

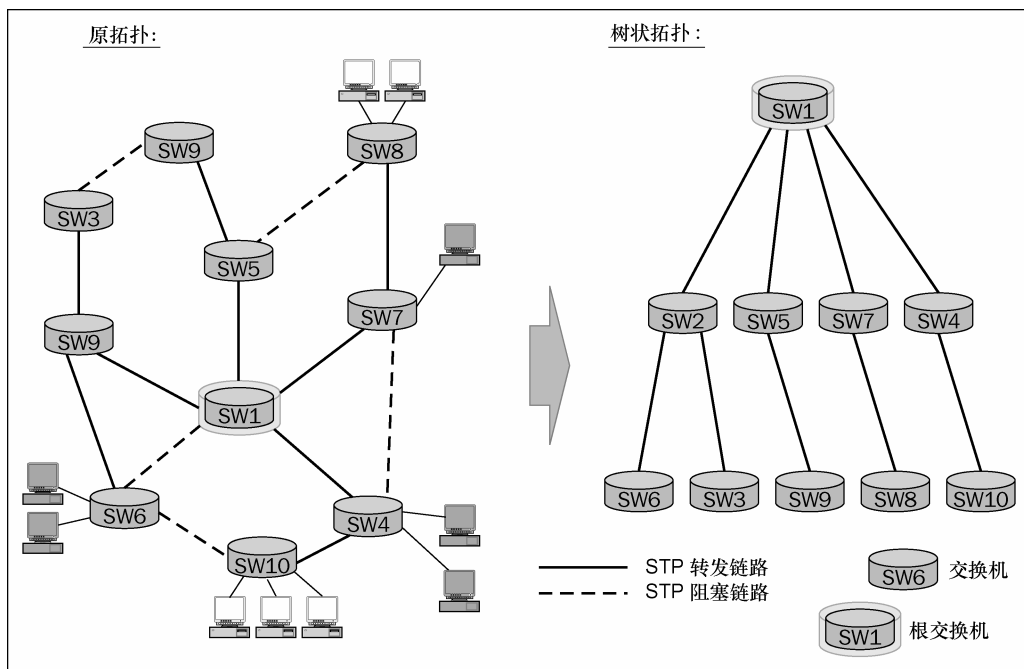


图7.8

运行 STP 的 LAN 交换机之间会以多播方式互发一种叫做 BPDU 的信令帧^①。

由图 7.9 可知, BPDU 的目的 MAC 地址为以太网多播地址, 源 MAC 地址为生成其的 LAN 交换机的 MAC 地址。

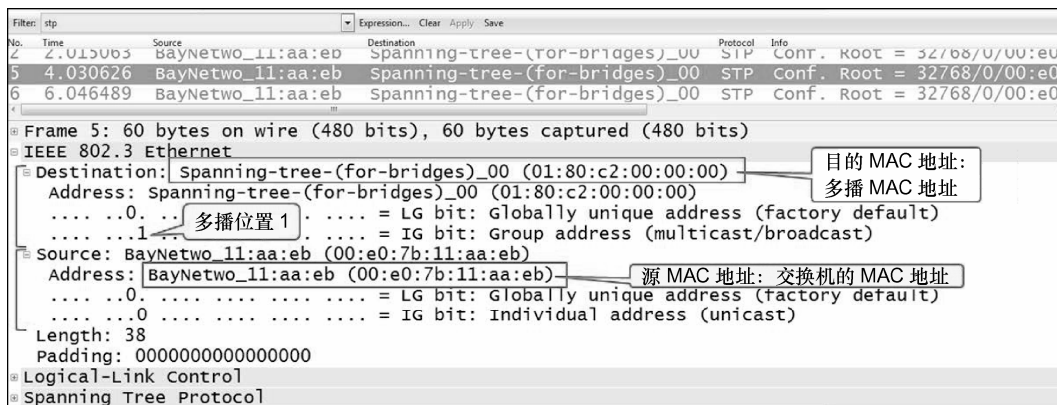


图7.9

① 译者注：原文是“BPDU are update frames that are sent by multicast between the LAN switches”。

BPDU 会封装在 802.3 Ethernet 帧内发送，其格式（即配置 BPDU 的格式）如图 7.10 所示。

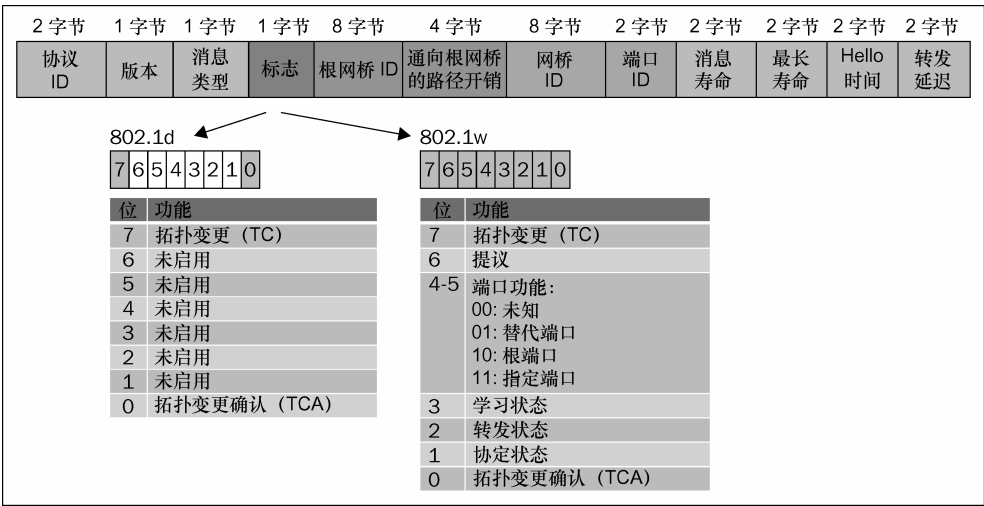


图 7.10

对 BPDU 各个字段的解释请见下表。

字段名	长度 (单位为字节)	描述	值	Wireshark 中用来引用该字段的显示过滤参数
协议 ID	2	协议标识符	始终为 0	stp.protocol
版本	1	STP 的版本	常规 STP = 0 RSTP = 2 MST = 3	stp.version
消息类型	1	BPDU 的类型	常规 STP = 0 RSTP = 2 MST = 2	stp.type
标记	1	协议标记	详见图 7.10	stp.flags
根网桥 ID	8	根网桥标识符，由根网桥优先级和根网桥的硬件（MAC）地址构成	根网桥的优先级值+根网桥的 MAC 地址	stp.root.prio stp.root.ext stp.root.hw
通向根网桥的路径开销	4	将数据帧转发至根网桥的成本	其值由 STP 计算而得。在根网桥发出的 BPDU 中该字段值为 0	stp.root.cost
网桥 ID	8	（发出本 BPDU 的）网桥标识符，由该网桥的优先级值和硬件（MAC）地址构成	（发出本 BPDU 的）网桥的优先级值+网桥的 MAC 地址	stp.bridge.prio stp.bridge.ext stp.bridge.hw

续表

字段名	长度 (单位为字节)	描述	值	Wireshark 中用来引用该字段的显示过滤参数
端口 ID	2	端口标识符	发出(本 BPDU 的)交换机端口的标识符	stp.port
消息寿命	2	根据当前 BPDU “判断”出的由根网桥生成的“原始”BPDU 的寿命	由根网桥生成的每个 BPDU 的消息寿命字段值都是 0，BPDU 只要被其他网桥中继转发一次，消息寿命字段值都会加一	stp.msg_age
最长寿命	2	BPDU (自被根网桥生成之时起,)可在网络中“存活”的最长时间	通常=20	stp.max_age
Hello 时间	2	网桥定期发送 BPDU 的间隔时间	通常=2 秒	stp.hello
转发延迟	2	交换机端口在侦听和学习状态逗留的时间	通常=15 秒	stp.forward

若在网络中运行 MSTP，则相应的 BPDU 帧里势必会包含更多的内容，以承载与 MSTP 有关的参数。

端口的 STP 状态

对运行常规 STP 的交换机而言，其端口会呈以下几种 STP 状态。

- **禁用 (Disabled):** 处于该状态下的交换机端口既不会转发任何数据帧，也不会侦听 BPDU。
- **阻塞 (Blocking):** 处于该状态下的交换机端口不会转发任何数据帧，但会侦听 BPDU。
- **侦听 (Listening):** 处于该状态下的交换机端口只能收、发 BPDU，既不能转发数据帧，也不能获悉其 MAC 地址。
- **学习 (Learning):** 处于该状态的交换机端口虽然不能转发数据帧，但可解析收到的数据帧，且能根据获悉到的 MAC 地址构建 MAC 地址表。
- **转发 (Forwarding):** 处于该状态的交换端口能正常收、发 BPDU，正常构建 MAC 地址表，正常转发数据帧。

将设备接入 LAN 交换机时，交换机端口的 STP 状态会经历以下变迁。

- 从禁用状态变迁至侦听状态，要花 20 秒的时间。

- 从侦听状态变迁至学习状态，要花 15 秒的时间。
- 从学习状态变迁至转发状态，要花 15 秒的时间。

对运行 RSTP/MSTP 的交换机而言，其端口会呈以下几种 STP 状态。

- **丢弃 (Discarding)**: 处于该状态下的交换机端口会丢弃所有数据帧。
- **学习 (Learning)**: 处于该状态的交换机端口不能转发数据帧，但可解析收到的数据帧，且能根据获悉到的 MAC 地址构建 MAC 地址表。
- **转发 (Forwarding)**: 处于该状态的交换端口能正常收、发 BPDU，正常构建 MAC 地址表，正常转发数据帧。

运行 RSTP/MSTP 的 LAN 交换机的端口从丢弃状态过渡到转发状态一般只需短短几秒，具体时长要视 LAN 的拓扑结构和复杂程度而定。

7.3.4 拾遗补缺

排除 STP 故障时，最好是直接登录 LAN 交换机查看其日志。要是在网络中还部署有 SNMP 网管系统，通过观察并分析 LAN 交换机发送的 SNMP trap 信息，也会对故障排除有所帮助。

接下来，将以三个 STP 版本的 BPDU 帧为例，让读者熟悉其某些重要字段。

- 图 7.11 所示为常规 STP 的 BPDU 帧。通过该 BPDU 的源 MAC 地址字段值，可判断出发出该 BPDU 的交换机为一台 Nortel 交换机；通过根网桥 ID 字段值和网桥 ID 字段值(两个字段里包含的 MAC 地址一模一样)，可判断出那台 Nortel 交换机就是 LAN 中的根交换机；通过端口 ID 字段值 (0x8003)，可判断出是那台 Nortel 交换机上编号为 3 的端口发出了该 BPDU。

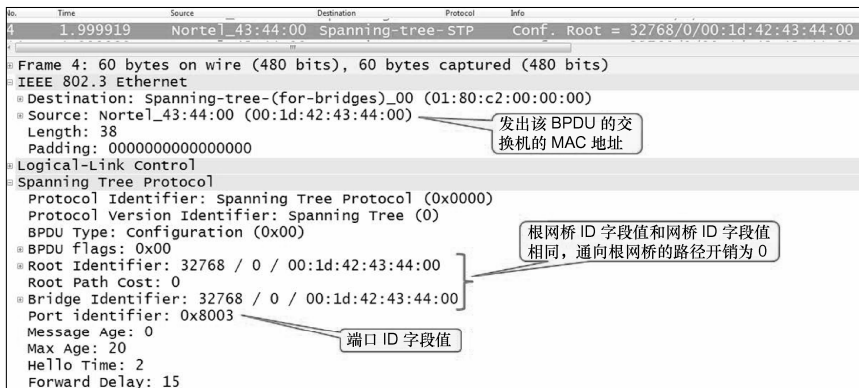


图 7.11

- 图 7.12 所示为 RSTP 的 BPDU 帧。通过协议 ID 字段值 (2)，可判断出该 BPDU 由运行 RSTP 的交换机发出；通过标记字段值 (0x3c，即 2、3、4、5 位置 1)，可判断出该发出该 BPDU 的交换机端口为 STP 指定端口。

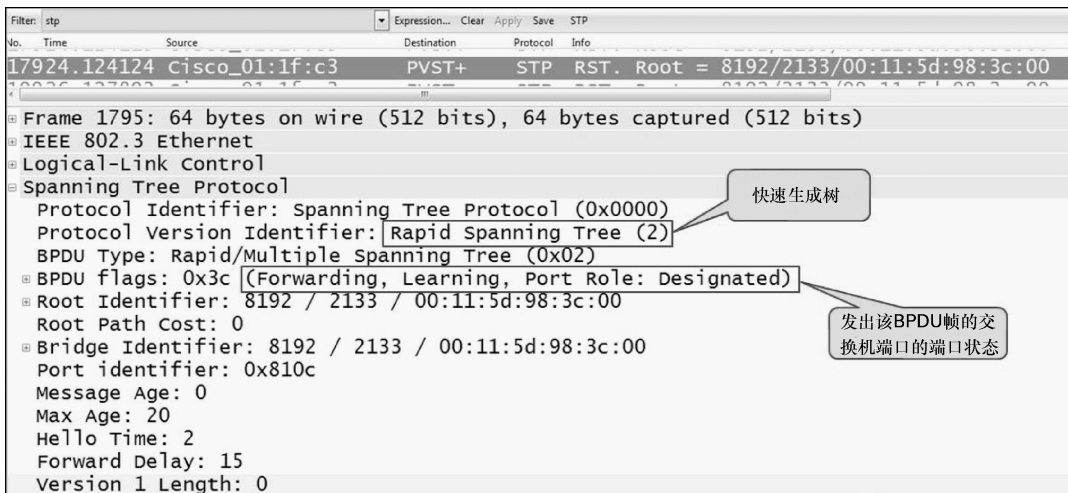


图 7.12

- 图 7.13 所示为 MSTP 的 BPDU 帧。由图可知，在标准的 BPDU 之后还附着了 MST 扩展信息。

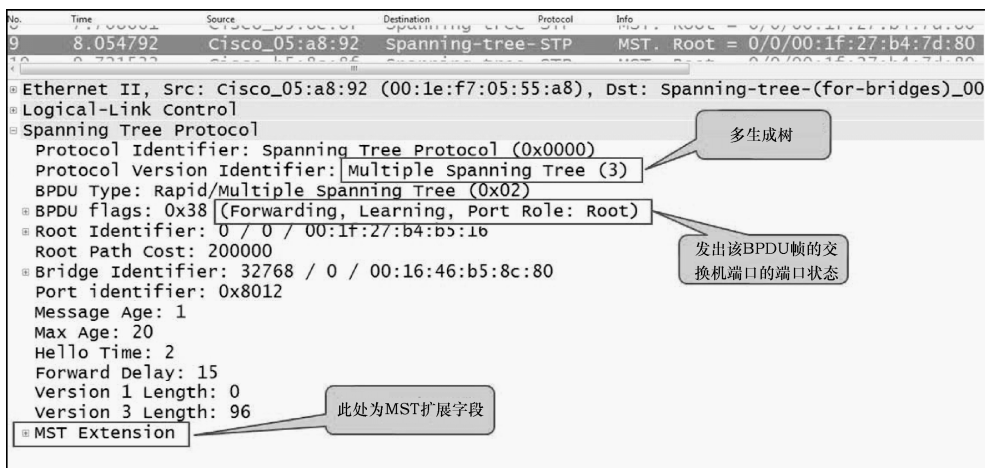


图 7.13

7.4 VLAN和VLAN tagging故障分析

VLAN 或虚拟（Virtual）LAN 是一项以虚拟的方式分割一台物理交换机的技术，其目的是要用一台物理交换机“组建”多个相互隔离的 LAN，虚拟 LAN 中的“虚拟”一词也正是来源于此^①。本节会介绍 VLAN 流量的监控方法。

本节的目的是要让读者掌握如何通过 Wireshark，来分析与 VLAN 有关故障。当然，解决相关故障的最直接的方法则是：登录交换机，执行相关排障或修复命令。

7.4.1 准备工作

VLAN 流量的监控分为以下两种：

- 如何监控在某个 VLAN 内传播的流量；
- 如何监控通过 Trunk 端口传播的带 VLAN 标记的流量。

监控在某个 VLAN 内传播的流量非常简单，只要稍作配置即可实现。现在来重点谈一谈如何监控带 VLAN 标记的流量。

使用 Wireshark 抓取通过 Trunk 端口传播的流量时，未见得能够看到数据包中的 VLAN 标记。Wireshark 能否显示出数据包的 VLAN 标记，则要取决于安装 Wireshark 的操作系统、实际用来抓包的网卡（NIC）以及网卡驱动程序。



注意

操作系统和网卡是否支持接收带 VLAN 标记的数据包，要完全取决于操作系统开发商和网卡芯片制造商。有关详情请查阅操作系统/网卡使用手册，或执行 Google 搜索。

图 7.15 所示的网络由好几个 VLAN 组成，其拓扑结构也颇为典型。上面一台核心层交换机（SW1）分别通过一条 Trunk 链路（所谓 Trunk 链路，是指用来传递带 VLAN 标记的以太网帧的链路）与下面两台接入层交换机（SW2 和 SW3）相连。该网络由 VLAN 10、20、30 这三个 VLAN 组成，各个 VLAN 的主机之间不能彼此通信。

7.4.2 操作方法

以正确的方法把 Wireshark 主机与图 7.14 中所示交换机相连。

监控在某个 VLAN 内传播的流量

1. 把 Wireshark 主机与核心层交换机上的某个端口相连。

^① 译者注：原文是“VLAN, or Virtual LAN, is a mechanism that divides a LAN into separate LANs without any connectivity between them, and this is where the name virtual comes from”。译者觉得作者给 VLAN 下的定义很是搞笑，译文未按原文翻译。

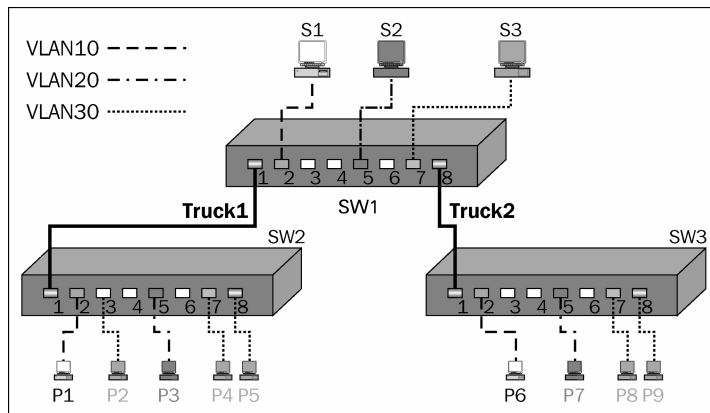


图7.14

2. 在 SW1 上配置端口镜像, 把受监控 VLAN 的流量重定向给连接了 Wireshark 主机的端口。试举一例, 要想监控 VLAN 10 内传播的流量, 而 Wireshark 主机连接的是 SW1 上的端口 4, 那么在 Cisco 交换机上应执行如下命令。

```
Switch(config)#monitor session 1 source vlan 10
Switch(config)#monitor session 1 destination interface fastethernet 0/4
```

上述命令一配, Wireshark 便可以抓到由 SW1 转发的 VLAN 10 的流量了。



注意

各交换机厂商都有自己的一套配置端口镜像的命令语法, 请登录它们的官网输入以下关键字进行搜索: SPAN (Cisco); port mirror 或 port mirroring (HP、DELL、Juniper 及其他厂商)。在刀片服务器机箱里执行流量监控时, 一般只能监控到某个物理端口的流量; 借助于某些软件(比如, Cisco Nexus 1000V 软交换机), 可以监控到刀片服务器机箱里特定服务器的流量。

监控通过 Trunk 端口传播的带 VLAN 标记的流量

监控通过 Trunk 端口传播的带 VLAN 标记的流量会麻烦点儿。麻烦出在实际用来抓包的网卡以及网卡驱动程序对待 VLAN 标记的态度上。

可按下列步骤来验证 Wireshark 主机所配备的网卡是否支持抓取带 VLAN 标记的数据包。

1. 在交换机上开启端口镜像功能, 让 Wireshark 主机直接抓取重定向而来的 Trunk 端口的数据包, 看看是否能看到 VLAN 标记。若是, 便说明 Wireshark 主机所配备的网卡支持抓取带 VLAN 标记的数据包。
2. 若否, 则需要配置 Wireshark 主机所配备的网卡。假设 Wireshark 主机的操作系统为 Windows XP, 请点击“开始”→“设置”→“网络连接”→“本地连接”, 在弹出的

“本地连接 状态” 窗口中按图 7.15 所示步骤行事。

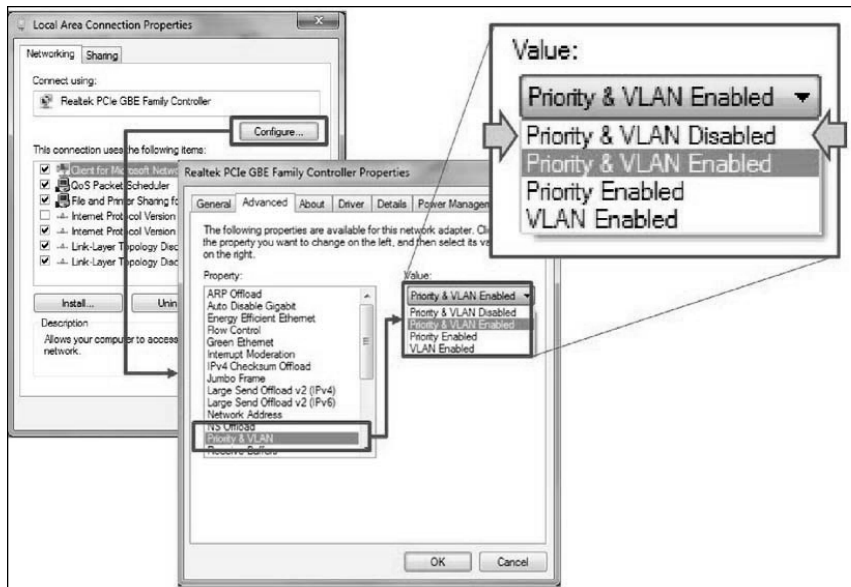


图 7.15

3. 选择了 **Priority & VLAN Disabled** 之后，再点“确定”，网卡就会把数据包中的 VLAN 标记传递给 WinPcap 驱动程序和 Wireshark 软件了。



注意

图 7.15 是以配置一台联想（Lenovo）笔记本自带的 Realtek 网卡来举例，机器不同、网卡不同，配置方法也必不相同，但配置原理必然相同。其原理是，在网卡上禁用数据包 VLAN 标记剔除功能，让网卡把带 VLAN 标记的数据包原封不动地传递给 WinPcap 驱动。如此一来，通过 Wireshark 软件，就能看见数据包（以太网帧头中）的 VLAN 标记了。

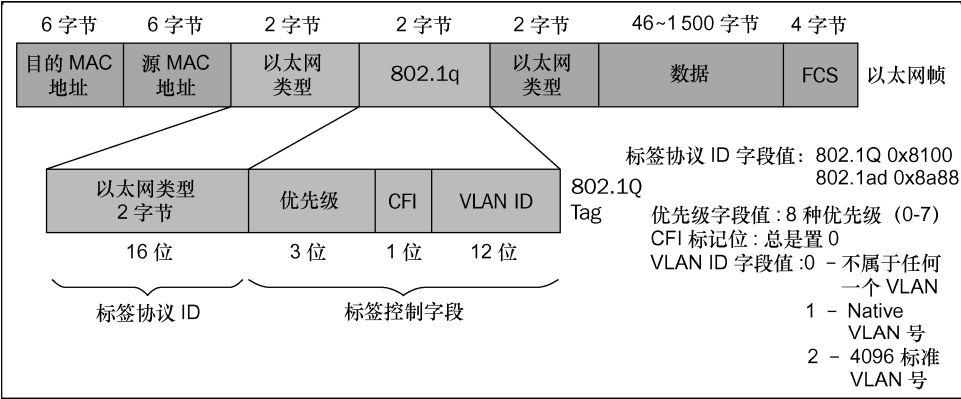
7.4.3 幕后原理

所谓 VLAN 标记，是数据帧帧头内的一块 4 字节的数据，其中记录了该帧所归属的 VLAN ID 以及其他信息。VLAN 标记的格式如图 7.16 所示。

收到数据包之后，大多数网卡及其驱动程序都会原封不动地交给高层处理。只要在配备了这些网卡的主机上安装 Wireshark 软件，在捕获的数据包中必将包括 VLAN ID（见图 7.17）。

但还有一些具备复杂功能的网卡（比如，Intel 和 Broadcom 吉比特芯片组的网卡），在保留默认配置的情况下，能自身消化掉数据包中的 VLAN ID。此时，要想让 VLAN ID 在 Wireshark

捕获的数据包中露面，就必须更改此类网卡的默认配置，禁用其消化 VLAN ID 的功能。



给以太网帧打上 VLAN 标记的方法

图 7.16

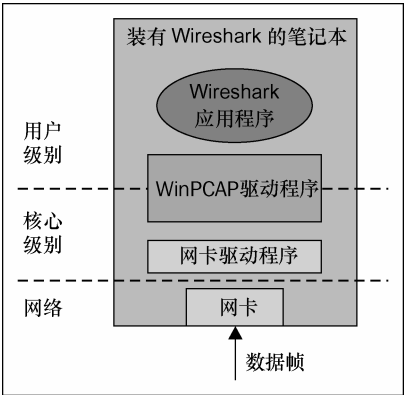


图 7.17

图 7.18 所示为一个打了 VLAN 标记的以太网帧，该帧携带的 VLAN 标记为 20 (VLAN ID=20)。

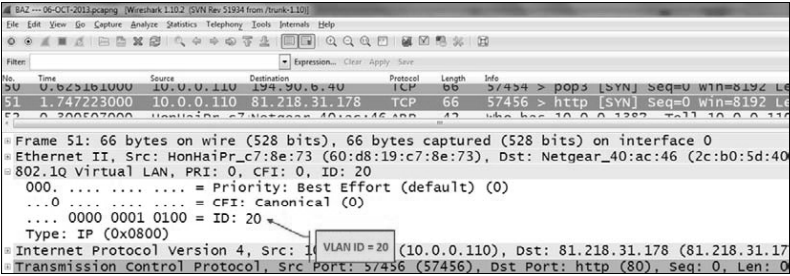


图 7.18

7.4.4 拾遗补缺

有时，可能会抓到携带双 VLAN 标记的以太网帧，这符合 IEEE 802.1ad 标准。此类以太网帧中的第一个 VLAN 标记叫做 S 标记（S-Tag）（IEEE 802.1ad），由服务提供商的边界设备所打；第二个 VLAN 标记名为 C 标记（C-Tag）（IEEE 802.1Q），由（服务提供商的）客户的设备所打。这一给以太网帧打两个以太网标记的机制也被称为 Q-in-Q 机制。

7.4.5 进阶阅读

- 欲知更多与 WinPcap 有关的信息，请访问 WinPcap 主页：<http://www.winpcap.org/>。
- 欲知更多与 UNIX/Linux tcpdump 库有关的信息，请访问 tcpdump 主页：<http://www.tcpdump.org/>。

7.5 无线 LAN (WiFi) 故障分析

近十年以来，无线 LAN 越来越受人们的欢迎，其标准也从老的 802.11b 发展到了 802.11g，再到最新的支持高速无线通信的 802.11n。

诸如 IEEE 802.11ac 之类新兴的无线 LAN 标准，以及符合这些标准的产品也已问世在即。此外，无线 LAN 的标准还有尚处于制定中的 802.11ad。

本节将介绍如何解决无线 LAN 故障，同时会重点关注如何使用 Wireshark 抓取 WiFi 帧，以执行基本的流量分析。

7.5.1 准备工作

当有用户因无线网络慢而进行投诉时，应第一时间携带配有无线网卡的笔记本赶赴用户所在位置（离用户使用无线网络的地方越近越好），并确保无线网卡已激活可用。

7.5.2 操作方法

图 7.19 所示为 Windows 操作系统里用来定位 WiFi 故障的标配工具，通过该工具可以获悉无线网络的以下特征。

- 信号强度：通过 Win 7 操作系统自带的 **Received Signal Strength Indicator (RSSI)** 工具，可了解到无线网络的信号强度。该工具还能显示出无线信号质量的好坏（比如，图 7.19 中的 Good），有时则会显示出具体的 dBm 数。
- 无线访问点 ID，即 **SSID (Service Set Identification)**。

