



信息安全技术丛书



Wireshark

网络分析实战

Network Analysis Using
Wireshark Cookbook

[以色列] Yoram Orzach 著

古宏霞 孙余强 译



人民邮电出版社
POSTS & TELECOM PRESS



Wireshark

网络分析实战

Network Analysis Using
Wireshark Cookbook

[以色列] Yoram Orzach 著
古宏霞 孙余强 译

人民邮电出版社
北京

图书在版编目 (CIP) 数据

Wireshark网络分析实战 / (以) 奥扎赫
(Orzach, Y.) 著 ; 古宏霞, 孙余强译. — 北京 : 人民
邮电出版社, 2015.2
ISBN 978-7-115-37771-5

I. ①W… II. ①奥… ②古… ③孙… III. ①计算机
网络—通信协议 IV. ①TN915.04

中国版本图书馆CIP数据核字(2014)第297046号

版权声明

Copyright © Packt Publishing 2013. First published in the English language under the title Network Analysis Using Wireshark Cookbook.
All Rights Reserved.

本书由英国 Packt Publishing 公司授权人民邮电出版社出版。未经出版者书面许可, 对本书的任何部分不得以任何方式或任何手段复制和传播。

版权所有, 侵权必究。

-
- ◆ 著 [以色列] Yoram Orzach
 - 译 古宏霞 孙余强
 - 责任编辑 傅道坤
 - 责任印制 张佳莹 彭志环
 - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
 - 邮编 100164 电子邮件 315@ptpress.com.cn
 - 网址 <http://www.ptpress.com.cn>
 - 三河市中晟雅豪印务有限公司印刷
 - ◆ 开本: 800×1000 1/16
 - 印张: 26.75
 - 字数: 583 千字
 - 印数: 1-3 000 册
 - 2015 年 2 月第 1 版
 - 2015 年 2 月河北第 1 次印刷
 - 著作权合同登记号 图字: 01-2013-9040 号
-

定价: 79.00 元

读者服务热线: (010) 81055410 印装质量热线: (010) 81055316

反盗版热线: (010) 81055315

内容提要

本书采用步骤式为读者讲解了一些使用 Wireshark 来解决网络实际问题的技巧。

本书共分为 14 章，其内容涵盖了 Wireshark 的基础知识，抓包过滤器的用法，显示过滤器的用法，基本/高级信息统计工具的用法，Expert Info 工具的用法，Wireshark 在 Ethernet、LAN 及无线 LAN 中的用法，ARP 和 IP 故障分析，TCP/UDP 故障分析，HTTP 和 DNS 故障分析，企业网应用程序行为分析，SIP、多媒体和 IP 电话，排除由低带宽或高延迟所引发的故障，认识网络安全等知识。

本书适合对 Wireshark 感兴趣的网络从业人员阅读，也适合高校网络相关专业的师生阅读。

关于作者

Yoram Orzach 毕业于色列技术学院 (Israel Institute of Technology), 持有该校科学学士学位。1991~1995 年, 他以系统工程师的身份就职于 Bezeq 公司, 从事传输及接入网相关的工作。1995 年, 他从 Leadcom 集团 (Leadcom group) 加盟 Netplus 公司, 并转型为技术管理者。自 1999 年起, 他开始担任 NDI 通信公司 (NDI Communications, <http://www.ndi-com.com/>) 的 CTO, 负责并参与该公司在全球范围内的数通网络的设计、实施及故障排除工作。Yoram 对大型企业网络、服务提供商网络及 Internet 服务提供商网络极有心得, Comverse、Motorola、Intel、Ceragon Networks、Marvel 以及 HP 等公司都接受过他提供的服务。Yoram 在网络设计、实施及故障排除方面浸淫多年, 在研发 (R&D)、工程、IT 团队的培训方面也有丰富的经验。

致 谢

首先，要感谢我的全家。感谢我的父母：我的父亲 Israel——世上独一无二的智者——全家遇难，他以 35 公斤的瘦弱之躯只身从（纳粹针对犹太人的）大屠杀中脱生，40 年后，他成为了电信行业首屈一指的专家；我的母亲 Selma 则教会我很多。感谢我贤惠的妻子 Ena，谢谢你过去 20 多年来在工作以及其他方面给我的支持。感谢我的孩子 Nadav、Dana 和 Idan，你们的成就让老爸自愧不如。感谢我的妹妹 Hana、妹夫 Ofer 以及我的外甥们。

要感谢我的诸多同事。最先应该感谢 Reuven Matzliach，20 世纪 90 年代末，他和我在 Comverse 公司开始筹建 IP 培训中心（IP college），并促成了该公司的网络从 TDM 向 IP 转型。在转型期间，他多次助我渡过难关。同样要感谢 Omer Fuchs 和 Moshe Sakal，感谢二位在那个大项目里给我的帮助。我还要感谢其他同事及朋友，只是人名太多无法一一列出。

要感谢 Lior Tzuberi，感谢你提供的诸多 WireShark 使用技巧及研究案例。要感谢 Hanan Man，感谢你那个有趣的网络设计。要感谢 Yoel Saban 和 Rami Kletshevsky，感谢你们所奉献的诸多经典的网络设计，你们的设计团队是我见过的最出色的团队之一。要感谢 Zvi Shacham，感谢你传授给我的数通网络的经验。要感谢 Asi Alajem，感谢你那个很有意思的网络设计；感谢 Oren Gerstner，感谢你提供的若干经典的无线网络案例。感谢 Chen Heffer，你是我所认识的最出色的网络安全专家。要感谢 Yoni Zini，感谢你在系统方面给我提供的帮助。感谢 Ibrahim Jubram，感谢你奉献的那些精彩的移动网络案例。感谢 Ofer Sela，感谢我们一起参与的那些非常有意思的项目。感谢 Amir Lavi 和 Eran Niditz，感谢你们提供的那些趣味横生的案例。感谢 Avner Mimon，感谢你提供的诸多绝佳的 WireShark 使用技巧。此外，还要感谢许许多多的其他人。

我要感谢教会我诸多网络技能的培训专家。30 年前，我曾以为给人上课也就是图一乐；是你们让我懂得，培训也是一种职业。我要感谢 Harriet Rubin、Merav Sagi、Rvital Keinan、Guy Einav、Raanan Dagan 以及其他许许多多的培训专家们。

我要特别感谢 Yoav Nokrean 跟他的儿子 Eran，要感谢你们给我出的那些主意，以及对我全方位的支持。

我还要感谢曾与我朝夕相处的诸位前同事；感谢以色列、欧洲、北美、东亚以及世界各地的客户。排除网络故障的手法总是千篇一律，而屋外的风景则十里不同天。

尤其要感谢某些设计出垃圾网络的网络设计“专家”、捣鼓出离奇 TCP/IP 实现的协议开发人员，以及连线缆都能接错的 IT 运维人员，外加那些以为设备只要接上线缆便能正常运行的 IT 施工部门。解决由那些“挫人”、“挫事”引起的故障是学习网络技术的最佳途径。

要感谢我教过的学生们，感谢你们问我的所有刁钻问题，以及带来的有意思的网络案例；每一次给你们上课我都能从中获益。我真不知道还有什么事能把笔记本电脑连接进真实的网络，解决故障更有意思了。

要感谢我所钦佩的一干网络及安全界的前辈——Vint Cerf、Bob Kahn、Radia Perlman、Adi Shamir、Ronald Rivest、Van Jacobson、Steven McCanne 等。没有你们，哪来今天的一切。

最后，要感谢 Packt Publishing 出版社能有出版以 Wireshark 为主题的图书的意愿，并包容我完成本书的写作。

关于技术审稿人

Charles L. Brooks 是 Security Technical Education 公司的创始人和首席咨询顾问，该公司的业务都与 IT 技术有关，包括 IT 图书的创作和评论、IT 培训和教案设计等。Charles 还在积极推广由波士顿大学开办的数据通信及网络技术方面的远程教育课程，并且在布兰代斯大学教授网络安全、软件开发安全、虚拟化安全以及云计算基础架构等方面的课程，在拉布研究生专业进修学院教授信息安全编程（涉及 MS 操作系统）方面的课程。创办 Security Technical Education 公司（www.securityteched.com）之前，Charles 在 EMC 和 RSA 公司任高级技术教育顾问一职，负责存储安全、大数据、网络安全分析及网络取证等主题的课件开发。加盟 EMC 公司之前，Charles 曾干过许多年的软件工程师、团队组长（team leader）以及软件架构师；最近，他还作为 GTE Internetworking and Genuity 公司的系统工程师，负责管理该公司所提供的 VPN 业务。

Charles 握有克拉克大学英语专业的学士和硕士学位、波士顿大学计算机信息系统科学专业的硕士学位，以及包括 CISSP、CEH 和 CHFI 在内的多张行业证书。

我要感谢 Helyn Pultz，感谢她这么多年来的鼓励、支持和忠告。

Praveen Darshanam 在 McAfee、Cisco 及 iPolicy Networks 公司任职多年，有 7 年以上的信息安全方面的经验。他的强项和兴趣包括：漏洞研究、（病毒或木马）特征的研究、Snort、应用程序安全，以及恶意软件分析等。他持有电子工程（EE）专业的学士学位和控制及仪表专业的硕士学位；他的学士学位获取自印度最好的学府之一。他还拥有包括 CHFI、CEH 和 ECSA 在内的多张行业证书。

Ritwik Ghoshal 是 Oracle 公司资深的安全分析师，负责 Oracle 公司的软、硬件安全性保障工作。他的工作范围包括网络安全、操作系统安全和虚拟化安全等。他于 2008 年加入 SUN 公司，在 2010 年为 Oracle 公司效力之前（Oracle 公司在那一年收购了 SUN 公司），他一直是 SUN 公司安全工程团队和 Solaris 团队的成员。在 Oracle 公司，Ritwik 继续负责所有 Sun 产品线以及 Oracle Linux 和虚拟化产品线的安全性保障工作。

Ritwik 于 2008 年获取了 Heritage Institute of Technology（印度加尔各答）计算机科学与工程专业的学士学位。

我要感谢我的父母以及 Sara E Taverner，感谢你们对我持续的帮助与支持。

Gilbert Ramirez 自 Wireshark 第一版发布之日起，就是该软件的贡献者。他为该软件添加了协议剖析模块和某些关键组件（比如，显示过滤引擎等），同时为 Wireshark 向 Windows 平台的移植做出过贡献。他目前供职于 Cisco 公司，负责软件系统及软件工具的构建。

Gilbert 是多本 Wireshark 主题书籍的作者，包括 *Wireshark & Ethereal Network Protocol Analyzer Toolkit*、*Ethereal Packet Sniffing* 以及 *Nessus, Snort, & Ethereal Power Tools* 等，这些书籍均由 Syngress Publishing Inc. 出版社出版。

前言

Wireshark 早已成为网络分析领域里的标配工具,随着 Internet 和 TCP/IP 网络的极速发展,该工具会受到网络分析专家及排障工程师的热捧,同时也会获得(网络协议或应用程序)研发工程师们的青睐,因为后者需要知道协议在网络中的实际运作方式以及在运行时所碰到问题。

本书的写作立足于实战。本书第 1 部分(从第 1 章到第 6 章)简要介绍了 Wireshark 软件(的架构)及其各个组件的使用方法。这部分的内容包括 Wireshark 的启动方法、在网络中的安置方法、信息统计(statistical)工具的使用方法以及专家(Expert)系统的使用方法。本书第 2 部分(从第 7 章到第 14 章)详述了如何使用 Wireshark 来分析并排除某些常用网络协议的故障;TCP/IP 协议栈(特别是 TCP 性能问题)是这部分的重点内容。此外,还会简要介绍 HTTP、SMTP、POP 以及 DNS 等协议,外加数据库、Citrix 和 Microsoft 终端服务器、IP 电话以及多媒体应用所使用的协议,以上协议都属于常用的 Internet 协议。本书最后一章(第 14 章)涉及网络安全,介绍了如何利用 Wireshark 去发现网络中的安全缺陷,同时交代了与安全性有关的其他问题。

本书的书名既然叫《Wireshark 网络分析实战》,那么其内容一定是由一系列利用 Wireshark 对网络故障进行有效性、针对性分析的诀窍所构成。书中包含的每个诀窍都与某一具体的网络故障相关联,在介绍如何使用 Wireshark 解决相关故障时,作者会指出应关注 Wireshark 工具的哪些地方、哪些(抓包)内容以及正在处理的故障的起因。为求表述圆满,每个诀窍都会包含相应主题的理论基础知识,好让尚未掌握基础概念的读者先行“武装”自己。

书中包含了许多示例,所有示例均来源于真实案例。作者在处理这些案例时,所花费的时间虽长短不一(有些只花了几分钟时间,有些则要花几小时甚至几天),但所遵循的原则只有一条,那就是:按部就班,选择正确的工具,当应用程序开发者肚里的“蛔虫”,外加从网络的角度思考问题。只要按此原则行事,兼之能活学活用 Wireshark,定能将故障查个水落石出。本书的目的也正在于此,享受这一切吧。

本书所含内容

第 1 章, Wireshark 简介,开启了对 Wireshark 的简要介绍,同时阐述了在有效开展网络分析时, Wireshark 主机(或程序)的布放(或安装)位置问题。在本章,读者会学到如何配置 Wireshark 的基本(运行)参数、启动窗口、时间参数及配色规则(coloring rules),但本章

的重点内容是如何配置首选项窗口 (Preferences window)。

第 2 章, 抓包过滤器的用法, 介绍了抓包过滤器的使用方法。要让 Wireshark 只抓取必要的数据包, 抓包过滤器不可或缺。本章会详述这一过滤器的配置方法, 外加如何使用这一过滤器, 让 Wireshark 只抓取网络工程师“心仪的”数据包。

第 3 章, 显示过滤器的用法, 本章说明了显示过滤器的使用方法。通过 Wireshark 抓取到数据包之后, 可定义显示过滤器, 让 Wireshark 只显示出“必要的”数据包。本章会详述这一过滤器的配置方法, 同时会介绍如何利用其来帮助排除网络故障。

第 4 章, 基本信息统计工具的用法, 说明了 Wireshark 自带的信息统计工具的基本用法, 包括如何用 Statistics 菜单下的某些菜单项, 生成与相互“沟通”的主机 (who is talking)、会话 (conversations) 及 HTTP 协议流量等有关的信息统计报表。

第 5 章, 高级信息统计工具的用法, 介绍了 Wireshark 提供的信息统计工具的高级用法, 包括如何用 Statistics 菜单下的有关菜单项, 生成 IO 图 (IO graph) 及 TCP 流图 (TCP stream graph), 上述图形可成为网络及应用程序性能分析的重要依据。

第 6 章, Expert Infos 工具的用法, 讲解了如何使用 Wireshark 内置的专家系统 (Expert system) 工具, 该工具的功能极为强大, 可对网络中发生的有可能会影响到应用程序正常交付的各种事件 (比如, TCP 重传、零窗口 [zero-window]、TTL 过低/路由环路、报文段失序等) “洞察秋毫”。

第 7 章, Ethernet、LAN 交换以及无线 LAN, 简要介绍了与 Ethernet 协议及 LAN 交换有关的基本概念, 探讨了可能会发生在第二层的网络故障。此外, 本章还重点介绍了无线 LAN (Wi-Fi) 相关知识, 包括如何测试该网络, 以及如何解决发生在该网络中的故障。

第 8 章, ARP 和 IP 故障分析, 简要介绍了与 ARP 和 IP 有关的基本概念, 探讨了 IP 连通性故障及路由环路故障的解决方法。此外, 本章还讲解了如何发现 IP 地址冲突, 如何解决 DHCP 及其他相关故障。

第 9 章, UDP/TCP 故障分析, 重点关注第四层协议——UDP 和 TCP——着重讨论与 TCP 性能有关的问题。本章包含多个定位 TCP 性能问题的诀窍, TCP 重传问题、重复确认问题 (duplicate ACK)、滑动窗口 (sliding-window) 问题 (即 window-full 和 zero-window 问题)、重置 (reset) 问题等都属于 TCP 性能问题。

第 10 章, HTTP 和 DNS, 重点关注 DNS、HTTP 及 HTTPS 协议, 将介绍这三种协议的运作方式, 同时还会讲解如何排除与三种协议有关的故障。

第 11 章, 企业网应用程序行为分析, 探讨了包括 FTP、Mail 协议以及与终端服务和数据

库有关的常用网络协议，在介绍网络故障对上述应用程序的影响的同时，还会给出基本的排障思路。

第 12 章，SIP、多媒体和 IP 电话技术，介绍了如何利用 Wireshark，去定位及排除依靠 IP 传送的语音及视频（voice and video over IP）故障，此类故障包括 VoIP SIP 连通性故障、RTP/RTCP 故障和视频故障（比如，画面停顿[picture freezing]和画质不佳）等。

第 13 章，排除由低带宽或高延迟所引发的故障，介绍了如何利用 Wireshark，去定位由低带宽、高延迟及高抖动所引发的故障。本章会细述当网络中存在高延迟及高抖动现象时，TCP 的种种表现，同时会交待怎样才能让 TCP 表现得更好。

第 14 章，认识网络安全，本章重点关注 TCP/IP 网络的安全性，包括如何发现可能会对网络构成危害的扫描行为及 SYN、DoS/DDoS 等攻击行为。本章还会提供发现攻击流量模式的多个诀窍，同时说明这些攻击是如何发动的。

附录、链接、工具及进阶阅读，提供了进一步学习 Wireshark 软件的重要链接，包括各种学习资源和其他各种辅助软件等。

阅读准备

阅读本书之前，请先下载并安装 Wireshark 软件，下载链接为 www.wireshark.org。

本书的读者对象

本书的读者对象包括使用 Wireshark 进行网络分析和排除网络故障的研发工程师、技术支持工程师以及 IT 行业的技术管理人员。阅读本书的读者需掌握网络的基本概念，但不要求读者对具体的协议或厂商实现有深入的了解。

目 录

第 1 章 Wireshark 简介	1
1.1 Wireshark 简介	1
1.2 安置 Wireshark (程序或主机)	2
1.3 开始抓包	9
1.4 配置启动窗口	14
1.5 配置时间参数	20
1.6 定义配色规则	22
1.7 数据文件的保存、打印及导出	24
1.8 通过 Edit 菜单中的 Preferences 菜单项, 来配置 Wireshark 主界面	28
1.9 配置 Preferences 窗口中的 Protocol 选项	33
第 2 章 抓包过滤器的用法	37
2.1 简介	37
2.2 配置抓包过滤器	38
2.3 配置 Ethernet 过滤器	42
2.4 配置主机和网络过滤器	46
2.5 配置 TCP/UDP 及端口过滤器	50
2.6 配置复合型过滤器	53
2.7 配置字节偏移和净载匹配型过滤器	55
第 3 章 显示过滤器的用法	58
3.1 简介	58
3.2 配置显示过滤器	59
3.3 配置 Ethernet、ARP、主机和网络过滤器	67
3.4 配置 TCP/UDP 过滤器	71
3.5 配置协议所独有的显示过滤器	78
3.6 配置字节偏移型过滤器	81
3.7 配置显示过滤器宏	82

第 4 章 基本信息统计工具的用法	84
4.1 简介	84
4.2 Statistics 菜单中 Summary 工具的用法	85
4.3 Statistics 菜单中 Protocol Hierarchy 工具的用法	87
4.4 Statistics 菜单中 Conversation 工具的用法	90
4.5 Statistics 菜单中 Endpoints 工具的用法	94
4.6 Statistics 菜单中 HTTP 工具的用法	96
4.7 配置 Flow Graph (数据流图), 来查看 TCP 流	101
4.8 生成与 IP 属性有关的统计信息	103
第 5 章 高级信息统计工具的用法	107
5.1 简介	107
5.2 配置与显示过滤器结合使用的 IO Graphs 工具, 来定位与网络性能有关的问题	108
5.3 用 IO Graphs 工具测算 (链路的) 吞吐量	112
5.4 IO Graphs 工具的高级配置方法 (启用 Y 轴 Unit 参数的 Advanced 选项)	117
5.5 TCP StreamGraph 菜单项中 Time-Sequence (Stevens)子菜单项的用法	125
5.6 TCP StreamGraph 菜单项中 Time-Sequence (tcp-trace)子菜单项的用法	128
5.7 TCP StreamGraph 菜单项中 Throughput Grap 子菜单项的用法	131
5.8 TCP StreamGraph 菜单项中 Round Trip Time Graph 子菜单项的用法	133
5.9 TCP StreamGraph 菜单项中 Window Scaling Graph 子菜单项的用法	135
第 6 章 Expert Info 工具的用法	137
6.1 简介	137
6.2 如何使用 Expert Info 工具执行排障任务	137
6.3 认识 Errors 事件	145
6.4 认识 Warnings 事件	147
6.5 认识 Notes 事件	149
第 7 章 Ethernet、LAN 交换及无线 LAN	152
7.1 简介	152
7.2 发现广播及错包风暴	152
7.3 生成树协议分析	159
7.4 VLAN 和 VLAN tagging 故障分析	168
7.5 无线 LAN (WiFi) 故障分析	172
第 8 章 ARP 和 IP 故障分析	177
8.1 简介	177

