第 7 章　下一代网际协议 IPv6

本章是有关下一代网际协议 IPv6 的描述，重点介绍 IPv6 的产生原因、IPv6 的地址与 IPv6 首部格式等。通过本章的学习，读者应重点掌握和理解以下内容:

- IPv4 向 IPv6 发展的必然性
- IPv6 的新特性
- IPv6 地址的分类及其结构
- IPv6 基本报头结构
- IPv6 扩展报头及应用

7.1　概述

Internet 协议的第 4 版（IPv4）为 TCP/IP 协议簇和整个 Internet 提供了基本的通信机制。它从 1970 年底被采纳以来，几乎保持不变。版本 4 的长久性说明了协议设计是灵活和强有力的。从设计 IPv4 起到现在，处理器、存储器、Internet 的主干网带宽、网络技术及 Internet 状况都发生了相当大的变化。IPv4 虽然逐渐适应了技术的进展，但也逐渐暴露出了如下问题：

（1）地址枯竭。

IPv4 的 32 比特地址结构提供了约 43 亿个地址，虽然数量不少，但利用率不高。首先，早期的分类地址模式造成了大量地址的浪费，如早期美国的大学或大公司，几乎都能得到一个完整的 A 类或 B 类地址，直至目前很多组织仍拥有大量未被使用的 IP 地址；其次，地址分配存在地域上的不平衡，已经分配的 IPv4 地址中，美国大约占有 60%，亚太和欧洲地区占有 30%，非洲和拉美占有不到 10%；再有，用于组播的 D 类和保留的 E 类地址占了所有地址的 12%，还有 2%不能使用的特殊地址。

基于以上原因，随着网络规模的不断发展，IPv4 地址面临着短时间内枯竭的问题。对 IPv4 地址耗尽的具体日期目前各方尚未达成一致，因此出现了多个版本的 IPv4 枯竭计数器，各自标明的耗尽日期也不尽相同，下面是比较权威的两组数据，以供参考。截至 2011 年 3 月，距离 IPv4 地址枯竭还有 369 天（源自 penrose.uk6x.com）/ 36 天（源自 inetcore.com）。

（2）NAT 技术具有局限性。

为解决 IPv4 比较紧缺的问题，目前网络普遍使用 NAT（Network Address Translation，网络地址转换）技术。NAT 技术将私有地址映射到公有地址上，使很多使用私有地址的用户可以访问因特网。但 NAT 技术破坏了端到端的应用模型，如果内部网络使用私有地址的主机需要充当服务器，配置起来比较麻烦。此外，地址转换设备支持越多的转换，越会给设备增加更大的负载，对转发性能也有一些影响。正是由于 NAT 的这些局限，使得它作为解决 IP 地址不

足的措施只能是权宜之计。

（3）路由表膨胀。

早期 IPv4 的地址结构也造成了路由表的容量过大。IPv4 地址早期为“网络号+主机号”结构，后来引入子网划分后为“网络号+子网号+主机号”结构，这两种结构不能进行地址块的聚合。CIDR 技术的出现，在一定程度上缓解了这个问题，但仍有历史遗留的大量地址空间无法改造。随着因特网中路由器和网络的增多，路由表容量的压力将会越来越大。

（4）地址配置不够简便。

IPv4 的地址配置使用手动配置方法或有状态的自动配置（如 DHCP，动态主机配置协议）。手动配置要求使用者懂得一定的计算机网络知识；自动配置需要管理员部署和维护 DHCP 服务。以上都需要 IP 协议能提供一种更简单、更方便的地址自动配置技术，减少工作量和管理难度。

（5）安全性和 QoS（服务质量）方面的问题。

IPv4 本身并没有提供安全性的机制，如果需要安全保证，则需要额外使用 IPSec、SSL 等安全技术。IPv4 虽然具有 QoS 相应设计，但是因为种种原因在实际当中并没得到普及和使用。在现实中涌现的大量新兴网络业务，如视频点播、IP 电话等，都需要 IP 网络在时延、抖动、带宽、出错率方面提供一定的服务质量保障。IPv4 在安全和 QoS 方面的缺陷使其已经不能满足目前因特网的使用需求。

鉴于以上原因，人们认识到需要设计一种新的 IP 协议来代替 IPv4。从 1990 年开始，互联网工程任务小组（Internet Engineering Task Force，IETF）开始规划 IPv4 的下一代协议，除要解决即将遇到的 IP 地址短缺问题外，还要发展更多的扩展功能。1994 年，各 IP 领域的代表们在多伦多举办的 IETF 会议中正式提议 IPv6 发展计划，该提议直到同年的 11 月 17 日才被认可，并于 1998 年 8 月 10 日成为 IETF 的草案标准。

