

- `tcp.analysis.zero_window`: 用来让 Wireshark 显示含零窗口通告信息的 TCP 数据包 (TCP 会话一端的主机通过此类 TCP 数据包, 向对端主机报告: 本机 TCP 窗口为 0, 请贵机停止通过该会话发送数据)。



注意

Wireshark 在调用 `tcp.analysis` 参数筛选数据包时, 并不会检查数据包的 TCP 头部, 所依据的是该软件自带的 `expert system` 对 TCP 传输机制的分析和理解。

- `tcp.flags`: 该参数一经调用, Wireshark 就会检查数据包 TCP 头部中各标记位的位置情况。以下所列为该参数名下的几个子参数。
 - `tcp.flags.syn == 1`: 用来让 Wireshark 显示 SYN 标记位置 1 的 TCP 数据包。
 - `tcp.flags.reset == 1`: 用来让 Wireshark 显示 RST 标记位置 1 的 TCP 数据包。
 - `tcp.flags.fin == 1`: 用来让 Wireshark 显示 FIN 标记位置 1 的 TCP 数据包。



注意

可利用 `tcp.flags` 过滤参数, 让 Wireshark 检查 IP 包的 TCP 头部中各标记位的位置情况。

- `tcp.window_size_value < <value>`: 该过滤参数一经调用, Wireshark 将会只显示 TCP 头部中窗口大小字段值低于指定值的数据包。可利用该参数来排除与 TCP 窗口过小有关的网络性能问题, 此类问题有时要拜赐于参与 TCP 会话的网络设备“反应过慢”。

3.4.2 配置方法

先举几个 TCP/UDP 显示过滤器的配置实例。

- 要让 Wireshark 只显示涌向 HTTP 服务器的所有流量, 显示过滤器应如此配置:
`tcp.dstport == 80`。
- 要让 wireshark 只显示由 IP 子网 10.0.0.0/24 内的主机访问 HTTP 服务器的所有流量,

显示过滤器应如此配置: `ip.src==10.0.0.0/24 and tcp.dstport == 80`。

- 要让 Wireshark 只显示某条特定 TCP 连接（在抓包文件中编号为 16 的 TCP 连接）中发生重传的所有 TCP 数据包，显示过滤器应如此配置: `tcp.stream eq 16 && tcp.analysis.retransmission`。

要想让 Wireshark 只显示某条 TCP 连接从建立到终结，会话双方生成的所有数据包，请在抓包主窗口选择一个隶属于该 TCP 连接（也叫 TCP Stream[TCP 流]）的 TCP 数据包，同时点右键，在右键菜单中选择 **Follow TCP Stream** 菜单项。一条 TCP Stream 就是 TCP 会话双方从建立连接到终止连接那段时间内交换的所有数据包。只要点击过 **Follow TCP Stream** 菜单项，在 Filter 输入栏内会自动出现 `tcp.stream eq <value>` 的字样。这里的 value 就是 Wireshark 在抓包文件中为这条 TCP 连接分配的标识号。对于前例，过滤参数中所引用的标识号为 16，可为任意数字（在所有抓包文件中，该标识号从 1 开始分配）。

导致 TCP 重传的原因有很多，本书第 9 章会对此展开深入讨论。



注意

当使用 Wireshark 分析 TCP 重传、重复确认以及其他影响网络性能的现象的原因时，应（借助于 `tcp.analysis` 过滤参数及 **Follow TCP Stream** 菜单项）把上述现象与具体的 TCP 连接建立起关联。

再举几个与 TCP/UDP 有关的显示过滤器配置实例。

- 要让 Wireshark 只显示某条特定 TCP 连接中出现窗口问题的 TCP 数据包，显示过滤器应如此配置。
 - `tcp.stream eq 0 && (tcp.analysis.window_full || tcp.analysis.zero_window)`
 - `tcp.stream eq 0 and (tcp.analysis.window_full or tcp.analysis.zero_window)`
- 要让 Wireshark 只显示 IP 地址为 10.0.0.5 的主机访问 DNS 服务器的流量，显示过滤器应如此配置: `ip.src == 10.0.0.5 && udp.port == 53`。
- 要让 Wireshark 只显示包含某指定字符串（区分大小写）的 TCP 数据包（比如，在百度中搜索关键字“**Windows**”），显示过滤器应如此配置: `tcp contains "Windows"`。
- 要让 Wireshark 只显示由 IP 地址为 10.0.0.3 的主机生成的所有 TCP 重传数据包，显示过滤器应如此配置: `ip.src ==10.0.0.3 and tcp.analysis.retransmission`。

- 要让 Wireshark 只显示涌向 HTTP 服务器的所有流量，显示过滤器应如此配置：
`tcp.dstport == 80`。
- 要让 Wireshark 只显示由指定主机建立 TCP 连接时生成的所有数据包（若某台主机在执行某种形式的 TCP 端口扫描，或某台主机感染了蠕虫病毒时，就会批量生成此类数据包），显示过滤器应如此配置：`ip.src==10.0.0.5 && tcp.flags.syn==1 && tcp.flags.ack==0`。
- 要让 Wireshark 只显示由指定主机发送的包含 HTTP cookie 的所有数据包，显示过滤器应如此配置：`ip.src==10.0.0.3 && (http.cookie || http.set_cookie)`。

3.4.3 幕后原理

图 3.8 和图 3.9 分别示出了 IPv4 包头和 TCP 头部的格式，而 UDP 头部的结构则比较简单，只包括源、目端口号字段、长度字段以及校验和字段，因此不再示出。先来看一下 IP 包头的结构。

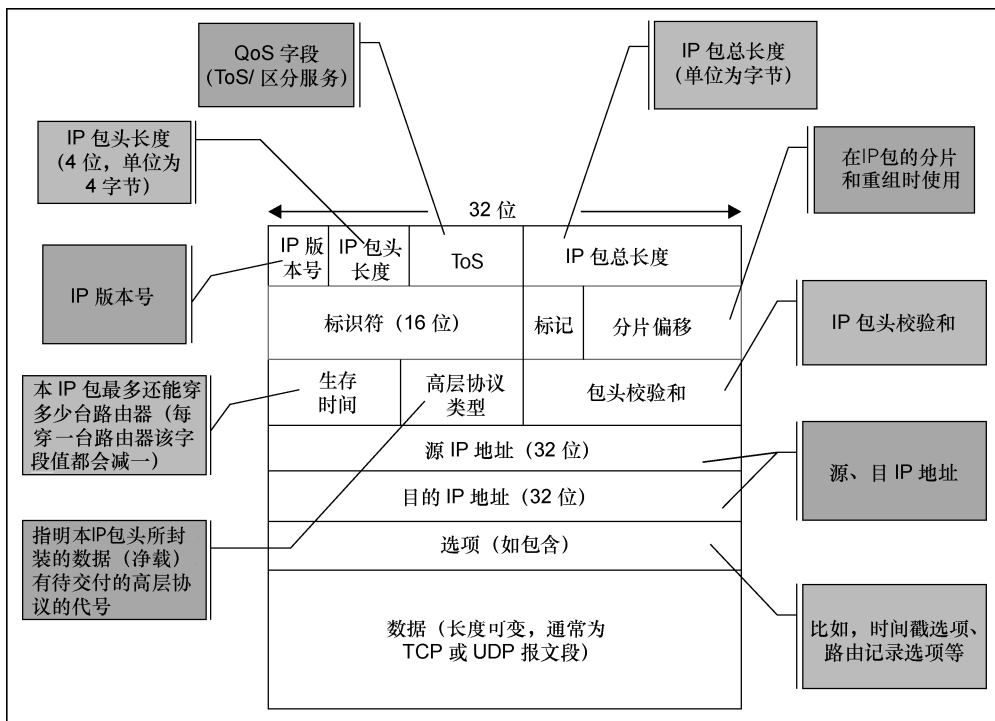


图3.8

下面简单介绍一下 IPv4 包头中的若干重要字段。

- **版本字段：**表示 IP 协议的版本号，其值为 4。
- **IP 包头长度字段：**用来指明 IP 包头的长度，单位为 4 字节，其值一般为 5，最大值为 15（考虑了 IP 包头中含有选项字段的情况）。
- **QoS（服务类型）字段：**一般都采用区分服务的置位方式，用来区分不同类型流量的贵贱程度。



注意

在发布于 1981 年 9 月的 RFC 791 中，曾把 QoS 字段命名为 ToS（服务类型）字段，并针对该字段中的每一位定义了一套置位方式。在 1998 年发布的 RFC 2474、RFC 2475，以及后来发布的其他 Internet 文档中，又围绕该字段另行定义了区分服务标准，以及另外一套置位方式，同时得到了广泛应用。

- **长度字段：**表示整个 IP 包的总长度。
- **标识符、长度以及分片偏移字段：**每个 IP 包都有一个 ID（标识符）。当 IP 包以分片方式传送时，接收方能凭借这三个字段来进行重组。
- **生存时间（TTL）字段：**该字段的起始值为 64、128 或 256（随发包主机的操作系统而异），数据包在转发过程中，路径沿途的每一台路由器都会将该字段值减一。这是为了防止网络中的数据包形成转发环路。若收到了 TTL 字段值为 1 的数据包，路由器在将其值递减为 0 同时，还会做丢弃处理。
- **高层协议类型字段：**用来指明 IP 包头所封装的高层协议类型，若其值为 6，就表示 IP 包封装的是 TCP 报文段；若为 1，则表示封装的是 ICMP 报文。
- **校验和字段：**该字段包含的是 IP 包的校验和。IP 包的发送方会采用某种错误检测机制，针对整个 IP 包计算一个值，并在发送时将该值填入校验和字段。收到 IP 包时，接收方也会先用相同的机制计算出一个值，再将该值与 IP 包的校验和字段值进行比对，若两值不等，则认为 IP 包在传送时发生了错误。
- **源、目 IP 地址字段：**顾名思义，这 2 个字段值分别为 IP 包的源和目的 IP 地址。
- **选项字段：**IPv4 数据包一般不含该字段。

接下来，再来看一下 TCP 头部的结构。

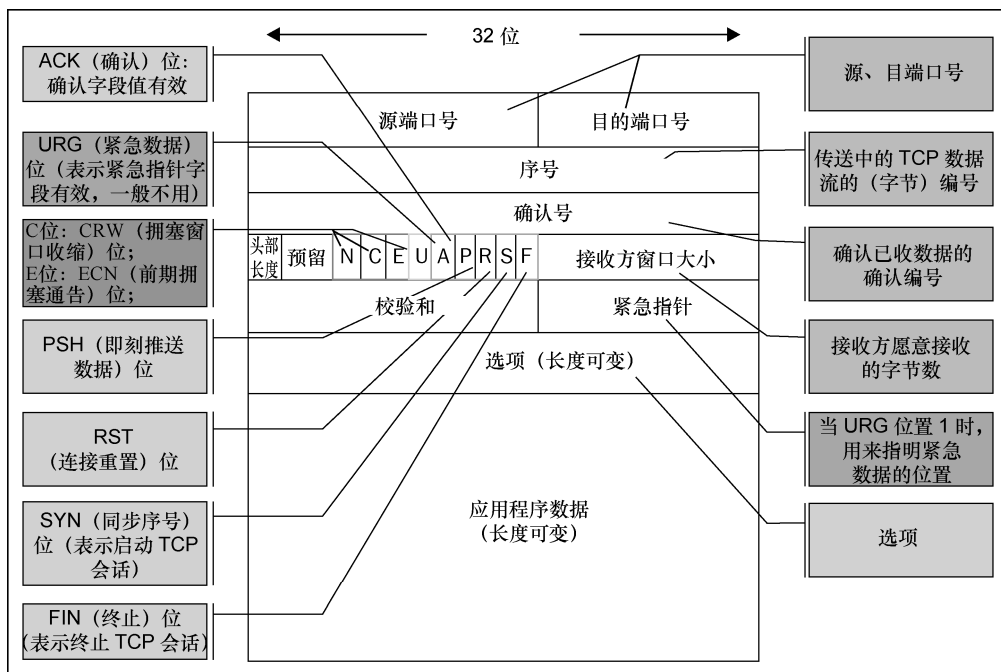


图3.9

下面来介绍一下 TCP 头部中的若干重要字段。

- **源、目端口号字段:** 这两个字段值, 再加上 IP 包头中的源、目 IP 地址即能唯一地标识一条 TCP 连接。
- **序列号字段:** 用来统计发送方通过 TCP 连接交付给接收方的数据的字节数。
- **确认号字段:** 该字段指明了 (执行确认的) TCP 发送方期待接收的下一个 TCP 数据包中的序列号字段值。本书第 9 章会对该字段的用途做深入探讨。
- **头部长度:** 用来表示 TCP 头部的长度, 同时还能指明 TCP 头部中是否包含有选项字段。
- **预留字段:** 该字段为预留的标记位字段。
- **8 个标记位:** 作用包括: 发起连接 (SYN 位)、终止连接 (FIN 位)、重置连接 (RST 位)、快推数据至应用层 (PSH 位)。本书第 9 章会详细介绍这些 TCP 标记位。
- **接收方窗口大小字段:** 用来表示接收方分配给应用程序的进程, 用来接收 TCP 数据的缓存的大小。

- **校验和字段：**用来存储经过校验和计算产生的值，计算范围覆盖 TCP 头部、数据以及 IP 头部中的某些字段。
- **选项字段：**包括时间戳选项字段、接收方窗口扩张选项字段及最长报文段大小（MSS）选项字段等。MMS 选项字段指明了该字段的通告方（即发出含 MMS 选项字段的 TCP 报文段的主机）希望（逆向）接收的 TCP 净载的最大长度。本书第 9 章将会对 MMS 选项字段做进一步的探讨。

3.4.4 拾遗补缺

TTL 字段是 IP 包头中非常有用的字段。通过检查该字段，就能弄清 IP 包所穿越的路由器的台数。在默认情况下，由不同操作系统生成的 IP 包的 TTL 字段值都比较固定（只有 64、128 和 256 这三种可能），而 IP 包在 Internet 上传输所穿路由器的台数最多也不能超过 30（在私有网络中，这一数字将会更低）。因此，若一 IP 包的 TTL 字段值为 120，则其所穿路由器的台数必定为 8；若 TTL 字段值为 52，则所穿路由器的台数将会是 12。

3.4.5 进阶阅读

- 欲进一步了解与 TCP/IP 协议栈有关的内容，请参阅第 9 章。

3.5 配置协议所独有的显示过滤器

本节会介绍如何针对常用的（应用层）协议（比如，DNS 协议、HTTP 协议、FTP 协议），来配置 Wireshark 显示过滤器。

本节的目标是要教会读者在排除网络故障时，如何利用显示过滤器来助一臂之力。在随后的章节里，也会出现与排除网络故障有关的内容。

3.5.1 配置准备

要配置显示过滤器，只需运行 Wireshark 软件，无需其他任何准备。

3.5.2 配置方法

本节会介绍如何针对若干常用的（应用）层协议，配置 Wireshark 显示过滤器。

HTTP 显示过滤器

以下所列为一些实战中常用的 HTTP 显示过滤器。

- 要让 Wireshark 只显示访问某指定主机名的 HTTP 协议数据包，显示过滤器应如此配置：`http.host == <"hostname">`。
- 要让 Wireshark 只显示包含 HTTP GET 方法的 HTTP 协议数据包，显示过滤器应如此配置：`http.request.method == "GET"`。
- 要让 Wireshark 只显示 HTTP 客户端发起的包含指定 URI 请求的 HTTP 协议数据包，显示过滤器应如此配置：`http.request.uri == <"Full request URI">`。譬如，`http.request.uri == "/v2/rating/mail.google.com"`。
- 要让 Wireshark 只显示 HTTP 客户端发起的包含某指定字符串的 URI 请求的 HTTP 协议数据包，显示过滤器应如此配置：`http.request.uri contains "URI String"`。譬如，`http.request.uri contains "mail.google.com"`（只显示包含字符串“mail.google.com”的 URI 请求的 HTTP 协议数据包）。
- 要让 Wireshark 只显示网络中传播的所有包含 cookie 请求的 HTTP 协议数据包（请注意，cookie 总是从 HTTP 客户端发往 HTTP 服务器），显示过滤器应如此配置：`http.cookie`。
- 要让 Wireshark 只显示所有包含由 HTTP 服务器发送给 HTTP 客户端的 cookie set 命令的 HTTP 协议数据包，显示过滤器应如此配置：`http.set_cookie`。
- 要让 Wireshark 只显示所有由 Google HTTP 服务器发送给本地 HTTP 客户端，且包含 cookie set 命令的 HTTP 协议数据包，显示过滤器应如此配置：`(http.set_cookie) && (http contains "google")`。
- 要让 Wireshark 只显示包含 ZIP 文件的 HTTP 数据包，显示过滤器应如此配置：`http matches "\.zip" && http.request.method == "GET"`。^①

DNS 显示过滤器

此处来举几个 DNS 显示过滤器示例。

要让 Wireshark 只显示所有 DNS 查询和 DNS 响应数据包，显示过滤器应如此配置。

- `dns.flags.response == 0`（DNS 查询）
- `dns.flags.response == 1`（DNS 响应）

^① 译者注：原文是“To check all the HTTP packets that contain a ZIP file, use `http matches "\.zip" && http.request.method == "GET"`”。译者认为原文有误，鉴于作者的笔法和不着调的思路，译文只能按字面意思翻译。

要让 Wireshark 只显示所有 answer count 字段值大于或等于 4 的 DNS 响应数据包^①，显示过滤器应如此配置：`dns.count.answers >= 4`。

FTP 显示过滤器

以下所列为实战中常用的 FTP 显示过滤器。

- 要让 Wireshark 只显示所有包含特定的 FTP 请求命令的 FTP 数据包，显示过滤器应如此配置：`ftp.request.command == <"requestedcommand">`。
- 要让 Wireshark 只显示所有通过 TCP 端口 21 传送的包含 FTP 命令的 FTP 数据包，显示过滤器应如此配置：`ftp`。要让 Wireshark 只显示所有从 TCP 端口 20 或从其他端口发出的包含实际 FTP 数据的 FTP 数据包，显示过滤器应如此配置：`ftp-data`^②。

3.5.3 幕后原理

Wireshark 显示过滤语句所采用的正则表达式的语法，等同于 Perl 语言所采用的正则表达式。

以下所列为正则表达式中元字符的含义。

- `^`：用来匹配行的开头。
- `$`：用来匹配行的结尾。
- `|`：用来表示二者任选其一。
- `()`：起分组的作用。
- `*`：匹配 0 次或多次前一模式（字符）。
- `+`：匹配 1 次或多次前一模式（字符）。
- `?`：匹配 0 次或 1 次前一模式（字符）。
- `{n}`：精确匹配 n 次前一模式（字符）。
- `{n,}`：匹配至少 n 次前一模式（字符）。
- `{n,m}`：匹配既不能低于 n 次也不能高于 m 次前一模式（字符）。

^① 译者注：即只显示答案部分（answer section）包含的 DNS 资源记录不低于 4 条的 DNS 响应数据包。要是读者不明所以，请先弄清 DNS 协议数据包的结构。

^② 译者注：原文是“To fetch FTP commands from port 2, use ftp, and to fetch FTP data from port 20 or any other configured port, use ftp-data”，原文没有抄错！作者说的每一句话都像是圣意，似乎不让读者揣测个够，他就当不好这个皇帝。译到这里，译者连想死的心都有了。译文酌情改动。

可利用上述元字符来配置非常复杂的显示过滤器，下面例举若干示例。

- 要让 Wireshark 只显示“携带”请求下载 ZIP 文件的 GET 命令的 HTTP 请求数据包，显示过滤器应如此配置：`http.request.method == "GET" && http.matches "\.zip" && !(http.accept_encoding == "gzip, deflate")`^①。
- 要让 Wireshark 只显示发往域名以 .com 结尾的 Web 站点的 HTTP 数据包，显示过滤器应如此配置：`http.host matches "\.com$"`。

3.5.4 进阶阅读

- Perl 语言所支持的正则表达式的语法列表及手册请见 <http://www.pcre.org/> 和 <http://perldoc.perl.org/perlre.html>。

3.6 配置字节偏移型过滤器

字节偏移型显示过滤器的通用格式为 `Protocols[x:y] == <value>`。这种过滤器实际上就是先通过 `x` 来定位到数据包协议头部中的某个字段（即该字段位于协议头部起始处第 `x` 个字节），并检查接下来 `y` 个字节的值是否等于 `value`。Wireshark 会根据检查结果来显示抓包文件中的相关数据。

这种过滤器的应用场合非常广泛，只要熟知各种协议头部的格式，对其中各字段的位置及长度了然于胸，就能随心所欲地使用它在抓包文件中筛选出自己想看的数据包。

3.6.1 配置准备

除了要运行 Wireshark 软件，打开抓包文件以外，本节无需任何准备工作。字节偏移型显示过滤器的通用格式为：

`Protocols[x:y] == <value>`

其中，`x` 指明了显示过滤器检查协议头部的位置（应从协议头部开始处的第几个字节开始检查），`y` 则表示显示过滤器所要检查的字节数。

3.6.2 配置方法

先举几个字节偏移型显示过滤器的例子，如下所示。

^① 译者注：原文是“To look for HTTP GET commands that contain ZIP files, use `http.request.method == "GET" && http.matches "\.zip" && !(http.accept_encoding == "gzip, deflate")`”。译者认为原文有误，依译者愚见，只要把显示过滤器配置为 `http.contains "\.zip" && http.request.method == "GET"`，甚至直接配置为 `http.contains "\.zip"` 就够了。

- 要让 Wireshark 只显示（在以太网内传送的）IPv4 多播数据包，字节偏移型显示过滤器应如此配置：`eth.dst[0:3] == 01:00:5e`（RFC 1112 第 6.4 节规定，IPv4 多播数据包在以太网内传送时，其以太网帧的多播目的 MAC 地址一定不出 MAC 地址空间 01-00-5E-00-00-00 ~ 01-00-5E-FF-FF-FF）。
- 要让 Wireshark 只显示（在以太网内传送的）IPv6 多播数据包，字节偏移型显示过滤器应如此配置：`eth.dst[0:2] == 33:33:00`（RFC 2464 第 7 节规定，IPv6 多播数据包在以太网内传送时，其以太网帧的多播目的 MAC 地址一定是以 33-33 打头）。

3.6.3 幕后原理

网管人员只要熟知各种协议报文结构，便可利用字节偏移型显示过滤器，直接根据数据包协议头部的第某某字节到某某字节的内容，在 Wireshark 抓包文件中做一番筛选。对于上一节所举的 Wireshark 字节偏移型示例，就必须熟悉以太网帧的结构。

3.7 配置显示过滤器宏

配置显示过滤器宏，是创建复杂的显示过滤器的便捷通道，可以一次配置，多次使用。

3.7.1 配置准备

要配置显示过滤器宏，进入 **Analyze** 菜单，选择 **Display Filter Macros** 菜单项，在弹出口中点击 **New** 按钮，如图 3.10 所示。

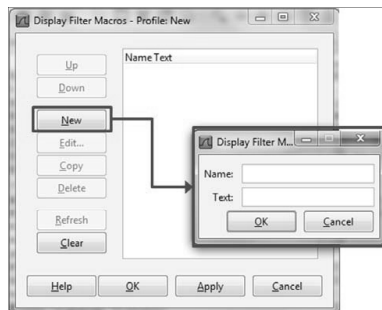


图3.10

3.7.2 配置方法

1. 配置显示过滤器宏：要先在 **Name** 文本框内输入一个名称，这也就是显示过滤器宏的名称，再朝 **Text** 文本框里输入需多次使用的显示过滤语句，输入完毕后点 **OK** 按钮。

2. 调用显示过滤器宏：在抓包主窗口的 **Filter** 输入栏内输入宏调用语句 `$(macro_name:parameter1;parameter2;parameter3 ...)`。
3. 现举例加以说明。配置一个名叫 **test01** 的显示过滤器宏，其作用是让 Wireshark 只显示指定源 IP 地址和指定目的端口号的 TCP 数据包。
4. 配置显示过滤器宏：在 **Name** 文本框内输入 **test01**，作为显示过滤器宏的名称；朝 **Text** 文本框内输入 `ip.src==$1 && tcp.dst port==$2`。其中，`$1` 和 `$2` 用来取代传递给显示过滤器宏的参数。然后，点 **OK** 按钮。最后，在 **Display Filter macros** 窗口内再点一次 **OK** 按钮，保存这一显示过滤器宏。
5. 调用显示过滤器宏 **test01**：若要让 Wireshark 只显示所有源 IP 地址为 10.0.0.4，目的端口号为 80 的数据包，则需在抓包主窗口的 **Filter** 输入栏内输入 `${test01:10.0.0.4;80}`。其中，`10.0.0.4` 和 `80` 分别表示要传递给宏 **test01** 的 IP 地址参数和 TCP 端口号参数。

3.7.3 幕后原理

显示过滤器宏的运作原理非常简单：先用符号“\$”加编号作为显示过滤器的位置参数；当随后在 **Filter** 输入栏内调用显示过滤器宏时，相关显示过滤参数会按编号的顺序传递进来。

第4章

基本信息统计工具的用法

本章涵盖以下主题：

- ▶ **Statistics** 菜单中 **Summary**（信息汇总）工具的用法；
- ▶ **Statistics** 菜单中 **Protocol Hierarchy**（协议层级）工具的用法；
- ▶ **Statistics** 菜单中 **Conversations**（会话）工具的用法；
- ▶ **Statistics** 菜单中 **Endpoints**（端点）工具的用法；
- ▶ **Statistics** 菜单中 **HTTP** 工具的用法；
- ▶ **Statistics** 菜单中 **Flow Graph**（数据流图）工具的用法；
- ▶ 基于 IP 的信息统计报表的创建方法。

4.1 简介

Wireshark 之所以普及，其自带的一整套信息统计工具功不可没。Wireshark 的信息统计工具既包括可列出端点及会话的简单统计工具（Endpoints 工具和 Conversations 工具），也包括 Flow Graph 和 I/O Graph 这样的高级工具。

在本章及下一章将会学习如何使用上述信息统计工具。本章会介绍能提供网络基本信息的简单信息工具。所谓网络基本信息是指：网络中哪些设备之间有过交流、哪些设备话多哪些设备话少，以及在链路上呼啸而过的数据包的长度等信息。在下一章，将介绍 Flow Graph 和 I/O Graph 等高级信息统计工具，这样的工具可让网管人员更详细地了解网络中的“风吹草动”。

有一些出现在 **Statistics** 菜单中的工具本书不会提及，这些工具要么是用途一目了然（比如，**Packet lengths**[数据包长度]工具），要么几乎没用（比如，**ANSP** 和 **BACnet** 工具等）。

Wireshark 自带的信息统计工具，都在其主窗口的 **Statistics** 菜单名下。

4.2 Statistics菜单中Summary工具的用法

本节会介绍如何通过 **Wireshark**，来获悉在网络中穿梭往来的数据包的总体信息。

4.2.1 准备工作

启动 **Wireshark** 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再选择 **Statistics** 菜单。

4.2.2 使用方法

Summary 工具归于 **Statistics** 菜单名下，要想使用此工具，请在 **Statistics** 菜单中点击 **Summary** 菜单项，如图 4.1 所示。

Summary 窗口会立刻弹出，如图 4.2 所示。

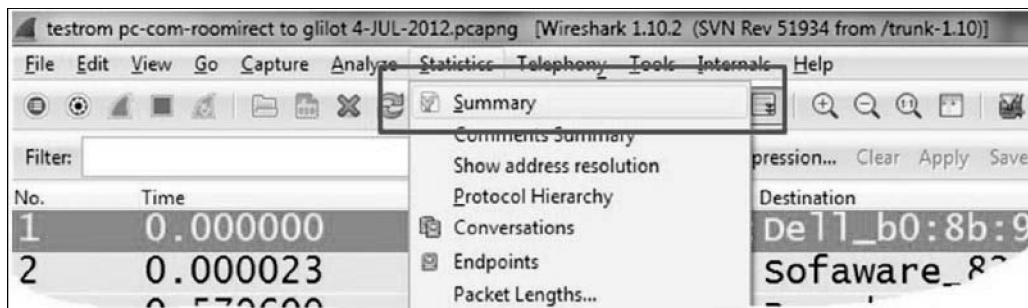


图4.1

以下是对 **Summary** 窗口中重要信息的解读。

1. 由图 4.2 可知，**Summary** 窗口的上半部分包括下述区域。
 - **File:** 通过该区域中的信息，可以了解抓包文件的各种属性，比如，抓包文件的名称、路径信息，以及抓包文件所含数据包的规模等信息。
 - **Time:** 通过该区域中的信息，可以获悉抓包的开始、结束以及持续时间。
 - **Capture:** 通过该区域中的信息，可以得知抓包文件由哪块网卡生成、OS 版本、