8.2	与 ARP 有关的连通性网络故障分析.....	178
8.3	IP 流量分析工具的用法.....	186
8.4	利用 GeoIP 来查询 IP 地址的归属地.....	189
8.5	发现 IP 包分片问题.....	192
8.6	路由选择故障分析.....	197
8.7	发现 IP 地址冲突.....	200
8.8	DHCP 故障分析.....	204
第 9 章	UDP/TCP 故障分析	208
9.1	简介.....	208
9.2	配置 Preferences 窗口内 protocol 选项下的 UDP 和 TCP 协议参数, 为排除故障做准备.....	209
9.3	TCP 连接故障.....	213
9.4	TCP 重传现象——源头及原因.....	219
9.5	重复确认 (duplicate ACKs) 和快速重传 (fast retransmissions) 现象.....	229
9.6	TCP 报文段失序现象.....	232
9.7	TCP Zero Window、Window Full、Window Change 以及其他包含 Window 字样的提示信息.....	235
9.8	TCP 重置 (reset) 及原因.....	240
第 10 章	HTTP 和 DNS	242
10.1	简介.....	242
10.2	筛选 DNS 流量.....	243
10.3	分析 DNS 协议的常规运作机制.....	247
10.4	DNS 故障分析.....	252
10.5	筛选 HTTP 流量.....	260
10.6	配置 Preferences 窗口中 protocol 选项下的 HTTP 协议参数.....	263
10.7	HTTP 故障分析.....	266
10.8	导出 HTTP 对象.....	272
10.9	HTTP 数据流分析及 Follow TCP Stream 窗口.....	274
10.10	HTTPS 协议流量分析——SSL/TLS 基础.....	277
第 11 章	企业网应用程序行为分析	286
11.1	简介.....	286
11.2	摸清流淌于网络中的流量的类型.....	287
11.3	FTP 故障分析.....	289
11.4	E-mail 协议 (POP、IMAP、SMTP) 流量及故障分析.....	295

11.5	MS-TS 和 Citrix 故障分析	305
11.6	NetBIOS 协议故障分析	308
11.7	数据库流量及常见故障分析	317
第 12 章	SIP、多媒体和 IP 电话	322
12.1	简介	322
12.2	使用内置于 Wireshark 的 IP 电话及多媒体流量专用分析工具	323
12.3	SIP 故障分析	330
12.4	RTP/RTCP 故障分析	341
12.5	视频及视频监控应用排障场景	349
12.6	IPTV 应用排障场景	353
12.7	视频会议应用排障场景	354
12.8	排除 RTSP 协议故障	356
第 13 章	排除由低带宽或高延迟所引发的故障	361
13.1	简介	361
13.2	测量通信链路的总带宽	361
13.3	测量每个用户及每种应用所占用的通信链路的带宽	366
13.4	借助 Wireshark, 获悉链路上的延迟及抖动状况	367
13.5	发现因高延迟/高抖动所引发的应用程序故障	370
第 14 章	认识网络安全	377
14.1	简介	377
14.2	发现异常流量模式	378
14.3	发现基于 MAC 地址和基于 ARP 的攻击	384
14.4	发现 ICMP 和 TCP SYN/端口扫描	385
14.5	发现 DoS/DDoS 攻击	393
14.6	发现高级 TCP 攻击	397
14.7	发现暴力破解 (brute-force) 攻击	400
附录	链接、工具及阅读资料	405

第1章

Wireshark简介

本章涵盖以下内容：

- ▶ 安置 Wireshark（主机/程序）；
- ▶ 开始抓包；
- ▶ 配置启动窗口；
- ▶ 配置时间参数；
- ▶ 调整配色规则；
- ▶ 保存、打印及导出数据；
- ▶ 配置用户界面（点击 EDIT 菜单的 Preferences 菜单项，会弹出 Preferences 窗口。所谓配置用户界面，就是配置该窗口中 User Interface 配置选项里的内容）；
- ▶ 配置协议参数（即配置 Preferences 窗口中 Protocol 配置选项里的内容）。

1.1 Wireshark简介

本章将介绍 Wireshark 所能行使的基本任务。本书的前言曾提到过网络排障以及内置于 Wireshark 能帮助排障的各种工具。一旦决定动用 Wireshark 协议分析软件，在使用之前，则有必要先确定该软件在网络中的部署（或安装）位置。除此之外，还得对该软件做一些基本的配置，至少应让其界面看起来更为友好。

用 Wireshark 执行基本的抓包操作，配置起来并不麻烦，但是该软件也包含了很多高级配

置选项，可用来应对某些特殊情况。这样的特殊情况包括令 Wireshark 在某条链路上持续抓取数据包的同时，将抓包文件切分为多个较小的文件；让 Wireshark 在抓包主窗口的数据包列表区域只显示（发包/收包）主机（或设备）的名称而非 IP 地址等。本章会介绍如何在 Wireshark 中配置这些高级选项，以应对上述特殊情况。

把 Wireshark 主机（或程序）安置（或安装）在网络中的哪个地方，令其行使抓包功能，是本章的一大重点。应将 Wireshark 置于防火墙“身前”还是“身后”呢？应置于路由器的 WAN 一侧，还是 LAN 一侧呢？到底应在上述的哪些地方才能正确采集到自己想要的数据呢？这些问题的答案、安置 Wireshark 的诀窍以及更多与 Wireshark 抓包有关的内容请见 1.2 节。

如何配置 Wireshark 时间参数（亦即如何配置 Wireshark，让其按网管人员的意愿，来显示数据包的收、发时间的格式），是本章的又一重点内容。在抓取时间敏感型应用程序数据，且希望弄清隶属于同一条 TCP 连接或 UDP 流的数据包间的“交互”时间时，如何配置 Wireshark 时间参数将显得尤为重要。

1.4 节会介绍 Wireshark 数据文件的操作，包括：如何保存抓包数据（是完整还是部分保存）；如何保存经过过滤的数据文件；如何以各种文件格式来导出抓包数据；如何合并抓包文件（比如，将两份 Wireshark 抓包文件合二为一，这两份抓包文件中的数据分别抓取自不同路由器上的以太网接口）等。

本章还会介绍如何调整 Wireshark 配色规则。所谓调整配色规则，是指配置 Wireshark，令其以不同的颜色来显示不同类型（或不同协议）的数据包。虽然 Wireshark 默认开启了一套配色方案，但在某些特殊情况下，网管人员可能需要对配色方案进行调整，让受监控的特殊协议的数据包或让 Wireshark 把识别出的某些值得关注的错误/事件，以引人注目的颜色显示。1.6 节将包括这些内容。

本章最后两节会讨论 Wireshark 首选菜单项的配置。这两节会讲解 Wireshark 用户界面的配置（包括如何配置 Wireshark 主界面；如何在抓包面板中添加或删除数据包属性栏；如何查看数据包属性栏等），以及针对具体协议的配置（包括如何将某个 TCP 端口号默认解析为应用服务名称；如何验证 TCP/UDP 校验和；如何验证 TCP 时间戳；如何解码数据包中各协议头部内的各个字段等）。

1.2 安置 Wireshark（程序或主机）

看到了网络故障的表象，决定通过 Wireshark 抓包来查明故障原委之前，应确定 Wireshark（程序或主机）的（安装或部署）位置。为此，需弄到一张精确的网络拓扑图（至少也得弄清楚故障所波及的那部分网络的拓扑结构），如图 1.1 所示。

安置 Wireshark 的原理非常简单。首先，应圈定要抓取哪些（哪台）设备发出的流量；其次，要把安装了 Wireshark 的主机（笔记本）连接到受监控设备所连交换机；最后，开启交换机的端口镜像（或端口监控）功能，把受监控设备发出的流量“重定向”给 Wireshark 主机。

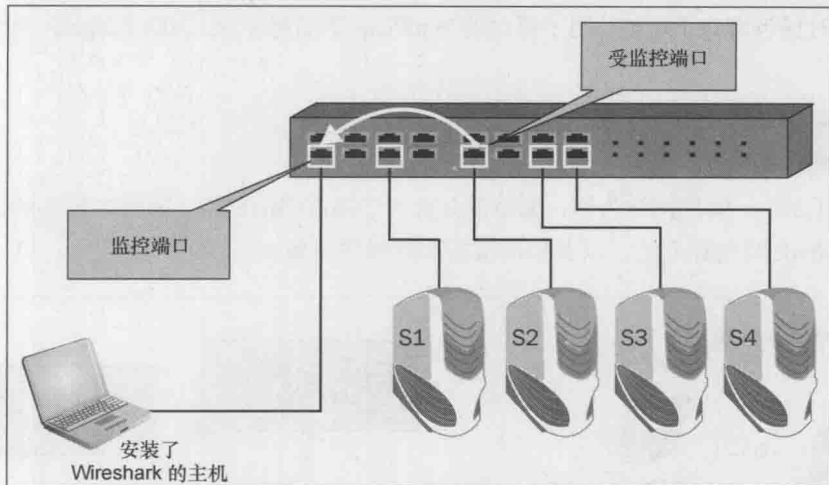


图1.1

可利用 Wireshark 监控 LAN 端口、WAN 端口、服务器/路由器端口或连接到网络的任何其他设备的流量。

如图 1.1 所示，利用 Wireshark 软件（安装在交换机左边的 PC 上）外加交换机的端口镜像（也叫做端口监控，需在交换机上激活该特性，流量镜像的方向已在图中标出）功能，便可以监控到进、出服务器 S2 的所有流量。当然，也可以在服务器 S2 上直接安装 Wireshark，如此行事，便能直接在服务器 S2 上监控进、出该服务器的流量了。

某些厂商的交换机还支持以下流量监控特性。

- **监控整个 VLAN 的流量：**即监控整个 VLAN（服务器 VLAN 或语音 VLAN）的流量。可借助该特性，在指定的某一具体 VLAN 内进行流量监控。
- **“多源归一”的流量监控方式：**以图 1.1 为例，借助该特性，可让 Wireshark 主机同时监控到服务器 S1 和 S2 的流量。
- **方向选择：**可选择监控入站流量、出站流量或同时监控出、入站流量。

1.2.1 准备工作

使用 Wireshark 抓包之前，请先访问 Wireshark 官网，下载并安装最新版本的 Wireshark。

Wireshark 软件的后续更新会发布在其官网 <http://www.wireshark.org> 的 **Download** 页面下，其最新的稳定版本也可以从该页面下载。

每个 Wireshark Windows 安装包都会自带 WinPcap 驱动程序的最新稳定版本，WinPcap 驱动程序为实时抓包所必不可少。用于抓包的 WinPcap 驱动程序为 UNIX Libpcap 库的 Windows 版本。

1.2.2 操作方法

现以图 1.2 这一典型网络为例，来简单分析一下该网络的架构、网络中设备的部署及运作方式、Wireshark 的安置方法，以及如何按需配置网络设备。

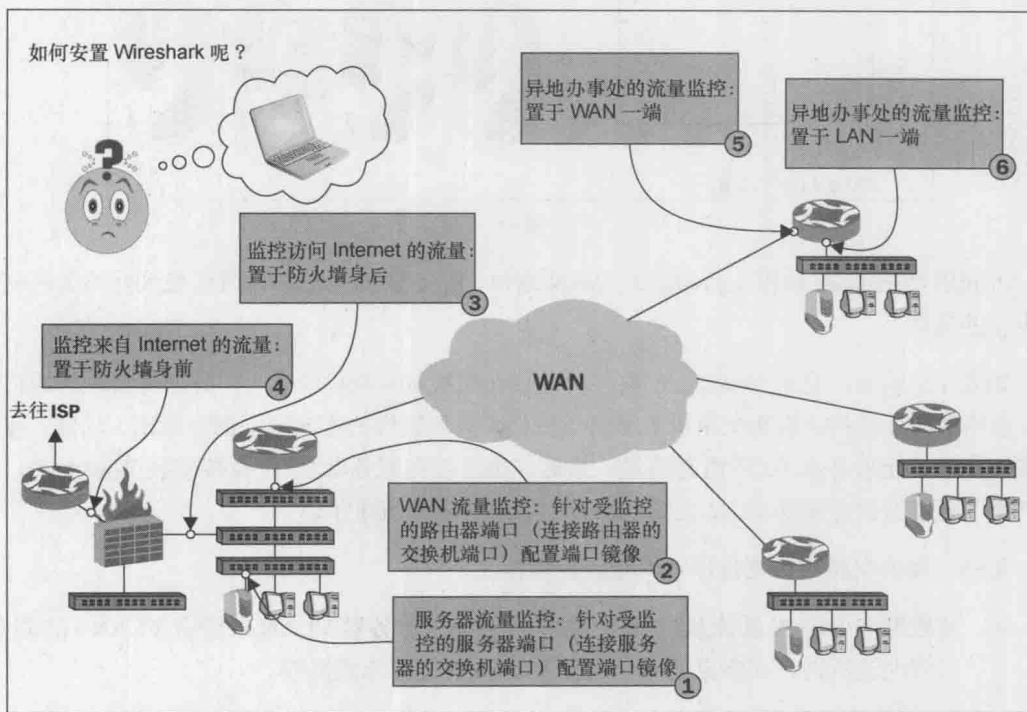


图 1.2

请读者仔细研究一下图 1.2 所示的简单而又常见的网络拓扑结构。

服务器流量监控

像服务器流量监控这样的需求，在实战中非常常见。要想监控到某台服务器（收/发）的流量，既可以在交换机上针对连接服务器的端口配置端口镜像（如图 1.2 中的编号①所示），

将流量“重定向”至 Wireshark 主机，也可以在服务器上直接安装 Wireshark。

路由器流量监控

要想监控进、出路由器的流量，监控其 LAN 端口（如图 1.2 中的编号②和⑥所示）或 WAN 端口（如图 1.2 中的编号⑤所示）都可以办到。

路由器 LAN 端口的流量监控起来比较简单，只要在交换机上配置端口镜像，把与路由器 LAN 口相连的端口的流量“重定向”至连接 Wireshark 主机的端口。要想监控路由器 WAN 口的流量，则要在路由器 WAN 口和 SP（服务提供商）网络之间部署一台交换机，在这台交换机上配置端口镜像，如图 1.3 所示。

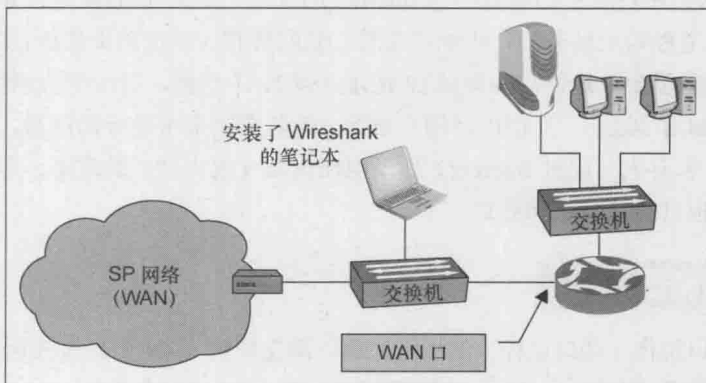


图1.3

在 SP 网络与路由器 WAN 口之间部署一台交换机，是一项会导致断网的操作。不过，真要如此行事的话，网络中断的时间最多也就一两分钟。

监控路由器的流量时，有一点请务必留意：发往路由器的数据包并不一定都会得到转发。有些数据包或许会在途中“走失”，而路由器既有可能因缓存溢出而对部分数据包“忍痛割爱”，也有可能把某些数据包从接收端口“原路送回”。

执行上述流量监控任务时，可能会用到以下两种设备。

- **TAP**：可在受监控链路上用一种叫做分路器（**Test Access Point**，**TAP**）的设备来取代图 1.3 中的交换机，这是一种简单的“三通”（三端口）设备，执行流量监控时，其所起作用跟交换机相同。与交换机相比，TAP 不但便宜而且使用方便。此外，TAP 还会把错包原样传递给 Wireshark，而 LAN 交换机则会把错包完全丢弃。交换机不但价格高昂，而且还需要花时间来配置，当然它所支持的监控功能也更多（比如，一般的 LAN 交换机都支持简单网络管理协议[SNMP]）。排除网络故障时，最好能用可网管交换机，哪怕是功能没那么丰富的可网管交换机也好。

- **HUB**: 可在受监控的链路上用一台 HUB 来取代图 1.3 中的交换机。HUB 属于半双工设备, 藉此设备, 路由器和 SP 设备之间穿行的每一个数据包都能被 Wireshark 主机“看”的一清二楚。使用 HUB 最大的坏处是, 会显著加剧流量的延迟, 从而对流量采集产生影响。如今, 监控 1Gbit/s 端口的流量可谓是家常便饭, 在这种情况下使用 HUB, 将会使速率骤降至 100Mbit/s, 这会对抓包产生严重影响。所以说, 在抓包时一般都不用 HUB。

防火墙流量监控

防火墙流量监控的手段有两种, 一种是监控防火墙内口 (如图 1.2 中的编号③所示) 的流量, 另外一种则是监控防火墙外口 (如图 1.2 中的编号④所示) 的流量。若监控防火墙内口, 则可以“观看”到内网用户发起的所有访问 Internet 的流量, 其源 IP 地址均为分配给内网用户的内部 IP 地址; 若监控防火墙外口, 则能“观看”到的所有 (经过防火墙放行的) 访问 Internet 的流量, 这些流量的源 IP 地址均为外部 IP 地址 (拜 NAT 所赐, 分配给内网用户的内部 IP 地址被转换成了外部 IP 地址); 而由内网用户发起, 但防火墙未予放行的流量, 监控防火墙外口是观察不到的^①。若有人 (通过 Internet) 发动对防火墙 (或内网) 的攻击, 要想“观察”到攻击流量, 观测点也只能是防火墙外口。

1.2.3 幕后原理

要想弄清端口镜像 (端口监控) 的运作原理, 需先理解 LAN 交换机的运作方式。LAN 交换机执行数据包转发任务时的“举动”如下所列。

1. LAN 交换机会“坚持不懈”地学习接入本机的所有设备的 MAC 地址。
2. 收到发往某 MAC 地址的数据帧时, LAN 交换机只会将其从学得此 MAC 地址的端口外发。
3. 收到广播帧时, 交换机会从除接收端口以外的所有端口外发。
4. 收到多播帧时, 若未启用 Cisco 组管理协议 (Cisco Group Management Protocol, CGMP) 或 Internet 组管理协议 (Internet Group Management Protocol, IGMP) 监听特性, LAN 交换机会从除接收端口以外的所有端口外发; 若启用了以上两种特性之一, LAN 交换机将会通过连接了相应多播接收主机的端口, 外发多播帧。
5. 收到目的 MAC 地址未知的数据帧时 (这种情况比较罕见), 交换机会从除接收端口以外的所有端口外发。

^① 译者注: 原文是 “On the internal port you will see all the internal addresses and all traffic initiated by the users working in the internal network, while on the external port you will see the external addresses that we go out with (translated by NAT from the internal addresses); you will not see requests from the internal network that were blocked by the firewall”。原文较差, 译文酌改。

综上所述，在 LAN 交换机上配置端口镜像去监控某个端口时，可“采集”到进、出该端口的所有流量。若只是将一台安装了 Wireshark 的笔记本接入 LAN 交换机，未在交换机上开启端口镜像功能，则只能抓到流入或流出该笔记本的所有单播流量，以及同一 VLAN 里的多播及广播流量。

1.2.4 拾遗补缺

使用 Wireshark 抓包时，还需提防几种特殊情况。

其中的一个特殊情况是，抓取整个 VLAN 的流量（VLAN 流量监控）。在基于 VLAN 执行抓包任务时，有几个重要事项需要铭记。第一个要注意的地方是，Wireshark 主机只能采集到与其直连的交换机承载的同一 VLAN 的流量。比方说，在一个交换式网络（LAN）内，有多台交换机都拥有隶属于 VLAN 10 的端口，要是只让 Wireshark 主机直连某台接入层交换机，那必然采集不到 VLAN 10 内其他接入层交换机上的主机访问直连核心层交换机的服务器的流量。

请看图 1.4 所示的网络，用户一般会分布在各个楼层，跟所在楼层的接入层交换机相连。各台接入层交换机会跟一台或两台（为了冗余）核心层交换机相连。Wireshark 主机要想抓全某个 VLAN 的流量，必须与承载此 VLAN 流量的交换机直接相连，才能采集到相应 VLAN 的流量。因此，要想抓全 VLAN 10 的流量，Wireshark 主机必须直连核心层交换机。

