属性类型名称（15.4.23）	碳
指定端口(40)	指定端口属性类型名称（15.4.24）	碳
拓扑更改确认（41）	拓扑改变确认属性类型名称（15.4.25）	碳
类型值（42）	类型值属性类型名称（15.5.2）	碳
数据库名称(43)	数据库名称属性类型名称（15.6.2）	碳
数据库大小(44)	数据库大小属性类型名称（15.6.3）	碳
条目数（45）	条目数属性类型名称（15.6.4）	碳
动态条目数(46)	动态条目数属性类型名称（15.7.2）	碳
老化时间（47）	老化时间属性类型名称（15.7.3）	碳
地址(48)	地址属性类型名称（IEEE 标准 802.1f 的 7.8.1）	德
端口号或映射(49)	端口号或映射属性类型名称（IEEE Std 802.1j-1996 的 7.8.2）	德
条目类型（50）	条目类型属性类型名称（15.8.4）	碳

属性标识符的分配 ***（持续）*** 对象标识符值的不变部 分 = {iso(1) 成员主体(2) us(840) ieee802dot1D(10009) 属性(7)}		
弧	**目的**	**地位**
条目索引(51)	条目索引属性类型名称（15.8.5）	碳
默认用户优先级（52）	默认用户优先级属性类型名称（15.4.12）	碳
用户优先级再生表(53)	用户优先级再生表属性类型名称（15.4.13）	碳
交通类别表(54)	流量类别表属性类型名称（15.4.12）	碳
出站访问优先级表(55)	出站访问优先级表属性类型名称（15.4.12）	碳
组注册条目数(56)	组注册条目数属性类型名称（15.7.4）	碳
地址(57)	地址属性类型名称（15.8.2）	碳
端口映射(58)	端口映射属性类型名称（15.8.3）	碳
应用程序名称(59)	应用程序名称属性类型名称（15.9.2）	碳
加入时间(60)	连接时间属性类型名称（15.12.3）	碳
离开时间(61)	离开时间属性类型名称（15.12.4）	碳
离开所有时间(62)	保留所有时间属性类型名称（15.12.5）	碳
注册失败(63)	失败注册属性类型名称（15.10.4）	碳
garpAttributeType(64)	GARP 属性类型属性类型名称（15.10.2）	碳
garp属性(65)	GARP 属性属性类型名称（15.11.2）	碳
申请人行政控制(66)	申请人行政控制属性类型名称（15.10.3）	碳
状态值(67)	状态值属性类型名称（15.11.3）	碳
originatorOfLastPDU(68)	最后一个 PDU 属性类型名称的发起者（15.11.4）	碳
garp计时器 (69)	Garp Timers 属性类型名称（15.12.2）	碳

属性组标识符的分配 对象标识符值的不变部 分 = {iso(1) 成员机构(2) us(840) ieee802dot1D(10009) 属性组(8)}		
弧	**目的**	**地位**
读取或发现桥接（0）	读取或发现桥接属性组名称（15.3.22）	碳
读取桥接协议参数(1)	读取桥接协议参数属性组名称（15.3.23）	碳
读端口（2）	读取端口属性组名称（15.4.26）	碳
读取转发端口计数器(3)	读取转发端口计数器属性组名称 (15.4.27)	碳
读取传输优先级（4）	读取传输优先级属性组名称（IEEE Std 802.1j-1996 的 7.4.25）	德
读端口参数(5)	读取端口参数属性组名称（15.4.28）	碳
读取数据库（6）	读取数据库属性组名称（15.6.5）	碳
读取数据库条目（7）	读取数据库条目属性组名称（IEEE Std 802.1j-1996 的 7.8.5）	德
读取数据库条目（8）	读取数据库条目属性组名称（15.8.6）	碳

行动类型的分配 对象标识符值的不变部 分 = {iso(1) 成员机构(2) us(840) ieee802dot1D(10009) 行动(9)}		
弧	**目的**	**地位**
重置桥接（0）	重置桥梁动作类型名称（15.3.24）	碳
forcePortState(1)	强制端口状态动作类型名称（15.4.29）	碳

通知类型的分配 对象标识符值的不变部 分 = {iso(1) 会员机构(2) us(840) ieee802dot1D(10009) 通知(10)}		
弧	**目的**	**地位**
不适用	不适用	不适用

附件 F

（信息性）

目标拓扑、迁移和互操作性

F.1 目标拓扑

图 F-1 给出了桥接 LAN 配置示例，该示例说明了桥接 LAN 中可能出现的双端口/多端口桥接、冗余路径等的简单组合。该示例是在人们关注双端口桥接连接共享媒体 LAN 时制作的，该示例的主要目的是说明生成树处理冗余路径的能力。

由于当前和未来的重点是采用多端口桥接和交换技术的集线器架构，因此在标准中包含此类拓扑的示例是适当的。这一点尤其重要，因为为了充分利用动态多播过滤功能，必须采用此类 LAN 拓扑。因此，下图展示了更具代表性的生成树拓扑，包括共享媒体元素和集线器架构。

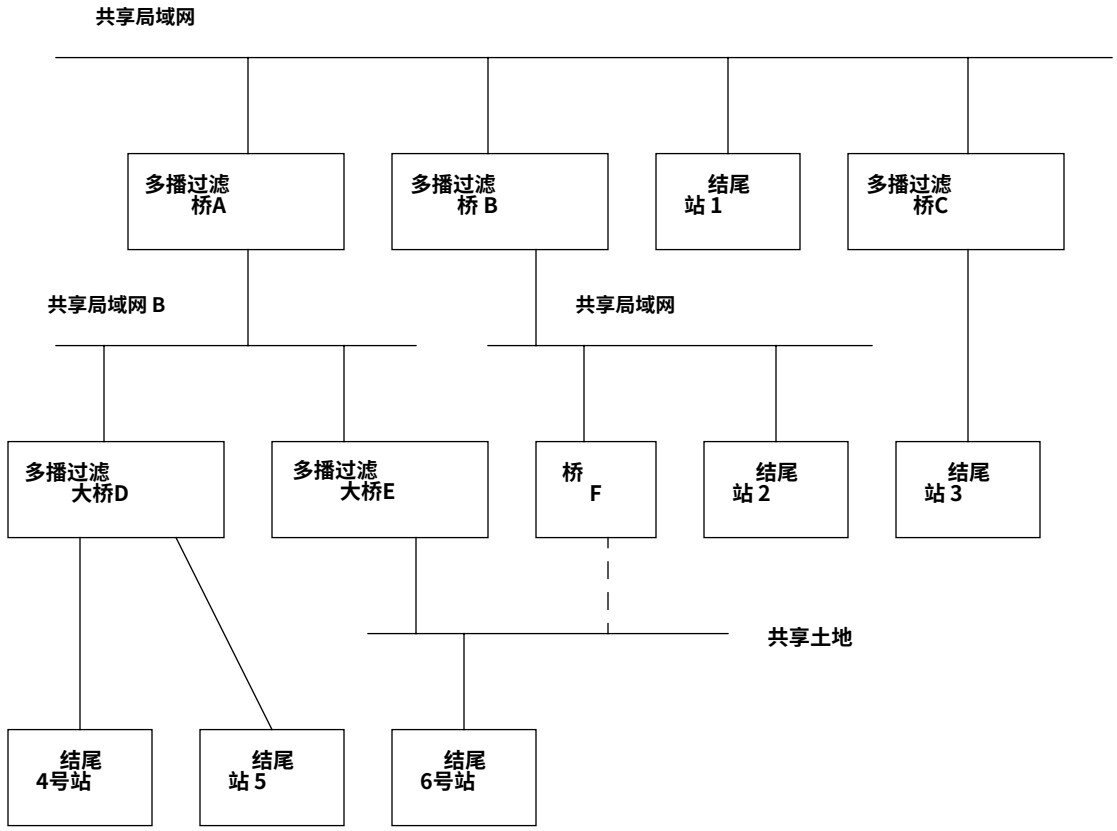


图 F-1 — 生成树拓扑示例

F.2 迁移注意事项

本小节探讨了混合模式局域网中可能遇到的一些问题及其可能的解决方案；即包含

a) 传统桥接器（仅支持基本过滤服务的桥接器）和增强型桥接器（支持扩展过滤服务的桥接器）的混合；
b) 不支持 GMRP 的传统终端站与支持 GMRP 的终端站的混合。

鉴于 LAN 环境中的用途、配置、流量模式和性能要求范围非常大，本文并非旨在提供权威性结论；必须根据给定桥接 LAN 的特定环境和要求来定义其迁移策略。

F.2.1 异构桥接环境

GMRP 的操作对传统网桥完全透明；即，所有 GMRP PDU 都由传统网桥透明转发。因此，可以在桥接 LAN 中以任意方式混合传统网桥和增强型网桥，而不会影响 GMRP 生成的有向图的完整性。

注意 — 仅当传统网桥不包含任何可能会干扰发往任何 GARP 应用程序地址的 PDU 传输的静态过滤条目时，情况才如此。

在异构桥接 LAN 中，将任何增强型桥接器放置在网络外围，而不是网络核心内，可能会在流量隔离方面产生最大效益。但是，这种效益也可能取决于 GMRP 感知和 GMRP 不感知终端站的混合和分布（参见 F.2.2）。一般而言，特定桥接器放置的位置越靠近桥接 LAN 的“中心”，其过滤数据库中定义的任何组在所有端口上都有注册成员的可能性就越大；因此，桥接器将过滤相对较少的组流量。因此，将传统桥接器放置在桥接 LAN 中的类似位置会对相邻 LAN 段产生相对较小的额外负载。相反，越靠近桥接 LAN 的外围，过滤数据库中定义的组在每个端口上都有不同成员模式的可能性就越大，因此扩展过滤服务将更有价值。