Wireshark 版本等信息。此外，该区域还包括了一个备注栏（Capture file Comments）。

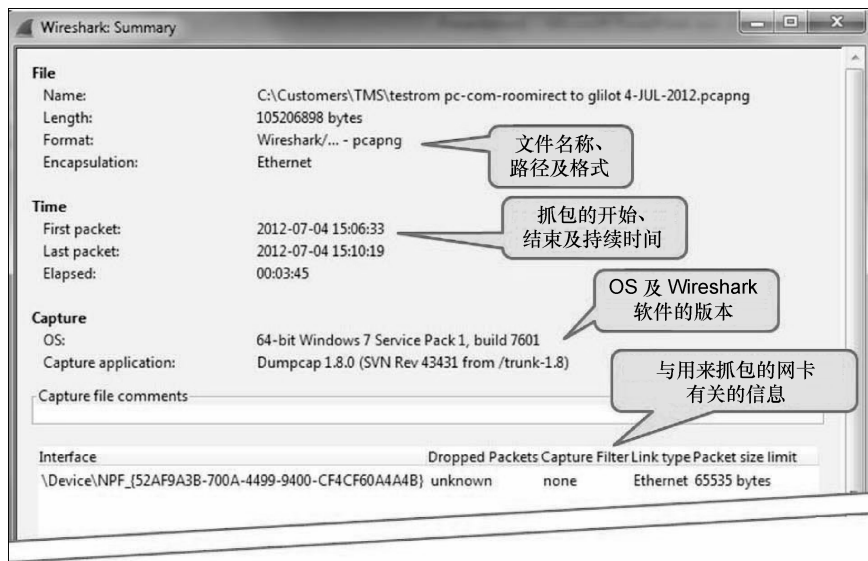


图4.2

2. 由图 4.3 可知，**Summary** 窗口的下半部分只有 **Display** 这一个区域，通过其内容，可了解到抓包文件的汇总统计信息，这些信息包括：

- 抓到的数据包的总数（即抓包文件所含数据包的总数）；
- （在应用过显示过滤器之后）显示出的数据包的数量以及占比情况；
- 带有人为标记的数据包的数量。

4.2.3 幕后原理

Summary 窗口所提供的信息默认来源于所有已抓取的数据包，但若应用了显示过滤器，则来源于经过过滤的数据包。要是有人问：“通过 Wireshark 能得知网络中平均每秒过往的数据包的个数及字节数吗？”在 **Summary** 窗口中就能找到答案。

4.2.4 拾遗补缺

通过 **Summary** 窗口，可以了解到在整个抓包过程中，平均每秒过往的所有数据包以及经过显示过滤器过滤的数据包的个数及字节数。

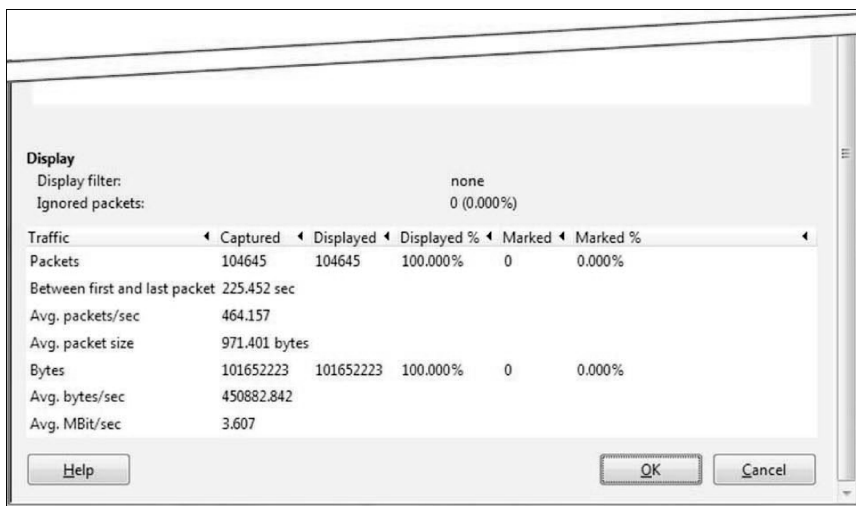


图4.3

4.3 Statistics菜单中Protocol Hierarchy工具的用法

本节会介绍如何通过 Wireshark 来获悉在网络中穿梭往来的数据包所归属的协议层级。

4.3.1 准备工作

启动 Wireshark 软件, 先打开一个抓包文件 (或选择一块网卡, 点击 **Start** 按钮开始抓包), 再选择 **Statistics** 菜单。

4.3.2 使用方法

Protocol Hierarchy 工具归于 **Statistics** 菜单名下, 要想使用此工具, 请在 **Statistics** 菜单中点击 **Protocol Hierarchy** 菜单项, 如图 4.4 所示。

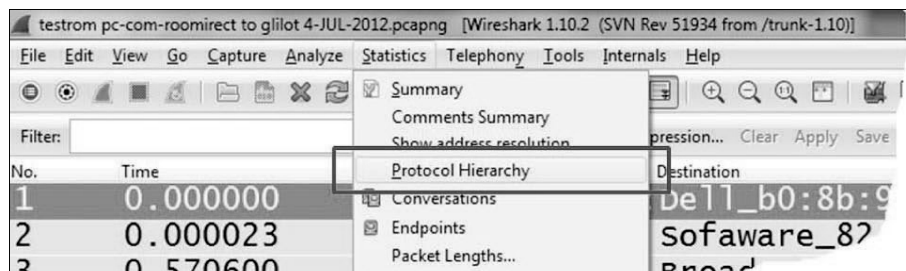


图4.4

Protocol Hierarchy Statistics 窗口将立刻弹出, 如图 4.5 所示。透过该窗口, 可以了解到

抓包文件所含数据包归属的协议类型的分布情况。

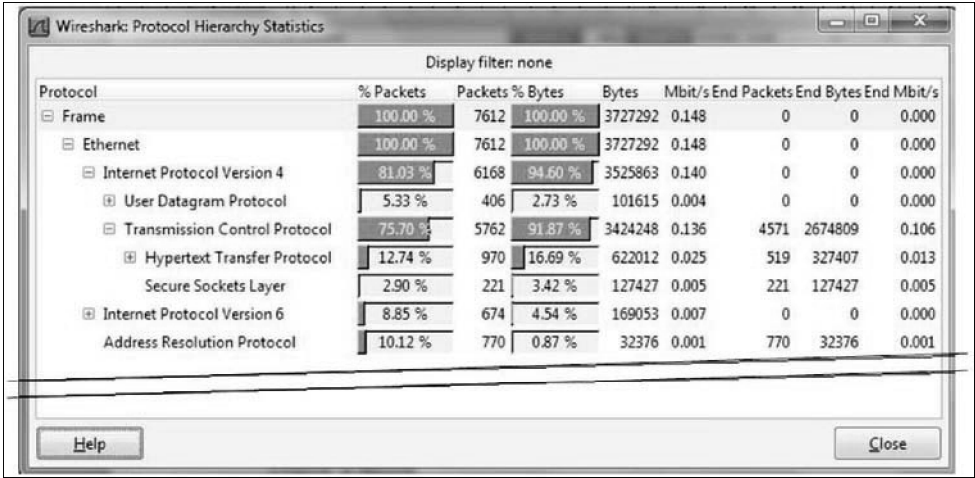


图4.5

下面是对 **Protocol Hierarchy Statistics** 窗口每一列的解释。

- **Protocol:** 用来表示数据包所归属的协议名称。
- **% Packets:** 指明了抓包文件中所含数据包在每一种协议类型中的占比情况（按数据包的个数来统计）。
- **Packets:** 指明了每一种协议类型的数据包的个数。
- **% Bytes:** 指明了抓包文件中所含数据包在每一种协议类型中的占比情况（按数据包的字节数来统计）。
- **Bytes:** 指明了每一种协议类型的数据包的字节数。
- **Mbit/s:** 指明了某种协议类型的数据包在抓包时段内的传输速率。
- **End Packets:** 指明了隶属于该协议类型的数据包的纯粹数量。试举一例，图中 TCP 协议的 Packets 和 End Packets 数量分别为 5762 和 4571 个，这就表示此抓包文件中以 TCP 头部封装的数据包的总数为 5762 个，但只有 4571 个纯 TCP 数据包，即这些数据包在 TCP 头部之后再无高层协议头部，而其他的 1191 个数据包在 TCP 头部之后还紧跟了高层协议头部（比如，HTTP 头部）。
- **End Bytes:** 指明了隶属于该协议类型的数据包的纯粹字节数。
- **End Mbit/s:** 指明了隶属于该协议类型的数据包在抓包时段内的纯粹传输速率。



注意

End Packets、**End Bytes** 以及 **End Mbit/s** 分别指明了纯粹为某种协议（此种协议的头部为数据包中的最高层协议头部）的数据包的个数、字节数以及在抓包时段内的传输速率。以 TCP 为例，虽然 FTP、HTTP、SSL 数据包都可算作 TCP 数据包，但这些应用层协议在建立或终止 TCP 连接时发出 TCP 数据包，是不含高层头部信息的（比如，用来建立 HTTP 连接的 TCP SYN 数据包等），此类数据包便算作“纯”TCP 数据包。图中 End Packets、End Bytes 以及 End Mbit/s 三列的头四行之所以都是 0，则是因为该转包文件抓到虽然分别是 Frame（帧）、Ethernet（以太网帧）以及 Internet Protocol Version 4（IPv4 数据包），但却全都包含了更高层的协议头部。

4.3.3 幕后原理

简而言之，由 **Protocol Hierarchy** 工具生成的统计信息也是根据抓包文件计算而得。读者需关注以下两点。

- 该工具生成的所有“%”（占比）信息都是针对同一层次的协议类型而言。比方说，图 4.5 中 IPv4、IPv6 和 ARP 数据包的“% Packets”分别为 81.03%、8.85%和 10.12%，这表示这三种（第三层）数据包在第二层以太网帧中所占比重分别为 81.03%、8.85%和 10.12%（三者相加正好等于 100%）。
- 图 4.5 中 TCP 数据包占总数据包的 75.5%，但 HTTP 数据包只占总数据包的 12.74%，除此之外，只有 2.90%的 SSL 数据包算是 TCP 数据包了。这是因为，在统计高层协议数据包时，只认高层协议头部，对本例而言，所有端口号为 80 或 443 但不含 HTTP 或 SSL 头部的 TCP 数据包（比如，用来建立连接的 TCP SYN 数据包，或不含 HTTP 头部只含 HTTP 数据的 TCP 数据包），都未被算作为 HTTP 或 SSL 数据包。

4.3.4 拾遗补缺

要让 Wireshark 在 **Protocol Hierarchy** 工具生成的统计信息中，把不含高层协议头部、只含高层协议数据的 TCP 数据包，也归为该高层协议数据包，请在首选项窗口（点 **Edit** 菜单，选择 **Preferences** 菜单项）中，点 protocol 前的+号，选中 TCP 协议，取消勾选 **Allow subdissector to reassemble TCP streams** 复选框。也可以在抓包主窗口的数据包结构区域中单击右键，在右键菜单中选择 **protocol preferences** 菜单项，取消勾选 **Allow subdissector to reassemble TCP streams** 子菜单项。

4.4 Statistics菜单中Conversation工具的用法

本节会介绍如何获取网络中的设备间的对话信息。

4.4.1 准备工作

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再选择 **Statistics** 菜单。

4.4.2 使用方法

Conversation 工具归于 **Statistics** 菜单名下，要想使用此工具，请在 **Statistics** 菜单中点击 **Conversations** 菜单项，如图 4.6 所示。

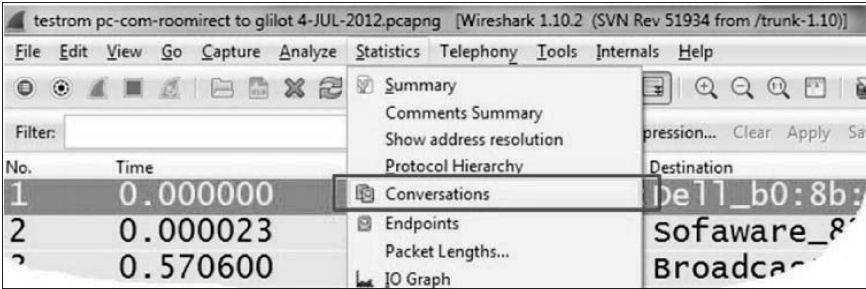


图4.6

Conversation 窗口会立刻弹出，如图 4.7 所示。

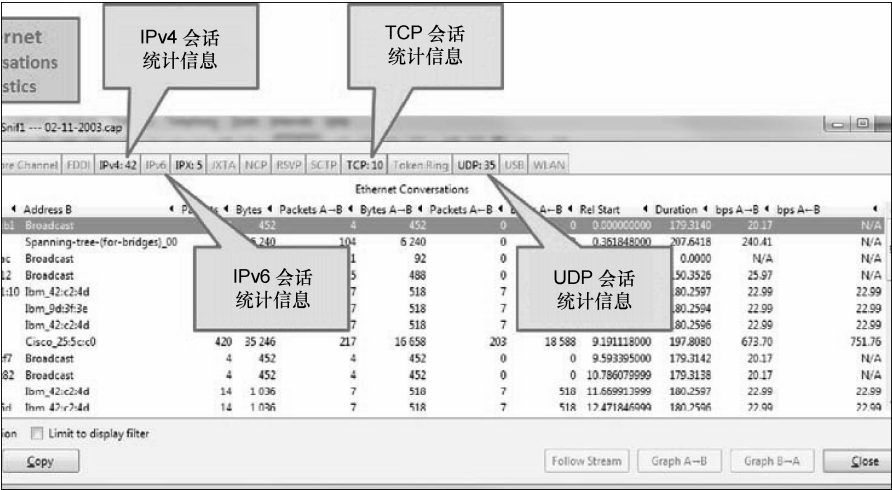


图4.7

可點選圖 4.7 中相應的標籤來觀看網絡中的主機之間在第二、三、四層上的對話。可點選：

- **Ethernet 標籤：**來觀察具有不同 MAC 地址的主機間發生過什麼樣的“交流”。
- **IPv4 標籤：**來觀察具有不同 IPv4 地址的主機間有過什麼樣的“溝通”。
- **TCP 或 UDP 標籤：**來觀察具有不同 IPv4 地址的主機間所建立的各种 TCP（或 UDP）會話。

通過觀察主機之間在第二、三、四層上的對話，既可以發現發生在第二層的廣播風暴，也可以查明發生在第三、四層的 Internet 鏈路負載過高等問題。



注意

若網管人員發現內網有大量數據包都涌向某一特定公網 IP 地址的 80 端口，則應試着用自己主機里的瀏覽器也訪問一下這一 IP 地址，看看該 Web 站點為什麼受內網用戶的熱捧。要是無所獲，請訪問任何一個標準的域名解析 Web 站點（比如，<http://who.is/>），並輸入這一公網 IP 地址，就可以得知內網用戶的 HTTP 流量都發到什麼樣的網站去了。

要是勾選了 **Conversation** 窗口左下角的 **Limit to display filter** 複選框，那麼該窗口將會顯示經過顯示過濾器過濾的信息。

要想用相應的名稱來取代 **Conversation** 窗口顯示的 MAC 地址、IP 地址，以及 TCP/UDP 端口號，需勾選左下角的 **Name resolution**。但在此之前，還應在主窗口的 **View** 菜單的 **Name Resolution** 菜單項下點選相應的子菜單項。

要是只想觀看某單一的 TCP 或 UDP 對話（conversation），需執行兩個步驟。

1. 在抓包主窗口的數據包列表區域中選擇隸屬於該 TCP 或 UDP 對話（即 TCP 或 UDP 流）的某個數據包，再點擊右鍵，於右鍵菜單中選擇 **Follow TCP Stream** 或 **Follow UDP Stream** 菜單項。這便定義好了一個已應用於抓包文件的顯示過濾器。
2. 選擇 **Statistics** 菜單，點擊 **Conversation** 菜單項，打開 **Conversation** 窗口，勾選左下角的 **Limit to display filter** 複選框。

4.4.3 幕後原理

一次對話（conversation）是指發生於一對特定端點（主機、服務器或網絡設備）間的所有流量。比方說，一次 IP 對話是指交換於具有不同 IP 地址的兩台主機間的所有流量；而一次 TCP 或 UDP 對話則包括了 4 大特徵（源、目 IP 地址外加源、目端口號）全都匹配的数据包。

4.4.4 拾遗补缺

借助于 **Conversations** 工具，能发现各种不易觉察的网络问题。

Ethernet 对话统计信息

在 **Conversations** 窗口中点击 **Ethernet** 标签，便会呈现 Ethernet 对话统计信息。通过这些信息，可以发现：

- 广播包的数量是否过于庞大：若是，则很有可能遭遇了广播风暴（要是情况比较严重，估计在 Ethernet 对话统计信息中，都看不到一个单播包）。



注意

将 Wireshark 主机接入发生了严重广播风暴的网络时，Wireshark 将会每秒抓取到数以万计的数据包，这不但会导致其停止显示数据包，而且还会使得屏幕卡顿。此时，只有断开 Wireshark 主机网卡的网线，才能看清抓到的数据包。

- 是否存在绝大多数数据包的源 MAC 地址全都相同的现象（即绝大多数数据包是否都是由具有某特定 MAC 地址的主机发出）：若是，则可能是某块主机的网卡发生了故障。此时，只要瞥一眼“address A”的前半部分——网卡芯片制造商的 ID（比如 ibm 或 cisco），或许就知道是哪台主机的网卡出故障了。



注意

虽然 MAC 地址的前半部分标明了网卡芯片制造商的 ID，但这只表示网络中有某台主机安装了该网卡芯片制造商生产的网卡，而 PC 机或笔记本制造商却未必生产网卡。因此，还得登录以太网交换机，查看该 MAC 地址是从哪个端口学得，这样才好找到那台因网卡发生故障而生成巨流的主机。

IP 对话统计信息

在 **Conversation** 窗口中点击 **IPv4** 标签，便会呈现 IPv4 对话统计信息。通过这些信息，可以发现：

- 是否有某个（若干）IP 地址有极高的曝光率：若有，且这个（或这几个）IP 地址归服务器所有，则纯属正常；不过，出现这种情况，也有可能是有人用黑客工具扫描网络，或某台 PC 机生成了过多的流量所导致。
- 是否有人用黑客工具扫描网络（第 14 章会对此做深入探讨）：网络扫描也分为两种，一种是正常扫描，比如，SNMP 网管系统发出 ping 包探测网络设备是否健在；另一种是异常扫描，即有人用黑客工具扫描网络（或中了病毒的主机不由自主地扫描网

络)。通过图 4.8，可以清楚地看见有台主机正在扫描网络。

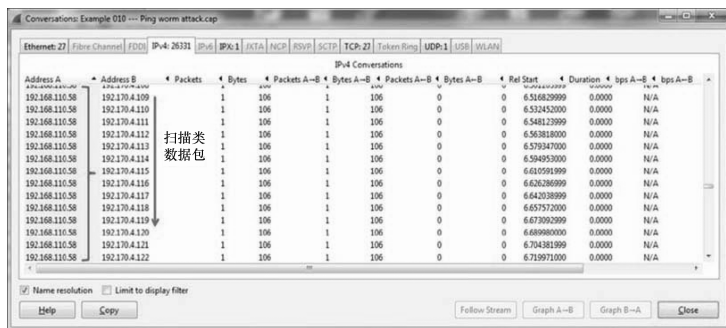


图4.8

图 4.8 显示了一个经典的扫描场景：有一个 IP 地址（**192.168.110.58**）正按序向同一 IP 子网内的其他 IP 地址（从 **192.168.110.109~122**）快速发出 ping 包（图 4.8 只显示出了一小部分 IP 对话）。实际上，是网络中那台设有 IP 地址 **192.168.110.58** 的主机感染上了蠕虫病毒，感染蠕虫病毒的主机会持续发送巨量 ping 包，其后果是导致网络链路严重拥塞。

TCP/UDP 对话统计信息

在 **Conversation** 窗口中点击 **TCP** 或 **UDP** 标签，便会呈现 TCP 或 UDP 对话统计信息。通过这些信息，可以发现：

- 是否有某一台主机打开过多的 TCP（或 UDP）连接。对单台主机而言，打开 10~20 个连接应属正常范畴，但若连接数过百，那就应该去好好查一查了。
- 是否有主机试图与稀奇古怪的 TCP/UDP 目的端口号建立连接。若有，则表示网络可能遇到了麻烦。如图 4.9 所示，通过观察 **TCP** 标签中的信息，可以很容易地发现有人在执行标准的 TCP 端口扫描。

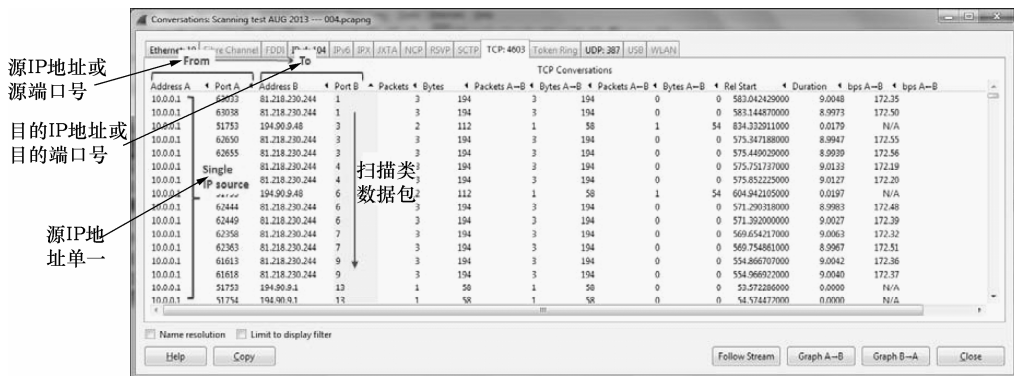


图4.9

4.5 Statistics菜单中Endpoints工具的使用法

本节会介绍如何查看抓包文件中与（数据包的发送或接收）端点（Endpoint）有关的统计信息。

4.5.1 准备工作

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再选择 **Statistics** 菜单。

4.5.2 使用方法

Endpoints 工具归于 **Statistics** 菜单名下，要想使用此工具，请在 **Statistics** 菜单中点击 **Endpoints** 菜单项，如图 4.10 所示。

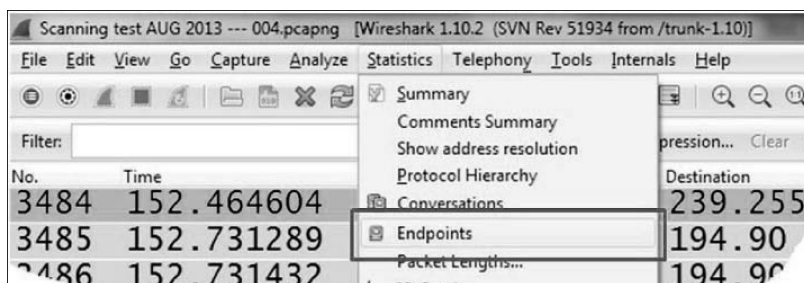


图 4.10

Endpoints 窗口将立刻弹出，如图 4.11 所示。

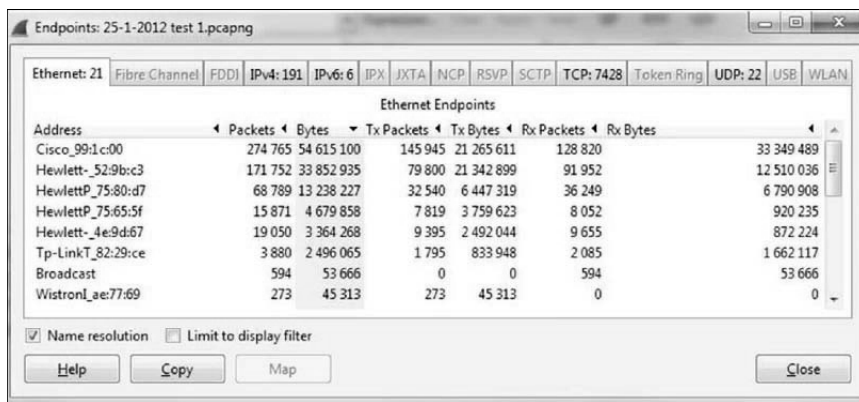


图 4.11

可点击 **Endpoints** 窗口内相应的标签，来观察与第二、三、四层端点（Ethernet 端点、IP 端点、TCP/UDP 端点）有关的统计信息。

4.5.3 幕后原理

借助于 **Endpoints** 窗口，可以了解到 Wireshark 探测到的与所有第二、三、四层端点有关的统计信息。通过这些统计信息，可以很好的解释以下现象。

- **Ethernet 端点（MAC 地址）少，IP 端点（IP 地址）多：**对于这种现象，可能的原因是有一台路由器来负责转发所有进、出本地 LAN（IP 子网）的 IP 流量。也就是说，对于源或目的 IP 地址不隶属于本地 IP 子网的所有数据包，其源或目的 MAC 地址都会是那台路由器内网 LAN 口的 MAC 地址，这属于正常情况。
- **IP 端点（IP 地址）少，TCP 端点（TCP 端口号）多：**说白一点，就是每个 IP 端点都试图建立或已经建立了多条 TCP 连接。对于这种现象，可能正常也可能不正常。若建立或试图建立多条 TCP 连接的 IP 端点为服务器，这就属于正常情况；否则，极有可能是有人在发动网络攻击（比如，TCP SYN 攻击）。

4.5.4 拾遗补缺

现以从某网络中心弄来的一份抓包文件为例，来教读者如何查看 Wireshark 的 **Endpoints** 工具。

如图 4.12 所示，通过点击 **Ethernet: 21** 标签，可判断出在这一网络中心的内网之中应部署有 4 台 HP 服务器外加一台 Cisco 路由器。判断依据是，在勾选了 **Name resolution** 复选框之后，**Address** 一列下的 MAC 地址中出现了网卡（网络设备）芯片制造商的 ID。

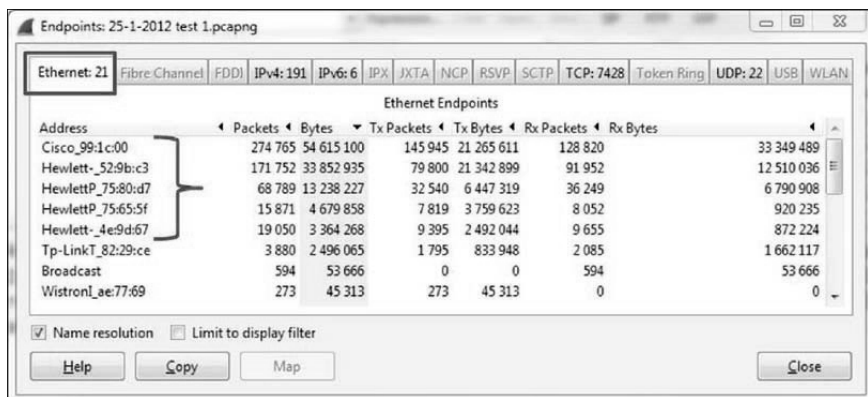


图4.12

在点击过 **IPv4: 191** 标签之后，可以看见隶属于 **192.168.10.0** 和 **192.168.30.0** 等 IP 子网的众多 IP 端点（见图 4.13）。

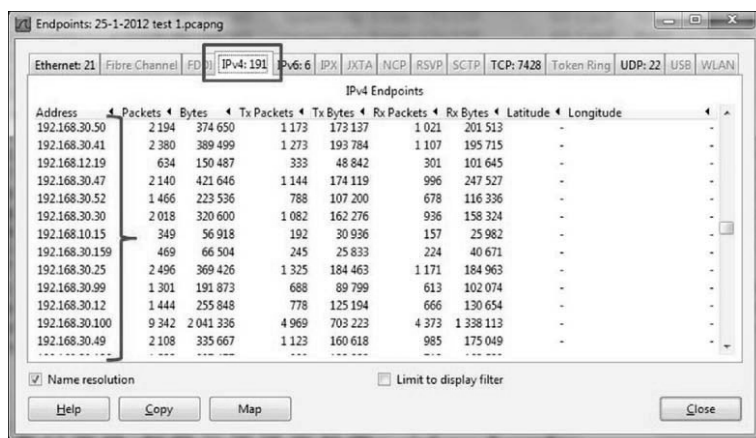


图4.13

4.6 Statistics菜单中HTTP工具的使用法

本节会介绍如何查看抓包文件中有关 HTTP 流量的统计信息。

4.6.1 准备工作

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再选择 **Statistics** 菜单。

4.6.2 使用方法

HTTP 工具归于 **Statistics** 菜单名下，要想使用此工具，请在 **Statistics** 菜单中点击 **HTTP** 菜单项，如图 4.14 所示。

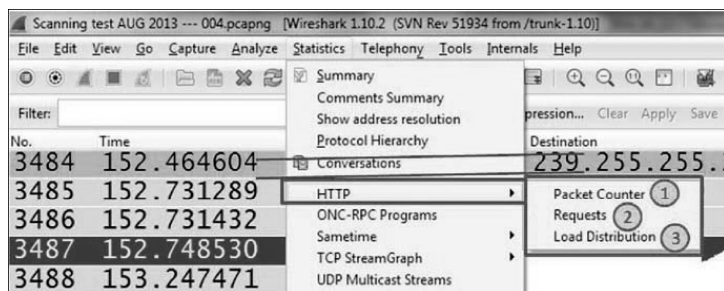


图4.14

HTTP 菜单项名下有以下三个子菜单项。

- **Packet Counter**（在图 4.14 中被标记为 1）：可用来了解抓包文件中 HTTP 数据包的总数，以及其中 HTTP 请求数据包和 HTTP 响应数据包分别为多少。
- **Requests**（在图 4.14 中被标记为 2）：可用来了解（主机）请求访问的 Web 站点的分布情况，以及所访问的 Web 站点上的具体资源（指向资源的 URL）。
- **Load Distribution**（在图 4.14 中被标记为 3）：可用来了解抓包文件中 HTTP 数据包（包括 HTTP 请求和 HTTP 响应数据包）在各 Web 站点间的分布情况（即访问过哪些 Web 站点）。