在图 1.4 中，若 Wireshark 主机直连 SW2，且在 SW2 上激活了相关端口镜像功能，开始监控 VLAN 30 的流量，则其只能抓取到进、出 SW2 P2、P4、P5 端口的流量，以及由 SW2 承载的同一 VLAN 的流量。该 Wireshark 主机绝不可能采集到 SW3 和 SW1 之间来回穿行的 VLAN 30 的流量。

基于整个 VLAN 来实施抓包任务时，可能会抓到重复的数据包，是另外一个需要注意的地方。之所以会出现这种情况，是因为启用端口镜像时，对于在不同交换机端口之间交换的同一 VLAN 的流量，Wireshark 主机会在流量接收端口的流入（input）方向及流量发送端口的流出（output）方向分别抓取一遍。

如图 1.5 所示，在交换机上已激活了端口镜像功能，对 VLAN 30 的流量实施监控。对于服务器 S4 向 S2 发送的数据包，当其（从连接 S4 的交换机端口）流入 VLAN 30 时，Wireshark 主机将抓取一次；当其从（从连接 S2 的交换机端口）流出 VLAN 30 时，Wireshark 主机将再抓取一次。这么一来，便采集到了重复的流量。

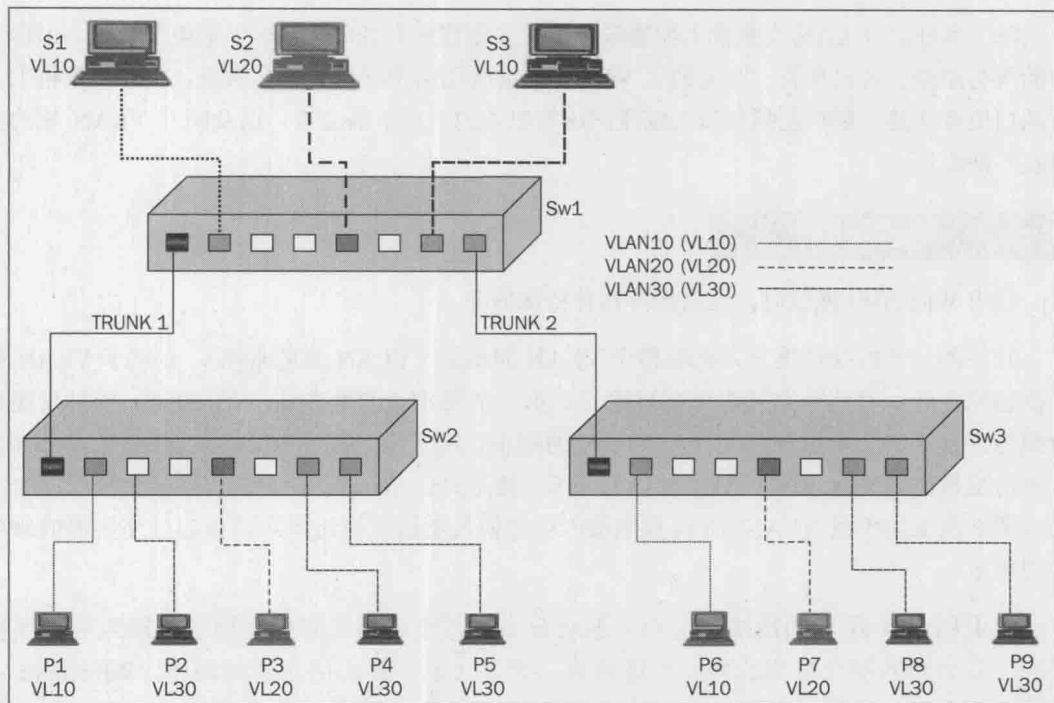


图1.4

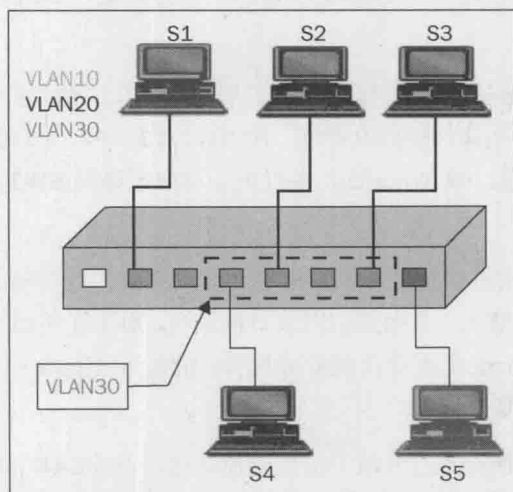


图1.5

1.2.5 进阶阅读

欲深入了解端口镜像相关信息，请参阅各网络设备厂商提供的操作手册。有些厂商也把端

口镜像称为“端口监控”或 SPAN (Switched Port Analyzer) (Cisco 公司)。

某些厂商的交换机支持远程流量监控 (能让直连本地交换机的 Wireshark 主机采集到远程交换机端口的流量) 以及高级过滤功能 (比如, 在把流量重定向给 Wireshark 主机的同时, 过滤掉具有特定 MAC 地址的主机发出的流量)。还有些高端交换机本身就具备抓取并分析数据包的功能。某些交换机还能支持虚拟端口 (例如, 聚合端口或以太网通道端口) 的流量监控。有关详情, 请阅读交换机的随机文档。

1.3 开始抓包

本节首先将介绍如何启动 Wireshark, 然后会讲解布放好 Wireshark 之后, 如何对其进行配置, 以应对不同的抓包场景。

1.3.1 准备工作

安装过 Wireshark 之后, 需点击桌面→开始→程序菜单或快速启动栏上相应的图标, 运行该数据包分析软件。

Wireshark 一旦运行, 便会弹出图 1.6 所示的窗口 (Wireshark1.10.2 运行窗口)。

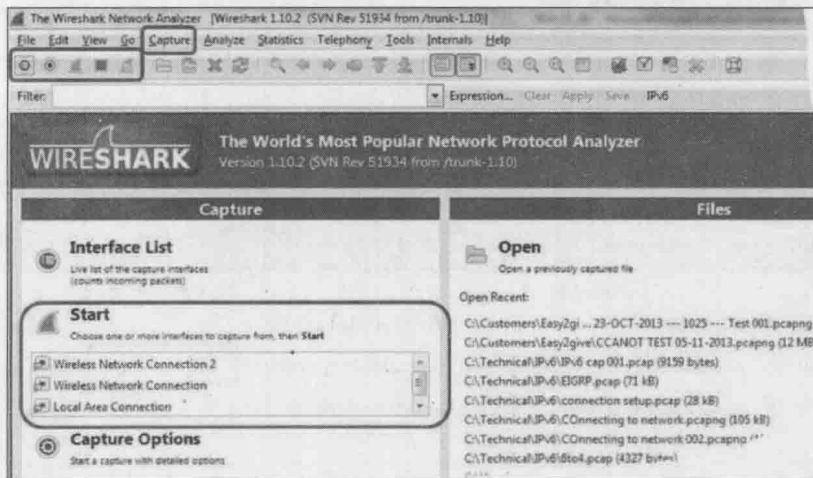


图 1.6

1.3.2 操作方法

要想让 Wireshark 软件能抓到数据包, 有以下三种途径: 点击 **Capture** 菜单下的相关菜单项; 点击快速启动工具栏里的绿色图标; 点击 Wireshark 主窗口左侧居中的 Start 区域里的相关

选项, 如图 1.6 所示。此外, 在抓包之前, 还可对 Wireshark 的某抓包选项进行配置。

如何选择实际用来抓包的网卡

若只是点击图 1.7 所示 Wireshark 快速启动工具栏里的绿色图标 (正数第三个图标), Wireshark 在抓包时, 实际使用的网卡将会是该软件默认指定的网卡 (如何更改这一默认配置, 详见 1.3.3 节)。要选择 Wireshark 抓包时实际使用的网卡, 请点击快速启动栏里左边第一个图标 (List the available capture interfaces 图标), Wireshark Capture Interfaces 窗口会立刻弹出, 如图 1.8 所示。

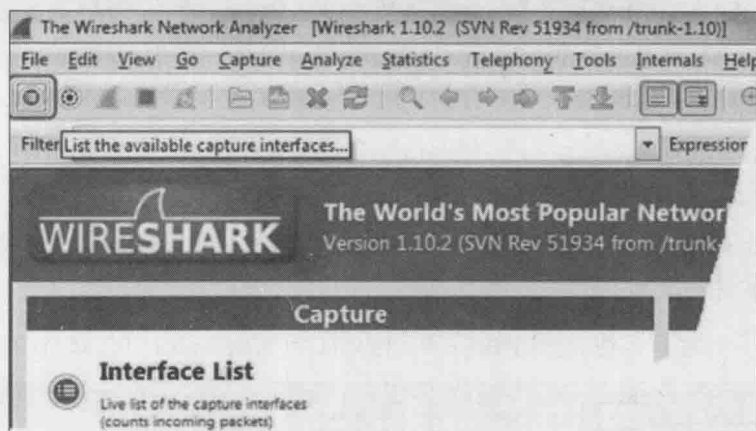


图 1.7

要想得知哪块网卡为有效网卡, 最佳途径是观察其是否能够收发流量。通过图 1.8, 可以得知 Wireshark 感知到的各块网卡正在收、发的数据包个数 (Packets 列) 及速率 (Packets/s 列)。

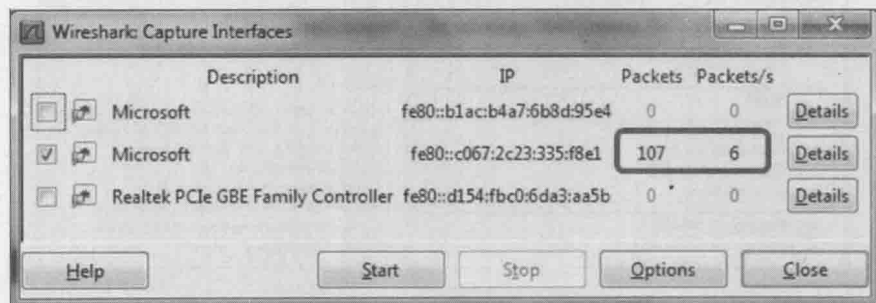


图 1.8

若 Wireshark 的版本不低于 1.10.2, 则可以选择一块以上的网卡来同时抓包。如此行事的好处是, 只要 Wireshark 主机配有多块网卡, 便可同时监控多个服务器端口、多个路由器 (或其他网络设备) 端口的流量。图 1.9 所示为这样的一个应用场景。

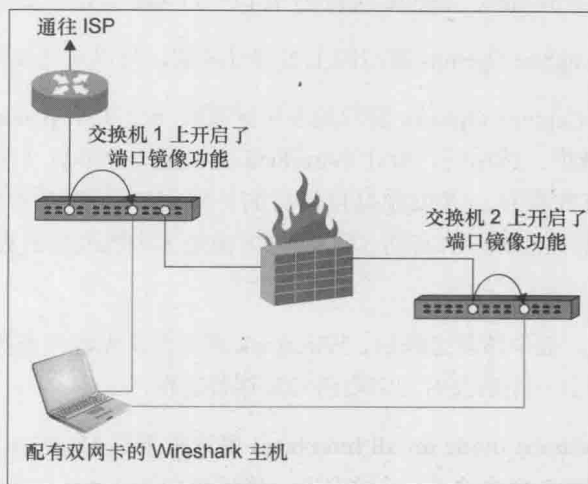


图 1.9

如何配置实际用来抓包的网卡

要想对实际用来抓包的网卡做进一步的配置，请点击 **Capture** 菜单中的 **Options** 菜单项，Wireshark Capture Options 窗口会立刻弹出，如图 1.10 所示。

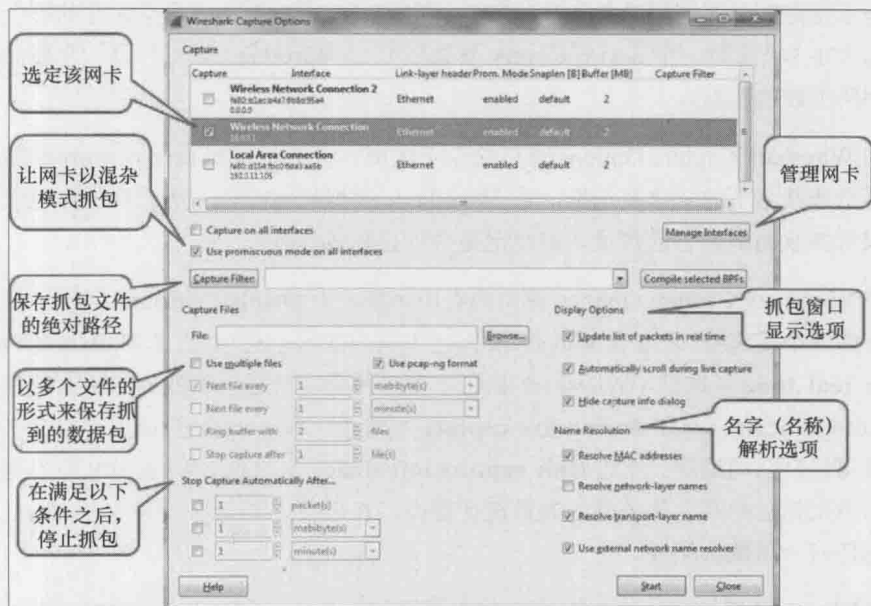


图 1.10

在图 1.10 所示的 Wireshark Capture Options 窗口中，可配置以下参数。

1. 在 Wireshark Capture Options 窗口的上半部分区域，可以点选实际用来抓包的网卡。
2. 在 Wireshark Capture Options 窗口的左中区域有一个 **Use promiscuous mode on all interfaces** 复选框。选中时，会让 Wireshark 主机抓取交换机（端口镜像功能）重定向给自己的所有数据包，哪怕数据包的目的（MAC/IP）地址不是本机地址；否则，Wireshark 主机只能抓取到目的（MAC/IP）地址为本机地址的数据包，外加广播及多播数据包。
3. 在某些情况下，选中该复选框后，Wireshark 将不会从无线网卡抓包。因此，若选用无线网卡抓包，且一无所获时，请取消勾选该复选框。
4. 在 **Use promiscuous mode on all interfaces** 复选框下有 **Capture Files** 字样，其后有个 files 输入栏，可在栏内输入一个文件名，然后再点选 **use multiple files** 复选框。这么一点，Wireshark 不但会把所抓数据保存在由其命名的文件内（其系统路径可由用户指定），而且还可根据特定的需求，以多个文件的形式存储。当以多个文件的形式存储时，在同一目录下，Wireshark 会自动在原始文件名后添加后缀“_xxxxxx_具体时间”来加以区分。若所要抓取的数据较多，Wireshark 的这一功能便非常有用。譬如，在网卡收到的流量较高，或需要长期抓取数据的情况下，就可以利用这一多文件存储功能，基于特定的时间间隔（点选第二个 **next file every** 复选框）或希望保存的每个抓包文件的大小（点选第一个 **next file every** 复选框），让 Wireshark 另行打开一个新的文件来保存所抓取的数据。
5. 在 Wireshark Capture Options 窗口的左下区域，有 **Stop Capture Automatically** 字样。可点选其名下的三个复选框，让 Wireshark 根据抓包时长、所保存的抓包文件的大小或所抓取的数据包的数量，来决定是否停止抓包任务。
6. 在 Wireshark Capture Options 窗口的右中区域，有 **Display Options** 字样。可点选其名下的三个复选框，来配置 Wireshark 抓包主窗口的显示选项。点选 **Update list of packets in real time** 复选框，Wireshark 抓包主窗口将会实时显示抓取到的所有数据包；点选 **Automatically scroll during live capture** 复选框，Wireshark 抓包主窗口会在实时显示数据包时自动滚屏；点选 **Hide capture info dialog** 复选框，Wireshark 将不再弹出与实际用来抓包的网卡相关联的流量统计窗口。一般而言，无需改变 Wireshark 软件的上述任何一项默认配置。
7. 在 Wireshark Capture Options 窗口的右下区域，有 **name resolution** 字样。可点选其名下的 4 个复选框，来调整与名字解析有关的配置。点选前三个复选框，就会让 Wireshark

在显示数据时,解析出与 MAC 地址、IP 地址以及第四层协议端口号相对应的名称(比如,MAC 地址所隶属的厂商名、与 IP 地址相对应的主机名或域名、与 TCP/UDP 端口号相对应的应用程序名等);点选最后一个复选框 **Use external network name resolver**,Wireshark 便会调用由操作系统指明的名字解析程序(比如,DNS 解析程序),来解析上述名称。

1.3.3 幕后原理

Wireshark 的抓包原理非常简单。把 Wireshark 主机上的网卡接入有线或无线网络开始抓包时,介于有线(或无线)网卡和抓包引擎之间的软件驱动程序便会参与其中。在 Windows 和 UNIX 平台上,这一软件驱动程序分别叫做 **WinPcap** 和 **Libcap** 驱动程序;对于无线网卡,行使抓包任务的软件驱动程序名为 **AirPacP** 驱动程序。

1.3.4 拾遗补缺

若(数据包的收、发)时间是一个重要因素,且还要让 Wireshark 主机从一块以上的网卡抓包,则 Wireshark 主机就必须与抓包对象(受监控主机或服务器)同步时间,可利用 **NTP**(网络时间协议)让 Wireshark 主机/抓包对象与某个中心时钟源同步时间。

当网管人员既需观察 Wireshark 抓包文件,也需检查抓包对象所生成的日志记录,以求寻得排障线索时,Wireshark 主机与抓包对象的系统时钟是否同步将会变得无比重要。比方说,Wireshark 抓包文件显示的发生 TCP 重传的时间点,与受监控服务器(生成的)日志显示的发生应用程序报错的时间点相吻合,则可以判断 TCP 重传是拜服务器(上运行的应用程序)所赐,与网络无关。

Wireshark 软件所采用的时间取自操作系统(Windows、Linux 等)的系统时钟。至于不同 OS 中 NTP 的配置方法,请参考相关操作系统配置手册。

以下所列为在 Microsoft Windows 7 操作系统内配置时间同步的方法。

1. 单击任务栏最右边的时间区域,会出现时间窗口。
2. 在时间窗口中点击“更改日期和时间设置”,会弹出“日期和时间”窗口。
3. 在“日期和时间”窗口中,点击“Internet 时间”标签,再点击“更改设置”,会弹出“Internet 时间设置”窗口。
4. 在“Internet 时间设置”窗口中,选中“与 Internet 时间服务器同步”复选框,在“服务器”后的输入栏内输入时间服务器(NTP)的 IP 地址,再点确定按钮。



注意

在 Microsoft Windows 7 及后续版本的操作系统中，默认包含了几个时间服务器（格式为域名）。可选择一个时间服务器，让网络内的所有主机都与其同步时间。

NTP 是一种网络协议，网络设备之间可藉此协议同步各自的时间。可把网络设备（路由器、交换机、防火墙）及服务器配置为 NTP 客户端，令它们与同一台 NTP 时间服务器（时钟源）对时（同步时间），时间精度要取决于那台时间服务器所处的层级（stratum）或等级（level）。NTP 时间服务器所处层级越高，其所提供的时间也就越精确。直连原子时钟并提供 NTP 对时服务的设备被称为 1 级时钟源，其精度也最高。

RFC 1059（NTPv1）是定义 NTP 的第一份标准文档，RFC 1119（NTPv2）则是第二份；目前常用的 NTPv3 和 v4 则分别定义于 RFC 1305 和 RFC 5905。

NTP 服务器 IP 地址表可从多处下载，比如 <http://support.ntp.org/bin/view/Servers/StratumOneTimeServers> 和 <http://wpollock.com/AUnix2/NTPstratum1PublicServers.htm>。

1.3.5 进阶阅读

可浏览以下站点，来了解与 PACP 驱动程序有关的信息。

- WinPcap: <http://www.winpcap.org>
- LibPcap: <http://www.tcpdump.org>

1.4 配置启动窗口

本节会介绍与 Wireshark 启动窗口有关的基本配置，同时会介绍抓包主窗口、文件格式以及可视选项的配置。

1.4.1 准备工作

启动 Wireshark 软件，首先映入眼帘的就是启动窗口。可在此窗口中调整以下各项配置参数，来满足抓包需求：

- 工具条配置；
- 抓包主窗口配置；

- 时间格式;
- 名字解析;
- 所抓数据包的配色;
- 抓包时是否自动滚屏;
- 字体大小;
- 主窗口数据包属性栏的配置;
- 配色规则。

先来熟悉一下 Wireshark 启动窗口内几个常用的工具条(栏), 如图 1.11 所示。

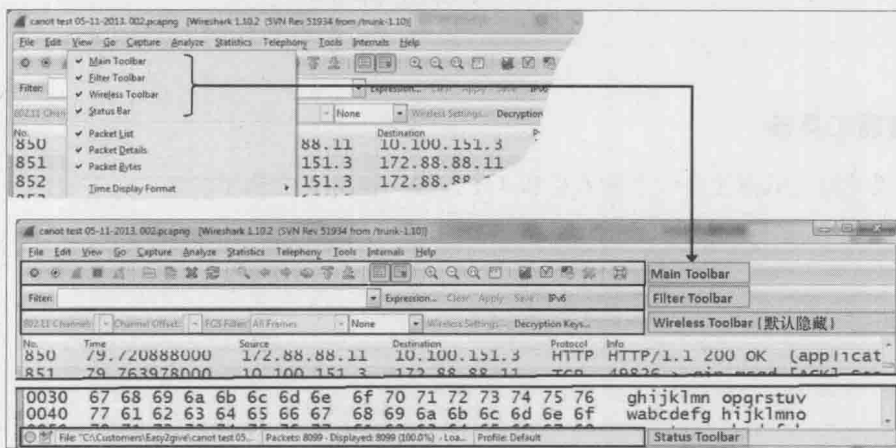


图 1.11

本节将重点介绍下列工具条的结构及用法:

- 主工具条 (Main Toolbar);
- 显示过滤器工具条 (Filter Toolbar);
- 状态栏 (Status Toolbar)。

主工具条

主工具条上各个(组)按钮的用途如图 1.12 所示。

主工具条最左边一组 5 个按钮都与抓包操作有关, 其余按钮分别涉及文件操作、数据包选择操作、字体缩放操作、配色及自动滚屏、抓包/显示过滤器的调整及应用、帮助等。

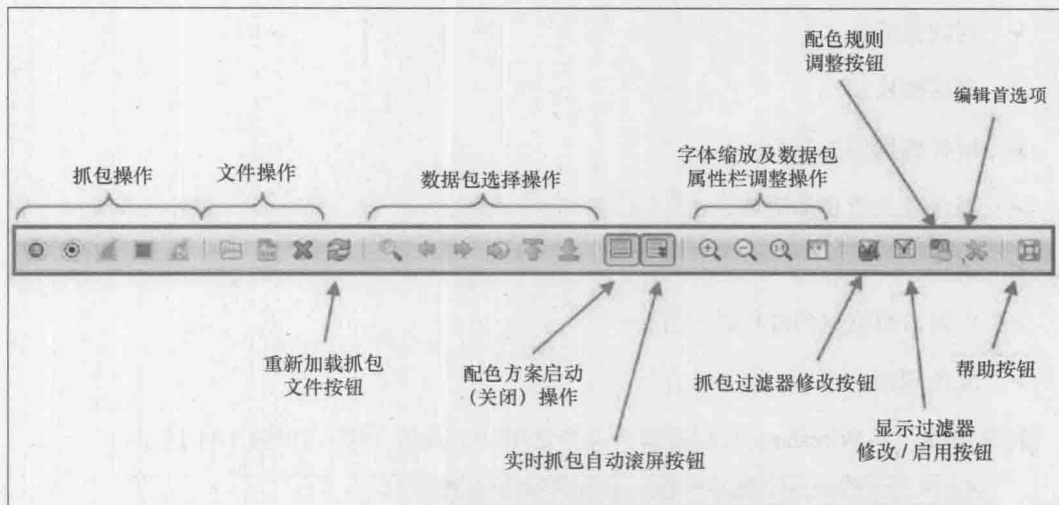


图1.12

显示过滤器工具条

显示过滤器工具条上有一个输入栏和 4 个按钮，如图 1.13 所示。

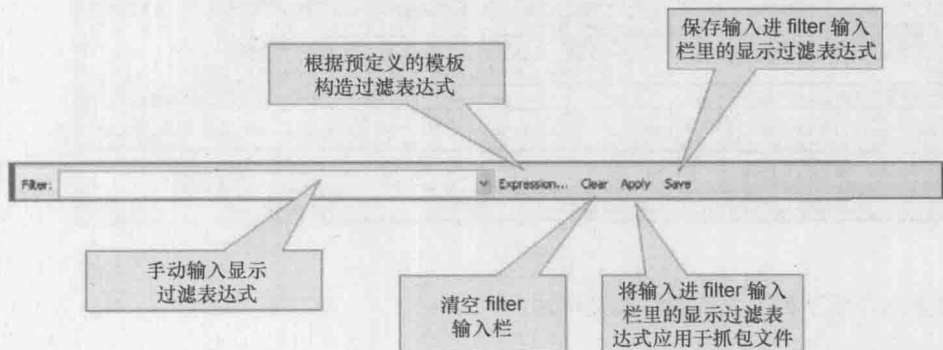


图1.13

状态栏

在 Wireshark 主窗口的最底部，有一个状态栏，分 5 个区域，如图 1.14 所示。

通过图 1.14 所示的 Wireshark 主窗口底部状态栏，可以：

- 观察到专家系统中的错误^①；

^①译者注：原文是“Errors in the expert system”，译文直译。有句土话叫“东一榔头，西一棒槌”，应该很贴切地形容了作者的写作风格。

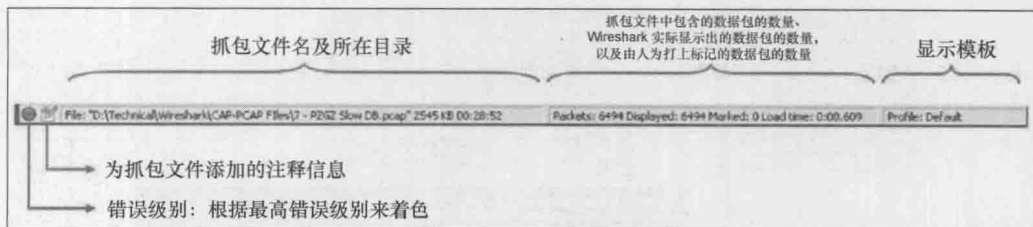


图1.14

- 选择为抓包文件添加注释信息;
- 观察到抓包文件的名称 (在抓包期间, 抓包文件名由 Wireshark 软件临时分配);
- 获知抓包文件中包含的数据包的数量、Wireshark 实际显示出的数据包的数量, 以及人为打上标记的数据包的数量。

1.4.2 配置方法

本节会按部就班地指导读者配置 Wireshark 启动窗口和抓包主窗口。

定制工具条

对于一般情况下的抓包, 根本无需调整与 Wireshark 工具条有关的任何配置。但若要抓取无线网络中的数据 (即要让 Wireshark 主机抓到无线网络里其他主机的无线网卡收发的数据), 则需要在 Wireshark 启动窗口内激活 wireless 工具栏。为此, 请点击启动窗口中的 View 菜单, 并选择 Wireless Toolbar 菜单项, 如图 1.15 所示。

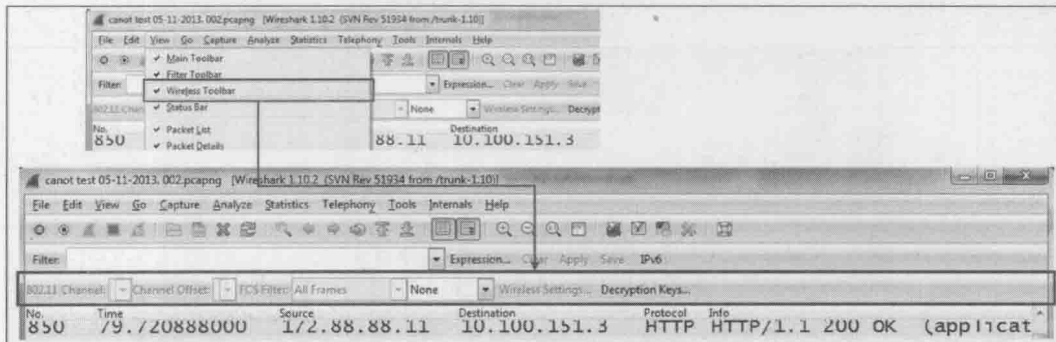


图1.15

定制抓包主窗口

可按图 1.16 所示来配置 Wireshark, 定制其抓包主窗口的界面。

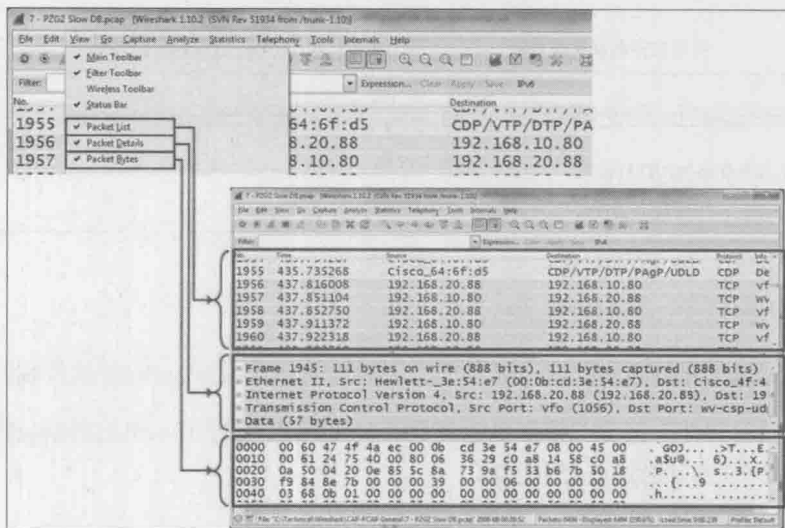
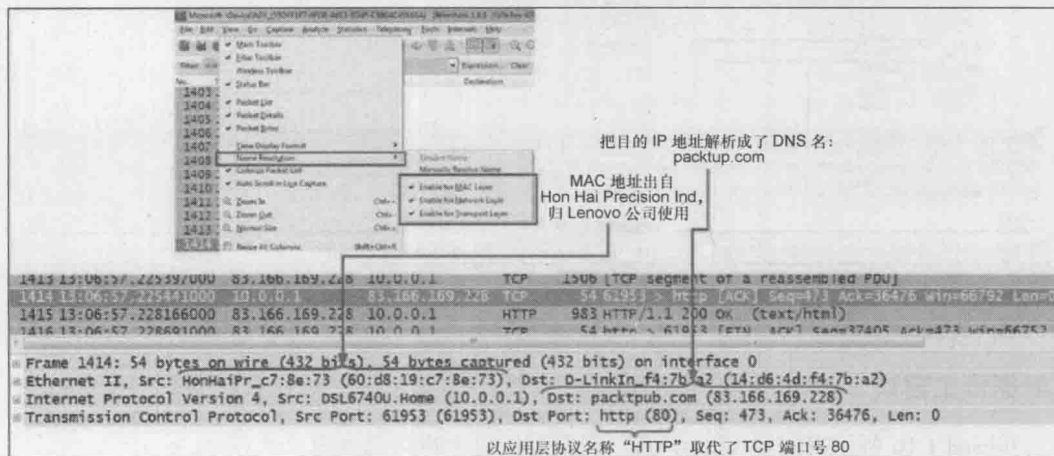


图1.16

一般而言，无需对 Wireshark 抓包主窗口的界面做任何调整。但在某些情况下，也有可能需取消勾选 View 菜单中的 Packet Bytes 菜单项，把抓包主窗口的空间都留给“数据包列表”（对应于 View 菜单中的 Packet List 菜单项）和“数据包内容”区域（对应于 View 菜单中的 Packet Details 菜单项）。

名字解析

在 Wireshark 软件里，名字解析功能一经启用，数据包中的 L2（MAC）/L3（IP）地址以及第 4 层（UDP/TCP）端口号将会分别以有实际意义的名称示人，如图 1.17 所示。



以应用层协议名称“HTTP”取代了 TCP 端口号 80

图1.17

由图 1.17 中画线的地方可知，数据包所含 MAC 地址、IP 地址以及 TCP 端口号都以名称进行相应的替换。

为数据包着色

通常，在使用 Wireshark 抓包时，应为抓取到的网络中的正常流量建立一个（视觉上的）基线模板。这样一来，便可以一边抓包，一边通过抓包主窗口显示出的数据包的色差，来发现潜在的令人生疑的以太网、IP 或 TCP 流量。

要让 Wireshark 体现出这样的色差，请在抓包主窗口的数据包列表区域，选择一个深受怀疑或需要着色的数据包，同时点右键，在右键菜单 Colorize Conversation 下点选 Ethernet、IP 或 TCP/UDP（TCP 和 UDP 只有一项可选，视数据包的第四层类型而定）子菜单项名下的各种颜色（color）菜单项。如此操作，会让该数据包所归属的（Ethernet、IP、UDP 或 TCP）会话中的所有其他数据包都以相同的颜色示人。

现举一个给抓包文件中的数据包上色的例子，如图 1.18 所示，作者将归属传输层安全（Transport Layer Security, TLS）会话的所有数据包以另外一种颜色示人。

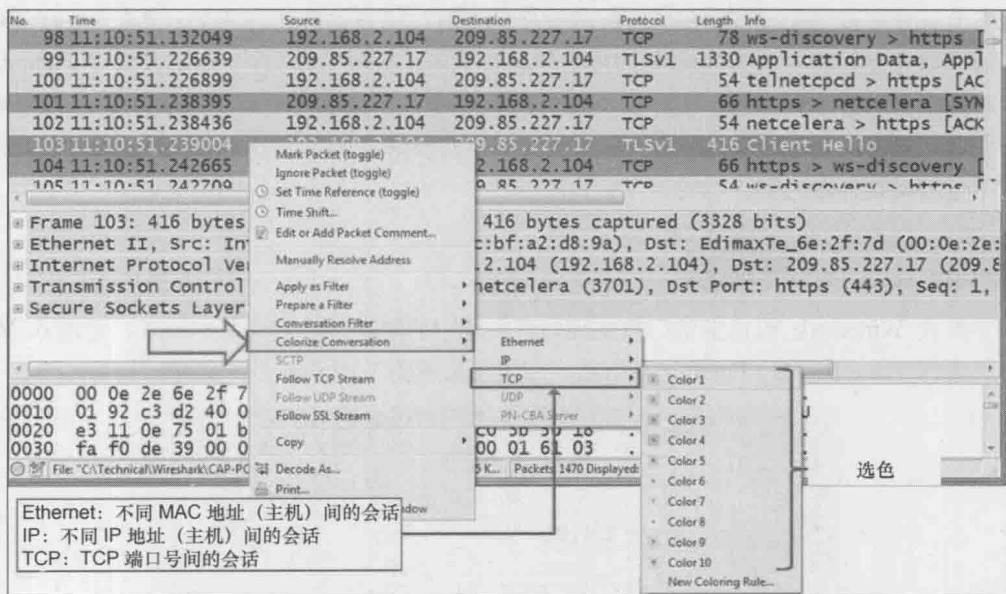


图 1.18

要取消配色规则，请按下列步骤行事。

1. 点击 **View** 菜单。
2. 选择底部的菜单项 **Reset Coloring 1-10**。

抓包时实时自动滚屏的配置

要配置 Wireshark 使其在抓包时自动滚屏，请按以下步骤行事。

1. 点 **View** 菜单。
2. 选择中间的 **Auto Scroll in Live Capture** 菜单项。

字体缩放

要缩放 Wireshark 抓包主窗口的字体，请按以下步骤行事。

1. 点 **View** 菜单。
2. 放大字体，请点中间的菜单项 **Zoom In** 或按 **Ctrl+“+”** 键。
3. 缩小字体，请点中间的菜单项 **Zoom Out** 或按 **Ctrl+“-”** 键。

1.5 配置时间参数

本节将介绍如何配置数据包的时间显示格式。对时间显示格式的调整，会在 Wireshark 抓包主窗口数据包列表区域的 **time** 列的内容里反映出来。在某些情况下，有必要让 Wireshark 以多种时间格式来显示数据包。比方说，在观察隶属同一连接的所有 TCP 数据包时，每个数据包的发送间隔时间是应该关注的重点；当所要观察的数据包抓取自多个来源时，则最应关注每个数据包的确切抓取时间。

1.5.1 准备工作

要配置 Wireshark 抓包主窗口数据包列表区域中的数据包的时间显示格式，请进入 **View** 菜单，选择 **Time Display Format** 菜单项，右边会出现图 1.19 所示的子菜单。

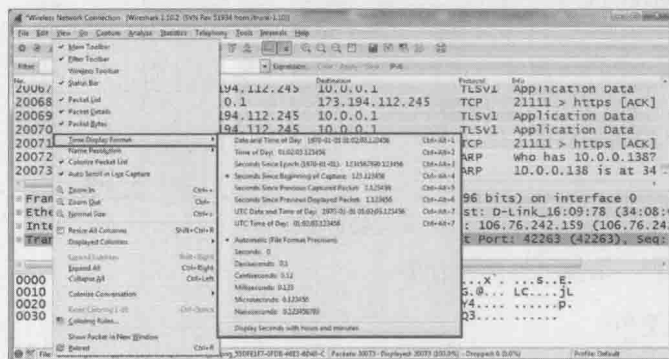


图 1.19

1.5.2 配置方法

图 1.19 所示 Time Display Format 菜单项的上半部分子菜单包含以下子菜单项。

- **Date and Time of Day** 和 **Time of Day**: 当通过 Wireshark 抓包来帮助排除网络故障, 且故障发生的时间也是定位故障的重要依据时(比如, 已获悉了故障发生的精确时间, 且还想知道相同时间网络中发生的其他事件时), 就应该根据具体情况, 选择这两个子菜单项之一。
- **Seconds Since Epoch** (自 **Epoch** 以来的秒数): Epoch 是指通用协调时间(格林威治标准时间的前称)的 1970 年 1 月 1 日早晨 0 点。这也是 UNIX 系统问世的大致时间。
- **Seconds Since Beginning of Capture** (自开始抓包以来的秒数): 此乃 Wireshark 默认选项。
- **Seconds Since Previous Captured Packet** (自抓到上一个数据包以来的秒数): 这也是一个常用选项, 此菜单项一经点选, 数据包列表区域的 time 列将显示每个数据包的抓取时间差。当监控时间敏感型数据包(比如, TCP 流量、实时视频流量、VoIP 语音流量)时, 就应该点选该子菜单项, 因为此类数据包的发送时间间隔对用户体验有至关重要的影响。
- **Seconds Since Previous Displayed Packet**: 在应用了显示过滤器, 让 Wireshark 只显示抓包文件中部分数据的情况下(比如, 只显示出隶属于某条 TCP 流的所有数据包), 通常都应该点选该子菜单项。此时, 网管人员更关心的应该是隶属于这条 TCP 数据流的各个数据包之间的抓取时间差。
- **UTC Date and Time of Day** 和 **UTC Time of Day**: 提供 UTC 时间。

可使用快捷键 Ctrl + Alt + 任意数字键来调整上述时间格式选项。

Time Display Format 菜单项的下半部分子菜单提供的是时间精度选项, 若无时间精度方面的需求, 则应保持默认设置。

1.5.3 幕后原理

为抓到的数据留时间“烙印”时, Wireshark 依据的是操作系统的时间。在默认情况下, 生效的是 **Seconds Since Beginning of Capture** 子菜单项功能。

1.6 定义配色规则

Wireshark 会根据事先定义的配色规则，用不同的颜色来“分门别类”地显示抓包文件中的数据。合理的定义配色规则，让不同协议的数据包以不同的颜色示人（或让不同状态下的同一种协议的数据包呈现出多种颜色），能在排除网络故障时帮上大忙。

Wireshark 支持基于各种过滤条件，来配置新的配色规则。这样一来，就能够针对不同的场景，定制不同的配色方案，同时还能以不同的模板来保存。也就是说，网管人员可在解决 TCP 故障时，启用配色规则 A；解决 SIP 和 IP 语音故障时，启用配色规则 B。



注意

可以通过定义模板（profile）的方式，来保存针对 Wireshark 软件自身的配置（比如，事先配置的配色规则和显示过滤器等）。要如此行事，请点击 **EDIT** 菜单下的 **Configuration Profiles** 菜单项。

1.6.1 准备工作

要定义配色规则，请按以下步骤行事。

1. 选择 **View** 菜单。
2. 点击底部的 **Coloring Rules** 菜单项，**Coloring Rules** 窗口会立刻弹出，如图 1.20 所示。