F.2.2 异构终端站环境

F.2.2.1 使用基本过滤模式和传统桥接

由于传统网桥对于 GARP 协议交换是透明的，因此终端站在仅由传统网桥服务的网段上的放置对于本次讨论来说并不重要。

F.2.2.2 使用转发所有组

如果端口过滤数据库中保存的静态和动态信息表明“转发所有组”的服务要求，则“转发所有组”是给定端口采用的默认组过滤行为 (7.9.4)。将“转发所有组”注册为服务要求 (6.6.7) 旨在用于两个目的：

a) 确保仅包含传统设备的桥接局域网区域可以接收源自桥接局域网任何其他区域的所有多播帧（通过静态配置明确禁止的帧除外）；

b) 允许需要混合接收的设备成功运行，例如路由器或网络监视器，如 F.3 中所述。

理想情况下，所有不支持 GMRP 的设备都将连接到由桥接端口服务的 LAN 段，这些端口以转发所有组作为其默认组过滤行为运行。由于这些段将吸引来自任何其他 LAN 段的所有多播流量，因此这些旧段理想情况下位于桥接 LAN 的同一区域。如 F.2.1 中所示，将这些段连接到靠近桥接 LAN 中心的桥接器也可能是合适的。

F.2.2.3 前向未注册组的使用

如果端口过滤数据库中保存的静态和动态信息表明需要“转发未注册组”服务，但不表明需要“转发所有组”服务，则“转发未注册组”是该端口采用的默认组过滤行为 (7.9.4)。“转发未注册组”注册的预期用途之一是提供从 GMRPunaware 环境到 GMRP 感知环境的迁移。未注册组的“直通 ”行为将允许 GMRP 未感知终端站继续在由“转发未注册组”过滤行为服务的段上运行，只要它们希望接收的多播地址与 GMRP 感知终端站使用的多播地址不同（F.2.2.5 考虑了所用地址相同的情况）。一般而言，将 GMRP 未感知终端站放置在由提供“转发所有组”作为其默认组过滤行为的端口服务的段上更为合适。

在 GMRP 感知设备能够区分一组“旧”多播地址（它们未注册）和一组“新”多播地址（它们注册）的情况下，转发未注册组也很有用。同样，这种情况只有在 GMRP 感知设备仅使用“旧”多播地址时才可行。

F.2.2.4 使用过滤未注册组

过滤未注册组需要明确注册组成员身份，以便终端站接收发往组的帧。如果端口过滤数据库中保存的静态和动态信息未指示转发所有组或转发未注册组的服务要求，则该端口将采用默认组过滤行为 (7.9.4)。它本质上是为仅与支持 GMRP 的终端站一起运行而设计的。它在很大程度上不适合在包含不支持 GMRP 的设备的任何 LAN 段上运行，除非网络管理员准备手动配置所有此类端口的 GARP 管理控制参数以满足所有连接的不支持 GMRP 的设备的已知要求。

另一种方法是在包含不支持 GMRP 的设备的任何共享媒体段上包含一个“代理成员”，该成员代表其段上的终端站加入一组适当的组。但是，网络管理员必须再次准备好手动配置“代理成员”才能使其工作，因此此解决方案与手动配置过滤数据库略有不同。

因此，一般来说，围绕使用过滤未注册组来设计迁移策略似乎是不合适的，除非 LAN 的物理配置允许隔离支持 GMRP 和不支持 GMRP 的终端站。

F.2.2.5 使用一组通用的地址

从不支持 GMRP 的环境迁移到支持 GMRP 的环境时，一个主要危险是，如果两种类型的终端站都是一组通用 MAC 地址的接收者，则不支持 GMRP 的终端站可能会因为支持 GMRP 的终端站使用该 MAC 地址注册组成员而失去权利。这种情况发生的可能性很大程度上取决于相关环境。鉴于 GMRP 注册机制的使用在许多情况下与新应用程序和软件实施紧密相关，这个问题的实际影响在实践中可能比理论上看起来要小。

然而，考虑到危险，任何迁移环境都需要进行分析，以确定问题是否真的存在。解决问题的策略可能再次围绕 F.2.2.2、F.2.2.3 和 F.2.2.4 中讨论的方法，即：

a) 在转发所有组模式段上隔离传统终端站；
b) 手动配置过滤数据库；
c) 使用“代理成员”终端站。

F.3 与高层多播协议的互操作性及相关问题

显然，GMRP 必须与桥接 LAN 中可能实施更高层多播协议的设备共存和互操作。这里的主要示例是支持 IP 多播和 IGMP（IP 组管理协议）的 IP 路由器；但是，可能会遇到具有类似特征的其他示例。

虽然这本身不是一个高层协议问题，但可能还需要容纳监控桥接 LAN 中的多播流量的设备，这些设备的操作依赖于混杂接收。

F.3.1 IP多播

IP 路由器需要接收发送到指定用于 IP 多播的多播地址范围内的任意地址的多播帧。

一种可行的方法是确保与路由器连接到同一 LAN 段的所有桥接端口都采用默认的组过滤行为“转发所有组”。这可以通过静态配置相关端口的过滤数据库来实现，也可以通过路由器向其 GMRP 参与者发出 REGISTER_SERVICE_REQUIREMENT 原语（指定“转发所有组”）来实现，后者又将发送包含 JoinIn 或 JoinEmpty 消息 (12.10.4.2) 的 GMRP PDU 以进行“转发所有组”注册。对于所有连接到同一 LAN 段的桥接端口（包括通过仅提供基本过滤服务的桥接间接连接的任何桥接端口），这将强制它们将其默认的组过滤行为更改为“转发所有组”。这将确保桥接 LAN 中的所有桥接端口都将向路由器方向转发多播帧，无论这些帧是否发往已注册的组。无论桥接 LAN 中存在多少个路由器，这种方法都能起作用；但是，它的特点是，路由器所连接的 LAN 段将“看到”桥接 LAN 中任何地方生成的所有多播流量，无论路由器是否对此感兴趣。

理想情况下，路由器将通过仅连接 GMRP 未知终端站的 LAN 段连接到网桥，从而确保桥接 LAN 中支持 GARP 的设备从组过滤服务中获得最大收益。将此段连接到靠近桥接 LAN 中心的网桥也可能是合适的，如 F.2.1 中对传统设备的讨论中所述。

参与 IP 多播接收的 GMRP 感知终端站必须能够使用 GMRP 来管理其组成员身份，这是 IP 多播使用所必需的。例如，为了使用 IGMP 加入给定的 IP 多播组，终端站必须在发出 IGMP 加入之前为该组执行 GMRP 加入。

注意——尽管是在 IP 多播环境中描述的，但此处描述的方法可能适用于实现支持多播的其他协议架构的路由器。

F.3.2 监控多播流量

将此标准中定义的动态多播过滤功能添加到桥接 LAN 意味着，此前可以感知桥接 LAN 中所有多播流量（除受静态过滤器限制的流量外）的网络监视器现在可能不知道某些多播流量，这取决于监视设备的连接点以及网络中该点存在的组注册模式。因此，多播的情况与单播的情况非常相似，其中过滤数据库中的动态条目导致某些单播流量在某些 LAN 段上不可见。

如果需要监控桥接 LAN 中的所有多播流量，最简单的方法是接收除发往静态过滤信息阻止成员资格的组的流量之外的所有流量，即监控器确保所有连接到与监控器相同的 LAN 段的桥接端口都采用转发所有组的默认组过滤行为，如 F.3.1 中所述。

此场景的理想配置是，监视器连接到已由采用“转发所有组”过滤行为的端口提供服务的 LAN 段，或者通过没有其他终端站、网桥或路由器连接的 LAN 段连接到网桥。将此段连接到靠近桥接 LAN 中心的网桥也可能是合适的，如 F.2.1 中对传统设备的讨论中所述。

值得注意的是，除非监视器连接到一个 LAN 段，并且该段服务的所有桥接端口通常都采用转发所有组行为（例如，如 F.2.1 中讨论的传统 LAN 段），否则这种方法可能会导致桥接 LAN 中多播帧的正常分布发生变化（即，在没有监视器本身引入的过滤模式变化的情况下会发生分布变化）。

附件 G

（信息性）

保持 MAC 桥接器中 FCS 字段的完整性

G.1 背景

在 MAC 桥接器的端口之间中继帧时，桥接器的功能之一是根据适用于中继帧的介质访问方法的 MAC 程序重新生成 FCS 字段。MAC 桥接器标准的要求之一是，任何此类 FCS 的重新生成都不得增加传输 MAC 上遇到的未检测到的 FCS 错误水平，使其超过保留 FCS 所遇到的水平（参见 6.3.7 和 7.7.6）。此处的争论点只是*未被发现*错误率；符合要求的桥接器将丢弃接收到的带有 FCS 错误的帧；但是，也存在发生不可检测的损坏的可能性，即，帧已损坏，但 FCS 算法未揭示该错误。

本附件讨论了与 FCS 再生有关的 MAC Bridge 操作的各种情况，以及为解决这些情况可采取的替代方法。需要考虑以下情况：

a) 从FCS算法的操作角度来看，源和目标MAC方法是相同的；

> 注：例如，源 MAC 方法和目标 MAC 方法相同时，就会出现这种情况。如果两种 MAC 方法中的 FCS 覆盖范围、FCS 算法、FCS 所涵盖字段的定义以及位/八位字节排序相同，则这种情况也可能发生在不同的 MAC 方法中。

b) 从FCS算法的操作角度来看，源和目标MAC方法有所不同；

c) FCS所覆盖的数据不会被Bridge的中继功能的操作所修改；
d) 通过 Bridge 的中继功能操作来修改 FCS 所涵盖的数据。

在每种情况下，重要的是确保在删除旧 FCS 和计算新 FCS 之间，由于帧数据部分损坏而产生额外未检测到错误的概率很低，这将导致传输的帧携带无效数据和有效 FCS。例如，由于环境噪声对桥梁硬件操作的影响，可能会发生此类损坏。