首先，来看一下 Packet Counter 子菜单项的用法

1. 点击该子菜单项，会立刻弹出 HTTP/Packet Counter Stats Tree 显示过滤器输入窗口，如图 4.15 所示。

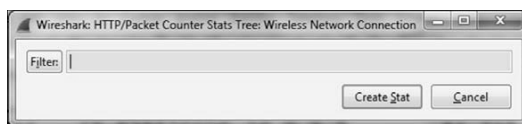


图4.15

2. 可在图 4.15 所示窗口内输入显示过滤器，Wireshark 会根据过滤结果，来显示有关 HTTP（流量）统计信息。要是不准备应用（输入）任何显示过滤器，请直接点击 **Create Stat** 按钮，Wireshark 会基于抓包文件中的所有 IP 包来生成有关 HTTP（流量）统计信息。点击 **Create Stat** 按钮，**HTTP/Packet Counter with filter** 窗口会立刻弹出，如图 4.16 所示。

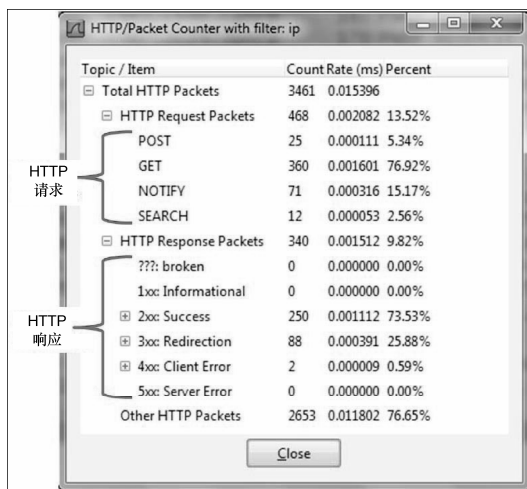


图4.16

要想获悉某指定节点所涉及的 HTTP 流量统计信息,请在 HTTP/Packet Counter Stats Tree with Filter 窗口的 Filter 输入栏内填入针对该节点配置的过滤表达式(语法等同于显示过滤器),再点击 **Create Stat** 按钮。

其次,来介绍一下 **Requests** 子菜单项的用法。

1. 点击该子菜单项,会弹出 HTTP/Requests Stats Tree 显示过滤器输入窗口,如图 4.17 所示。



图4.17

2. 根据需求在 Filter 输入栏内输入显示过滤语句,若要基于抓包文件中的所有 IP 包来生成有关 HTTP 请求的统计信息,请直接点击 **Create Stat** 按钮。这会导致 **HTTP/Request with filter** 窗口的弹出,如图 4.18 所示。

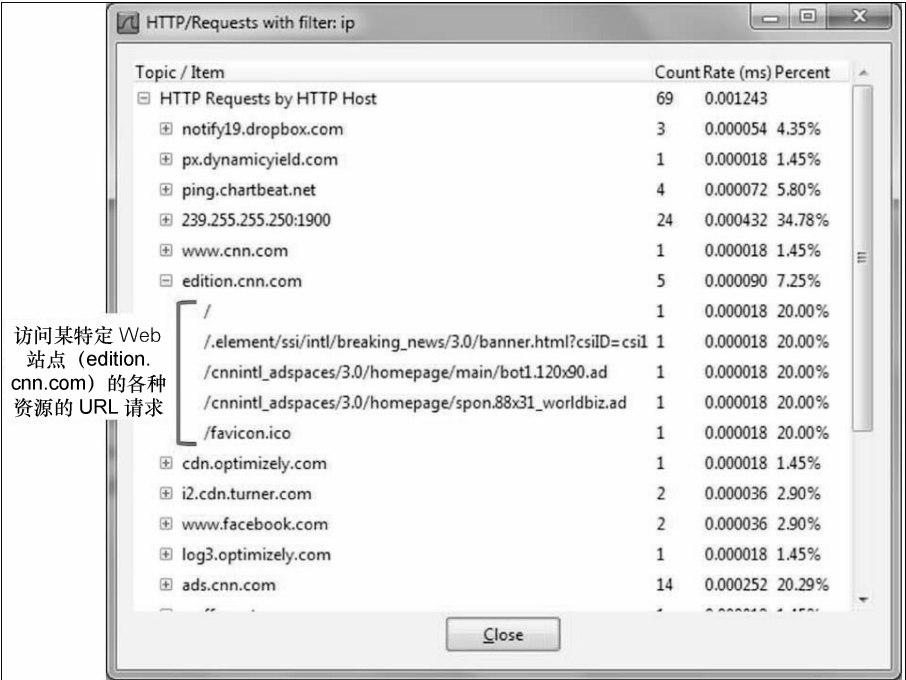


图4.18

3. 要想基于某个指定的 Web 站点，来生成有关 HTTP 请求的统计信息，可先在 Filter 输入栏里输入显示过滤表达式 `http.host contains <host_name>` 或 `http.host==<host_name>`（具体输入哪一个，要看是想精确匹配 Web 站点的域名，还是想只匹配域名中的某个字符串），再点击 **Create Stat** 按钮。
4. 试举一例，要想看看内网用户都访问了 Web 站点 `www.ndi-com.com` 上的哪些资源，请在 Filter 输入栏里输入显示过滤表达式 `http.host contains ndi-com.com`，如图 4.19 所示。

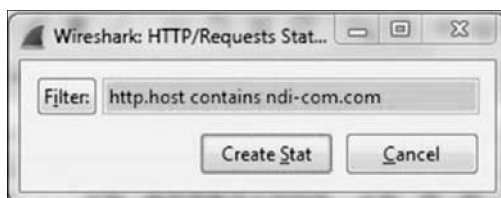


图4.19

5. 点击 **Create Stat** 按钮，会弹出 **HTTP/Request with filter: http.host contains ndi-com.com** 窗口，如图 4.20 所示。藉此窗口，内网用户访问过的 Web 站点 `www.ndi-com.com` 上的所有资源都一目了然。



图4.20

最后，来研究一下 **Load Distribution** 子菜单项的用法。

1. 点击该子菜单项，会弹出 **HTTP/Load Distribution Stats Tree** 显示过滤器输入窗口，如图 4.21 所示。
2. 据需求在 Filter 输入栏内输入显示过滤语句，若要基于抓包文件中的所有 IP 包来生成有关 HTTP 负载分配的统计信息，请直接点击 **Create Stat** 按钮。这会导致 **HTTP/ Load**

Distribution with filter 窗口的弹出，如图 4.22 所示。

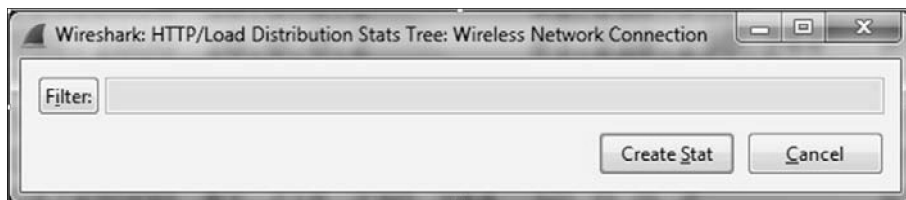


图4.21

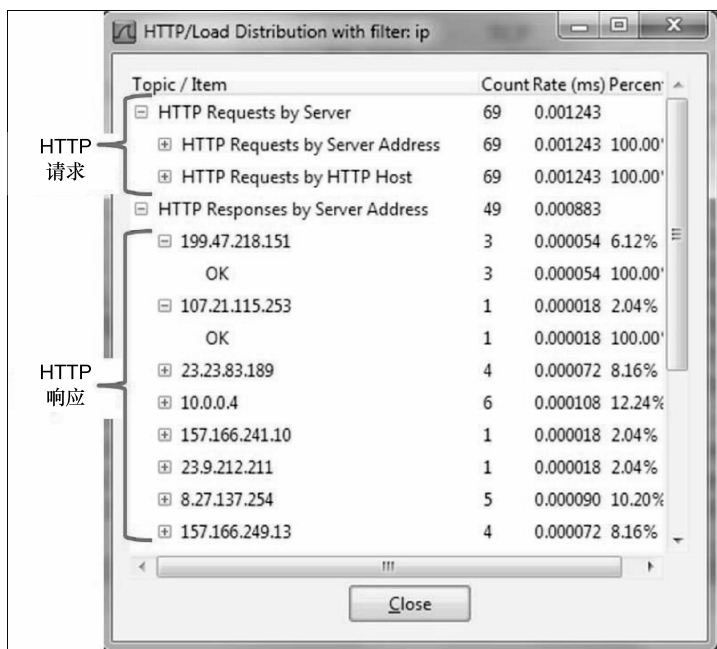


图4.22

4.6.3 幕后原理

在访问某个 Web 站点时，浏览器通常会发出多次 HTTP 请求，从多个位置下载资源。譬如，当访问 Web 站点 www.cnn.com 时，浏览器会被该 Web 站点传回的 HTTP 响应消息“牵引”至 edition.cnn.com，然后需要借助若干 URL，从多个位置请求（下载）资源。

4.6.4 拾遗补缺

要想对 HTTP 数据包做进一步的分析，还需借助于特殊的工具。**Fiddler** 是最常使用的 HTTP 数据包分析工具，下载链接为 <http://www.fiddler2.com/fiddler2/>。

Fiddler 是一款专门为排除 HTTP 协议故障而开发的软件，其用户界面能详尽的显示与 HTTP 有关的数据。

4.7 配置Flow Graph（数据流图），来查看TCP流

本节会介绍 **Statistics** 菜单中 **Flow Graph** 工具的用法。

4.7.1 配置准备

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再点击 **Statistics** 菜单下的 **Flow Graph** 菜单项，**Flow Graph** 窗口会立刻弹出，如图 4.23 所示。

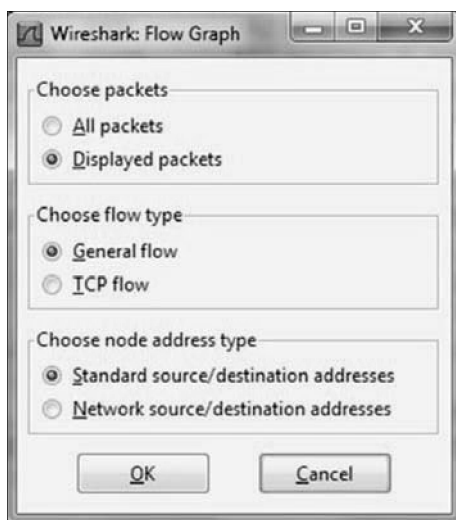


图4.23

4.7.2 配置方法

在 **Flow Graph** 窗口中有若干选项（单选框）可供选择，下面是对这些选项的解释。

1. **Choose packet**（选择数据包）区域内的单选框。

- 选中 **All packets** 单选框，Wireshark 会基于抓包文件中的所有数据包来生成 Flow Graph。
- 选中 **Displayed packets** 单选框，Wireshark 会基于抓包文件中经过显示过滤器过滤的数据包来生成 Flow Graph。

2. **Choose flow type**（选择数据流类型）区域内的单选框。
- 选中 **General flow** 单选框，Wireshark 会针对抓包文件中的所有数据包或所有经过显示过滤器过滤的数据包（视点选了 **Choose packets** 区域内的哪个单选框而定），来生成总体的数据流图。
 - 选中 **TCP flow** 单选框，Wireshark 会针对抓包文件中的所有数据包或所有经过显示过滤器过滤的数据包，来生成含 TCP 标记、序列号、ACK 号以及报文段长度的 TCP 数据流图。

4.7.3 幕后原理

只是根据抓包文件生成简单的统计信息而已，幕后原理无需赘言^①。

4.7.4 拾遗补缺

TCP 故障有时很是让人头疼，要想弄清故障的原委，就必须绘制出 TCP 端点间的数据流图。一般而言，绘制 TCP 数据流图的最佳方式是借助于某款绘图软件，这样的软件应具有优良的图形界面。当然，也可以用不同颜色的彩色铅笔在纸上手工绘制。

针对 Wireshark 开发的 **Cascade Pilot** 软件包正是绘制 TCP 数据流图的绝佳工具，详情请见 http://www.riverbed.com/us/products/cascade/wireshark_enhancements/cascade_pilot_personal_edition.php。

图 4.24 所示为作者手工绘制的 TCP 数据流图示例。

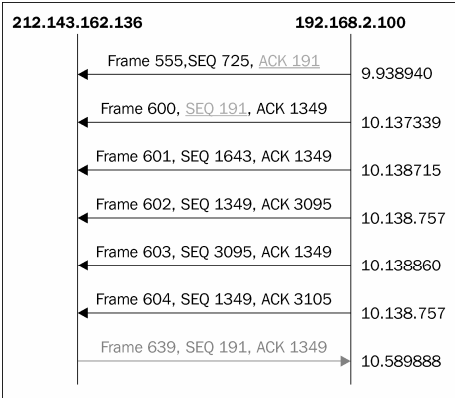


图4.24

^① 译者注：原文是“Simply by creating simple statistics from the captured file: nothing special to say here”，按原文字面直译。

万事开头难，TCP 数据流图只要多画几次，就会熟能生巧了。

4.8 生成与IP属性有关的统计信息

本节会介绍如何让 Wireshark 基于抓包文件，生成与各种 IP 属性有关的统计信息。要让 Wireshark 生成与 IP 属性有关的统计信息，需借助于 **Statistics** 菜单名下的以下三个菜单项：

- IP Addresses;
- IP Destinations;
- IP Protocols Types.

4.8.1 准备工作

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再选择 **Statistics** 菜单。

4.8.2 操作方法

要让 Wireshark 基于抓包文件，生成有关 IP 地址的统计信息，请按以下步骤行事。

1. 点击 **Statistics** 菜单下的 **IP Addresses** 菜单项。
2. **IP Addresses Stats Tree** 窗口会立刻弹出，如图 4.25 所示。

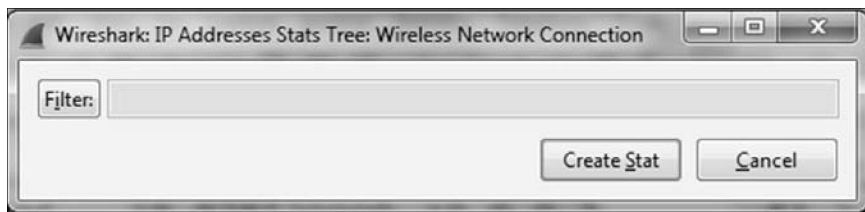


图4.25

3. 若直接点击 **Create Stat** 按钮，Wireshark 则会基于整个抓包文件，来生成有关 IP 地址的统计信息。
4. 可点击 **Filter** 按钮，构造显示过滤表达式，或在 **Filter** 输入栏内输入显示过滤表达式，再点 **Create Stat** 按钮，让 Wireshark 基于抓包文件中的部分数据，生成有关 IP 地址的统计信息。试举一例，如图 4.26 所示，先在 **Filter** 输入栏内输入显示过滤表达式

ip.addr==10.0.0.2，再点 **Create Stat** 按钮，Wireshark 所生成的统计信息将只涉及源自或发往这一 IP 地址的数据包。

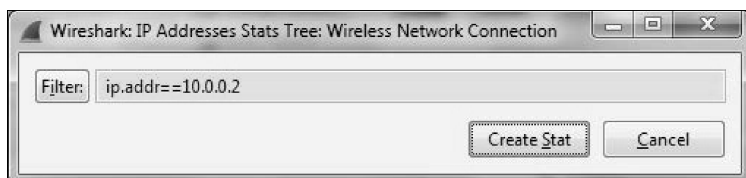


图4.26

5. 图 4.27 所示为在应用了图 4.26 输入的过滤表达式之后，Wireshark 生成的有关 IP 地址 10.0.0.2 的数据包统计信息。

Topic / Item	Count	Rate (ms)	Percent
IP Addresses	61710	0.024383	
10.0.0.2	61710	0.024383	100.00%
173.194.78.125	266	0.000105	0.43%
77.234.43.96	26	0.000010	0.04%
173.192.85.27	17	0.000007	0.03%
10.0.0.138	157	0.000062	0.25%
194.90.196.99	16	0.000006	0.03%
199.7.48.72	14	0.000006	0.02%
82.166.201.137	569	0.000225	0.92%
82.166.201.187	12	0.000005	0.02%

图4.27

要让 Wireshark 基于抓包文件，生成与指定的 IP 目的地址和 UDP/TCP 端口号有关的统计信息，请按以下步骤行事。

1. 点击 **Statistics** 菜单下的 **IP Destinations** 菜单项。
2. **IP Destinations Stats Tree** 窗口会立刻弹出，如图 4.28 所示。

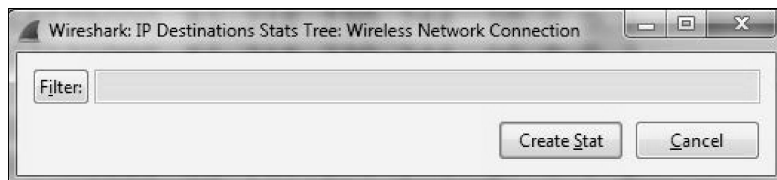


图4.28

3. 若直接点击 **Create Stat** 按钮，Wireshark 则会基于整个抓包文件，来生成所有 IP 地址的发包统计信息，其中会记录下相关 IP 地址（的主机）在发包时所使用的传输层协议（UDP 或 TCP）以及目的端口号。
4. 图 4.29 所示为 Wireshark 生成的 **IP Destinations** 统计信息示例。

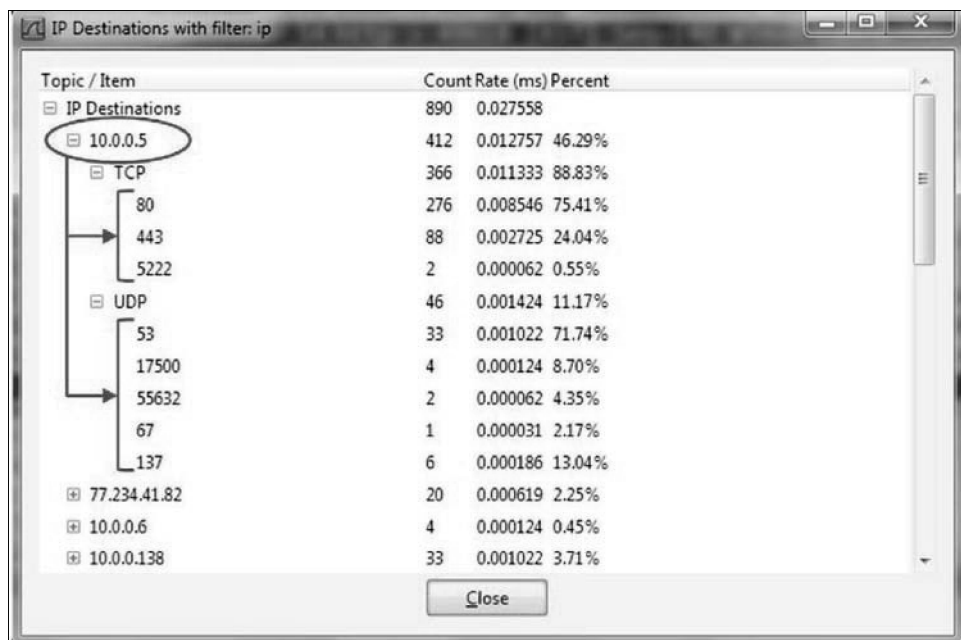


图4.29

5. 由图 4.29 所示的统计信息可知，IP 为 **10.0.0.5** 的主机既发出过目的端口号为 **80**、**443** 和 **5222** 的 TCP 数据包，也发出过目的端口号为 **53**、**67**、**137**、**17500**、**55632** 的 UDP 数据包。

通过 IP Destinations，还可以很方便地了解到网络中各主机向每个 TCP/UDP 目的端口发出的数据包的个数，因此它也是非常管用的排障工具。

要让 Wireshark 基于抓包文件，生成与 IP 协议类型有关的统计信息，请按以下步骤行事。

1. 点击 **Statistics** 菜单下的 **IP Protocol Types** 菜单项。
2. **IP Protocol Types Stats Tree** 窗口会立刻弹出，如图 4.30 所示。
3. 直接点击 **Create Stat** 按钮，Wireshark 会生成与 IP 协议类型有关的统计信息，如图 4.31 所示。

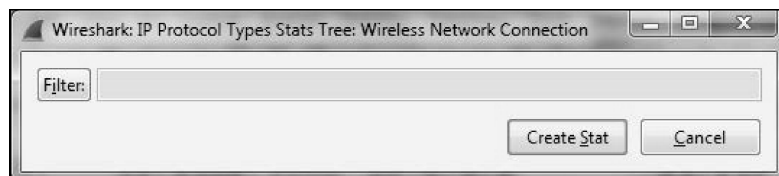


图4.30

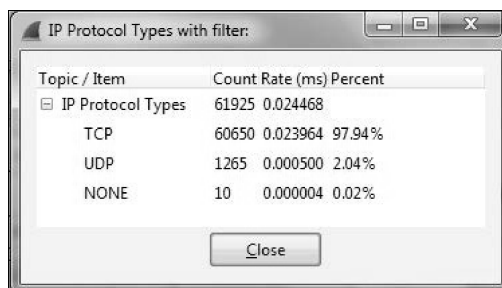


图4.31

4.8.3 幕后原理

不涉及任何幕后原理，只是让 Wireshark 基于抓包文件生成相应的统计信息而已。

4.8.4 拾遗补缺

Wireshark 内置了诸多统计信息生成工具（比如，**Summary** 工具、**Protocol Hierarchy** 工具、**Conversations** 工具和 **Endpoint** 工具等），可利用这些工具来生成各式各样的统计信息。排除网络故障时，由这些工具生成的统计信息或许能帮上网管人员的大忙。

第5章

高级信息统计工具的使用法

本章涵盖以下主题：

- ▶ 配置与显示过滤器结合使用的 IO Graph 工具，来定位与网络性能有关的问题；
- ▶ 用 IO Graphs 工具，来测量网络的吞吐量；
- ▶ IO Graphs 工具的高级配置方法（启用 Y 轴 unit 参数的 Advanced 选项）；
- ▶ TCP Stream Graphs 菜单项中 Time-Sequence(Stevens)子菜单项的用法；
- ▶ TCP Stream Graphs 菜单项中 Time-Sequence(tcp-trace)子菜单项的用法；
- ▶ TCP Stream Graphs 菜单项中 Throughput Graph 子菜单项的用法；
- ▶ TCP Stream Graphs 菜单项中 Round Trip Time 子菜单项的用法；
- ▶ TCP Stream Graphs 菜单项中 Window Scaling Graph 子菜单项的用法。

5.1 简介

上一章介绍了内置于 Wireshark 软件的基本信息统计工具的使用法，这些工具包括 Summary 工具、Protocol Hierarchy 工具、Conversations 工具和 Endpoint 工具等。本章会介绍 IO Graphs、TCP Stream Graphs 等高级信息统计工具，同时还会对 UDP multicast streams 等高级工具做简单介绍。

凭借本章所要介绍的 Wireshark 高级信息统计工具，可以更为细致入微地观察到网络中的风吹草动。Wireshark 高级信息统计工具主要有以下两种。

- IO Graphs 工具：借助于该工具，同时配搭预先定义的显示过滤器，便可以生成各种易于阅读的信息统计图表。比如，可生成单一 IP 主机吞吐量统计图表、两台或多台主机间流量负载统计图表、应用程序网络吞吐量统计图表，以及 TCP 现象分布统计图表等。
- TCP Stream Graphs 工具：利用该工具，便可深窥单条 TCP 连接（TCP 数据流）的内在。因此，该工具能帮助网管人员分析 TCP 故障，定位故障起因。

本章会介绍上述工具的使用方法，下一章会讨论如何利用这些工具来定位并解决网络故障。

5.2 配置与显示过滤器结合使用的IO Graphs工具，来定位与网络性能有关的问题

本节会介绍 IO Graphs 工具的使用方法，以及如何配置该工具来排除网络故障。

5.2.1 配置准备

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再点击 **Statistics** 菜单中的 **IO Graphs** 菜单项，会弹出 **IO Graphs** 窗口。若在抓包过程中如此行事，通过 **IO Graphs** 窗口显示的信息，便可获知网络的实时统计信息。

5.2.2 配置方法

Statistics 菜单中的 **IO Graphs** 菜单项一经点击，**IO Graphs** 窗口会立刻弹出，如图 5.1 所示。

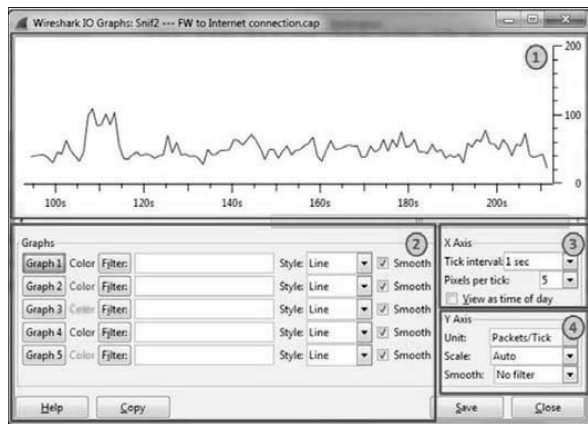


图5.1

在图 5.1 所示的 IO Graphs 窗口中，上半部分被标以数字 **1**，该区域为图形显示区域。IO Graphs 窗口下半部分的左侧被标以数字 **2**，该区域为显示过滤器配置区域。在区域 2 内，可针对区域 1 中的图形应用显示过滤器。IO Graphs 窗口下半部分的右侧上下两个区域分别被标以数字 **3** 和 **4**，这是 X 轴（横坐标）和 Y 轴（纵坐标）参数配置区域。对 3、4 两个区域内相关参数的调整，会在区域 1 中的 X 轴和 Y 轴上有所体现。现在，来看一下如何配置并调整 IO Graphs 窗口中 2、3、4 区域内的各项参数。

显示过滤器的配置

1. 在区域 2 中 Graph 按钮后的 Filter 输入栏内按照显示过滤器的语法输入过滤表达式。只有经过显示过滤器过滤的数据包才会体现在区域 1 中与 Capture 按钮相对应的图形里。该区域内有 5 个 Graph 按钮，最多可配置 5 个显示过滤器。
2. 在 Style 下拉菜单列表中圈定区域 1 里的图形以哪种风格示人，可选择的风格包括：**Line**（线）、**Impulse**（脉冲）、**Fbar**（粗线）或 **Dot**（点）。
3. 按下最左边的 **Graph** 按钮。只有如此，方能在区域 1 里激活与区域 2 中所填 Filter 和所选 Style 下拉菜单项配套的图形，请读者切莫遗忘。

X 轴（时间轴）参数的配置

X 轴参数全都在区域 3 中配置。

1. 在 **Tick interval**（计时单位）下拉菜单中指定一个计时单位：计时单位的取值范围为 0.001 秒~10 分钟。



注意

若把 X 轴的 Tick interval 参数值指定为 1 秒（1 sec），把 Y 轴区域的 Unit 参数值（速率单位）指定为 packets/tick，且区域 1 中的图形反映出的峰值为 1000，则意味着在抓包时间段内，Wireshark 测量出的流量传输峰值速率为 1000 个数据包/秒。可要是把 X 轴的 Tick interval 参数值更改为 0.1 秒（0.1 sec），那么区域 1 中的图形反映出的峰值状况势必有所不同，这是因为计时单位从 1 秒被调整为了 0.1 秒。

2. 在 **Pixels per tick**（时间刻度）下拉菜单中指定一个时间刻度：时间刻度的取值范围为 1~10。
3. 勾选 **View as time of day**（以一天当中的具体时刻来显示）复选框：一旦勾选，区域 1 中图形的 X 轴的时间格式将会按一天当中的具体时刻来显示；若未勾选，则区域 1 中图形的 X 轴的时间格式将会以抓包时长来显示。

Y 轴（速率轴）参数的配置

Y 轴参数全都在区域 4 中配置。

1. 在 **Unit**（速率单位）下拉菜单中选择一个菜单项，指定一个速率单位。可选菜单项包括 **Packets/Tick**、**Bytes/Tick**、**Bits/Tick** 或 **Advanced**。
2. 在 **Scale**（范围）下拉菜单中指定 Y 轴的范围（长度）：Y 轴长度有线性（**Linear**）和对数（**Logarithmic**）两种表现形式，可分别从下拉菜单中选择。当然，也可以沿用默认值 **Auto** 或根据需求从下拉菜单中指定一个值。
3. **Smooth**（平滑速率）参数：若要了解数据包的平均传输速率（亦即在每个计时单位内的平均传输速率），可在 **Smooth** 下拉菜单中选择除 **No filter** 以外的其他值，取值范围在 4~1024 之间。

5.2.3 幕后原理

IO Graphs 工具是 Wireshark 软件中最重要的工具之一，网管人员可藉此工具来在线监控网络性能，或对网络故障做离线分析。

使用 IO Graphs 工具时，如何配置与 X 轴和 Y 轴参数结合使用的显示过滤器是重中之重。

现以图 5.2 和图 5.3 为例，来讲解 IO Graphs 工具的使用法。网管人员要想了解（抓包文件中）IP 地址为 10.0.0.2 的内网主机访问 Internet 的情况，需为此配置以下两个显示过滤器。