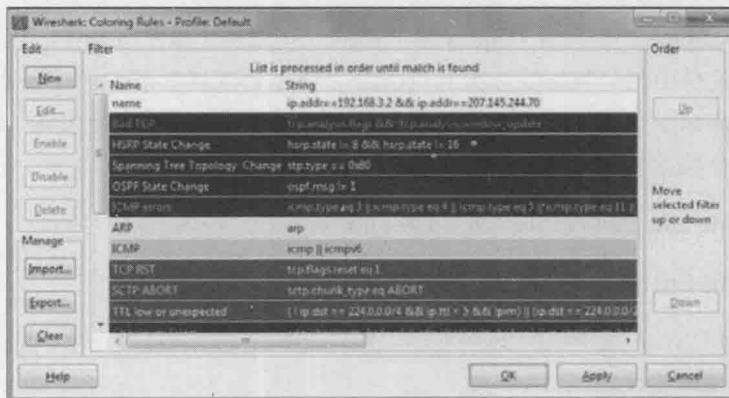


图 1.20

1.6.2 配置方法

现在，来看一下如何定义一条新的配色规则。

点击 **Coloring Rules** 窗口中的 **New** 按钮，**Edit Color Filter** 窗口会立刻弹出，如图 1.21 所示。

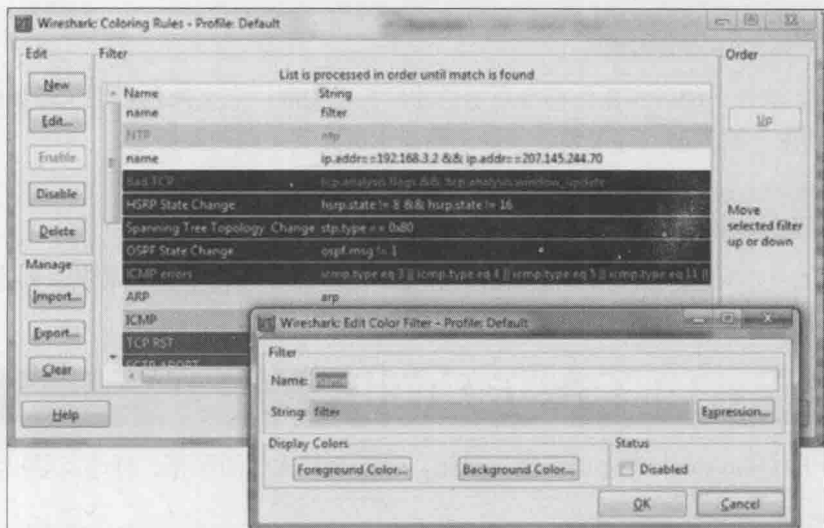


图1.21

新的配色规则就在此窗口内配置，请按以下步骤行事。

1. 在 **Name** 输入栏内填入这条规则的名称。譬如，要想专为 NTP 协议数据包定制配色规则，那就在该输入栏内填入 NTP。
2. 在 **String** 字段内填入显示过滤表达式，指明本配色规则对哪些数据包生效。也可以点击 **Expression** 按钮，根据 Wireshark 预制的显示过滤参数来构造显示过滤表达式。欲知更多与显示过滤器和显示过滤表达式有关的内容，请阅读第 3 章。
3. 点击 **Foreground Color** 按钮，为本配色规则选择一款前景色。此款颜色将成为受本配色规则约束的数据包在抓包主窗口的数据包列表区域里的前景色。
4. 点击 **Background Color** 按钮，为本配色规则选择一款背景色。此款颜色将成为受本配色规则约束的数据包在抓包主窗口的数据包列表区域里的背景色。
5. 要是想修改现有的配色规则，请点击 **Edit** 按钮。还可以点击 **Import** 按钮，导入现成的配色方案，或点击 **Export** 按钮导出当前的配色方案。



注意

Coloring Rules 窗口中配色规则的排放次序是有讲究的。请务必确保配色规则的排放次序与配色方案的执行次序相匹配。比方说，作用于应用层协议数据包的配色规则应置于作用于 TCP/UDP 数据包的配色规则之前，只有如此，方能避免 Wireshark 为应用层协议数据包干扰 TCP/UDP 数据包的颜色。

1.6.3 幕后原理

Wireshark 软件中的许多操作都与显示过滤器紧密关联，定义配色规则也是如此，因为受配色规则“约束”的数据包都是经过预定义的显示过滤器过滤的数据包。

1.6.4 拾遗补缺

可从 <http://wiki.wireshark.org/ColoringRules> 下载到很多经典的 Wireshark 数据包配色方案，在 Internet 上也能搜到许多其他的配色方案示例。

1.7 数据文件的保存、打印及导出

本节将讨论 Wireshark 软件中的文件操作，包括数据文件的保存、打印及导出等。

1.7.1 准备工作

运行 Wireshark 软件，点击主工具条上的抓包按钮，开始抓包（或打开一个已经保存的抓包文件）。

1.7.2 操作方法

在 Wireshark 主抓包窗口下，既可把抓来的所有数据都保存进一个文件，也能以不同的格式或文件类型导出自己所需要的数据。

现在，来讲解一下如何执行这些操作。

数据保存操作

要把抓包文件中的所有数据包都保存进一个文件（或将现有抓包文件另存为一新文件），请按以下步骤行事。

1. 点击 **File** 菜单里的 **Save** 菜单项（或按 Ctrl+S 键），在弹出窗口的“文件名”输入栏内输入有待保存的文件名。
2. 点击 **File** 菜单里的 **Save as** 菜单项（或按 Shift+Ctrl+S 键），在弹出窗口的“文件名”输入栏内输入有待保存的抓包文件的新名称。

若要保存抓包文件（或已抓数据包）中的部分数据（比如，经过显示过滤器过滤的数据），请按以下步骤行事。

1. 点击 **File** 菜单里的 **Export Specified Packets** 菜单项，**Export Specified Packets** 窗口会

立刻弹出，如图 1.22 所示。

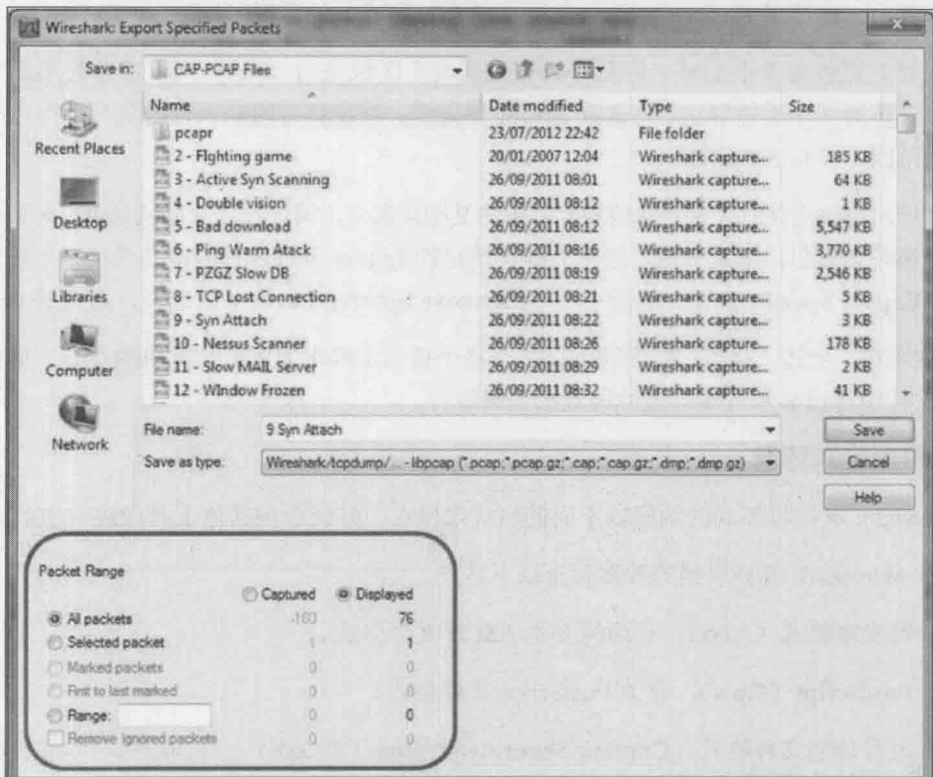


图1.22

2. 可在 Export Specified Packets 窗口的左下角区域，点击相应的单选框，来选择文件的导出方式。
3. 要把抓包文件中的所有数据包或所有已抓数据包作为一个文件导出，请同时选择 **All packets** 和 **Captured** 单选框，再点 **Save** 按钮。
4. 要把抓包文件（或已抓数据包）中经过显示过滤器过滤的数据包作为一个文件导出，请同时选择 **All packets** 和 **Displayed** 单选框，再点 **Save** 按钮。
5. 要把已选中的数据包（即在数据包列表区域中用鼠标点选的数据包）作为一个文件导出，请选择 **Selected packet** 单选框，再点 **Save** 按钮。
6. 要把所有带标记的数据包（给数据包打标的方法是：先在“数据包列表”区域选中一个数据包，点击右键，在右键菜单中选择 **Mark packet toggle** 菜单项）作为一个文件导出，请选择 **Marked packet** 单选框，再点 **Save** 按钮。

7. 要把“数据包列表”区域中位列 2 个带标记的数据包之间的所有数据包作为一个文件导出, 请选择 **First to last marked** 单选框, 再点 **Save** 按钮。
8. 要把抓包文件中编号 (详见“数据包列表”区域里的“**No.**”列) 连续的那部分数据包作为一个文件导出, 请选择 **Range** 单选框, 并在其后的输入栏内填写数据包的编号范围, 再点 **Save** 按钮。
9. 导出抓包文件时, 要是希望放弃其中的某些数据包, 请先在“数据包列表”区域里选中那些数据包, 单击右键, 选择右键菜单里的 **Ignore packet toggle** 菜单项; 然后, 选择 **Export Specified Packets** 窗口中的 **Remove Ignored Packets** 复选框, 再点 **Save** 按钮。

再次重申, 上述“存盘”操作既可以基于整个抓包文件中的所有数据包来进行, 也可以基于抓包文件中经过显示过滤器过滤的数据包来进行。

保存数据的格式选取

Wireshark 支持将抓到的数据以不同的格式来保存, 好让各种其他工具做进一步的分析。

可让 Wireshark 将抓取到的数据存为以下格式。

- **纯文本格式 (*.txt)**: 存为纯文本 ASCII 文件格式。
- **PostScript (*.pst)**: 存为 PostScript 文件格式。
- **逗号分割文件格式 (Comma Separated Values) (*.csv)**: 存为逗号分割文件格式。这种格式的文件可为电子表格程序 (比如, Microsoft Excel) 所用。
- **C 语言数组格式 (*.c)**: 把数据包的内容以 C 语言数组的格式保存, 可以很方便地将这种文件格式导入 C 程序。
- **PSML-XML 格式 (*.psml)**: 存为 PSML 文件格式。PSML 是一种基于 XML 的文件格式, 只能保存数据包的汇总信息。欲了解与此文件格式有关的详细信息, 请见: http://www.nbee.org/doku.php?id=netpdl:psml_specification。
- **PDML-XML 格式 (*.pdml)**: 存为 PDML 文件格式。PSML 也是一种基于 XML 的文件格式, 但能保存数据包的详细信息。欲了解与此文件格式有关的详细信息, 请见: http://www.nbee.org/doku.php?id=netpdl:pdml_specification。

要以不同的文件格式来保存 Wireshark 抓取到的数据, 请点击 **File** 菜单的 **Export Packet Dissections** 菜单项下相应的子菜单项。一旦点击, **Export File** 窗口会立刻弹出, 如图 1.23 所示。

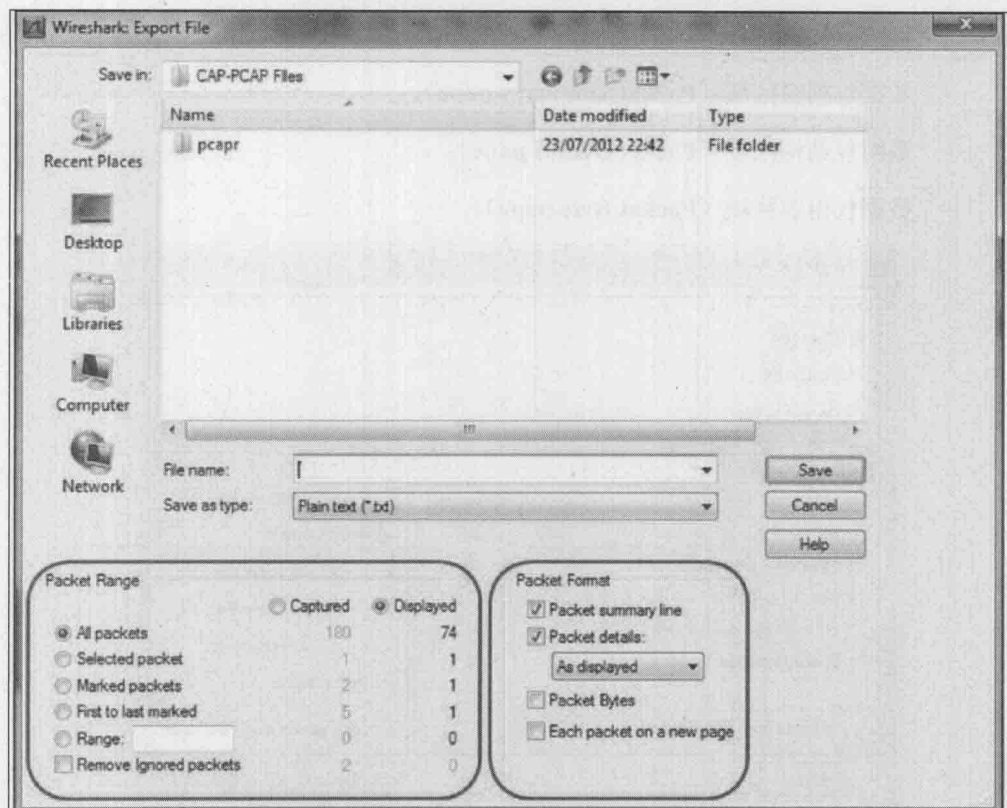


图1.23

在图 1.23 所示的 Export File 窗口的下半部分，用方框圈定了 2 个区域。左侧区域中的单选或复选框用来完成待存数据的筛选操作，这在上一节已做过深入探讨。右侧区域中的复选框和下拉菜单则用来确定应以何种“精细程度”，来保存抓取到的数据包（“精细程度”分为三种：数据包列表、数据包的详细结构以及数据包的内容，这分别对应于在抓包主窗口中出现的那三个区域里的内容）。

如何打印数据

要想打印数据，请点击 **File** 菜单里的 **Print** 菜单项，**Print** 窗口会立刻弹出，如图 1.24 所示。

可在 **Print** 窗口中做如下选择。

- 在窗口的上半部分区域，可选择有待打印的文件格式。
- 在窗口的左下区域，可选择有待打印的数据包（操作方法类似于文件保存）。
- 在窗口的右下区域，可选择有待打印的数据包的具体内容，可通过其中的复选框来选

择打印抓包主窗口中以下区域的内容：

- 数据包列表区域（**Packet Summary pane**）；
- 数据包结构区域（**Packet Details pane**）；
- 数据包内容区域（**Packet Byte pane**）。

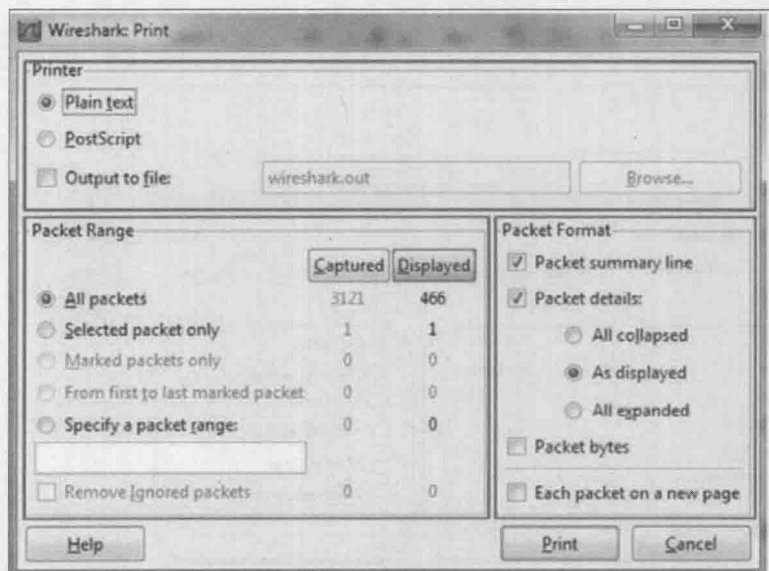


图1.24

1.7.3 幕后原理

Wireshark 支持以文本格式或 postscript 格式来打印数据（以后一种格式打印时，打印机应为 postscript 感知的打印机），同时支持将数据打印至一个文件。选妥了 **Print** 窗口中的各个选项，点击 **Print** 按钮之后，会弹出操作系统自带的常规“打印”窗口，可在其中选择具体的打印机来打印。

1.8 通过 Edit 菜单中的 Preferences 菜单项，来配置 Wireshark 主界面

借助于 **Edit** 菜单中的 **Preferences** 菜单项，能控制 Wireshark 软件的主界面及软件自身的诸多参数，包括数据包的呈现方式、抓包文件的默认存盘位置，以及用来抓取数据包的网卡等。

本章将介绍 Wireshark 主界面及软件自身的常用参数的配置，熟知这些参数的配置，可帮助我们更好地应对不同的抓包场景。

1.8.1 配置准备

要配置 Wireshark 的用户界面（主界面），请点击 **Edit** 菜单中的 **Preferences** 菜单项，**Preferences** 窗口会立刻弹出，如图 1.25 所示。

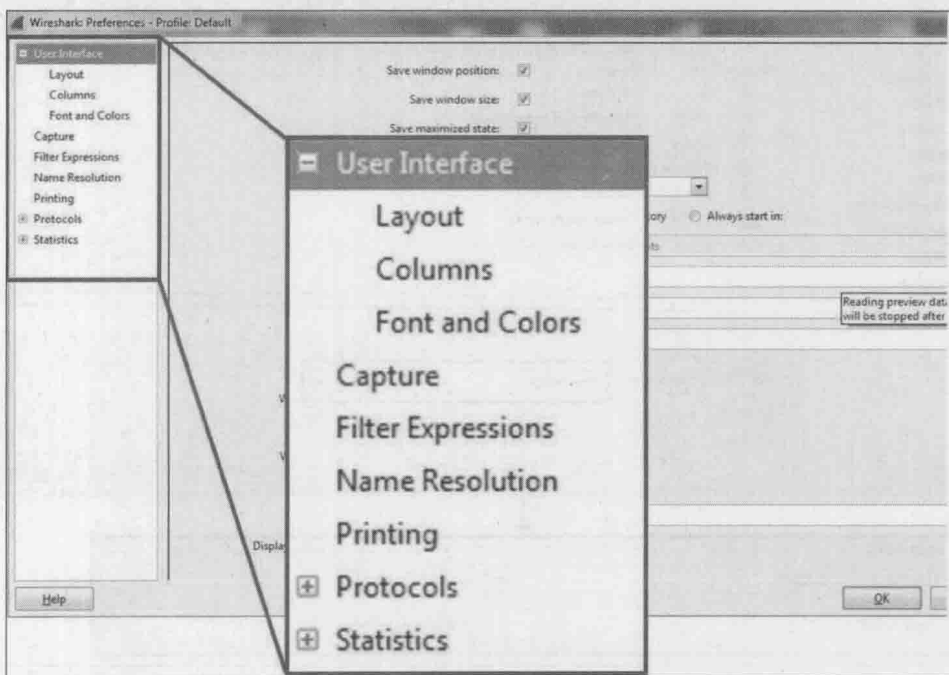


图1.25

在本章接下来的内容中，会介绍 **Preferences** 窗口中以下参数的配置：

- **Columns**（数据包属性列）；
- **Capture**（抓包方式）；
- **Name Resolution**（名字解析）。

1.8.2 配置方法

本节会介绍如何调整事关 Wireshark 软件自身的配置参数。

调整及添加数据包属性列

在默认情况下，显现在抓包主窗口的数据包列表区域里的数据包属性列有：No.（编号）、Time（抓取时间）、Source（源地址）、Destination（目的地址）、Protocol（协议类型）、Length（长度）以及 Info（信息），可通过图 1.26 所示的 Preferences 窗口来调整出现在数据包列表区域里的数据包属性列。