IPv6 被设计成不仅有较大的地址空间，而且有更好的性能。可以说，IPv6 除了将地址扩大为 128 位之外，在首部格式、地址分配、组播支持、安全与扩展性等方面也都作出了改进。IPv6 继承了 IPv4 的优点并弥补了 IPv4 的不足。IPv6 与 IPv4 并不兼容，但与其他协议兼容，即 IPv6 完全可以取代 IPv4。

IPv6 所引进的变化可以分成以下 6 类：

- 更大的地址空间。新的地址大小是 IPv6 最显著的变化。IPv6 把 IPv4 的 32 位地址增大到了 128 位。IPv6 的地址空间足够大，在可预见的将来不会耗尽。
- 灵活的首部格式。IPv6 使用一种全新的、与 IPv4 不兼容的数据报格式。IPv6 删除和修改了 IPv4 首部的一些字段，并且创造性地用扩展首部替代了 IPv4 的选项字段。与 IPv4 相比，处理 IPv6 首部的速度更快，而且 IPv6 首部实现的功能更多和更具扩展性。
- 对自动配置的支持。IPv6 引入了无状态的地址自动配置，该机制是 IPv6 的基本组成部分，无需专门的设备支持。该机制比 DHCP 更简单，使用更方便，这使得网络（尤其是局域网）的管理更加方便和快捷。
- 支持资源分配。IPv6 提供了一种机制，允许对网络资源进行预分配，它以此取代了 IPv4 的服务类型说明。这些新的机制支持实时视频等应用，这些应用要求保证一定的带宽和时延。此外，对增强的组播支持也使得网络上的多媒体有了长足发展的机会。
- 更小的路由表。IPv6 的地址分配一开始就遵循聚类（Aggregation）的原则，这使得路

由器能在路由表中用一条记录表示一片子网，大大减小了路由器中路由表的长度，提高了路由器转发数据包的速度。

- 更高的安全性。在 IPv6 的首部中，增加了安全的扩展首部，支持 IPv6 协议的节点就可以自动支持 IPSec，使加密、验证和虚拟专用网（VPN）的实施变得更加容易。

7.2 IPv6 地址

在 IPv6 中，每个地址占 128 位，地址空间大于 3.4×10^{38}。如果整个地球表面（包括陆地和水面）都覆盖着计算机，那么 IPv6 允许每平方米拥有 7×10^{23} 个 IP 地址。如果地址分配速率是每微秒分配 100 万个地址，则需要 10^{19} 年时间才能将所有可能的地址分配完毕。可见在想象得到的将来，IP 的地址空间是不可能用完的。考虑到 IPv6 的地址分配方式，不是每一个地址都可以得到使用，但是分配到每个人，其数量仍然是巨大的。

1. IPv6 地址格式

巨大的地址范围还必须使维护互联网的人易于阅读和操纵这些地址。IPv4 所用的点分十进制记法现在也不够方便了。读者可以想象用点分十进制记法的 128 位（16 字节）的地址写法会有多么不便。因此，依据 RFC 4291（IP Version 6 Addressing Architecture），IPv6 的地址有 3 种格式：首选格式、压缩表示和内嵌 IPv4 地址的 IPv6 地址表示。

首选格式中，IPv6 地址的 128 位中每 16 位为一段（field），每段的 16 位二进制数又分别转换为 4 个十六进制数，这样 128 位的 IPv6 地址就被分成了 8 段，每段之间用冒号分隔。这种表示方法叫“冒号十六进制表示方法”。

下面是一个二进制表示的 128 位的 IPv6 地址：

0010000000000001000011011011100000000000000000000000000000000000

0000000000001000000010000000000001000000000110001000000101111010

将其分为 8 段，每 16 位一段：

0010000000000001　0000110110111000　0000000000000000　0000000000000000

0000000000001000　0000100000000000　0010000000001100　0100000101111010

每段都转换为 4 个十六进制数，段之间用冒号隔开，就成为了如下的地址形式：

2001:0DB8:0000:0000:0008:0800:200C:417A

在 IPv6 地址的每段中，前导的 0 可以去掉，但每段要至少保留一个数字，上述 IP 地址去掉前导 0 的过程如图 7-1 所示。

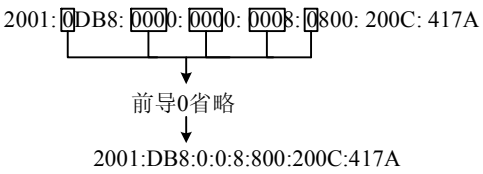


图 7-1　IPv6 地址中省略前导 0

为了使地址更加简洁，IPv6 使用压缩表示的格式，如果 IPv6 地址存在一个或多个连续的全 0 段，这一个或多个段用::表示。

如地址 2001: DB8: 0: 0: 8:800:200C:417A 中，第 3、4 段均为全 0（即每段的 16 位均为 0），则 3、4 段可压缩为::，压缩后该地址为 2001: DB8:: 8:800:200C:417A。