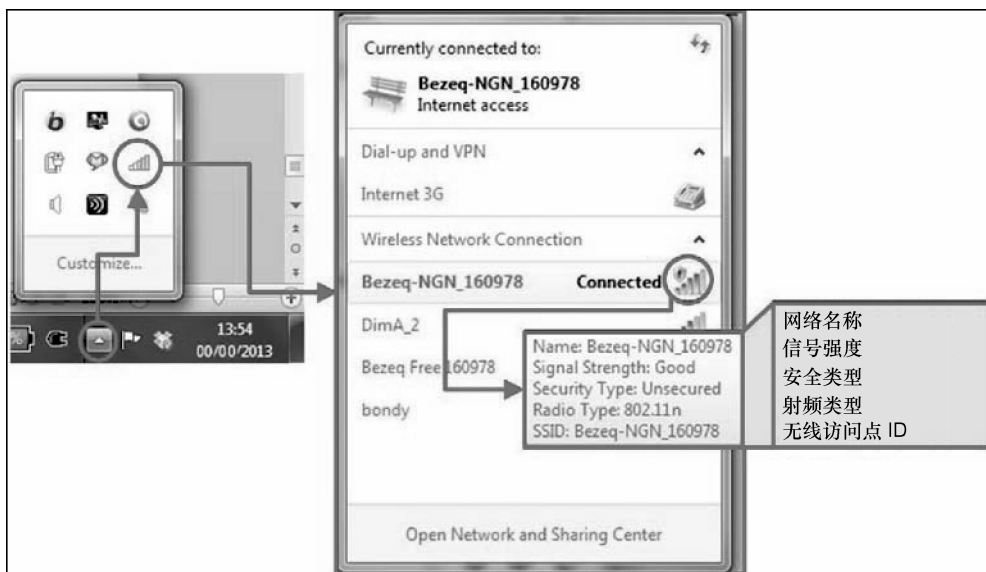


图7.19

- 无线网络所使用的安全协议。
- 射频类型（图 7.19 所示 WiFi 连接显示其接入的是 802.11n 无线网络）

还可以使用专门的软件（大多都是免费软件，其中有部分是由笔记本电脑厂商所开发），来搜索 WiFi 网络及其工作频率。图 7.20 所示为一款名为 WiFi Locator 的软件（<http://tcpmonitor.altervista.org/>）搜索到的无线网络列表；当然，有许多其他软件也具备无线网络搜索功能。

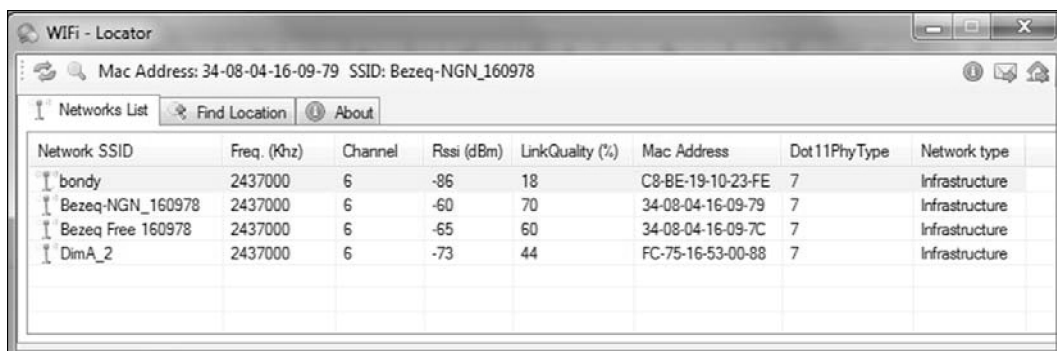


图7.20

请注意出现在图 7.20 中的 Rssi 一栏下的数字，数字越高，就表示与之相对应的无线网络信号越强。

- **-60dBm 及以上**：表示无信号质量上佳。
- **-80dBm 到-60dBm**：表示无线信号质量尚可。
- **-80 dBm 到-90dBm**：表示无线信号较弱。
- **-90 dBm 及以下**：表示无线信号极弱。

要是无线信号质量在中等以上，但用户仍旧不停地投诉，那就应该看看是否存在频率干扰或其他射频问题了。



注意

来介绍一下作者设计无线网络的经验：要让无线网络承载标准的企业级应用程序流量，信号强度不能低于-75dBm；要想跑 VoIP 流量的话，信号强度至少应在-65dBm 以上。

要检查是否存在信号干扰问题，需安装特殊软件，来监控在特定周期内的无线网络信号强度。有一款名叫 **inSSIDer** 的软件不但能监控指定周期内具体无线访问点的信号强弱，还能图形的方式精确加以呈现。图 7.21 中所示就出自这款软件^①。



图 7.21

^① 译者注：原文从此处开始上下不接，特此申明。

1. 检查是否存在以下问题。

- 在同一区域内是否有不同的 AP 运行于同一信道。
- 所在区域信号强度是否不足（RSSI 值低于-90dBm）。

2. 使用频谱分析仪来检查本区域是否存在信号干扰问题，可能的干扰源包括：机场、港口及驻军等。Fluke Networks、Agilent 以及 Anritsu 都是主流的频谱分析仪生产厂商。

3. 用 Wireshark 抓取并分析 WiFi 控制帧。首先，应检查 AP 是否能正常广播 beacon（信标）帧。图 7.22 所示为 Wireshark 抓取的 beacon 帧的帧结构。

为了发现附近的 AP，（配有无线网卡的）移动工作站会发出 beacon probe request（探测请求）帧。

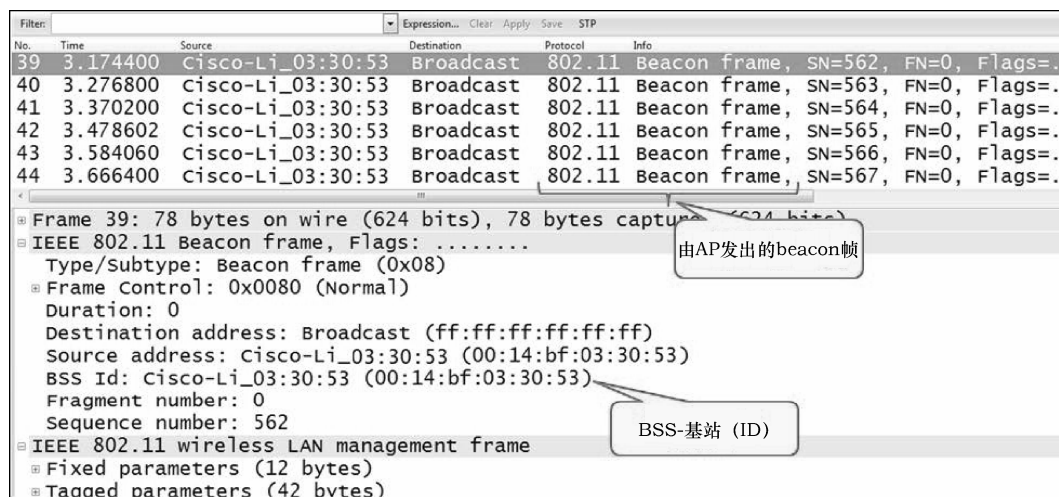


图7.22



注意

欲从另一 WiFi 设备获取信息时（比如，为确定信号范围内的无线接入点时），移动工作站会发出 probe request 帧^①。

为了向 AP 注册，移动工作站可确认由 AP 发出的 beacon 帧。

4. 接收并确认 beacon 帧之后，移动工作站将通过标准的 DHCP 过程获取其 IP 地址相关参数；与 DHCP 有关的内容详见本书第 8 章。

^① 译者注：原文是“A probe request frame is sent by a station when it wants to obtain information from another Wi-Fi device, for example, to determine the access points within the range”。译文按原文字面意思直译。



注意

AP 会定期发出 beacon 帧，并在其中包含与其所提供的无线网络有关的信息，以宣告该无线网络的存在。信息包括：时间戳、SSID 以及与 AP 有关的其他参数。移动工作站配备的无线网卡也会持续扫描所有 802.11 射频信道，侦听 beacon 帧，以选择最佳接入点，并与之关联。

7.5.3 幕后原理

无线 LAN 的标准由 IEEE 802.11 委员会负责制定。最早的无线 LAN 标准是 802.11a，历经 802.11b、802.11g、802.11n，直到最新的支持高带宽的 802.11ac 和 802.11ad。

如图 7.23 所示，无线 LAN 中最主要的网络设备是 AP，以及与之相连的无线客户端。

802.11n 是如今最为常见的无线 LAN 标准，采用先进的多进多出（**Multiple Input Multiple Output, MIMO**）调制方法，通过多天线技术（最多可支持 4 根）以及某些其他技术来增加带宽。

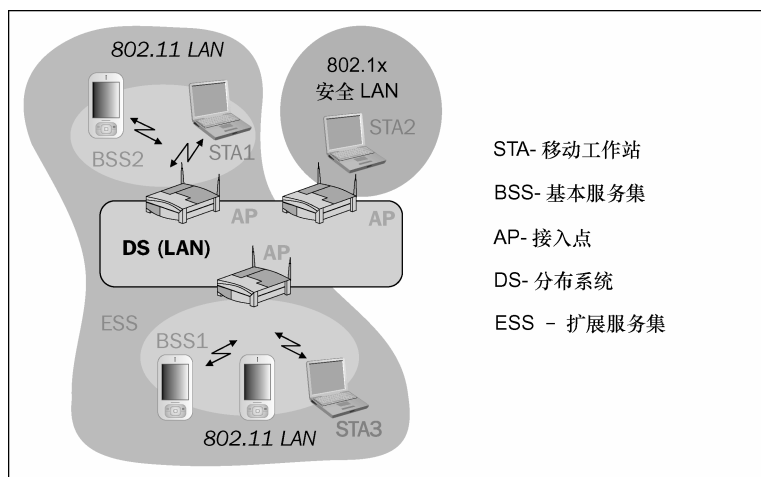


图7.23

第 8 章

ARP和IP故障分析

本章涵盖以下主题：

- ▶ 与 ARP 有关的连通性故障分析；
- ▶ IP 流量分析工具的用法；
- ▶ 用 GeoIP 工具去查询 IP 地址的归属地；
- ▶ 路由选择故障分析；
- ▶ 发现 IP 地址冲突；
- ▶ DHCP 故障分析。

8.1 简介

本章将介绍如何用 Wireshark 分析 IP 和 ARP 协议，探讨如何解决 IP、ARP、DHCP 以及路由选择等故障，同时会分析在排除上述协议及路由选择故障时所面临的问题^①。首先，将介绍上述协议在运作时的正常举动。接下来，会讲解这些协议可能会出故障的地方，以及如何排除故障。

一般而言，在分析网络故障时，思路应自下而上。譬如，在出现网络中断的情况下，应考虑按以下思路排除故障^②。

① 译者注：原文是 “In this chapter we will learn how to analyze Layer 3 (IP) and Layer 3 to Layer 2 resolution (ARP). We will discuss the Internet Protocol (IP), Address Resolution Protocol (ARP), Dynamic Host Configuration Protocol (DHCP), routing issues and others, and the problems that you might face while troubleshooting these protocols.” 原文让人不寒而栗，译文酌改。

② 译者注：原文是 “In general, when we analyze a network problem, we will go bottom up: if you cannot get connectivity, look for the problem in the following order” 。

1. **物理层 (L1)**: 检查电缆的连接情况、交换机上接口 LED 灯的状态、PC 是否开机等。该步骤需要网管人员亲力亲为。
2. **数据链路层 (L2)**: 检查 ARP 解析是否正常。若某远程节点无法访问, 请检查流量转发沿途中每一对直连节点/路由器间的 L2 连通性。该步骤需要通过命令行工具来完成 (具体内容详见 8.2 节)。
3. **网络层 (L3)**: 先 ping 目的节点的 IP 地址, 若 ping 不通, 再执行 traceroute 操作。该步骤需要通过命令行工具来完成, 有时可能需要借助于 Wireshark (去抓取 ping 或 traceroute 操作生成的流量)。
4. **传输层 (L4)**: 检查运行于对端 (远程) 节点上的进程/服务能否响应, 执行该步骤时, 需借助于 Wireshark。
5. **应用层 (L5-7)**: 检查 (安装在远程节点上的) 应用程序是否有问题。

第 7 章已经介绍了如何排除第一、二层 (L1、L2) 网络故障。本章会探讨如何排除第三层 (IP) 以及二、三层之间的网络故障 (ARP 协议故障)。第 9 章会探究如何排除传输层 (UDP/TCP) 网络故障。第 10 章到第 14 章则会介绍如何排除涉及具体应用程序的网络故障。

8.2 与 ARP 有关的连通性网络故障分析

以太网内的 IP 主机之间要想互相通信, 除了要知道对方主机的 IP 地址 (目的 IP 地址) 以外, 还得获悉其 MAC 地址, ARP 的用途就是根据已知目的主机的 IP 地址, 解析出与其对应的 MAC 地址^①。也就是说, 在目的主机的 IP 地址已知、MAC 地址未知的情况下, 要想向其发送 IP 数据包, 得先行发送 ARP 请求数据包, 以求解析出目的主机的 MAC 地址。收到目的主机发出的 ARP 响应数据包, 获知其 MAC 地址之后, 才能向其发送 IP 数据包 (原因很简单: IP 数据包要封装进第二层以太网帧才能在以太网内传送)。



注意

ARP 数据包的目的 MAC 地址为广播地址, 因此 ARP 协议的“作用域”为本地 LAN。要想向本 IP 子网内 (MAC 地址未知、IP 地址已知) 的主机发送 IP 数据包, 就得先行发送 ARP 请求数据包, 解析出相应主机的 MAC 地址。要想向本 IP 子网之外的主机发送 IP 数据包, 也得先行发送 ARP 请求数据包, 解析出本子网中默认网关 (出口路由器) 的 MAC 地址。

^① 译者注: 原文是 “ARP is used by IP to resolve the destination MAC address out of the IP address of the device that we wish to communicate with”。作者废话一大堆, 还扯不到点子上, 译文酌改。

8.2.1 准备工作

可使用以下三种途径来排查基本的连通性故障。

- 通过标准的命令行工具（对于 Windows XP 系统，可点击“开始”→“运行”，在“运行”窗口的输入栏内输入 `cmd`，再点“确定”；对 UNIX/Linux 系统，可使用任何一种可用的 Shell）。
- 通过 Wireshark 抓包。

登录进 LAN 交换机或路由器，执行各种 `show` 命令（Cisco 设备），来了解相关信息。



注意

管理 LAN 交换机、路由器或其他网络设备的方法很多，既可以使用控制电缆（Cisco 设备使用的是一种蓝色的扁平电缆），也可以先为它们配置一个 IP 地址，然后通过 Telnet、HTTP 或 SNMP 来管理。通过以上各种方法，都可以了解到网络设备的状态信息、流量信息以及报错信息。

8.2.2 操作方法

本节将会举几个连通性故障的例子，同时会讲解具体的排障思路。请看图 8.1 所示的网络，该网络的拓扑结构既典型而又简单。现在，就来说一说此类网络中可能会发生的连通性故障。

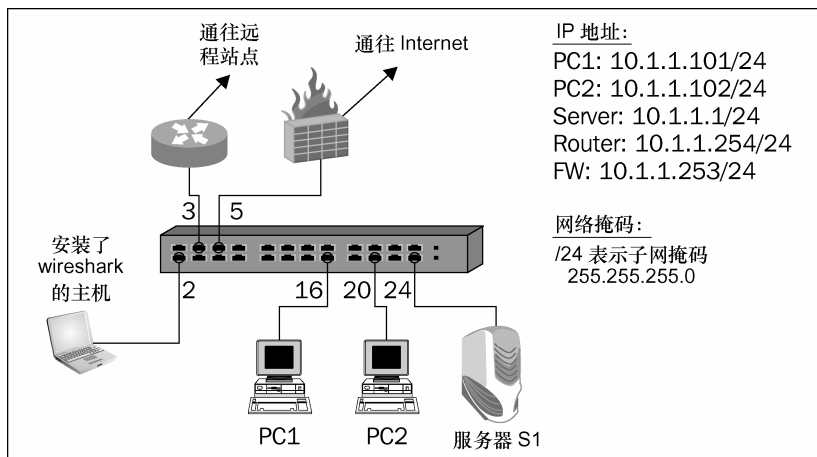


图8.1

由图 8.1 可知，网络中那台交换机的端口 16 和 20 各连接了一台 PC（PC1 和 PC2）；端口 24 连接了一台服务器（S1）；端口 3 连接了一台路由器，该路由器（通过 WAN 链路）连接到了远程站点；端口 5 连接了一台防火墙，这台防火墙把整个网络与 Internet 连接在了一起；那

台安装了 Wireshark 的主机则与端口 2 相连。

让我们来探讨一下在图中所示网络中可能会发生的故障，以及解决故障的方法。

故障场景 1：网络中的所有 PC 都不能访问服务器（S1）。

1. 在 PC1 和 PC2 上 ping S1。
2. 若 ping 不通，请进入命令行窗口，输入命令 `ARP-a`，显示 PC1 和 PC2 的 ARP 缓存表，看看能不能发现 S1 的 IP 地址和 MAC 地址。
3. 如能发现，则需登录 S1，检查其是否安装并启用了软件防火墙。软件防火墙很可能会阻挡 ping 包（ICMP request 数据包）。在 S1 上关停防火墙软件，到 PC1 和 PC2 上再 ping S1。



注意

安装于主机上的防火墙、VPN 客户端或杀毒软件都有可能阻挡 ICMP request 数据包。当诊断网络连通性问题时，千万别忘了这茬。

4. 如能 ping 通，但应用软件仍不能正常使用，请参阅第 9 章。
5. 如 ping 不通，请登录 LAN 交换机，执行相关命令，显示其 MAC 地址表，验证交换机能否从正确的端口学得 PC1、PC2 和 S1 的 MAC 地址。



注意

只要是可网管交换机，一般都有显示 MAC 地址表的命令。MAC 地址表所含信息为交换机上各个端口学到的 MAC 地址的记录。

6. 若在 PC1 和 PC2 的 ARP 缓存表中看不到 S1 的 IP 地址和 MAC 地址，请检查是不是物理层故障（比如，检查网线是不是没做好、交换机端口或主机网卡是否有问题）。此时，解决故障的手段包括：更换网线、更换交换机上连接 PC 和服务器的端口，或换一台交换机。

接下来，会介绍如何借助 Wireshark 来解决此类连通性故障。

7. 当有人反映连不上（隶属于同一 LAN 的）某特定的主机（服务器、打印机）时，若借助于端口镜像功能，用 Wireshark 抓取此人使用的主机收发的流量，势必会看见该主机发出了诸多 ARP 请求数据包，但却收不到相关主机（服务器、打印机）回复的任何 ARP 应答数据包^①。图 8.2 所示为出现上述情形时，使用 Wireshark 抓取到的数据包。

^① 译者注：原文是 “When clients complain about connectivity problems to a specific device (server, printer, and so on), you connect Wireshark to the port (with port mirror), and you see many ARP requests and no answer”。

No.	Time	Source	Destination	Protocol	Info
329	443.349	WistronI_ae:77:69	Broadcast	ARP	172.30.116.100 正尝试解析 172.30.116.254 的 MAC 地址
330	444.547	172.30.116.100	239.255.255.250	SSDP	M-SEARCH * HTTP/1.1
331	444.890	WistronI_ae:77:69	Broadcast	ARP	who has 172.30.116.254? Tell 172.30.116.100
332	445.545	WistronI_ae:77:69	Broadcast	ARP	who has 172.30.116.254? Tell 172.30.116.100
333	445.659	ThomsonT_66:84:7d	Broadcast	ARP	who has 172.30.116.254? Tell 172.30.116.253
334	445.659	ThomsonT_66:84:7d	Broadcast	ARP	who has 172.30.116.1? Tell 172.30.116.253
335	445.659	G-ProCom_24:39:eb	ThomsonT_66:84:7d	ARP	172.30.1 172.30.116.253 正尝试解析 172.30.116.254 的 MAC 地址
336	445.660	ThomsonT_66:84:7d	Broadcast	ARP	who has 172.30.116.254 的 MAC 地址

图8.2

8. 由图 8.2 可知, IP 地址为 **172.30.116.100** 和 **172.30.116.253** 的主机发出了 ARP 请求数据包, 尝试解析 IP 主机 **172.30.116.254** 的 MAC 地址, 但未收到任何答复。
9. 此时, 需登录 LAN 交换机, 检查其是否学到了 IP 主机 **172.30.116.254** 的 MAC 地址。若未学到, 请检查是否为物理层故障^①。

ARP 中毒和中间人 (Man-in-the-Middle) 攻击

攻击者用其主机网卡的 MAC 地址, 来污染受攻击主机的 ARP 缓存, 是中间人攻击的一种手段。ARP 缓存一旦受到污染, 每台受攻击主机只要与其他设备通信, 其所有流量便会被攻击主机所截取。攻击主机在读取流量之后, 还可以奉还给实际的目的主机。

之所以把上述攻击手段归为中间人攻击, 是因为攻击设备位居任意两台受攻击设备的通信路径之间。此外, 由于发动攻击的主机能用错误信息来污染受攻击主机的 ARP 缓存, 因此也有人把这种攻击手段称为 ARP 中毒。

图 8.3 所示为这一中间人攻击的示例。

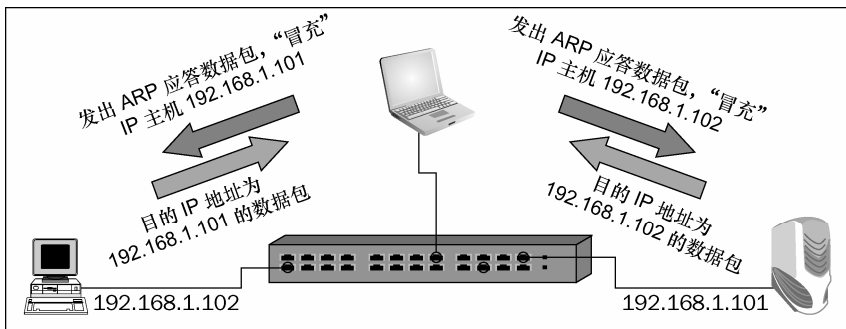


图8.3

^① 译者注: 原文是 “In this case, check in the switch if the MAC address has been learned, and if not, check for physical problems”。作者啊, 你登录进交换机查看 MAC 地址表有用吗? 不先查看 ARP 表, 你知道 IP 主机 172.30.116.254 的 MAC 地址是什么吗?

图 8.4 所示为有人发动中间人攻击时，Wireshark 所抓到的 ARP 应答数据包。

No.	Time	Source	Destination	Protocol	Info
72	13.707605	WistronI_ae:77:69	AsustekC_37:3d:84	ARP	10.0.0.101 is at f0:de:f1:ae:77:69
89	12.280901	WistronI_ae:77:69	AsustekC_6d:94:d3	ARP	10.0.0.100 is at f0:de:f1:ae:77:69
145	30.496627	WistronI_ae:77:69	AsustekC_6d:94:d3	ARP	10.0.0.100 is at f0:de:f1:ae:77:69
156	8.501596	WistronI_ae:77:69	AsustekC_37:3d:84	ARP	10.0.0.101 is at f0:de:f1:ae:77:69
175	9.575341	WistronI_ae:77:69	AsustekC_6d:94:d3	ARP	10.0.0.100 is at f0:de:f1:ae:77:69
190	6.928713	WistronI_ae:77:69	AsustekC_6d:94:d3	ARP	10.0.0.100 is at f0:de:f1:ae:77:69

图8.4

免费 ARP

当一主机欲验证其 IP 地址是否与其他主机冲突时，便会发出一种特殊的 ARP 数据包——免费 ARP 数据包。在这种 ARP 数据包中，发送方 IP 地址字段值与目标方 IP 地址字段值相同（见图 8.5）。网络中存在免费 ARP 数据包，并不表示发生了异常情况。有些设备会主动发出免费 ARP 数据包，比如，某些厂商的家用宽带路由器会定期发送免费 ARP 数据包，目的是要刷新内网主机的 ARP 缓存，使其避免遭受 ARP 欺骗攻击。

No.	Time	Source	Destination	Protocol	Info
173	0.000000	AsustekC_6d:94:d3	Broadcast	ARP	who has 10.0.0.7? Tell 10.0.0.1
925	7.271417	D-LinkIn_f4:7b:a2	HonHaiPr_c7:8e:7	ARP	who has 10.0.0.6? Tell 10.0.0.138
926	0.000020	HonHaiPr_c7:8e:73	D-LinkIn_f4:7b:a	ARP	10.0.0.6 is at 60:d8:19:c7:8e:73

Frame 173: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: AsustekC_6d:94:d3 (14:da:e9:6d:94:d3), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
Hardware type: Ethernet (1)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (1)
Sender MAC address: AsustekC_6d:94:d3 (14:da:e9:6d:94:d3)
Sender IP address: 10.0.0.1 (10.0.0.1)
Target MAC address: 00:00:00_00:00:00 (00:00:00:00:00:00)
Target IP address: 10.0.0.1 (10.0.0.1)

图8.5

ARP 扫描 (ARP sweep)

ARP 扫描是指某台主机以发出各种 ARP 数据包的方式，来扫描整个网络，意在窃取信息或发动攻击。

当面临 ARP 扫描时，需关注：

- ARP 数据包的源 MAC 地址是什么？ARP 数据包的具体类型是什么，是 ARP 请求还是 ARP 应答数据包？

- ARP 数据包的数量级（单位为数据包/秒）是多少？

ARP 数据包的源 MAC 地址是什么？是 ARP 请求还是 ARP 应答数据包？

LAN 中 IP 设备的正常运行离不开 ARP 请求和 ARP 应答数据包。以下所列为作者对这两种 ARP 数据包的某些看法。

- 若 ARP 请求数据包源于多个不同的 MAC 地址（即源 MAC 地址多有不同），则属于正常情况——ARP 请求数据包是 LAN 中 IP 设备间相互通信的支柱，但前提是数目不能过多。
- 若 ARP 请求数据包大都源自于单一 MAC 地址，则要验证拥有该 MAC 的网络设备到底是谁：
 - 它可能是一台正在自动探测网络的网管设备；
 - 它可能是一台发出免费 ARP 请求数据包的路由器（如图 8.5 所示）；
 - 要是无法确定这个 MAC 地址归谁所有，则网络很有可能正面临着蠕虫病毒或 ARP 中毒攻击。请务必仔细查找原因！
- 若有莫名其妙的 ARP 应答数据包现身于网络（即 ARP 应答数据包并非是由“配套”的 ARP 请求数据包所触发），则同样需要仔细查找原因。

ARP 数据包的数量级

在 LAN 中，IP 主机之间在首次交流之前，会先发 ARP 数据包，来获知对方的 MAC 地址（详情请见下一节）。精确测算一个网络每分钟会产生多少个 ARP 数据包非常困难，但可按照以下思路来估计个大概。

- 每台 IP 设备每分钟生成 1~2 个 ARP 数据包纯属合理。比方说，若网络中的设备数为 100，则每分钟诞生 200~300 个或每秒钟诞生 3~4 个 ARP 数据包是合理的。
- 即便每秒钟诞生的 ARP 数据包略多于 3~4 个（比如，5~10 个），也未必就有问题，但需要仔细查找原因。



注意

网络中 ARP 数据包的数量取决于网络设备实际的用途和具体的行为。若网络中的所有主机只连接一台网络设备（比如，都通过一台路由器连接到外部世界），则 ARP 数据包的数量一定不会太多。然而，若有网络设备向其所有邻居设备定期发送消息，则势必会诞生许多 ARP 数据包。无论如何，只要觉得情况有异，就应该仔细查找原因——细节决定成败。

8.2.3 幕后原理

在 LAN 中，穿行于主机间的 IP 数据包都会封装在以太网帧之内（见图 8.6）。

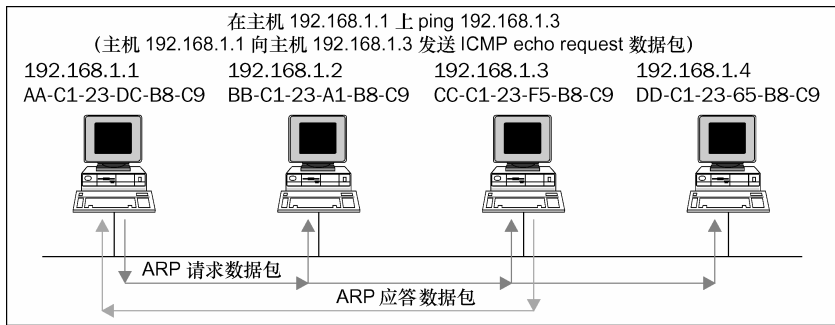


图8.6

举一个 ping 同一 LAN 内 IP 主机的例子：在主机 192.168.1.1 上 ping 192.168.1.3。要想 ping 通 192.168.1.3（即成功向其发送 ICMP echo request 数据包），主机 192.168.1.1 还需要知道与目的 IP 地址 192.168.1.3 相对应的目的 MAC 地址。

为此，主机 192.168.1.1 会以以太网广播方式，发出 ARP 请求数据包，向 LAN 内的所有主机询问：谁设有 IP 地址 192.168.1.3。只要设有这一 IP 地址的主机在线，该主机便会回复 ARP 应答数据包，其中会包含自己的 MAC 地址。

于是，主机 192.168.1.1 便在其内存的 ARP 缓存表 (cache) 里记录下了 IP 地址 192.168.1.3 与这一 MAC 地址之间的对应关系。自此以后，再向 192.168.1.3 发送 IP 包时，主机 192.168.1.1 便不用执行上述 ARP 解析操作了。

对于入驻 ARP 缓存表的每一条记录（前面提到的 IP 地址 192.168.1.3 与 ARP 应答数据包中所包含的 MAC 地址之间的对应关系，便算作 IP 主机 192.168.1.1 的 ARP 缓存表里的一条记录），每一台主机都会保存一段时间（保存的具体时长，视主机安装的操作系统而定）。跟 192.168.1.3 最后一次沟通之后，再等若干分钟，主机 192.168.1.1 将会在 ARP 缓存表内清除与 IP 地址 192.168.1.3 挂钩的记录。

ARP 请求数据包会以广播方式发出，如图 8.7 所示。

ARP 应答数据包将以单播方式发出，其目的 MAC 地址为请求地址解析的主机的 MAC 地址，源 MAC 地址为目标主机的 MAC 地址。此外，在 ARP 应答数据包中，还设有专门的字段，用来装载目标主机的 IP 地址及 MAC 地址。ARP 应答数据包的格式如图 8.8 所示。