如果 a) 和 c) 均为真，则接收帧中携带的 FCS 仍然有效，理想的方法是将该值重用为传输帧的 FCS。

如果 b) 或 d) 为真，则 Bridge 需要在传输时重新计算 FCS，因此可能需要采取额外的预防措施，以防止内存损坏导致未检测到的 FCS 错误级别增加。

G.2 CRC 和 FCS 背后的基本数学思想

标准循环冗余校验（CRC）算法基于以下思想：

a) 一个 n 位消息被视为一个 n-1 度的多项式 M(x)；

b) 为了生成长度为 r 位的 CRC 值，使用度为 r 的生成多项式 G(x)；

c) 选择 M(x) 的最后 r 位的值，使得 M(x) ÷ G(x) 的余数为 0（即，M(x) = 0 mod G(x)）。

可以添加消息，其中每个系数为 0 或 1：

d) M3(x) = M1(x) + M2(x)。通过对具有相同位位置的系数（即具有相同指数的系数）进行按位加法（XOR）将消息加在一起。

消息减法：

e) 加法和减法是等价运算（因为 A = A XOR B XOR B）；因此，使用上面示例 d）中的消息，可以通过添加 M2(x) 从 M3(x) 再生 M1(x)。

线性：

f) 如果 M(x) = 0 mod G(X)，则 (M(x) + E(x)) = 0 mod G(x) IFF E(x) = 0 mod G(x)。换句话说，如果将错误模式 E(x) 添加到消息 M(x)，则如果 M(x) 和 E(x) 在除以生成多项式时都没有余数，则结果消息在除以生成多项式时将不会产生余数。

g）x米M(x) = 0 mod G(x) IFF M(x) = 0 mod G(x). 可以通过添加零填充来移动消息，而不会影响 CRC 的完整性。

h) 因此，CRC 算法具有以下特性：它将检测出应用于消息的任何长度为 r 位或更少的突发错误，因为当该消息除以生成多项式时，此类错误必然会产生非零余数。或者，将两条都具有有效 CRC 的消息相加，将产生一条具有有效 CRC 的消息。

在以太网中，使用的算法与标准 CRC 不同，称为帧校验序列 (FCS)。FCS 基于 CRC，但有以下区别：

— 在计算FCS值之前，对M(x)的前32位进行补码；

— CRC 值的计算方法如上所述，通过将 M(x) 的前 n-32 位除以 G(x)，将余数作为 CRC 值；

— 将 CRC 值取补数并插入为 M(x) 的最后 32 位。

通过应用上面的加法和线性规则 d）和 f），以太网帧及其 FCS 可以看作是由以下组成消息相加而成：

— 一个 n 位消息 M(x)，其最后 32 位携带标准 CRC；

— n 位“FCS 调整因子”，A_n（x）调整CRC以形成FCS。

图 G-1 说明了通过添加调整因子，将带有 CRC 的消息转换为带有 FCS 的等效消息。调整因子由两部分组成：

— 第一个组件添加到 M(x) 时会对 CRC 进行调整，这相当于在计算 CRC 位之前对 M(x) 的前 32 位进行补充，并对 M(x) 的前 32 位进行补充。

— 第二个组件，当添加到部分调整后的 M(x) 时，将 M(x) 的前 32 位恢复为其原始值，并补充（调整后的）CRC 以形成最终的 FCS。

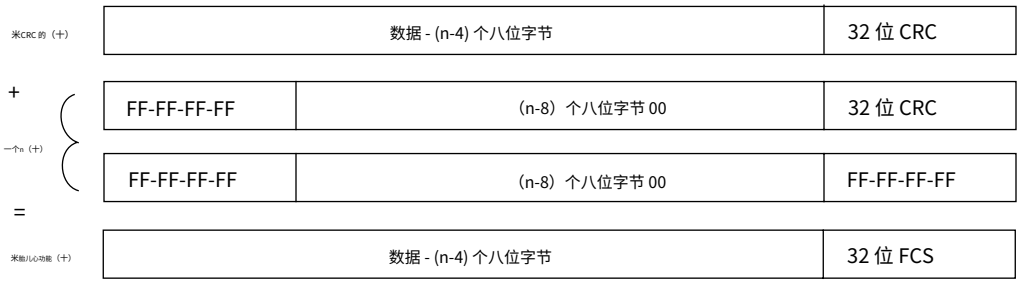


图 G-1 — 将 CRC 转换为 FCS

G.3 检测无损电路方法

图 G-2 说明了这种方法。这种方法的基础是，修改后的消息中使用的 FCS 总是使用正常的 FCS 生成算法（检查生成器 B）从头开始重新计算；但是，原始消息数据和 FCS 也用作 FCS 检查器（检查器 A）的输入，以检查 FCS 生成器的输入是否正确。因此，如果在收到消息并进行 FCS 检查后的时间段内，消息在 Bridge 内存中时出现错误，则检查器 A 将在生成修改后消息的新 FCS 的同时检测到错误。

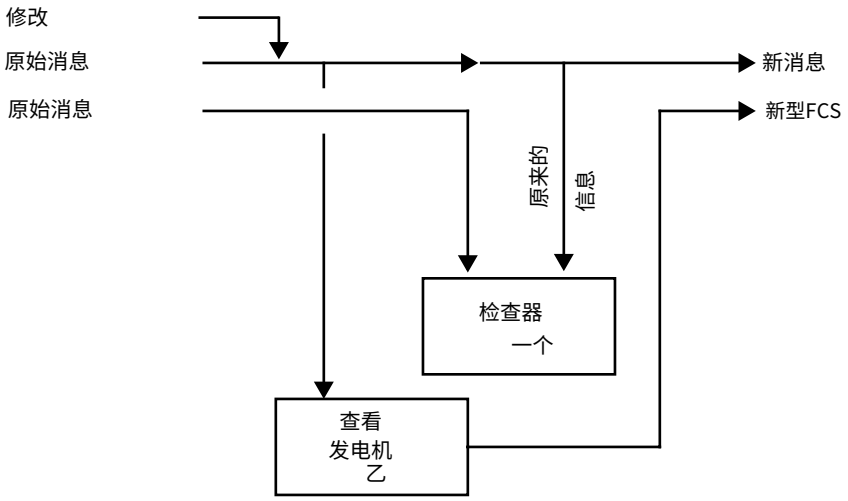


图 G-2—检测无损电路

检验器 A 同时接收未经修改的原始消息和用于构建新消息的原始消息；这发生在检验生成器 B 生成新的 FCS 之后（或同时）。检验器 A 检测到的任何差异都表明用于生成新消息及其 FCS 的信息是可疑的，因此应丢弃该消息。

G.4 FCS 的算法修改

如果由于数据更改而非 FCS 算法操作更改而需要重新计算 FCS，则一种选择是检查对消息所做的更改并计算反映这些更改的 FCS 调整因子，而不是从头开始为给定消息重新计算 FCS。有两种情况：

a) 报文整体长度不变，但是FCS覆盖的一个或多个八位字节的数据内容发生变化；

b) 由于添加或删除了 FCS 所涵盖的八位字节，因此消息的总长度增加或减少，但原始八位字节的内容保持不变（尽管由于插入/删除，某些八位字节的位置可能发生了变化）。

G.4.1 数据改变，长度不变

调整消息，$MF_1(x)$，为了将字段 F 的值从当前值 F 改变为$_1$变为新值 F_2包括以下步骤：

— 删除长度为 n 的消息的 FCS 调整因子 A_n（十）；
— 删除字段 F 的当前值对消息及其 FCS 的贡献$_1$（十）；
— 将字段的新值 F 添加到消息及其 FCS 中$_2$（十）；
— 为长度为 n 的消息添加 FCS 调整因子 A_n（十）。