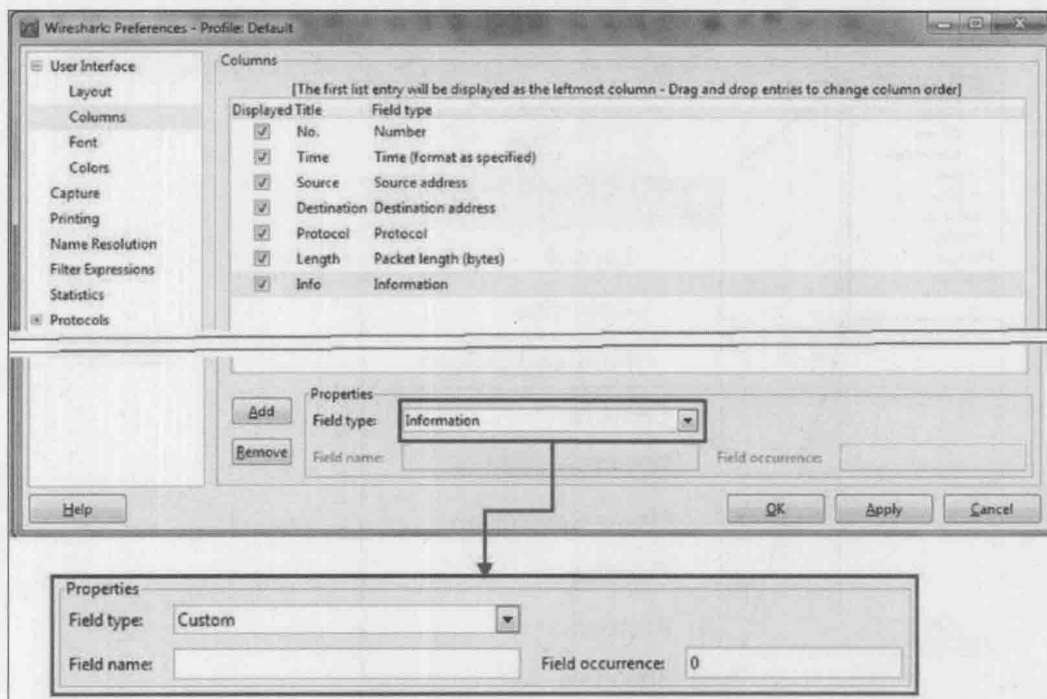


图1.26

要给数据包列表区域添加一新列，可通过以下两个途径。

1. 从 **Preferences** 窗口下部的 **Field type** 下拉菜单栏里，选择预先定义好的属性列。这些预先定义好的属性列包括 time delta、IP DSCP value、src port 和 dest port 等。
2. 点击 **Add** 按钮，从 **Preferences** 窗口下部的 **Field type** 下拉菜单栏里，选择 **Custom** 菜单项，定制数据包属性列。此时，可在图 1.26 所示的 **Field name** 输入栏内输入可在显示过滤器中露面的任一参数，然后点击 **OK** 按钮，将这一新列添加至抓包主窗口。之后，在抓包主窗口选中该列，点击右键，选择右键菜单中的 **Edit Column Details** 菜单项，便可以为新添加的这一新列改名了。

下面举几个用 **Custom** 菜单项定制数据包属性列的例子。

1. 要想在抓包主窗口中新增一列，以便观看 TCP 数据包的 TCP 窗口大小，需在 **Field name** 输入栏内输入显示过滤器参数 `tcp.window_size`。
2. 要想在抓包主窗口中新增一列，以便观看每个 IP 数据包包头中的 TTL 字段值，需在 **Field name** 输入栏内输入显示过滤器参数 `ip.ttl`。
3. 要想在抓包主窗口中新增一列，以便观看每个 RTP 数据包中 marker 位置 1 的实例，需在 **Field name** 输入栏内输入显示过滤器参数 `rtp.marker`。
4. 在分析网络故障时，酌情使用 **Custom** 菜单项定制数据包属性列，可加快定位故障的原因，与此有关的内容本书后文再叙。

调整跟执行抓包任务有关的配置

执行抓包任务之前，可调整涉及抓包的配置。为此，请在 **Preferences** 窗口中点击 **Capture** 选项，如图 1.27 所示。

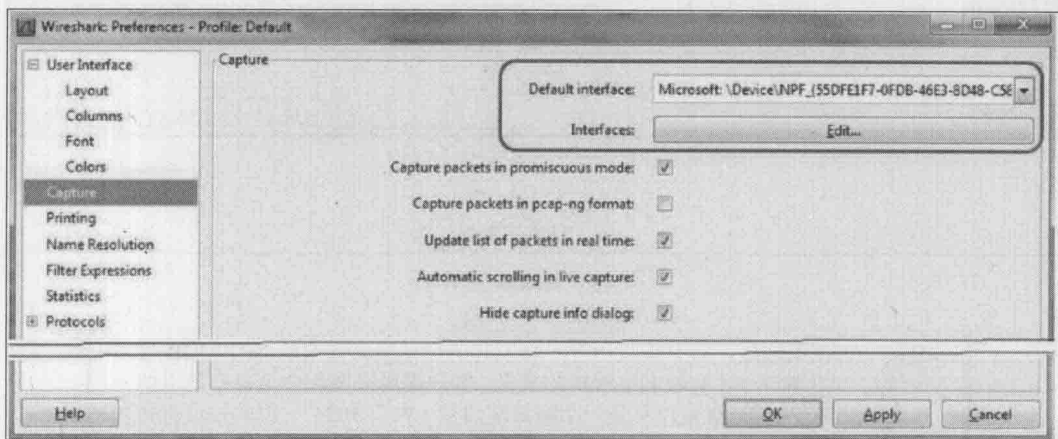


图1.27

要想更改默认用来抓取数据包的网卡，请点击 **Edit** 按钮，在弹出的 **Interface Option** 窗口中选择相应的网卡，再点 **OK** 按钮即可（重启 Wireshark 软件之后才能生效）。当然，这里更改的只是默认配置，可在每次重新开始抓包之前，更换用来接收数据包的网卡。

调整名字解析

Wireshark 支持以下三个层级的名字解析（见图 1.28）。

- 在第二层（L2）：Wireshark 可把 MAC 地址的前半部分解析并显示为网卡芯片制造商

的名称或 ID。比方说,可把一个 MAC 地址的前三个字节 **14:da:e9** 解析并显示为 **AsusTeckC** (ASUSTeK Computer Inc)。

- 在第三层 (L3): Wireshark 可把 IP 地址解析并显示为 DNS 名称,比如可把 **157.166.226.46** 这一 IP 地址,解析并显示为 **www.edition.cnn.com**。
- 在第四层 (L4): Wireshark 可把 TCP/UDP 端口号解析并显示为应用程序(服务)名称,比如可把 TCP 80 端口解析并显示为 HTTP,把 UDP 53 端口解析并显示为 DNS。

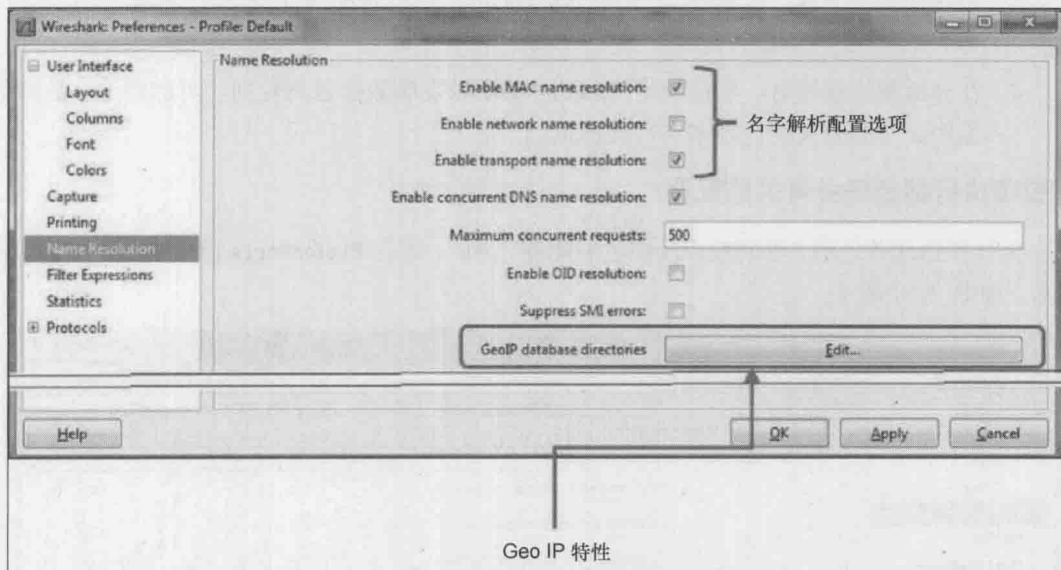


图1.28



注意

在第四层,只有接收由客户端发起的会话建立请求的 TCP/UDP 目的端口号才有做名字解析的意义。客户端用来发起会话建立请求的 TCP/UDP 源端口号是一个随机端口号(大于 1024),因此并没有转换成应用程序(服务)名的必要。

在默认情况下, Wireshark 只会针对第二层 MAC 地址和第四层 TCP/UDP 端口号做名字解析。开启 IP 地址的名字解析功能之前要三思而后行,因为这会让 Wireshark 委托 OS 发出大量 DNS 查询消息,从而拖慢 Wireshark 的运行速度。

1.8.3 幕后原理

原理非常简单,只是调整 Wireshark 某些菜单项的配置而已。要修改 Wireshark 主界面,除了本节介绍的配置选项之外,还有其他配置选项可供选择。更多详情请参考

www.wireshark.org 上的 Wireshark 配置手册。

1.9 配置 Preferences 窗口中的 Protocol 选项

借助于 **Preferences** 窗口中的 **Protocol** 选项，可调整 Wireshark 对相关协议流量的抓取和呈现方式。本节会介绍如何借助 **Preferences** 窗口中的 **Protocol** 配置选项，来调整 Wireshark 对常见协议流量的抓取和呈现方式。

1.9.1 配置准备

1. 点击 **Edit** 菜单中的 **Preferences** 菜单项，Preferences 窗口会立刻弹出，如图 1.29 所示。

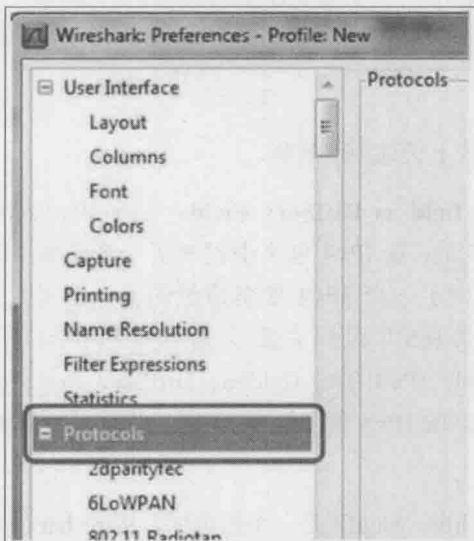


图1.29

2. 在 Preferences 窗口的左边找到 Protocols 配置选项，点其左边的“+”号，会出现一份“协议”列表。在这份列表里包含了诸多常用或不常用协议。此处只讨论某些常用协议的配置，对相关协议的介绍详见 7~14 章。

1.9.2 配置方法

本节将介绍以下“基本”协议的配置方法（“基本”意指常用，并不是指简单）：

- IPv4/IPv6;
- TCP/UDP。

配置 Protocol 选项里的 IPv4 和 IPv6 协议

在 Preferences 窗口的 Protocols 选项里选中 IPv4 或 IPv6 协议时，窗口的右边会出现若干子选项，如图 1.30 所示。

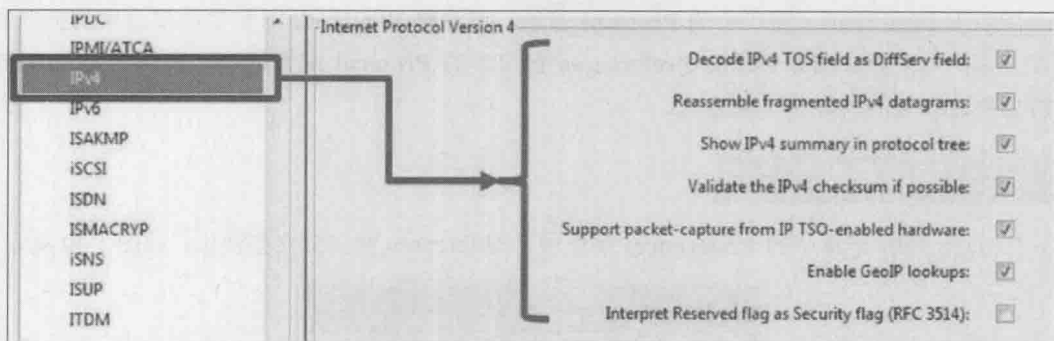


图1.30

下面是对 IPv4 协议下若干子选项的解释。

- **Decode IPv4 ToS field as DiffServ Field:** 制定 IPv4 协议标准之初，为能在 IPv4 网络中保证服务质量，在 IPv4 包头中设立了一个叫做“服务类型（ToS）”的字段。后来，IETF 又制定了一套 IPv4 服务质量的新标准（区分服务，DiffServ），打的也是 IPv4 包头中原 ToS 字段的主意，只是对其中各个位的置位方式有了新的定义。若未勾选与 Decode IPv4 ToS field as DiffServ Field 子选项相关联的复选框，Wireshark 便会按老的 IPv4 服务质量标准，来解析所抓 IPv4 数据包包头中的 ToS 字段。
- **Enable GeoIP lookups:** GeoIP 是一个数据库，Wireshark 可根据该数据库里的内容，来呈现（其所抓数据包 IP 包头中源和目的）IP 地址所归属的地理位置。若勾选与 Enable GeoIP lookups 子选项相关联的复选框，Wireshark 便会针对所抓 IPv4 和 IPv6 数据包的 IP 地址来呈现其所归属的地理位置。该子选项功能涉及名字解析，一旦开启，便会拖慢 Wireshark 实时运行速率。

配置 Protocol 选项里的 UDP 和 TCP 协议

UDP 协议非常简单，因此 Protocols 选项里 UDP 协议名下可配置的子选项并不多；而 TCP 协议则很是复杂，Protocols 选项里 TCP 协议名下可配置的参数（子选项）也要多于 UDP 协议（见图 1.31）。

更改 TCP 协议名下的各个参数，其实也就是调整 Wireshark 对 TCP 数据包的解析方式。

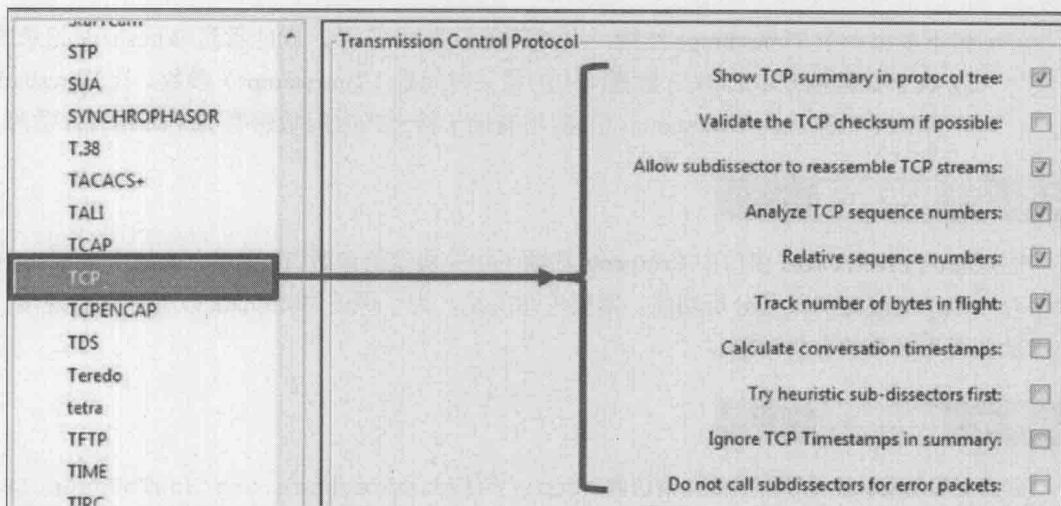


图1.31

- **Validate the TCP checksum if possible:** 有时，Wireshark 会抓到超多校验和错误（checksum errors）的数据包，这要归因于在抓包主机的网卡上开启的 TCP Checksum offloading（TCP 校验和下放）功能。该功能一开，便会导致 Wireshark 将抓到的本机生成的数据包显示为“checksum errors”（具体原因后文再表）。因此，若 Wireshark 抓到了超多校验和错误的数据包，则有必要先取消勾选与 Validate the TCP checksum if possible 子选项关联的复选框，再去验证是否真的存在校验和问题。
- **Analyze TCP Sequence numbers:** 要让 Wireshark 对 TCP 数据包做详尽分析，就必须勾选与 Analyze TCP Sequence numbers 参数相关联的复选框，因为“TCP Sequence numbers（TCP 序列号）”是 TCP 最重要的特性之一。
- **Relative Sequence Numbers:** 主机在建立 TCP 连接时，会启用一个随机的序列号，并将其值存入相互交换的第一个报文段的 TCP 头部的序列号字段^①。只要勾选了与 Relative Sequence Numbers 参数相关联的复选框，Wireshark 就会把（一股 TCP 数据流中的）第一个 TCP 报文段的（TCP 头部的）序列号字段值显示为 0，后续 TCP 报文段的序列号字段值将依次递增，从而隐藏了真实的序列号字段值。在大多数情况下，都应该让 Wireshark 显示 TCP 报文段的相对序列号（relative number），以方便网管人员查看。
- **Calculate conversations timestamps:** 与该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，只要是 TCP 数据包，就会在 transmission control protocol

① 译者注：原文是“when TCP opens a connection, it starts from a random sequence number”。

树下多出一个 Timestamps 结构，点击其前面的“+”号，就能看到 Wireshark 记录的该 TCP 数据包在本股 TCP 数据流中的相关时间戳（Timestamps）参数。让 Wireshark 显示 TCP 数据包的 Timestamps 信息，将有助于排查当时时间敏感型 TCP 应用程序故障。

1.9.3 幕后原理

通过修改 **Preferences** 窗口中 **Protocols** 选项下相关协议的参数，便能开启或禁用 Wireshark 软件对相应协议流量的某些分析功能。需要注意的是，为了保证 Wireshark 软件的运行速度，应尽量禁用不必要的分析功能。

1.9.4 进阶阅读

欲知更多与 GeoIP 有关的信息，请访问 <http://wiki.wireshark.org/HowToUseGeoIP>。

第2章

抓包过滤器的用法

本章涵盖以下内容：

- ▶ 抓包过滤器简介；
- ▶ 配置 Ethernet 过滤器；
- ▶ 配置基于主机或网络的过滤器；
- ▶ 配置 TCP/UDP 及端口过滤器；
- ▶ 配置复合型过滤器；
- ▶ 配置字节偏移和净载匹配型过滤器。

2.1 简介

第 1 章介绍了如何安装 Wireshark 软件，如何配置该软件以行使其基本功能，以及该软件在网络中的部署（或安装）位置。本章及下一章将讨论 Wireshark 抓包过滤器和显示过滤器的用法。

抓包过滤器和显示过滤器的重要区别如下所列。

- ▶ 抓包过滤器配置于抓包之前，一经应用，Wireshark 将只抓取经过抓包过滤器过滤的数据（包或数据帧），其余数据一概不抓。本章将介绍这一过滤器。
- ▶ 显示过滤器配置于抓包之后，此时，Wireshark 已抓得所有数据。网管人员会利用显示过滤器，让 Wireshark 只显示自己心仪的数据。这种过滤器将在下一章介绍。



注意

抓包过滤器的配置语法派生自 libpcap/WinPcap 库中 tcpdump 的语法，而显示过滤器的配置语法则在若干年后定义。因此，两种过滤器的配置语法并不相同。

在某些情况下，可能只想让 Wireshark 抓取某块网卡所接收到的部分数据，举例如下。

- 只想让 Wireshark 从某条数据量极大的受监控链路上，抓取必要的数据包时。
- 只想让 Wireshark 从某个受监控的 VLAN 内，抓取发往/来自某指定服务器的数据包时。
- 只想让 Wireshark 抓取由某种或某几种应用（程序）生成的数据包时（比如，若网管人员怀疑网络中存在的故障与 DNS 有关，便配置了抓包过滤器，让 Wireshark 只抓取进出 Internet 的 DNS 查询及响应消息）。

除了上面例举的几种情况之外，还有很多时候，也只需让 Wireshark 从网络中抓取特定而非全部数据包。抓包过滤器一经配置并加以应用，Wireshark 将只会抓取经过其过滤的数据包，其余数据包一概不抓，网管人员可借此来采集自己“心仪的”数据包。



注意

使用抓包过滤器时，请务必小心。在许多情况下，运行于网络中的某些应用程序会与某些意想不到的东西（比如，某台服务器）之间有着微妙的关联。因此，在使用 Wireshark 排除网络故障时，若开启了抓包过滤器，请确保未过滤掉某些看似不想关的数据包，否则的话，将发现不了导致故障的真正原因。现举一个常见的简单示例，若故障的表象为 HTTP 应答缓慢，但罪魁祸首却是 DNS 服务器不响应 DNS 查询，那么配置抓包过滤器，让 Wireshark 只抓取发往/来自 TCP 80 端口的流量，再怎么分析抓包文件都将是徒劳无功。

本章会讲解几种抓包过滤器的配置方法。

2.2 配置抓包过滤器

配置抓包过滤器之前，请读者务必考虑以下两个问题：需要让 Wireshark 抓取什么样的数据包；配置抓包过滤器的目的是什么。请别忘了，通不过抓包过滤器过滤检查的数据包，将会被 Wireshark 丢弃。