下列是一些地址的例子：

```
FF01:0:0:0:0:0:0:101          一个组播地址
0:0:0:0:0:0:0:1               环回地址
0:0:0:0:0:0:0:0               未指定地址
```

可以被压缩为：

```
FF01::101             一个组播地址
::1                   环回地址
::                    未指定地址
```

需要注意的是，为避免歧义，该压缩方法只能使用一次，一般压缩较长的部分，如图 7-2 所示。

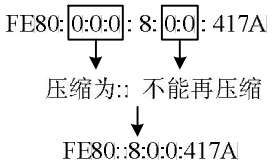


图 7-2　压缩方法只能用一次

在 IPv6 过渡机制中，为和 IPv4 地址共存，还使用了内嵌 IPv4 地址的 IPv6 地址表示。在这种表示方法中，IPv6 地址的第一部分用冒号十六进制表示，而 IPv4 地址部分是点分十进制格式：

x:x:x:x:x:x:d.d.d.d （x 表示一个 4 位的十六进制数，d 表示 IPv4 地址中的一个十进制数）

2. IPv6 地址分类

IPv6 地址用于标识不同的网络接口，按其标识网络接口的多少，IPv6 地址有 3 种类型：单播（Unicast）、组播（Multicast）和任播（Anycast）。广播地址已不再有效，其功能由组播地址来实现。

（1）单播地址。一个单接口的标识符，可以作为源地址和目的地址。送往一个单播地址的包将被传送至该地址标识的接口上。单播地址按其作用范围不同，又可分为：链路－本地地址（Link-local Address）、站点－本地地址（Site-local Address，目前已被唯一本地地址取代）、可汇聚的全球单播地址（Aggregatable Global Unicast Address），如图 7-3 所示。

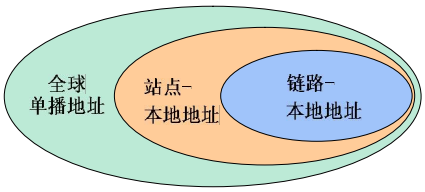


图 7-3　单播地址按作用范围分类

可汇聚的全球单播地址用于全球范围内的通信，通俗地说就是 IPv6 的公网地址，其前缀的高 3 位固定为 001，根据 RFC3177（IAB IESG Recommendations on IPv6 Address Allocations to Sites）的建议，其地址结构如图 7-4 所示。

其中全球路由选择前缀和高 3 位 001 共占 48 位，用于进行全球范围内的路由；子网标识符占 16 位，用于组织内部标识子网；接口标识符占 64 位，用于标识链路上的不同接口。

图 7-4　可汇聚的全球单播地址结构

链路－本地地址用于单个链路上的设备通信。两个设备在单个链路上是指设备间没有三层设备（如路由器）分隔，只有一层或二层设备相连。当支持 IPv6 的节点上线时，每个接口默认地自动配置链路－本地地址，该地址专门用来和链路上的其他主机通信。链路－本地地址主要用于寻找邻居或路由器等操作。在主机启动后尚未获取较大范围的地址时，可以使用链路－本地地址进行通信。其地址结构如图 7-5 所示。

图 7-5　链路－本地地址结构

下面以 Windows XP 为例，说明怎样查看链路－本地地址。

首先，在 Windows XP 下安装 IPv6 协议。其具体步骤为：

1）打开命令提示符，方法是单击“开始”→“所有程序”→“附件”→“命令提示符”命令。

2）在命令提示符下依次输入 netsh→int ipv6→install，如图 7-6 所示。

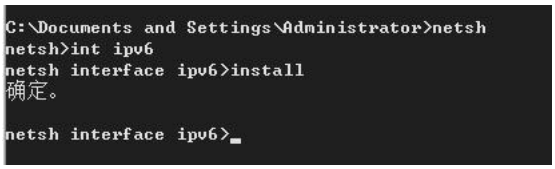


图 7-6　IPv6 协议的安装

3）在命令提示符下查看本机 IPv6 地址，依次输入 netsh→int ipv6→show add，如图 7-7 所示。

图 7-7　查看链路－本地地址

查看到本机的链路－本地地址为 fe80::be30:5bff:fec2:9feb。可以看出，链路－本地地址的十六进制形式为 FE80::InterfaceID，其前缀固定为 FE80::/64，其接口标识符使用 EUI-64 方法自动生成。