换句话说，

$$MF_2(x) = MF_1(x) - A_n(x) - F_1(x) + F_2(x) + \text{一个}_n(十)$$

A 的加减$_n$（x）相互抵消，所以，

$$MF_2(x) = MF_1(x) - F_1(x) + F_2(十)$$

此外，校正因子 F_1（x）和 F_2（x）可以用一个因子 F 代替$_3$（x），这是 F 产生的字段 F 的值对 M(x) 及其 FCS 的贡献$_1$异或函数$_2$。 所以，

$$MF_2(x) = MF_1(x) + F_3(十)$$

如图 G-3 所示。

或者，

MF 的 FCS_2（x）=（MF 的 FCS_1（x））+调整

在哪里

调整 = (F 的 FCS_1（x））+（F 的 FCS_2（十））

或者等价地，

MF 的 FCS_2（x）=（MF 的 FCS_1（x））+（F 的 FCS_3（十））

调整因子（F的FCS_3（x））恰好是将 F 的值从 F 改回时使用的相同因子$_2$至 F_1。

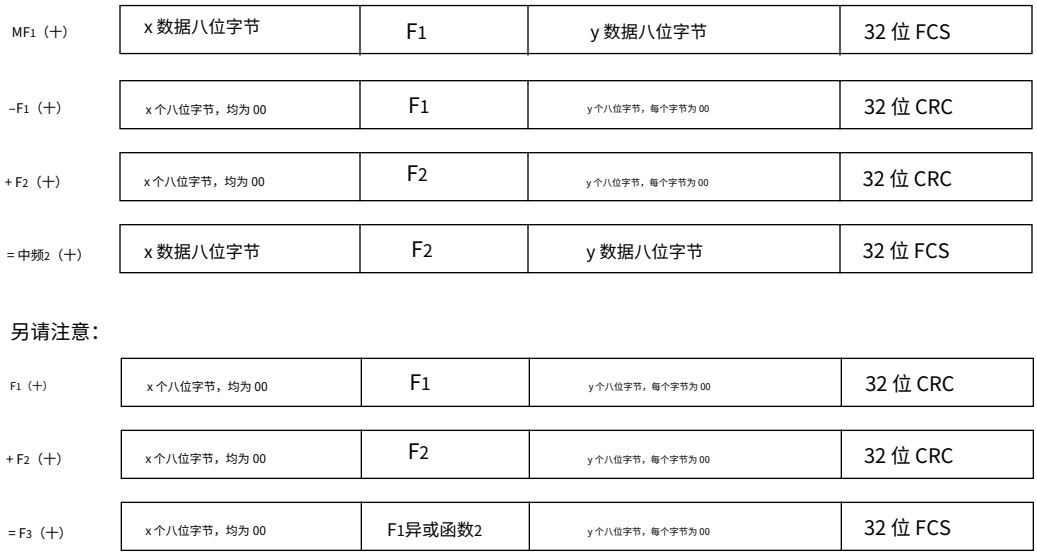


图 G-3—字段变更调整

这种情况相对简单（相对于长度变化的情况），这是线性的结果，并且 FCS 调整因子 An（x）对于任何给定的消息长度 n 都是常数，而不是消息中携带的数据的函数。

G.4.2 长度改变，原始数据不变

调整长度为 n 的消息 M(x) 以适应长度为 i 的插入字段 I 涉及以下步骤：

— 删除长度为 n 的消息的 FCS 调整因子 An（十）；
— 删除消息头部分（插入字段之前的部分）对消息和FCS的贡献H(x)；

— 将报头加上插入的字段 I, HI(x) 提供给消息和 FCS 的贡献添加进去；

— 为长度为 n+i 的消息添加 FCS 调整因子 An+i（十）。

所以，

$$MI(x) = M(x - A)n（x）-H（x）+ HI（x）+ An+i（十）$$

如图 G-4 所示。

或者，

MI(x) 的 FCS = (M(x) 的 FCS) + 调整

在哪里

调整 = （A 的 CRCn(x)) + (H(x) 的 CRC) + (HI(x) 的 CRC) + ((An+i（十））

与字段更改示例一样，*调整*如果从消息中删除字段 I，则将使用相同的调整因子来调整 MI(x) 的 FCS。

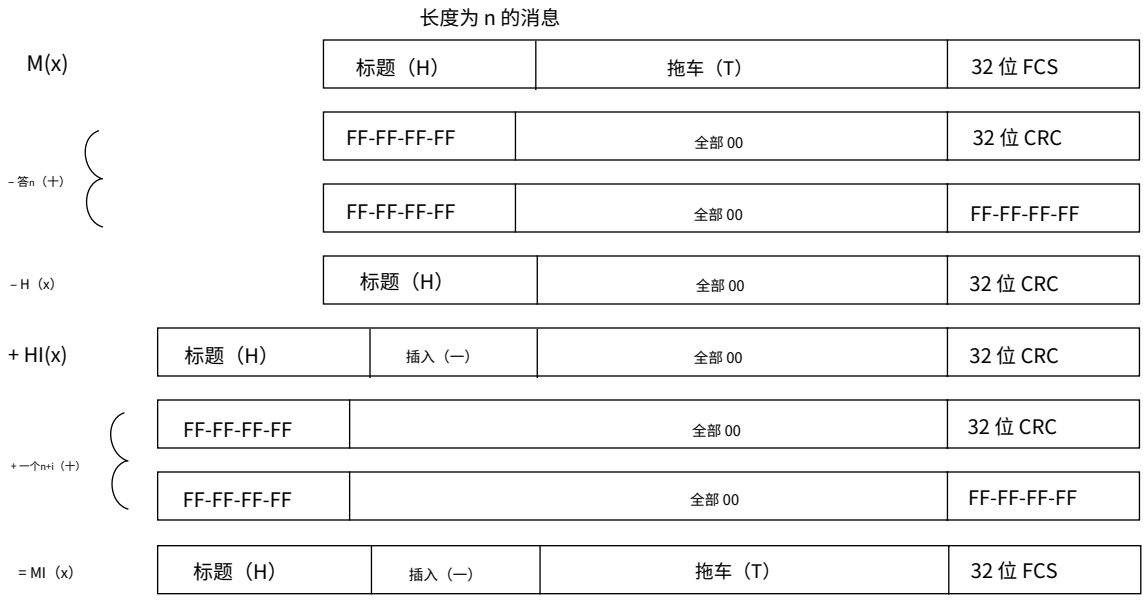


图 G-4 — 字段插入调整

G.4.3 可检测性的保存

对于这里描述的任何一种算法调整方法，重要的问题是 FCS 检测消息错误的能力在这些转换过程中是否能够得到保留；换句话说，如果在修改、添加或删除字段之前将可检测的错误模式 E(x) 添加到消息 M(x) 中，那么在调整消息之后是否仍然可以检测到该错误。

注：以下分析假设 E(x) 是可检测的错误模式，长度为 32 位或更短。本文尚未尝试分析该方法在较长错误模式下的性能。

当错误成分位于消息中不受转换影响（改变或转移）的部分时，很明显，由于转换不考虑消息中的错误部分，因此该错误的可检测性保持不变。

在字段更改的情况下，其中字段 F_1将被替换为 F_2，并且在更改字段之前已在消息的某些部分应用了 E(x)，则新的 FCS 值将按如下方式计算：

MF 的 FCS_2（x）=（MF 的 FCS_1（x））+（F 的 FCS_1（x））+（F 的 FCS_2（十））

然而，应用于FCS的校正因子仅包含基于F的成分$_1$和 F_2，并且没有与误差模式 E(x) 相关的贡献。因此，在场的值变为 F 之后$_2$这样，无论错误模式出现在消息的什么位置，按照上面计算出来的新的 FCS 值仍然能够检测出错误模式 E(x)。

在字段插入（或移除，因为插入和移除已经被证明是等价的）的情况下，其中 E(x) 已应用于 H(x)，应用于 FCS 的校正因子将包含两个与 E(x) 相关的分量；第一个基于 E(x) 的原始位置，第二个基于 E(x) 移动 i（插入字段的长度），如下所示：

MI(x) 的 FCS = (M(x) 的 FCS) + 调整

在哪里

调整 = （A 的 $CRC_n(x)$) + (H(x) 的 CRC) + (HI(x) 的 CRC) + ((A_{n+i}（十））
+ (E(x) 的 CRC) + ((E(x) 移位 i) 的 CRC)

(E(x) 的 CRC) 分量的存在确保最终消息仍将指示 FCS 错误。（请注意，即使 (E(x) 的 CRC) 和 ((E(x) 移位 i) 的 CRC) 结果是相同的值，最终消息仍将指示 FCS 错误，因为其 FCS 现在对于没有错误分量的 MI(x) 的值是正确的。）

以上内容并未详尽分析所有字段插入和删除的情况，特别是没有分析错误模式跨越插入/删除边界的情况；然而，可以证明，在这些情况下，FCS 的错误检测特性也得到了保留。

从上述示例可以得出的结论是，只要用于计算这些 FCS 校正因子的字段值与原始消息和最终消息中的字段值相同（例如，原始形式和移位形式都使用相同的 H+E 值），那么 FCS 的属性就会保留。但是，如果用于计算校正因子的值与消息中的值不同（例如，由于内存损坏而可能发生这种情况），那么 FCS 的属性将不再保留。因此，如果使用这些算法方法，则必须以确保满足这些条件的方式设计实现。

G.5 结论

如果需要在传输之前重新计算FCS，那么算法FCS修改方法或检测无损电路方法可以为实现此目的提供基础，同时仍然保留原始FCS提供的错误检测覆盖范围。

检测无损电路方法需要访问原始消息的全部内容，以计算必要的 FCS 调整或检查用于创建新消息及其 FCS 的数据的完整性。相比之下，算法方法只需要访问原始消息的标头、插入和 FCS。

对于算法方法，在实施过程中需要小心谨慎，以确保校正算法的输入与原始消息和修改后消息的内容一致，而在检测无损电路中，这种一致性检查是使用原始 FCS 检查新 FCS 的生成方式所固有的。

附件 H

（信息性）

流量类别加速和动态多播过滤的设计注意事项

以下内容旨在记录本标准制定过程中做出的一些较为重要的设计决策。本附件的主要目的是确保此类决策的历史清晰，从而避免重复审查已解决的问题并重新发明解决方案。

H.1 通用属性注册协议

H.1.1 使用未经证实的协议

GARP 的运行本质上依赖于这样的假设：只要状态机能够确保在每组相关消息中发送一条以上的关键消息（JoinIn、JoinEmpty 和 Empty），那么所有预期接收者都有可能收到这些消息。早期设计 GARP 时，如果涉及明确确认，就会出现许多问题，特别是：

a) 如果多个 GARP 参与者连接到一个段，则不希望所有参与者都响应一个参与者的请求。将响应限制为单个参与者是可行的；但是，单个参与者的响应表示“我明白了”，其价值值得怀疑，因为不能保证其他参与者也看到了相同的消息；

b) 开发一种能够高度保证正确操作的分发机制显然是可能的（例如，X.500 的分布式目录协议）；然而，这种方法的复杂性与 MAC 桥接器的基本设计目标相冲突，即，它们基于简单的机制，因此可以设计成具有高性能/成本比；

c) 如果假设处理单个数据包丢失的基本设计目标，则基于多次传输消息的策略可以在保持简单性的同时提供适当的可靠性级别；
d) MAC 服务的用户可以使用其他方式来确定服务是否正在提供。例如，如果用户请求接收视频频道，很快就会知道是否正在接收。因此，相关用户/应用程序可以采用“更高层”恢复策略来处理这种情况，而不会影响 Bridge 的复杂性。这在任何情况下都可能是合适的，因为 Bridge 无法在不同的用户要求和错误条件组合中确定适当的行动方案。

