

RFC826—An Ethernet Address Resolution Protocol or Converting Network Protocol
Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware
以太网地址转换协议或转换网络协议地址为 48 比特以太网地址用于在以太网硬件上传输
组织：中国互动出版网 (<http://www.china-pub.com/>)

RFC 文档中文翻译计划 (<http://www.china-pub.com/compters/emook/aboutemook.htm>)

E-mail: ouyang@china-pub.com

译者：沈进 (simon_shen shen_jin@263.net)

译文发布时间：2001-9-6

版权：本中文翻译文档版权归中国互动出版网所有。可以用于非商业用途自由转载，但必须保留本文档的翻译及版权信息。

Network Working Group
Request For Comments: 826

David C. Plummer
(DCP@MIT-MC)
November 1982

以太网地址转换协议或转换网络协议地址 为 48 比特以太网地址用于在以太网硬件上传输

(RFC826—An Ethernet Address Resolution Protocol or Converting Network Protocol
Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware)

目录

1.摘要	1
2.说明	2
3.问题	2
4.动机	2
5.定义	3
6.包格式	3
7.发包	3
8.收包	4
9.为什么这么做.....	5
10.网络监控和排错.....	6
11.一个例子.....	6
12.相关情况.....	7

1. 摘要

通过路由机制，协议 P 在发送主机 S 上的实现决定了 S 需要传输到目标主机 T，而 T 连在和 S 相连的 10 兆以太网电缆上。实际传输以太网包必须产生一个 48 比特以太网地址。主机的协议 P 地址并不总是和相应的以太网地址兼容(长度或值不同)。现在这个协议允许动态地发布

RFC826—An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware
以太网地址转换协议或转换网络协议地址为 48 比特以太网地址用于在以太网硬件上传输信息，这些信息可用来构造转换协议 P 地址空间内的地址 A 为 48 比特以太网地址的一张表。
允许在非 10 兆以太网硬件使用的协议已经被综合总结，无线电网络就是这种硬件。

[这篇RFC的目的是提出一种转换协议地址(例如IP地址)为本地网络地址(例如以太网地址)的方法。这个问题现在受到 ARPA Internet 社区的普遍关注，这里提出的方法仅供读者参考，并不是 Internet 标准的描述。]

2. 说明

这个协议起初是为 DEC/Intel/Xerox 的 10 兆以太网设计的，现在已允许用在其它类型的网络上。许多讨论将直接针对 10 兆以太网。总之，合适的话将遵循以太网的特定讨论。

DOD Internet 协议将作为 Internet 的规范被参考。

这里用到的数字，在以太网标准中是高位字节在前的，这和例如 PDP-11，VAX 等机器的字节编址相反，因此对下面描述的操作字段(ar\$op)必须特别小心。

需要处理硬件名字空间已达成一致。直到官方认可，请求可发送到

David C. Plummer

Symbolics, Inc.

243 Vassar Street

Cambridge, Massachusetts 02139

或发邮件到 DCP@MIT-MC。

3. 问题

世界总的来说是杂乱的，同时网络增加了这种杂乱。几乎在网络架构的每一层，都有几个潜在的协议可以使用。例如在高一点的层次有用于远程登录的 TELNET 和 SUPDUP。低一点的有 CHAOS, DOD TCP, Xerox, BSP 或 DECnet 等可靠字节流协议。甚至在与硬件较接近的逻辑传输层也有 CHAOS, DOD Internet, Xerox PUP, DECnet 等协议。10 兆以太网通过使用以太网包头中的类型字段来使这些协议(而且更多)能在一根电缆上共存。然而，10 兆以太网在物理电缆上需要 48 比特地址，而大多数协议地址不是 48 比特，它们并不需要与硬件的 48 比特以太网地址有什么关系。例如 CHAOS 的地址是 16 比特，DOD Internet 的地址是 32 比特，Xerox PUP 的地址是 8 比特。这就需要有一个协议来动态地区分一个<协议，地址>对一个 48 比特以太网地址的对应关系。

4. 动机

随着更多的制造商提供遵循 DEC, Intel 和 Xerox 发布的规范的接口产品，10 兆以太网的使用也在增加。随着使用的增加，为这个接口开发的软件也越来越多。有两个选择：(1) 每个实现者用自己的方法做某种形式的地址转换；(2) 每个实现者使用统一标准，这样代码

RFC826—An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware
以太网地址转换协议或转换网络协议地址为 48 比特以太网地址用于在以太网硬件上传输
可以不加修改的移植到其它系统。这个建议试图建立一个标准。

5. 定义

下面的定义是作为对填在以太网包头的类型字段的值的参考。

ether_type\$XEROX_PUP,
ether_type\$DOD_INTERNET,
ether_type\$CHAOS,

一个新的值

ether_type\$ADDRESS_RESOLUTION

再定义以下的值(后面讨论)

ares_op\$REQUEST (= 1, 高位字节在前) 和
ares_op\$REPLY (= 2), 和
ares_hrd\$Ethernet (= 1).