站点－本地地址与 IPv4 中的私有地址类似。使用站点－本地地址作为源或目的地址的数据报文不会被转发到本站点（相当于一个私有网络）外的其他站点。站点－本地地址使用

FEC0::/10 前缀。因为站点－本地地址的一些缺陷，目前该地址已被唯一本地地址（Unique Local Address）取代，唯一本地地址使用 FC00::/7 前缀。

单播地址的接口标识符（InterfaceID）用于标识链路上的不同接口，可以自动生成或手动配置。在以太网中一般使用 EUI-64 方法生成接口标识符。因为以太网接口即为以太网卡，所以接口标识符和网卡对应。EUI-64 方法用网卡 MAC 地址生成接口标识符。EUI-64 转换过程中首先将 16 位的 1111111111111110（0xFFFE）插入到 MAC 地址的前 24 位和后 24 位之间，再将 MAC 地址的 U/L 位（全局/本地位，是第一个字节的第 7 位）置为 1。一个转换示例如图 7-8 所示。

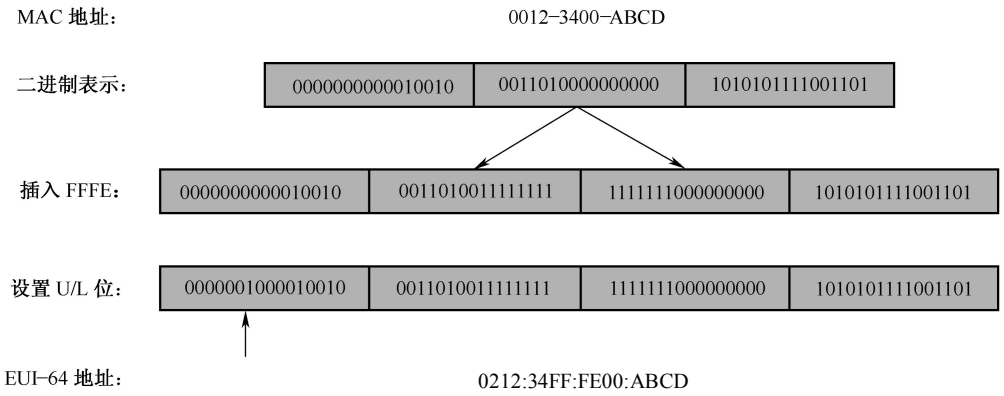


图 7-8　MAC 地址到 EUI-64 格式接口标识符的转换过程

（2）组播地址。一组接口（一般属于不同节点）的标识符，只可作为目的地址。送往一个组播地址的数据包将被传送至由该地址标识的所有接口上。因为一个组播地址对应多个接口，所以需要清楚一个给出的组播地址与哪些接口对应。IPv6 的组播地址比较重要，不仅取代了 IPv4 中的广播地址，而且完成了其他一些常见功能。

IPv6 组播地址前 8 位为 11111111，即使用 FF::/8 前缀，其结构如图 7-9 所示。

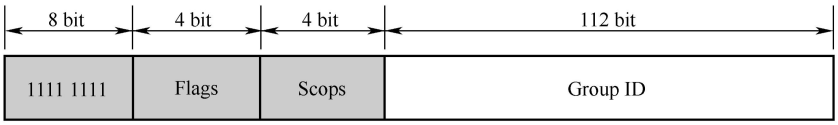


图 7-9　组播地址结构

Flags（标志）字段指出在组播地址上设置的标志。该字段为 4 位。从 RFC 2373 起，定义的唯一标志是该字段的最后一位，该位被定义为 Transient（T）标志，其余三位必须置 0。当设置为 0 时，T 标志指出组播地址是由 Internet 编号授权委员会（IANA）永久指派的多播地址。当设置为 1 时，T 标志指出组播地址是临时（非永久指派）的组播地址。

Scope（作用域）字段指出组播通信发生的 IPv6 网络的作用域。该字段的大小为 4 位。用来限制组播数据流在网络中发送的范围。以下是作用域在 RFC 2373 中的定义：

0：预留。

1：节点本地范围。

2：链路本地范围。

5：站点本地范围。

8：组织本地范围。

E：全局范围。

F：预留。

其中，链路本地范围、站点本地范围、全局范围与单播地址中的相应范围含义相同。节点本地范围代表一个节点内部的范围，仅用于在节点内部发送环回测试的组播数据；组织本地范围代表属于一个组织的多个站点的范围。

例如，使用组播地址 FF02::2 的通信有链路本地作用域。IPv6 路由器永远不会将此通信转发到本地链路以外。

Group ID（组 ID）字段标识了组播组，并且在作用域中是唯一的。该字段的大小为 112 位。永久指派的组 ID 独立于作用域。临时组 ID 仅与特定的作用域有关。