H.1.2 申请人状态机的设计

申请人状态机的设计旨在实现终端站和网桥中 GARP 参与者的要求。严格来说，申请人在终端站中的要求比网桥中的要求略微简单；但是，出于整体清晰和简单的原因，将两者合并为一个描述。主要区别如下。

对于不需要了解其他参与者（例如，属性流量的接收者）注册状态的终端站，其主要关注的是确保连接到其 LAN 段的所有其他参与者都能看到其加入消息。如果其中一个离开消息误入歧途，则 Leave All 垃圾收集

机制最终会解决这个问题，因此其他参与者可靠地接收 Leaves 并不是一个大问题。因此，对于这种终端站中的申请人，状态机可以进一步简化，只需发送两个 JoinIn 即可进入 Active Member 状态，发送一个 Leave 即可进入 Observer 状态。

对于不仅关心自身成员状态而且关心其他参与者的注册状态的终端站或桥接端口，注册器的情况更为复杂。这造成的主要区别是，如果同一网段上还有其他成员或潜在成员，则所有参与者必须听到至少两个成员。这样可以确保每个参与者（包括实际生成消息的参与者）都能听到相关 LAN 网段上的至少一个加入消息。听到两个不同的参与者与看到两条加入消息不同；两条消息可能都由同一个参与者生成。因此，GARP 中发送的加入消息带有参与者注册状态的指示。JoinIn 表示“我希望加入，并且我已经看到一个或多个针对此属性的其他加入消息”；JoinEmpty 表示“我希望加入，并且我没有看到针对此属性的任何其他加入消息”。实际上，发送的加入消息的类型反映了注册器对该属性的注册状态。通过这种改进，认为自己处于成员状态的参与者可以通过发送 JoinIn 来响应 JoinEmpty（表明 JoinEmpty 的发起者尚未注册其成员身份）。

当参与者试图成为会员时，可能会出现以下情况：

a) 该属性的会员资格中没有其他参与者；
b) 该属性的会员资格中还有另外一名参与者；
c) 该属性的会员资格中有两名或两名以上的其他参与者。

在情况 a) 中，新成员将发送两条 JoinEmpty 消息，并进入 Quiet/Active Member 状态。LAN 段上的所有其他参与者将看到一条或两条 JoinEmpty 消息，因此其注册者状态将为该属性的 IN。

在情况 b) 中，新成员发送两条 JoinIn 消息，因为注册器状态为 IN（存在另一个成员）；这将确保另一个成员注册该属性。

在情况 c) 中，新成员已看到其他两个参与者加入，因此它可以进入被动成员状态而根本不需要发送加入消息，并且根据类似的原理，它可以再次成为观察员而不需要发出离开消息。

H.1.3 注册器状态机的设计

注册器状态机完全是被动的，即它不传输 GARP PDU；它的工作只是记录属性的当前注册状态。IN 和 LV 状态表示当前注册；MT 状态表示取消注册。如果自上次看到 Leave 或 LeaveAll 以来已经过了 LeaveTime，并且没有收到 Join 作为响应，则会发生 LV 到 MT 的转换。

H.1.4 GARP状态机操作分析

本节展示一些 GARP 协议序列示例，以便

a) 说明 GARP 的正常运行；
b) 说明在某些故障情况下 GARP 的操作。

每个插图由以下元素组成：

c) 描述场景、相关拓扑和组成站/桥的文本和图表；

d) 显示场景中涉及的每个组件的状态机中发生的状态转换的序列。这些序列使用以下方式记录：*状态序列表；*表格的每一行显示某一时刻所有组件的组合状态，以及任何“系统状态”，即整个系统中的状态，但仅通过查看组件终端站/网桥的状态无法看到的状态。“系统状态”的一个例子是已发送但尚未在目的地收到的 Join PDU。表格中较早的行表示在时间序列中较早发生的组合状态。表格包含每种状态机类型的列，以及最后一列用于描述性注释。在适当的情况下，使用 12.8 中引入的缩写。

H.1.4.1 初始加入场景

拓扑：一个叶 LAN 段，连接一个站 A；参见图 H-1。

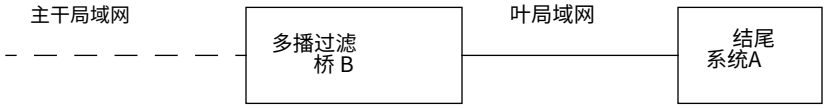


图 H-1 — 拓扑：叶 LAN 场景

第一种情况：A 请求属性 AT 的成员资格，而该属性 AT 迄今为止还没有成员；网桥 B 正常响应。交换期间未发生数据包丢失。表 H-1 分析了此情况。可以看出，在步骤 4 中，B 收到来自 A 的第一个 JoinEmpty 后，认为该属性已注册。

表 H-1 — 初始加入：无数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	画外音 — —	画外音 公吨 —	起始条件：属性 AT 在所有状态机中未注册。 无“系统”状态；A 没有注册器状态机。
2	申请人 注册主任 “系统”	副总裁 — —	画外音 公吨 —	A 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
3	申请人 注册主任 “系统”	AA — 杰伊（奥地利）	画外音 公吨 —	A发送JoinEmpty，进入AA。
4	申请人 注册主任 “系统”	AA — —	画外音 在 —	B 收到 JoinEmpty；注册器注册该属性 （IN），申请人状态不变。
5	申请人 注册主任 “系统”	质量保证 — 杰伊（奥地利）	画外音 在 —	A发送第二个JoinEmpty，进入QA。
6	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	第二个JoinEmpty对B的状态没有影响。

第二种情况：A 请求属性 AT 的成员资格，而该属性迄今为止还没有成员；B 未看到 A 的第一个 JoinEmpty PDU。表 H-2 分析了此情况。B 直到收到 A 的第二个 JoinEmpty PDU（步骤 6）后才注册该属性。

表 H-2 — 初始加入：第一个加入 PDU 的数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	画外音 — —	画外音 公吨 —	起始条件：属性 AT 在所有状态机中未注册。无“系统”状态；A 没有注册器状态机。
2	申请人 注册主任 “系统”	副总裁 — —	画外音 公吨 —	A 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
3	申请人 注册主任 “系统”	AA — 杰伊（奥地利）	画外音 公吨 —	A发送JoinEmpty，进入AA。
4	申请人 注册主任 “系统”	AA — —	画外音 公吨 —	B 无法看到 JoinEmpty；注册商和申请人状态未发生变化。
5	申请人 注册主任 “系统”	质量保证 — 杰伊（奥地利）	画外音 公吨 —	A发送第二个JoinEmpty，进入QA。
6	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	第二个 JoinEmpty 导致 B 的 Registrar 进入 IN 状态。

第三种情况：A 请求属性 AT 的成员资格，而该属性 AT 迄今尚未拥有任何成员；桥接器 B 未看到 A 的第二个 Join PDU。表 H-3 分析了此情况。从实际目的来看，与第一种情况没有区别。

表 H-3 — 初始加入：第二个加入 PDU 的数据包丢失

步	状态：	终点站A	桥 B	评论
1	申请人 注册主任 “系统”	画外音 — —	画外音 公吨 —	起始条件：属性 AT 在所有状态机中未注册。 无“系统”状态；A 没有注册器状态机。
2	申请人 注册主任 “系统”	副总裁 — —	画外音 公吨 —	A 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
3	申请人 注册主任 “系统”	AA — 杰伊（奥地利）	画外音 公吨 —	A发送JoinEmpty，进入AA。
4	申请人 注册主任 “系统”	AA — —	画外音 在 —	B 收到 JoinEmpty；注册器注册该属性（IN），申请人状态不变。
5	申请人 注册主任 “系统”	质量保证 — 杰伊（奥地利）	画外音 在 —	A发送第二个JoinEmpty，进入QA。
6	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	第二个JoinEmpty就丢失了，对B的状态没有任何影响。

H.1.4.2 最后离开的情景

拓扑：如 H.1.4.1 初始加入场景所述。

第一种情况：A 请求取消属性 AT 的成员资格注册，此前只有 A 是该属性的成员；网桥 B 正常响应。交换过程中未发生数据包丢失。表 H-4 分析了此情况。可以看出，B 认为该属性仍需注册，直到离开计时器到期。其他成员有两次机会声明该属性，因为在 B 最终取消该属性注册之前，会发送 Leave Empty 和 Empty 消息。