只要发现了任何异常流量（哪怕是诸如免费 ARP 数据包、路由器执行 ARP 扫描所触发的 ARP 数据包等），都应仔细分析其原因。

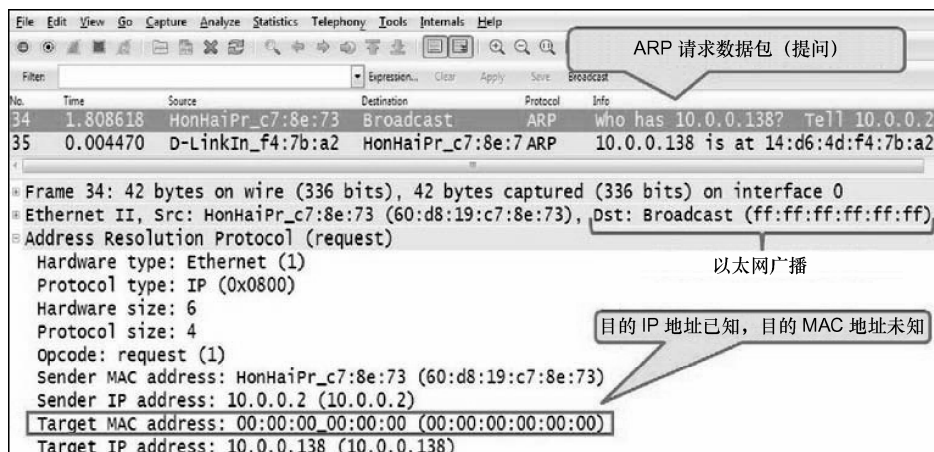


图8.7



图8.8

8.2.4 拾遗补缺

现举一个简单的例子来说明 ARP 的运作原理。图 8.9 所示为一个有意思的网络布局：那两台主机共处同一 LAN，但各自所设的 IP 地址却分属不同的 IP 子网，同时各自的 IP 地址还被设成了本机默认网关的 IP 地址。请问那两台主机可以相互 ping 通 (或以任何方式通信) 吗？

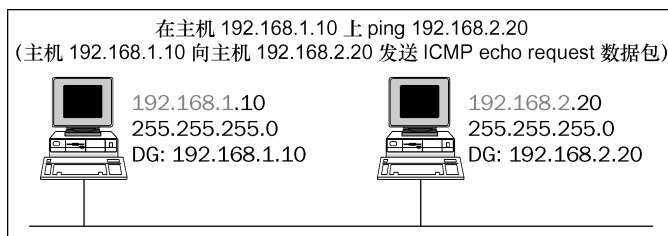


图8.9

一般人都会说根本不可能 ping 通，因为那两台主机的 IP 地址分属不同的 IP 子网，要想彼此通信，非得借助于路由器不可。但实情未必如此，让我们来仔细分析一下。

- 在主机 192.168.1.10 上 ping 192.168.2.20 (即发出目的 IP 地址为 192.168.2.20 的 ICMP echo request 数据包)。由于主机 192.168.1.10 的默认网关也是本机 IP 地址，因此会认为目的 IP 主机 192.168.2.20 跟自己共处同一 LAN，于是便发出了 ARP 请求数据包，以期解析出跟 IP 地址 192.168.2.20 配套的 MAC 地址。
- 由于本次通信的源、目主机共处同一 LAN，因此目的主机 192.168.2.20 也能收到源主机 192.168.1.10 发出的 ARP 请求数据包（切莫忘记，ARP 请求数据包是广播方式发送）。收到 ARP 请求数据包之后，192.168.2.20 会把自己的 MAC 地址封装进 ARP 应答数据包，答复 192.168.1.10^①。
- 现在，源主机 192.168.1.10 已经知道了目的主机 192.168.2.20 的 MAC 地址。于是前者就能向后者发出 ICMP echo request 数据包了，后者收到之后也能回复 ICMP echo reply 数据包，哪怕两台主机的 IP 地址不隶属同一 IP 子网。

与代理 ARP (Proxy ARP) 有关的问题也值得重点谈一谈。计算机领域里的代理 (Proxy) 泛指一种设备，这种设备能代替其他设备执行某些操作；而代理 ARP 则是指一种特殊的功能，开启了这种功能的设备（通常为路由器）能代替另一台设备应答 ARP 请求数据包。

代理 ARP 的应用场合有很多，如下所示。

- 在一台路由器之前部署了 WAN 加速/优化设备：一旦将这种 WAN 加速/优化设备配置为桥接（或）透明模式运行，此设备就会代替路由器应答 ARP 请求数据包。
- 在服务器之前部署了以透明模式运行的防火墙、WAF 以及其他设备。
- 运行了某种特殊的软件：这种特殊的软件会安装在某台服务器上，但需为之分配一个有别于服务器的 IP 地址^②。

8.3 IP 流量分析工具的使用法

IP 协议是 TCP/IP 协议栈中的网络层协议，可用来承载 HTTP、视频、IP 电话或其他应用程序（或应用层协议）数据。本节将介绍几种能帮助分析 IP 流量的 Wireshark 工具。

① 译者注：该 ARP 应答数据包的目的 MAC 地址为主机 192.168.1.10 的 MAC 地址，由于两台主机共处同一 LAN，因此 ARP 应答数据包虽以单播方式发出，主机 192.168.1.10 也能接收得到。

② 译者注：原文是 “In case of a software that requires an IP address in addition to the IP of the server it is installed on”。

8.3.1 准备工作

把 Wireshark 主机接入网络, 在交换机上开启端口镜像功能, 将进、出有待监控的主机或服务器的流量重定向至 Wireshark 主机, 启动 Wireshark 软件, 选择一块网卡, 点击 **Start** 按钮开始抓包。

8.3.2 使用方法

以下 Wireshark 工具可帮助分析 IP 流量:

- IP Statistics 工具;
- IP Name Resolution 工具。

可利用内置于 Wireshark 的若干种工具, 来查看其所抓数据包的源 IP 地址和目的 IP 地址^①。

要想查看所抓流量的源 IP 地址和目的 IP 地址, 请按以下步骤行事。

1. 进入 **View** 菜单, 选中 **Name Resolution** 菜单项下的 **Enable for the Network layer** 子菜单项。若抓包文件已经打开, 请在选中该子菜单项之后, 再点击 **Reload** 按钮。点击之后, 在抓包主窗口的数据包列表区域中, 所有数据包的源和目的 IP 地址都将被相应的域名替换, 如图 8.10 所示^②。



图8.10

2. 要想查看所抓数据包的源、目 DNS 名称的汇总统计信息, 请点击 **Statistics** 菜单中的 **Conversations** 菜单项, 在弹出的 **Conversations** 窗口中选中左下角的 **Name resolution** 复选框, 如图 8.11 所示。

① 译者注: 原文是 “When you monitor a communication line, connectivity to a server, traffic to the Internet, or any other type of traffic, there are several tools for monitoring the source and destination IPs”。

② 译者注: 这明明显示的是流量的源和目的 DNS 名称, 作者却非要说是 IP 地址, 真替作者着急。

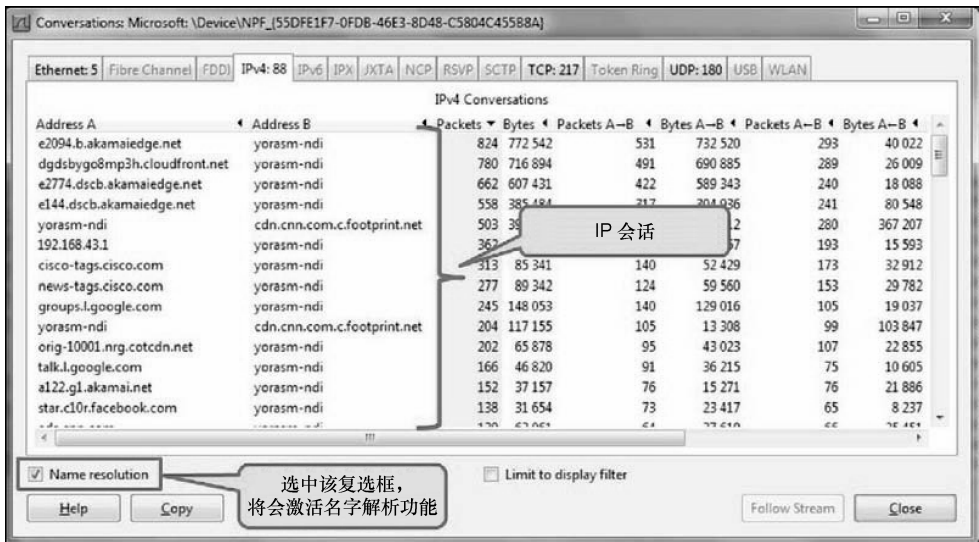


图8.11

8.3.3 幕后原理

Wireshark 软件会借用（安装其的）主机所配置的 DNS 服务器，来执行 IP 地址和域名之间的转换。在某些情况下，把所抓数据包源、目 IP 地址转换为名称来显示，能有助于发现规律性的异常流量。比方说，可以发现：

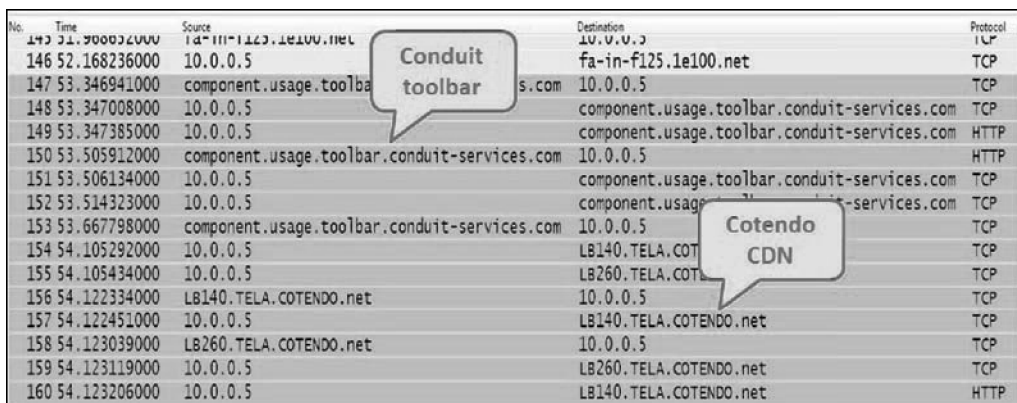
- 访问某特定 Web 站点的流量，而根据公司制度，该 Web 站点不允许访问；
- 软件自动升级所触发的流量，比如，访问杀毒软件 Web 站点或 Microsoft update 站点的流量。为使 Internet 链路得到充分利用，可部署一台中央服务器专门负责从 Internet 下载各种升级包，让所有内网 PC 都从这台服务器下载本机所需的升级包；
- 安装在内网主机的浏览器上的工具栏插件所触发的汹涌流量（请考虑一下公司内网里的每一台主机除了常规的上网流量之外，再打开约 50~100 条连接所触发的访问 Internet 的流量吧！）^①。

8.3.4 拾遗补缺

若在某台主机的浏览器中安装了 Conduit toolbar 插件，在运行该插件的同时，使用 Wireshark 抓包，将会看见该主机向多个 Web 站点发包，这些 Web 站点既有我们熟悉的，也有

^① 译者注：原文是“Toolbar traffic can cause a huge amount of traffic if installed on organization devices (think about 50-100 opened connections on every device in your company in addition to regular traffic)”。

我们不熟悉的。由图 8.12 所示的 Wireshark 截屏可知，安装 Conduit toolbar 插件的主机正在向 Conduit 和 **Content Delivery Network (CDN)** Web 站点发包。



No.	Time	Source	Destination	Protocol
143	51.900032000	fa-in-f125.1e100.net	10.0.0.5	TCP
146	52.168236000	10.0.0.5	fa-in-f125.1e100.net	TCP
147	53.346941000	component.usage.toolbar.conduit-services.com	10.0.0.5	TCP
148	53.347008000	10.0.0.5	component.usage.toolbar.conduit-services.com	TCP
149	53.347385000	10.0.0.5	component.usage.toolbar.conduit-services.com	HTTP
150	53.505912000	component.usage.toolbar.conduit-services.com	10.0.0.5	HTTP
151	53.506134000	10.0.0.5	component.usage.toolbar.conduit-services.com	TCP
152	53.514323000	10.0.0.5	component.usage.toolbar.conduit-services.com	TCP
153	53.667798000	component.usage.toolbar.conduit-services.com	10.0.0.5	TCP
154	54.105292000	10.0.0.5	L8140.TELA.COTENDO.net	TCP
155	54.105434000	10.0.0.5	L8260.TELA.COTENDO.net	TCP
156	54.122334000	L8140.TELA.COTENDO.net	10.0.0.5	TCP
157	54.122451000	10.0.0.5	L8140.TELA.COTENDO.net	TCP
158	54.123039000	L8260.TELA.COTENDO.net	10.0.0.5	TCP
159	54.123119000	10.0.0.5	L8260.TELA.COTENDO.net	TCP
160	54.123206000	10.0.0.5	L8140.TELA.COTENDO.net	HTTP

图8.12

要想了解安装 Conduit toolbar 插件的主机具体访问了哪些 Web 站点，打开过哪些页面，可借助于 **Statistics** 菜单下的 **HTTP** 信息统计工具（需搭配相应的 IP 显示过滤器）。

以下所列为高效使用浏览器工具栏插件的规则。

- 应在单位或公司的制度里明文规定哪些工具栏插件允许使用，哪些不允许使用，禁止用户安装不允许使用的工具栏插件。
- 对 Internet 链路进行流量监控，对工具栏插件所触发的流量做到及时发现。

8.4 利用GeoIP来查询IP地址的归属地

Wireshark 1.1.2 及以上版本支持使用 GeoIP（商业版）和 GeoLite（免费版）数据库，来查询其所抓数据包中的 IP 地址的归属地信息（比如，该 IP 地址来源于哪个国家、哪个城市、哪个 AS 等）。

8.4.1 准备工作

1. 访问 Web 站点：<http://dev.maxmind.com/geoip/legacy/geolite/>。
2. 下载以下文件（二进制）。
 - GeoLite Country（IPv4 国家）
 - GeoLite City（IPv4 城市）

- GeoLite ASN (IPv4 自制系统号)
- GeoLite Country (IPv6 国家)
- GeoLite City (IPv6 城市)
- GeoLite ASN (IPv6 自治系统号)



注意

自治系统 (**Autonomous System, AS**) 是定义外部网关协议 (**Exterior Gateway Protocols, EGP**) 的标准文档里使用的术语, 所指为受控于同一网络运营机构的所有路由器。要想通过 EGP 与不同的 Internet 服务提供商 (**ISP**) 建立对等关系, 进而接入 Internet, 就必须为本方网络申请 AS 号, 而 ISP 也有自己的 AS 号。边界网关协议版本 4 (**BGP4**) 是 EGP 事实上的标准。

8.4.2 查询方法

下载完上述文件之后, 请按以下步骤行事。

1. 将所有文件都归结于同一目录 (也可以分别放置于不同的目录, 但会为后续操作带来不便)^①。
2. 让 Wireshark 知道上述文件的安置目录。为此, 请点击 **Edit** 菜单中的 **Preferences** 菜单项, 在弹出的 **Preferences** 窗口中选 **Name Resolution** 参数, 再点击 **GeoIP database directories** 旁的 **Edit** 按钮。
3. 在弹出的 GeoIP database path 窗口中指定上述文件的安置目录, 如图 8.13 所示。



图8.13

^① 译者注: 还得解压缩。

4. 点击 **OK** 按钮，关闭 GeoIP database path 窗口，重启 Wireshark 软件。
5. 重新开始抓包（或打开一个抓包文件），然后点击 **Statistics** 菜单下的 **Endpoints** 菜单项，在弹出的 Endpoints 窗口中选择 IPv4 标签栏。在该标签栏下，应该可以看见 GeoIP 信息了（见图 8.14）。

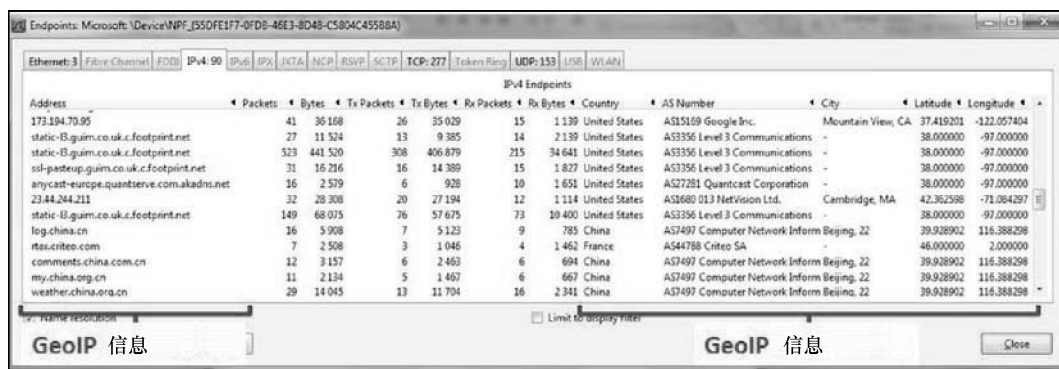


图8.14

6. 在抓包主窗口的数据包结构区域内，也可以看见 GeoIP 信息（见图 8.15）。不过，需要先点击 **Edit** 菜单中的 **Preferences** 菜单项，在弹出的 **Preferences** 窗口中点击 **Protocol** 配置选项前的+号，再选中 IPv4 协议，勾选 **Enable GeoIP lookup** 旁的复选框。



图8.15

8.4.3 幕后原理

IP 地址由 Internet 标准化组织（ISO）旗下的 Internet 编号分配机构（IANA）负责分配。IANA 会先把 IP 地址分配给各个区域性的 Internet 注册机构（RIPE-NCC、APNIC、AFRINIC、LACNIC 和 ARIN 等），后者会继续分配给各个国家的 ISP。最后，由 ISP 分配给各自的客户。GeoIP 是一个存储 IP 地址的归属地信息的数据库，Wireshark 会基于该数据库，来解析其所抓数据包中的 IP 地址的归属地。

maxmind.com/geoip/geolite#IP_Geolocation-1.

8.4.4 拾遗补缺

使用 GeoIP 的好处:

- 可以了解到内网用户都在访问那些 Web 站点；
- 可以解析出自外而内的 Internet 流量的源 IP 地址的归属地；
- 找乐。

8.5 发现IP包分片问题

当 IP 数据包的长度超过其所穿链路的 MTU 值时，在发送之前，须先分为几片。一般而言，IP 数据包以分片方式发送也不能说不正常，但却有可能会对性能产生影响。

8.5.1 准备工作

把 Wireshark 主机接入网络，在交换机上开启端口镜像功能，将进、出有待监控的主机或服务器的流量重定向至 Wireshark 主机，启动 Wireshark 软件，选择一块网卡，点击 **Start** 按钮开始抓包。IP 包分片主要会影响到交互式应用程序（比如数据库应用）的性能，应该在这些应用程序上面查找问题。

8.5.2 操作方法

当 IP 包以分片方式发送时，若使用 Wireshark 软件抓包，在抓包主窗口中会看见身背 fragmented IP Protocol 字样的数据包，同时还会夹杂出现 TCP 或 UDP 数据包，如图 8.16 所示。

当定位与网络或应用程序性能有关的故障（比如，数据库客户端与服务器端之间的连接异常缓慢）时，请执行以下步骤来验证其中是否存在 IP 包分片现象。

1. 检查数据库客户端与服务器端之间的 IP 连通性，以确保无其他网络层面的问题。
2. 用 Wireshark 抓取数据库客户端与服务器端之间流量，检查是否存在 IP 包分片的现象。
图 8.16 所示为 Wireshark 抓到经过分片的 IP 数据包时的景像。
3. 如怀疑性能问题与 IP 包分片有关，请找一位懂行的 DBA 去调整数据库软件及数据库服务器的配置参数，让服务器不把经过分片的 IP 包传送至网络。



图8.16

4. 一般而言，在诸如以太网之类的网络环境中，封装在每个 IP 数据包中的数据净载（包括 TCP 头部在内）不应超过 1480 字节。因此，封装在单个 TCP 报文段内，由数据库服务器软件生成的每个数据单元的长度不应超过 1460 字节^①。

若 IP 包的某些协议头部需“占用”更多的字节（比如，当启用了某种隧道传输机制或者需要让 TCP 头部包含某些选项字段时），请告知 DBA，让其设法降低封装进单个 TCP 报文段中的数据单元的长度。在调整由数据库软件生成的数据单元的最大长度时，应该让 DBA 注意把握分寸，要设法使封装 TCP 报文段的 IP 数据包刚好不分片。

8.5.3 幕后原理

要想彻底弄清 IP 数据包的分配问题，需先理解下面两个定义数据单元长度的术语（见图 8.17）。

- **最大传输单元（MTU）**：是指由 IP 包头和数据部分组成的 IP 数据包的长度^②。
- **最大报文段大小（MMS）**：是指 TCP 报文段内数据净载的最大长度，亦即上层协议交付给 TCP 的每个数据单元的最大长度。

图 8.18 展示了 IPv4 数据包的分片机制。

① 译者注：原文是“The recommended packet size in Ethernet is not greater than 1460 bytes minus the TCP header size. Thus, the segments coming out of the interface should have a size of 1420-1440 byte”。译文酌改。

② 译者注：原文是“This is the size of the IP packet including the header and the data”。译文按字面意思翻译，但译者想说，作者的网络理论知识水平奇差，他给 MTU 下的定义是错误的。

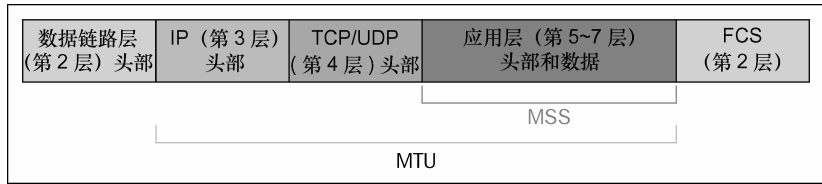


图8.17

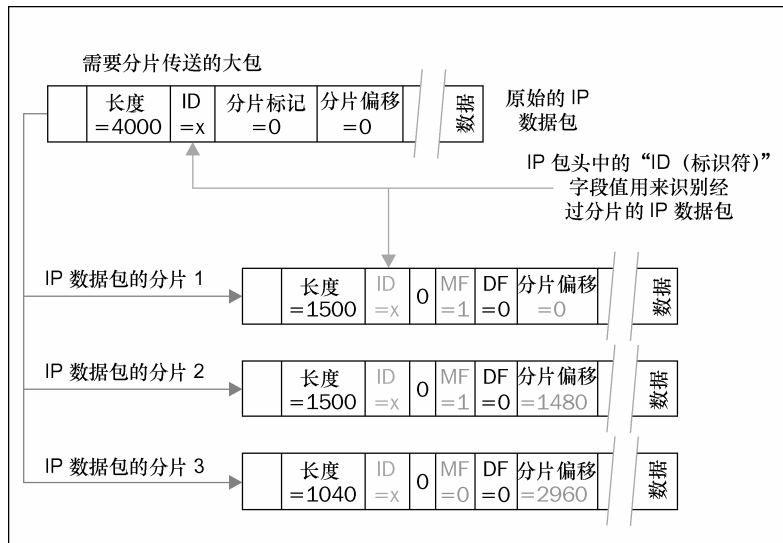


图8.18

1. 原始的大 IP 数据包流入 NIC 或路由器（接口），该 IP 数据包过长，需要分片发送。
2. 这一大 IP 数据包会被分为若干小 IP 数据包发送（具体分为几个，要视大 IP 数据包的原始长度而定）。
3. IP 数据包以分片方式发送时，IP 包头中的以下字段将发挥作用。
 - **标识符（ID）字段值：** 等于原始数据包包头的 ID 字段值。
 - **标记字段中的位 0（bit 0）：** 总是置 0。
 - **标记字段中的位 1（DF 位）：** 置 0，表示本 IP 数据包可被分片；置 1，表示本 IP 数据包不能被分片。
 - **标记字段中的位 2（MF 位）：** 置 0，表示本 IP 数据包是最后一个分片；置 1，表示本 IP 数据包是众多分片之一，且不是最后一个分片。
 - **分片偏移字段：** 表示分片 IP 数据包中数据净载的首字节，相对于原始 IP 数

据包（未分片 IP 数据包）中数据净载的首字节的偏移字节数。

对于 IPv4 数据包，生成它的主机以及传播途中的路由器都可以对它进行分片。

对于 IPv6 数据包，只有生成它的主机才能对它进行分片，IPv6 数据包传播途中的路由器则无权对其分片。IPv6 数据包的分片由专门的分片扩展包头来掌控。

8.5.4 拾遗补缺

在发往既定目的网络的沿途，一个 IP 数据包可被多次分片，但只有目的主机才能重新组装分片。

以图 8.19 中的网络为例，图中所示为一大型网络的缩略图。该大型网络由数百个远程分支机构网络和一个居中的数据中心网络构成，那几百个分支机构网络都通过架设在 SP（服务提供商）网络上的 IPSec/GRE 隧道连接到数据中心网络。

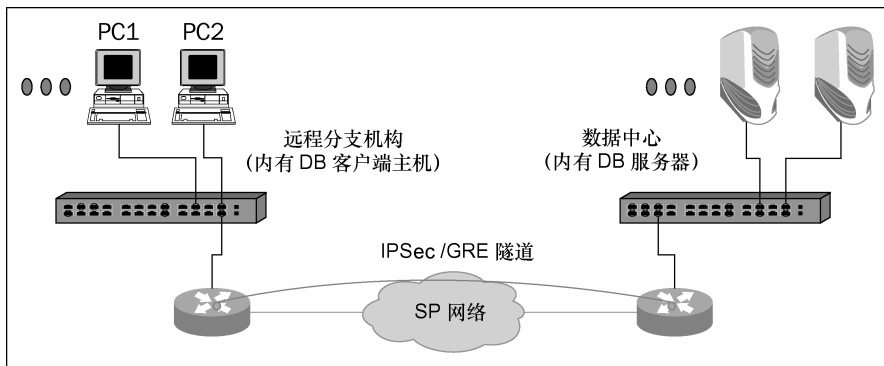


图8.19

每个分支机构网络中都有 5~10 台安装了 DB 客户端的 PC，需要访问数据中心网络内的 DB 服务器。

问题出在数据库应用程序身上——该应用程序创建的单个 IP 数据包的长度为 1800 字节，需要以分片的方式来发送^①。

- 首先，在安装应用程序的 PC 上，NIC 要先把每一个大 IP 包分解为两个 IP 数据包（一个是 1500 字节，一个是 300 字节），才能放到网络上发送。
- 其次，由于分支机构网络和数据中心网络之间通过 IPSec/GRE 隧道互连，因此在分

^① 译者注：原文是“The problem was that in some of the database applications, the database created frames of 1800 bytes that were fragmented”。对照上下文，作者的说法就根本讲不通，译文酌改。

支机构网络的出口路由器上，还需要对 1500 字节的常规 IP 数据包做第二次分片（分为 1420 字节和 80 字节的 IP 数据包各一）。

也就是说，由 PC 发出的每一个数据包，到了 DB 服务器这里就变成了三个。因为连接数据库服务器的 PC 实在太多，而数据库服务器的配置也有点过时，加之还得重组所有以分片方式发送的数据包，所以很多用户都反映该应用程序使用起来极为缓慢。

图 8.20 所示为 Wireshark 抓到的由 PC 发出的访问 DB 服务器的 IP 数据包。图中的数据包结构区域显示的是编号为 443 的 IP 数据包，这也是 PC 发出的首个分片的 IP 数据包。

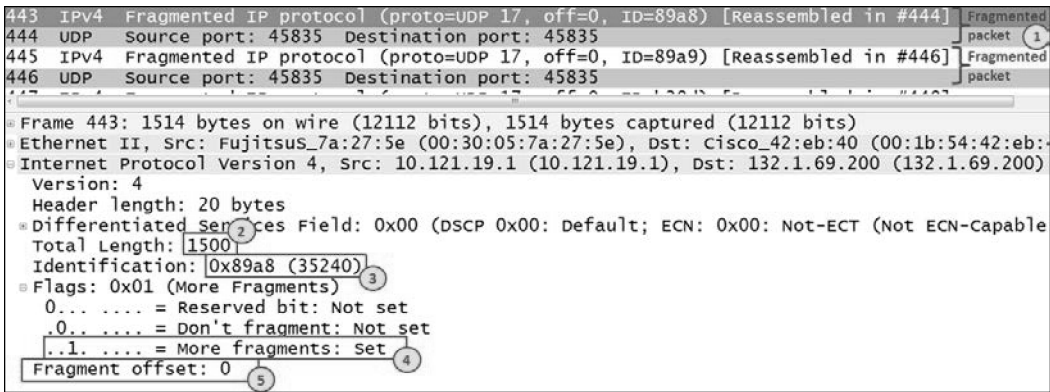


图8.20

由图 8.20 可知，编号为 443 和 444 的数据包①是原始 IP 数据包的 2 个分片。编号为 443 的 IP 数据包的长度为 1500 字节②；标识符字段值为 (0x89a8) ③；MF 位置 1④，表示其后跟随更多分片；分片偏移字段值为 0⑤，表示其为原始 IP 数据包的首个分片。

图 8.21 所示为抓包文件中编号为 444 的 IP 数据包——原始 IP 数据包的第二个分片。

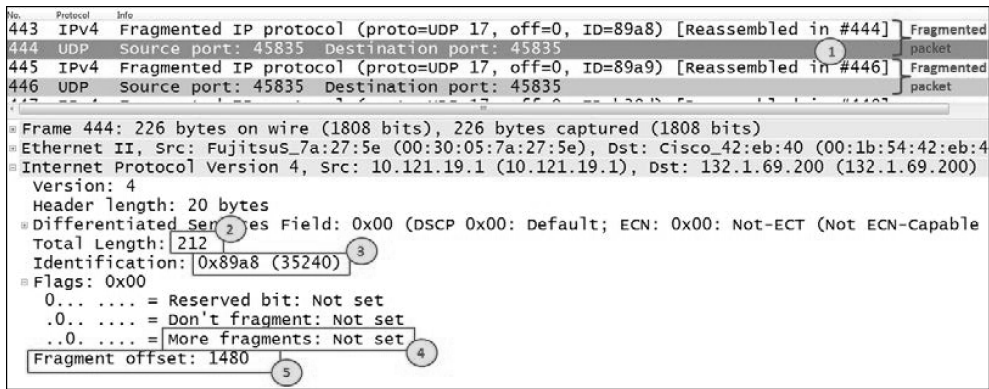


图8.21

由图 8.21 可知, 编号为 444 的 IP 数据包的长度为 212 字节②^①; 标识符字段值为 (0x89a8) ③, 表示其与编号为 443 的数据包是同一个原始 IP 数据包的两个分片; MF 位置 0④, 表示其为原始 IP 数据包的最后一个分片; 分片偏移字段值为 1480⑤, 可推断其为原始 IP 数据包的第二个分片。

8.6 路由选择故障分析

路由选择故障属于关键性网络故障。路由选择故障包括: 路由环路和通往目的网络的路由失效等。一般而言, 排除路由选择故障, 无需借助于 Wireshark。在大多数情况下, 只要通晓路由协议原理外加某些网络常识, 便足可完成排障任务了。本节会介绍借助 Wireshark 排除 IP 路由选择故障的基本技巧。

8.6.1 准备工作

首先, 要掌握 ping、tracert (或 traceroute) 等基本排障命令的用法。凭借这两条命令, 再登录进路由器 (执行相关 show 命令), 一般的路由选择故障都能解决。

本节会给出当网络中发生路由选择故障时, 可能会在 Wireshark 抓包主窗口中显现的重要信息。

8.6.2 操作方法

本节不会像往常那样告知读者具体的操作方法, 而是会指出 (在 Wireshark 抓包文件中) 需要关注的地方。

处理路由选择故障时, 在 Wireshark 抓包文件中需要关注的地方如下所示。

- 是否出现了身背 **Time To Live exceeded** 字样的 IP 数据包。在 IP 数据包的包头内, 有一个名为生存时间 (Time To Live) 的字段。当 Wireshark 抓到 TTL 字段值为 0 的 IP 数据包时, 通常就表示网络中发生了数据包转发环路, 而数据包转发环路往往是拜路由环路所赐。请读者牢记, Wireshark 会让 TTL 字段值为 0 的 IP 数据包身背 **Time To Live exceeded** 字样。
- 是否包含身背以下字样的 ICMP 数据包。
 - **Destination network unreachable**: 当 Wireshark 抓到身背 **Destination network unreachable** 字样的 ICMP 数据包时, 通常都表示发送该 ICMP 数据

① 译者注: 前面才说是 300 字节呢, 怎么一下就变了?