因为 IPv6 的组播地址取代了 IPv4 中的广播地址，所以又定义了相关作用范围内的一些组播地址，这些地址均为 IANA 永久指派的组播地址。

（3）任播地址。一组接口（一般属于不同节点）的标识符，只可作为目的地址。发送到任播地址的数据报文被传送给此地址所标识的一组接口中距离源节点最近（根据使用的路由协议进行度量）的一个接口。

单播允许源节点向单一目标节点发送数据报，组播允许源节点向一组目标节点发送数据报，而任播则允许源节点向一组目标节点中的一个节点发送数据报，而这个节点由路由系统选择，对源节点透明；同时，路由系统选择“最近”的节点为源节点提供服务，从而在一定程度上为源节点提供了更好的服务，也减轻了网络负载。

为了易于传输到最近的任意广播组成员，路由结构必须知道指派任意广播地址的接口以及按照路由度量的距离。目前，任意广播地址只被用于目标地址，并且只被指派给路由器。任意广播地址从单播地址空间指派。任意广播地址的作用域是指派任意广播地址的单播地址类型的作用域。

3. 无状态的地址分配

IPv6 单播地址配置可以分为手动地址配置和自动地址配置两种方式。自动地址配置方式又可以分为无状态地址自动配置和有状态地址自动配置两种。

在无状态地址自动配置方式下，网络接口接收路由器宣告的全局地址前缀，再结合接口 ID 得到一个全局单播地址。在有状态地址自动配置的方式下，主要采用动态主机配置协议（DHCP），需要配备专门的 DHCP 服务器，网络接口通过客户机/服务器模式从 DHCP 服务器处得到地址配置信息。

与手动地址配置相比，无状态地址自动配置无需用户进行操作，提高了地址配置的自动化程度；与有状态地址自动配置（DHCP）相比，无状态地址自动配置只需路由器通告前缀，而无需记录地址的分配情况，减少了设备的负担。

在无状态地址自动配置过程中，路由器负责前缀通告。节点收到路由器通告的地址前缀后，加上自动生成的地址后缀（接口 ID）即可得到完整的地址。节点在生成地址后缀时，一般使用 EUI-64 方法基于 MAC 地址生成，因此在同一网段中不会出现地址冲突。

无状态地址自动配置的具体过程由 IPv6 的 ND（Neighbor Discovery，邻居发现）协议完成，感兴趣的读者可查阅相关资料。

7.3　IPv6 基本格式

IPv6 的数据报格式如图 7-10 所示。

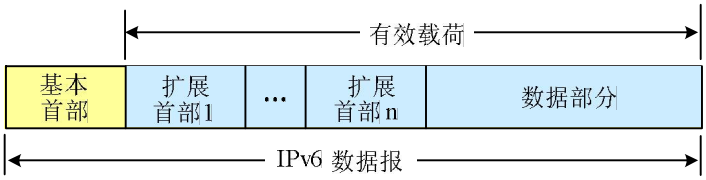


图 7-10　IPv6 数据报一般格式

IPv6 数据报由 IPv6 基本报头和 IPv6 有效载荷组成。基本首部（Base Header）的大小固定，其后的有效载荷中允许有零个或多个扩展首部（Extension Header），再后是上层协议的数据。

IPv6 通过将 IPv4 报头中的某些字段裁减或移入到扩展报头，减小了 IPv6 基本报头的长度。IPv6 使用固定长度的基本报头，从而简化了转发设备对 IPv6 报文的处理，提高了转发效率。尽管 IPv6 地址长度是 IPv4 地址长度的 4 倍，但 IPv6 基本报头的长度只有 40 字节，为 IPv4 报头长度（不包括选项字段）的 2 倍。