读者既可以使用 Wireshark 自带的预定义抓包过滤器，也可以根据本章内容自行定义抓包过滤器。

2.2.1 配置准备

打开 Wireshark 软件，按本章内容行事。

2.2.2 配置方法

抓包过滤器应在抓包之前配置，配置步骤如下所列。

1. 要配置抓包过滤器，请点击工具条上左边第二个 **Show the capture options...**按钮，如图 2.1 所示。

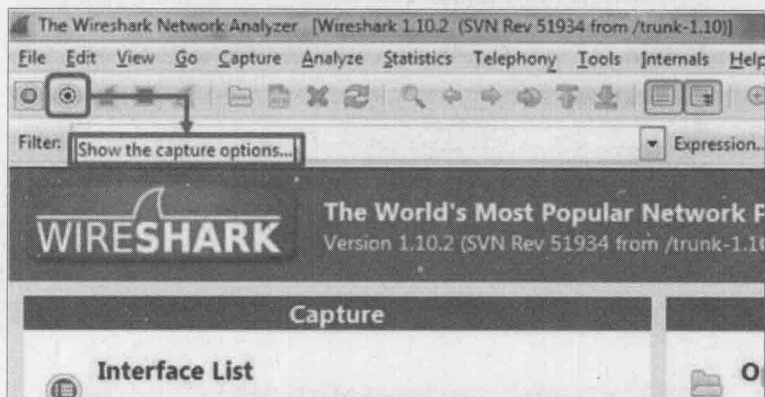


图2.1

2. **Wireshark: Capture Options** 窗口会立刻弹出，如图 2.2 所示。

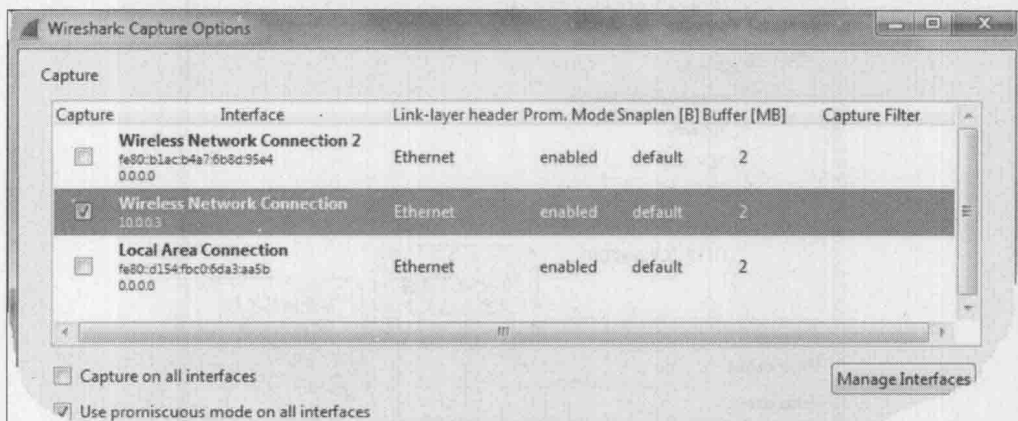


图2.2

3. 选择一块网卡，并双击之，有待配置的抓包过滤器将会生效于这块网卡（可按第 1 章所述来验证哪块网卡为有效的抓包网卡）。
4. **Edit Interface Settings** 窗口会立刻弹出，如图 2.3 所示。

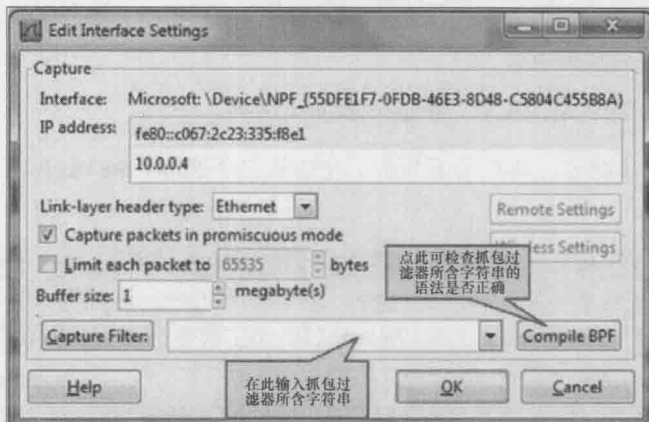


图2.3

5. 现在, 可在 **Capture Filter** 文本框内输入抓包过滤器所含字符串, 或点击 **Capture Filter** 按钮, 在弹出的 **Capture Filter** 窗口中输入抓包过滤器所含字符串和过滤器名称, 如图 2.4 所示。

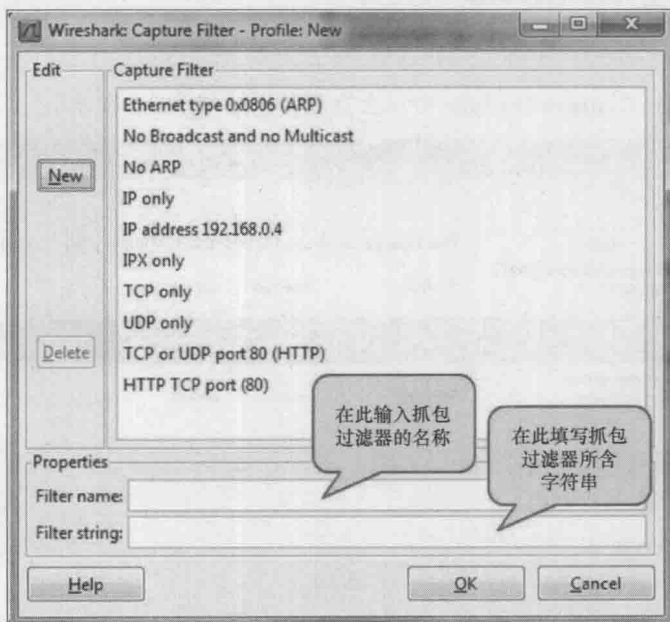


图2.4

2.2.3 幕后原理

可凭籍 **Edit Interface Settings** 窗口中的 **Capture Filter** 文本框, 并基于伯克利数据包过滤

器（**Berkeley Packet Filter, BPF**）的语法，来配置抓包过滤器。在填写完抓包过滤器所含字符串之后，点击 **Compile BPF** 按钮，BPF 编译器将会检查所填字符串的语法，若通不过检查，在弹出的 **Complie Results** 窗口中，会看到“syntax error（语法有误）”的字样。

除此以外，在 **Capture Filter** 文本框内输入抓包过滤器所含的字符串时，若语法正确，该文本框会变绿，否则将会变红。

伯克利数据包过滤器（BPF）只会对输入进 **Capture Filter** 文本框内的字符串的语法进行检查，不会检查其条件是否满足。比方说，若在 **Capture Filter** 文本框内只输入 `host` 不加任何参数，该文本框将会变红，且通不过 BPF 编译器的检查；但若输入的是 `host 192.168.1.1000`，该文本框将会变绿，且能够通过 BPF 编译器的检查。



注意

BPF 所遵循的语法，来源于 Steven McCanne 和 Van Jacobson 于 1992 年在伯克利大学劳伦斯伯克利实验室所写论文 *The BSD Packet Filter: A New Architecture for User-level Packet Capture*。这篇论文电子版的下载链接为 <http://www.tcpdump.org/papers/bpf-usenix93.pdf>。

构成抓包过滤器的字符串名为过滤表达式。这一表达式决定了 Wireshark 对数据包的态度（是抓取还是放弃）。过滤表达式由一个或多个原词（**primitive**）构成。一般而言，每个原词都会包含一个标识符（名称或数字），这一标识符可能会位列一或多个限定符之后。限定符的种类有以下 3 种。

- **type（类型）限定符**：标识符（其形式为名称或数字）所指代的事物。可能存在的类型限定符包括：主机名或主机地址标识符指代的 `host` 限定符、网络号标识符指代的 `net` 限定符、TCP/UDP 端口号标识符指代的 `port` 限定符等。
- **dir（方向）限定符**：指明了发往和/或来自某个标识符（所指代的主机）的数据包的具体流动方向。譬如，`Dir` 限定符 `src` 和 `dst` 分别表示数据包源于/发往某个标识符（所指代的主机）。
- **Proto（协议类型）限定符**：精确指明了数据包所匹配的协议类型。比方说，`Proto` 限定符 `ether`、`ip` 和 `arp` 分别用来指明以太网帧、IP（Internet 协议）数据包和 ARP（地址解析协议）帧。

标识符是用来进行匹配的实际条件。标识符既可以是一个 IP 地址（比如，10.1.1.1），也可以是一个 TCP/UDP 端口号（比如 53），还可以是一个 IP 网络地址（比如，用来表示 IP 网络 192.168.1.0/24 的 192.168.1）。

请看下面这个抓包过滤器。

tcp dst port 135

其中, dst 为 dir 限定符; port 为 type 限定符; tcp 为 Proto 限定符。

2.2.4 拾遗补缺

可配置 Wireshark, 让不同的抓包过滤器生效于不同的网卡, 如图 2.5 所示。

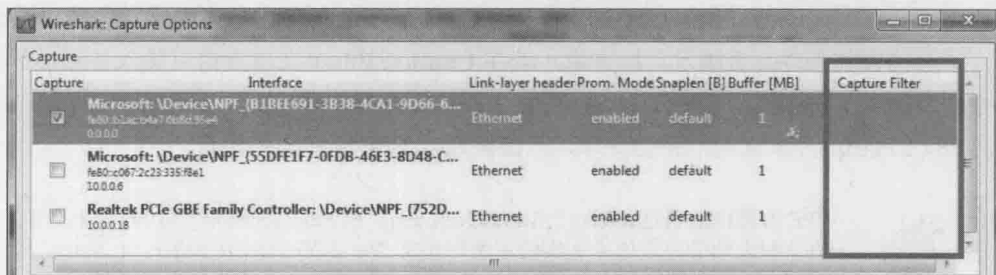


图2.5

当 Wireshark 主机安装了双网卡, 且需让两块网卡分别抓包时, 就有可能需要如此配置。

抓包过滤器(所含字符串)都保存在 Wireshark 安装目录下的 cfilters 文件内。该文件不但会保存预定义的抓包过滤器, 还会保存用户手工配置的过滤器, 可将该文件复制进其他的 Wireshark 主机。cfilters 文件的具体位置要视 Wireshark 主机的操作系统及 Wireshark 软件的安装路径而定。

2.2.5 进阶阅读

1. Wireshark 抓包过滤器(的运作机制)基于 tcpdump 程序, 读者可参考以下链接:
http://www.tcpdump.org/tcpdump_man.html。
2. 读者可参阅 Wireshark 手册页(链接为<http://wiki.wireshark.org/CaptureFilters>)来获取更多与抓包过滤器有关的信息。

2.3 配置Ethernet过滤器

本书提及的 Ethernet 过滤器所指为第二层过滤器, 即根据 MAC 地址来行使过滤功能的抓包过滤器。本节会介绍这种过滤器及其配置方法。

2.3.1 配置准备

下面列出了一些简单的第二层过滤器。

- `ether host <Ethernet host>`: 让 Wireshark 只抓取源于或发往由标识符 Ethernet host 所指定的以太网主机的以太网帧（即所抓以太网流量的源或目的 MAC 地址，与 Ethernet host 所定义的 MAC 地址相匹配）。
- `ether dst <Ethernet host>`: 让 Wireshark 只抓取发往由标识符 Ethernet host 所指定的以太网主机的以太网帧（即所抓以太网流量的目的 MAC 地址，与 Ethernet host 所定义的 MAC 地址相匹配）。
- `ether src <Ethernet host>`: 让 Wireshark 只抓取由标识符 Ethernet host 所指定的以太网主机发出的以太网帧（即所抓以太网流量的源 MAC 地址，与 Ethernet host 所定义的 MAC 地址相匹配）。
- `ether broadcast`: 让 Wireshark 只抓取所有以太网广播流量。
- `ether multicast`: 让 Wireshark 只抓取所有以太网多播流量。
- `ether proto <protocol>`: 所抓以太网流量的以太网协议类型编号，与标识符 `<protocol>` 所定义的以太网协议类型编号相匹配。
- `vlan <vlan_id>`: 让 Wireshark 只抓取由标识符 `<vlan_id>` 所指定的 VLAN 的流量。

要想让抓包过滤器中的字符串起反作用，需在原词之前添加关键字 `not` 或符号“`!`”。举例如下。

抓包过滤器 `Not ether host <Ethernet host>` 或 `! Ether host <Ethernet host>` 的意思是：让 Wireshark 舍弃源自或发往由标识符 Ethernet host 所指定的以太网主机的以太网流量（即所抓以太网流量的源或目的 MAC 地址，与 Ethernet host 所定义的 MAC 地址不匹配）。

2.3.2 配置方法

请看图 2.6 所示的网络，该网络中的一台路由器、一台服务器外加多台 PC 都连接到了同一台 LAN 交换机上。此外，有一台安装了 Wireshark 的笔记本也接入了该 LAN 交换机。在 LAN 交换机上已开启了端口镜像功能，并将整个 VLAN 1（该 LAN 交换机上的所有端口都隶属于 VLAN 1）的流量都重定向给了那台 Wireshark 主机。

图中（紧随 IP 地址）的符号 `/24` 所指为该 IP 地址的 24 位子网掩码，其二进制和十进制的写法为：11111111.11111111.11111111.00000000 和 255.255.255.0。

以下所列为以图 2.6 所示的网络为基础，根据特定需求所配置的若干抓包过滤器。

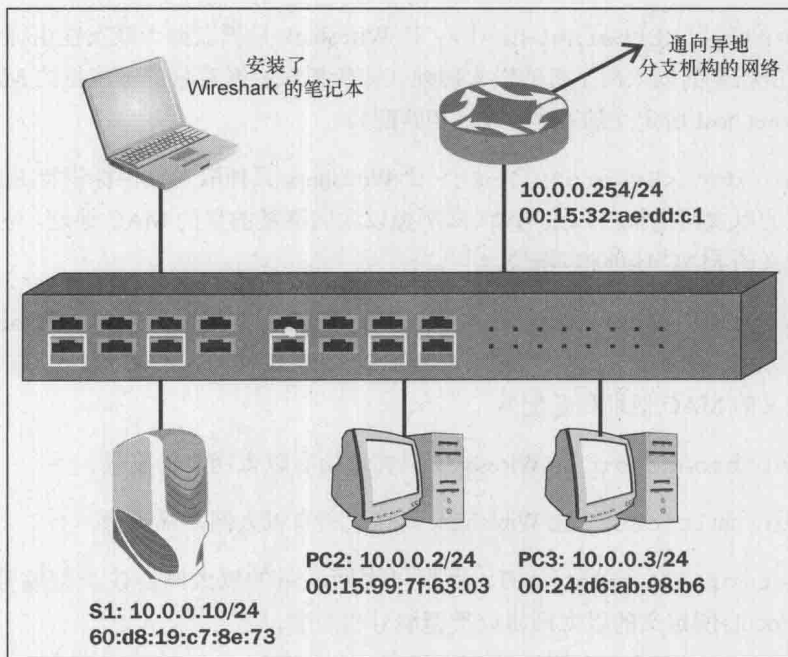


图2.6

1. 要是只想让 Wireshark 抓取源于或发往某一具体 MAC 地址的流量，如源于或发往图中 PC3 的流量，抓包过滤器应如此配置：`ether host 00:24:d6:ab:98:b6`。
2. 要是只想让 Wireshark 抓取发往某一具体 MAC 地址的流量，如发往图中 PC3 的流量，抓包过滤器应如此配置：`ether dst 00:24:d6:ab:98:b6`。
3. 要是只想让 Wireshark 抓取源于某一具体 MAC 地址的流量，如源于图中 PC3 的流量，抓包过滤器应如此配置：`ether src 00:24:d6:ab:98:b6`。
4. 要是只想让 Wireshark 抓取以太网广播流量，抓包过滤器应如此配置：`ether broadcast` 或 `ether dst ff:ff:ff:ff:ff:ff`。
5. 要是只想让 Wireshark 抓取以太网多播流量，抓包过滤器应如此配置：`ether multicast`。
6. 要是只想让 Wireshark 抓取特定以太网类型的流量（以太网类型用十六进制数表示），比如，只抓取以太网类型为 0x0800 的流量，抓包过滤器应如此配置 `ether proto 0800`。

2.3.3 幕后原理

Wireshark Ethernet 抓包过滤器的运作原理非常简单：Wireshark 抓包引擎会先拿用户事先

指定的源和/或目的主机 MAC 地址，与抓取的以太网流量的源和/或目的 MAC 地址相比较，再筛选出源和/或目的 MAC 地址相匹配的流量。

所谓以太网广播流量，是指目的 MAC 地址为广播地址（MAC 地址为全 1，其十六进制写法为 ff:ff:ff:ff:ff:ff）的以太网流量。因此，只要启用了以太网广播过滤器，Wireshark 只会抓取目的 MAC 地址为 ff:ff:ff:ff:ff:ff 的以太网流量。以下所列为常见的以太网广播流量。

- 第三层 IPv4 广播流量，其所对应的第二层以太网帧为以太网广播帧；比方说，目的 IP 地址为 192.168.1.255（此乃 C 类广播地址）的 IPv4 数据包，与其相对应的第二层以太网帧的目的 MAC 地址就是以太网广播地址 ff:ff:ff:ff:ff:ff。
- 有特殊用途的以太网广播流量，比如，IPv4 ARP（地址解析协议）流量，其目的 MAC 地址也是以太网广播地址 ff:ff:ff:ff:ff:ff。



注意

一般而言，有特殊用途的以太网广播流量对网络设备之间的“互通有无”来说必不可缺。除 IPv4 ARP 流量之外，此类流量还包括 RIP 路由协议流量等。

可利用多播过滤器，让 Wireshark 只抓取 IPv4/IPv6 多播流量。

- 但凡 IPv4 多播流量，其以太网帧的目的 MAC 地址必以 01:00:5e 打头。目的 MAC 地址以 01:00:5e 打头的所有以太网帧都将被视为以太网多播帧。
- 但凡 IPv6 多播流量，其以太网帧的目的 MAC 地址均以 33:33 打头。目的 MAC 地址以 33:33 打头的所有以太网帧也将被视为以太网多播帧。

以太网类型所指为以太网帧帧头的 ETHER-TYPE 字段，其值用来表示由以太网帧帧头所封装的高层协议流量的协议类型。当 ETHER-TYPE 字段值为 0x0800、0x86dd 以及 0x0806 时，则以太网帧帧头所封装的分别是 IPv4、IPv6 以及 ARP 流量。

2.3.4 拾遗补缺

- 要想让 Wireshark 只抓取某一特定 VLAN 的流量，抓包过滤器的语法应为：vlan <vlan number>。
- 要想让 Wireshark 只抓取某几个 VLAN 的流量，抓包过滤器的语法应为：vlan <vlan number> and vlan <vlan number> and vlan <vlan number>...。

2.3.5 进阶阅读

ETHER-TYPE 类型值约有近百, 但其中的大多数都不常用, 详情请见 <http://www.mit.edu/~map/Ethernet/Ethernet.txt>。

2.4 配置主机和网络过滤器

所谓主机和网络过滤器, 是指基于 IP 地址的第三层过滤器, 本章会介绍此类过滤器的使用及配置方法。

2.4.1 配置准备

以下所列为一些简单的第三层过滤器。

- `ip` 或 `ipv6`: 让 Wireshark 只抓取 IPv4 或 IPv6 流量。
- `host <host>`: 让 Wireshark 只抓取源于或发往由标识符 `host` 所指定的主机名或 IP 地址的 IP 流量。
- `dst host <host>`: 让 Wireshark 只抓取发往由标识符 `host` 所指定的主机名或 IP 地址的 IP 流量。
- `src host <host>`: 让 Wireshark 只抓取源于由标识符 `host` 所指定的主机名或 IP 地址的 IP 流量。



注意

通过标识符 `host`, 既可以指定 IP 地址, 也可以指定与某个 IP 地址相关联的主机名称。比如, 抓包过滤器 `host www.packtpub.com` 一经配置, Wireshark 就只会抓取发往或源于 Packt 网站的流量了, 即所抓数据包的源或目的 IP 地址 (在某种 Hostname-to-IP address 解析机制里) 跟主机名称 `www.packtpub.com` 相绑定。

- `gateway <host>`: 让 Wireshark 只抓取穿 `host` 而过的流量。标识符 `gateway` 所指定的 `host` 必须为主机名称, 且必须同时在某种 Hostname-to-IP address 解析机制 (比如, 主机名文件、DNS 或 NIS 等) 以及 Hostname-to-Ethernet address 解析机制 (比如, `/etc/ethers` 文件等) 里登记在案。也就是说, 该过滤器一经配置, Wireshark 所抓流量的源或目的 MAC 地址一定为标识符 `gateway` 所指定的 `host` 的 MAC 地址, 但源或目的 IP 地址绝不会是标识符 `gateway` 所指定的 `host` 的 IP 地址。