包的路由器丢失了通往目的网络的路由。

- **Destination host unreachable:** 当 Wireshark 抓到身背 **Destination host unreachable** 字样的 ICMP 数据包时，通常都表示目的网络中的主机未上线运行，或已上线运行但未设默认网关。

8.6.3 幕后原理

在 IP 数据包的包头内，有一个名为生存时间（Time To Live, TTL）的字段，该字段值的取值方法如下所列。

- 生成 IP 数据包的主机会为 IP 包头的 TTL 字段设置一个初始值，通常为 64、128 或 256，具体数字随主机的操作系统而异。
- IP 数据包每穿越一台路由器，其 TTL 字段值便会减一。试举一例，TTL 字段值为 128 的 IP 数据包穿越 10 台路由器之后，该字段值会降至 118。
- 收到 TTL 字段值为 1 的 IP 数据包时，路由器会将该字段值降至 0，然后做丢弃处理，同时向该 IP 数据包的源 IP 地址发出 ICMP TTL 超时消息。



注意

凭借 IP 包头中的 TTL 字段值，即能判断出 IP 数据包在行进过程中穿越的路由器的台数。能如此判断，基于两个前提条件：首先，即便对于在 Internet 上发送的 IP 数据包，其所穿路由器的台数也不会高于 30；其次，生成 IP 数据包的主机只可能将 IP 数据包包头的 TTL 字段值设为 64、128 或 256 这三值之一。也就是讲，若一 IP 数据包的 TTL 字段值为 110，则可断定该 IP 数据包在行进过程中已经穿越了 18 台路由器（128-110），因为它不可能穿越 146 台路由器（256-110），这不符合上面提到的第一个前提条件。

8.6.4 拾遗补缺

图 8.22 所示网络展现了一个典型的路由选择故障场景。由图可知，该网络由一个居中的数据中心站点外加两个远程分支机构站点构成。为了追求分支机构站点和数据中站点间链路的冗余性，各个站点的 WAN 路由器以环状方式互连^①。

在各个站点的 WAN 路由上，除了设有通往 Internet 的默认静态路由以外，还开启了动态路由协议 EIGRP。图 8.23 所示为相关 WAN 路由器的部分路由表表项（与本场景有关的路由表表项）。

^① 译者注：原文是“The network was built this way in order to provide redundancy from the remote offices to the central data center”。作者除了能在写作时“装神弄鬼”，似乎也没别的了。

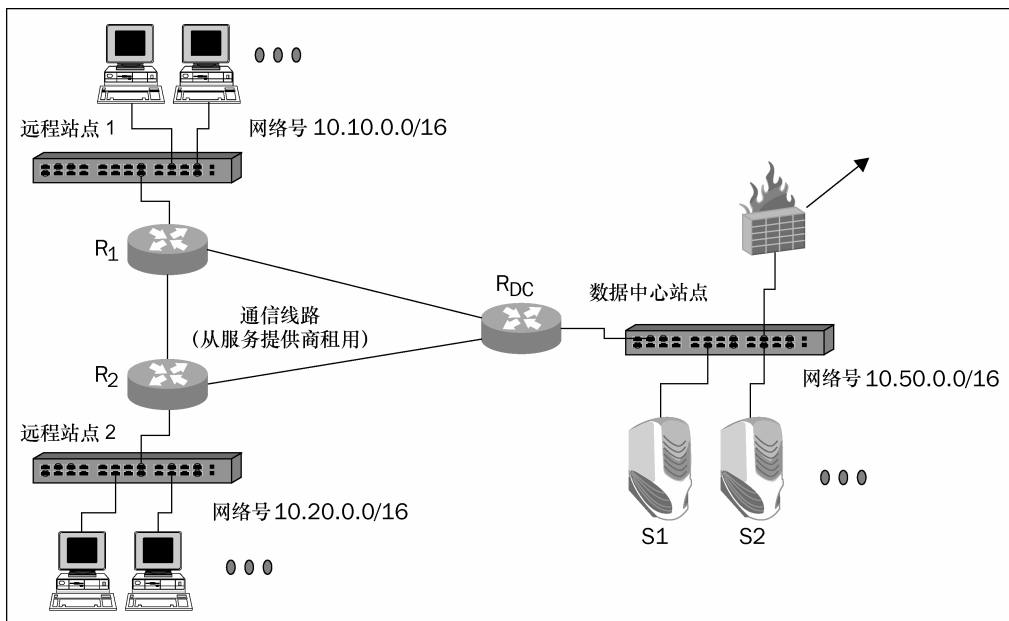


图8.22

路由器	目的网络	下一跳	管理开销	路由协议
R _{DC}	10.10.0.0/16	R ₁	5	EIGRP
	10.20.0.0/16	R ₂	5	EIGRP
	0.0.0.0/0	FW	1	Static
R ₁	10.50.0.0/16	R _{DC}	5	EIGRP
	0.0.0.0/0	R _{DC}	1	Static
	0.0.0.0/0	R _{DC}	1	Static
R ₂	10.50.0.0/16	R _{DC}	5	EIGRP
	0.0.0.0/0	R _{DC}	1	Static
	0.0.0.0/0	R _{DC}	1	Static
FW	10.10.0.0/16	RDC	1	Static
	10.20.0.0/16	RDC	1	Static
	0.0.0.0/0	ISP	1	Static

图8.23

导致故障的直接原因比较奇葩：从 RDC 到 R1 和 R2 的专线被一台拖拉机压断了。

这就导致分支机构站点 1 和 2 立刻与数据中心站点失去了“联系”，但问题是：拥有几百台 PC 的公司总部网络（位于数据中心站点内）却突然变得奇慢无比，在访问 Internet 时更是慢得离谱。

作者赶到现场之后，将安装了 Wireshark 的笔记本接入该公司总部网络的核心交换机，借助于端口镜像功能，抓取了整个 VLAN 的流量。经过仔细观察，作者居然发现了流量转发环路。让我们来看一看环路是怎么发生的。

1. 数据中心站点用来连接分支机构站点的 2 条专线被拖拉机压断，RDC 与 R1、R2 的 EIGRP 邻居关系中断。
2. 服务器发出的目的网络为 10.10.0.0/16 或 10.20.0.0/16 的数据包，会先交由自己的默认网关 RDC 转发。
3. 收到目的网络为 10.10.0.0/16 或 10.20.0.0/16 的数据包之后，RDC 会继续转发给防火墙，这得归咎于 RDC 上设置的那条默认静态路由。当与那两个目的网络相对应的 EIGRP 明细路由在路由表里退位之后，RDC 会遵循默认静态路由，把所有目的网络地址未知（不匹配明细路由）的流量都转发给了防火墙。
4. 收到目的网络为 10.10.0.0/16 或 10.20.0.0/16 的数据包之后，防火墙会回发给 RDC，这要拜防火墙上设置的通往那两个目的网络的静态路由所赐。在防火墙上设置那两条静态路由的本意，是要把分支机构 1 和 2 访问 Internet 的回程流量引导至 RDC。
5. 最终的结果是，数据中心站点内服务器发出的归宿为分支机构站点 1 和 2 的所有数据包，像乒乓球那样在 RDC 和防火墙之间“来回跳跃”，从而对公司总部网络内的用户访问 Internet 产生了影响。

8.7 发现IP地址冲突

IP 地址冲突是最棘手的网络故障之一，但只要对导致 IP 地址冲突的各种原因摸得门清，这类故障解决起来倒也不难。

8.7.1 准备工作

若怀疑网络中存在 IP 地址冲突，首先应考虑使用内置于主机操作系统（如 MS Windows）的 ping 和 arp 命令。要是不能确定网络故障是不是由 IP 地址冲突所引发，则需将安装了 Wireshark 的主机接入 LAN 交换机，然后每个 LAN（VLAN）挨个抓包。

8.7.2 操作方法

当 IP 地址冲突涉及关键网络设备时，所表现出的症状包括：服务器访问速度变慢、上网速度变慢、所有设备都 ping 不通等。

1. 当某台服务器的访问速度明显变慢时，其 IP 地址跟别的设备冲突是原因之一。要证实这一点，应先试着 ping 一下这一 IP 地址。



注意

对安装了某些操作系统的主机而言,当 IP 地址跟别的设备发生冲突时,其网卡驱动程序会失效(若主机的操作系统为 Windows,在屏幕的右下角弹出一个提示框)。然而,安装了另外一些操作系统的主机在 IP 地址跟别的设备发生冲突时,其网卡驱动程序非但不会失效,而且也不会生成任何提示信息,这才是进一步引发故障的导火索。

2. 在命令行下执行命令 `arp -a` (Windows 操作系统),若在命令的输出中发现刚 ping 的目的 IP 地址与两个不同的 MAC 地址挂钩,则表明存在 IP 地址冲突。
3. 执行 Google 搜索,看看那两个 MAC 地址的前三个字节跟哪个网卡芯片制造商有瓜葛。这有助于加快定位引发 IP 地址冲突的元凶。
4. 要想得知拥有那两个 MAC 地址的主机分别跟交换机的哪个端口相连,请登录交换机(当然,交换机必须是可网管交换机),根据其生成的 MAC 地址表来判断。



注意

有多款软件都可用来显示交换机各端口所连设备清单(包括设备的 MAC 地址、IP 地址、DNS 名称等)。可在 Google 上搜索 switch port mapper 或 switch port mapping tools,来下载这些软件。

5. 要是 ping 和 arp 双管齐下也看不出个所以然来,请把 Wireshark 主机接入 LAN 交换机,启用端口镜像功能,抓取相关 VLAN 的流量。Wireshark 能提供更多与 IP 地址冲突有关的线索。
6. 图 8.24 所示为 Wireshark 生成的有关 IP 地址冲突的提示信息。



图8.24

8.7.3 幕后原理

当同一 LAN 内有两台(或两台以上的)设备设有相同的 IP 地址时,若在一台主机上 ping 这一 IP 地址,将会导致主机的 ARP 缓存中进驻该 IP 地址的两条(或多条) ARP 缓存条目,每一条都会分别与一个不同的 MAC 地址相对应。

对于安装了某些操作系统（比如，Windows 系统）的主机而言，若 IP 地址跟别的设备冲突，则其不但会弹出窗口，告知 IP 地址冲突，而且还会让网卡驱动程序失效（比如，安装了 Windows 操作系统的主机）。

而对于安装了另外一些操作系统的主机而言，若 IP 地址跟别的设备冲突，则其不会有任何表示。此时，就得 ping 和 arp 命令双管齐下。

在执行抓包任务时，只要看见 Wireshark 抓到了身背 duplicate IP 字样的数据包，一定不能掉以轻心。

8.7.4 拾遗补缺

IP 地址冲突通常是指同一 LAN 内有两台设备都设有相同的 IP 地址，但最奇葩的情况则莫过于三台设备都设有相同的 IP 地址。

现在来看看这一奇葩的 IP 地址冲突现象，请看图 8.25 所示的网络拓扑。

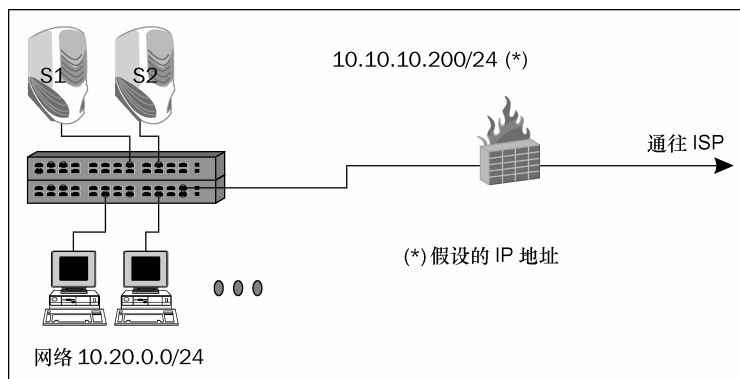


图8.25

在图 8.25 所示网络中约有 150 台主机要通过防火墙访问 Internet。故障现象是，上网速度慢得离谱。

以下所列为在内网主机上 ping 公网主机（ping 任何一台公网主机均可）的输出。

```
Reply from 173.194.35.148: bytes=32 time=98ms TTL=51
Request timed out.
Reply from 173.194.35.148: bytes=32 time=124ms TTL=51
Request timed out.
Reply from 173.194.35.148: bytes=32 time=134ms TTL=51
Request timed out.
Reply from 173.194.35.148: bytes=32 time=582ms TTL=51
Request timed out.
```

在上网（访问 Internet）变慢之后，该网络的网管人员对网络进行了整治，但整治以后，上网却变得更慢，以下所列为在同一台内网主机上 ping 同一台公网主机的输出。

```
Reply from 173.194.35.148: bytes=32 time=98ms TTL=51
Request timed out.
Request timed out.
Reply from 173.194.35.148: bytes=32 time=124ms TTL=51
Request timed out.
Request timed out.
Reply from 173.194.35.148: bytes=32 time=134ms TTL=51
Request timed out
Request timed out
```

当作者接手处理该故障以后，第一件事也是在网主机上 ping 公网主机，然后再执行命令 `arp -a`，观察其输出。作者发现，跟 IP 地址 10.10.10.200 相对应的 MAC 地址居然有 3 个。也就是说，网络中发生了 IP 地址冲突，冲突的设备还不止两台。作者顺藤摸瓜，得到了图 8.26 所示的网络拓扑图。

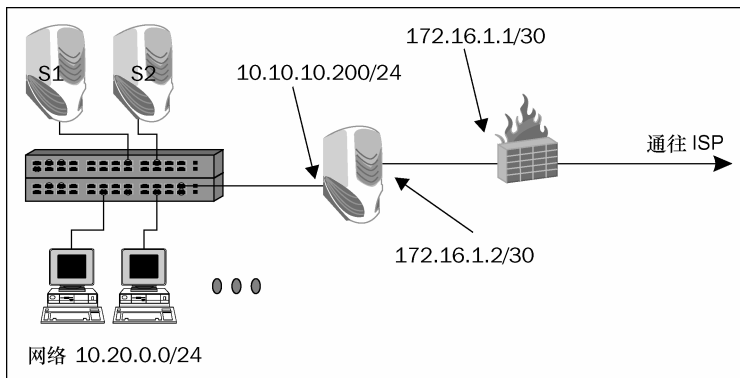


图8.26

由图 8.26 可知，内网主机的默认网关其实还不是防火墙，而是那台夹在防火墙和交换机之间的 Web 过滤设备，其 IP 地址是 10.10.10.200。防火墙和 Web 过滤设备之间的互连子网网段为 172.16.1.0/30。

现在，让作者来揭晓答案。

1. 网络中的主机是以 DHCP 方式自动获取 IP 地址，但 DHCP 服务器在分配 IP 地址时只是排除了 IP 地址范围 10.10.10.201~254，内网主机所设默认网关的 IP 地址 10.10.10.200 不在此列。
2. 该网络原本只有一台交换机，网管人员又新购了一台，将两者配置为以堆叠方式运行。

这台新交换机的管理 IP 地址也被配置为以 DHCP 方式自动获取，凑巧的是，它偏偏获取到了默认网关的 IP 地址 10.10.10.200。此时，这台新交换机的管理 IP 地址跟默认网关（Web 过滤设备）的 IP 地址冲突。

3. IP 地址冲突以后，上网速度立马变慢，该网络的管网人员怀疑是 Web 过滤设备有问题。于是，便断掉了防火墙和 Web 过滤设备的链路，把防火墙内网接口的 IP 地址先设成 10.10.10.200，再直接与交换机相连。更搞笑的是，网管人员只把连接 Web 过滤设备外部接口的网线给扯了，却忘了拽掉内部接口的网线，其 IP 地址 10.10.10.200 也忘了改。
4. 如此一来，在这一小小的 LAN 内便出现了三台设备同时设有 IP 地址 10.10.10.200 的局面，如图 8.27 所示。

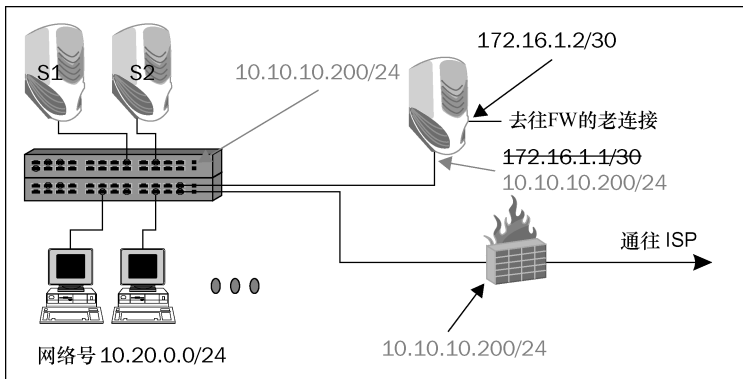


图8.27

通过上例以及作者处理过的无数案例，可以总结出一个重要结论：一定要及时更新自己所维护的网络的拓扑结构图。

8.8 DHCP故障分析

动态主机配置协议（Dynamic Host Configuration Protocol, DHCP）是一种能够让设备在接入网络时自动获取 IP 地址的协议。本节将介绍如何定位并解决常见的 DHCP 故障。

8.8.1 准备工作

当 DHCP 服务器已在网络中就位，而接入网络的 PC 不能自动获取 IP 地址时，需把 Wireshark 主机接入 LAN 交换机，同时启用端口镜像功能，将相关端口（连接了不能自动获取 IP 地址的 PC 的端口）流量重定向至 Wireshark 主机。然后，在不能自动获取 IP 地址的 PC 上，执行 `ipconfig /release` 和 `ipconfig /renew` 命令（或将该主机的网卡所连网线先拔

出，再插回）。最后，分析 Wireshark 抓取到的 DHCP 流量。

8.8.2 操作方法

下一小节详述了 DHCP 协议的运作流程，接入网络的主机在执行 IP 地址获取动作时，只要有任何地方违背了该流程，便不可能获取得到。此时，请按以下思路来解决故障。

1. 主机是否已发出 **DHCP Discover** 数据包？
2. 若已发出，则表示主机（DHCP 客户端）正常。
3. 若未发出，则需要：
 - 检查主机的 TCP/IP 配置情况，应将主机的 IP 地址获取方式配置为自动获取；
 - 检查主机的物理连接情况。当主机通过 WiFi 接入网络时，若关联不成功，则无法发出 **DHCP Discover** 数据包。
4. 若（借助于 Wireshark 得知）主机在发出 **DHCP Discover** 数据包之后，收到了一台 DHCP 服务器发出的 **DHCP Offer** 数据包，则表示情况正常，需继续观察。
5. 若（借助于 Wireshark 得知）主机发出 **DHCP Discover** 数据包之后，收到了两台 DHCP 服务器发出的 **DHCP Offer** 数据包，则表明该 LAN 内存在一台以上的 DHCP 服务器，网内主机可能会抓到多套 IP 地址。应关闭其中的一台（或多台）DHCP 服务器，至少应停掉一台（或多台）服务器上开启的 DHCP 服务。
6. 若（借助于 Wireshack 得知）主机收到了 **DHCP Offer** 数据包，然后发出了 **DHCP Request** 数据包（向 DHCP 服务器索要 IP 地址），且收到了包含 IP 地址参数的 **DHCP Ack** 数据包，则表示情况正常；若（观察到）主机未能收到 DHCP 服务器发出的 **DHCP Ack** 数据包，且再三发出 **DHCP Request** 数据包，但未收到任何回应，则表示 DHCP 服务器已不能正常运作，请检查之。
7. 若（借助于 Wireshark 得知）主机在发出 **DHCP Request** 数据包之后，收到了 **DHCP Decline** 数据包，则表示 DHCP 服务器拒绝提供 IP 地址信息，可能的原因如下所列：
 - DHCP 服务器已无可供分配的 IP 地址；
 - DHCP 服务器已将该主机之前获取到的 IP 地址分配给了另一台设备。导致该问题的原因是，DHCP 服务器已被配置为向每一台 DHCP 客户端设备提供“固定”的 IP 地址，此时需检查 DHCP 服务器的配置。

8.8.3 幕后原理

DHCP 协议看来简单，但其实却颇为复杂。以下所列为 DHCP 服务器和 DHCP 客户端设备之间的交流过程。

1. **DHCP 发现：**DHCP 客户端设备对其某个版本的 IP 协议栈进行初始化，同时以广播方式发出 DHCP Discover 数据包，寻找本 LAN 内的 DHCP 服务器。DHCP Discover 数据包为 UDP 数据包，其源、目端口号分别为 68 和 67。
2. **DHCP 租用提供：**提供 DHCP 服务的 DHCP 服务器会侦听 UDP 67 端口，若收到 DHCP Discover 数据包，则会回复 DHCP Offer 数据包，表明本服务器愿意提供 IP 地址“租用”服务。
3. **DHCP 租用请求：**收到 DHCP Offer 数据包之后，DHCP 客户端设备会回复 DHCP Request 数据包，请求 DHCP 服务器提供与本机 IP 配置有关的信息^①。
4. **DHCP 租用确认：**收到 DHCP Request 数据包之后，DHCP 服务器回复 DHCP Ack 数据包，其中会包括由 DHCP 服务器提供的 IP 地址、子网掩码、默认网关 IP 地址、DNS 服务器 IP 地址等 IP 配置信息。

图 8.28 所示为一台 DHCP 服务器与一台 DHCP 客户端设备相互交流时，Wireshark 抓取的全套 DHCP 报文。



图8.28

8.8.4 拾遗补缺

把 DHCP 客户端主机接入网络时，若获取到的 IP 地址不对路，则一定是有人悄悄地在 LAN 内私设 DHCP 服务器。这台 DHCP 服务器一般都是普通的宽带路由器。要想发现这一私接宽

^① 译者注：原文此句之后还有一句“The request will be, for example, the IP address that we requested before (because we had it before), for our MAC address so that the server will recognize us as a prior client with a saved IP address and other parameters”。译者认为此句有蛇足之嫌，掠过不译。

带路由器的行为，请按以下步骤行事。

1. 在主机上执行 `ipconfig` 命令，观察其输出，若发现主机获取到的 IP 地址“不对路”，则表示有人私设 DHCP 服务器。
2. 若怀疑这台 DHCP 服务器是有人私接的宽带路由器，请在主机上 `ping` 通过 DHCP “抓来”的默认网关的 IP 地址。该默认网关的 IP 地址很可能就是宽带路由器（DHCP 服务器）自身的 IP 地址。
3. 在主机上执行 `arp -a` 命令，通过其输出来获取默认网关（宽带路由器）的 MAC 地址。凭借这一 MAC 地址，即可判断出拥有这一 MAC 地址的网络设备制造商。这样一来，便可获悉宽带路由器的品牌（D-Link、TP-Link、Netgear 等）。此外，还可以登录 LAN 交换机，显示交换机的 MAC 地址表，根据 MAC 地址表的内容来获知宽带路由器与哪个端口相连。然后，可在交换机上禁用该端口，让那台私接的 DHCP 服务器无法提供地址分配服务。
4. 当然，借助于 Wireshark 软件和交换机端口镜像技术，获悉 DHCP 服务器的 MAC 地址也相当容易。

第9章

UDP/TCP故障分析

本章涵盖以下主题：

- ▶ 配置 Wireshark Preferences 窗口内 Protocol 选项下的 UDP 和 TCP 协议参数，为排除 TCP/UDP 相关故障做准备；
- ▶ TCP 连接问题；
- ▶ TCP 重传问题——源自哪里，因何而起；
- ▶ 重复确认和快速重传问题；
- ▶ TCP 报文段失序事件；
- ▶ 与 TCP 窗口有关的问题（zero window、window full、窗口大小改变等事件）；
- ▶ TCP 重置及其原因。

9.1 简介

用户数据报协议（**UDP**）和传输控制协议（**TCP**）的用途是在末端应用程序之间（比如，Web 服务器端和 Web 客户端之间、Email 服务器端和 Email 客户端之间）传递数据。许多应用程序都可以使用这两种协议来传递数据，不同的应用程序可通过端口号来区分。源、目端口号外加源、目 IP 地址便构成了所谓的套接字（socket）。图 9.1 所示为 Web 浏览器是如何利用 TCP 与 Web 服务器建立连接的。Web 服务器会监听从所周知的 TCP 80 端口，当 Web 浏览器访问 Web 服务器时，浏览器所驻留的操作系统会随机分配一个大于或等于 1024 的端口号，作为 TCP 连接的源端口号（比如，1024 端口）。

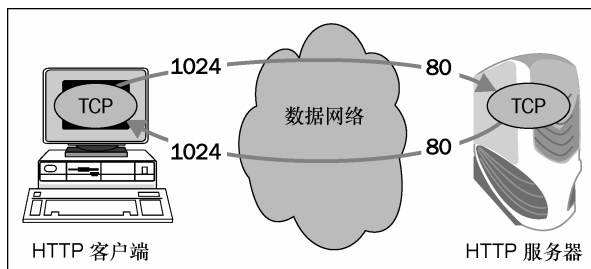


图9.1

收到 TCP 连接建立请求之后，侦听 TCP 80 端口的 Web 服务器会对 TCP 1024 端口进行响应。

TCP 是属于可靠的面向连接型协议；而 UDP 不支持可靠传输，为非面向连接型协议。



注意

还有另外一种第四层协议，名为流控传输协议（SCTP）。该协议被视为 TCP 的改进版本，多用在服务提供商网络。本书不涉及 SCTP。

本章将重点关注 TCP 的运作方式、与之有关的各种故障，以及如何使用 Wireshark 来定位并解决此类故障。

9.2 配置Preferences窗口内protocol选项下的UDP和TCP协议参数，为排除排障做准备

在一般情况下，通过 Wireshark 分析 UDP/TCP 数据包时，无需更改 **Preferences** 窗口中 **Protocol** 选项下的 UDP 和 TCP 协议的默认配置，但并不是说绝对不需要修改。

9.2.1 配置准备

1. 启动 Wireshark 软件，点击 **Edit** 菜单下的 **Preferences** 菜单项。
2. 在弹出的 **Preferences** 窗口中，点击 **Protocol** 配置选项前的+号，在其中选择 UDP 或 TCP 协议进行配置。