6. 包格式

为了把<协议, 地址>对映射到 48 比特以太网地址用于传输, 需要一个体现地址转换协议的包格式。包格式如下所示。

以太网传输层(并不是用户需要访问的):

48 比特: 目的以太网地址

48 比特: 源以太网地址

16 比特: 协议类型 = ether_type\$ADDRESS_RESOLUTION

以太网包数据:

16 比特: (ar\$hrd) 硬件地址空间(例如: Ethernet, Packet Radio Net。)

16 比特: (ar\$pro) 协议地址空间。对于以太网硬件, 它属于类型字段 ether_type\$<协议>的集合

8 比特: (ar\$hln) 每种硬件地址的字节长度

8 比特: (ar\$pln) 每种协议地址的字节长度

16 比特: (ar\$op) 操作码 (ares_op\$REQUEST | ares_op\$REPLY)

n 字节: (ar\$sha) 源硬件地址, n 从 ar\$hln 字段得到

m 字节: (ar\$spa) 源协议地址, m 从 ar\$pln 字段得到

n 字节: (ar\$tha) 目的硬件地址(如果知道的话)

m 字节: (ar\$tpa) 目的协议地址。

7. 发包

当网络层往下传来一个包, 路由将决定这个包下一跳的协议地址, 并根据目的协议地

以太网地址转换协议或转换网络协议地址为 48 比特以太网地址用于在以太网硬件上传输地址决定用哪个硬件进行传输。在 10 兆以太网需要地址转换。一些更低的层次(像硬件驱动层)必须咨询地址转换模块(也许在以太网支持模块中实现)把<协议类型, 目的协议地址>对转换成 48 比特以太网地址。地址转换模块试图在一个表中寻找这个对。如果找到, 则返回相

应的 48 比特以太网地址给调用者(硬件驱动层)。如果找不到, 也许应通知调用者这个包正在被丢弃(假定包会被高层重传), 同时发出一个类型字段为 ether_type\$ADDRESS_RESOLUTION 的以太网包。地址转换模块在 ar\$hrd 字段中填 ares_hrd\$Ethernet, 在 ar\$pro 字段中填要被转换的协议类型, 在 ar\$hln 字段中填 6(48 比特以太网地址字节数), 在 ar\$pln 字段中填该协议地址的字节数, 在 ar\$op 字段中填 ares_op\$REQUEST, 在 ar\$sha 字段中填自己的 48 比特以太网地址, 在 ar\$spa 字段中填自己的协议地址, 在 ar\$tpa 字段中填要访问机器的协议地址。不能在 ar\$tha 字段中填特殊的值, 因为它的值正是要得到的。如果实现上简单的话, ar\$tpa 字段可以填硬件的广播地址(在 10 兆以太网上所有机器)。根据原先的路由机制, 这个包将被广播到所有在以太网电缆上的工作站。

8. 收包

当收到地址转换包时, 收包模块把它送到运行类似下面算法的地址转换模块。条件不成立意味着处理结束, 并丢弃包。

? 我用 ar\$hrd 字段中的硬件吗?

是的: (几乎肯定)

[检查 ar\$hln 的硬件地址长度(可选)]

? 我用 ar\$pro 字段中的协议吗?

是的:

[检查 ar\$pln 的协议地址长度(可选)]

Merge_flag := false

如果<协议类型, 发送者协议地址>对在我的转换表中, 用包中的发送者硬件地址更新表, 并把 Merge_flag 设成 true。

? 我是目的协议地址吗?

是的:

如果 Merge_flag 是 false, 在转换表中加入三元组<协议类型, 发送者协

议地址, 发送者硬件地址>。

? 操作码是 ares_op\$REQUEST 吗? (现在看操作码)

是的:

交换硬件和协议字段, 把本地硬件和协议地址填在发送者字段中。

在 ar\$op 字段中填 ares_op\$REPLY。然后从收到包的硬件上把这个包发送到目的硬件地址。

注意到在检查操作码之前, <协议类型, 发送者协议地址, 发送者硬件地址>三元组就被加入转换表中。这是建立在通信是双向的假设上的, 如果 A 有某种理由与 B “交谈”, B 也会有某种理由与 A “交谈”。还注意到如果<协议类型, 发送者协议地址>对已存在表项中, 新的硬件地址将覆盖旧的。相关情况给出了这样做的动机。