图 7-11 所示为 IPv6 基本首部的格式。每个 IPv6 数据报都从基本首部开始。IPv6 基本首部的不少字段可以和 IPv4 首部中的字段直接对应。

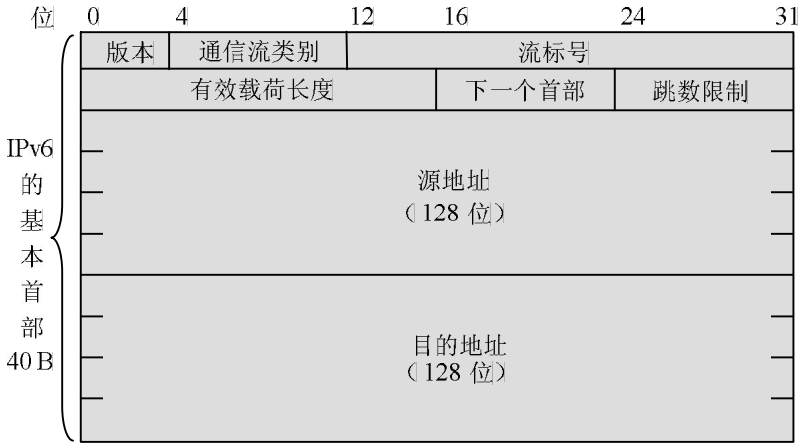


图 7-11　IPv6 数据报基本首部格式

下面介绍 IPv6 基本首部中的各字段。

（1）版本（Version）。此字段占 4 位，它指明了协议的版本，对于 IPv6 该字段总是 6。

（2）通信流类别（Traffic Class）。此字段占 8 位，指明数据报的流类型。该字段执行与 IPv4 首部服务类型相同的功能。

（3）流标号（Flow Label）。此字段占 20 位，该字段标明了一个流，其目的是不需要查看内部数据路由器就能识别属于同一流的数据并以类似的方式进行处理。

IPv6 提出流的抽象概念。所谓流就是互联网上从一个特定源站到一个特定目的站（单播

或多播）的一系列数据报，而源站要求在数据报传输路径上的路由器保证指明的服务质量。例如，两个要发送视频的应用程序可以建立一个流，它们所需要的带宽和时延在此流上可得到保证。另一种方式是，网络提供者可能要求用户指明他所期望的服务质量，然后使用一个流来限制某个指明的计算机或指明的应用程序所发送的业务流量。流也可以用于某个给定的组织，用它来管理网络资源，以保证所有的应用能公平地使用网络。

所有属于同一个流的数据报都具有同样的流标号。源站在建立流时是在 2^{20}-1 个流标号中随机选择一个，即流标识符。流标号 0 保留，作为指出没有采用流标号。源站随机地选择流标号并不会在计算机之间产生冲突，因为路由器在将一个特定的流与一个数据报相关联时，使用的是数据报的源地址和流标号的组合。

（4）有效载荷长度（Payload Length）。此字段占 16 位，指明除首部自身的长度外，IPv6 数据报所载的字节数。可见一个 IPv6 数据报可容纳 64KB 的数据。由于 IPv6 的首部长度是固定的，因此没有必要像 IPv4 那样指明数据报的总长度（首部与数据部分之和）。

（5）下一个首部（Next Header）。此字段占 8 位，标识接在 IPv6 基本首部后面的扩展首部的类型。

（6）跳数限制（Hop Limit）。此字段占 8 位，用来防止数据报在网络中无限期地存在。源站在每个数据报发出时即设定某个跳数限制。每一个路由器在转发数据报时，要先将跳数限制字段中的值减 1。当跳数限制的值为零时，就要将此数据报丢弃。这相当于 IPv4 首部中的寿命字段，但比 IPv4 中的计算时间间隔要简单些。

（7）源站 IP 地址。此字段占 128 位，是此数据报的发送站的 IP 地址。

（8）目的站 IP 地址。此字段占 128 位，是此数据报的接收站的 IP 地址。

下面是一个 IPv6 数据报的实例，如图 7-12 所示。

```
⊞ Frame 6: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
⊞ Ethernet II
⊟ Internet Protocol Version 6
  ⊞ 0110 .... = Version: 6
  ⊞ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 40
    Next header: ICMPv6 (0x3a)
    Hop limit: 128
    Source: fe80::240:5ff:fe42:e967 (fe80::240:5ff:fe42:e967)
    [Source SA MAC: AniCommu_42:e9:67 (00:40:05:42:e9:67)]
    Destination: fe80::20d:88ff:fe47:5826 (fe80::20d:88ff:fe47:5826)
    [Destination SA MAC: D-Link_47:58:26 (00:0d:88:47:58:26)]
⊟ Internet Control Message Protocol v6
    Type: Echo (ping) request (128)
    Code: 0
    Checksum: 0x065f [correct]
    Identifier: 0x0000
    Sequence: 52
  ⊞ Data (32 bytes)
```