- 在显示过滤器配置区域（即图 5.2 中的区域 2）的 **Graph 2** 按钮后的 Filter 输入栏内，输入显示过滤器 **ip.src==10.0.0.2**。配置该显示过滤器的目的，是要让 Wireshark “筛选”出源 IP 地址为 10.0.0.2 的所有流量，并针对这些流量在区域 1 中生成与 **Graph 2** 相对应的图形。“源 IP 地址为 10.0.0.2 的所有流量”亦即 IP 地址为 10.0.0.2 的内网主机访问 Internet 的流量（上行流量）。请注意，与 **Graph 2** 相对应的图形在区域 1 中将会以红色显示。
- 在显示过滤器配置区域（即图 5.2 中的区域 2）的 **Graph 3** 按钮后 Filter 输入栏内，输入显示过滤器 **ip.dst==10.0.0.2**。配置该显示过滤器的目的，是要让 Wireshark “筛选”出目的 IP 地址为 10.0.0.2 的所有流量，并针对这些流量在区域 1 中生成与 **Graph 3** 相对应的图形。可把“源 IP 地址为 10.0.0.2 的所有流量”视为 IP 地址为 10.0.0.2 的内网主机访问 Internet 的回程流量（下行或下载流量）。请注意，与 **Graph 2** 相对应的图形在区域 1 中将会以绿色示人。

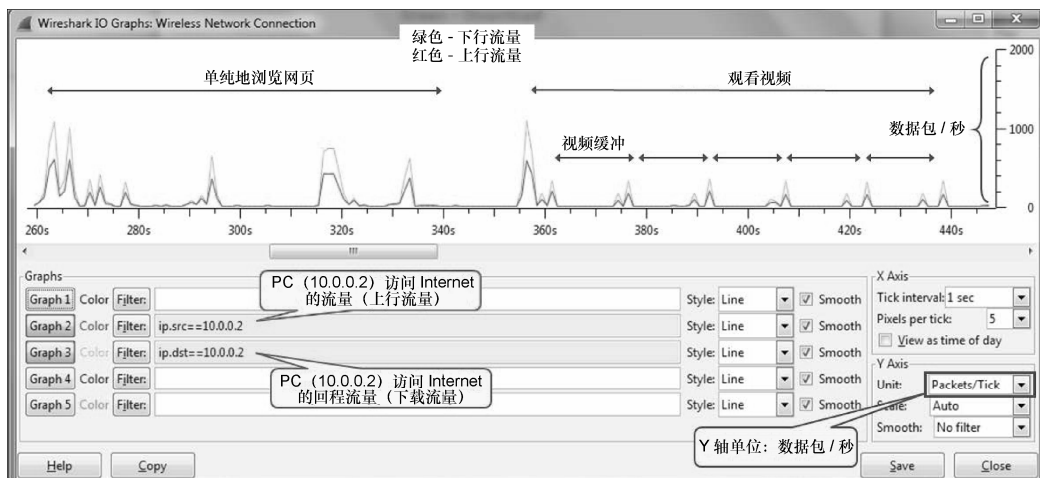


图5.2

由图 5.2 可知,使用 IO Graphs 工具测算 IP 地址为 10.0.0.2 的主机访问 Internet 的上、下行流量时,将 X 轴的 **Tick interval** (计时单位) 参数设成了 1 秒,将 Y 轴的 **Unit** (速率单位) 参数设成了 **Packets/tick** (数据包/秒)。从图 5.2 的图形显示区域可以看出,IP 地址为 10.0.0.2 的主机在正常浏览网页时所触发的上、下行流量的速率(在图形显示区域的左侧)和观看视频时所触发的上、下行流量的速率(在图形显示区域的右侧)都几乎相同^①。

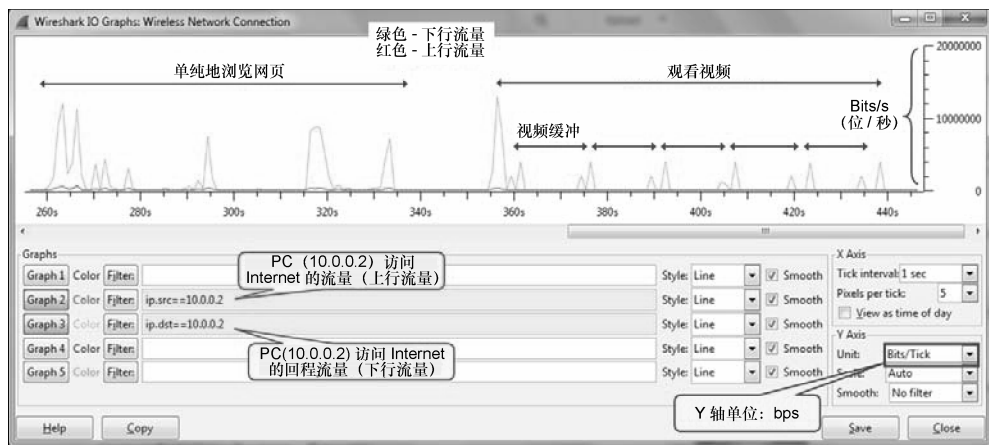


图5.3

图 5.3 所示为把 Y 轴的 **Unit** (速率单位) 参数更改为 **Bits/s** (位/秒) 后的显示情况。

^① 译者注: 请注意,图 5.2 中的测速单位是“数据包/秒”。通过图 5.2 得出的结论是,无论 IP 地址为 10.0.0.2 的主机是浏览网页还是观看视频,每秒生成的上、下行数据包的个数都大致相同。

从上图的图形显示区域可以看出，IP 地址为 10.0.0.2 的主机不论是正常浏览网页还是观看视频，所触发的下行流量的位速 (bit/s) 都要远远高于上行流量的位速（上行流量都是小包，下行流量都是大包）。

5.2.4 拾遗补缺

再举一例，这回来看看 IP 地址为 10.0.52.164 的主机通过 FTP 下载文件时生成的流量，如图 5.4 所示。跟前例一样，图 5.4 同样示出了 Y 轴的 **Unit** (速率单位) 参数分别为 **Packets/tick** 和 **Bits/Tick** 时，Wireshark 针对主机 10.0.52.164 生成的流量速率统计图形。**Packets/Tick** (在计时单位内传输的数据包的个数) 是一个非常重要的参数，在本书的 7~14 章会屡次提及。

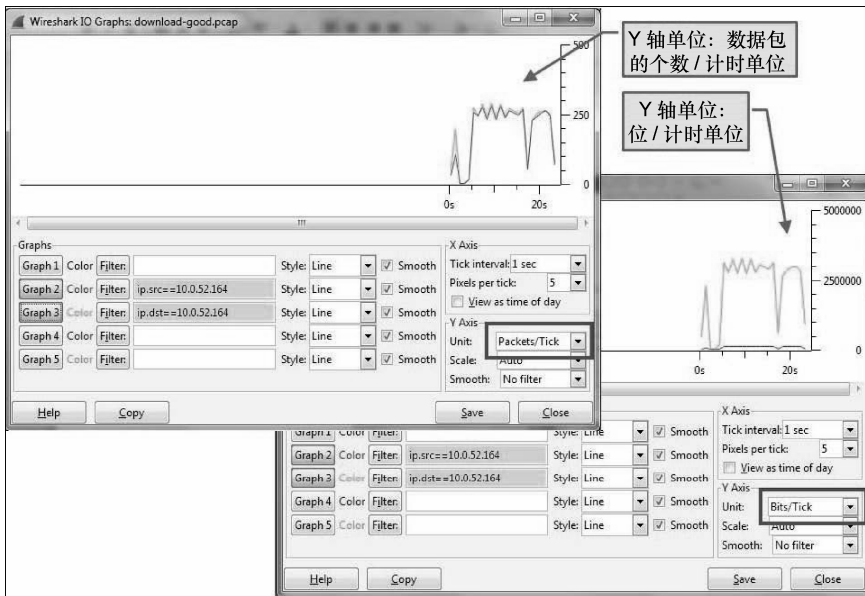


图5.4

5.3 用IO Graphs工具测算（链路的）吞吐量

IO Graphs 工具同样是测算网络吞吐量的一把利器。借助于该工具，同时再搭配预先设定的显示过滤器，便可以测算出各种流量的吞吐量。本节会举几个测算网络吞吐量的实例。

5.3.1 准备工作

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再选择 **Statistics** 菜单。

把 Wireshark 主机连接到已激活端口镜像功能的交换机端口（具体操作步骤详见第 1 章），应保证通过该交换机端口能接收到过往于有待监控的主机或服务器的所有流量。启动 Wireshark 软件，先选择相应的网卡开始抓包，再点击 **Statistics** 菜单中的 **IO Graphs** 菜单项。

所谓测算网络吞吐量，既可以指测算通信线路上两台设备（PC-服务器、IP 电话-IP 电话等）间相互通信的吞吐量，也可以指测算两台设备上运行的某一具体应用程序间相互通信的吞吐量（见图 5.5）^①。

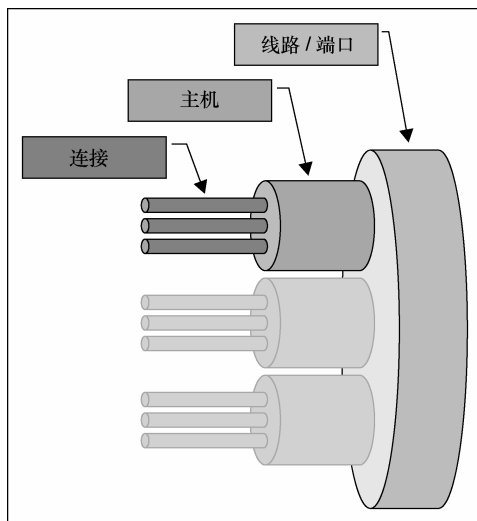


图5.5

测算链路上末端设备之间基于单条连接所发生的流量，并了解流量的来源，是定位网络故障的第一步^②。

常规的流量测算方法包括：测算主机到主机的流量、测算发往某指定服务器的所有流量、测算发往某指定服务器运行的某种应用程序的所有流量、统计发生在某指定服务器与 TCP 性能有关的所有现象。

5.3.2 测算方法

本节会给出若干测算网络流量时常用的显示过滤器^③。

① 译者注：原文是“While measuring the throughput, we can measure the throughput on a communication line between end devices (PC to server, phone to phone, PC to the Internet, and so on) or to a specific application”。

② 译者注：原文是“The process of isolating network problems starts from measuring traffic over a link between end devices on single connections and seeing where it comes from”。原文无厘头，译文直译。

③ 译者注：原文是“In this recipe, we will provide some basic filters for measuring traffic in the network”。作者的写作思路如天马行空般开阔，译文直译。

测算端设备之间的吞吐量

要测算两台端设备之间的流量，应配置显示过滤器，让 Wireshark 筛选出相应的流量。譬如，要测算发生于主机 10.2.10.101 和 10.2.10.240 之间的 IP 流量，显示过滤器应如此配置。

```
ip.addr eq 10.2.10.240 and ip.addr eq 10.2.10.240
```

既可以直接在 **IO Graphs** 窗口中输入上面这一过滤表达式，也可按以下步骤来构造显示过滤表达式。

1. 在抓包主窗口的数据包列表区域任选一个源 IP 地址为 10.2.10.101，目的 IP 地址为 10.2.10.240（或源 IP 地址为 10.2.10.240，目的 IP 地址为 10.2.10.101）的 IP 数据包。
2. 选中该 IP 包，单击右键，在右键菜单的 **Conversation filter** 菜单项中选择 IP 子菜单项。抓包主窗口顶部的 Filter 输入栏内将会自动生成显示过滤表达式 `ip.addr eq 10.2.10.240 and ip.addr eq 10.2.10.240`。
3. 将这一过滤表达式复制进 **IO Graphs** 窗口中 **Graph 2** 名下的 Filter 输入栏（Graph 1～5 均可输入，视所要生成的流量图中线条的颜色而定）。

现在，只要点击 **IO Graphs** 窗口中的 **Graph 2** 按钮，Wireshark 就会生成与 IP 主机 10.2.10.101 和 10.2.10.240 间的互访流量相对应的流量图了。

图 5.6 显示了上述操作过程。

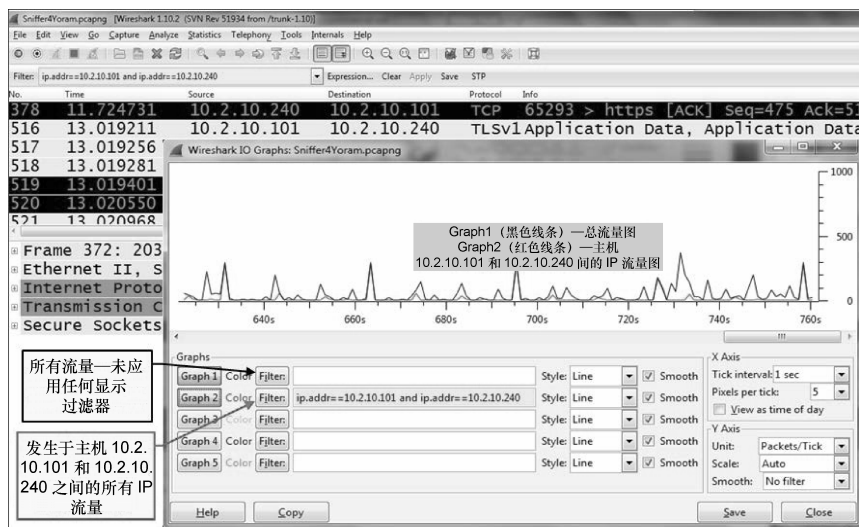


图5.6

测算应用程序的吞吐量

要想测量由某一种应用程序所触发的网络流量,从而达到评判其性能的目的,需先围绕(该应用程序所监听的) TCP/UDP 端口号,或访问该应用程序所触发的 TCP/UDP 连接,配置显示过滤器,筛选相关流量。

借助 **IO Graphs** 窗口,生成与某种应用程序挂钩的流量图的方法多种多样,下面介绍其中的一种。

1. 在抓包主窗口的数据包列表区域内,任选一个隶属于由该应用程序所触发的 UDP/TCP 数据包(即在交换于固定的 IP 地址+固定的 UDP/TCP 端口号之间的数据流中任选一个 UDP/TCP 数据包)。
2. 在选定的 UDP/TCP 数据包上单击右键,点击右键菜单中的 **Follow UDP stream** 或 **Follow TCP stream** 菜单项。这么一点,将导致 Wireshark 在其抓包主窗口顶部的 Filter 输入栏内自动生成一个显示过滤表达式,其格式为: `tcp.streameq<number>` 或 `udp.streameq<number>`, 其中 `number` 表示抓数据文件中的这股 TCP/UDP 数据流的编号。
3. 将这一显示过滤表达式复制进 **IO Graphs** 窗口中某一 Graph 名下的 Filter 输入栏。只要按下该 Graph 按钮,在 **IO Graphs** 窗口的图形显示区域,便会显示出 Wireshark 针对某条 TCP 或 UDP 数据流生成的图形了。

图 5.7 显示了上述操作过程。

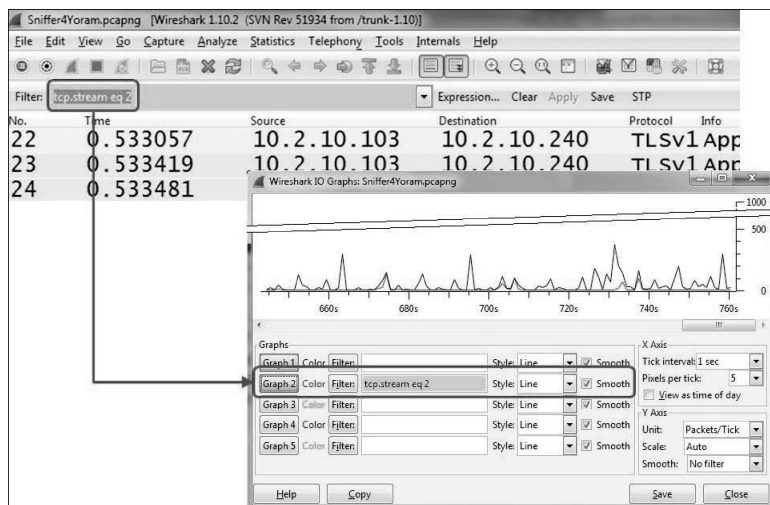


图 5.7

若要让 Wireshark 通过 **IO Graphs** 窗口中的图形来反映某股 TCP/UDP 数据流在传输过程中发生的变故，则需要给基于 TCP/UDP 数据流的显示过滤器添加更多的条件，请看下面这两个例子。

- 要通过 **IO Graphs** 窗口中的图形来反映发生在某股 TCP 数据流中的 TCP 重传现象（TCP 重传现象预示着发生了丢包、网络“卡”或其他与网络性能有关的问题），显示过滤器应如此配置：`tcp.streameq 2 and tcp.analysis.retransmissions`。
- 要通过 **IO Graphs** 窗口中的图形来反映发生在某股 TCP 数据流中的 TCP 零窗口（TCP zero window）现象（TCP 零窗口现象往往意味着 TCP 接收方或发送方反应迟钝），显示过滤器应如此配置：`tcp.streameq 2 and tcp.analysis.zero_window`。

5.3.3 幕后原理

IO Graphs 工具的强大之处体现在，可把由各种显示过滤器筛选出的流量，以各种图形的方式加以展现。在显示过滤器的帮助下，无论是哪一种数据包，无论其具有哪一种特征，都能通过 **IO Graphs** 工具以各种图形的方式加以展示。

5.3.4 拾遗补缺

本节会再举几个使用显示过滤器，来筛选具有某种特征的数据包的示例。

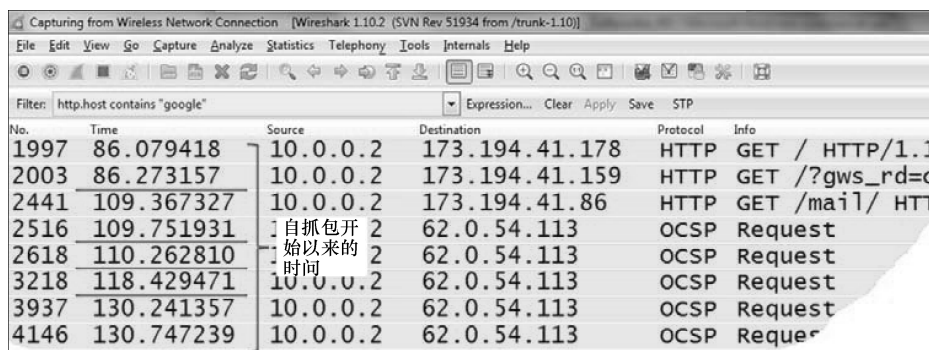
筛选由特定用户发出的 SMS 消息

其实，就是要筛选出包含 `Submit_SM` 命令的 **SMPP**（**Short Message Peer to Peer**）数据包。`Submit_SM` 命令是发送 SMS 的 **SMPP** 命令。要筛选这种数据包，显示过滤表达式应如此配置：`smpp.destination_addr == "phone number"`。

通过 IO Graphs 窗口中的图形，来反映对 Google Web 页面的访问量

1. 启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），再点击 **Statistics** 菜单中的 **IO Graphs** 菜单项，这将导致 **IO Graphs** 窗口的弹出。
2. 在 **IO Graphs** 窗口中 **Graph 1** 名下的 Filter 输入栏内输入显示过滤表达式 `http.host contains "google"`，再点击该 **Graph 1** 按钮，便激活了图形显示区域里与 **Graph 1** 相对应的图形。
3. 图 5.8 所示为在 Wireshark 抓包主窗口中输入显示过滤表达式 `http.host contains "google"` 后的情况。

4. 图 5.9 所示为 Wireshark 通过 **IO Graphs** 窗口，以图形方式展示的对 Google Web 页面的访问量。



No.	Time	Source	Destination	Protocol	Info
1997	86.079418	10.0.0.2	173.194.41.178	HTTP	GET / HTTP/1.1
2003	86.273157	10.0.0.2	173.194.41.159	HTTP	GET /?gws_rd=c
2441	109.367327	10.0.0.2	173.194.41.86	HTTP	GET /mail/ HT
2516	109.751931	10.0.0.2	62.0.54.113	OCSP	Request
2618	110.262810	10.0.0.2	62.0.54.113	OCSP	Request
3218	118.429471	10.0.0.2	62.0.54.113	OCSP	Request
3937	130.241357	10.0.0.2	62.0.54.113	OCSP	Request
4146	130.747239	10.0.0.2	62.0.54.113	OCSP	Request

图5.8

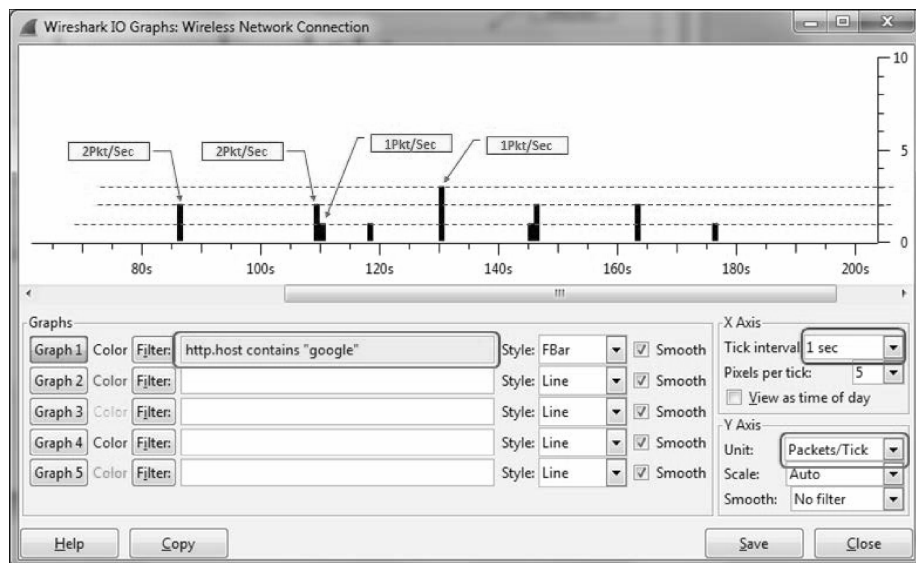


图5.9

由图 5.9 可知，自开始抓包开始后的第 86 秒，有人 2 次访问了 Google Web 页面；大约在第 109 秒，又有人访问了 2 次。

5.4 IO Graphs工具的高级配置方法（启用Y轴Unit参数的Advanced选项）

前几节的内容都是在介绍 **IO Graphs** 工具的常规用法，在由该工具生成的与网络性能有

关的图形中，采用的 Y 轴 Unit（速率单位）参数不是 Packets/Tick（数据包/计时单位），就是 Bytes/Tick（字节/计时单位），或是 Bits/Tick（位/计时单位）。不过，还有一些网络性能指标在使用图形方式表示时，是无法使用以上三种速率单位的，这也正是 Y 轴 Unit（速率单位）下拉菜单中还有 **Advanced** 选项的原因所在。

5.4.1 配置准备

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），点击 **Statistics** 菜单中的 **IO Graphs** 菜单项。在弹出的 **IO Graphs** 窗口中，选择 Y 轴区域内 **Unit** 下拉菜单中的 **Advanced** 选项。该选项一经点选，**IO Graphs** 窗口将陡然增大一倍。在该窗口的过滤器配置区域中，多出了 **Clac** 下拉菜单，其名下有若干菜单项可供选择，下面将介绍这些菜单项的作用。

- **SUM (*)**: 只要在其右边的输入栏中填入适当的条件（比如，显示过滤参数 `ip.len`），再点下 **Graph** 按钮，Wireshark 就会统计在每个计时单位内（具体的计时单位通过选择 X 轴参数配置区域内的 **Tick interval** 下拉菜单项来指定），实际传输的 IP 数据包的总字节数（即累加相关 IP 数据包包头的总长度字段值，可通过在 **Filter** 输入栏内填入显示过滤表达式来指明具体计算哪一类 IP 数据包的总字节数），并生成相关图形^①。
- **COUNT FRAMES (*)**: 只要在其右边的输入栏中填入适当的条件，再点下 **Graph** 按钮，Wireshark 就会统计在每个计时单位内，发生的匹配该条件的现象，并生成相关图形。
- **COUNT FIELDS (*)**: 只要在其右边的输入栏中填入适当的条件（比如，某种协议头部中的字段名），再点下 **Graph** 按钮，Wireshark 就会统计出在每个计时单位内，所传数据包中该字段出现的次数，并生成相关图形。
- **MAX (*)**: 只要在其右边的输入栏中填入适当的参数，再点下 **Graph** 按钮，Wireshark 就会统计出在每个计时单位内，所传数据包中相关参数的最高值。
- **MIN (*)**: 只要在其右边的输入栏中填入适当的参数，再点下 **Graph** 按钮，Wireshark 就会统计出在每个计时单位内，所传数据包中相关参数的最低值。
- **AVG (*)**: 只要在其右边的输入栏中填入适当的参数，再点下 **Graph** 按钮，Wireshark 就会统计出在每个计时单位内，所传数据包中相关参数的平均值。

^① 译者注：原文只有一句 “This draws a graph with the summary of a parameter in the tick interval”，译者不才，一句英文要用这么多汉字来译，要是读者嫌烦，请视而不见。

➤ **LOAD (*)**: 用来生成与响应时间有关的图形。

5.4.2 配置方法

要启用 **IO Graphs** 工具的高级功能, 请按以下步骤行事。

启动 **Wireshark** 软件, 先打开一个抓包文件 (或选择一块网卡, 点击 **Start** 按钮开始抓包), 点击 **Statistics** 菜单中的 **IO Graphs** 菜单项。

1. 在弹出的 **IO Graphs** 窗口中, 选择 Y 轴区域内 **Unit** 下拉菜单中的 **Advanced** 菜单项, 如图 5.10 所示。

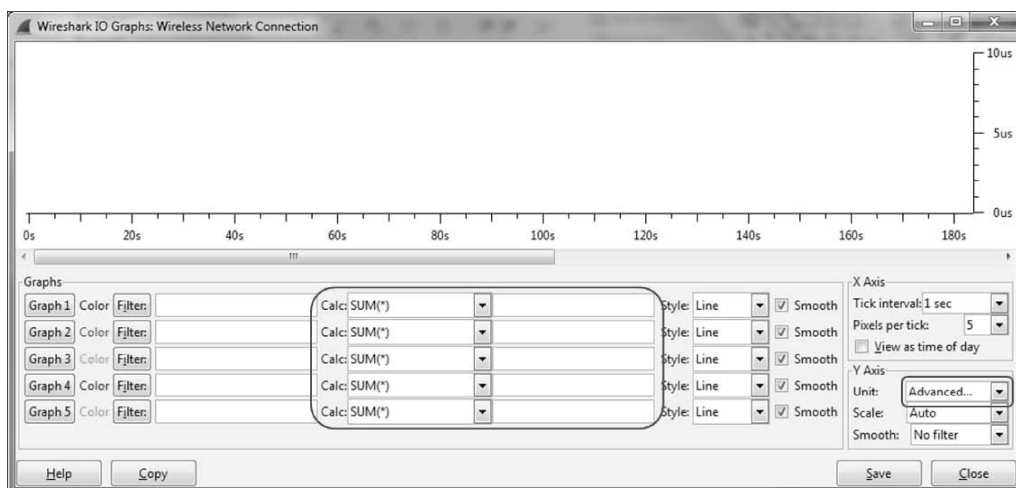


图5.10

2. 在 **IO Graphs** 窗口的过滤器配置区域中, 多出了 **Clac** 下拉菜单, 其名下有若干菜单项可供选择 (在图 5.10 中已经圈出)。
3. 可供选择的下拉菜单项包括: **SUM(*)/COUNT FRAMES (*)/COUNT FIELDS (*)/MAX(*)/MIN(*)/AVG(*)/LOAD(*)**。在下拉菜单后面, 还有一个输入栏, 只有在其中填入了适当的条件, 与其对应的 Graph 方能在 **IO Graphs** 窗口的图形显示区域露面。

接下来, 将以举例的方式来原因 **IO Graphs** 工具的高级配置方法。

如何生成数据帧接收 (抓取) 时间间隔统计图形

排除网络故障时, 通过观察 **Wireshark** 抓包文件中相关数据帧 (封装 TCP 报文段的数据帧) 的抓取时间间隔, 通常应能判断出是否存在与 TCP 性能有关的问题。让 **Wireshark** 生成 (封装了 TCP 报文段的) 数据帧抓取时间间隔统计图形, 无疑是一种比较直观的了解