总结: ar\$hrd 和 ar\$hln 字段使非 10 兆以太网可以使用这个协议和包格式。对于 10 兆

RFC826——An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48-bit Ethernet Address for Transmission on Ethernet Hardware
以太网地址转换协议或转换网络协议地址为 48 比特以太网地址用于在以太网硬件上传输以太网, $\langle ar\$hrd, ar\$hln \rangle$ 就是 $\langle 1, 6 \rangle$ 。对于其它硬件网络, $ar\$prozi$ 字段也许不再对应以太网类型字段, 但会和地址转换要看的协议有关。

9. 为什么这么做

定期广播并不是所期望的, 假设一个以太网上有 100 台主机, 每隔 10 分钟广播地址转换信息(可能通过参数设置), 这样每隔 6 秒钟就有一个包。这完全合理, 但有用吗? 工作站一般不会互相通信(因此转换表中有 100 个没用的表项), 它们主要和大型机, 文件服务器或网桥通信, 而仅和很少数量的主机通信(例如交互谈话)。本文描述的协议只在需要时发送信息, 并且每台机器每次启动时只发一次。

这种包格式不允许在一个包中进行多于一个的转换。这是为了简单。如果复杂的话, 包将较难被分析, 并且很多信息是没用的。想想一个有四种协议的网桥告诉工作站四个协议地址, 而其中三个工作站从来都不会用到。

这种包格式允许应答包重用请求包的存储空间, 应答包和请求包具有相同的长度, 有些字段也相同。

硬件字段($ar\$hrd$)的值来自一个列表。现在只有为 10 兆以太网定义的一个值($ares_hrd \$Ethernet = 1$)。已经在讨论在 Packet Radio Networks 上使用这个协议, 这需要为希望使用这个协议的其它硬件介质分配值。

对于 10 兆以太网, 协议字段($ar\$pro$)的值来自集合 $ether_type\$$, 这是对已分配的协议类型的自然重用。把它和操作码($ar\$op$)结合起来, 将有效地减半可使用这个协议转换的协议的数量, 同时将对网络监控和排错造成更多的困难(见下面网络监控和排错)。希望不会有 32768 个协议, 但 Murphy 制造了一些不允许我们作这个假设的规则。

理论上, 长度字段($ar\$hln$ 和 $ar\$pln$)是多余的, 因为通过硬件类型(在 $ar\$hrd$ 中)和协议类型(在 $ar\$pro$ 中)就可以决定协议地址的长度。它们被包括是为了可选的一致性检查和网络监控和排错(见下面)。

操作码决定了是请求(可能导致一个应答)还是对先前请求的应答。16 比特长了一些, 但这个字段是必须的。

发送者的硬件地址和协议地址绝对是有用的, 通过它们才能从转换表中得到结果。

在请求包格式中, 目的协议地址是必须的, 这样机器才能决定是否把发送者信息放到转换表中, 是否发送应答。如果假设应答是由请求引起的, 那么在应答包中这个字段不是必须的。包括它是为了完整性, 网络监控, 和使上面描述的算法更简单(把发送者信息放到转换表中后才去看操作码)。

目的硬件地址被包括进来是为了完整性和网络监控。它在请求包中毫无意义, 因为机器要问的就是这个数字。它在应答包中是处理请求机器的地址。在某些实现中(例如不检查 14 比特的以太网头), 把这个字段作为包的硬件地址发送到硬件驱动器, 存在寄存器或栈空间中。

地址间没有填充字节。包数据被看作字节流, 其中只有 3 个字节对可看作字($ar\$hrd$, $ar\$pro$ 和 $ar\$op$), 它们在发送时高位字节在前。

10. 网络监控和排错

以上的地址转换协议允许机器在以太网上获得高层协议活动(例如 CHAOS, Internet, PUP, DECnet)的信息。它能决定哪个以太网地址正在使用(通过值), 以及每个协议类型的协议地址。事实上, 监控者不必使用任何一种高层协议。它象下面这样工作:

当收到地址转换包, 它总是把<协议类型, 发送者协议地址, 发送者硬件地址>存入转换表。硬件和协议地址的长度可从包的 ar\$hln 和 ar\$pln 字段得到。如果操作码是应答, 监控者可以丢弃这个包。如果操作码是请求, 并且目的协议地址与监控者的协议地址相同, 监控者通常会发应答包。监控者将只得到一个映射, 因为请求的应答将被直接发送到请求主机。监控者可试着发自己的请求, 但要小心, 这会造成两个监控者陷入请求发送循环。

由于没有把协议和操作码合并成一个字段, 监控者不必知道每个高层协议的请求操作码对应的应答操作码。长度字段要带有可“分析”协议地址的足够信息, 虽然它并不带有协议地址的意义。