9.2.2 配置方法

本节会介绍如何配置 UDP 或 TCP 协议名下的各项参数。

UDP 参数

勾选或取消勾选图 9.2 所示的 4 项 UDP 参数后的复选框，都会对 Wireshark 抓取（或显示）UDP 数据包的方式产生影响。

下面是对图 9.2 中所示四项 UDP 参数的解释。

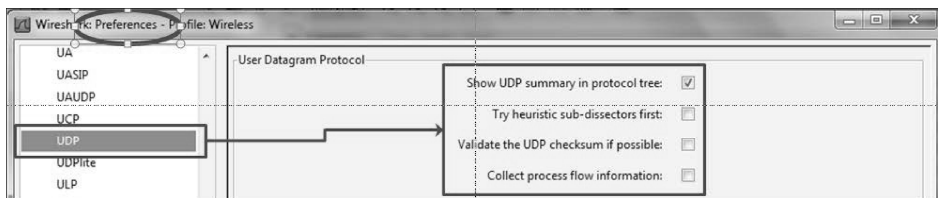


图9.2

- **Show UDP summary in protocol tree:** 与该参数关联的复选框一经勾选（默认勾选），在抓包主窗口的数据包结构（协议树）区域内，凡是 UDP 数据包，Wireshark 软件除了会以 `user datagram protocol` 字样来标明其正身之外，还会显示源、目 UDP 端口号；若取消勾选该复选框，则抓包主窗口的数据包结构（协议树）区域内将不会显示 UDP 数据包的源、目端口号。
- **Try heuristic sub-dissectors first:** 与该参数关联的复选框一经勾选，Wireshark 在调用常规解析器之前，会尝试先用 `dissectors` 解析器来解码 UDP 数据包。
- **Validate the UDP checksum if possible:** 若勾选与该参数关联的复选框，则 Wireshark 将会验证 UDP 校验和。
- **Collect process flow information:** 若勾选与该参数关联的复选框，则 Wireshark 将会采集与（本机某个）进程挂钩的 UDP 流量信息。

在大多数情况下，保持默认设置就足够了。

TCP 参数

下面是对图 9.3 中所示各项 TCP 配置参数的解释。

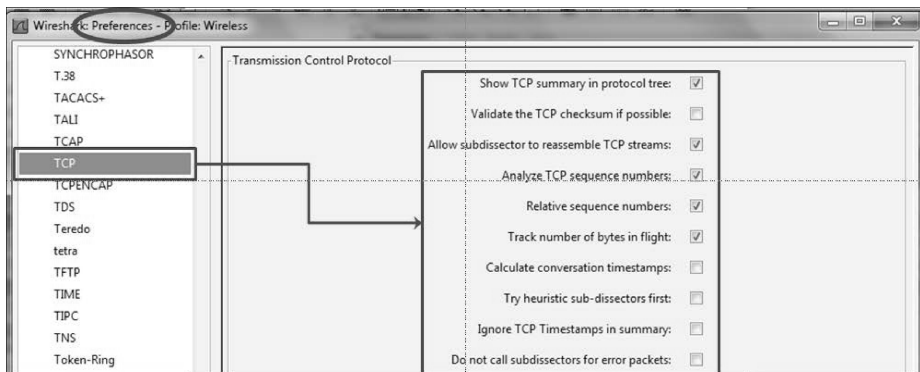


图9.3

- **Show TCP summary in protocol tree:** 与该参数关联的复选框一经勾选（默认勾选），在抓包主窗口的数据包结构（协议树）区域中，凡是 TCP 数据包，Wireshark 软件除了会用 `transmission control protocol` 字样来标明其正身之外，还会简要显示相关 TCP 属性，比如源、目 TCP 端口号、TCP 序列号、TCP 报文段长度等；若取消勾选该复选框，则抓包主窗口的数据包结构（协议树）区域内只会显示 `transmission control protocol` 字样。
- **Validate the TCP checksum if possible:** 与该参数关联的复选框一经勾选，Wireshark 软件的运行效率就会显著下降，在大多数情况下都无需勾选（对该参数的解释详见 1.9.2 节）。
- **Allow subdissector to reassemble TCP streams:** 用来执行 TCP 流分析（默认勾选）。
- **Analyze TCP sequence numbers:** 与该参数关联的复选框一经勾选（默认勾选），Wireshark 就会以分析所抓 TCP 数据包中序列号字段值的方式，来跟踪在传输过程中是否发生过 TCP 重传以及重复确认等现象。Analyze TCP sequence numbers 功能是 Wireshark 最重要的功能之一。
- **Relative sequence numbers:** 对该参数的解释详见 1.9.2 节。
- **Track number of bytes in flight:** 与该参数关联的复选框一经勾选（默认勾选），Wireshark 就会统计已发送至网络但尚未得到确认的 TCP 报文段的字节数。
- **Calculate conversation timestamps:** 对该参数的解释详见 1.9.2 节。
- **Try heuristic sub-dissectors first:** 与该参数关联的复选框一经勾选，Wireshark 将在调用“常规”解析器之前，先尝试使用 `dissectors` 解析器来解码 TCP 数据包。
- **Ignore TCP Timestamps in summary:** 与该参数关联的复选框一经勾选，在抓包主窗口的数据包列表区域中，TCP 数据包的时间戳选项信息就不会露面。
- **Do not call subdissector for error packets:** 若勾选与该参数关联的复选框，则 Wireshark 将不会调用任何一种解析器来解码重传或失序的 TCP 报文段。

9.2.3 幕后原理

再多讲几句与上述 TCP 配置参数有关的话题。

对于 Wireshark 所抓 TCP 数据流中的第一个 TCP 报文段，其（TCP 头部中的）序列号字段值总是以 0 示人，后续 TCP 报文段的序列号字段值将依次递增，这就是所谓的相对序列号（**relative sequence number**）。说其相对，是因为在这第一个 TCP 报文段的 TCP 头部中，序

列号字段的实际值为一个 $0 \sim 2^{32}$ 之间的整数, 由 TCP 进程随机选择, 但 Wireshark 为便于跟踪, 将该值设置并显示为 0。TCP 标准并未对如何选择 (TCP 头部中的) 序列号字段 (的初始) 值做任何硬性规定。

与 **Calculating conversations timestamps** 参数相关联的复选框一经勾选, 在抓包主窗口的数据包结构区域中, 凡是 TCP 数据包, 都会在 **transmission control protocol** 树下多出一个 **Timestamps** 结构, 点击前面的 “+” 号, 就能看到 Wireshark 记录的该 TCP 数据包在其所归属的 TCP 数据流中的与时间戳有关的信息^①。在 TCP 数据包的 TCP 头部中, 有多个可选项 (选项字段), 其中有一个叫做时间戳选项, 其格式如图 9.4 所示^②。

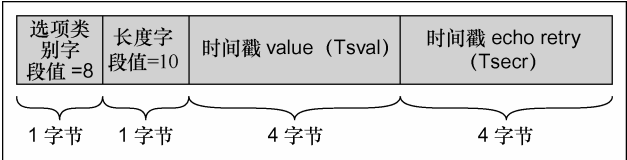


图9.4

可通过观察 TCP 数据流中各个 TCP 报文段的时间戳选项中的 Tsval 和 Tsecr 字段值, 来了解时间敏感型 TCP 应用程序在延迟方面的变化。以下所列为 RFC 1323 记载的 TCP 头部中时间戳选项的用法。

- 发送方在发出 TCP 报文段时, 可在其 TCP 头部的时间戳选项的 Tsval 字段内, 填入一个 32 位的值。
- 接收方在收到上述 TCP 报文段之后, 发出 ACK TCP 报文段进行确认时, 需在其 TCP 头部的 Tsval 字段内, 原封不动地填入那个 32 位值。

这样一来, 发送方每次收到由接收方发出的 ACK TCP 报文段 (对应于发送方发出的每一个 TCP 报文段) 时, 便可以精确计算出数据包的往返时间 (RTT) 了。这样的机制被称为往返时间测量机制 (**Round Trip Time Measurement, RTTM**)。

9.2.4 拾遗补缺

UDP 是一种非常简单的协议, 其头部也很简单, 只有 4 个字段: 源端口号字段、目的端口号字段、数据包长度字段以及校验和字段。收到 UDP 数据包时, 接收方会根据校验和字段值, 来判断是丢弃还是接受。UDP 未针对数据包在传输途中所遭遇的变故, 来设立

① 译者注: 原文是 “The calculating conversations timestamps refers to the timestamp option of the TCP packet” 。
② 译者注: TCP 头部中的时间戳选项, 并不是勾选 **Calculating conversations timestamps** 复选框后, 在数据包结构区域的 **transmission control protocol** 树下暴露出的 **Timestamps** 结构。

任何重传或恢复机制，但有些基于 UDP 的应用层协议（比如，DNS 协议）内置有重传或恢复机制。

与 UDP 相比，TCP 要复杂的多，它是一种面向连接的可靠数据传输协议，内置有序列号机制、流控机制以及拥塞避免机制。本章会对 TCP 协议的上述机制做深入探讨。

SCTP 也是一种可靠的面向连接的协议，支持单连接、多数据流的传输方式，还可以选择将多条用户消息打包进单个 SCTP 数据包。此外，该协议还内置有 cookie、多宿主及其他机制。开发 SCTP 协议的初衷是要利用其在蜂窝网络中承载信令消息，后来则被改进为某些应用程序的传输层协议。

9.3 TCP连接故障

一对 TCP 进程在彼此通信时，会先建立 TCP 连接，通过 TCP 连接发送数据，数据发送完毕之后再关闭 TCP 连接。用浏览器浏览网页、通过邮件客户端连接邮件服务器、利用 Telnet 工具访问路由器，或使用基于 TCP 的其他应用程序时，都需要经历上述过程。

建立 TCP 连接时，客户端会从 TCP 源端口向服务器端 TCP 目的端口发出连接建立请求^①。

在建立或关闭 TCP 连接的过程中，可能会发生某些问题^②。本节的主要目标就是要利用 Wireshark，来分析并解决这些问题。

9.3.1 准备工作

对于下列任一故障，都可以借助于 Wireshark 来分析其起因。

- 尝试运行某种（基于 TCP 的）应用程序，但其根本无法运行。比如，尝试用浏览器打开网页，但一无所获^③。
- 试图用 Email 客户端收取邮件，但却连不上 Email 服务器^④。
- 导致故障的原因可能很简单（比如，服务器宕机、服务器上相关服务未启动、通向服务器所在 IP 子网的路由失效等），也可能很复杂（比如，DNS 故障、IP 地址冲突、服务器内存因不足而无法接受客户端所发起的连接等）。

① 译者注：原文是 “When TCP opens the connection, it sends a request for open connection from the source port to destination port”。原文奇差无比。

② 译者注：原文是 “Some problems can occur during the establishment or closing of the application”。这种文字又是作者不动脑子的产物。

③ 译者注：原文是 “You try to run an application and it does not work. You try to browse the Internet and you don't get any response”。作者行文无任何思路。

④ 译者注：这与上面那个打不开网页的故障有何区别？

本节将聚焦于如何借助 Wireshark 来解决上述故障。

9.3.2 排障方法

本小节将重点关注使用 Wireshark 诊断 TCP 连接故障时，应遵循的排障思路。TCP 连接故障通常会导致某些应用程序无法运行。

当某种应用程序的客户端软件（比如，数据库客户端软件、E-mail 客户端软件或视频监控客户端软件等）连接不上其服务器端时，应尝试按以下步骤来解决故障^①。

1. 验证（安装服务器端软件的）主机和服务器端软件是否都能正常运行。
2. 验证客户端软件是否能正常运行，检查（安装客户端软件的）主机是否连网正常，检查该主机的 IP 地址是否配置正确（一般而言，主机的 IP 地址配置分手工配置和 DHCP 自动分配两种）。
3. 在（安装客户端软件的）主机上 ping 服务器主机，验证 IP 连通性能否正常建立。



注意

在某些情况下，应用程序的客户端软件能正常连接其服务器端，但安装客户端软件的主机却 ping 不通安装服务器端软件的主机。这极有可能是因为有防火墙从中作祟，设在防火墙上的安全策略可能会封锁 ICMP 流量，也就是说，ping 不通安装服务器端软件的主机，并不表示应用程序就无法正常运行。所谓的防火墙，有可能是专门的硬件防火墙，也有可能是安装在主机上的软件防火墙。

4. 在运行客户端软件的同时，启动 Wireshark 开始抓包分析。首先，应仔细观察在抓包主窗口数据包列表区域中露面的 TCP 数据包，看看是否存在以下现象。
 - 客户端主机连发三次 SYN 位置 1 的 TCP 报文段，但服务器端主机未作任何回应。
 - 客户端主机发出了 SYN 位置 1 的 TCP 报文段之后，收到了 RST 位置 1 的 TCP “确认”报文段。

只要发现了以上两种现象中的任何一种，那么不是应用程序自身的问题（比如，应用程序的服务器端软件没有启动），就是有防火墙从中作祟，封锁了相关流量。

由图 9.5 所示的 Wireshark 抓包主窗口的截屏可知，IP 地址为 10.0.0.3 的内网主机无法访问 IP 地址为 81.218.31.171 的 Web 服务器（详见编号为 61~63 的数据包，这 3 个数据包由内网主机 10.0.0.3 发出，全都是 SYN 位置 1 的 TCP 报文段）。这既有可能是因为有防火墙

^① 译者注：原文是 “When you try to run an application, for example, a database client, a mail client, watching cameras servers, and so on, and you don't get any output, follow these steps”。

从中作祟，也有可能是因为 Web 服务器出了故障。通过此图还可以得知，同一台主机可以正常访问另外一台 IP 地址为 108.160.163.43 的 Web 服务器（详见编号为 65~67 的数据包）。现在可以得出结论，IP 地址为 10.0.0.3 的内网主机并不是所有网站都不能访问，只是不能访问 IP 地址为 81.218.31.171 的网站。

无法访问 IP 地址为 81.218.31.171 的网站，连 TCP 连接都建立不了 (SYN/SYN/SYN)				
60	4.994	51000	10.0.0.138	
61	5.088	214000	10.0.0.3	81.218.31.171
62	5.090	244000	10.0.0.3	81.218.31.171
63	5.178	158000	10.0.0.3	81.218.31.171
64	6.247	500000	10.0.0.3	173.194.78.125
65	6.449	442000	10.0.0.3	108.160.163.43
66	6.480	809000	108.160.163.43	10.0.0.3
67	6.480	936000	10.0.0.3	108.160.163.43
68	6.481	512000	10.0.0.3	108.160.163.43
69	6.512	241000	108.160.163.43	
70	6.512	988000	108.160.163.43	
可以正常访问 IP 地址为 108.160.163.43 的网站 (Wireshark 显示 TCP 连接可以正常建立, SYN/SYN-ACK/ACK)				
<pre> 62 51910 > http [SYN] Seq=0 win=8192 62 51909 > http [SYN] Seq=0 win=8192 62 51912 > http [SYN] Seq=0 win=8192 66 51919 > xmpp-client [SYN] Seq=0 win=8192 66 51921 > http [SYN] Seq=0 win=8192 66 http > 51921 [SYN, ACK] Seq=0 Ack= 54 51921 > http [ACK] Seq=1 Ack=1 win 543 GET /subscribe?host_int=340855826& http > 51921 [ACK] Seq=1 Ack=290 W HTTP/1.1 200 OK (text/html) </pre>				

图9.5

再举一个类似的 TCP 连接故障示例，这个故障要更加复杂。用户需要登录一台 IP 地址为 135.82.12.1 的视频监控服务器，去监控远程站点。故障现象是：用户在内网主机 10.0.0.3 的浏览器内输入该视频监控服务器的 IP 地址 135.82.12.1 时，可以看到登录界面，但却无法登录进系统。由图 9.6 所示 Wireshark 抓包主窗口的截屏可知，内网主机 10.0.0.3 已经建立了通往目的 IP 地址 135.82.12.1 TCP 80（HTTP）端口的连接，未出现任何问题。

Filter: ip.addr==135.82.12.1		Expression... Clear Apply Save				
No.	Time	Source	Destination	Protocol	Length	Info
2113	17.665372	10.0.0.3	135.82.12.1	TCP	66	62423 > http [SYN] Seq=0 win=8:
2120	17.746627	135.82.12.1	10.0.0.3	TCP	66	http > 62423 [SYN, ACK] Seq=0 ,
2121	17.746693	10.0.0.3	135.82.12.1	TCP	54	62423 > http [ACK] Seq=1 Ack=1
2122	17.747085	10.0.0.3	135.82.12.1	HT	316	GET / HTTP/1.1
2130	17.862143	135.82.12.1	10.0.0.3	54	http > 62423 [ACK] Seq=1 Ack=21	
2189	18.736301	135.82.12.1	10.0.0.3	已正常建立了通往目的 IP 地址 135.82.12.1 的 TCP 80 (HTTP) 端口的连接		
2191	18.767301	135.82.12.1	10.0.0.3	[TCP segment of a reassembled I		
				[TCP segment of a reassembled I		

图9.6

可只要在那份抓包文件中筛选出目的 IP 地址为 135.82.12.1（视频监控服务器）的所有流量，便可以发现，内网主机 10.0.0.3 还在尝试建立通往目的 IP 地址 135.82.12.1 TCP 6036 端口的连接。

由图 9.6 所示 Wireshark 抓包主窗口的截屏可知，内网主机 10.0.0.3 在尝试建立通往目的 IP 地址 135.82.12.1 TCP 6036 端口的连接时，连接被重置（IP 地址为 135.82.12.1 的视频监控服务器回复的是 RST 位和 ACK 位同时置 1 的 TCP 报文段）。原来，那台视频监控服务器的用户名/密码认证功能要通过 TCP 6036 端口来完成，而网络中设于防火墙上的安全策略阻断了

TCP 目的端口号为 6036 的流量，但却没有阻断 TCP 目的端口号为 80 的流量。

还有一种情况，也有可能会导致该故障，那就是客户机需要通过端口地址转换（PAT），才能访问到那台视频监控服务器，但（防火墙或路由器）被配置为只转换目的端口号为 TCP 80 端口的流量，不转换目的端口号为 TCP 6036 端口的流量（见图 9.7）。

No.	Time	Source	Destination	Protocol	Length	Info
2620	36.423135	10.0.0.3	135.82.12.1	TCP	54	62438 > http [ACK] Seq=915
2621	36.423135	10.0.0.3	135.82.12.1	TCP	66	62442 > 6036 [SYN] Seq=0 W
2622	36.423135	135.82.12.1	10.0.0.3	TCP	54	6036 > 62442 [RST, ACK] Seq
2623	36.423135	10.0.0.3	194.90.1.5	ICMP	74	Echo (ping) request id=0x0
2624	36.423135	194.90.1.5	10.0.0.3	ICMP	74	Echo (ping) reply id=0x0
2626	37.329129	10.0.0.3	135.82.12.1	TCP	62	62442 > 6036 [SYN] Seq=0 W
2627	37.369547	135.82.12.1	10.0.0.3	TCP	54	6036 > 62442 [RST, ACK] Seq
2628	38.023274	10.0.0.3	194.90.1.5	ICMP	74	Echo (ping) request id=0x0

图9.7

简而言之，若某款基于 TCP/UDP 协议的应用软件无法正常使用，请确保支撑该软件的客户端及服务器端程序运行的所有 TCP/UDP 端口，没有被任何网络设备（如防火墙、路由器等）或其他应用程序（如瑞星、360 等）封锁。



注意

当有新的应用程序上线运行时，最好在（安装其）客户端（程序）和服务端（程序）的主机上分别安装 Wireshark，将支撑该应用程序运行的所有 TCP/UDP 端口都了解清楚。只有如此，才能降低防火墙误拒流量事件发生的概率，软件开发人员未必会告诉你应用程序在网络层面的所作所为（有时，他们自己可能也搞不清楚）。

9.3.3 幕后原理

建立 TCP 连接时，要经历三次握手过程，如图 9.8 所示。

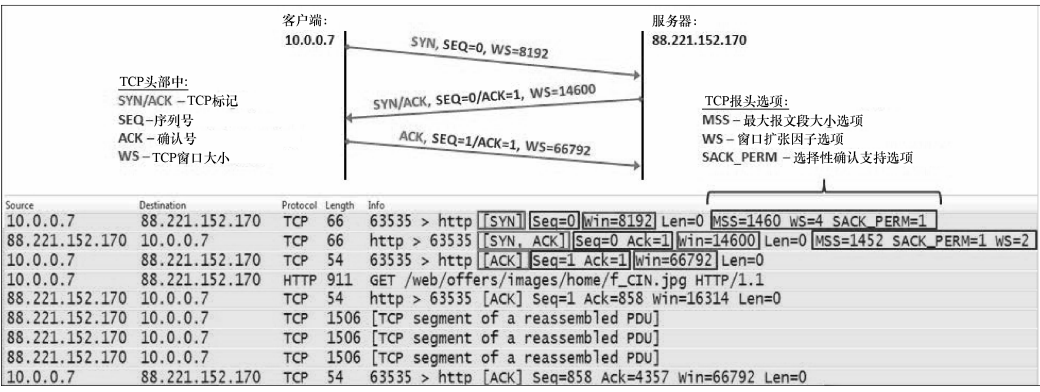


图9.8

1. TCP 客户端发出 SYN 标记位置 1 的 TCP 报文段，以期建立 TCP 连接。在此报文段中：
 - 会指明本方 TCP 初始序列号，即 TCP 客户端发往 TCP 服务器的首字节的编号；
 - 会指明本方窗口大小，即客户端操作系统分配给 TCP 进程的缓存大小（内存空间）；
 - 会（让 TCP 头部）携带某些选项，比如，MSS 或选择性确认选项等。
2. 收到 TCP 客户端发出的 TCP 连接请求报文段后，TCP 服务器会发出 SYN 和 ACK 标记位同时置 1 的 TCP 报文段，进行确认。在此报文段中：
 - 会指明本方 TCP 初始序列号，即 TCP 服务器发往 TCP 客户端的首字节的编号；
 - 会指明本方窗口大小，即服务器端操作系统分配给 TCP 进程的缓存大小（内存空间）；
 - 会根据 TCP 客户端传递过来的（TCP 头部中的）选项，设置自己的 TCP 头部的各个选项。
3. 收到 TCP 服务器发出的 SYN 和 ACK 标记位同时置 1 的 TCP 报文段后，TCP 客户端会发出 ACK 标记位置 1 的 TCP 报文段，进行确认。在此报文段中：
 - 会指明本方窗口大小，即客户端操作系统分配给 TCP 进程的缓存大小。尽管已在首个（SYN 标记位置 1 的）TCP 报文段中通告过该参数，但该参数对服务器至关重要，服务器必须实时掌握客户端的窗口大小，反之亦然。

在建立三次握手过程中，经常会在 TCP 头部中露面的选项如下所列。

- **最长报文段（Maximum Segment Size, MSS）**：建立 TCP 连接时，客户端和服务端通常都会在（本方发出的）SYN 标记位置 1 的报文段中包含这一选项，向对方宣告自己期望接收的 TCP 报文段的最大长度。所谓“TCP 报文段的最大长度”，（MSS 值）只包括了 TCP 数据部分，不包括 IP 包头和 TCP 头部^①。
- **窗口扩张（Windows Scale, WSOPT）^②**：有了 WSOPT 选项，就能定义一个作用于 16 位“窗口大小”字段的“扩张因子”（scale factor），以起到大幅提高“窗口大小”字段

① 译者注：原文是“This is the maximum size of the TCP datagram, that is, the number of bytes from the beginning of the TCP header to the end of the entire packet”。对于原文，译者只能说“呵呵”了。

② 译者注：原文是“Windows Size (WSopt)”。对作者无语。

容量的目的。该选项包含了一个 8 位 shift count 字段，取值范围为 0~14。WSOFT 选项只能在 SYN 标记位置 1 的 TCP 报文段中露面，若 shift count 字段值为 3，则“窗口大小”的“扩张因子”为 2^3 ，这就使得“窗口大小”字段的容量暴增 8 倍。

- **选择性确认支持（SACK-Permitted）和选择性确认（SACK）：** SACK-Permitted 选项用来表示 TCP 报文段的发送方是否支持 SACK 功能，只会在 TCP 三次握手的前两个 TCP 报文段中露面。SACK 选项则用来通知 TCP 发送方，本方已接收并缓存了非连续数据块，发送方可根据 SACK 选项中的信息来判断究竟是哪个（些）数据块传丢，从而重传相应的数据块。
- **时间戳（Timestamps options，TSopt）：** 本章前文已对该选项做过详细介绍。

三次握手结束时，TCP 客户端和 TCP 服务器：

- 会一致同意建立 TCP 连接；
- 能获悉对方（所传数据的）初始序列号值；
- 能得知对方可用来接收数据的缓存容量（TCP 窗口大小）。



注意

建立 TCP 连接时，三次握手缺了任何一次，TCP 连接都无法成功建立，这包括：客户端在发出第一个（SYN 标记位置 1 的）TCP 报文段之后，收不到服务器端发出的 SYN 和 ACK 标记位同时置 1 的 TCP 报文段，或收到了服务器端发出的 RST 位置 1 的 TCP 报文段；在前两次握手成功之后，服务器收不到客户端发出的 ACK 标记位置 1 的 TCP 报文段等情形。

9.3.4 拾遗补缺

以下是作者排除 TCP 相关故障的经验总结。

- 当 TCP 客户端发出 SYN 标记位置 1 的 TCP 报文段，请求建立 TCP 连接时，若收到了 RST 标记位置 1 的 TCP 报文段，需检查是不是防火墙封掉了该 TCP 客户端所要连接的 TCP 目的端口。
- TCP 客户端在发起 TCP 连接时，若连发三次 SYN 标记位置 1 的 TCP 报文段都未得到回应，则不是应用程序的服务器端（软件或主机）未能正常运作，就是防火墙封掉了该 TCP 客户端所要连接的 TCP 目的端口^①。
- 当出现第四层连通性故障时，请务必核实网络中是否部署有开启了 NAT 或 PAT 功能

^① 译者注：原文是“Triple SYN without any answer occurs either due to an application that didn't respond, or a firewall that blocks the request on a specific port”。

的设备，这些设备可能会干扰 UDP 或 TCP 的正常运作。

9.4 TCP重传现象——源头及原因

当 TCP 发送方发出一或数个 TCP 报文段后，便会坐等 TCP 接收方对相关报文段的确认，若等不来确认报文段，TCP 发送方便会重新传送。显而易见，若 TCP 发送方重新传送（已经发出过的）TCP 报文段，则表明相应的 TCP 报文段未被 TCP 接收方接收，或 TCP 接收方发出的确认报文段传丢。导致 TCP 重传的原因很多，发现 TCP 重传因何而起将是本节的重点。

9.4.1 准备工作

当用户普遍反应某种或某些基于 TCP 的应用程序的运行速度明显变慢时，TCP 重传可能罪魁祸首之一^①。此时，应借助于端口镜像技术，用 Wireshark 抓取相关应用程序客户端或服务端所驻留的主机的流量，并进行流量分析。

本节会介绍如何使用 Wireshark 来帮助定位和解决 TCP 重传问题。

9.4.2 操作方法

1. 启动 Wireshark 主机，选择正确的网卡抓取流量。
2. 点击 **Analyze** 菜单下的 **Expert Info** 菜单项。
3. 在弹出的 Expert Infos 窗口中，点击 **Notes** 标签，看看 Wireshark 有没有感知到 TCP 重传事件（即看看是否出现含 **Retransmissions** 字样的子事件）。
4. 若有，请点击与 **Retransmissions** 字样相关联的子事件之前的+号，再单击其名下的每一行，便可以在抓包主窗口的数据包列表区域定位到相应的 TCP 重传数据包。
5. 现在，作者要问：TCP 重传是因何而起呢？



注意

用 Wireshark 从某条通信链路或服务器上的某块网卡抓取流量时，捕获到的流量会涉及众多 IP 地址，牵涉到多种应用程序，甚至是同一种应用程序的不同访问模式（比如，同一数据库的不同查表模式）。此时，最重要的是要找到发生 TCP 重传现象的具体的 TCP 数据流。

^① 译者注：原文是 “When you see that the network becomes slow, one of the reasons for this can be retransmissions”。作者连问题都描述不清，凭什么去解决问题？。

6. 可通过以下三种方法来查明 TCP 重传的源头。

- 在 Expert Info 窗口的 **Notes** 标签栏中, 对于 TCP 重传子事件名下的每一行, 在抓包主窗口的数据包列表区域都能定位到与其相对应的 TCP 重传数据包 (适合有经验的网管人员)。
- 在 Wireshark 抓包主窗口的 Filter 输入栏内, 输入显示过滤表达式 `expert.message == "Retransmission (suspected)"`, 然后点击 **Apply** 按钮。Wireshark 会立刻筛选出抓包文件中所有的 TCP 重传数据包。
- 在应用过上述过滤表达式之后, 点击 **Statistics** 菜单下的 **Conversations** 菜单项, 在弹出的 Conversations 窗口中, 勾选底部的 **Limit to display filter** 复选框, 让 Wireshark 显示出存在 TCP 重传现象的所有 IP 及 TCP 会话。

案例分析 1: 发生在单一源 IP 地址和众多目的 IP 地址之间的 TCP 重传

由图 9.9 所示的 Wireshark 截屏可知, IP 地址为 10.0.0.5 的主机在访问 Internet 时 (即连接多个 Internet Web 站点的 TCP 80 端口时) 发生了多次 TCP 重传现象。由于 TCP 重传的方向单一 (全都是由主机 10.0.0.5 发起), 因此可以断定, 主机 10.0.0.5 向众多 Internet Web 站点发出 TCP 报文段之后, 并未收到相应的确认报文段。这有可能是因为主机 10.0.0.5 发出的 TCP 报文段在 Internet 链路上传丢, Internet Web 站点由于无法收到这些报文段, 故而未能确认, 也有可能是 Internet Web 站点发出的确认报文段在 Internet 链路上传丢, 主机 10.0.0.5 收不到确认报文段, 误以为自己发出的 TCP 报文段传丢, 而发起了 TCP 重传。



No.	Time	Source	Destination	Protocol	Length	Info
96	29.203230000	10.0.0.5	212.199.184.51	HTTP	366	[TCP Retransmission] GET /iefilter.html HTTP/1.1
98	29.312500000	10.0.0.5	94.127.73.180	TCP	783	[TCP Retransmission] 55317 > http [PSH, ACK]
99	29.343596000	10.0.0.5	77.234.43.92	TCP	638	[TCP Retransmission] 55310 > http [PSH, ACK]
100	29.390393000	10.0.0.5	81.218.31.136	HTTP	491	[TCP Retransmission] GET /24x24/30.png HTTP/1.1
101	29.390393000	10.0.0.5	108.168.157.82	TCP	1506	[TCP Retransmission] 55326 > http [ACK] Seq=
103	29.437199000	10.0.0.5	74.125.232.145	TCP	1070	[TCP Retransmission] 55320 > http [PSH, ACK]
106	29.530780000	10.0.0.5	74.125.232.145	TCP	1071	[TCP Retransmission] 55321 > http [PSH, ACK]
110	29.811579000	10.0.0.5	212.199.184.51	HTTP	366	[TCP Retransmission] GET /iefilter.html HTTP/1.1
114	30.513564000	10.0.0.5	94.127.73.180	TCP	590	[TCP Retransmission] [TCP segment of a reasse
115	30.544753000	10.0.0.5	77.234.43.92	TCP	590	[TCP Retransmission] [TCP segment of a reasse
116	30.591557000	10.0.0.5	81.218.31.136	HTTP	491	[TCP Retransmission] GET /24x24/30.png HTTP/1.1
117	30.638349000	10.0.0.5	74.125.232.145	TCP	1071	[TCP Retransmission] [TCP segment of a reasse
118	30.638366000	10.0.0.5	108.168.157.82	TCP	1071	[TCP Retransmission] 55326 > http [ACK] Seq=
121	30.731954000	10.0.0.5	74.125.232.145	TCP	1071	[TCP Retransmission] [TCP segment of a reasse
125	31.012738000	10.0.0.5	212.199.184.51	HTTP	366	[TCP Retransmission] GET /iefilter.html HTTP/1.1
158	31.387223000	10.0.0.5	81.218.31.136	HTTP	489	[TCP Retransmission] GET /shadow.png HTTP/1.1
159	31.387356000	10.0.0.5	81.218.31.136	HTTP	503	[TCP Retransmission] GET /save-center/rollover

图9.9

显而易见, 一定是 Internet 链路出了问题。该如何印证这一观点呢?