表 H-4 — 最后离开：无数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	起始条件：A 的申请人状态机和 B 的注册商状态机中的属性 AT 处于活动状态。A 没有注册商状态机。
2	申请人 注册主任 “系统”	洛杉矶 — —	画外音 在 —	A 的申请人从 GARP 应用程序接收 ReqLeave 原语。A 准备进入 LA 状态以发出 Leave 请求。
3	申请人 注册主任 “系统”	画外音 — LE（奥地利）	画外音 在 —	传输机会发生；A传输Leave Empty，进入VO状态。
4	申请人 注册主任 “系统”	画外音 — —	洛 路威酪轩 —	B 看到 Leave Empty。注册器启动 Leave Timer；申请者进入 LO 并等待传输机会。
5	申请人 注册主任 “系统”	画外音 — —	画外音 路威酪轩 吃）	B 向 AT 发送一个空消息，以提示所有剩余成员重新加入。
6	申请人 注册主任 “系统”	画外音 — —	画外音 路威酪轩 —	Empty 消息对 A 没有影响。
7	申请人 注册主任 “系统”	画外音 — —	画外音 公吨 —	离开计时器到期；B 的注册器取消注册该属性。

第二种情况：A 请求取消属性 AT 的成员资格注册，而此前只有 A 是该属性 AT 的成员；桥接器 B 未看到 A 的 Leave Empty PDU。表 H-5 分析了此情况。平均而言，此情况需要 B 半个 Leave All Period 加上 Leave Period 才能最终取消该属性的注册。

表 H-5 — 最后离开：第一个离开 PDU 的数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	起始条件：A 的申请人状态机和 B 的注册商状态机中的属性 AT 处于活动状态。A 没有注册商状态机。
2	申请人 注册主任 “系统”	洛杉矶 — —	画外音 在 —	A 的申请人从 GARP 应用程序接收 ReqLeave 原语。A 准备进入 LA 状态以发出 Leave 请求。
3	申请人 注册主任 “系统”	画外音 — LE（奥地利）	画外音 在 —	传输机会发生；A传输Leave Empty，进入VO状态。
4	申请人 注册主任 “系统”	画外音 — —	画外音 在 —	B 未能看到 Leave Empty。
5	申请人 注册主任 “系统”	画外音 — —	洛 路威酪轩 全部离开	B 发送 LeaveAll 消息。B 的注册器进入 LV 并启动 Leave Timer。申请人进入 LO。
6	申请人 注册主任 “系统”	画外音 — —	洛 路威酪轩 —	全部离开消息对 A 没有影响。
7	申请人 注册主任 “系统”	画外音 — —	画外音 路威酪轩 吃）	发生一次传输机会；B 为 AT 传输一个空消息，申请人返回 VO。
8	申请人 注册主任 “系统”	画外音 — —	画外音 路威酪轩 —	Empty 消息对 A 没有影响。
9	申请人 注册主任 “系统”	画外音 — —	画外音 公吨 —	离开定时器到期。B 的注册器取消注册该属性。

第三种情况：严格来说，还有进一步的单数据包丢失情况需要考虑，包括 B 的 Leave All 和 Empty 消息的丢失，但顺序分析与第二种情况相同，因为无论是哪种情况，Empty 或 Leave All 消息都不会对 A 的状态产生影响。

H.1.4.3 离开/重新加入场景—单个成员

拓扑：如 H.1.4.1 初始加入场景所述。

第一种情况：A 是属性 AT 的成员，没有其他成员。桥接器 B 发出 Leave All 消息；A 重新加入。交换期间未发生数据包丢失。表 H-6 分析了此情况。可以看出，A 始终保持注册状态。如果 B 在 A 发送第一个 Join Empty 之前（步骤 3 和步骤 4 之间）成功发送了 Empty，则会出现此情况的轻微变化；但是，这对步骤 4 到步骤 7 没有影响。

表 H-6 — 单个成员全部离开/重新加入：无数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	起始条件：A 的申请人状态机和 B 的注册商状态机中的属性 AT 处于活动状态。A 没有注册商状态机。
2	申请人 注册主任 “系统”	质量保证 — —	洛 路威酪轩 全部离开	B 发送 LeaveAll 消息。这会导致 B 进入申请人和注册员的 LO 和 LV 状态，并启动 Leave Timer。
3	申请人 注册主任 “系统”	副总裁 — —	洛 路威酪轩 —	A变得非常焦虑，等待机会发送JoinEmpty。
4	申请人 注册主任 “系统”	AA — 杰伊（奥地利）	洛 路威酪轩 —	A 传输 JoinEmpty，并且变得不那么焦虑了。
5	申请人 注册主任 “系统”	AA — —	画外音 在 —	B 看到加入，注册会员资格并重新进入 VO。
6	申请人 注册主任 “系统”	质量保证 — 杰伊（奥地利）	画外音 在 —	A 传输第二个 JoinEmpty，然后变为 Quiet。
7	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	第二个 Join Empty 对 B 没有影响。

第二种情况：A 是属性 AT 的成员，该属性 AT 没有其他成员。桥接器 B 发出 Leave All 消息；A 未看到该消息，直到 B 发出 Empty 消息后才重新加入。表 H-7 分析了此情况。可以看出，尽管 LeaveAll 消息丢失，A 仍始终保持注册状态。

表 H-7 — 单个成员离开/重新加入：离开时数据包丢失所有 PDU

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	起始条件：A 的申请人状态机和 B 的注册商状态机中的属性 AT 处于活动状态。A 没有注册商状态机。
2	申请人 注册主任 “系统”	质量保证 — —	洛 路威酪轩 全部离开	B 发送 LeaveAll 消息。这会导致 B 进入申请人和注册员的 LO 和 LV 状态，并启动 Leave Timer。
3	申请人 注册主任 “系统”	质量保证 — —	洛 路威酪轩 —	A 没有看到 LeaveAll。
4	申请人 注册主任 “系统”	质量保证 — —	画外音 路威酪轩 吃）	对于 B 来说，出现了一个传输机会；它发送了一个 Empty。
5	申请人 注册主任 “系统”	弗吉尼亚州 — —	画外音 路威酪轩 —	A 看到空虚，变得非常焦虑。
6	申请人 注册主任 “系统”	AA — 杰伊（奥地利）	画外音 路威酪轩 —	对于 A 来说，有一个传输机会出现，它传输一个 JoinEmpty，并且变为 Anxious。
7	申请人 注册主任 “系统”	AA — —	画外音 在 —	B 看到 JoinEmpty，并重新注册该属性。
8	申请人 注册主任 “系统”	质量保证 — 杰伊（奥地利）	画外音 在 —	A 传输第二个 JoinEmpty，然后变为 Quiet。
9	申请人 注册主任 “系统”	AA — —	画外音 在 —	第二个 JoinEmpty 没有效果。

第三种情况：A 是属性 AT 的成员，该属性 AT 没有其他成员。桥接器 B 发出 Leave All 消息；A 用 JoinEmpty 消息进行响应，但 B 看不到该消息，因此 B 发出了一条 Empty 消息。表 H-8 分析了这种情况。可以看出，尽管丢失了第一个 JoinEmpty 消息，A 仍始终保持注册状态。Empty 消息的作用是重置 A 的 Joins 计数，因此在 A 变为 Quiet 之前，会发送三条 Join Empty 消息。

表 H-8 — 单个成员离开/重新加入：首次加入 PDU 时的数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	质量保证 — —	画外音 在 —	起始条件：A 的申请人状态机和 B 的注册商状态机中的属性 AT 处于活动状态。A 没有注册商状态机。
2	申请人 注册主任 “系统”	质量保证 — —	洛 路威酪轩 全部离开	B 发送 LeaveAll 消息。这会导致 B 进入申请人和注册员的 LO 和 LV 状态，并启动 Leave Timer。
3	申请人 注册主任 “系统”	副总裁 — —	洛 路威酪轩 —	A变得非常焦虑，等待机会发送JoinEmpty。
4	申请人 注册主任 “系统”	AA — 杰伊（奥地利）	洛 路威酪轩 —	A 传输 JoinEmpty，并且变得不那么焦虑了。
5	申请人 注册主任 “系统”	AA — —	洛 路威酪轩 —	B 未能看到 Join Empty。
6	申请人 注册主任 “系统”	AA — —	画外音 路威酪轩 吃）	对于 B 来说，出现了一个传输机会；它发送了一个 Empty。
7	申请人 注册主任 “系统”	弗吉尼亚州 — —	画外音 路威酪轩 —	A 看到空虚，变得非常焦虑。
8	申请人 注册主任 “系统”	AA — 杰伊（奥地利）	画外音 路威酪轩	A 传输第二个 JoinEmpty，并且变得焦虑。
9	申请人 注册主任 “系统”	AA — —	画外音 在 —	B 看到 JoinEmpty，并重新注册该属性。
10	申请人 注册主任 “系统”	质量保证 — 杰伊（奥地利）	画外音 在	A 传输第三个 JoinEmpty，并且变为 Quiet。
11	申请人 注册主任 “系统”	AA — —	画外音 在 —	第三个JoinEmpty没有任何效果。

H.1.4.4 主干局域网初始加入场景

拓扑：一个主干 LAN 段，连接两个网桥 A 和 B；参见图 H-2。

这些场景仅关注连接到主干网段的桥接端口的行为。

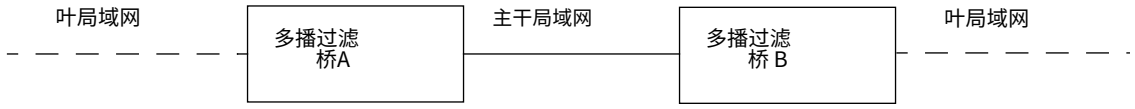


图 H-2 — 拓扑：主干 LAN 场景

第一种情况：A 请求主干网上属性 AT 的成员资格，这是由于申请人在其叶 LAN 上进行了初始注册。B 稍后注册了相同的属性。两个网桥均正常响应。交换期间未发生数据包丢失。表 H-9 分析了此情况。可以看出，B 在收到来自 A 的第一个 Join 后认为该属性已注册；同样，A 在收到 B 的第一个 JoinIn 后认为该属性已注册。