地址转换协议的一个成功实现还可为不成功的实现排错。假设一个硬件驱动器成功地广播了以太网类型为 ether_type\$ADDRESS_RESOLUTION 的包。由于实现的错误或维护表的复杂性, 包格式可能不正确。因为请求是广播, 监控者会收到这个包, 如果需要可显示出来进行排错。

11. 一个例子

假设在同一根 10 兆以太网电缆上有机器 X 和 Y。它们有以太网地址 EA(X) 和 EA(Y), DOD Internet 地址 IPA(X) 和 IPA(Y)。假设 Internet 的以太网类型为 ET(IP)。机器 X 刚启动, 并且它迟早都会向机器 Y 发包。X 知道要发包给 IPA(Y), 并把 IPA(Y) 告诉硬件驱动器(这里是以太网驱动器)。驱动器让地址转换模块把<ET(IP), IPA(Y)>转换成 48 比特以太网地址, 但因为 X 刚启动, 它没有这些信息。它先不发包, 生成一个地址转换包,

```
(ar$hrd) = ares_hrd$Ethernet  
(ar$pro) = ET(IP)  
(ar$hln) = EA(X) 的长度  
(ar$pln) = (IPA(X) 的长度)  
(ar$op) = ares_op$REQUEST  
(ar$sha) = EA(X)  
(ar$spa) = IPA(X)  
(ar$tha) = 任意值  
(ar$tpa) = IPA(Y)
```

并广播到电缆上的所有机器。

机器 Y 收到这个包, 判断自己是否懂这种硬件类型(以太网), 是否理解这种协议(Internet), 包是否是给自己的((ar\$tpa)=IPA(Y))。然后把<ET(IP), IPA(X)>映射到 EA(X) 的信息记下来(可能会覆盖已有表项)。然后又意识到是请求, 于是就交换字段, 把 EA(Y) 填入发送者以太网地址字段(ar\$sha), 把操作码设为应答, 再把包直接发送(不是广播)到 EA(X)。这个时候, Y 已经知道怎样向 X 发送, 而 X 还不知道怎样向 Y 发送。s

机器 X 收到 Y 发送的包, 生成<ET(IP), IPA(Y)>到 EA(Y) 的映射, 意识到是个应答包,

RFC826—An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware
以太网地址转换协议或转换网络协议地址为 48 比特以太网地址用于在以太网硬件上传输
于是丢弃。下次 X 的 Internet 模块试图向 Y 发送包，地址转换就会成功了，并且包也能到达。
如果 Y 的 Internet 模块要向 X 发送，它也会成功，因为 Y 已经从 X 的地址转换请求中记住了需要的信息。

12. 相关情况

也许希望转换表会过期，这些的实现超出本协议的范围。这里有一个较详细的描述(感谢 MOON@SCRC@MIT-MC)。

当主机移动时，假设移动时清除了地址转换表，那么从该主机发起的任何连接都可以工作。但是发起过到该主机连接的其它主机并没有任何理由会知道去丢弃它们的旧地址。而 48 比特以太网地址是唯一的，任何时候都是固定的，不会变。如果主机名(和其它协议地址)在不同物理硬件上被重新分配，主机就“移动”了。而且从经验来说，总会存在由于硬件或软件错误产生的错误路由信息，但这种错误不允许永远存在。也许发起某个连接的失败，会使地址转换模块认为由于对方当机或转换表项错误等原因而不可到达对方。从而删除这个信息。也许收到一个来自某个主机的包，会更新用来向该主机发送的转换表项的时钟。如果一定时间没有收到来自某个主机的包，这条转换表项会被删除。这将产生为每个收到的包扫描转换表的额外负担。或许使用散列或索引会快一些。

收到地址转换包的建议算法试图缩短主机移动以后的恢复时间。如果<协议类型，发送者协议地址>已经在转换表中，那么发送者的硬件地址将覆盖这个表项。因此在良好的以太网上，当请求广播到达后，每个工作站都将得到这个新的硬件地址。

另一种方法是有一个守护进程在处理超时。经过一定时间，守护进程考虑删除一个表项。它先用表里的以太网地址直接发送地址转换请求包(如果需要可重传几次)。如果在一段短时间内，没有收到应答，则删除表项。这个请求是直接发送的，不会影响以太网上的每个工作站。删除表项就是把必须重新获得的有用信息删除。

因为主机只发送关于它们自身的信息，而不会发送任何其它主机的信息，重新启动一个主机会使它的地址映射表成为最新的。通过机器间的传输，错误信息不会永远存在。机器

中唯一可存在的错误信息是不知道其它机器已经修改了 48 比特以太网地址。也许手工更新(或清除)地址映射表就够了。

如果认为重要的话，这篇文档需要更多地思考。任何地址转换类型的协议都用的到。