TCP 性能的方法。

请看图 5.11 所示的抓包文件。

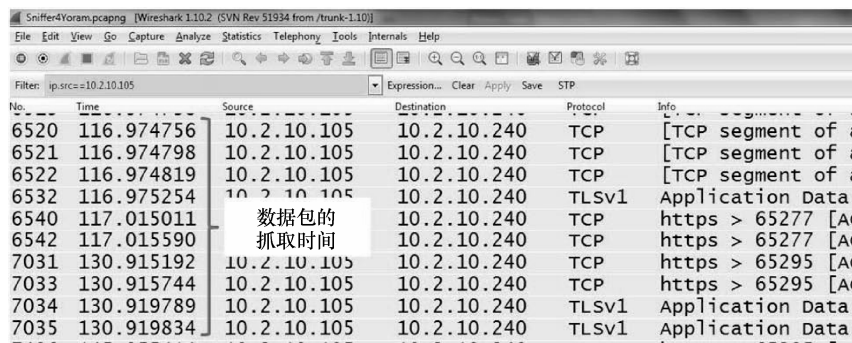


图5.11

拜赐图 5.11 中 Filter 输入栏里的显示过滤语句 `ip.src ==10.2.10.105` 所赐，数据包列表区域里所呈现的所有数据包的源 IP 地址必然是 10.2.10.105。

现在，将沿用上例，来看看如何让 Wireshark 以各种直观的图形方式来显示相关数据帧（封装了源 IP 地址为 10.2.10.105 的 IP 包的数据帧）的抓取时间间隔。

- 在 **IO Graphs** 窗口的 **Graph 1** 按钮右边的 Filter 输入栏内填入显示过滤语句 `ip.src ==10.2.10.105`，然后选择 **Clac** 下列菜单栏里的 **MAX(*)** 菜单项，同时在其右边的输入栏内输入 `frame.time_delta`，最后按下 **Graph 1** 按钮^①。上述操作会使得 Wireshark 在图形显示区域以 Impluse（脉冲线）方式，显示以 1 秒为单位的时间间隔内，抓取源 IP 地址为 10.2.10.105 的 IP 数据包（帧）的最长时间间隔图。
- 在 **IO Graphs** 窗口的 **Graph 2** 按钮右边的 Filter 输入栏内填入显示过滤语句 `ip.src ==10.2.10.105`，然后选择 **Clac** 下列菜单栏里的 **AVG(*)** 菜单项，同时在其右边的输入栏内输入 `frame.time_delta`，最后按下 **Graph 2** 按钮^②。上述操作会使得 Wireshark 在图形显示区域以 Dot（圆点）方式，显示以 1 秒为单位的时间间隔内，抓取源 IP 地址为 10.2.10.105 的 IP 数据包（帧）的平均时间间隔图。
- 在 **IO Graphs** 窗口的 **Graph 3** 按钮右边的 Filter 输入栏内填入显示过滤语句 `ip.src ==10.2.10.105`，然后选择 **Clac** 下列菜单栏里的 **MIN(*)** 菜单项，同时在其右边的输入栏内输入 `frame.time_delta`，最后按下 **Graph 3** 按钮^③。上述操作会使得

① 译者注：还应选择 Style 下拉菜单中的 Impulse 菜单项，作者说漏了。

② 译者注：还应选择 Style 下拉菜单中的 Dot 菜单项，作者说漏了。

③ 译者注：还应选择 Style 下拉菜单中的 Dot 菜单项，作者又说漏了。

Wireshark 在图形显示区域以 Dot 方式，显示以 1 秒为单位的时间间隔内，抓取源 IP 地址为 10.2.10.105 的 IP 数据包（帧）的最短时间间隔图。

图 5.12 中的 Wireshark IO Graphs 窗口完整显示了以上操作步骤，以及与 Graph 1、Graph 2、Graph 3 相对应的图形。

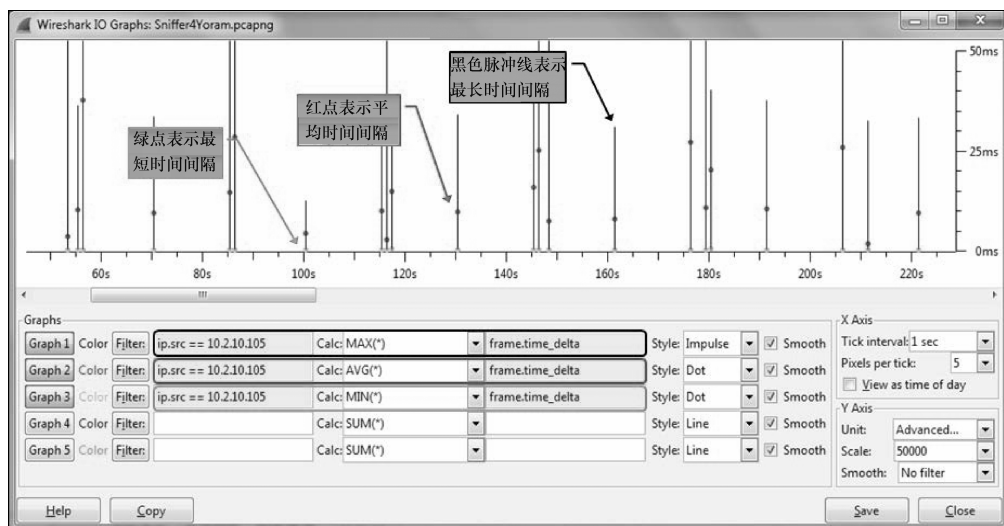


图5.12

图 5.12 中由 Wireshark 生成的在单位时间内抓取数据包（帧）的最长、最短以及平均间隔时间图到底有什么作用呢？能在排查网络故障时助我们一臂之力吗？第 10 章会给出答案。

如何获悉某股 TCP 数据流中发生了多少次 TCP 重传事件

与 TCP 有关的事件多种多样，比如 TCP 重传事件、滑动窗口事件、ACK（或等不来 ACK）等^①。要了解在某个时间段内发生过多少次与 TCP 有关的事件，还得动动 **IO Graphs** 窗口中 **Calc** 下列菜单栏里 **Count (*)** 菜单项功能的脑筋，具体操作步骤如下所列。

1. 启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），点击 **Statistics** 菜单中的 **IO Graphs** 菜单项。
2. 在弹出的 **IO Graphs** 窗口中，选择 Y 轴区域内 **Unit** 下拉菜单中的 **Advanced** 菜单项。
3. 在变大的 **IO Graphs** 窗口中，执行下列操作。
 - 以建立 TCP 会话的两台主机的 IP 地址为参数，定义显示过滤器，比如，

^① 译者注：原文是“TCP events can be of many types: retransmissions, sliding window events, ACKs (or lack of them), and others”。

`ip.addr == 192.1.1.2 and ip.addr == 192.1.1.21`。在 **Graph** 按钮右边的 Filter 输入栏内填入这一过滤表达式。

- 在 **Clac** 下拉菜单栏里选择 **COUNT FRAMES(*)** 菜单项, 然后在其右边的输入栏里填入表示 TCP 事件的“条件”。对于本例, 是要获悉发生了多少次 TCP 重传事件, 故而应填入 `tcp.analysis.retransmission`。
- 点击 **Graph** 按钮, 激活图形显示区域中对应于该按钮, 与特定 TCP 事件有关的图形。

在图 5.13 所示 Wireshark IO Graphs 窗口的图形显示区域, 可以很直观地看见整个抓包过程中, 在三股不同的 TCP 数据流中分别发生过多少次 TCP 重传事件。

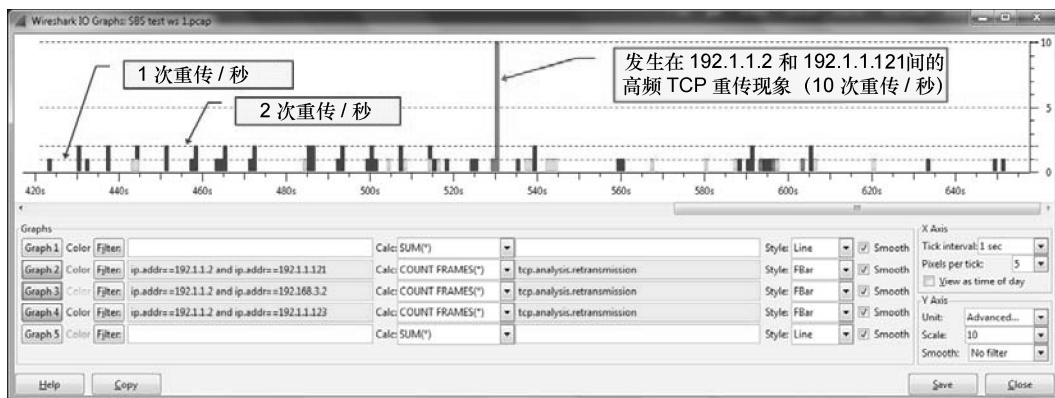


图 5.13

通过图 5.13 还可以看出, 在每一股 TCP 数据流 (每一对 IP 端点间的 TCP 数据流) 中发生的 TCP 重传事件, 都分别用了不同的颜色来表示。

如何获悉某股 TCP 数据流中数据包的抖动情况^①

网络中发生的剧烈抖动现象, 会显著影响多种网络协议的数据传输, 对 TCP 协议流量的传输影响最大(抖动是指构成同一条会话或同股数据流的各个数据包之间存在的延迟差异)。借助于 IO Graphs 工具生成的相关图形, 可很好地了解某一条连接 (TCP 会话或 TCP 数据流) 中的数据包间的延迟变化情况。当然, 在使用 IO Graphs 工具时, 同样要点选 Y 轴区域内 **Unit** 下拉菜单中的 **Advanced** 菜单项, 具体操作步骤如下所列。

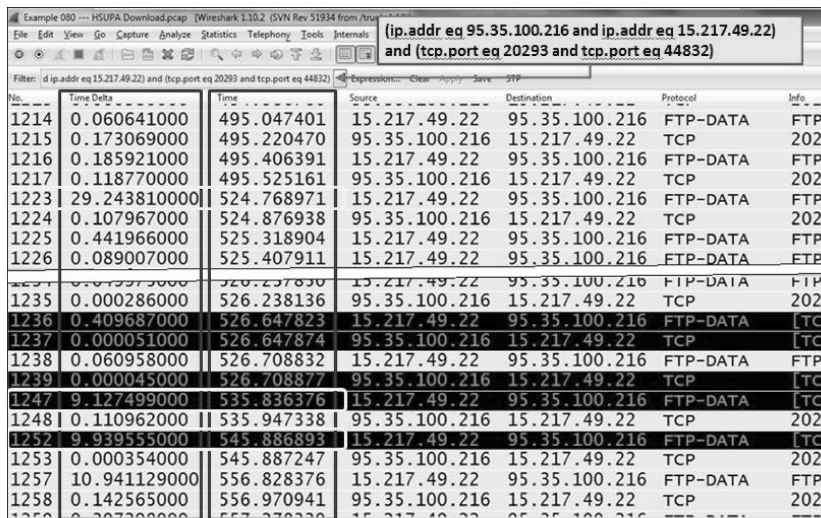
- 启动 Wireshark 软件, 先打开一个抓包文件 (或选择一块网卡, 点击 **Start** 按钮开始抓包), 在抓包主窗口中任选一个隶属于某股有待观察的 TCP 数据流的 TCP 数据包。然

^① 译者注: 原文标题是 “How to monitor a number of field appearances”, 作者写作时心不在焉, 译文酌改。

后点击右键，在右键菜单中选择 **Conversation filter** 菜单项中的 **TCP** 子菜单项。抓包主窗口上部的 **Filter** 输入栏内会自动生成一个显示过滤器。

2. 点击 **Statistics** 菜单中的 **IO Graphs** 菜单项。
3. 在弹出的 **IO Graphs** 窗口中，选择 Y 轴区域内 **Unit** 下拉菜单中的 **Advanced** 菜单项，**IO Graphs** 窗口将会陡然增大一倍。
4. 在 **IO Graphs** 窗口中显示过滤器配置区域内，按以下方法填写过滤表达式并提供相关条件。
 - 将抓包主窗口中 **Filter** 输入栏内的显示过滤表达式复制进 **Graph** 按钮右边的 **Filter** 输入栏。
 - 在 **Clac** 下拉菜单栏右边的输入栏内输入 `frame.time_delta`。
 - 在 **Clac** 下拉菜单栏里选择 **AVG(*)**、**MAX(*)** 或 **MIN(*)** 下拉菜单项。
5. 选择 **IO Graphs** 窗口中 X 轴配置区域中的参数，点击 **Graph** 按钮，在图形显示区域里激活与此按钮相对应的图形。

试举一例，请看图 5.14。在图中的抓包主窗口中，除了有一个 **Time** 数据包属性列（用来表示 Wireshark 所抓数据包的实时时间）之外，该属性列的左边还多出了一个 **Time delta**（时间差异）数据包属性列，可藉此列来获悉 Wireshark 所抓连续两个数据包之间的时差。



No.	Time Delta	Time	Source	Destination	Protocol	Info
1214	0.060641000	495.047401	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1215	0.173069000	495.220470	95.35.100.216	15.217.49.22	TCP	202
1216	0.185921000	495.406391	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1217	0.118770000	495.525161	95.35.100.216	15.217.49.22	TCP	202
1223	29.243810000	524.768971	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1224	0.107967000	524.876938	95.35.100.216	15.217.49.22	TCP	202
1225	0.441966000	525.318904	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1226	0.089007000	525.407911	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1235	0.000286000	526.238136	95.35.100.216	15.217.49.22	TCP	202
1236	0.409687000	526.647823	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1237	0.000051000	526.647874	95.35.100.216	15.217.49.22	TCP	202
1238	0.060958000	526.708832	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1239	0.000045000	526.708877	95.35.100.216	15.217.49.22	TCP	202
1247	9.127499000	535.836376	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1248	0.110962000	535.947338	95.35.100.216	15.217.49.22	TCP	202
1252	9.939555000	545.886893	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1253	0.000354000	545.887247	95.35.100.216	15.217.49.22	TCP	202
1257	10.941129000	556.828376	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1258	0.142565000	556.970941	95.35.100.216	15.217.49.22	TCP	202

图5.14

由图 5.14 可知，在显示抓包过滤器筛选出的这一股 TCP 数据流中，数据包之间的延迟差异极为明显。譬如，Wireshark 在抓到编号为 1217 和 1239 的数据包之后，分别等了 29.24 秒和 9.12 秒，才等来了同一 TCP 数据流的下一个数据包（编号分别为 1223 和 1247），类似的情况多不胜数。

图 5.15 所示为按上述步骤配置 IO Graphs 工具之后，Wireshark 生成的更为直观的 TCP 数据流的抖动图。

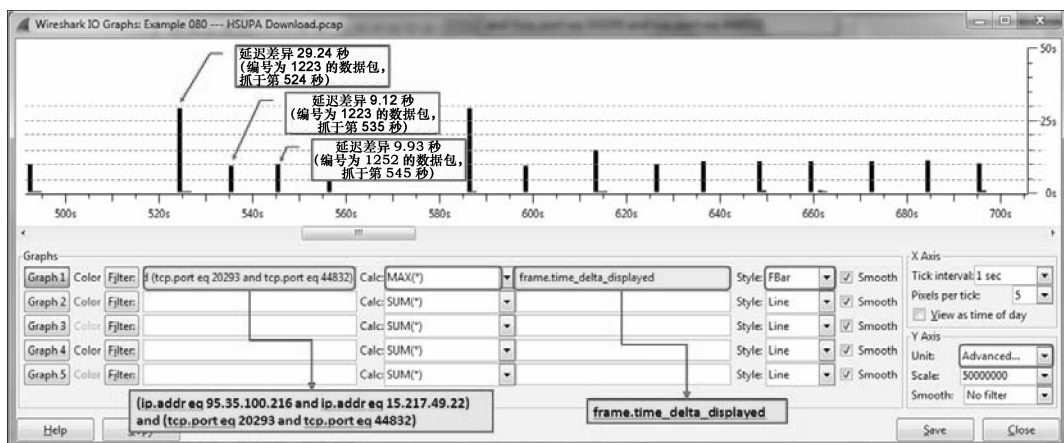


图5.15

同一股数据流中数据包间的延迟差异在图 5.15 中可谓是一目了然，随后几章会说明导致该问题的原因，并给出解决办法。

5.4.3 幕后原理

IO Graphs 工具是 Wireshark 软件所奉献的最高效也最强大工具之一。“标准”的 IO Graphs 工具可用来生成与网络性能有关的基本信息统计图，一旦启用了其 Y 轴区域内 Unit 下拉菜单中的 **Advanced** 菜单项，便可以让 Wireshark 生成各种更为直观的图形（比如生成与单股或多股 TCP 数据流有关的高级信息统计图），对网络或应用程序的运行情况做深度分析。

在 IO Graphs 工具的显示过滤器配置区域内，有一个 Filter 输入栏，只要输入正确的显示过滤表达式，便可以让其基于某一对主机、某台服务器或某条（TCP）连接来生成相关统计图形。要想获知流量的细节，就得启用 IO Graphs 工具的高级功能，下面举两个例子。

- 观察某股 TCP 数据流中各数据包间的延迟差异。
- 观察某股 RTP/视频流中含标记位的数据包。

5.4.4 拾遗补缺

可在抓包时不停地点击 **Statistics** 菜单中的 **IO Graphs** 菜单项, 根据实际需求, 让 **IO Graphs** 窗口实时显示与网络流量有关的信息统计图形^①。

5.5 TCP StreamGraph菜单项中Time-Sequence (Stevens)子菜单项的用法

TCP StreamGraph 工具是 **Wireshark** 提供的用来深度分析应用程序行为的工具之一。本节会介绍如何使用该工具来洞察应用程序的举动, 并定位相关故障原因。

5.5.1 使用准备

启动 **Wireshark** 软件, 打开一个抓包文件, 或选择一块网卡, 点击 **Start** 按钮开始抓包。虽然可在抓包的同时使用 **TCP StreamGraph** 菜单项中的 **Time-Sequence(Stevens)** 工具, 但该工具并不会在线实时生成统计信息, 故建议在使用其之前, 先停止抓包。

5.5.2 使用方法

要观看由 **TCP StreamGraph** 菜单项下 **Time-Sequence(Stevens)** 工具生成的统计信息, 请按以下步骤行事。

1. 在抓包主窗口的数据包列表区域内, 选中一个隶属于有待监控的某股 **TCP** 数据流的数据包。



注意

在数据包列表区域内, 隶属于某股 **TCP** 流的数据包, 必然是有来有往。因此, 在该区域内选择数据包, 让 **TCP StreamGraph** 工具生成统计信息时, 要看清方向 (要关注数据包的源和目的 **IP** 地址)。比方说, 要让 **TCP StreamGraph** 工具生成与 **HTTP** 下载流量有关的统计信息, 就应该选择下行方向的数据包 (即数据包的源 **IP** 地址为公网地址 [源端口号一般为 80], 目的 **IP** 地址为内网地址)。

2. 选择 **Statistics** 菜单下的 **TCP StreamGraph** 菜单项, 点击其名下的 **Time-Sequence Graph (Stevens)** 子菜单项。**TCP Graph** 窗口会立刻弹出, 如图 5.16 所示。

Time-Sequence Graph (Stevens) 子菜单项所生成的图形实际上反映的是, 在单位时间内, 受监控的 **TCP** 流在某个方向所传数据的字节数。出现在图 5.16 中的是一条陡然增高的实线, 这

^① 译者注: 原文是 “You can always click on **IO Graph**, and it will bring you to the reference packet in the packet pane”。

反映出了一次常规的文件下载行为。

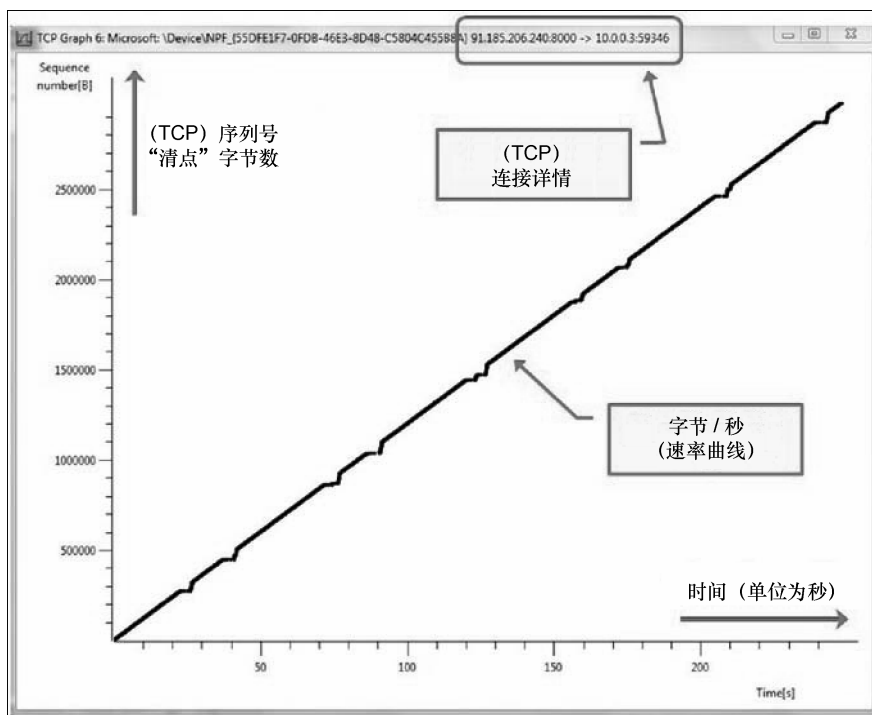


图5.16

要想测算这次文件下载所占带宽，只需计算单位时间内所传输的数据包的总字节数，请看图 5.17。

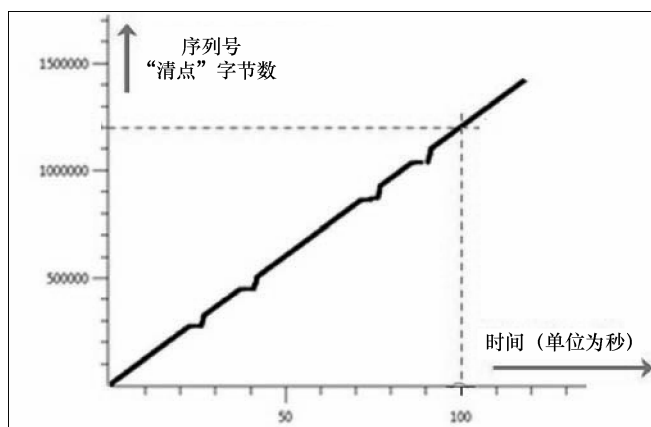


图5.17

由图 5.17 可知，对于本次文件下载，在 100 秒内发生了 1,200,000 字节的流量。于是，可以计算出本次文件下载所占带宽为 12KB/s 或 96 kbit/s。

3. 要想放大 TCP Graph 窗口内某个点附近的区域，只需用鼠标圈定该点所在区域；要想缩小经过放大的区域，请按“减号”键。
4. 用鼠标左键单击图形中的一个点，便可在抓包主窗口的数据包列表区域定位到与其相对应的数据包^①。
5. 要想调整 TCP Graph 窗口的相关显示参数，可在一个与其一起弹出的小窗口内进行，如图 5.18 所示。

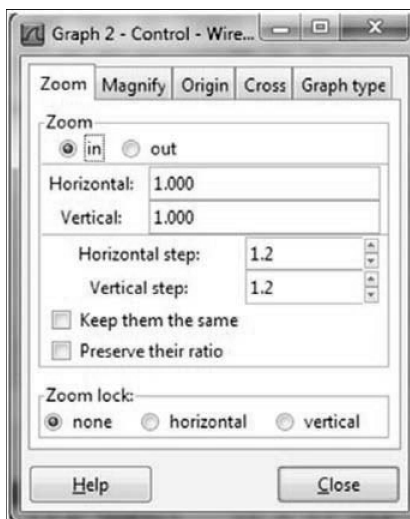


图5.18

5.5.3 幕后原理

要想针对某股 TCP 数据流，统计（单位时间内）在某个指定方向上传输的数据包的字节数，只需清点一下相关数据包的 TCP 头部中的序列号字段值。**TCP StreamGraph** 菜单项下的 **Time-Sequence Graph (Stevens)** 工具行使的就是这个功能，只是以图形的方式加以呈现。不过，这样统计出来的字节数实际上只包括了在某个方向上传输的（经由 TCP 头部封装的）应用程序数据（包含应用程序头部，但不含 IP 包头及 TCP 头部）。

通过 **TCP StreamGraph** 菜单项下的 **Time-Sequence Graph (Stevens)** 工具所生成的图形，

^① 译者注：原文是“Right-clicking on a point in the graph will take us to the packet pane in the captured file”。译者试过，鼠标右键不灵，只能用左键，译文酌改。

对分析（基于 TCP 的）应用程序的举动大有裨益（对此，将后文再表）。比方说，一条连绵不断的斜线就预示着正常的文件传输，而斜线要是时断时续，则表示文件传输存在问题；斜线的角度越大，就表示文件的传输速率极高，反之，则表示文件传输缓慢（当然，还得视 X、Y 轴的刻度而定）。

5.5.4 拾遗补缺

用鼠标左键单击图形中的一个点，即可在抓包主窗口的数据包列表区域定位到与其相对应的数据包。也就是说，可直接单击图形中有疑问的点，然后便可以直接返回抓包主窗口，找到有疑问的数据包了。

在观看图形时，一定要清楚此图是针对哪一种应用程序生成的。用 **TCP StreamGraph** 菜单项中 **Time-Sequence Graph (Stevens)** 工具生成的相同的图形，对 A 应用程序来说可能是正常的，但对 B 应用程序来说则未必。

5.6 TCP StreamGraph菜单项中Time-Sequence (tcp-trace)子菜单项的用法

TCP StreamGraph 菜单项中的 **Time-Sequence(tcp-trace)** 子菜单项功能脱胎于 UNIX **tcpdump** 工具，可提供有待监控的 TCP 连接的诸多详细信息。这些信息可用来分析与此 TCP 连接有关的种种问题，包括 TCP 确认、TCP 重传，以及 TCP 窗口大小等信息。

5.6.1 使用准备

启动 Wireshark 软件，打开一个抓包文件，或选择一块网卡，点击 **Start** 按钮开始抓包。虽然可在抓包的同时使用 **TCP StreamGraph** 菜单项中的 **Time-Sequence(tcp-trace)** 工具，但该工具并不会在线实时生成统计信息，故建议在使用其之前，先停止抓包。

5.6.2 使用方法

要观看由 **TCP StreamGraph** 菜单项下 **Time-Sequence(tcp-trace)** 工具生成的统计信息，请按以下步骤行事。

1. 在抓包主窗口的数据包列表区域内，选中一个隶属于有待监控的某股 TCP 数据流的数据包。
2. 选择 **Statistics** 菜单下的 **TCP StreamGraph** 菜单项，点击其名下的 **Time-Sequence**

(tcp-trace)子菜单项。TCP Graph 窗口会立刻弹出，如图 5.19 所示。



注意

在数据包列表区域内，隶属于某股 TCP 流的数据包，必然是有来有往。因此，在选择数据包，让 TCP StreamGraph 工具生成统计信息时，要看清方向（要关注数据包的源和目的 IP 地址）。比方说，要让 TCP StreamGraph 工具生成与 HTTP 下载流量有关的统计信息，就应该选择下行方向的数据包（即数据包的源 IP 地址为公网地址[源端口号一般为 80]，目的 IP 地址为内网地址）。

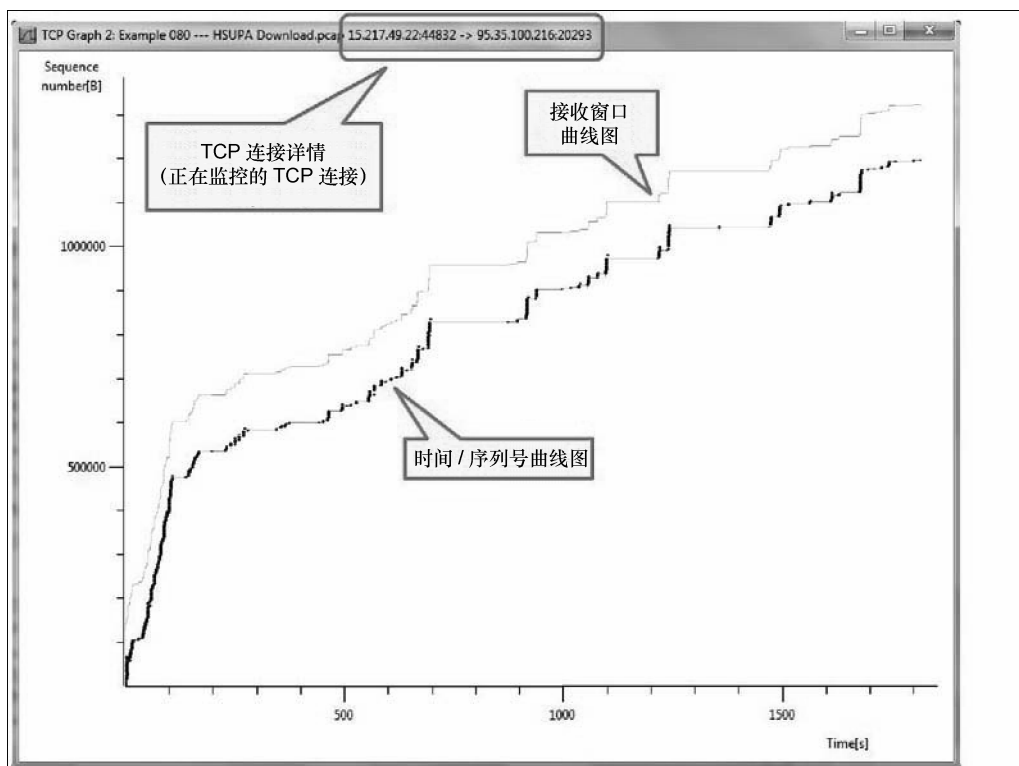


图5.19

图 5.19 中位置靠下的黑色曲线表示的是在单位时间内，受监控的 TCP 连接在某个方向上所传数据的字节数；而位置靠上的灰白色曲线反映的是（在整个数据传输过程中 TCP 报文段的）TCP 接收窗口。当两条曲线之间空间较大时，就表示（接收方主机）尚有缓存，可供 TCP 传输数据；若两者近乎重叠，则表示 TCP 窗口已满（window -full），不能继续传输数据。