表 H-9 — 主干网初始加入：无数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	画外音 公吨 —	画外音 公吨 —	起始条件：属性 AT 未在所有状态机中注册。 无“系统”状态。
2	申请人 注册主任 “系统”	副总裁 公吨 —	画外音 公吨 —	A 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
3	申请人 注册主任 “系统”	AA 公吨 杰伊（奥地利）	画外音 公吨 —	A发送JoinEmpty，进入AA。
4	申请人 注册主任 “系统”	AA 公吨 —	画外音 在 —	B 收到 JoinEmpty；注册器注册该属性 （IN），申请人状态不变。
5	申请人 注册主任 “系统”	AA 公吨 —	副总裁 在 —	B 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
6	申请人 注册主任 “系统”	AA 公吨 —	AA 在 吉伊（AT）	B 发出 JoinIn。
7	申请人 注册主任 “系统”	质量保证 在 —	AA 在 —	B 的 JoinIn 导致 A 注册该属性，并且还抑制了 A 的第二次 Join。
8	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 吉伊（AT）	B 发出第二个 JoinIn 并且变得安静。
9	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 —	B 的第二次 JoinIn 没有效果。

第二种情况：由于申请人在其叶 LAN 上进行了初始注册，A 请求主干网上的属性 AT 成员资格。桥接器 B 未看到 A 的第一个加入 PDU。A 在加入时间之后发出第二个加入。B 随后加入同一属性。表 H-10 分析了此情况。B 直到看到 A 的第二个加入（A 首次加入尝试后一个加入时间）才为 A 注册属性。

表 H-10 — 主干网初始加入：首次加入 PDU 时的数据包丢失

步	状态：	一个	乙	评论
1	申请人 注册主任 “系统”	画外音 公吨 —	画外音 公吨 —	起始条件：属性 AT 未在所有状态机中注册。 无“系统”状态。
2	申请人 注册主任 “系统”	副总裁 公吨 —	画外音 公吨 —	A 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
3	申请人 注册主任 “系统”	AA 公吨 杰伊（奥地利）	画外音 公吨 —	A发送JoinEmpty，进入AA。
4	申请人 注册主任 “系统”	AA 公吨 —	画外音 公吨 —	B 未能看到 JoinEmpty。
5	申请人 注册主任 “系统”	质量保证 公吨 杰伊（奥地利）	画外音 公吨 —	A发送第二个JoinEmpty，进入QA。
6	申请人 注册主任 “系统”	质量保证 公吨 —	画外音 在 —	B 收到 JoinEmpty；注册器注册该属性 （IN），申请人状态不变。
7	申请人 注册主任 “系统”	质量保证 公吨 —	副总裁 在 —	B 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
8	申请人 注册主任 “系统”	质量保证 公吨 —	AA 在 吉伊（AT）	B 发出 JoinIn。
9	申请人 注册主任 “系统”	质量保证 在 —	AA 在 —	B 的 JoinIn 导致 A 注册该属性。
10	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 杰伊（奥地利）	B 发出第二个 JoinIn 并且变得安静。
11	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 —	B 的第二次 JoinIn 没有效果。

第三种情况：由于申请人在其叶 LAN 上进行了初始注册，A 请求主干网上的属性 AT 成员资格。网桥 B 未看到 A 的第一个加入 PDU，但自己针对同一属性发出了加入。表 H-11 分析了此情况。直到 A 第二次尝试加入后，B 才为 A 注册该属性。

表 H-11 — 主干网初始加入：两个网桥同时加入，但出现数据包丢失
首次加入

步	状态：	桥A	桥 B	评论
1	申请人 注册主任 “系统”	画外音 公吨 —	画外音 公吨 —	起始条件：属性 AT 在所有状态机中未注册。无成员。无“系统”状态。
2	申请人 注册主任 “系统”	副总裁 公吨 —	画外音 公吨 —	A 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
3	申请人 注册主任 “系统”	AA 公吨 杰伊（奥地利）	画外音 公吨 —	A发送JoinEmpty，进入AA。
4	申请人 注册主任 “系统”	AA 公吨 —	画外音 公吨 —	B 未能看到 JoinEmpty。
5	申请人 注册主任 “系统”	AA 公吨 —	副总裁 公吨 —	B 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
6	申请人 注册主任 “系统”	AA 公吨 —	AA 公吨 杰伊（奥地利）	B发送JoinEmpty；并进入AA。
7	申请人 注册主任 “系统”	弗吉尼亚州 在 —	AA 公吨 —	A 收到 JoinEmpty。这使其进入 VA，并注册属性。
8	申请人 注册主任 “系统”	AA 在 吉伊（AT）	AA 公吨 —	A发送JoinIn，进入AA。
9	申请人 注册主任 “系统”	AA 在 —	质量保证 在 —	B 看到 JoinIn ；它注册该属性，并且来自 B 的进一步连接受到抑制。
10	申请人 注册主任 “系统”	质量保证 在 杰伊（奥地利）	质量保证 在 —	A 发出第二个 JoinIn 并且变得安静。
11	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 —	A 的第二次 JoinIn 对 B 没有影响。

H.1.4.5 共享媒体 LAN 段场景

拓扑：一个主干 LAN 段，连接两个网桥 A 和 B；参见图 H-3。

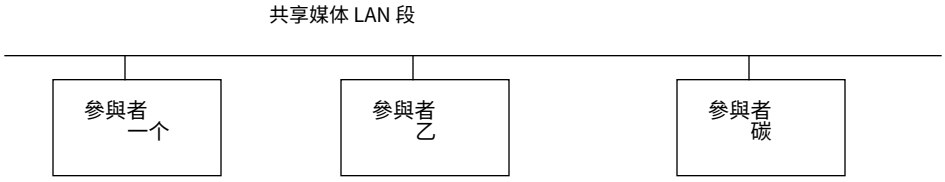


图 H-3 — 拓扑：共享媒体 LAN 段场景

第一种情况：说明 H.1.2 中提到的要点，即如果给定 LAN 段上已经有两个或更多成员，则其他成员可以加入和离开，而无需发出任何 GARP 消息。三个参与者连接到该段。A 和 B 为属性 AT 加入（与表 H-9 相同的序列）。随后，C 可以加入和离开，而无需发出任何 GARP 消息。表 H-12 中分析了此情况。

表 H-12 - 共享媒体：第三方可以加入/离开而无需发送 GARP 消息

步	状态：	一个	乙	碳	评论
1	申请人 注册主任 “系统”	画外音 公吨 —	画外音 公吨 —	画外音 公吨 —	起始条件：属性 AT 未在所有状态机中注册。 无“系统”状态。
2	申请人 注册主任 “系统”	副总裁 公吨 —	画外音 公吨 —	画外音 公吨 —	A 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
3	申请人 注册主任 “系统”	AA 公吨 杰伊（奥地利）	画外音 公吨 —	画外音 公吨 —	A发送JoinEmpty，进入AA。
4	申请人 注册主任 “系统”	AA 公吨 —	画外音 在 —	画外音 在 —	B和C收到JoinEmpty；注册器注册属性（IN），申请人状态不变。
5	申请人 注册主任 “系统”	AA 公吨 —	副总裁 在 —	画外音 在 —	B 的申请者从 GARP 申请者处收到属性 AT 的 ReqJoin 原语。这使其进入 VP 并等待传输机会。
6	申请人 注册主任 “系统”	AA 公吨 —	AA 在 吉伊（AT）	画外音 在 —	B 发出 JoinIn。
7	申请人 注册主任 “系统”	质量保证 在 —	AA 在 —	AO 在 —	B 的 JoinIn 使得 A 注册该属性，同时也抑制了 A 的第二次 Join，也使得 C 的 Applicant 进入 AO（Anxious Observer）状态。
8	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 吉伊（AT）	AO 在 —	B 发出第二个 JoinIn 并且变得安静。
9	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 —	质量保证 在 —	B 的第二次 JoinIn 对 A 没有影响，但是导致 C 进入 QO（Quiet Observer）状态。
10	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 —	量子点 在 —	C 的申请者从 GARP 应用程序接收到属性 AT 的 ReqJoin 原语。由于 C 已看到 AT 的两个 JoinIn，因此它能够直接进入 QP（安静/被动成员）状态，而无需发出任何 GARP 消息。
11	申请人 注册主任 “系统”	质量保证 在 —	质量保证 在 —	质量保证 在 —	C 的申请者从 GARP 申请中收到属性 AT 的 ReqLeave 原语。由于 C 的申请者加入 AT 时未发出任何消息，因此它可以直接进入 QO（安静/观察者）状态离开，而无需发出任何 GARP 消息。

第二种情况：显示 LAN 段上存在两个以上参与者的情况下的“最后离开”序列。 A 已声明属性 AT； B 和 C 已在其注册器状态机中注册 AT。 A 离开。 协议交换正常进行。 表 H-13 分析了此情况。

注意 — 第二种情况主要是为了说明最后一个成员离开所需的时间与所涉及的参与者数量无关。这是 GARP 早期版本中存在的问题，其中 LV 状态（或同等状态）中的注册器收到空消息会导致离开计时器重新启动。在所述的协议中，最后一个成员离开后取消注册所需的时间等于一个离开时间，无论参与者数量有多少。