图 7-12　IPv6 数据报首部

该数据报是在图 7-13 的 ping 命令过程中，由主机发送的第一个 ping 命令的报文。

IPv6 数据报首部各字段的取值和含义如下：

版本（Verison）=6：说明此 IP 数据报是 IPv6 的数据报。

流类别（Traffic Class）=0：说明此 IPv6 数据报属于默认的流类型，无需特殊处理。

流标号（Flow Label）=0：不使用流功能，所以该字段的取值一般为 0。

有效载荷长度（Payload Length）=40：说明有效载荷一共为 40 字节。

```
C:\Documents and Settings\jsjwl>ping6 fe80::20d:88ff:fe47:5826%5

Pinging fe80::20d:88ff:fe47:5826%5
from fe80::240:5ff:fe42:e967%5 with 32 bytes of data:

Reply from fe80::20d:88ff:fe47:5826%5: bytes=32 time<1ms
Reply from fe80::20d:88ff:fe47:5826%5: bytes=32 time<1ms
Reply from fe80::20d:88ff:fe47:5826%5: bytes=32 time<1ms
Reply from fe80::20d:88ff:fe47:5826%5: bytes=32 time<1ms
```

图 7-13　IPv6 下的 ping 命令

下一个首部（Next Header）=58（即十六进制 3a）：说明载荷部分数据为 ICMPv6 的数据，由此可以看出此 IPv6 数据报没有扩展首部，载荷部分直接为上层协议数据。

跳数限制（Hop Limit）=128：说明此数据报在传输过程中最多跨越 128 台路由器。

源站 IP 地址=fe80::240:5ff:fe42:e967：该地址为数据报的发送主机的地址，由取值可以看出，这是一个链路－本地地址。

目的站 IP 地址= fe80::20d:88ff:fe47:5826：该地址为数据报接收方主机的地址，也是一个链路－本地地址。该地址是执行 ping 命令时由用户指定的。

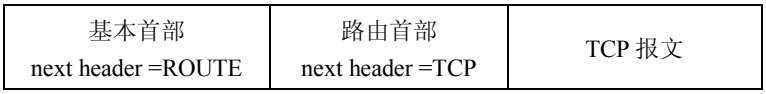
7.4　IPv6 扩展首部

基本的 IPv6 首部对于执行转发等基本功能是足够的，但一些扩展功能，如源站指定路由等，还需要更多的字段。因为基本首部是固定的，所以 IPv6 将实现扩展功能的部分放到有效载荷中。根据需要，IPv6 基本首部后面的有效载荷中可以有 0 个、1 个或多个连续的扩展首部，每个扩展首部分别实现不同的扩展功能，参见图 7-10。

通过扩展首部，IPv6 比 IPv4 提供了更多扩展功能。IPv4 选项字段受限于 40 个字节，而 IPv6 扩展首部仅受限于分组大小。而且，除了逐跳选项扩展首部外，路由器只处理基本首部，而不处理其余扩展首部，这样提高了转发效率并减少了中间路由器的负担。

每个基本首部和扩展首部都包含一个下一个首部（Next Header）字段。从基本首部开始，每个下一个首部字段指明后续首部的类型。最后一个扩展首部的下一个首部字段指明了后面高层协议的类型。例如，图 7-14 展示了 3 个数据报的下一个首部字段。

（a）只有一个基本首部



（b）一个基本首部和一个扩展首部

（c）一个基本首部和两个扩展首部

图 7-14　IPv6 数据报的首部与扩展首部

图 7-14 的（a）（b）（c）分别包含 0、l、2 个扩展首部，每个首部的下一个首部指明了接下去的首部的类型。其中因为第一个数据报包含 0 个扩展首部，所以其基本首部的下一个首部字段直接指明了高层协议的类型是 TCP。

下面介绍几种扩展首部及其功能。

1．逐跳选项扩展首部

逐跳选项（hop-by-hop options）扩展首部所携带的信息在数据报传送的路径上每一个路由器都必须加以检查。到目前为止，只定义了一个选项：超大净荷，格式如图 7-15 所示。

图 7-15　超大净荷选项

这个选项支持超过 65535 字节的净载荷，当使用这个选项时，IPv6 固定报头中的净载荷字段要设置为 0。

hop-by-hop 扩展首部包括以下几个字段：

（1）下一个首部（8 bit）。

（2）扩展首部的长度（8bit）：长度以 8 字节为单位，但不包括最开始的 8 个字节，所以这个字段目前值为 0。

（3）选项类型（8bit）：选项类型中低 5 bit 指明一个具体的选项。选项类型中最高的 2bit 指明当一个节点不能识别这一选项时应采取如下行动：

00：跳过此选项，继续处理这个首部。

01：丢弃此数据报，但不发送 ICMP 报文。

10：丢弃此数据报，向源站发送 ICMP 报文，指出不能识别的选项类型。

11：丢弃此数据报，向源站用非多播方式发送 ICMP 报文，指出不能识别的选项类型。

选项类型中第 3 个高位比特指明在数据报从源站到目的站的传送过程中不允许改变（0）或允许改变（1）。

这里的 194 对应二进制 11000010。高两位 11，指明当节点不能识别此选项的含义时应采取的行动；第三位为 0，指明在数据报从源站到目的站的传送过程中不允许改变；后 5 位为 00010，定义为超大净载荷选项报头。