1. 点击 **Statistics** 菜单下的 **IO Graph** 菜单项。

2. 观察 IO Graphs 窗口生成的流量统计信息图，如图 9.10 所示。

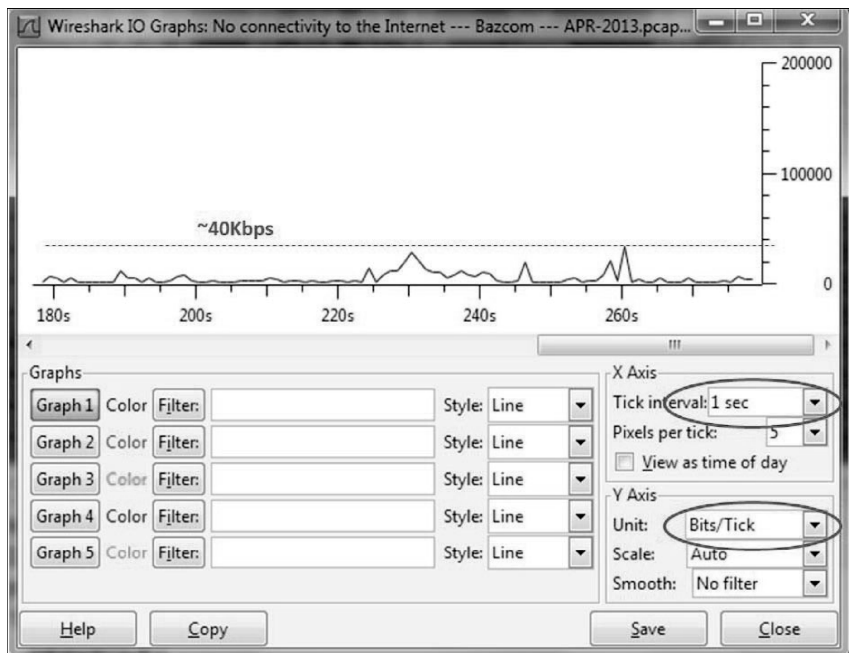


图9.10

对于本例，IO Graphs 窗口生成的流量统计信息图表明，在使用 Wireshark 抓包时，Internet 链路的负载几乎为空，之所以会出现这种情况，既有可能是本方 Internet 链路中断，也有可能是 ISP 网络内的上游链路中断。

3. 可登录用来连接 Internet 链路的设备或通过 SNMP 网管软件，来了解连接 Internet 链路的（路由器/交换机）接口的状态及丢包情况。当然，要想通过 SNMP 网管软件了解相关信息，还需在网络设备上激活 SNMP 代理功能。

案例分析 2：发生在同一条 TCP 连接中的 TCP 重传

若 TCP 重传现象只发生在同一股 TCP 数据流之中，则预示着某种应用程序运行速度缓慢，如图 9.11 所示。

要借助 Wireshark 来分析发生在同一条 TCP 连接中的 TCP 重传问题，请按以下步骤行事。

1. 再次点击 **Statistics** 菜单下的 **Conversations** 菜单项，在弹出的 Conversations 窗口中，仍然勾选底部的 **Limit to display filter** 复选框。拜图 9.10 所示 Wireshark 抓包主窗口的 Filter 输入栏内的显示过滤器所赐，在 Conversations 窗口中只会出现一条存在 TCP

重传问题的会话。

No.	Time	Source	Destination	Protocol	Length	Info
239	19.426155	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
240	19.427305	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
241	19.441252	10.1.1.200	10.90.30.12	TCP	60	64800 → 64800 [RST] Seq=121
252	20.319262	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
259	20.420397	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
269	21.275252	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
270	21.276400	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
288	22.323076	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
290	22.488386	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
291	22.489537	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
292	22.522244	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
301	23.124370	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
303	23.363588	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
314	24.025662	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121
315	24.124575	10.90.30.12	10.1.1.200	TCP	60	64800 → 64800 [RST] Seq=121

图9.11

2. 点击 Conversations 窗口中的 **IPv4** 标签栏，便可获知发生 TCP 重传问题的那两台主机的 IP 地址，如图 9.12 所示。

Address A	Address B	Packets	Bytes	Packets A-B	Bytes A-B	Packets B-A	Bytes B-A	Rel Start	Duration	bps A-B	bps B-A
10.1.1.200	10.90.30.12	218	270 572	10	1 362	208	269 210	0.962677000	88.6572	122.90	24292.21

图9.12

3. 点击 Conversations 窗口中的 **TCP** 标签栏，便可获知 TCP 重传发生在哪些源、目端口号之间，如图 9.13 所示。

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A-B	Bytes A-B	Packets B-A	Bytes B-A	Rel Start	Duration	bps A-B	bps B-A
10.1.1.200	2390	10.90.30.12	1181	188	265 608	0	0	188	265 608	1.993708001	87.4711	N/A	24292.19
10.1.1.200	1972	10.90.30.12	1184	8	1 591	2	389	6	1 202	30.834527001	15.7228	197.93	611.59
10.1.1.200	1972	10.90.30.12	1189	1	591	0	0	1	591	1.027 77.301798001	12.3181	240.95	666.99
10.1.1.200	1972	10.90.30.12	1182	1	591	0	0	1	591	18.917254000	5.2073	N/A	941.75
10.1.1.200	1972	10.90.30.12	1186	1	591	0	0	1	591	52.665895000	5.3078	N/A	1055.04
10.90.30.12	1178	10.1.1.200	21	4	384	1	60	3	324	0.962677000	88.2566	N/A	29.37

图9.13

要想进一步分析 TCP 重传到底因何而起，请按以下步骤行事。

1. 借助于 Wireshark 软件 Statistics 菜单下的 IO Graph 工具，来检测通信链路是否拥塞。



注意

当使 IO Graph 工具来判断通信链路的负载状况时，其图形显示区域内表示流量的曲线几乎呈一条直线，且接近受监控链路的带宽阈值，则可以说明通信链路的负载极高。否则，IO Graph 工具的图形显示区域内表示流量的曲线会起起伏伏，落差很大。

2. 若通信链路负载不高，则应该去找找 TCP 服务器端的原因。对于本例，TCP 服务器端（主机）和客户端（主机）的 IP 地址为 10.1.1.200 和 10.90.30.12。通过图 9.13 可以看出，TCP 重传大多是由 TCP 客户端 10.90.30.12 发起，因此可以初步断定 TCP 服务器端反应慢。
3. 通过图 9.11 可以看出，Wireshark 检测到了存在 TCP 重传问题的 TCP 连接所承载的应用为 FTP-DATA。通过图 9.13 可以看出，FTP 服务器（IP 地址为 10.1.1.200）是以 FTP passive（被动）模式运行。此外，还可以判断出，FTP 客户端 10.90.30.12 已先行打开了一条目的 IP 地址为 10.1.1.200，目的 TCP 端口为 2350 的 TCP 连接，以期执行 FTP 数据上传。但不知何故，又再次建立了一条目的 IP 地址为 10.1.1.200，目的 TCP 端口为 1972 的 TCP 连接。FTP 客户端 10.90.30.12 通过这条 TCP 连接执行 FTP 数据上传时，发生了多次重传。于是，可以断定是 FTP 服务器（IP 地址为 10.1.1.200）反应慢，导致了多次 TCP 重传（实际情况也正是如此）。

案例分析 3：有规律可循的 TCP 重传现象

当发现 TCP 重传问题时，应仔细观察所有的 TCP 重传报文段是否都具有相同的特征，这一点非常重要。

由图 9.14 所示的 Wireshark 截屏可知，Wireshark 感知到的所有 TCP 重传现象均发生于同一条 TCP 连接。都是由 IP 主机 192.168.1.21（TCP 服务器端）从 TCP 源端口号 139（NetBIOS 会话服务[NetBIOS Session Service]端口），向 IP 主机 192.168.1.99（TCP 客户端）的 TCP 目的端口号 1064 发起的 TCP 重传。

这看起来很像是应用程序服务器端问题，但回过头再看 Wireshark 抓包主窗口的数据包列表区域所显示的所有 TCP 重传报文段，便可以发现一些很有意思的事情，如图 9.15 所示。

由图 9.15 可知，所有 TCP 重传报文段在时间间隔上都有规律可循——每隔约 30 毫秒便发生一次 TCP 重传。作者已经把 Wireshark 抓包主窗口的数据包列表区域中 Time 一栏的显示格式，更改为了“seconds since the previously displayed packet”。



图9.14



图9.15

对于本例，导致 TCP 重传的真正原因是，IP 主机 192.168.1.99（TCP 客户端）上运行的财务软件使得该主机每隔 30~36 秒变慢一次。

因应用程序故障而导致的 TCP 重传

当应用程序的客户端或服务端发生故障时，也会导致 TCP 重传问题。在这种情况下，通过 Wireshark 抓包，可观察到：在相关 TCP 连接中，会连续发生 5 次 TCP 重传，但每次 TCP 重传的时间间隔各不相同。连续发生 5 次 TCP 重传之后，发起 TCP 重传的主机就会认为 TCP 连接中断。此时，发起 TCP 重传的主机或许还会发出 RST 位置 1 的 TCP 报文段，尝试关闭

TCP 连接（视具体的应用程序的实现方法而定）。TCP 连接中断之后，应该会发生以下两件事之一。

- TCP 客户端发出 SYN 位置 1 的 TCP 报文段，试图重新建立 TCP 连接。对于实际操作应用程序的最终用户而言，可能会暂时感觉到软件动弹不得，10~15 秒之后将会恢复正常。
- TCP 客户端不发 SYN 位置 1 的 TCP 报文段，最终用户必须重新启动应用程序（或重启应用程序的某个功能）。

图 9.16 所示为 5 次 TCP 重传之后，TCP 客户端与服务器端之间通过三次握手重新建立 TCP 连接的过程。

No.	Time	Source	Destination	Protocol	Length	Info
1159	0.000406	192.168.201.93	192.168.3.50	TCP	60	http > tscchat [ACK] Seq=220556 Ack=29208
1160	0.220322	192.168.201.93	192.168.3.50	TCP	590	[TCP Retransmission] [TCP segment of a reassembled PDU]
1161	0.656270	192.168.201.93	192.168.3.50	TCP	590	[TCP Retransmission] [TCP segment of a reassembled PDU]
1162	1.203085	192.168.201.93	192.168.3.50	TCP	590	[TCP Retransmission] [TCP segment of a reassembled PDU]
1163	2.406248	192.168.201.93	192.168.3.50	TCP	590	[TCP Retransmission] [TCP segment of a reassembled PDU]
1164	4.812443	192.168.201.93	192.168.3.50	TCP	590	[TCP Retransmission] [TCP segment of a reassembled PDU]
1165	9.625596	192.168.201.93	192.168.3.50	TCP	62	agentview > http [SYN] Seq=0 win=65535 Len=0
1166	0.004414	192.168.3.50	192.168.201.93	TCP	60	http > agentview [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
1167	0.000033	192.168.201.93	192.168.3.50	TCP	54	agentview > http [ACK] Seq=1 Ack=1 Win=65535 Len=0
1168	0.000164	192.168.201.93	192.168.3.50	TCP	90	[TCP segment of a reassembled PDU]
1169	0.000070	192.168.201.93	192.168.3.50	TCP	62	[TCP segment of a reassembled PDU]

图 9.16

因抖动而导致的 TCP 重传

TCP 颇能容忍延迟，但却不能容忍抖动（抖动是指每个 TCP 报文段之间的延迟变化）。当网络发生抖动时，便会触发 TCP 重传。要想得知 TCP 重传是不是因抖动而起，请按以下步骤行事。

1. ping TCP 连接的目标端 IP 地址，观察 ping 命令输出中“time”值的变化情况，获得通信链路延迟（及延迟变化）的第一手信息。

2. 检查导致抖动的具体原因，抖动可能会由以下原因所致。

- 链路拥塞或链路状态不稳定。此时，可通过 ping 命令的输出了解延迟的变化情况。链路带宽过低以及链路时通时断都属于链路状态不稳定。
- 安装应用程序服务器端软件的主机资源不足或硬件配置太低。此时，TCP 重传只会发生在与某种应用程序相关联的 TCP 连接之中。
- 网络设备过载（CPU 或内存资源不足）。此时，需登录网络设备，来了解其资源使用情况。

- 借助于第 13 章介绍的内置于 Wireshark 软件的各种工具，来查找原因。

**注意**

（TCP 报文段的）重传是 TCP 协议的天性之一，但前提是重传的频率不能太高。当重传的 TCP 报文段达到（TCP 报文段总数的）0.5%时，就会对性能产生严重影响；若达到了 5%，相应的 TCP 连接将会中断。当然，这还要取决于具体的应用程序对 TCP 重传的敏感程度。

解决 TCP 重传问题的思路

当在某条链路（是指 Internet 链路、连接安装了 TCP 服务器端软件的主机的链路或连接某个远程站点网络的 WAN 链路）上发现存在 TCP 重传现象时，应按以下思路解决问题。

- 归纳总结：TCP 重传总是与某台特定主机（某个 IP 地址）、某条特定 TCP 连接或某种具体的应用程序相关联吗？
- 逐一排除：TCP 重传是由链路状态不稳定（或链路丢包）、TCP 服务器端或客户端主机反应慢、应用程序反应慢或是其他原因所导致吗？
- 若上述原因均不存在，请检查网络中是否存在抖动现象。

9.4.3 幕后原理

本小节将介绍 TCP 的常规运行方式，以及运行时可能会发生的问题。

TCP 序列号/确认机制的常规运行方式

重传机制是内置于 TCP 的众多机制之一，其作用是恢复受损、丢失、重复或失序交付的数据。

重传机制的实现方法也非常简单：发送方先为通过 TCP 传输的数据按字节编号（序列号机制），然后再通过 TCP 报文段的形式发出，并期待接收方的确认；若在一定的时长内未得到确认，发送方将重传数据。

收到 TCP 报文段之后，接收方会检查序列号（字段值），以验证报文段是否是按序抵达；若否，则对报文段重新排序，并按正确的顺序交付给应用层。

TCP 序列号/确认机制的运行方式如下所列。

- 建立 TCP 连接时，双方会彼此通告自己的初始序列号。
- 开始发送数据时，（用来承载数据的）TCP 报文段的头部中会包含序列号（字段值）。第一个 TCP 报文段中的序列号表示本报文段所承载的数据净载的首字节编号。后续 TCP 报文段中的序列号则等于：上一个 TCP 报文段的序列号，加上该报文段所承载的数据的字节数，再加上 1（如图 9.17 所示）。

- 每发出一个 TCP 报文段，发送方便会在发出的那一刻，启动一个 RTO（Retransmission Timeout[重传超时]）计时器，若在该计时器到期之前，未收到接收方发出的确认报文段，便会重新传送。



注意

TCP 重传超时机制基于 Van Jacobson 碰撞避免及控制算法，该算法决定了 TCP 可以容忍网络的高延迟，但不能容忍剧烈的抖动。）

- 收到 TCP 报文段之后，接收方会发出相应的确认报文段，确认接收，同时告知发送方发出下一个报文段。

可通过图 9.17 所示的 Wireshark 抓包主窗口截图来研究 TCP 序列号/确认机制的运作方式。

Filter:	Expression...	Clear	Apply	Save
Source	Destination	Protocol	Length	Info
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
10.0.0.7	62.219.24.171	TCP	54	53203 > http [ACK] Seq=1 Ack=1201
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
10.0.0.7	62.219.24.171	TCP	54	53203 > http [ACK] Seq=1 Ack=1201
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
10.0.0.7	62.219.24.171	TCP	54	53203 > http [ACK] Seq=1 Ack=1201
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
10.0.0.7	62.219.24.171	TCP	54	53203 > http [ACK] Seq=1 Ack=1201
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic
10.0.0.7	62.219.24.171	TCP	54	53203 > http [ACK] Seq=1 Ack=1201
62.219.24.171	10.0.0.7	HTTP	1506	Continuation or non-HTTP traffic

图9.17

- HTTP 客户端主机 10.0.0.7 正从 HTTP 服务器 62.219.24.171 下载文件（已配置 Wireshark，在其抓包主窗口的数据包列表区域新增了 tcp.seq 和 tcp.ack 数据包属性列，具体的添加方法请参考 1.8.2 节）。
- HTTP 服务器 62.219.24.171 连发 2 个序列号分别为 120185105 和 120186557 的 TCP 报文段。收到这两个报文段之后，HTTP 客户端 10.0.0.7 发出了确认号为 120188009 的确认报文段，通知 HTTP 服务器发出下一个报文段。随后，HTTP 服务器便发出了序列号为 120188009 的报文段，后面还紧跟一个序列号为 120189461 的报文段。

HTTP 服务器 62.219.24.171 和 HTTP 客户端 10.0.0.7 之间的上述交流过程，如图 9.18 所示。

何为 TCP 重传，因何而起

发出 TCP 报文段之后，若发送方收不到或不能按时收到相应的确认报文段，便会出现以下两种情况。

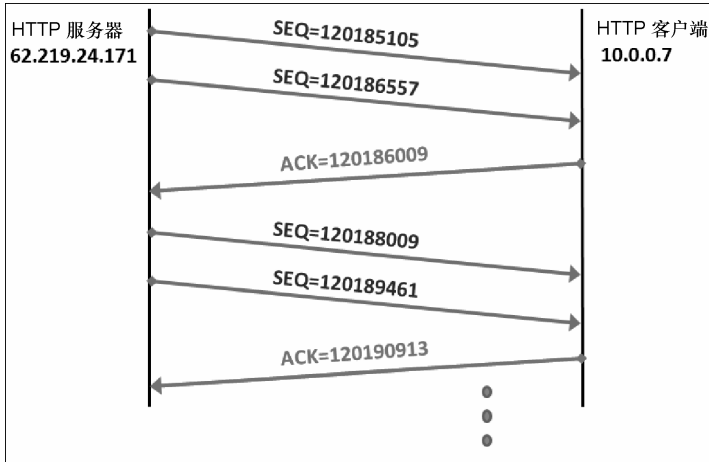


图9.18

1. 发送方会按之前所述，重传未得到确认的 TCP 报文段。
2. 发送方会降速传输。

通过图 9.19 不难发现，只要发生 TCP 重传，发送方便会降速传输（为清晰起见，作者为该图添加了红色直线 [需要在 Wireshark 中显示]）。

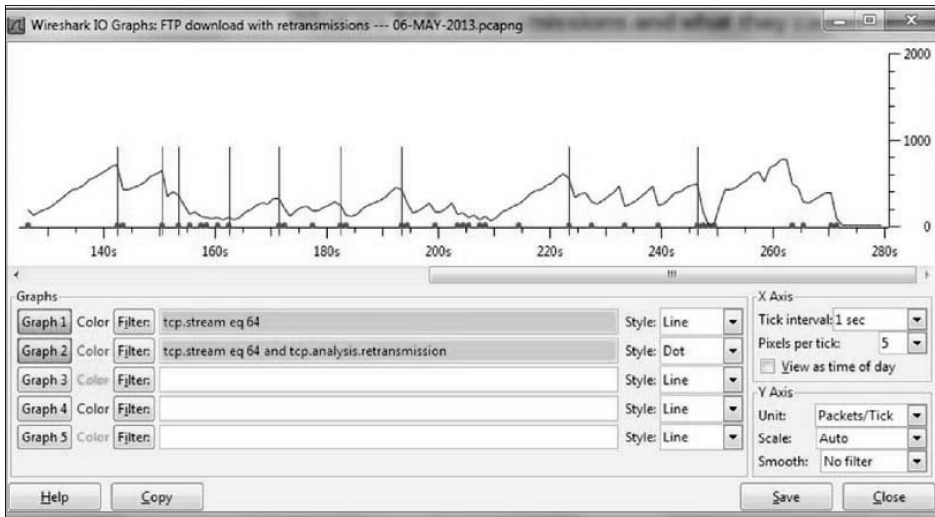


图9.19

9.4.4 拾遗补缺

TCP 不惧怕高网络延迟，就怕延迟发生剧烈的变化。定义 TCP 在网络发生抖动（以及其

他诸多变化)时的算法被称为 Van Jacobson 算法(该算法以其发明人命名)。根据 Van Jacobson 算法的定义,对网络延迟而言,TCP 所能容忍的极限是网络平均延迟的 3~4 倍。也就是说,若正常情况时网络的延迟为 100ms,TCP 容忍 300~400ms 的延迟应该不成问题,但前提是延迟的变化不能太过频繁(即不能有剧烈的抖动)。

9.4.5 进阶阅读

Van Jacobson 算法的详情请见 <http://ee.lbl.gov/papers/congavoid.pdf>。

9.5 重复确认 (duplicate ACKs) 和快速重传 (fast retransmissions) 现象

重复确认和快速重传现象是使用 TCP 传输数据时的常见现象。这一现象也有可能是由性能问题所引发,本节将探讨如何发现这一现象,还将介绍出现这一现象时意味着什么。

9.5.1 准备工作

在操纵基于 TCP 的应用程序时,若感觉到应用程序的运行速度明显变慢,则 TCP 重复确认很可能是原因之一。此时,应把 Wireshark 主机接入网络,并借助于交换机端口镜像功能,来抓取并分析出问题的安装 TCP 应用程序客户端软件或服务器端软件的主机的流量。

9.5.2 操作方法

在大多数情况下,导致 TCP 重复确认的原因不外有三:高延迟、高抖动或发送方“反应慢”(发送方在发出 TCP 报文段之后,无法及时响应接收方发出的确认报文段)。

在查找 TCP 应用程序的运行速度明显变慢的原因时,请千万别忘了还有 TCP 重复确认这茬^①。

1. 若(Wireshark 感知到的)行使重复确认功能的 TCP 报文段数量合理(比如,1%~2%的比例),则可能没有问题。
2. 若(Wireshark 感知到的)行使重复确认功能的 TCP 报文段数量惊人(比如,10%的比例),则很有可能是:
 - 因通信链路拥塞,而产生了剧烈的抖动;
 - TCP 服务器端(主机或软件)或客户端(主机或软件)不正常(具体是哪一端,需仔细查明)。

^① 译者注:原文是 “When looking for a reason for slow communication, duplicate ACKs can be one of the reasons for it”。

3. TCP 快速重传是一种机制，能促使发送方根据接收方的反馈，来立刻重传 TCP 报文段，无需等待重传计时器的超时^①。
4. 图 9.20 所示的 Wireshark 截图清晰地反映了由 TCP 重复确认所触发的 TCP 快速重传。由图可知，在收到了接收方(对同一个 TCP 报文段的)第 51 次重复确认(身背 TCP DUP ACK xxx #字样的 TCP 报文段)之后，发送方以快速重传的方式，发出了序列号字段值为 14593377 的 TCP 报文段(身背 TCP fast retransmission 字样的 TCP 报文段)。请注意，14593377 是接收方发出的那 51 个身背 TCP DUP ACK xxx #字样的 TCP 报文段的确认编号字段值。

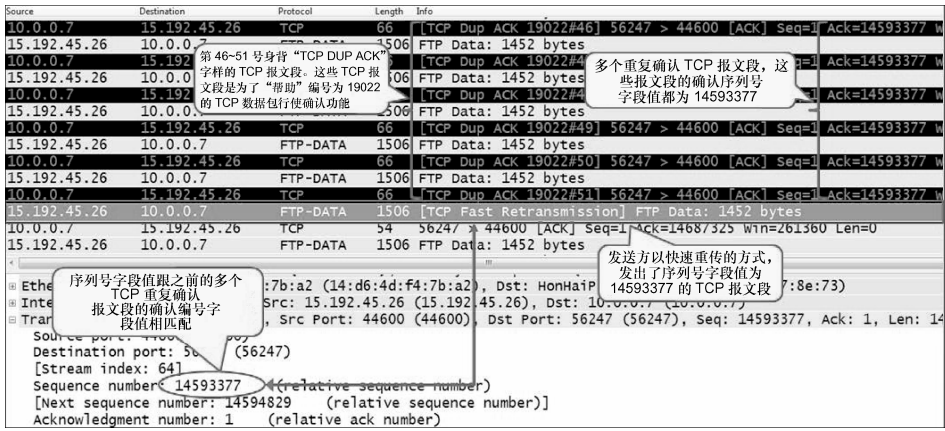


图 9.20

5. 以下所列发生 TCP 重复确认和快速重传现象时的应对方法。
- 若 TCP 重复确认报文段和快速重传报文段数量较低（比例低于 1%），则可以容忍。
 - 若 TCP 重复确认和快速重传现象发生在 Internet 链路上，或蜂窝/无线网络内，则也无计可施，因为上述线路或网络的特征就是高延迟、高抖动。
 - 若 TCP 重复确认和快速重传现象发生于带宽较高的企事业单位的内部网络中，则应仔细查找原因。若发生在 LAN 内，则问题比较严重，应重点检查交换机或服务器的缓存及 CPU 负载状况。若发生在 WAN 内，应重点检查 WAN 线路的延迟及负载状况。

^① 译者注：原文是“A fast retransmission is a packet that is sent in response to the duplicate ACKs”。作者根本就不知道什么是 TCP 快速重传，译文译自 *TCP/IP Illustrated, Volume 1 (Second Edition)*。还有就是，本节对 TCP 快速重传的许多描述都不正确，只是相关概念解释起来太过复杂，译者也只好按原文字面意思翻译了。

9.5.3 幕后原理

当接收方觉察到发送方发出的 TCP 报文段丢失（即未收到发送方发出的序列号字段值为预期值的 TCP 报文段）或收到的 TCP 报文段的序列号字段值出乎预料时，会一再生成确认编号字段值为特定值的 TCP 报文段进行重复确认，其目的是让发送方重新传送先前传丢的 TCP 报文段^①。

连续三次收到确认编号字段值相同的 TCP 报文段（一次原始的 TCP 报文段，外加两次确认编号字段值相同的 TCP 报文段）时，发送方会假定相应的序列号字段值的 TCP 报文段传丢，会立刻执行快速重传，无论与此 TCP 报文段挂钩的重传超时计时器是否到期^②。这一重传 TCP 报文段的方式被称为快速重传。

重复确认同样会降低相关 TCP 连接的数据传输效率。效率到底会降低多少，要视 TCP 的版本（TCP Tahoe、TCP Reno 和 TCP New-Reno）而定。对于早期的 TCP 版本，只要收到用来执行重复确认的 TCP 报文段（身背 TCP DUP ACK 字样的 TCP 报文段），发送方便会把数据传输速率降至之前的一半。倘若收到的此类 TCP 报文段过多，数据传输速率将会降至最低。