表 H-13 — 共享媒体：三个参与者的离开顺序

步	状态：	一个	乙	碳	评论
1	申请人 注册主任 “系统”	质量保证 公吨 —	画外音 在 —	画外音 在 —	起始条件：属性 AT 在 A 的申请人以及 B 和 C 的注册商状态机中处于活动状态。
2	申请人 注册主任 “系统”	洛杉矶 公吨 —	画外音 在 —	画外音 在 —	A 的申请人从 GARP 应用程序接收 ReqLeave 原语。A 准备进入 LA 状态以发出 Leave 请求。
3	申请人 注册主任 “系统”	画外音 公吨 LE（奥地利）	画外音 在 —	画外音 在 —	传输机会发生；A传输Leave Empty，进入VO 状态。
4	申请人 注册主任 “系统”	画外音 公吨 —	洛 路威酪轩 —	洛 路威酪轩 —	B 和 C 看到 Leave Empty。他们的注册器启动 eave Timer；他们的申请人进入 LO 并等待传输机会。
5	申请人 注册主任 “系统”	画外音 公吨 —	画外音 路威酪轩 吃）	洛 路威酪轩 —	出现了 B 的传输机会；它向 AT 传输一个空消息，以提示任何剩余成员重新加入。
6	申请人 注册主任 “系统”	画外音 公吨 —	画外音 路威酪轩 —	画外音 路威酪轩 —	Empty 消息对 A 没有影响，但抑制了 C 传输 mpty 的意图；C 进入 VO。
7	申请人 注册主任 “系统”	画外音 公吨 —	画外音 公吨 —	画外音 公吨 —	离开计时器在 B 和 C 中到期；它们的注册器都取消了该属性的注册。

H.2 用户优先级和流量类别

本节记录了选择用户优先级值和流量类别之间的默认映射背后的一些原因，如表 7-2 所示。

H.2.1 目标

分配用户优先级语义和使用有限数量的支持队列的方法有很多种。本附件并不试图调查所有可能性，而是旨在列出一组合理的默认值，供在典型的 LAN Bridged 环境中使用，以提供集成服务，即满足本标准的原始目标。

此标准允许管理优先级、队列映射和队列服务规则，以最好地支持用户的目标。然而，人们普遍认识到，一组众所周知且易于理解的默认设置将极大地促进即插即用互通和集成服务的部署。因此，这里提倡的默认设置既是为了支持 IETF（特别是 ISSLL 和 IS802）中新兴的集成服务映射工作，也是为了在不管理网桥的情况下提供有用的服务。

H.2.2 流量类型

对应用程序的服务质量需求和生成的网络流量的完整描述以及流量本身的特征描述可能非常复杂 — 肯定过于复杂，无法用简单的 0 到 7 来表示。流量分类的务实目标是从根本上简化需求，以保持网桥的高速、低成本处理特性。在边际上，潜在的带宽效率被换成了简单性 — 从历史上看，这在 LAN 中是一个不错的决定。

以下列出的流量类型似乎得到了广泛的支持，每种流量类型都可以通过与其他流量类型的简单隔离而受益：

a) 网络控制——特点是需要“到达那里”来维护和支持网络基础设施。

b)“语音”——特点是延迟小于 10 毫秒，因此抖动最大（通过单个校园的 LAN 基础设施单向传输）。

c)“视频”——特点是延迟少于 100 毫秒。
d) 受控负载——重要的商业应用受到某种形式的“准入控制”，一方面是对网络需求的预先规划，另一方面是在数据流启动时为每个数据流预留带宽。

e) 卓越努力——或“CEO 的最大努力”，信息服务组织向其最重要的客户提供的最大努力类型的服务。
f) 尽力而为——我们今天所知的 LAN 流量。
g) 背景——网络上允许的批量传输和其他活动，但不应影响其他用户和应用程序对网络的使用。

H.2.3 我们管理什么？

用户优先级和流量类别有助于管理

a）延迟，以支持新的应用程序；
b) 吞吐量，以满足以各种流量的带宽为中心的服务水平协议。

通过不同的流量分类和桥接队列，可以在更高的网络负载水平下支持延迟和带宽保证。如果类别较少，则重点是满足延迟要求——在突发数据环境中，如果没有不同的流量分类，保证低于 10 毫秒的延迟所需的带宽盈余是不经济的。随着可使用的流量类别数量的增加，重点可以转移到管理吞吐量上。

本标准中定义的简单默认队列服务策略，严格优先级，支持延迟管理。带宽共享的主动管理必然需要一些桥接管理，以确定份额等。

H.2.4 流量类型到流量类别的映射

表 H-14 描述了随着 Bridge 队列数量的增加，上面介绍的流量类型的分组。每种类型的分组显示为 {*区分类型*, 类型, 类型, . . .}。“区分类型”在 Bridge 中不会以任何不同方式处理，但在这里以斜体显示，以说明对于任何给定数量的队列，哪些流量类型推动了类型到类的分配。

表 H-14 — 流量类型到流量类别的映射

数量 队列	流量类型
1	{*尽力而为*、出色的努力、背景、语音、受控负载、视频、网络控制}
2	{*尽力而为*，出色的努力，背景} {*嗓音*、受控负载、视频、网络控制}
3	{*尽力而为*，出色的努力，背景} {*受控负载*，视频} {*嗓音*、网络控制}
4	{*背景*} {*尽力而为*，出色的努力} {*受控负载*，视频} {*嗓音*、网络控制}
5	{*背景*} {*尽力而为*，出色的努力} {*受控负载*} {*视频*} {*嗓音*、网络控制}
6	{*背景*} {*尽力而为* {*出色的努力*} {*受控负载*} {*视频*} {*嗓音*、网络控制}
7	{*背景*} {*尽力而为*} {*出色的努力*} {*受控负载*} {*视频*} {*嗓音*} {*网络控制*}

此分组被提议作为默认用户优先级到流量类别的映射，流量类型与用户优先级数字的对应关系如下所示。它旨在与默认队列处理配合使用。其余用户优先级值（仅讨论了七个而不是八个流量类型）也在优先级层次结构中分配。

随着越来越多的类别出现，流量类型逐步细分的背后逻辑如下：

a) 使用单一队列时，没有选择，一切功能均与符合 ISO/IEC 10038: 1993 的网桥一样。如果要识别一种不同的流量类型，那就是尽力而为。
b) 为了在出现突发尽力而为数据的情况下支持集成服务（语音、视频和数据），必须隔离所有时间关键型流量。此外，还需要将要接收优质服务且在准入控制下运行的其他流量与不受控制的流量分开。优先流量的数量将受到语音低延迟（本地 LAN 基础设施的单向延迟低于 10 毫秒）需求的限制，语音将成为附加队列的定义类型。

c) 最好使用另一个队列来将受控负载（即具有准入控制或策略约束的良好流量）与对延迟极为敏感的语音流量分开。与使用两个队列相比，这允许更多受控负载（和视频）流量，同时仍允许满足语音延迟。但是，新队列中的所有流量仍必须满足交互式视频延迟（如果存在任何此类流量），因此可能仍会对良好流量的吞吐量进行人为的低限制。

d) 到目前为止，首先使用附加队列来解决延迟问题，然后使用附加队列来恢复延迟限制较少的良好流量的吞吐量。现在，对于主要带宽用户来说，这已基本实现，重点可以转移到区分不同类型的不受控制的突发流量，根据业务重要性进行区分。第一步是通过隔离背景流量来处理高吞吐量背景流量，这可能会导致正常尽力而为（和卓越努力）流量的 TCP 窗口和计时器变慢。

e) 将下一个队列分配给（交互式）视频、延迟和抖动小于 100 毫秒的流量，将受控负载流量从延迟限制中解放出来，提高可实现的吞吐量。
f) 下一个队列致力于提供一点额外的工作安全性，将表现不够好而无法归类为受控负载的关键业务应用程序分离到它们自己的优秀努力类别中。

g) 最后，网络控制可以与语音分离，尽管这种做法的好处不太可能在默认队列处理中感受到，因为网络控制可能不像语音那样对延迟敏感（但可能比后台更敏感），但需要更高的交付保证。

h) 未显示八个队列，因为仅显示了七种不同类型的流量。从目前定义默认值的角度来看，为尽力而为分配额外的优先级以支持突发数据应用程序的带宽共享管理似乎是合理的。

表 H-15 显示了流量类型与用于选择表 7-2 中的默认值的 user_priority 值的对应关系，并为流量类型定义了一组首字母缩写词：

表 H-15 — 流量类型缩写

用户优先级	缩写	流量类型
1	黑金	背景
2	—	空闲的
0（默认）	是	尽力而为
3	电子工程	出色的努力
4	氯	受控负载
5	六	“视频”，延迟和抖动小于 100 毫秒
6	画外音	“语音”，延迟和抖动小于 10 毫秒
7	数控	网络控制

表 H-16 使用表 H-15 中定义的首字母缩略词，压缩了表 H-14 中显示的队列的流量类型列表，并且仅显示给定数量队列的定义流量类型。

表 H-16 — 定义流量类型

数量 队列	定义流量类型
1	是
2	是 画外音
3	是 氯 画外音
4	黑金 是 氯 画外音
5	黑金 是 氯 六 画外音
6	黑金 是 电子工程 六 画外音
7	黑金 是 电子工程 六 画外音 数控
8	黑金 — 是 电子工程 六 画外音 数控

H.2.5 流量类型和用户优先级值

令牌环站用于传输的默认用户优先级为 0，这是 ISO/IEC 10038：1993 兼容网桥对当前从以太网接收的帧所采用的默认值。更改此默认值会导致一些混乱，并且可能缺乏互操作性。同时，当今的默认流量类型肯定是尽力而为。解决这一困境的建议是继续将零用于默认用户优先级和尽力而为数据传输，并将背景、备用值和卓越努力分别与用户优先级值 1、2 和 3 相关联。这意味着值 1 和 2 实际上传达的优先级低于 0。