3. 图 5.20 所示为在图 5.19 特定区域用鼠标圈出一块，放大之后的样子。

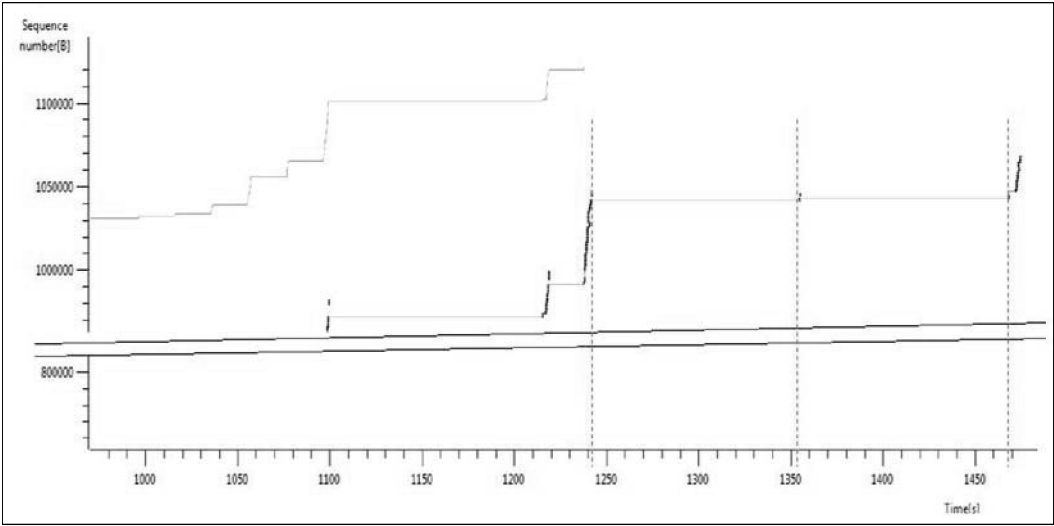


图5.20

4. 图 5.21 所示为抓包主窗口的数据包列表区域中与图 5.20 所示区域相对应的数据包。

No.	Time	Source	Destination	Protocol	Info
2234	1272.884906	15.217.49.22	95.35.100.216	FTP-DATA	FTP Data: 1398 bytes
2235	1272.885204	95.35.100.216	15.217.49.22	TCP	20293 > 44832 [ACK]
2236	1273.405891	15.217.49.22	95.35.100.216	FTP-DATA	[TCP Previous segment]
2237	1273.405939	95.35.100.216	15.217.49.22	TCP	[TCP Dup ACK 2235#1]
2238	1273.464842	15.217.49.22	95.35.100.216	FTP-DATA	FTP Data: 1398 bytes
2239	1273.464888	15.100.216	15.217.49.22	TCP	[TCP Dup ACK 2235#2]
2335	1386.067221	15.217.49.22	95.35.100.216	FTP-DATA	[TCP Retransmission]
2336	1386.236423	95.35.100.216	15.217.49.22	TCP	20293 > 44832 [ACK]
2337	1386.608202	15.217.49.22	95.35.100.216	FTP-DATA	[TCP Retransmission]
2338	1386.608252	15.100.216	15.217.49.22	TCP	[TCP Dup ACK 2336#1]
2385	1499.617524	15.217.49.22	95.35.100.216	FTP-DATA	[TCP Retransmission]
2386	1499.617876	95.35.100.216	15.217.49.22	TCP	20293 > 44832 [ACK]

图5.21

由图 5.21 可知，Wireshark 在抓到编号为 2239 的 TCP 数据包之后（抓取时间为自开始抓包以来的第 1273.464888 秒），传输中断；在抓到编号为 2338 的数据包之后，传输再次中断。

传输中断现象在 TCP Graph 窗口中的 Time-Sequence（时间/序列号）曲线图中一目了然，但只有在抓包主窗口的数据包列表区域中，定位到对应于曲线图的数据包，才好去了解传输中断的原因。

5.6.3 幕后原理

TCP StreamGraph 菜单项下的 Time-Sequence(tcp-trace)子菜单项功能脱胎于 UNIX

tcpdump 命令，启用该菜单项功能，就会让 Wireshark 统计（特定 TCP 数据流在特定方向上）由 TCP 接收方所通告的 TCP 窗口大小（TCP 接收方分配给 TCP 会话的系统缓存容量）、重传的数据包以及对所收数据的确认（ACK）情况等。

由 **Time-Sequence(tcp-trace)**子菜单项功能生成的曲线图所反映出的信息量极其丰富，可为网络排障的诊断提供重要的线索。这样的曲线图能栩栩如生地反映出 TCP 数据传输过程中的诸多现象，比如，TCP 窗口的填充速度高于预期的大规模 TCP 重传等现象。

5.6.4 拾遗补缺

把由 **Time-Sequence(tcp-trace)**工具生成的图形放得越大，所观察到的 TCP 数据传输过程中的细节便越多，如图 5.22 所示。

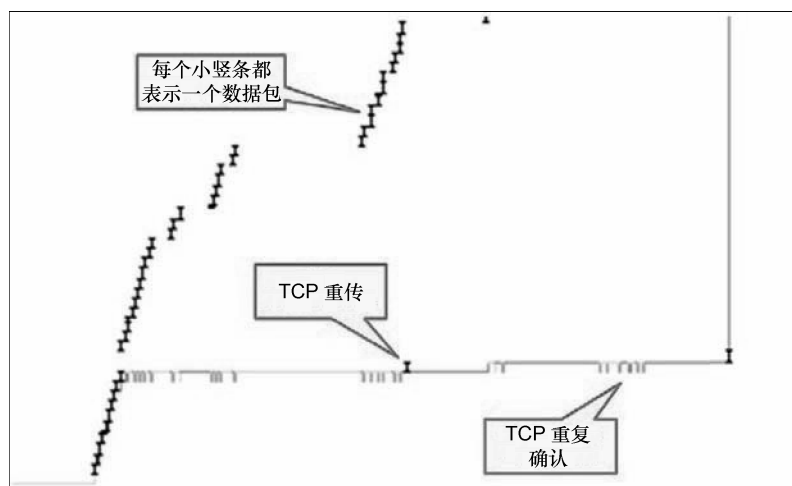


图5.22

（经过放大的）曲线图中的每个小竖条都表示携带应用层数据的 TCP 数据包，其 TCP 起始序列号和终止序列号（即所在位置）都与纵坐标（sequence number）上的数字相对应。图 5.22 中脱离于大部队的那个黑色小竖条表示一次 TCP 重传，而那个灰色的小竖条则表示重复确认。本书第 9 章会细述诸如 TCP 重传和重复确认之类的 TCP 事件。

5.7 TCP StreamGraph菜单项中Throughput Graph子菜单项的用法

借助于 **TCP StreamGraph** 菜单项中的 **Throughput Graph** 子菜单项功能，不但能了解某条 TCP 连接的吞吐量，而且还能判断 TCP 连接是否稳定。

5.7.1 使用准备

启动 Wireshark 软件，打开一个抓包文件，或选择一块网卡，点击 **Start** 按钮开始抓包。虽然可在抓包的同时使用 **TCP StreamGraph** 菜单项中的 **Throughput Graph** 工具，但该工具并不会在线实时生成统计信息，故建议在使用其之前，先停止抓包。

5.7.2 使用方法

要观看由 **TCP StreamGraph** 菜单项下 **Throughput Graph** 工具生成的统计信息，请按以下步骤行事。

1. 在抓包主窗口的数据包列表区域内，选中一个隶属于有待监控的某股 TCP 数据流的数据包。
2. 选择 **Statistics** 菜单下的 **TCP StreamGraph** 菜单项，点击其名下的 **Throughput Graph** 子菜单项。TCP Graph 窗口会立刻弹出，如图 5.23 所示。

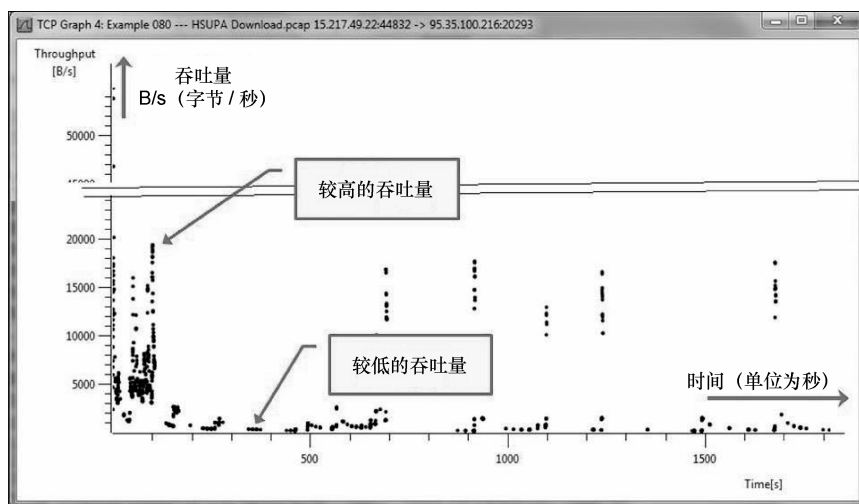


图5.23

通过图 5.23，可以观察到这条 TCP 连接的数据传输速率（吞吐量）很不稳定，最高时能达到 20000 字节/秒，但最低时还不到 1000 字节/秒。这种情况，既有可能是由于跨 Internet 运营商（服务器是电信 IP 地址，客户机是联通 IP 地址）传输数据所导致，也有可能是其他原因，比如，浏览国外网站。

5.7.3 幕后原理

TCP StreamGraph 菜单项下的 **Throughput Graph** 工具只统计（单位时间内）在某一指

定方向上传的数据包的字节数，亦即“清点”数据包的 TCP 头部中的序列号字段值，然后再以图形的方式加以呈现。因此，以此统计出来的吞吐量（流量传输速率）实际上只是在某个方向上传的（经由 TCP 头部封装的）应用程序数据（包含应用程序头部，不含 IP 包头及 TCP 头部）的吞吐量，单位为字节/秒（B/s）。

5.7.4 拾遗补缺

若 TCP 连接的数据传输速率非常稳定，则 **Throughput Graph** 工具将几乎生成一条实线，如图 5.24 所示。

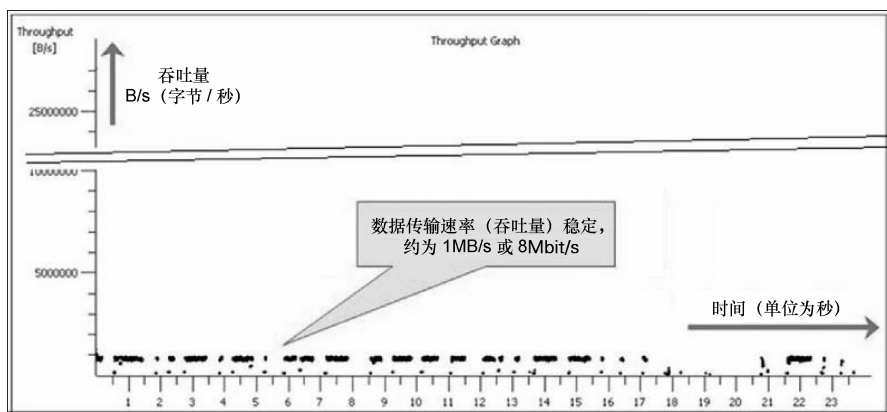


图5.24

5.8 TCP StreamGraph 菜单项中 Round Trip Time Graph 子菜单项的用法

借助于 **TCP StreamGraph** 菜单项中的 **Round Trip Time Graph** 子菜单项功能，能了解到某条 TCP 连接中特定方向上的所有 TCP 报文段的往返时间（RTT）。所谓某 TCP 报文段的往返时间是指，具有某特定序列号字段值的 TCP 报文段从发出直至收到接收方的 TCP 确认报文段所经历的时间。通过观察 **Round Trip Time Graph** 工具生成的图形，可以很好地了解指定 TCP 连接的性能。

5.8.1 使用准备

启动 Wireshark 软件，打开一个抓包文件，或选择一块网卡，点击 **Start** 按钮开始抓包。虽然可在抓包的同时使用 **TCP StreamGraph** 菜单项中的 **Round Trip Time Graph** 工具，但该工具并不会在线实时生成统计信息，故建议在使用其之前，先停止抓包。

5.8.2 使用方法

要观看由 **TCP StreamGraph** 菜单项下 **Round Trip Time Graph** 工具生成的统计信息，请按以下步骤行事。

1. 在抓包主窗口的数据包列表区域内，选中一个隶属于有待监控的某股 TCP 数据流的数据包。
2. 选择 **Statistics** 菜单下的 **TCP StreamGraph** 菜单项，点击其名下的 **Round Trip Time Graph** 子菜单项。TCP Graph 窗口会立刻弹出，如图 5.25 所示。

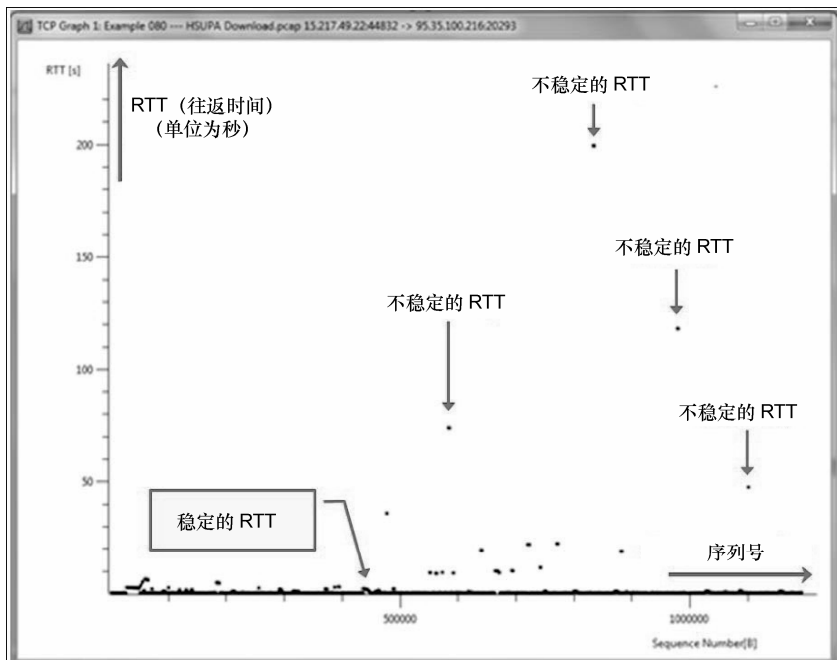


图5.25

由图 5.25 可知，这条 TCP 连接中的大多数 TCP 报文段，从发出直至收到接收方确认所消耗的时间都非常之短。当然，在数据的传输过程中，也发生了几次 TCP 报文段长时间得不到确认的情况（即有若干 TCP 报文段的 RTT 值较高）。

5.8.3 幕后原理

由 **TCP StreamGraph** 菜单项下 **Round Trip Time Graph** 工具生成的图形实际上为（该 TCP 连接中所传 TCP 报文段的 TCP 头部的）序列号字段值与（收到确认相应 TCP 报文段所

耗)时间之间的关系图。也就是说,该图会记录下该 TCP 连接中(在指定方向上)传输的每一个 TCP 报文段从发出直至收到接收方的确认所消耗的时间。

5.8.4 拾遗补缺

若在 **Round Trip Time Graph** 工具生成的图形中发现 TCP 报文段的 RTT 值很不稳定,也并不表示网络一定发生了问题,要视用户的行为或具体应用程序的运作方式而定^①。

5.9 TCP StreamGraph 菜单项中 Window Scaling Graph 子菜单项的用法

借助于 **TCP StreamGraph** 菜单项中的 **Window Scaling Graph** 子菜单项功能,能了解到通过 TCP 连接传送数据时,由接收方所通告的窗口大小^②。对于任何一条 TCP 连接,当接收方发送 TCP 报文段确认收到的数据时,会以设置 TCP 头部中窗口字段值的形式,向发送方通告本方接收数据的能力。通过观察 **Window Scaling Graph** 工具生成的图形,可以很好地了解特定 TCP 连接的性能。

5.9.1 使用准备

启动 Wireshark 软件,打开一个抓包文件,或选择一块网卡,点击 **Start** 开始抓包。虽然可在抓包的同时使用 **TCP StreamGraph** 菜单项中的 **Window Scaling Graph** 工具,但该工具并不会在线实时生成统计信息,故建议在使用其之前,先停止抓包。

5.9.2 使用方法

要观看由 **TCP StreamGraph** 菜单项下 **Window Scaling Graph** 工具生成的统计信息,请按以下步骤行事。

1. 在抓包主窗口的数据包列表区域内,选中一个隶属于有待监控的某股 TCP 数据流的数据包。
2. 选择 **Statistics** 菜单下的 **TCP StreamGraph** 菜单项,点击其名下的 **Window Scaling Graph** 子菜单项。TCP Graph 窗口会立刻弹出,如图 5.26 所示。

① 译者注:原文还有一句“You can see that it took time to acknowledge a packet because there is a problem, or because a server is waiting for a response, or because a client is browsing a web server and the user is waiting between clicks on new links”。译者认为这句纯属画蛇添足,掠过不译。

② 译者注:原文是“The Window Scaling Graph of the TCP stream graph enables us to look at the window size published by the receiving side”。译文按字面意思翻译,但译者认为原文表达并不精确,让译者举例加以解释。对于 TCP 连接“IP 地址 A: 端口号 80>IP 地址 B: 端口号 1026”,出现在由该工具生成的图形中的窗口大小其实是数据发送方的窗口大小,即 IP 地址 A 的窗口大小。

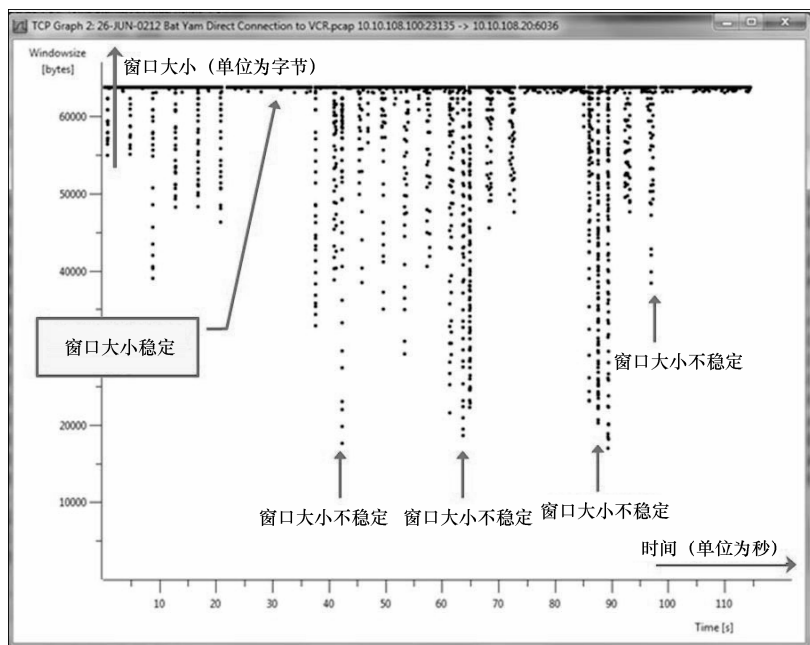


图5.26

由图 5.26 可知，由数据接收端所通告的窗口时大时小。这表示在数据传输过程中，接收方经常无暇处理收到的数据，于是便以降低接收窗口的形式，告知发送方：自己的接收能力有限，请不要发得太快^①。

5.9.3 幕后原理

TCP StreamGraph 菜单项下的 **Window Scaling Graph** 工具会记录在 TCP 连接的特定方向上传递的每一个 TCP 报文段的 (TCP 头部的) 窗口字段值，然后再以图形的方式加以呈现^②。与 TCP 窗口有关的内容，详见第 9 章。

5.9.4 拾遗补缺

当 TCP 接收方所通告的窗口变窄时，相关应用程序的吞吐量也会相应降低。TCP 窗口大小完全受控于建立 TCP 连接的两个端点 (亦即服务器和客户端)，TCP 窗口大小的变化与网络 (自身的) 性能无关。

^① 译者注：其实是 10.10.108.100 “告诉” 10.10.108.20：本机接收缓存有限，请慢些传送数据。

^② 译者注：原文是 “The software here simply watches the window size on the connection and draws it”。

第 6 章

Expert Info 工具的法

本章涵盖以下主题：

- ▶ 如何使用 Expert Info 工具，执行排障任务；
- ▶ 认识 Errors 事件；
- ▶ 认识 Warnings 事件；
- ▶ 认识 Notes 事件。

6.1 简介

Expert Info 工具是内置于 Wireshark 软件中的最强大的工具之一，该工具不但能在抓包过程中自动“识别”网络中发生的异常情况，甚至还能给出导致异常情况的具体原因。本章将介绍 **Expert Info** 工具的法。本书后文还会细述如何将该工具与其他工具相结合，来发现并解决网络故障。

6.2 节会介绍如何在排除网络故障时使用 **Expert Info** 工具，随后几节会描述如何解读由该工具生成的信息。

6.2 如何使用 Expert Info 工具执行排障任务

借助于 **Expert Info** 工具，可获知由 Wireshark 软件（在抓包过程中）识别出的各种网络事件和异常情况。Wireshark 会把识别出的事件按出故障概率的高低，通过 **Expert Info** 工具，以事件类别的方式来加以显示，同时还使用了下列颜色来加以区分。

- Errors: 红色。
- Warnings: 黄色。
- Note: 淡蓝色。
- Chats: 蓝色。
- Comments: 绿色。

在抓包主窗口中，只要在数据包列表区域内选中了在特征上匹配上述某一类事件的数据包，窗口底部状态栏的最左边就会燃起相应颜色的 LED。

6.2.1 使用准备

启动 Wireshark 软件，先打开一个抓包文件，或选择一块网卡，点击 **Start** 按钮开始抓包。

6.2.2 使用方法

要使用 **Expert Info** 工具，请在抓包主窗口内选择 **Analyze** 菜单，点击其名下的 **Expert Info** 菜单项，Expert Infos 窗口会立刻弹出，如图 6.1 所示。

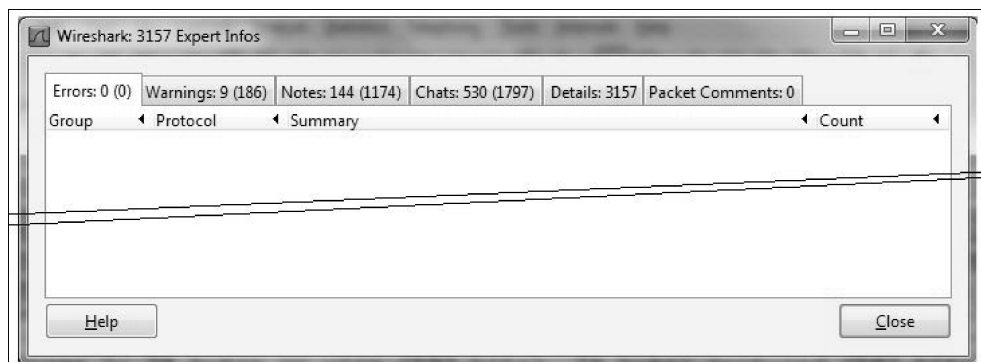


图6.1

不难发现，图 6.1 所示的 Expert Infos 窗口由 **Errors**、**Warnings**、**Notes**、**Chats**、**Details** 以及 **Packet Comments** 这 6 个标签栏构成。其中，前 5 个标签栏的名称也就是 Wireshark 通过 **Expert Infos** 工具给各种事件分类的名称。



在每个标签栏的事件名称的右边都有一个数字，用来表示该类事件中有多少类子事件。

下面是对 Expert Infos 窗口中各标签栏的介绍。

- **Errors:** 表示 Wireshark 在抓到的数据包中感知到或识别出了严重的错误，譬如，感知到了畸形数据包（畸形的 SPOOLSS 或 GTP 协议数据包等）或识别出了某些数据包的某种协议头部的某些字段值跟预期值不符（比如，IPv4 数据包的 IP 包头中校验和字段值跟预期值不符 [IPv4 数据包通不过校验和检查]）。

图 6.2 所示为 Wireshark 抓到了畸形的 TCP 和 SSL 数据包时，Expert Infos 窗口中 Errors 标签栏里的信息。

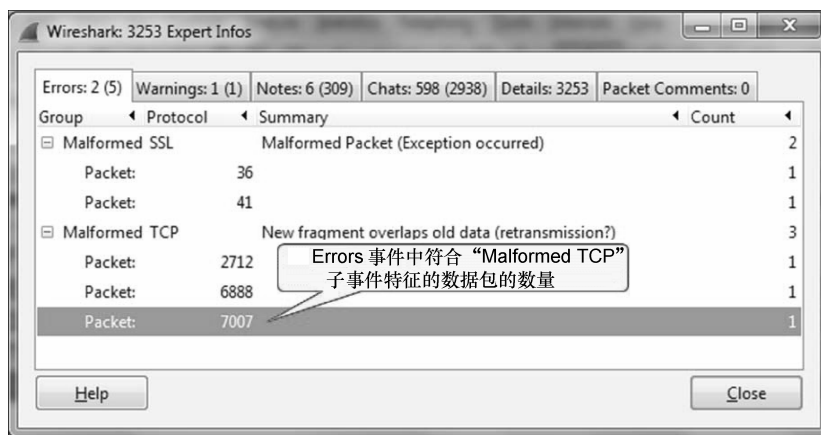


图6.2

图 6.3 所示为 Wireshark 抓到了符合 Errors 事件中另外两个子事件特征的数据包时，Expert Infos 窗口中 Errors 标签栏里的信息。由图可知，Wireshark 在抓到的数据包中，识别出了若干缺少某个（或某些）选项字段的 **BOOTP/DHCP** 数据包（缺失的选项字段已在 Summary 栏下注明）。

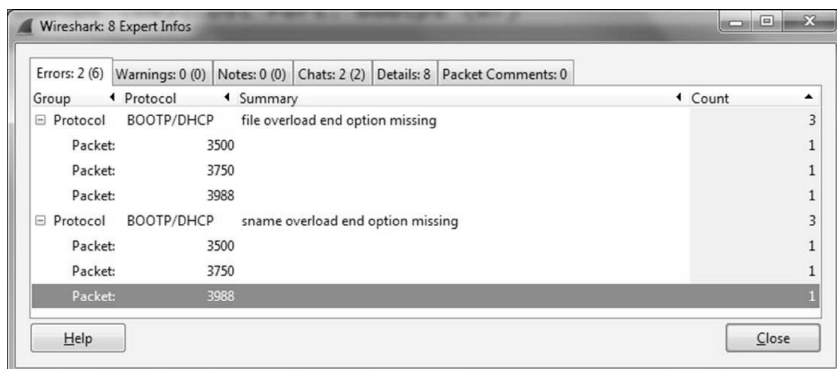


图6.3

- **Warnings:** 表示 Wireshark 在抓到的数据包中感知到或识别出了一般性问题，比如，感知到了存在 TCP zero window、TCP window full、TCP 报文段失序、TCP 报文段丢失等现象，或识别出了相关网络协议在运作时发出的数据包的内容与正常情况不一致。所谓一般性问题几乎都是应用程序问题或通信问题。图 6.4 所示为 Wireshark 抓到了符合 Warnings 事件特征的数据包时，Expert Infos 窗口中 Warnings 标签栏里的信息。

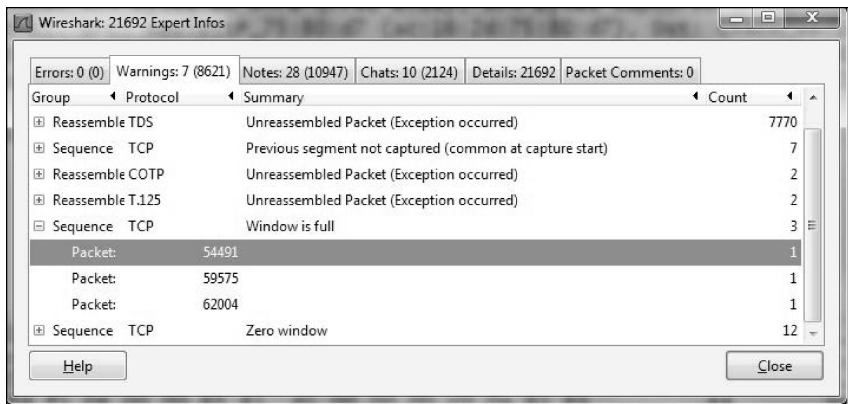


图6.4

- **Notes:** 表示 Wireshark 在抓到的数据包中，感知到了可能会引发故障的异常现象，比如，感知到了 TCP 重传、重复确认以及快速重传等现象。虽然上述现象可能会对网络产生严重影响，但也属于 TCP 的正常行为。Wireshark 把某些数据包归类为 Notes 事件，只是想提醒用户，这些数据包有导致问题的嫌疑。
- **Chats:** 出现在该标签栏中各类子事件名下的数据包都符合常规流量的特征。这样的子事件包括 TCP connection establish request (SYN)、TCP connection finish (FIN)、TCP connection reset (RST) 以及包含各种状态码的各种 HTTP 事件等。
- **Details:** 该标签栏里的信息是上述事件所含各种子事件的详情。老版本的 Wireshark 软件把该标签栏的功能直接安插在 **Analyze** 菜单的名下。
- **Packet Comments:** 可在抓包主窗口的数据包列表区域内给每个数据包添加注释信息。Packet Comments 标签栏会把含注释信息的数据包一一记录在案。