图 9.21 所示为因 TCP 重复确认和快速重传而导致传输效率下降的典型示例。

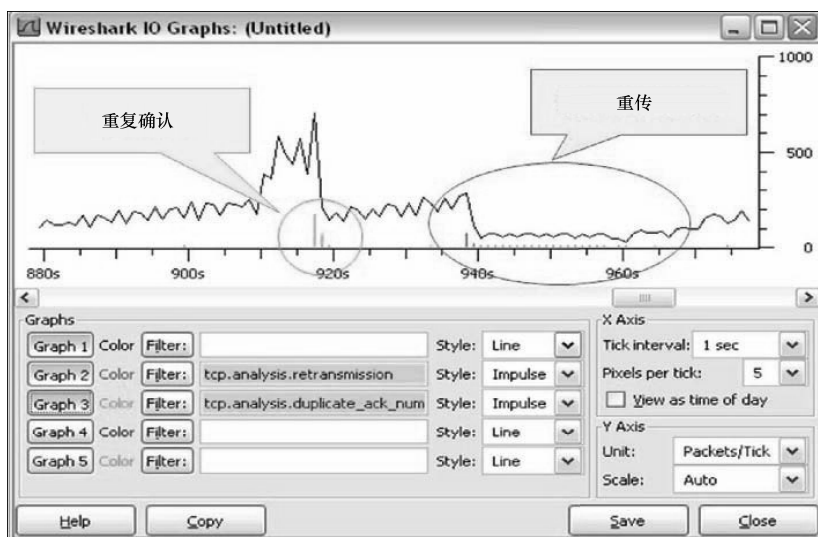


图9.21

① 译者注：原文是“A duplicate ACK happens if a packet is detected as missing (the expected sequence number was not received), or when an unexpected sequence number was received”。这样的文字简直目不忍睹。

② 译者注：原文是“On the sender side, when it receives three identical ACKs (the original ACK and two duplicate ACKs) with the same value, it assumes there is a packet loss and it resends the missing packet, regardless of whether the RTO is expired or not”。这跟 TCP/IP Illustrated, Volume 1 (Second Edition) 一书所述完全不同，请读者留意。

由图 9.21 可知，收到了第一个身背 TCP DUP ACK 字样的 TCP 报文段（DupACK）之后，发送方就把 TCP 传输速率降到了之前的 60%，在执行 TCP 重传期间，传输速率降到了最低点。

9.5.4 拾遗补缺

重复确认所使用的机制被称为快速恢复^①。近年来，为了提升 TCP 在高延迟、高抖动网络环境（尤其是蜂窝网络环境）中的运行效率，人们推出了某些高级 TCP 版本。与通用 TCP 版本相比，这些高级 TCP 版本在发送方和接收方的交互方式上做出了改进（比如，支持更为快速的恢复机制等），但 TCP 的基本属性仍维持不变。

9.6 TCP 报文段失序现象

借助 Wireshark 执行网络流量监控任务时，其抓包主窗口经常会提示某些 TCP 数据包存在 **previous segment loss**（上一个 TCP 报文段丢失）和 **out-of-order segments**（TCP 报文段失序）现象。以上两种现象都与（封装 TCP 报文段的）IP 数据包失序抵达接收方有关，有时也预示着网络状况异常^②。

当 Wireshark 感知到 TCP 报文段失序抵达接收方时，既有可能是因为网络状况异常，也有可能是由于抓包不慎。本节将深入探究导致 TCP 报文段失序抵达接收方的原因。

9.6.1 准备工作

把 Wireshark 主机接入交换机，激活交换机的端口镜像功能，将有待监控的交换机端口的流量重定向至 Wireshark 主机。本节重点介绍由 Wireshark 软件识别出的以下三种现象。

- **Previous segment lost:** 只要 Wireshark 在某股 TCP 数据流中感知到了接收方收到的某个 TCP 报文段的序列号字段值高于预期，就会在抓包主窗口的数据包列表区域中让该 TCP 报文段身背 **Previous segment lost** 字样，意在提醒网管人员：在身背该字样的 TCP 报文段之前，还有一或多个（序列号字段值更低的）TCP 报文段未能抵达接收方^③。

① 译者注：原文是“The mechanism that is used in duplicate ACKs is called fast recovery”。

② 译者注：原文是“Another phenomenon that you will see in networks is previous segment loss and out-of-order segments. Both relate to packets arriving out of order, and in some cases indicate a problem”。作者口齿不清，译文酌改。

③ 译者注：原文是“This occurs when a packet arrives with a sequence number higher than the next expected sequence number on that connection, indicating that one or more packets prior to the flagged packet did not arrive”。译文酌改。

- **Out-of-order packet:** 当 Wireshark 在某股 TCP 数据流中感知到序列号字段值低的 TCP 报文段先发后至，便会在抓包主窗口的数据包列表区域中让后发先至的 TCP 报文段身背 Out-of-order packet 字样，以提醒网管人员。
- **Previous segment not captured** (Wireshark 1.8 版本及以上): 情况与 Previous segment lost 类似。

9.6.2 操作方法

Wireshark 软件在什么情况下会感知到以上三种现象？

只要出现下列情况，Wireshark 软件就有可能感知到这三种现象。

- **开始抓包之初:** 在启动 Wireshark 开始抓包的同时，有待监控的 TCP 连接也刚好建立，Wireshark 可能碰巧未抓全用来建立 TCP 连接的三次握手报文段（SYN/SYN-ACK/ACK），在这种情况下，Wireshark 会生成相关提示信息，提醒用户有错误发生^①。
- **真的存在丢包情况:** 此时，Wireshark 会让某些 TCP 报文段在抓包主窗口的数据包列表区域中露面时，身背 TCP Previous segment not captured、TCP Dup ACK 等字样，提醒网管人员存在丢包现象（见图 9.22）。

No.	Source	Destination	Protocol	Length	Info	Seq
331	10.0.0.6	62.90.90.210	TCP	90	[TCP Dup ACK 223#54] 57999 > http	369
332	62.90.90.210	10.0.0.6	TCP	1474	[TCP Previous segment not captured]	319501
333	10.0.0.6	62.90.90.210	TCP	90	[TCP Dup ACK 223#55] 57999 > http	369
334	62.90.90.210	10.0.0.6	TCP	1474	[TCP segment of a reassembled PDU]	320921
335	10.0.0.6	62.90.90.210	TCP	90	[TCP Dup ACK 223#56] 57999 > http	369
336	62.90.90.210	10.0.0.6	TCP	1474	[TCP Previous segment not captured]	326601
337	10.0.0.6	62.90.90.210	TCP	90	[TCP Dup ACK 223#57] 57999 > http	369
338	62.90.90.210	10.0.0.6	TCP	1474	[TCP segment of a reassembled PDU]	328021
339	10.0.0.6	62.90.90.210	TCP	90	[TCP Dup ACK 223#58] 57999 > http	369
340	62.90.90.210	10.0.0.6	TCP	1474	[TCP Previous segment not captured]	332281
341	10.0.0.6	62.90.90.210	TCP	90	[TCP Dup ACK 223#59] 57999 > http	369

图9.22

图 9.22 所示的 Wireshark 截屏清楚地揭示了网络中存在的严重丢包现象。由图可知，客户机 10.0.0.6 正尝试访问存储在 Web 服务器 62.90.90.210 上的资源（比如，浏览器服务器上的网页）。在此期间，Web 服务器正向客户机发送多个 1420 字节的 TCP 报文段（客户机上的浏览器正在加载页面）。不难发现，在编号为 334 和 336 的数据包之间，应该传丢了 3 个 TCP 报文段；在编号为 338 和 340 的数据包之间，应该传丢了 2 个 TCP 报文段。因此，Wireshark 让编号为 336 和 340 的数据包背上了 TCP previous segment is not captured 的字样。

① 译者注：原文是“*At the beginning of capture: This event occurs when you start a capture during an open connection. In this case, you will see packets on a connection without the SYN/ SYN-ACK/ACK, therefore, Wireshark thinks something went wrong*”。作者的文字越来越差，译文酌改。

- **延迟差异（抖动）：**在 TCP 报文段（隶属于同一条 TCP 连接）从源主机向目的主机的交付过程中，若封装 TCP 报文段的 IP 数据包走不同的线路（路由），就有可能发生序列号字段值低的 TCP 报文段“先发后至”的情况，因为 IP 数据包所走每条线路的延迟不一。可借助于 `tracert` 工具来发现这一隶属于单条 TCP 连接的 TCP 报文段分别走不同线路的情况。对于单位或企业的内部网络，则可以在路由器上配置 SNMP trap，让网管软件自动报警。
- **抓包问题：**TCP 报文段实际已成功交付，但若 Wireshark 未能抓全所有数据包（因抓包不慎所致），也会在抓包主窗口的数据包列表区域中呈现 TCP previous segment is not captured 字样。导致这种情况的原因如下所列。
 - 流量太过汹涌（流量速率在 150Mbit/s 以上），以至于 Wireshark 难以抓全。要避免这种情况，请使用商业抓包工具。
 - 安装 Wireshark 的计算机配置太低（CPU 主频过低或内存不足），导致 Wireshark 软件对某些流量忍痛割爱。只要用 Wireshark 执行抓包任务的人智商没有问题，就应该能够觉察到这种情况。
 - LAN 交换机的端口缓存（buffer）容量太小，连累 Wireshark 软件抓不全所有流量。可登录 LAN 交换机（通过 console 电缆或 Telnet 方式），执行相关命令，增加其端口缓存容量。
 - 使用主机的无线网卡抓包，由于某些原因导致 Wireshark 软件抓不全所有流量。具体原因详见第 7 章。

9.6.3 幕后原理

发送方向接收方交付的每一个 TCP 报文段都有自己的编号（序列号），这些编号与 TCP 报文段的长度配套。因此，只要 TCP 报文段未能按序交付，Wireshark 就能够感知并生成相关提示信息。Wireshark 生成与 TCP 报文段失序交付有关的提示信息，无非以下两种原因。

- **TCP 报文段真的失序抵达接收方：**在这种情况下，在 Wireshark 抓包主窗口的数据包列表区域中，除了会出现身背 TCP previous segment is not captured 等字样的 TCP 报文段，在其附近，还会出现身背 TCP Dup ACK 和 retransmission 字样的 TCP 报文段。具体原因详见 *TCP/IP Illustrated, Volume 1 (Second Edition)*。
- **抓包问题：**对于这种情况，在 Wireshark 抓包主窗口的数据包列表区域中，只会出现身背 TCP previous segment is not captured 字样的 TCP 报文段，在其附近，不会出现身背 TCP Dup ACK 和 retransmission 字样的 TCP 报文段。

9.7 TCP Zero Window、Window Full、Window Change以及其他包含Window字样的提示信息

滑动窗口（Sliding Window）机制是 TCP 最重要的一项机制，这一流控（Flow Control）机制的作用是，控制 TCP 连接的每个端点发送或接收的数据量^①。

本节将讨论如何解决与 TCP 窗口有关的各种问题。

9.7.1 准备工作

把 Wireshark 主机接入交换机，激活交换机的端口镜像功能，将有待监控的（连接服务器的）交换机端口的流量重定向至 Wireshark 主机，同时紧盯抓包主窗口的数据包列表区域，观察是否有身背含 Window 字样的 TCP 报文段露面。

9.7.2 操作方法

读者应知道身背以下几种含 Window 字样的 TCP 报文段的含义。

身背 TCP Zero Window、Zero Window Probe 和 Zero Window Violation 字样的 TCP 报文段

TCP Zero Window 是指收到了发送方发出的 TCP 报文段之后，接收方在进行确认时，发出了窗口大小字段值为 0 的 TCP 确认（ACK）报文段。接收方之所以会如此行事，是要告知发送方：本方 TCP 接收缓存耗尽，请暂停发送数据。当 Wireshark 抓到窗口大小字段值为 0 的 TCP 报文段时，便会让其身背 TCP Zero Window 字样。只要 Wireshark 抓到了身背 TCP Zero Window 字样的 TCP 报文段，问题一般都出数据接收方，可能的原因如下所列。

- 通过 TCP 接收数据的主机配置太低，无法为相关进程分配足够的内存。
- 运行在数据接收方的应用程序无法从操作系统得到足够的缓存，只能发出窗口大小字段值为 0 的 TCP 报文段，让数据发送方暂停发送数据。
- 运行在数据接收方的应用程序消耗的内存太多，操作系统对其做了资源限制。

收到窗口大小字段值为 0 的 TCP 确认报文段之后，数据发送方会发出一种特殊的 TCP 报

^① 译者注：原文是 “One of the most important mechanisms of TCP is the Sliding Window mechanism, and the Flow Control mechanism that uses it in order to control the amount of data that a TCP end node is willing to accept on the connection”。作者连文字都写不通顺，却非要去解释这些复杂的 TCP 概念性问题。

文段,探测数据接收方的 TCP 接收窗口是否仍然为 0。这种特殊的 TCP 报文段被称为 **TCP Zero Window Probe** 报文段。Wireshark 只要感知到这种 TCP 报文段,便会让其身背 TCP Zero Window Probe 字样。收到身背 TCP Zero Window Probe 字样的 TCP 报文段之后,数据接收方必须要进行确认,在确认 TCP 报文段内会通告本方接收窗口大小。TCP Zero Window Probe 报文段的发送频率受控于一种叫做 TCP Persist 的计时器。数据发送方在第一次探测之前,需等待 TCP Persist 计时器到期,以后的每次探测,所等待的时间都将是上一次的两倍,直到数据接收方的 TCP 接收窗口不再为 0。TCP Zero Window Probe 报文段只包含(有待发送的下一个)1 字节数据,数据接收方接纳或不接纳这个 1 字节数据,要视自己的 TCP 接收缓存的富裕程度而定^①。

在收到了窗口大小字段值为 0 的 TCP 确认报文段之后,若数据接收方仍然不管不顾,继续向接收窗口为 0 的数据发送方传送数据,这一举动就被称为 TCP Zero Window Violation。若 Wireshark 在所抓数据包中感知到了数据发送方的这一举动,便会让相关 TCP 报文段身背 TCP Zero Window Violation 字样。出现 TCP Zero Window Violation 现象,一般都表示相关应用程序的 TCP 实现方式存在瑕疵^②。

若 Wireshark 抓到了身背 TCP Zero Window、Zero Window Probe 以及 Zero Window Violation 字样的 TCP 报文段,在弄清具体原因之前,需先查明上述 TCP 报文段的出处。

- 若全都出自某台主机(TCP 服务器端主机或客户端主机),则表示该主机存在问题。
- 若全都出自某种应用程序,则表示这种应用程序存在问题。
- 若在执行某种应用程序的某项操作(比如,打开特定的表格、打印文件、创建报表等操作)时,触发了那些 TCP 报文段,则一定是该应用程序的某个功能某块存在问题。

身背 TCP Window Update 字样的 TCP 报文段

TCP 连接的一方(发出 TCP 报文段)向对方通告本方的接收缓存容量时,可以随时调整容量的大小,以便实时通知对方提升或降低数据的发送速率。当出现以下两种情况时,Wireshark 会让相关 TCP 报文段身背 TCP Window Update 字样。

- 当接收方从 TCP Zero Window 状态中恢复,发送方发出首个窗口大小字段值不为 0 的 TCP 确认报文段时,Wireshark 会让该 TCP 报文段身背 TCP Window Update

① 译者注:原文是“TCP Zero Window Probe is a message that is sent by the sender in order to see if the receiver's Zero Window condition still exists. This message works by sending the next byte of data to the receiver. If the receiver answers with window that is still zero, the sender doubles his timer before probing again”。

② 译者注:原文是“The sender ignores the Zero Window condition of the receiver and sends additional bytes of data. TCP Zero Window Violation can indicate a TCP error or bug in the protocol stack”。

字样。在 Wireshark 抓包主窗口的数据包列表区域中,若发现了身背 TCP Window Update 字样的 TCP 报文段,无需采取任何行动,只需查明为什么会出现 TCP Zero Window 现象。

- 当接收方频繁改变自己的接收缓存大小时, Wireshark 也会让相关 TCP 报文段身背 TCP Window Update 字样。此时,应检查是什么原因导致接收方如此行事,可能的原因包括:应用程序问题、内存分配问题或其他与性能有关的问题。

若通过 Wireshark 观察到了上述现象,不用担心, TCP 就是这么运作的^①。

身背 TCP Window Full 字样的 TCP 报文段

当 Wireshark 识别出发送方发出的 TCP 报文段,将会是填满接收方的接收缓存的最后一个 TCP 报文段时,便会让该 TCP 报文段身背 TCP Window Full 字样。在收到发送方收发出的确认报文段(通告自己的接收缓存)之前,这一身背 TCP Window Full 字样的 TCP 报文段,将会是发送方发出的最后一个包含数据的 TCP 报文段。

收到这一身背 TCP Window Full 字样的 TCP 报文段之后,若应用程序不能从 TCP 接收缓存中取走任何数据,接收方便会发出身背 TCP Zero Window 字样的 TCP 报文段,请求发送方暂缓发送数据。

TCP Window Full 和 TCP Zero Window 现象的起因应该相同:不是运行应用程序的主机就是应用程序自身反应过慢。图 9.23 所示为 Wireshark 感知到上述现象的典型示例。

No.	Source	Destination	Protocol	Length	Info
182995	192.168.2.138	192.168.1.58	TCP	590	[TCP segment of a reassembled PDU]
182996	192.168.2.138	192.168.1.58	TCP	590	[TCP segment of a reassembled PDU]
182997	192.168.1.58	192.168.2.138	TCP	54	http > 47185 [ACK] Seq=1 Ack=131685 win=14
183816	192.168.2.138	192.168.1.58	TCP	197	[TCP Window Full] [TCP segment of a reassembled PDU]
183828	192.168.1.58	192.168.2.138	TCP	54	[TCP ZeroWindow] http > 47185 [ACK] Seq=1
183966	192.168.2.138	192.168.1.58	TCP	60	[TCP ZeroWindowProbe] [TCP segment of a reassembled PDU]
183967	192.168.1.58	192.168.2.138	TCP	54	[TCP ZeroWindowProbeAck] [TCP ZeroWindow]
184128	192.168.2.138	192.168.1.58	TCP	60	[TCP ZeroWindowProbe] [TCP segment of a reassembled PDU]
184129	192.168.1.58	192.168.2.138	TCP	54	[TCP ZeroWindowProbeAck] [TCP ZeroWindow]
184388	192.168.2.138	192.168.1.58	TCP	60	[TCP ZeroWindowProbe] [TCP segment of a reassembled PDU]
184389	192.168.1.58	192.168.2.138	TCP	54	[TCP ZeroWindowProbeAck] [TCP ZeroWindow]
185002	192.168.2.138	192.168.1.58	TCP	60	[TCP ZeroWindowProbe] [TCP segment of a reassembled PDU]
185003	192.168.1.58	192.168.2.138	TCP	54	[TCP ZeroWindowProbeAck] [TCP ZeroWindow]
186460	192.168.2.138	192.168.1.58	TCP	60	[TCP ZeroWindowProbe] [TCP segment of a reassembled PDU]
186461	192.168.1.58	192.168.2.138	TCP	54	[TCP ZeroWindowProbeAck] [TCP ZeroWindow]
189877	192.168.2.138	192.168.1.58	TCP	60	47185 > http [RST, ACK] Seq=131828 Ack=1

图9.23

通过图 9.23 可以看出下述情况。

^① 译者注:原文是“If you see this kind of phenomena, there is nothing to worry about. This is how TCP works”。译文按原文字面意思直译。

1. Wireshark 识别出了编号为 183816 的 TCP 报文段是数据接收主机 192.168.1.58 的 TCP 接收窗口被填满之前, 所能接收的最后一个包含数据的 TCP 报文段。
2. Wireshark 同样识别出了下一个 TCP 报文段(编号为 183828)是数据接收主机 192.168.1.58 发出的窗口大小字段值为 0 的 TCP 报文段, 其目的是让数据发送主机 192.168.1.138 暂停发送数据。
3. 随后, Wireshark 又识别出了主机 192.168.1.138 和 192.168.1.58 互相交换了若干个 TCP Zero Window 和 Zero Window Probe 报文段。
4. 图中所示最后一个数据包是主机 192.168.1.138 发出的 RST 位置 1 的 TCP 报文段, 目的是重置 TCP 连接。
5. 若该 TCP 连接能从 TCP Zero Window 状态中恢复, 不被重置, 则 Wireshark 势必会识别出主机 192.168.1.58 发出的身背 TCP Window Update 字样的 TCP 报文段, 其目的是让主机 192.168.1.138 继续发送数据。

9.7.3 幕后原理

以下所列为 TCP 滑动窗口机制的运作方式。

1. TCP 连接建立之后, 发送方开始向接收方传送包含数据的 TCP 报文段, 后者会使用接收缓存(窗口)来存放。
2. 收到了包含数据的 TCP 报文段后, 接收方会发出 TCP 确认(ACK)报文段, 向发送方确认已收数据字节。接收方发出 TCP 确认报文段, 则表示其接收缓存(窗口)腾出了相应的空间^①。
3. 发送方发出包含数据的 TCP 报文段, 接收方发出 TCP 确认报文段, 确认数据的接收, 同时消耗接收缓存中的数据, 给接收缓存腾出空间。这一过程会一直持续, 直到数据传送完毕。
4. 当接收方发出 TCP 确认报文段确认数据的接收时, 若顺带提升或降低了 TCP 头部中的窗口大小字段值, 则意味着发送方应相应提高或降低数据的传送速率, 具体的计算公式如图 9.24 所示(可能会随不同的 TCP 版本而发生某些变化)。

^① 译者注: 原文是“After several packets, the receiver sends an ACK to the sender, confirming the acceptance of the bytes sent by it. Sending the ACK empties the receiver window”。译者认为, 在 TCP 接收方, 若应用程序不去读去或消耗 TCP 接收缓存中的数据, 发出 TCP 确认报文段是不能让 TCP 接收缓存腾出空间的。此外, 作者似乎根本就不懂什么是 TCP 滑动窗口机制, 建议读者阅读 *TCP/IP Illustrated, Volume 1 (Second Edition)* 一书的 15.5 节。

$$\text{Throughput [字节/秒]} = \frac{\text{Window Size [字节]}}{\text{RTT [秒]}}$$

- ✓ Throughput –在TCP连接中传播的应用程序数据的传送速率
- ✓ Window Size – 接收方通过TCP确认报文段通告的窗口大小（字段值）
- ✓ RTT–发送方和接收方之间的往返延迟

图9.24

9.7.4 拾遗补缺

还可以使用内置于 Wireshark 的 IO Graph 和 Time-Sequence(tcp-trace)工具,来定位与 TCP 窗口有关的问题。图 9.25 所示为由 Time-Sequence(tcp-trace)工具生成的图形,图中上、下两道线之间的空间表示接收窗口的大小,若两道线重合则表示接收窗口为 0 (Zero Window)。那上、下两道线非别表示接收窗口的右侧和左侧^①。

在 TCP 传输过程中,若上、下两道线之间的距离始终保持固定,且留有较大的空间(如图 9.25 所示),则表示接收方的接收缓存富裕;若上、下两道线之间的空间狭小,则表示接收方的接收缓存吃紧。只要那两道线不发生重叠,发送方就能继续向接收方传输数据。

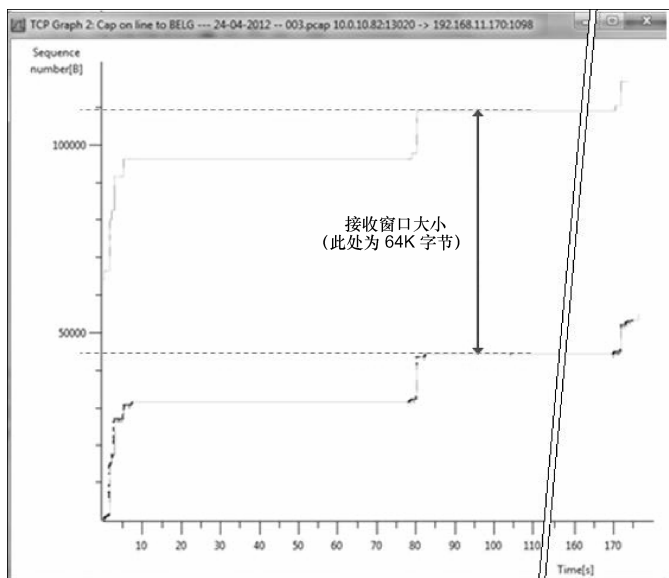


图9.25

① 译者注：原文是 “You can also use the TCP throughput graphs and the IO graphs to view these problems. In the TCP throughput graphs, use the TCP trace graph, where the upper line indicates the window size, and its distance from the lower line indicates what is on the left-hand side of the window. No distance between them indicates a Zero Window”. 译者不认可作者的说法，译文酌改。如有不妥，请指正。

9.8 TCP重置 (reset) 及原因

在正常情况下，建立 TCP 连接时要经历三次握手过程，关闭 TCP 连接时则要经历四次握手过程。然而，在发生故障或在追求高效的情况下，TCP 支持以更为便捷的重置 (reset) 方式，来关闭已然建立起的连接。这也正是本节所要探讨的内容。

9.8.1 准备工作

把 Wireshark 主机接入交换机，激活交换机的端口镜像功能，将有待监控的（连接服务器的）交换机端口的流量重定向至 Wireshark 主机，同时紧盯抓包主窗口的数据包列表区域，观察 Wireshark 是否抓到了身背 reset 字样的数据包。TCP 重置可能会因多种原因而起，有时只不过是应用程序或 TCP 协议的正常行为，有时则与网络或应用程序故障有关。本节将探讨导致 TCP 重置的各种原因，同时会指出问题，传授解决问题的方法。

9.8.2 操作方法

TCP 重置是指 TCP 连接中的一方向另一方发出 RST 标记位置 1 的 TCP 报文段，这将会导致该 TCP 连接的快速中断。

发生 TCP 重置，但并非由故障引发的情形

执行四次握手，是关闭 TCP 连接的常规做法。也就是说，要正常关闭 TCP 连接，建立连接的双方要互发 FIN/ACK 位同时置 1 和 ACK 位置 1 的 TCP 报文段各一。通过浏览器打开一个 Web 页面时，可能会建立几十或几百条 TCP 连接（用来传送主页面、图片、广告等各种资源），要按常规方式关闭这些 TCP 连接，将会诞生无数 FIN/ACK 位同时置 1 和 ACK 位置 1 的 TCP 报文段。为了规避这种情况，很多 Web 服务器都会在传送完 HTTP 客户端所要下载的资源之后，以重置的方式断开 TCP 连接。由此触发的 TCP 重置现象，就属于应用程序的正常行为。

由故障引发的 TCP 重置现象

在某些情况下，TCP 重置是由故障（未必都是通信故障）引发。

- **由防火墙导致的 TCP 重置：**当尝试连接远程服务器，但总是连接不上时，使用 Wireshark 抓包，若发现 TCP 连接总是在建立之初被重置，就有可能是防火墙在里面捣鬼。图 9.26 所示的 Wireshark 截屏便反映出了上述情况，由图可知，在建立 TCP 连接的首个 SYN 位置 1 的 TCP 报文段出发以后，总是会有一个 RST 位置的 TCP 报文段来响应。

No.	Source	Destination	Protocol	Length	Info
54	192.168.1.123	192.168.1.141	TCP	54	ftp-data > 39042 [RST, ACK] Seq=1 Ack=
55	192.168.1.141	192.168.1.123	TCP	74	39020 > ftp [SYN] Seq=0 win=5840 Len=0
56	192.168.1.123	192.168.1.141	TCP	54	ftp > 39020 [RST, ACK] Seq=1 Ack=1 win=
57	192.168.1.141	192.168.1.123	TCP	74	56045 > ssh [SYN] Seq=0 win=5840 Len=0
58	192.168.1.123	192.168.1.141	TCP	54	ssh > 56045 [RST, ACK] Seq=1 Ack=1 win=
59	192.168.1.141	192.168.1.123	TCP	74	47648 > telnet [SYN] Seq=0 win=5840 Le
60	192.168.1.123	192.168.1.141	TCP	54	telnet > 47648 [RST, ACK] Seq=1 Ack=1
61	192.168.1.141	192.168.1.123	TCP	74	44370 > 24 [SYN] Seq=0 win=5840 Len=0
62	192.168.1.123	192.168.1.141	TCP	54	24 > 44370 [RST, ACK] Seq=1 Ack=1 win=
63	192.168.1.141	192.168.1.123	TCP	74	48264 > smtp [SYN] Seq=0 win=5840 Len=
64	192.168.1.123	192.168.1.141	TCP	54	smtp > 48264 [RST, ACK] Seq=1 Ack=1 wi
65	192.168.1.141	192.168.1.123	TCP	74	49404 > 26 [SYN] Seq=0 win=5840 Len=0
66	192.168.1.123	192.168.1.141	TCP	54	26 > 49404 [RST, ACK] Seq=1 Ack=1 win=
67	192.168.1.141	192.168.1.123	TCP	74	46880 > nsw-fe [SYN] Seq=0 win=5840 Le
68	192.168.1.123	192.168.1.141	TCP	54	nsw-fe > 46880 [RST, ACK] Seq=1 Ack=1
69	192.168.1.141	192.168.1.123	TCP	74	41799 > 28 [SYN] Seq=0 win=5840 Len=0
70	192.168.1.123	192.168.1.141	TCP	54	28 > 41799 [RST, ACK] Seq=1 Ack=1 win=

图9.26

- 因 TCP 端点故障而导致的 TCP 重置：有多种 TCP 端点故障都会导致 TCP 重置，下面举两个例子。
 - 数据发送方在连续发起重传 5 次之后，若等不来数据接收方的 TCP 确认报文段，便会发出 RST 位置 1 的 TCP 报文段，拆除该 TCP 连接。
 - 在 TCP 连接已经建立的情况下，若在特定时长内无数据传送（具体时长为操作系统默认），主动打开连接的一方往往会发出 RST 位置 1 的 TCP 报文段，拆除 TCP 连接，但并非总是如此。