（4）净载荷长度的字节数（8bit），当前值为 4：表示用接下来的 4 个字节（32bit）表示净载荷长度。

接下来的 4 字节表明净载荷长度。小于 65535 字节的长度是不允许的，第一台路由器将会丢弃此类分组并作为不能识别的选项处理，向源站发 ICMP 出错消息。

32bit 长的字段可指明超过 4GB 长的数据。对于这种数据报不能有分片扩展首部。这有利于传送大量的视频数据或在超级计算机之间传送 GB 量级的数据，也有利于使 IPv6 最佳地使用任何传输介质可供使用的容量。

2. 路由选择扩展首部

路由选择扩展首部用于源站选路，它具有如下一些字段（见图 7-16）：

（1）下一个首部（8bit）。

（2）路由选择类型（8bit），目前置为零。

（3）地址数目（8bit）：即在此扩展首部中所指明的地址数（0～23），这些地址指明了数据报必须要通过的中间路由器。

（4）下一个地址（8bit）：指明下一个要找的地址。这个字段在初始化时为零，以后每经过一个路由器，就将此字段的值加 1。

（5）比特掩码（24bit）：每一个比特对应于 24 个地址中的一个。若某个比特为 1，则表示是严格的源站选路，即该比特所对应的地址必须成为它前一个地址的下一站地址。反之，若某个比特为 0，则表示是不严格的源站选路，即该比特所对应的地址不一定必须是它前一个地址的下一站地址。

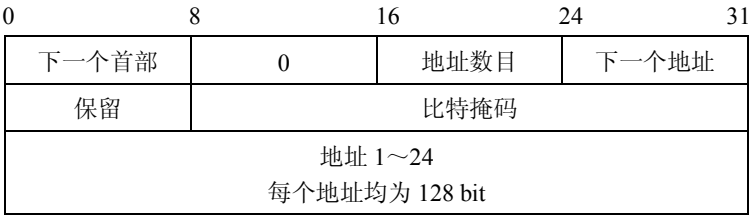


图 7-16　路由选择扩展首部

3. 分片扩展首部

IPv6 将分片限制为由源站来完成。在发送数据前，源站必须进行一种称为路径的最大传送单元发现（Path MTU Discovery）的技术，以此来确定沿着这条路径到目的站的最小 MTU。在发送数据报前，源站先将数据报分片，保证每个数据报片都小于此路径的 MTU。因此，分片是端到端的，中间的路由器不需要进行分片。

IPv6 基本首部中不包含用于分片的字段，而是在需要分片时，源站在数据报片的基本首部的后边插入一个小的分片扩展首部（见图 7-17）。

图 7-17　分片扩展首部

IPv6 保留了 IPv4 分片的大部分特征。下一个首部字段指明紧接着这个扩展首部的下一个首部。保留字段是为今后使用的。片偏移字段共 13bit，它指明本数据报片在原来的数据报中的偏移量，其单位是 8 个字节。可见每个数据报片必须是 8 个字节的倍数。再后面的保留字段占 2bit，也是为今后使用的。M 字段中只有 1 个比特。M=1 表示后面还有数据报片，M=0 表示已经是最后一个数据报片。标识符字段采用 32bit，可以适应更高速的网络，它用来唯一地标识原来的数据报。

一、选择题

1．IPv6 采用的地址表示格式为（　　）。

A．冒号十六进制　　B．点分十进制

C．冒号十进制　　D．点分十六进制

2．IPv4 地址包含网络号、主机号、子网掩码等。与之相对应，IPv6 地址包含了（　　）。

A．前缀、接口标识符、前缀长度　　B．网络号、主机号、前缀长度

C．前缀、接口标识符、网络长度　　D．网络号、主机号、网络长度

3．IPv6 链路－本地地址属于（　　）。

A．广播地址　　B．组播地址　　C．单播地址　　D．任播地址

4．IPv6 站点－本地地址属于（　　）。

A．单播地址　　B．广播地址　　C．组播地址　　D．任播地址

5．下列（　　）是正确的 IPv6 地址。

A．2001:410:0:1:45ff　　B．2001:410:0:1::45ff

C．2001:410:0:1:0:45ff　　D．2001:410::1:0:0:0:0:45ff

二、简答题

1．IPv6 与 IPv4 相比发生了哪些变化？这些变化对网络的发展将产生怎样的影响？

2．将地址 0000:0DB8:0000:0000:0008:0800:200C:417A 用零压缩法写成简洁形式。

3．IPv6 数据报是否需要分片，如果需要是如何实现的？