要给数据包添加注释信息，请先在抓包主窗口的数据包列表区域选中数据包，再单击右键，点击右键菜单中的 **Packet Comment** 菜单项（如图 6.5 所示），在弹出的 **Edit or Add Packet Comments** 窗口内输入所要添加的注释信息。

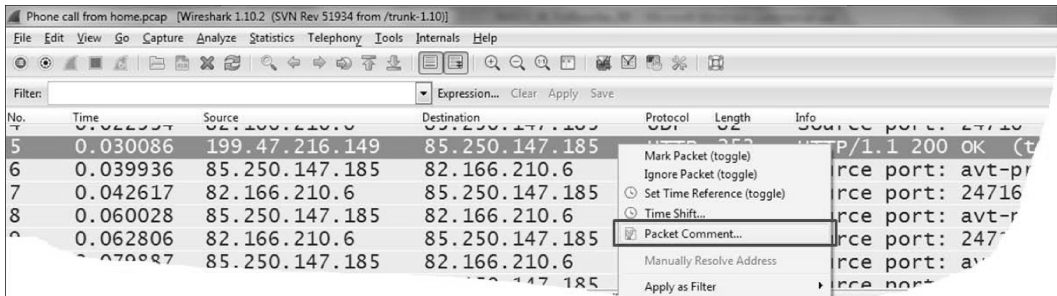


图6.5

只需在 Expert Infos 窗口的事件标签栏里单击符合某个子事件特征的数据包，便可在抓包主窗口的数据包列表区域里直接定位到该数据包。



注意

需要注意的是，在某些情况下，Wireshark “感知”到的某些 Warning 事件可能无关紧要，但“识别”出的某些 Notes 事件却偏偏会影响网络的性能。排除网络故障时，需关注其内在，切勿只看表面。只有如此，方能查明故障的来源。

6.2.3 幕后原理

Expert Infos 工具是内置于 Wireshark 中的一套专家系统，能自动提供与网络异常状况有关的信息，在某些情况下，还能给出导致网络异常的可能原因。对于这套专家系统的诊断结果，无论合不合乎情理，总是需要再三分析。

Wireshark 摆乌龙的情况时有发生，既有可能是杯弓蛇影，谎报军情（误报故障）；也有可能一叶障目（感知不到网络的异常状况）。



注意

不要忘记，解决网络故障靠的是网管人员的大脑（以及知识储备）。Wireshark 虽然比较智能，但也只不过是工具而已。

当出于某种原因（有利或不利原因），只能抓到部分数据（未能抓全数据）时，由于 Wireshark 并不知道自己所抓数据不完整，因此便会通过 **Expert Infos** 工具来指出网络中存在异常状况。本书后文所举诸多示例都涉及这种情况。

6.2.4 拾遗补缺

可基于 **Expert Infos** 工具所生成的事件信息或划定的事件分类，来配置显示过滤器，让

Wireshark 只显示符合某种事件特征的数据包。为此，请按以下步骤行事。

1. 点击抓包主窗口中数据包列表区域上方的 **Expression...** 按钮。
2. 在弹出的 Filter expression 窗口的左侧找到 **Expert-Expert Info** 字样，然后单击前面的“+”号。这将暴露出 `expert.message`、`expert.group` 和 `expert.severity` 三个显示过滤参数，如图 6.6 所示。

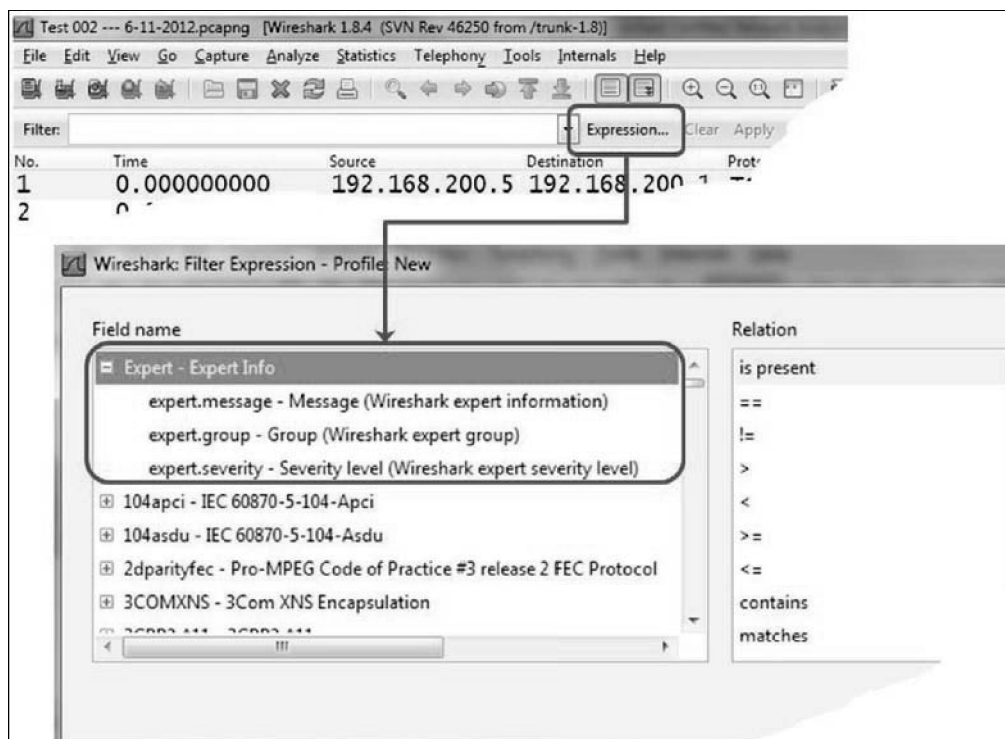


图6.6

- **expert.group**: 所指为 Expert Infos 工具生成的专家消息（expert message）的编组（或分类）。当显示过滤器中包含该参数时，Wireshark 便会根据专家消息的类型（比如，校验和问题、TCP 序列号问题以及安全性问题等），来进行过滤。图 6.7 所示为 **Expert Infos** 工具生成的专家消息的分类情况：

下面是对 `expert.group` 参数名下各类专家消息的介绍。

- **Checksum（校验和类）**: 匹配通不过校验和检查的数据包。
- **Sequence（序列号类）**: 匹配序列号有问题的 TCP 数据包。
- **Response（响应类）**: 匹配包含某种应用程序（针对某些请求操作，比如 HTTP

请求，返回）的响应代码的数据包（比如，包含 HTTP 4xx 代码的数据包）。

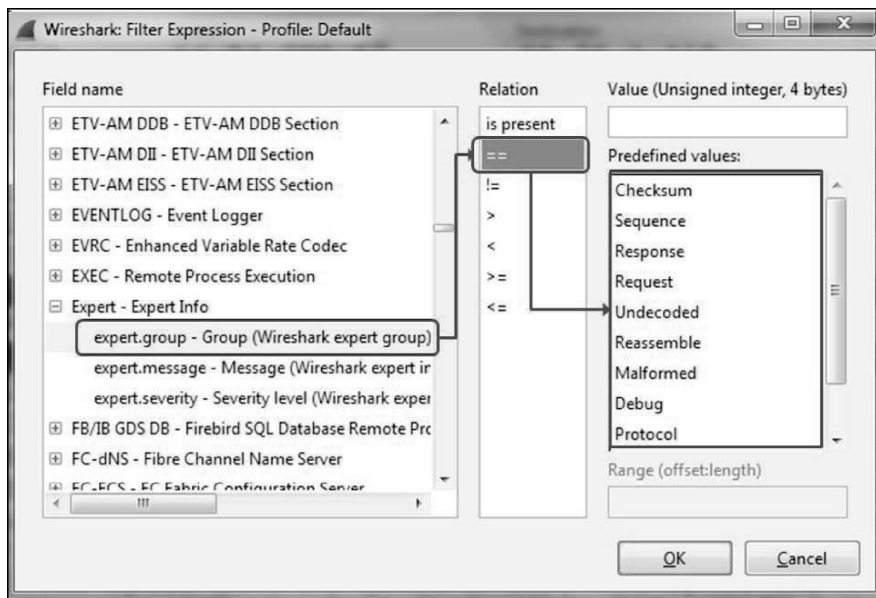


图6.7

- ▣ **Request（请求类）**：匹配供某种应用程序用来执行相关请求功能的数据包。
- ▣ **Undecoded（不能识别类）**：匹配 Wireshark 协议解析器不能识别（无法解码）的数据包。
- ▣ **Reassemble（重组问题类）**：匹配在重组（经过分片的数据包）时发生问题的数据包（一般都是指遗失了某个数据包的某一分片）。
- ▣ **Malformed（畸形数据包类）**：匹配畸形数据包，或因 Wireshark 协议解析器故障而终止解析的数据包。
- ▣ **Debug**：在 Wireshark 发布版中不应出现匹配此类特征的数据包。
- ▣ **Protocol（协议类）**：匹配违背协议规范的数据包（比如，数据包中的字段值有违标准，或缺少了某些原本应该存在的字段），Wireshark 协议解析器有可能会继续解析此类数据包。
- ▣ **Comment（含注释信息类）**：匹配包含注释信息的数据包。至于如何给数据包添加注释，前文已做过详细介绍，在此不加赘述。

➤ **expert.message**：所指为 **Expert Infos** 工具生成的具体的专家消息。当显示过滤

器中包含该参数时，Wireshark 便会基于专家消息中特定的字串（若关系词 [relation] 为 **contains**，则为包含特定的字串；若关系词为 **matches**，则为匹配特定的字串），来“筛选”数据包。

- **expert.severity**: 所指为 Wireshark 对感知到事件按出故障概率的高低，通过 **Expert Infos** 工具呈现出的事件类别。可供该参数选择的条件包括 **error**、**warning**、**note** 等。若在显示过滤器中包含该参数，则 Wireshark 在执行过滤功能时，所依据的条件就是数据包的特征是否符合特定的事件类别。

可配置 Wireshark，令其在 **Expert Infos** 窗口的各标签栏内也燃起相应颜色的 LED，配置步骤如下所列。

1. 点击抓包主窗口的 **Edit** 菜单下的 **Preferences....** 菜单项。
2. 在弹出的 **Preferences....** 窗口中点击 **User Interface** 配置选项。
3. 勾选右边面板最下方的 **Display LEDs in the Expert Infos dialog tab labels** 复选框，如图 6.8 所示。

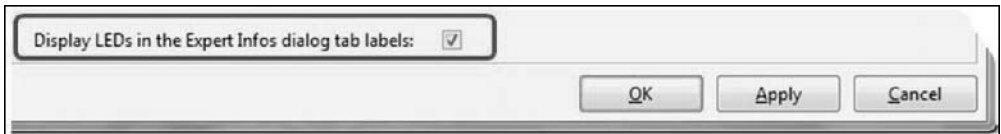


图6.8

4. 点击 **OK** 按钮。
5. 重新打开 **Expert Infos** 窗口，各标签栏内就会燃起相应颜色的 LED 了（见图 6.9）。

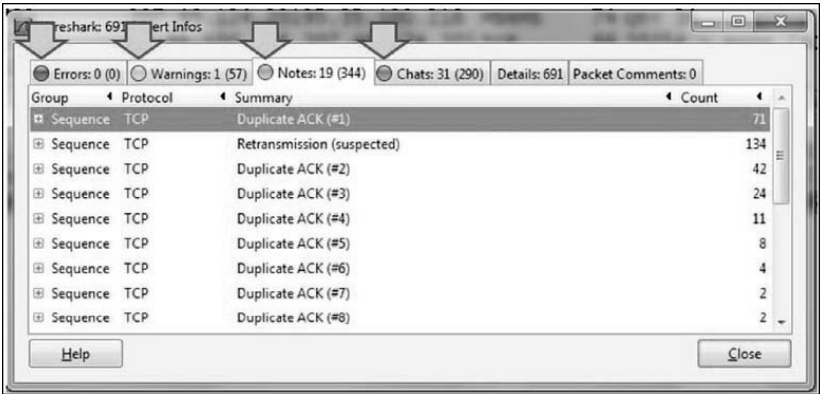


图6.9

6. 当然，在抓包主窗口中，只要在数据包列表区域内选中了在特征上匹配某一类事件的数据包，窗口底部状态栏最左边的 LED 也会呈现相应的颜色。

6.2.5 进阶阅读

➤ 第9章。

6.3 认识Errors事件

本节将引领读者去认识由 Wireshark 在抓包时所感知到或识别出的各种 Error 事件。

6.3.1 准备工作

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包）。

6.3.2 操作方法

1. 点击抓包主窗口中 **Analyze** 菜单下的 **Expert Info** 菜单项，**Expert Infos** 窗口会立刻弹出。
2. 点击 **Errors** 标签栏（其实根本用不着点击，Errors 标签栏是默认的最外层标签栏），如图 6.10 所示。

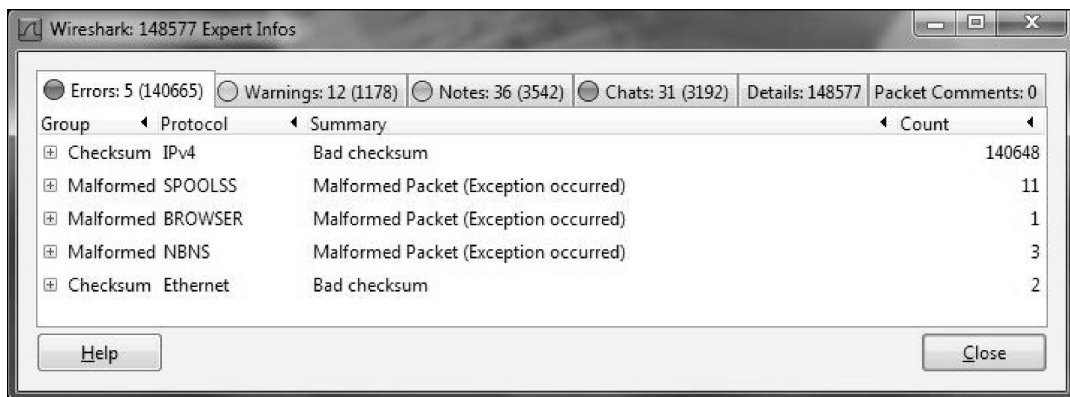


图6.10

由图 6.10 可知，Wireshark 感知到了 5 个子类的 Error 事件，主要分为以下两种情况。

- **校验和错误**：Ethernet 及 IP 校验和错误，这既有可能是 Ethernet 帧或 IP 数据包真的

通不过检验和检查所致，也有可能是 `checksum offload` 参数方面的配置问题^①。

- **伪造的数据包问题：**这一般都涉及具体的应用层协议^②。当然，遇到这种情况，也有可能是 Wireshark 的协议解析器问题。

6.3.3 幕后原理

校验和机制是一种错误检测机制，用来检测数据包在传输过程中是否发生了损坏。现以 IP 校验和计算来说明其原理：生成 IP 数据包的设备先把整个数据包按 16 位分成若干等份，再计算每一等份的二进制反码之和，然后计算二进制反码之和的反码，也就是说，最终的计算结果为每等份的反码之和的反码。最终的计算结果将被存入该 IP 数据包的校验和字段。接收 IP 数据包的设备会执行步骤相同的计算（执行计算时，要算上实际的校验和字段值），并判断最终的计算结果是否为 0。若不为 0，则表示 IP 数据包在传输过程中发生了损坏。错误检查机制既可以基于完整的数据包来执行，也可以基于协议头部来执行，每一种协议（Ethernet、IP、TCP、UDP）都有自己的一套错误检查机制。

某些操作系统支持一种名为 `checksum offload`（校验和计算下放）的功能，这也是为了节省 CPU 资源。也就是说，操作系统把执行 IP、TCP、UDP 校验和计算这样的网络功能下放给了 NIC（网卡），让 NIC 以硬件的方式执行相关计算（倘若 NIC 驱动支持），不再由 TCP/IP 协议栈来完成。若开启了出站方向（TX）上的 `checksum offload` 功能，NIC 便会在数据包即将上线之前，完成相关校验和计算。而 Wireshark 在抓取本机生成的数据包时，会在其“到达”NIC 之前完成，而此时由本机生成的数据包的 IP、TCP、UDP 头部的校验和字段值并不正确（对于本机生成的数据包，其 IP、TCP、UDP 头部的校验和字段值的填写任务归 NIC 负责）。于是，Wireshark 就会感知到数据包校验和有误之类的 Error 事件。

因为如此，即便 Wireshark 通过 **Expert Infos** 工具反映出抓到了许多校验和出错的数据包，但只要这些数据包的源 IP 地址为本机 IP，那就有极有可能是拜赐于 `checksum offload` 机制。可配置 Wireshark，令其不检查所抓数据包的 IP 或 TCP 校验和。

- 要关闭 IP 校验和的检查，请点击抓包主窗口的 **Edit** 菜单下的 **Preferences** 菜单项，在弹出的 **Preferences....** 窗口中，点击 Protocol 前的“+”号，选择其中的 **IPv4** 协议，取消勾选 **Validate the IPv4 checksum if possible** 复选框。
- 要关闭 TCP 校验和的检查，请点击抓包主窗口的 **Edit** 菜单下的 **Preferences** 菜单项，在弹出的 **Preferences....** 窗口中，点击 Protocol 前的“+”号，选择其中的 **TCP** 协议，

① 译者注：原文是“these can be in Ethernet, IP, or other protocols. In this case, it can be because of real errors or offload”。原文真挫，挫到了极致。

② 译者注：原文是“ These are usually in the application protocols”。作者啊，你用词能不能明确点啊？

取消勾选 **Validate the TCP checksum if possible**:复选框。

6.3.4 拾遗补缺

对于 Wireshark 自称抓到畸形数据包这种情况,要一分为二来看待。出现这种情况,既有可能是 Wireshark 抓到了真的畸形数据包,也有可能是拜 Wireshark 软件自身的 bug 所赐。此时,可使用其他抓包工具来定位问题。可访问 Wireshark 官网,报告该软件可能存在的 bug。



注意

当 Wireshark 自称抓到了大把畸形数据包或通不过校验和检查的数据包时,问题极有可能出在 checksum offload 机制或 Wireshark 协议解码器上面。对任何一个网络而言,穿梭于其中的数据包只要有百分之一、二的害群之马(即具备 Error 事件特征的数据包),不但会导致事故频发(比如,导致 TCP 重传),而且还会使得网速慢过预期。因此,只要网络用起来大致正常, Wireshark 是不可能抓到那么多真的害群之马的。

6.3.5 进阶阅读

➤ 第9章。

6.4 认识Warnings事件

如前所述,当 Wireshark 在抓到的数据包中感知到了一般性问题时,便会通过 **Expert Infos** 工具生成 Warning 事件信息,而所谓的一般性问题几乎都是应用程序问题或通信问题。本节会介绍什么是 Warnings 事件。

6.4.1 准备工作

启动 Wireshark 软件,先打开一个抓包文件(或选择一块网卡,点击 **Start** 按钮开始抓包)。

6.4.2 操作方法

1. 点击抓包主窗口中 **Analyze** 菜单下的 **Expert Info** 菜单项, **Expert Infos** 窗口会立刻弹出。
2. 点击 **Warnings** 标签栏,如图 6.11 所示(图中所示 Warnings 事件名下的所有子事件只是用来举例)。

下面是对图 6.11 所示的 Warnings 事件的几种子事件的简单介绍。

- 含 **Reassembly** 字样的几种子事件:大多是指 Wireshark 抓到了未能重组的数据包。一般而言,这都是 Wireshark 协议解析器问题。

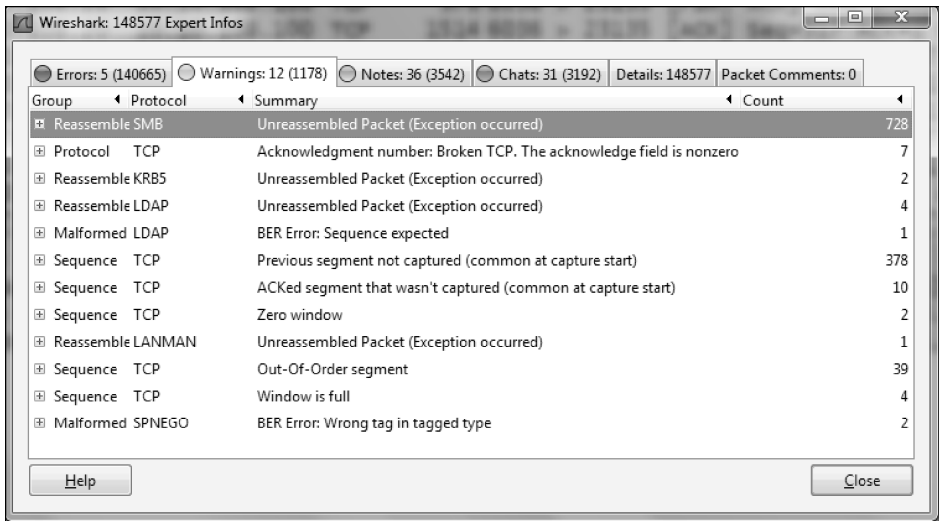


图6.11

- 与 TCP 窗口有关的两种子事件（在 Summary 一栏里含 **Window** 字样的子事件）：几乎都是指 Wireshark 感知到了网络中存在 TCP zero window 或 window full 问题。一般而言，都是建立 TCP 连接的端设备（客户端或服务端）忙不过来所致。
- 与 TCP 报文段丢失或失序有关的三种子事件（在 Summary 一栏里含 **Segment** 字样的子事件）：Wireshark 生成 TCP 报文段丢失事件信息，是因其未抓全隶属于某股 TCP 数据流的所有 TCP 报文段；生成 TCP 报文段失序事件信息，是因其感知到了 TCP 报文段未按发送主机发出的顺序到达接收主机。这三种子事件都属于 TCP 故障，一般是由网络故障所引发。

6.4.3 幕后原理

Wireshark 能感知到其所抓数据包的种种特征，比如：

- Wireshark 会关注数据包 TCP 头部中的窗口大小字段值，并会检测该字段值是否缩减为 0，同时还将通过 **Expert Infos** 工具生成相关信息；
- Wireshark 能识别出 TCP 报文段是否（在传输途中）失序，并会通过 **Expert Infos** 工具生成相关信息；所谓失序是指 TCP 报文段未按发送主机发出的顺序到达接收主机；
- Wireshark 能感知出接收主机在通过 TCP 收到数据之后，是否做出了确认，并会通过 **Expert Infos** 工具生成相关信息。

凭借 Wireshark 所提供的以上信息，再结合其他信息，定能有效定位与 TCP 有关的故障。本书第 9 章会细述如何排除涉及 TCP 的故障。

6.4.4 拾遗补缺

请注意，Warnings 事件只是 Wireshark 自认为比较重要的事件，却也未可轻信。比方说，若 Wireshark 未能抓全（整股数据流传输过程中的）所有数据包，则其必会生成 Warnings 事件，但这未必就是网络问题，可能只是抓包时机把握不对。

6.4.5 进阶阅读

➤ 第 9 章。

6.5 认识 Notes 事件

如前所述，当 Wireshark 在抓到的数据包中，只要感知到或识别出可能会导致问题的异常现象（TCP 重传、重复确认以及快速重传等现象）时，便会通过 **Expert Infos** 工具生成 **Notes** 事件信息。虽然这些异常现象可能会使网速变慢，但有时也属于 TCP 的正常行为。

6.5.1 准备工作

启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包）。

6.5.2 操作方法

1. 点击抓包主窗口中 **Analyze** 菜单下的 **Expert Info** 菜单项，**Expert Infos** 窗口会立刻弹出。
2. 点击 **Notes** 标签栏，如图 6.12 所示（图中所示 Notes 事件名下的所有子事件只是用来举例）。

下面是对图 6.12 所示 Notes 事件的几种子事件的简单介绍。

- TCP 重传、重复确认以及快速重传这三种子事件（即 Summary 一栏里含 **Retransmissions**、**Fast Retransmission**、**Duplicate ACK** 字样的子事件）：通常都预示着网速慢、丢包或通过 TCP 传输数据的主机（应用程序）忙不过来。
- 与 TCP keep-alive 有关的子事件（即 Summary 一栏里含 **Keep-Alive** 字样的子事件）：通常预示着 TCP 或（基于 TCP 的）应用程序问题。
- IP 数据包的生存时间事件（即 Summary 一栏里含 **Time to live** 字样的子事件）：通常

预示着路由问题。

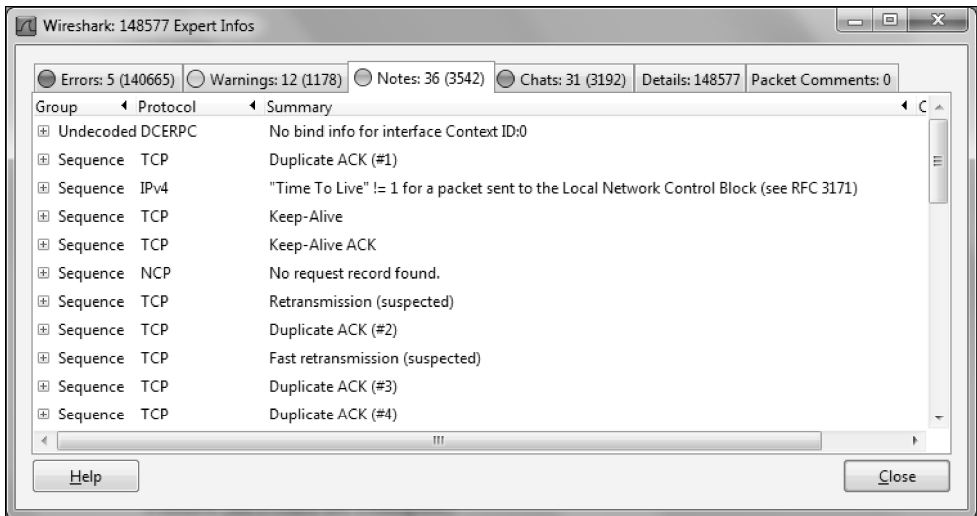


图6.12



注意

第 9 章到第 12 章会讨论 Notes 事件名下的其他几类子事件。

6.5.3 幕后原理

Wireshark 能感知到其所抓数据包的种种特征。对于 TCP 数据包，Wireshark 能通过检查其 TCP 头部中的序列号字段和确认号值，来发现 TCP 重传或其他与 TCP 序列号有关的问题。此外，Wireshark 还能识别出看起来正常但其实有问题的 TCP keep-alive 数据包。对于 IP 数据包，Wireshark 能通过检查其 IP 包头的 TTL 字段值，来判断是否发生了路由环路故障。当然，最为重要的是，Wireshark 能把感知到的所有风吹草动，用 **Expert Infos** 工具向网管人员汇报。

凭借 Wireshark 所提供的以上信息，再结合其他信息，定能有效定位与网络性能有关的故障。

6.5.4 拾遗补缺

本节介绍的 Notes 事件名下的每一种子事件，并非是由固定的某一种类型的网络故障所导

致。以 TCP 重传现象为例，造成这种现象既有可能是因为丢包错误所导致，也有可能是因为网络状态差（带宽低、延迟高）而引发的数据包未能按时到达所导致，还有可能是拜服务器或客户端停止响应所赐。Wireshark 会通过 **Expert Infos** 工具提示存在 TCP 重传这样的现象，至于如何定位原因及解决问题，还得靠我们自己。第 9 章将细述如何解决相关网络故障。

6.5.5 进阶阅读

第 9 章会介绍如何解决与 TCP 性能有关的故障。该章内容包括：什么是 TCP 重传和快速重传现象，为何会发生这样的现象；什么是 TCP 确认和重复确认；什么是 TCP zero window；TCP 窗口大小的改变，以及 TCP 滑动窗口问题等^①。

^① 译者注：原文是“what are ACKs and duplicate ACKs, zero window, window changes and other TCP sliding windows issues, and more”。原文如此，读者要骂，就骂作者好了，跟译者没有半点干系。

第7章

Ethernet、LAN交换及无线LAN

本章涵盖以下主题：

- ▶ 发现广播及错包风暴（broadcast and error storms）；
- ▶ 生成树协议分析；
- ▶ VLAN 及 VLAN tagging 问题分析；
- ▶ 无线（WiFi）问题分析。

7.1 简介

本章将关注如何发现并解决第二层网络故障，将重点介绍 Ethernet 相关故障（比如，广播风暴和错包风暴），以及如何定位故障之源头。此外，本章还会介绍几种第二层协议及技术，比如生成树协议、VLAN 技术及无线 LAN 技术。