9.8.3 幕后原理

TCP 重置是指 TCP 连接中的一方向另一方发出 RST 标记位置 1 的 TCP 报文段，这会导该 TCP 连接的快速中断。需要牢记的是，有些应用程序在 TCP 连接时会采取“重置”方式，而另一些则会循规蹈矩，采取完整的四次握手机制。有时，RST 标记位置 1 的 TCP 报文段在 Wireshark 抓包文件中露面，只不过是程序设计者有意为之。此外，对于在抓包文件中露面的每一个 RST 标记位置 1 的 TCP 报文段，都应仔细查找原因。

第 10 章

HTTP和DNS

本章涵盖以下主题：

- ▶ 筛选 DNS 流量；
- ▶ 分析 DNS 的常规运作机制；
- ▶ DNS 故障分析；
- ▶ 调整与 HTTP 协议有关的 Wireshark 配置参数；
- ▶ HTTP 故障分析；
- ▶ 导出 HTTP 对象；
- ▶ HTTP 数据流分析；
- ▶ HTTPS 流量分析（SSL/TLS 基础知识）。

10.1 简介

DNS（域名系统）协议是一种用来在名称（域名、主机名）和 IP 地址之间相互解析的协议。当浏览 Internet Web 站点时，就需要用 DNS 协议将 Web 站点的名称（域名）解析为 IP 地址。在企业内网中，在访问某些服务器时，也要用 DNS 协议将服务器名转换为 IP 地址。

HTTP 和 HTTPS（Secured HTTP）协议都是在上网冲浪时经常使用的协议^①。HTTPS 协议

^① 译者注：原文是“Hyper Text Transfer Protocol (HTTP) and Secured HTTP (HTTPS) are both used for browsing the Internet, or connecting to other software that are hosted inside your organization or in the cloud”。作者对 HTTP 和 HTTPS 协议的定义差到了极点。

是 HTTP 和 SSL/TLS 的结合体，用来提供通信加密功能以及认证网络服务器的功能。HTTPS 常被用来传输敏感信息，会在连接网银系统、Web 邮箱（通过浏览器打开 Google 或 hotmail 邮箱）或其他安全性应用时使用。

本章会介绍以上三种协议，探讨它们的运作机制，同时会介绍如何使用 Wireshark 来发现相关故障。

10.2 筛选DNS流量

DNS（域名系统）协议是一种用来在名称和 IP 地址之间相互解析的协议。本节会介绍如何配置 Wireshark 显示过滤器，来筛选 DNS 流量^①。

10.2.1 准备工作

将 Wireshark 主机接入 LAN 交换机，激活交换机的端口镜像功能，再把有待监控的（连接服务器的）交换机端口流量重定向至 Wireshark 主机；运行 Wireshark 软件，选择正确的网卡，开始抓包。

10.2.2 操作方法

可利用 Wireshark 显示过滤器，来帮助定位与 DNS 有关的故障。下面举几个在实战中经常会用到的 Wireshark 显示过滤器示例。

- 要让 Wireshark 只显示 DNS 流量，可直接在抓包主窗口的 Filter 输入栏内输入 `dns`，再点 **Apply** 按钮。
 - 要让 Wireshark 只显示 DNS 查询消息，显示过滤器的写法为：`dns.flags.response == 0`。
 - 要让 Wireshark 只显示 DNS 响应消息，显示过滤器的写法为：`dns.flags.response == 1`（见图 10.1）。
- 要让 Wireshark 只显示响应代码（Response Code, RCODE）字段值为某一具体值的 DNS 响应消息，显示过滤器的写法为：`dns.flags.rcode == x`^②。当然，也可以通过图形界面来构造，如图 10.2 所示。

① 译者注：原文是 “In this recipe, we will learn how to filter important parameters that are related to the DNS service”。作者啊，你技术不精也就罢了，连文字都写不通。

② 译者注：原文是 “For filtering error codes, we have the following filters”。译文酌改。

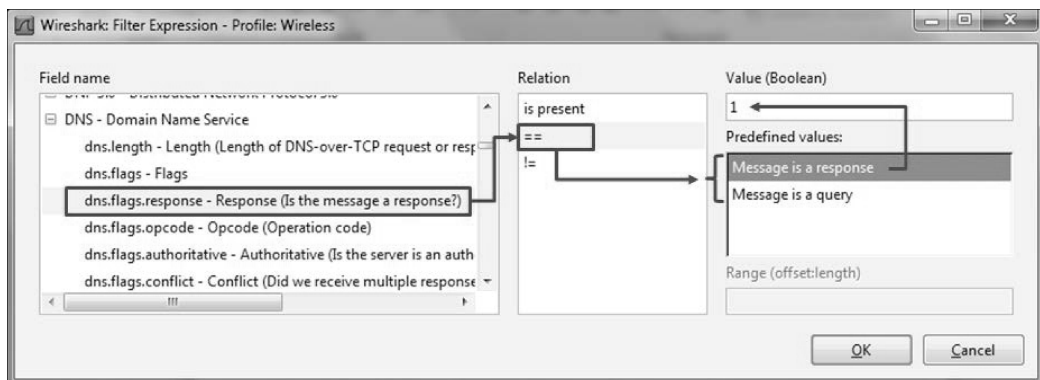


图10.1

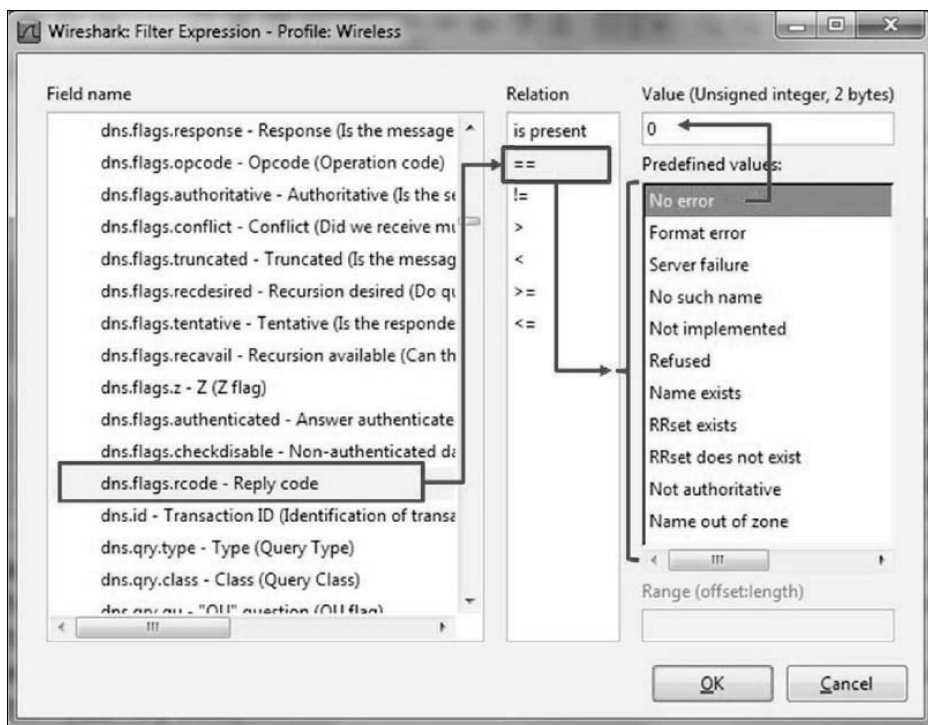


图10.2

- 要让 Wireshark 只显示 RCODE 字段值为 0 (NoError) 的 DNS 应答消息，需把 `dns.flags.rcode == x` 中的 `x` 替换为 0。
- 要让 Wireshark 只显示 RCODE 字段值为 3 (NXDomain, 表示域名不存在) 的 DNS 应答消息，需把 `dns.flags.rcode == x` 中的 `x` 替换为 3。

- 要让 Wireshark 只显示 query section 中的 query name 字段值匹配或包含指定 URL 的 DNS 数据包，显示过滤器的写法如下所列^①。
 - 要让 Wireshark 只显示 query section 中的 query name 字段值匹配指定 URL 的 DNS 数据包，显示过滤器的写法为：`dns.qry.name == "URL Name"`。
 - 要让 Wireshark 只显示 query section 中的 query name 字段值包含指定 URL 的 DNS 数据包，显示过滤器的写法为：`dns.qry.name contains "URL Name"`。
- 要让 Wireshark 只显示具有指定 OpCode（操作代码）字段值的 DNS 数据包，显示过滤器的写法如下所列。
 - 要让 Wireshark 只显示 OpCode 字段值为 0 的 DNS 数据包（即让 Wireshark 只显示 DNS 常规查询消息，以及相应的 DNS 响应消息），显示过滤器的写法为：`dns.flags.opcode == 0`（见图 10.3）。

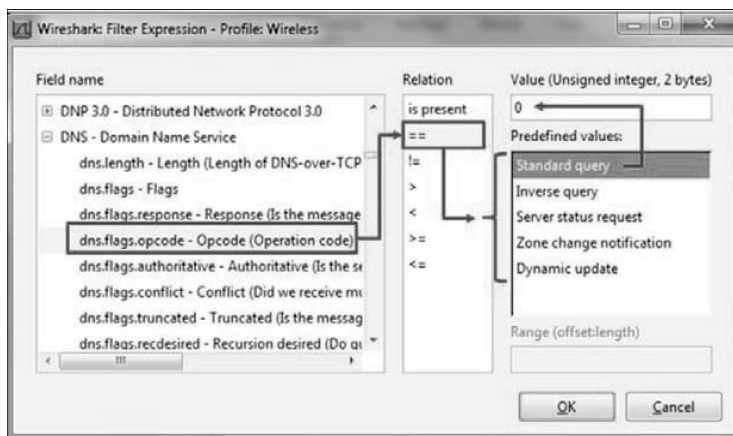


图 10.3

- 要让 Wireshark 只显示 OpCode 字段值为 1 的 DNS 数据包（即让 Wireshark 只显示 DNS 逆向查询消息，以及相应的 DNS 响应消息），显示过滤器的写法为：`dns.flags.opcode == 1`。
- 要让 Wireshark 只显示 OpCode 字段值为 2 的 DNS 数据包（即让 Wireshark 只显示 DNS 服务器状态请求查询消息，以及相应的 DNS 服务器状态响应消息），显示过滤器的写法为：`dns.flags.opcode == 2`。

^① 译者注：原文为“*For search problems, we have the following filters*”。说简单点，就是让 Wireshark 只显示携带了某个特定域名的 DNS 查询或响应消息。

- 要让 Wireshark 只显示 OpCode 字段值为 4 的 DNS 数据包（即让 Wireshark 只显示 DNS 区域变更通知查询消息），显示过滤器的写法为：`dns.flags.opcode == 4`。
- 要让 Wireshark 只显示 OpCode 字段值为 5 的 DNS 数据包（即让 Wireshark 只显示 DNS 动态更新查询消息，以及相应的 DNS 动态更新响应消息），显示过滤器的写法为：`dns.flags.opcode == 5`。
- 要让 Wireshark 只显示 RD（Recursion Desire）标记位置 0 或置 1 的 DNS 数据包，显示过滤器的写法如下所列。
 - 要让 Wireshark 只显示 RD 标记位置 1 的 DNS 查询及响应消息（RD 标记位置 1 的 DNS 查询消息一般为主机发出，目的是要求接收该消息的 DNS 服务器执行递归查询），显示过滤器的写法是：`dns.flags.recdesired == 1`（见图 10.4）。

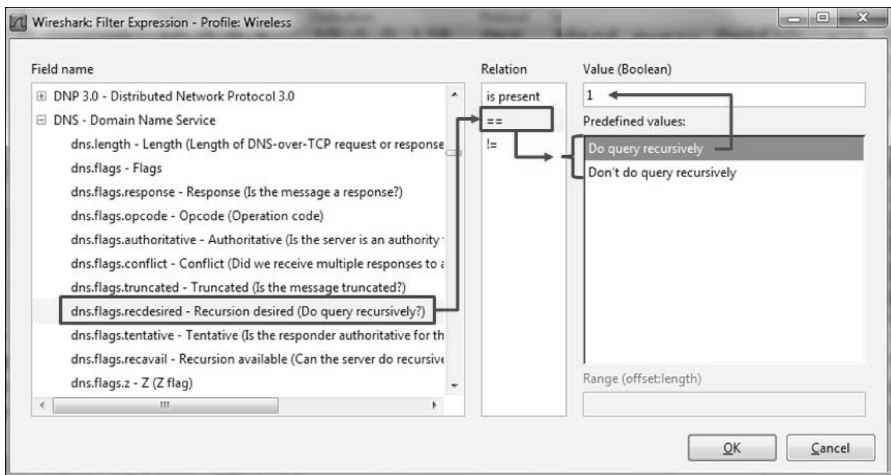


图 10.4

- 要让 Wireshark 只显示 RD 标记位置 0 的 DNS 查询及响应消息，显示过滤器的写法是：`dns.flags.recdesired == 0`。

还可点击抓包主窗口中的 **Expression** 按钮，在弹出的 Filter Expression 窗口的左侧找到 DNS 协议，点击前面的+号，以图形化的方式构造各种 DNS 显示过滤器。

10.2.3 幕后原理

本书第 3 章已经详述了显示过滤器的三种配置方法，当配置用来筛选 DNS 流量的显示过滤器时，也要使用这三种方法，此处再复述一遍。

- 借助于显示过滤器窗口。点击抓包主窗口中显示过滤器工具条上的 **Expression** 按钮，在弹出的 **Filter Expression** 窗口的左侧找到 DNS 协议，点击其前面的“+”号，选择适当的显示过滤参数来构造显示过滤器。
- 在抓包主窗口的显示过滤器工具条的 **Filter** 输入栏里直接输入过滤语句。
- 在抓包主窗口的包结构显示栏中，将某个 DNS 数据包的某一属性值选定为显示过滤器的过滤条件。

10.2.4 拾遗补缺

DNS 协议颇为复杂，本章旨在教会读者解决与该协议及其实现有关的常见问题，因此只会例举若干常用的 DNS 流量显示过滤器，Wireshark DNS 流量显示过滤参数的完整列表请见 <http://www.wireshark.org/docs/dfref/d/dns.html>。

10.3 分析DNS协议的常规运作机制

本节将讲述如何判断 DNS 协议是否正常运作，会以某些场景为例，来看看 DNS 协议究竟会出那些问题。

10.3.1 准备工作

将 Wireshark 主机接入 LAN 交换机，激活交换机的端口镜像功能，将有待监控的交换机端口（连接 DNS 服务器或需要使用 DNS 的主机的端口）的流量重定向至 Wireshark 主机；运行 Wireshark 软件，选择正确的网卡，开始抓包。

10.3.2 操作方法

若 DNS 故障只存在于个别用户主机，在开启 LAN 交换机的端口镜像功能时，只需将连接该主机的交换机端口的流量重定向至 Wireshark 主机^①。

若 DNS 故障普遍存在于网络，在开启 LAN 交换机的端口镜像功能时，则应设法将 DNS 服务器的流量重定向至 Wireshark 主机。

- 若使用的 DNS 服务器部署在内网（即在所有内网主机的 TCP/IP 配置中，“DNS 服务器”配置项填的是内网 IP 地址），在开启 LAN 交换机的端口镜像功能时，应将连接了内网 DNS 服务器的交换机端口的流量重定向至 Wireshark 主机。

^① 译者注：原文是 “In case of user complains, configure the port mirror for monitoring the user device”，直译为“如遇用户投诉，配置端口镜像，监控用户设备”。作者的排障理念译者完全不能接受。译者认为，遇到用户投诉时，应首先了解是大面积投诉还是单个用户投诉，然后才能执行相应的排障动作，所以未按原文字面意思翻译。

- 若使用的 DNS 服务器部署在外部网络（即在所有内网主机的 TCP/IP 配置中，“DNS 服务器”配置项填的是外网 IP 地址，比如，ISP 的 DNS 服务器 IP 地址），在开启 LAN 交换机的端口镜像功能时，应将连接了通向那台外网 DNS 服务器的 ISP 链路的交换机端口的流量重定向至 Wireshark 主机。

10.3.3 幕后原理

域名或名称解析是 DNS 协议的主要功能之一，在网上冲浪时会用到。然而，在某些单位的内部网络中，DNS 协议同样发挥着非常重要的作用。在定义 DNS 协议的标准文档中，将 DNS 的功能归纳为以下三项。

- **域名空间：**定义了 DNS 名称的构成及分配方法。
- **域名注册：**定义了如何注册 DNS 名称，以及通过 DNS 服务器网络传递 DNS 名称的方法。
- **域名解析：**定义了如何将域名解析为 IP 地址。

本章将重点介绍 DNS 的域名解析功能，无论是上网冲浪，还是收发邮件，或是访问内网服务器，都会用到 DNS 的这一功能。DNS 协议的基本运作机制如图 10.5 所示。

DNS 协议的基本运作机制

应用程序（Web 浏览器和邮件客户端等）会通过内置于操作系统的解析器与 DNS 服务器沟通。解析器需要事先得知 DNS 服务器的 IP 地址，才能与之沟通（这一 IP 地址既可以是内网地址，也可以是外网地址）。查询 DNS 服务器的方式随操作系统而异。解析器和 DNS 服务器之间会相互交换 DNS 请求和 DNS 响应消息（见图 10.5）。

每个单位都可以在自己的内部网络中部署 DNS 服务器，该服务器需要与 ISP 的 DNS 服务器交互。对于家庭或小型分支机构网络，既可以由连接 Internet 的宽带路由器充当 DNS 服务器，也可以直接使用 ISP 的 DNS 服务器。也就是说，需要在每台主机的 TCP/IP 配置中，把宽带路由器的 IP 地址或 ISP 的 DNS 服务器的 IP 地址填入“DNS 服务器配置项”。

- 若使用宽带路由器充当 DNS 服务器，则内网主机发出的 DNS 查询消息，会先被宽带路由器上的 DNS 服务器处理；若该 DNS 服务器无法处理（查询不到相关域名），宽带路由器将会向 ISP 的 DNS 服务器发出查询请求。
- 若直接使用 ISP 的 DNS 服务器，则内网主机发出的 DNS 查询消息，将直接交由 ISP 的 DNS 服务器处理。

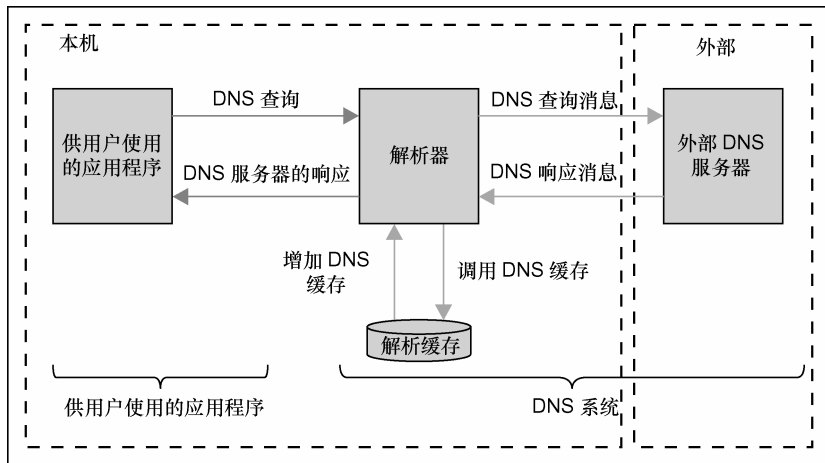


图 10.5

DNS 名字空间

DNS 名字空间基于分层的树形结构，如图 10.6 所示，以下是对其结构的简要描述。

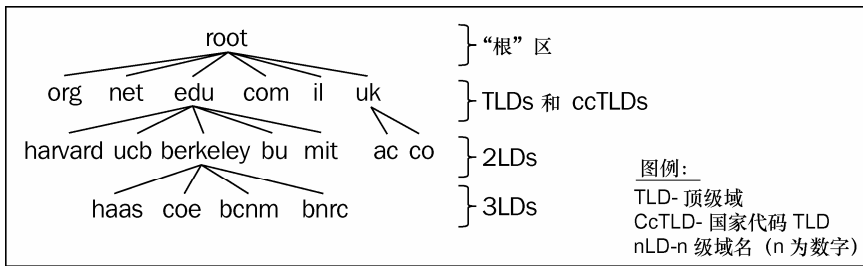


图 10.6

- 若干根名字服务器 (<http://www.iana.org/domains/root/servers>)。
- 若干顶级域 (Top Level Domain, TLDs) 名字服务器 (<http://www.iana.org/domains/root/db>)。
- 每个顶级域名都有属于自己的域名服务器。每个顶级域名还会包含众多二级域名^①。
- 众多二级域 (Second Level Domains, SLDs)，包含隶属于各组织机构或国家的域或域名。隶属于二级域的域名由指定的组织机构或国家来管理。

现以图 10.7 为例，对几个重要的 DNS 术语下定义。

① 译者注：原文还有一句“TLDs are the highest-level servers, for example, country servers as illustrated in the next diagram”。字面意思为“TLDs 是顶级服务器，比如，出现在图 10.6 中的国家服务器”。译者认为原文有误，未将其列入正文。在译者看来，作者对 DNS 协议的理解最多只有初级水平，因此他在这章里对许多 DNS 概念的解释都是错误的。

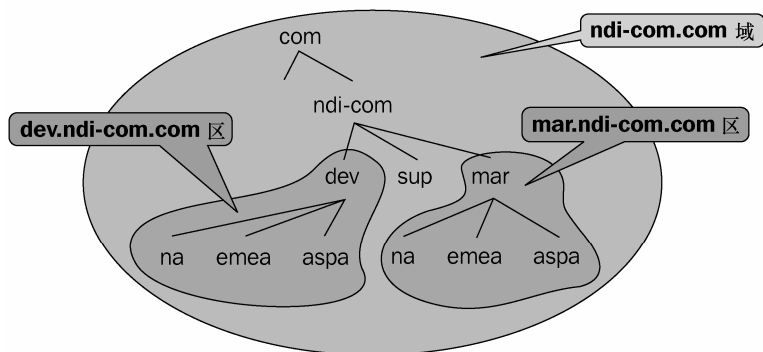


图10.7

- **域 (Domain):** 一个域是指由域名空间中所有分枝构成的一颗子树，域的名称（域名）就是这颗子树的端节点的名称，比如图 10.7 所示二级域 **ndi-com.com**。
- **区 (Zone):** 所有的顶级域以及众多二级（或低级）域都以“授权”（delegation）的方式，被划分成了更小也更容易管理的单元（unit），这样的单元被称为区。会有一或多台 DNS 服务器来专门负责记录并维护每个区下辖的所有域名信息。

DNS 解析过程

在以下两种情况下会用到 DNS 服务器。

- **组织机构内部网络通信:** DNS 服务器将会部署在内部网络，用来把内网域名解析为内部 IP 地址。
- **互联网通信:** 在浏览网页、收发 Email 之前，需要借助于 DNS 服务器，把公网域名解析为公网 IP 地址。

有时，还会出现组织机构内部网络中的主机需借助 DNS 服务器，同时访问内网和公网的情况。对于这种情况，当需要执行内网域名解析时，内网主机发出的 DNS 查询消息会由内网 DNS 服务器直接处理；当需要执行公网域名解析时，内网 DNS 服务器会把内网主机发出的 DNS 查询消息，转交给 ISP 的 DNS 服务器处理。

该怎样在自用的主机或 PC 上配置 DNS 服务器的 IP 地址呢？就理论而言，若只是为了上网冲浪，则可将世界上任何一台提供 DNS 解析服务的 DNS 服务器的 IP 地址（比如，Google 的 DNS 服务器 8.8.8.8），配置为本机使用的 DNS 服务器。但通常人们都会在主机的 DNS 服务器配置项中填入离自己最近的 DNS 服务器的 IP 地址（比如，提供上网线路的 ISP 的 DNS 服务器的 IP 地址）。若需同时访问内网和公网，则应将内网 DNS 服务器和 ISP 的 DNS 服务器分别配置为“首选 DNS 服务器”和“备用 DNS 服务器”。

有很多实用工具都可用来检测本机配置的 DNS 服务器的应答效果，如下所列。

- Google 的 **Namebench** (<http://code.google.com/p/namebench/downloads/detail?name=namebench-1.3.1-Windows.exe&can=2&q=>)。
- GRC 的 **DNS Benchmark** (<https://www.grc.com/dns/benchmark.htm>)。

要是本机配置的 DNS 服务器的应答效果“不佳”，就应该换一个 DNS 服务器的 IP 地址试试。

10.3.4 拾遗补缺

当运行于主机的应用程序进程需查询某一指定域名（或服务器名）的 IP 地址时，会与本机解析器交互，而本机解析器则与本机配置的 DNS 服务器交互。当 DNS 服务器的数据库中无待查的域名记录时，便会以递归（**recursive**）或反复（**iterative**）这两种模式之一，来响应查询请求^①。

如图 10.8 所示，当用户想访问 Web 站点 **www.packtpub.com**（比如，在浏览器里输入 **http://www.packtpub.com**，然后按下回车）时，若主机不知道这一域名的 IP 地址，则其解析器会向本地 DNS 服务器（主机的 TCP/IP 配置的 DNS 服务器配置项里已经填入这台 DNS 服务器的 IP 地址）发出 DNS 查询消息（图 10.8 中的消息 1），请求该 DNS 服务器告知域名 **www.packtpub.com** 的 IP 地址。

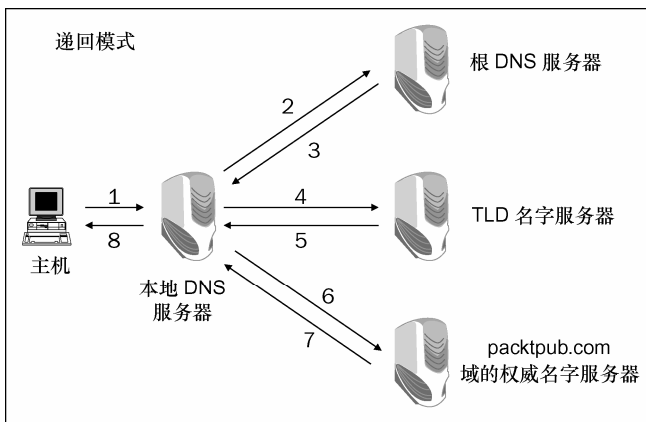


图10.8

若本地 DNS 服务器也不知道域名 **www.packtpub.com** 的 IP 地址，且对 **packtpub.com** 域或 COM TLD 的名字服务器的 IP 地址同样不得而知，则会将主机发出的 DNS 查询消息转发给另一台 DNS 服务器，以期查得相应的 IP 地址。对于本例，本地 DNS

① 译者注：原文中作者对 DNS 递归解析和反复解析的解释全都大错特错，译者对译文做了大幅调整，如有不妥，请指正。

服务器将会联系根名字服务器之一(根名字服务器的 IP 地址世人皆知)(图 10.8 中的消息 2), 根名字服务器会返回 DNS 响应消息, 其中会包含 COM TLD 的名字服务器的 IP 地址(图 10.8 中的消息 3)。

有了上述信息之后, 本地 DNS 服务器继续联络 COM TLD 的名字服务器(图 10.8 中的消息 4), COM TLD 的名字服务器会返回 `packtpub.com` 域的名字服务器的 IP 地址(图 10.8 中的消息 5)。

向 `packtpub.com` 域的名字服务器发出 DNS 查询消息(图 10.8 中的消息 6), `packtpub.com` 域的名字服务器做出了应答(图 10.8 中的消息 7)之后, 本地 DNS 服务器便知道了 Web 站点 `www.packtpub.com` 的 IP 地址。

此时, 本地 DNS 服务器将会向主机发出 DNS 响应消息(图 10.8 中的消息 8), 告知其 Web 站点 `www.packtpub.com` 的 IP 地址。这样一来, 用户便可以在主机上成功访问该 Web 站点了。

在上述 DNS 查询/解析过程中, 主机发出的是 DNS 递归查询请求, 要求本地 DNS 服务器返回精确的待查域名的 IP 地址(或返回出错说明)。本地 DNS 服务器执行的是 DNS 递归解析操作, 同时还得执行 DNS 反复查询操作——由其发出的 DNS 查询消息并不要求 `packtpub.com` 域或 COM TLD 的名字服务器给出精确答案。也就是说, 本地 DNS 服务器同时在执行 DNS 反复查询和递归解析操作, 而根名字服务器或 COM TLD 的名字服务器执行的是 DNS 反复解析操作。一言以蔽之, 收到 DNS 递归查询消息(RD 标记位置 1 的 DNS 查询消息)时, DNS 服务器(若支持 DNS 递归解析功能)一定要精确给出被查域名的 IP 地址(或返回出错说明); 收到 DNS 递归反复消息(RD 标记位置 0 的 DNS 查询消息)时, DNS 服务器只要提供更靠近被查域名的所在区(zone)的名字服务器的 IP 地址即可。根名字服务器、TLD 名字服务器以及某些 SLD 名字服务器只支持反复解析操作, 而 ISP 的 DNS 服务器一般都支持递归解析操作。

10.4 DNS故障分析

上一节简单介绍了 DNS 协议的基本运作机制。本节将介绍如何发现 DNS 故障及故障根源。

10.4.1 准备工作

无论是上网冲浪, 还是在内网中使用网络应用程序, 只要有 DNS 故障, 就会让人觉得网速很慢。本节会介绍如何判断网速很慢是否因 DNS 故障所致, 同时会提供解决 DNS 故障的方法。