由于第二层故障不但会影响高层协议的运作，而且还会在高层协议的运作过程中有所体现，因此在解决网络故障时，只有先确保网络的第二层完好无损，才有继续排查第三、四层协议的必要性。譬如，若在第二层有丢包现象，则势必会导致 TCP（第四层协议）重传，但最终的表象将会是应用程序无法运行或运行缓慢。

7.2 发现广播及错包风暴

广播及错包风暴应属通信网络中的最难解决的故障之一了。导致此类故障的原因很多，比如，第二层环路、针对第二层的攻击、网络适配器（网卡）故障，或某款应用程序（某台主机上的某个服务）持续不断地在网络中发包等。

本节会介绍几个发现、分类以及解决此类故障的秘诀。



注意

广播风暴是指在网络中传播的广播包的数量每秒高达数千乃至数万。一般而言，广播风暴发生之日，便是网络瘫痪之时。

7.2.1 准备工作

当网络中发生了广播风暴，网管人员得到征召前去处理时，收到不外是“网速怎么这么慢呀”或“网络老是中断”之类的反馈。因此，要想精确定位故障原因，以下常识必须具备。

- 路由器是不会转发广播流量的。
- 每个 VLAN 都是一个单独的广播域，VLAN 之间不会交换广播流量，所以说一个 VLAN 也被称为一个广播域。
- 任何一台 LAN 交换机都不会转发错包（至少运转正常的 LAN 交换机不会如此行事）。
- 除非有特殊配置，否则 LAN 交换机会从除接收端口以外的所有端口转发多播流量。
- 只有开启多播路由功能的路由器才会转发多播流量。
- 在每一个 Ethernet（LAN）中，都会存在数量合理的广播数据包；若非如此，主机之间便不能正常通信。但倘若广播数据包的数量过多，则反过来又会影响网络的正常运行。



注意

广播包数量过多跟广播风暴完全是两码事。广播包过多（比如，每秒三五百个）会加重网络的负担，但几乎不会降低用户对网络的使用体验，而广播风暴则会彻底导致网络瘫痪。

7.2.2 操作方法

要查明导致广播（或错包）风暴的原因，请按以下步骤行事。

1. 由于最先感觉出网络慢或断网的肯定是用户，所以应首先向他们咨询以下问题。
 - 是整个网络都有问题，还是总部网络或某个分支机构网络有问题？
 - （若问题发生在总部或某个分支机构，那就继续问）是整个（总部或分支机构）网络有问题，还是其中的某个 VLAN 有问题。

当然，在询问用户的时候可千万不要使用 VLAN 这样的专业词汇，用户可不懂网络。

应该这么问：是整个公司的网络有问题，还是某个或某些部门的网络有问题，或是某个部门中的某些人的网络有问题？



注意

在一个组织机构的网络中，VLAN 通常都基于每个（若干）部门、每个（若干）地理区域或每个行政职能单位来配置；比如，既可以把整个人力资源部或财务部的 PC 置入一个 VLAN，也可以把运行同一套业务软件的 PC 置入一个 VLAN。这样一来，只要问一下某个部门的某个员工，或使用某款业务软件的某个人，就能够缩小排障范围了。

2. 第二个问题应该好问的多：是所有与网络有关的应用程序都不能用了呢，还是使用起来卡的要命？若发生了广播风暴，网络会变得非常之卡，在大多数情况下，与网络有关的应用程序都将完全停止运作。到了如此田地，网管人员应扪心自问：

- 是生成树问题吗？
- 是某台设备触发了广播风暴吗？
- 是路由环路问题吗（本书第 8 章路由将深入探讨环路问题问题）？



注意

作者经常自问：“网络中广播包的数量达到多少才算是过多呢？”该问题的答案取决于网络设备的配置、其上所运行的协议，以及网络中主机的数量。

在一个运转正常的网络中，每台设备每分钟制造 1~2 个广播包（最多不超过 4~5 个）应算是合理。譬如，若网络中每个 VLAN 内有 100 台设备，则每秒广播包的数量最多不应超过 9~10 个（5 个广播包×100 台设备/60 秒）。倘若真的超过了这一数字，只要每秒数不过千，并且网管人员知道广播包的出处，也不能说网络存在问题。

生成树问题

只要生成树协议发生故障，充斥于网络中的广播包的数量将会达到每秒数千甚至上万（生成树协议的运作方式，以及因其故障而导致广播风暴的原因请见下一节）。此时，用来抓包的 Wireshark 软件，甚至连安装其的主机可能都会“卡死”。要想定位故障原因，只有先拔掉连接 Wireshark 主机抓包网卡的网线，才能让 Wireshark 正常显示抓到的数据包。然后，需登录交换机，检查 STP 相关配置，查看 STP 的运行状态及日志输出^①。

^① 译者注：原文是“Close Wireshark, disconnect cables to isolate the problem, and check the STP configuration in the switches”。译文酌改。

某台设备（主机）触发了广播风暴

当广播风暴是由某台设备（主机）所引起时，通常具有以下典型特征。

- 广播包速率极高（数千甚至上万个/秒）^①。
- 在绝大多数情况下，广播包都发源于单一源头，但遭到攻击除外（当网络遭受攻击时，广播包可能来源于多处）。
- 广播包速率恒定，亦即 Wireshark 所抓（广播）数据帧之间的时间间隔几乎完全相等。

现在，将以图 7.1 到图 7.3 为例，来讲解如何根据上述三大典型特征，去定位由某台主机所触发的广播风暴。

由图 7.1 可知，Wireshark 抓到了大把广播包，广播包的源 MAC 地址都一模一样（归一块 HP 网卡所有），目的 MAC 地址自然是 ff:ff:ff:ff:ff:ff。

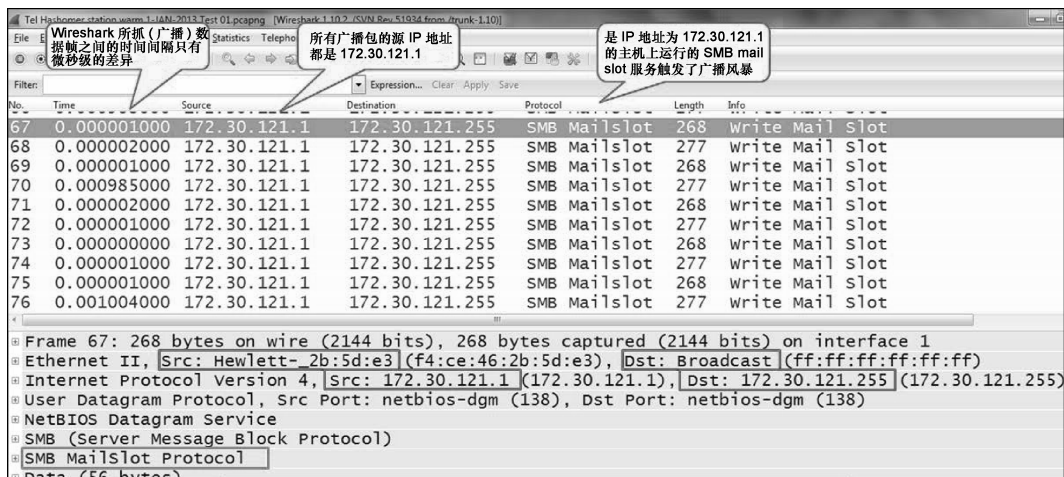


图 7.1

图 7.1 中抓包主窗口 Time 属性栏所启用的时间格式是“当前帧与 Wireshark 所显示出的上一帧之间的接收时间间隔（单位为秒）”（seconds since the previous displayed packet）。可在抓包主窗口中 **View** 菜单的 **Time Display Format** 菜单项下，选择数据包在 Time 属性栏里所呈现的时间格式。

图 7.2 所示为通过 Wireshark IO Graphs 工具所生成的图形，由图可知，目的 MAC 地址为

^① 译者注：原文是“Significant number of broadcasts per second (thousands and more)”。想问下作者，这能算是“典型”特征，难道由其他原因所导致的广播风暴就没有这样的特征？

ff:ff:ff:ff:ff:ff 的广播包速率已达到了 5000 个/秒。

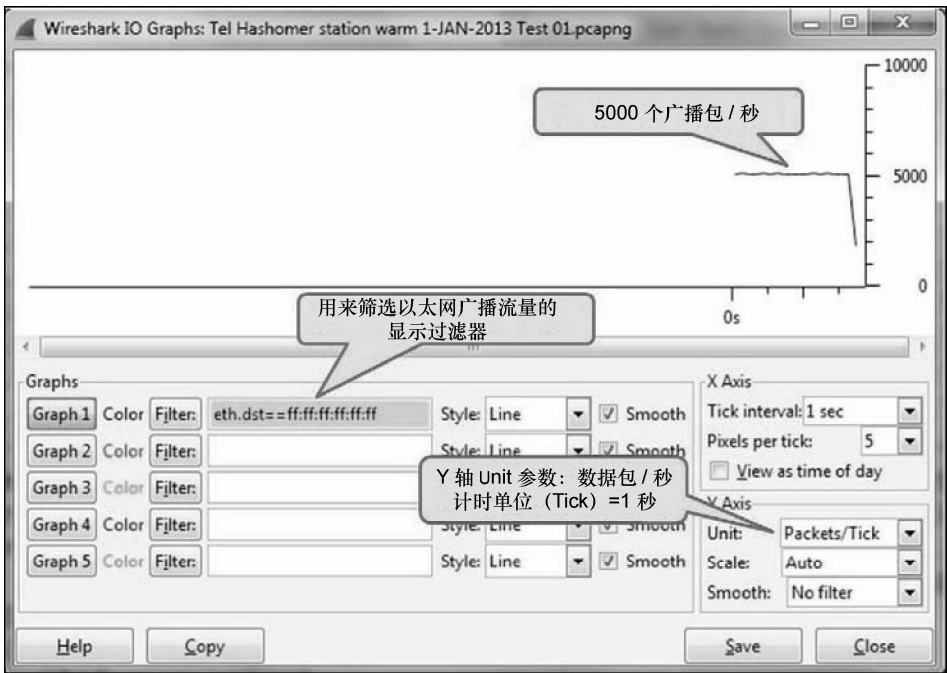


图 7.2

图 7.3 所示为由 **Statistics** 菜单名下的 **Conversation** 工具生成的 Conversations 窗口,通过该窗口中 Ethernet 或 IPv4 标签栏内的信息都能看出:网络中充斥着大量的广播数据包(本次抓包时长仅为 18 秒)。

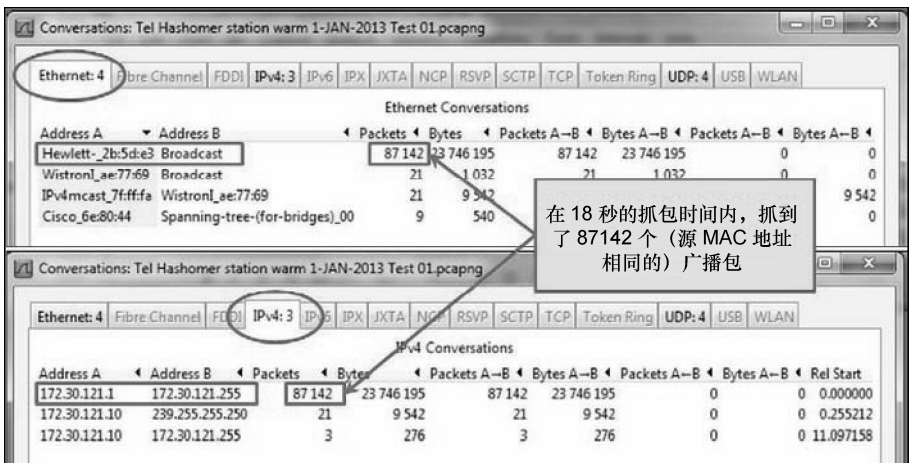


图 7.3

在本例中,导致广播风暴的罪魁祸首是一台安装了名为 SMB Mailslot 服务的主机。在 MAC 地址和 IP 地址已知的情况下,应该很快就能找到那台发送广播包的主机,只要仔细检查该主机,禁用运行于其上的 SMB Mailslot 服务,便可以解决广播风暴问题了。



注意

在禁用生产网内任何一台主机上的某个服务(特别是隶属于主机操作系统的服务)之前,应再三斟酌。在禁用之后,要等主机稳定运行一段时间之后,才能离开。

有时间规律可循的广播风暴

还有一种广播风暴,其发作的时间极有规律,发作的时间间隔也几乎保持恒定,请看图 7.4。

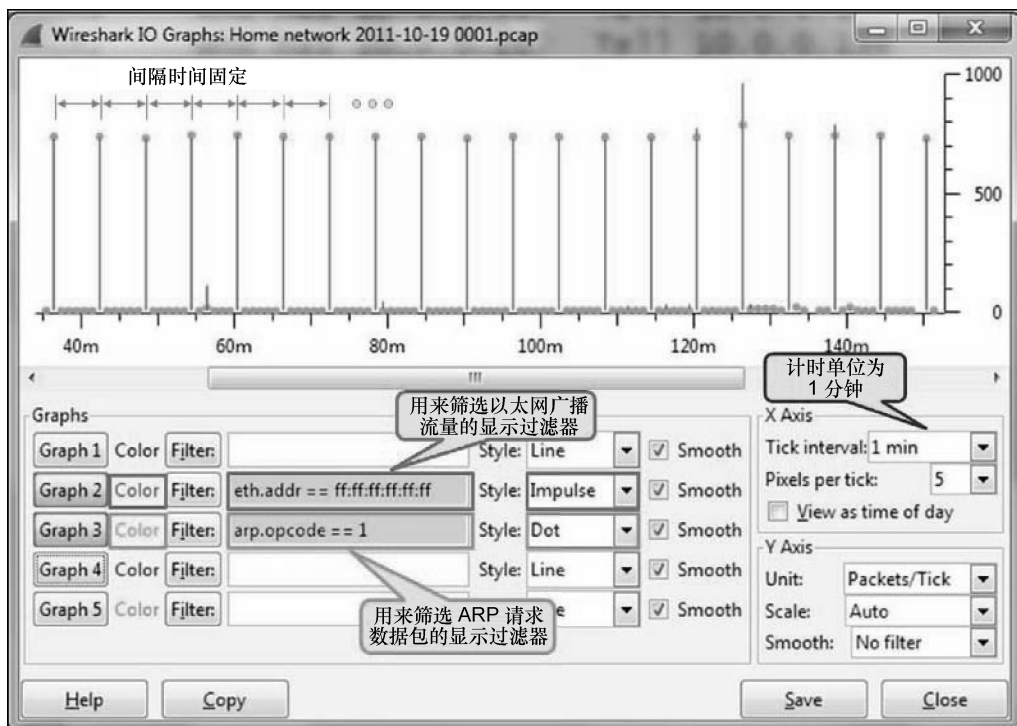


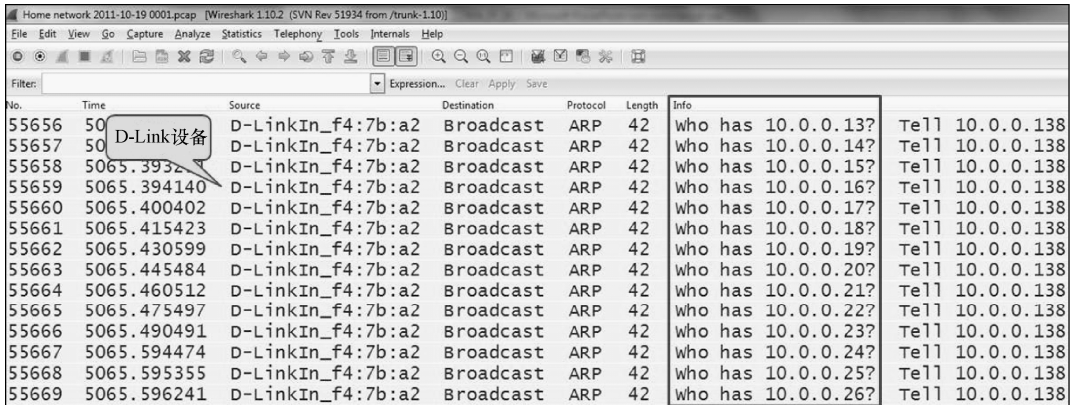
图 7.4

由图 7.4 可知,在 **IO Graphs** 工具所生成的图形中,X 轴参数计时单位(Tick interval)被设成了 1 分钟,此外,还针对 Graph 2 和 Graph 3 分别应用了下面这两个显示过滤器。

- 针对 Graph 2 (所生成的图形颜色为红色,形状为脉冲[impulse])应用了显示过滤器 `eth.addr == ff:ff:ff:ff:ff:ff`,意在筛选出所有以太网广播数据包。

- 针对 Graph 3（所生成的图形颜色为绿色，形状为点状[dot]）应用了显示过滤器 `arp.opcode ==1`，意在筛选出所有 ARP 请求数据包。

通过 **IO Graphs** 窗口图形显示区域中的红、绿两种图形，可以很容易地看出，ARP 请求数据包每隔 5 分钟就要来一次大规模喷发。只要点一下图形显示区域中任一（绿色）小点，就能在抓包主窗口的数据包列表区域定位到相应的（ARP 请求）数据包。图 7.5 所示为每 5 分钟喷发一次的 ARP 请求流量，在 Wireshark 抓包主窗口数据包列表区域里的样子。



No.	Time	Source	Destination	Protocol	Length	Info
55656	50	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.13? Tell 10.0.0.138
55657	50	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.14? Tell 10.0.0.138
55658	5065.3932	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.15? Tell 10.0.0.138
55659	5065.394140	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.16? Tell 10.0.0.138
55660	5065.400402	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.17? Tell 10.0.0.138
55661	5065.415423	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.18? Tell 10.0.0.138
55662	5065.430599	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.19? Tell 10.0.0.138
55663	5065.445484	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.20? Tell 10.0.0.138
55664	5065.460512	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.21? Tell 10.0.0.138
55665	5065.475497	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.22? Tell 10.0.0.138
55666	5065.490491	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.23? Tell 10.0.0.138
55667	5065.594474	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.24? Tell 10.0.0.138
55668	5065.595355	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.25? Tell 10.0.0.138
55669	5065.596241	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	who has 10.0.0.26? Tell 10.0.0.138

图7.5

由图 7.5 可知，所谓 ARP 请求流量定期喷发问题，是因一台 D-Link 路由器扫描内网所致。这台路由器的上述举动是坏还不好判断，第 14 章会细述与此有关的内容。熟知各种网络设备的习性，在任何情况下都不是坏事。

7.2.3 幕后原理

在 Ethernet 中，第三层 IP 广播包在传播之前会被先封装为第二层以太网帧。对于设有 IPv4 地址的设备所发出的每一个 IP 广播包（在其目的 IP 地址中，至少最后 1 个字节为全 1[二进制]），封装其的以太网帧的目的 MAC 地址必定为全 1（二进制）。

以下所列几种在 IP 网络中常见的广播包。

- 配合 TCP/IP 协议运行的广播包，比如，ARP 请求数据包、DHCP 请求数据包等。
- 某些应用层协议生成的广播包，比如，NetBIOS 名字服务（**NetBIOS Name Service, NBNS**）查询数据包、NetBIOS 服务器消息（**NetBIOS Server Message Block, SMB**）通告数据包，以及网络时间协议（**Network Time Protocol, NTP**）数据包等。

- 某些应用程序（比如，Dropbox、Microsoft Network Load Balancing 或某些证券期货类应用等）也会生成广播包。

IPv6 只分单播、多播和任播，无广播一说。因此，IPv6 协议要依靠多播来行使 IPv4 协议通过广播来完成的诸多功能，比如，邻居发现功能（相当于 IPv4 的地址解析功能）、地址自动分配功能（DHCP）。与此有关的内容将在本书后文介绍。

7.2.4 拾遗补缺

在作者处理过的众多案例里，经常会碰到同一个问题，那就是如何配置 LAN 交换机上的多播/广播风暴控制特性（在 Cisco 交换机上，要通过 `storm-control broadcast /multicast level` 命令来激活该特性）。

据作者所知，有很多人在配置多播/广播风暴控制特性时，都把广播（或多播）数据包的速率配置为 50、100 或 200 个/秒，其实这远远不够。只要在网络中部署了基于广播（或多播）的应用程序，广播（或多播）包的流动速率将会超过上述配置值。这样一来，必将导致交换机向网管系统发送 trap 消息，甚至会自动 shutdown 多播数据包速率超限的端口。交换机到底会如何行事，则要视其多播/广播风暴特性的配置参数来决定（Cisco 交换机依靠 `storm-control action{shutdown | trap}` 命令来决定是发送 trap 消息，还是 shutdown 相关端口）。

其实，只要把广播（或多播）数据包的速率阈值指定的高一点就可以规避上述问题了。当广播风暴来临时，网络中广播包的速率可能会接近每秒上万。因此，把广播（或多播）数据包的速率阈值设置为每秒 1000~2000，则既可以起到安全防护的目的，也不会对常规的网络操作造成任何影响。

7.2.5 进阶阅读

- 第 8 章。

7.3 生成树协议分析

读者应该都和生成树协议（STP）打过交道，最起码也听说过这种协议。作者之所以给本节冠以“生成树协议分析”之名，是因为该协议有以下三种主要版本。

- （常规的）生成树协议（**Spanning Tree Protocol, STP**）：基于 1998 年颁布的 IEEE 802.1D 标准（亦称 802.1D-1998）。
- 快速生成树协议（**Rapid Spanning Tree Protocol, RSTP**）：基于 2001 年颁布的 IEEE

802.1W 标准，后被追加至 802.1D 标准（802.1D-2004）。

- **多生成树（Multiple Spanning Tree, MST）**：最初，定义于 IEEE 802.1S 标准，后来并入了 IEEE 802.1Q 标准。

除以上列出的三个 STP 版本之外，Cisco 及其他网络设备厂商也开发过几个 STP 的私有版本。本节将聚焦于 STP 标准版本 STP/RSTP/MST，重点关注如何排除与此有关的故障。

7.3.1 准备工作

查明 STP 故障的最佳途经就是登录 LAN 交换机，执行（LAN 交换机）厂商的相关命令（比如，Cisco IOS 或 JUNOS CLI）去发现并解决故障^①若在 LAN 交换机上启用了 SNMP 功能，则网管控制台可能会收到与 STP 有关的 SNMP traps 信息。

本节的主旨是如何利用 Wireshark 来协助排除 STP 故障。

请打开笔记本，启动 Wireshark，开始在 LAN 里抓包吧。

7.3.2 分析方法

要解决 STP 故障，先得回答网络中与 STP 有关的以下两个问题。

- 网络中运行的是哪个版本的 STP？
- 在故障显现的同时，发生过网络拓扑变更事件吗？

网络中运行的是哪个版本的 STP

通过对网桥协议数据单元（**Bridge Protocol Data Unit, BPDU**）的解析，Wireshark 能识别出网络中运行的是哪个版本的 STP。BPDU 是一种在开启 STP 功能的交换机之间传递的信令帧，以（第二层）多播方式发送。

- 运行常规 STP 的交换机会发出协议版本 ID 字段值为 0 的 BPDU。
- 运行 RSTP/MST 的交换机会发出协议版本 ID 字段值为 3 的 BPDU。



注意

在定义 STP 的相关标准文档中，根本就没有出现过 switch（交换机）这样的字眼，只能看见 bridge（网桥）或 multiport bridge（多口网桥）之类的同义词。本书会交替使用网桥和交换机这两个术语。

^① 译者注：原文是“The best way to find out STP problems is to log in to the LAN switches and use the vendor's commands (for example, Cisco IOS or Juniper JUNOS CLI) to find and fix the problem”。原文不忍卒读，译文直译。

发生过网络拓扑变更事件吗？

解决 STP 故障时，应重点关注网络中是否多次发生拓扑变更事件。网络拓扑发生变化也在情理之中，但若发生的次数太多，则会对网络性能产生影响。

典型的拓扑变更事件包括 LAN 链路中断、LAN 内有新交换机上线运行等。图 7.6 所示为 Wireshark 抓到的表示发生拓扑变更事件的 BPDU^①。

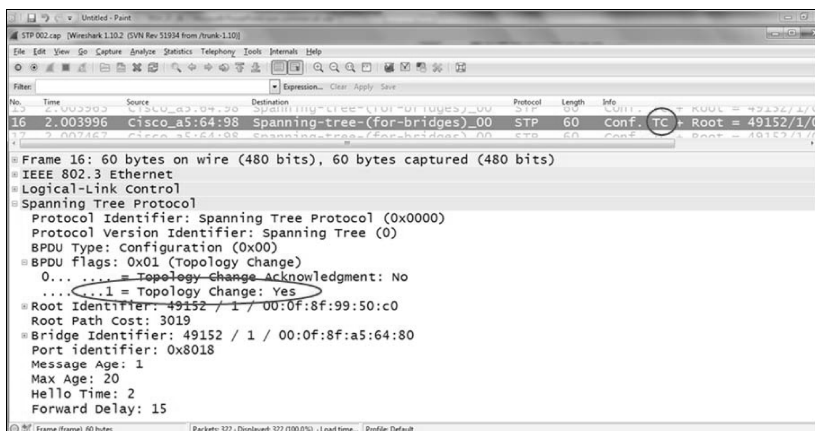


图 7.6

要是 Wireshark 感知出超多的网络拓扑变更事件（一般都是由用户频繁开关 PC 所引起），请登录 LAN 交换机，在用来直连主机（即不支持 STP 的设备）的端口上激活 portfast（速端口）特性（Cisco 交换机的私有特性，其他网络设备厂商也有类似特性，其具体称谓请查阅各厂商的随机文档）。



注意

对于运行常规 STP (IEEE 802.1D) 的 Cisco 交换机而言，当有主机接入时，须坐等约 1 分钟，才能开始正常收发数据包。在此期间，该主机将无法与任何其他网络设备通信。为防止此类情况的发生，Cisco 交换机支持一种名叫 portfast 的特性，只要在直连主机的交换机端口上激活该特性，主机在正常收发数据之前将只需稍等片刻（通常为 8~10 秒）^②。

倘若在执行上述操作之后，网络拓扑变更事件依旧持续发生，那就需要开展更深层次的排查工作。

① 译者注：图 7.6 中所示的 BPDU 为 TC（拓扑变更）标记位置 1 的配置 BPDU，其作用是缩短交换机 MAC 地址表的 MAC 地址超时周期。

② 译者注：原文是 “In the old STP (IEEE 802.1d), after connecting a device to a switch port, it takes the switch around a minute to start and forward packets. This can be a problem when a client tries to log in to the network servers during this period of time. The portfast feature forces the port to start forwarding within a few seconds (usually 8 to 10), in order to prevent these kinds of problems”。原文极度无厘头，译文未按字面意思翻译。

7.3.3 幕后原理

开发生成树协议的目的，是要确保 LAN 中无第二层环路产生。若像图 7.7 所示那样用多条链路把两台或两台以上的交换机连在一起，LAN 中将会产生环路。

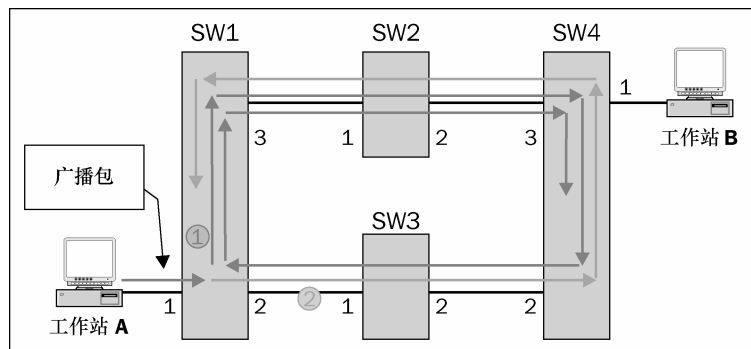


图7.7

现在来看一下 LAN 中的广播风暴是如何因环路而起的。

1. 工作站 A 把一广播包发送进 LAN。这一广播包可以是 ARP 帧、NetBIOS 数据包或其他任何目的 MAC 地址为全 1 的以太网帧。
2. 由于交换机会向除接收端口以外的所有端口转发广播包，因此从端口 1 收到广播包之后，SW1 会从端口 2、3 向外发送。
3. 收到广播包之后，SW2 和 SW3 会分别通过端口 2 外发给 SW4。
4. SW4 会把从端口 2 和端口 3 收到的广播包，再分别从端口 3 和端口 2 外发。
5. 现在，便诞生了 2 个一模一样的广播包，（广播包的发源地）SW1 上的端口 1 和端口 3 也将会分别收到一个。
6. SW1 会再次从端口 3 和端口 1 外发，其余交换机也会无止境地复制广播包，成千上万个广播包很快会封锁整个 LAN。当然，到底有多快，则要取决于那几台交换机的转发速度。

启用了生成树协议的交换机之间会在逻辑层面自动构建树状拓扑（无环拓扑），从而能起到预防环路的效果。也就是说，交换机之间势必会有被生成树协议阻断的冗余链路，若在用链路失效，生成树协议也能感知得到，会自动让交换机激活先前被阻断的冗余链路。

图 7.8 所示为在一个充满冗余链路的 LAN 内，STP 是怎样创建树状拓扑结构的。