- `net <net>`: 让 Wireshark 只抓取源于或发往由标识符 `net` 所标识的 IPv4/IPv6 网络号的流量。
- `dst net <net>`: 让 Wireshark 只抓取发往由标识符 `net` 所标识的 IPv4/IPv6 网络号的流量。
- `src net <net>`: 让 Wireshark 只抓取源于由标识符 `net` 所标识的 IPv4/IPv6 网络号的流量。
- `net <net> mask <netmask>`: 让 Wireshark 只抓取源于或发往由标识符 `net` 和 `mask` 共同指明的 IPv4 网络号的流量（对 IPv6 流量无效）。
- `dst net <net> mask <netmask>`: 让 Wireshark 只抓取发往由标识符 `net` 和 `mask` 共同指明的 IPv4 网络号的流量（对 IPv6 流量无效）。
- `src net <net> mask <netmask>`: 让 Wireshark 只抓取源于由标识符 `net` 和 `mask` 共同指明的 IPv4 网络号的流量（对 IPv6 流量无效）。
- `net <net>/<len>`: 让 Wireshark 只抓取源于或发往由标识符 `net` 指明的 IPv4 网络号的流量。
- `dst net <net>/<len>`: 让 Wireshark 只抓取发往由标识符 `net` 指明的 IPv4 网络号的流量。
- `src net <net>/<len>`: 让 Wireshark 只抓取源于由标识符 `net` 指明的 IPv4 网络号的流量。
- `broadcast`: 让 Wireshark 只抓取 IP 广播包。
- `multicast`: 让 Wireshark 只抓取 IP 多播包。
- `ip proto <protocol code>`: 让 Wireshark 只抓取 IP 包头的协议类型字段值等于特定值（等于由标识符 `proto` 所指明的 `protocol code` [协议代码] 值）的数据包。IP 数据包种类繁多，随 IP 包头的协议类型字段值而异，比如，TCP 数据包（协议类型字段值为 6）、UDP 数据包（协议类型字段值为 17）和 ICMP 数据包（协议类型字段值等于 1）等。
- `ip6 proto <protocol>`: 让 Wireshark 只抓取 IPv6（主）包头中下一个包头字段值等于特定值（等于由标识符 `proto` 所指明的 `protocol` 值）的 IPv6 数据包。请注意，无法使用该原词根据 IPv6 扩展包头链中的相关字段值来执行过滤。



注意

在 IPv6 包头中，有一个名为“下一个包头”的字段，用来指明本包头后跟随着哪一种可选扩展包头。IPv6 数据包可以形成扩展包头层层嵌套的局面。对当前版本的 Wireshark 而言，其抓包过滤器不支持基于 IPv6 扩展包头链中的相关字段值来执行过滤。

- `icmp[icmptype]==<identifier>`: 让 Wireshark 只抓取特定类型[icmptype]的 ICMP 数据包，<identifier>表示的是 ICMP 头部中的类型字段值，比如，0 (ICMP echo reply 数据包) 或 8 (ICMP echo request 数据包) 等。

2.4.2 配置方法

现在，根据上一节的内容，举几个抓包过滤器的配置实例。

1. 要让 Wireshark 只抓取源于或发往主机 10.10.10.1 的所有流量，抓包过滤器应如此配置：`host 10.10.10.1`。
2. 要让 Wireshark 只抓取源于或发往主机 `www.cnn.com` 的所有流量，抓包过滤器应如此配置：`host www.cnn.com`。
3. 要让 Wireshark 只抓取发往主机 10.10.10.1 的所有流量（即目的 IP 地址为 10.10.10.1 的数据包），抓包过滤器应如此配置：`dest host 10.10.10.1`。
4. 要让 Wireshark 只抓取源自主机 10.10.10.1 的所有流量（即源 IP 地址为 10.10.10.1 的数据包），抓包过滤器应如此配置：`src host 10.10.10.1`。
5. 要让 Wireshark 只抓取源于或发往 IP 网络 192.168.1.0/24 的所有流量，抓包过滤器应如此配置：`net 192.168.1` 或 `net 192.168.1.0 mask 255.255.255.0` 或 `net 192.168.1.0/24`。
6. 要让 Wireshark 只抓取单播流量，抓包过滤器应如此配置：`not broadcast` 或 `not multicast`。
7. 要让 Wireshark 只抓取源于或发往 IPv6 网络 2001::/16 的 IPv6 数据包，抓包过滤器应如此配置：`net 2001::/16`。
8. 要让 Wireshark 只抓取源于或发往 IPv6 主机 2001::1 的所有流量，抓包过滤器应如此配置：`host 2001::1`。
9. 要让 Wireshark 只抓取 ICMP 流量，抓包过滤器应如此配置：`ip proto 1`。
10. 要让 Wireshark 只抓取 ICMP echo request 流量，抓包过滤器应如此配置：`icmp[icmptype]==icmp-echo` 或 `icmp[icmptype]==8`。在以上两个过滤器中，

icmp-echo 和 8 分别表示 ICMP echo request 数据包的名称和类型（即 ICMP 数据包的 ICMP 头部中的类型字段值和与之对应的名称）。

2.4.3 幕后原理

配置主机过滤器时，若根据主机名执行过滤，则 Wireshark 会（通过某种名字解析服务）把用户输入的主机名转换为 IP 地址，并抓取与这一 IP 地址相对应的流量。比方说，若所配抓包过滤器为 `host www.cnn.com`，Wireshark 会通过某种名字解析服务（多半为 DNS）将其转换为某个 IP 地址，并抓取源于或发往这一 IP 地址的所有数据包。请注意，在此情形，倘若 CNN Web 站点将（访问它的）流量转发给设有另一 IP 地址的其他 Web 站点，Wireshark 也只会抓取 IP 地址为前者的数据包。

2.4.4 拾遗补缺

以下所列为一些常用的抓包过滤器。

- `ip multicast`: 用来抓取 IP 多播数据包。
- `ip broadcast`: 用来抓取 IP 广播数据包。
- `ip[2:2] == <number>`: 用来抓取特定长度的 IP 数据包（`number` 表示 IP 包头中的 IP 包总长度字段值）。
- `ip[8] == <number>`: 用来抓取具有特定 TTL（生存时间）的 IP 数据包（`number` 表示 IP 包头中的 TTL 字段值）。
- `ip[9] == <number>`: 用来抓取特定协议类型的 IP 数据包（`number` 表示 IP 包头中的协议类型字段值）。
- `(ip[12:4] == 1 ip[16:4])`: 表示数据包的源和目的 IP 地址相同。

本章的最后一节会对上述过滤器的语法做进一步的解释。图 2.7 揭示了上述过滤器的基本原理，中括号内的数字用来确定抓包过滤器所要关注的相关协议头部（图中所示为 IP 包头，还可以关注 TCP、UDP 或其他协议头部）的内容，第一个数字指明了抓包过滤器应从协议头部的第几个字节开始关注，第二个数字则定义了所要关注的字节数。

2.4.5 进阶阅读

欲知更多与 Wireshark 抓包过滤器有关的内容，请访问 `tcpdump` 手册页的主页：
http://www.tcpdump.org/tcpdump_man.html。

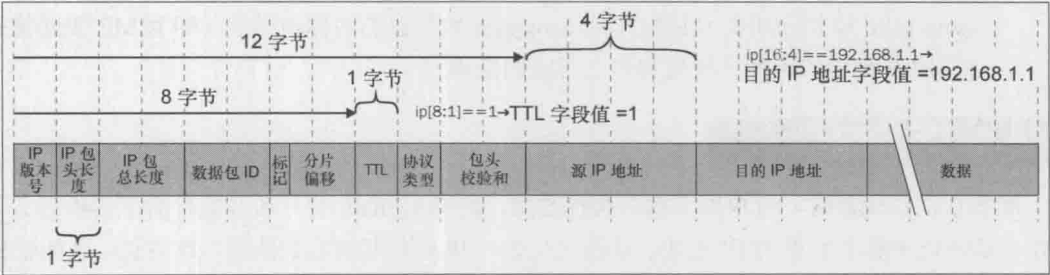


图2.7

2.5 配置TCP/UDP及端口过滤器

本章会介绍在使用 Wireshark 抓包时，如何根据第四层协议 TCP/UDP 的端口号来实施过滤，同时会介绍这种抓包过滤器的用法。

2.5.1 配置准备

以下所列为几种基本的第四层抓包过滤器。

- port <port>: 当根据第四层协议（如 TCP 或 UDP）来实施抓包过滤时，该过滤器一经应用，Wireshark 所抓数据包的（第四层协议的）源或目的端口号将匹配标识符 port 所指明的端口号。
- dst port <port>: 当根据第四层协议（如 TCP 或 UDP）来实施抓包过滤时，该过滤器一经应用，Wireshark 所抓数据包的（第四层协议的）目的端口号将匹配标识符 port 所指明的端口号。
- src port <port>: 当根据第四层协议（如 TCP 或 UDP）来实施抓包过滤时，该过滤器一经应用，Wireshark 所抓数据包的（第四层协议的）源端口号将匹配标识符 port 所指明的端口号。

以下所列为几种根据端口范围来执行过滤的第四层抓包过滤器。

- tcp portrange <p1>-<p2> 或 udp portrange <p1>-<p2>: 用来抓取（源或目的）端口范围介于 p1 和 p2 之间的 TCP 或 UDP 数据包。
- tcp src portrange <p1>-<p2> 或 udp src portrange <p1>-<p2>: 用来抓取源端口范围介于 p1 和 p2 之间的 TCP 或 UDP 数据包。
- tcp dst portrange <p1>-<p2> 或 udp dst portrange <p1>-<p2>: 用来抓取目的端口范围介于 p1 和 p2 之间的 TCP 或 UDP 数据包。

2.5.2 配置方法

现在，根据上一节的内容，举几个抓包过滤器的配置实例。

1. 让 Wireshark 只抓目的端口号为 80 的数据包（HTTP 流量），抓包过滤器应如此配置：
`dst port 80` 或 `dst port http`。
2. 让 Wireshark 只抓源或目的端口号为 5060 的数据包（SIP 流量），抓包过滤器应如此配置：
`port 5060`。
3. 让 Wireshark 只抓所有 TCP 连接中用来发起（SYN 标记位置 1）连接或终止连接（FIN 标记位置 1）的数据包（TCP 连接属于全双工连接，客户端与服务器之间会建立双向连接。也就是说，建立 TCP 连接时，客户端向服务器发起连接之后，服务器也会向客户端发起连接，终止连接亦然），抓包过滤器应如此配置：
`tcp [tcpflags] & (tcp-syn | tcp-fin) != 0`。



注意

请注意，在 `tcp [tcpflags] & (tcp-syn | tcp-fin) != 0` 中，使用的是位与运算，并非逻辑与运算。例如，`010 or 101` 等于 `111`，不等于 `000`。

4. 让 Wireshark 只抓所有 RST 标记位置 1 的 TCP 数据包，抓包过滤器应如此配置：
`tcp [tcpflags]& (tcp-rst) != 0`。
5. 要想让 Wireshark 只抓取特定长度的数据包，抓包过滤器的写法有以下两种^①。
 - `less <length>`：让 Wireshark 只抓取不长于标识符 `less` 所指定的长度的数据包，其等价写法为：`len <= <length>`。
 - `greater <length>`：让 Wireshark 只抓取不短于标识符 `greater` 所指定的长度的数据包，其等价写法为：`<len >= <length>`。
6. 让 Wireshark 只抓源或目的端口范围在 2000 到 2500 之间的 TCP 数据包，抓包过滤器的写法为：`tcp portrange 2000-2500`。
7. 让 Wireshark 只抓源或目的端口范围在 5000 到 6000 之间的 UDP 数据包，抓包过滤器的写法为：`udp portrange 5000-6000`。

有些应用程序在运行时可能需要关联某段连续（而非某个具体）的 TCP 或 UDP 端口号，

^① 译者注：在此处介绍这种抓包过滤器，只会让人丈二和尚摸不着头脑。

若要抓取涉及此类应用程序的流量，则可以根据端口范围来配置抓包过滤器。

2.5.3 幕后原理

第四层协议（主要是指 TCP 或 UDP）属于互连末端应用程序的协议。一端节点（比如，Web 客户端）向另一端节点（比如，Web 服务器）发出连接建立请求时，最常见的举动就是发送（第四层协议）报文。运行在两个末端节点之上，用来发起或接受连接的进程的代号为（第四层）端口号。第 9 章会对此展开深入探讨。

对 TCP 和 UDP 而言，端口号就是用来标识应用程序的代号。这两种第四层协议之间的差别在于，前者属于面向连接的可靠协议，而后者则是无连接（即不建立连接）的不可靠协议。还有一种名叫流控传输协议（**Stream Control Transport Protocol, SCTP**）的第四层协议，这是一种高级版本的 TCP 协议，也使用端口号。

TCP 头部设有若干个标记位，这些标记位的主要作用是建立、维护及拆除连接。当（TCP 报文段的）发送方将其中某一标记位置 1 时，其意在（向 TCP 报文段的接收方）传递某种信号。以下所列为 TCP 头部中几种常用的标记位。

- **SYN**: 用来表示打开连接。
- **FIN**: 用来表示拆除连接。
- **ACK**: 用来确认（通过 TCP 连接）收到的数据。
- **RST**: 用来表示立刻拆除连接。
- **PSH**: 用来表示应将数据提交给末端应用程序（进程）处理。

利用（第四层）抓包过滤器，既可以让 Wireshark 只抓取某特定应用程序生成或接收的流量，也能够筛选出开启了某个标记位的 TCP 流量。

2.5.4 拾遗补缺

下列第四层抓包过滤器可供读者在某些反常情况下（比如，网络遭到攻击时）使用。

- $\text{tcp}[13] \ \& \ 0x00 = 0$: 用来抓取所有标记位都未置 1 的 TCP 流量（在怀疑遭遇了空扫描[null scan]攻击时使用）。
- $\text{tcp}[13] \ \& \ 0x01 = 1$: 用来抓取 **FIN** 位置 1，但 **ACK** 位置 0 的 TCP 流量。
- $\text{tcp}[13] \ \& \ 0x03 = 3$: 用来抓取 **SYN** 和 **FIN** 位同时置 1 的 TCP 流量。

- $\text{tcp}[13] \& 0x05 = 5$: 用来抓取 **RST** 和 **FIN** 位同时置 1 的 TCP 流量。
- $\text{tcp}[13] \& 0x06 = 6$: 用来抓取 **SYN** 和 **RST** 位同时置 1 的 TCP 流量。
- $\text{tcp}[13] \& 0x08 = 8$: 用来抓取 **PSH** 位置 1, 但 **ACK** 位置 0 的 TCP 流量。

图 2.8 揭示了上述 TCP 抓包过滤器的幕后原理。由图中所示 TCP 头部的格式可知, 上述 TCP 抓包过滤器中, $\text{tcp}[13]$ 所含数字 13 指代的是 TCP 头部中的标记字段 (自 TCP 头部的起始处偏移 13 个字节), 而 “=” 后面的数字 1、3、5 等则表示的是标记字段中各 TCP 标记位的置位情况。

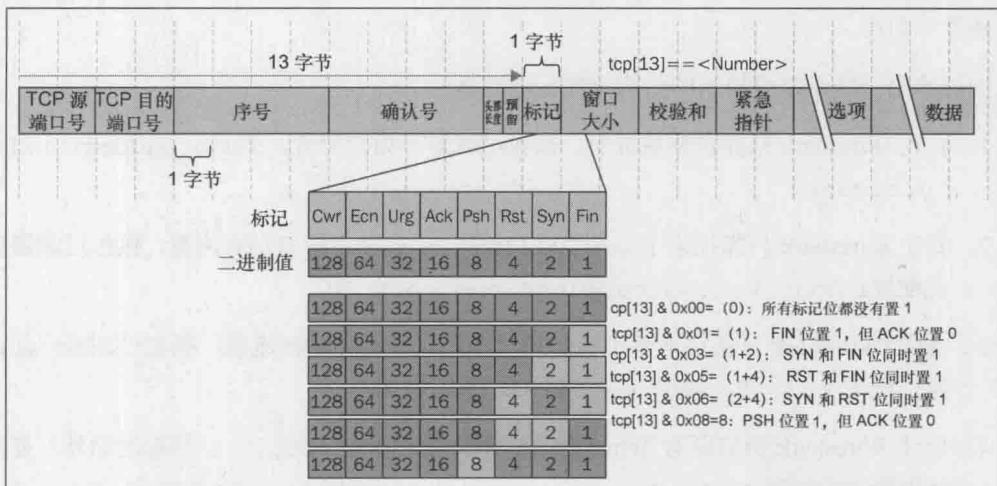


图2.8

2.5.5 进阶阅读

第 9 章会详细介绍 TCP 和 UDP 这两种第四层协议。

2.6 配置复合型过滤器

复合型过滤器也叫结构化过滤器, 由多个过滤条件构成, 过滤条件之间通过 not、and 或 or 之类的操作符来进行关联。

2.6.1 配置准备

结构化抓包过滤器的格式如下所示。

```
[not] primitive [and | or [not] primitive ...]
```

以下所列为创建 Wireshark 抓包过滤器时经常用到的操作符。

- ! 或 not
- && 或 and
- || 或 or

2.6.2 配置方法

编写结构化抓包过滤器也很简单，只需根据本章前几节的内容“拼接”好满足需求的一个个“条件”即可。

下面给出一些经常会用到的结构化抓包过滤器。

1. 要让 Wireshark 只抓单播数据包，抓包过滤器应如此配置：not broadcast and not multicast。
2. 要让 Wireshark 只抓往来于 www.youtube.com 站点的 HTTP 流量，抓包过滤器应如此配置：host www.youtube.com and port 80。
3. 要让 Wireshark 只抓往来于主机 192.180.1.1 的 Telnet 流量，抓包过滤器应如此配置：tcp port 23 and host 192.180.1.1。
4. 要让 Wireshark 抓取所有 Telnet 流量，但由主机 192.168.1.1 发起的除外，抓包过滤器应如此配置：tcp port 23 and not src host 192.168.1.1。

2.6.3 幕后原理

再举一个复杂的结构化抓包过滤器示例。

要让 Wireshark 抓取所有 TCP 源端口范围为 5000~6000 的 Telnet 流量（即源端口范围为 5000~6000，目的端口号为 23 的 TCP 流量），抓包过滤器应如此配置：tcp dst port 23 and tcp src portrange 5000-6000。

2.6.4 拾遗补缺

最后再举几个比较有意思的结构化抓包过滤器，其具体涵义由读者自行分析。

- host www.mywebsite.com and not (port 80 or port 23)
- host 192.168.0.50 and not tcp port 80

- host 10.0.0.1 and not host 10.0.0.2

2.6.5 进阶阅读

欲掌握更多与结构化抓包过滤器有关的内容及示例，请访问以下链接。

- <http://www.packetlevel.ch/html/tcpdumpf.html>
- <http://www.packetlevel.ch/html/txt/tcpdump.filters>

2.7 配置字节偏移和净载匹配型过滤器

就过滤功能而言，字节偏移和净载匹配型过滤器要更加灵活，网管人员可凭籍该工具来配置自定义型抓包过滤器（自定义型过滤器是指所含字段为非 Wireshark 解析器预定义的过滤器，可针对私有协议流量实施过滤）。只要网管人员熟悉所接触的网络协议，且对协议数据包的结构摸得门清，就能针对包中所含特定字符串定制特殊的抓包过滤器，让 Wireshark 在抓包时根据这一过滤器来筛选流量。本章会讲解如何配置这种特殊类型的抓包过滤器，同时还会列举几个在实战中可能会经常用到的配置示例。

2.7.1 配置准备

要配置字节偏移和净载匹配型抓包过滤器，请运行 Wireshark 软件，并按 2.2 节所述步骤行事。

2.7.2 配置方法

1. 字节偏移和净载匹配型抓包过滤器一经应用，Wireshark 便会用其中所含字符串与所抓数据包的相关协议头部中的某些字段值进行比对，并根据比对结果实施过滤。这种过滤器的格式如下所示。

```
proto [Offset: bytes]
```

有了这种过滤器，便可以让 Wireshark 在抓包时，根据 IP、UDP、TCP 等协议头部中的某些字段值来实施过滤。

2. 要想针对 IP 层来实施过滤，字节偏移和净载匹配型抓包过滤器的格式如下所示。

```
ip [Offset: Bytes]
```

3. 要想根据第四层协议头部中的某些字段值，乃至应用程序的某些特征来实施过滤（比如，针对 UDP、TCP 头部中的某些字段值，或 FTP、HTTP 流量的某些特征来实施过

滤), 最常用的字节偏移和净载匹配型抓包过滤器有以下两种。

`tcp[Offset:Bytes]` 或 `udp[Offset:Bytes]`

2.7.3 幕后原理

下面给出了字节偏移和净载匹配型抓包过滤器的常规写法。

`proto` (协议类型, 如 IP、UDP、TCP 等) [从协议头部的开始所偏移的字节数: 抓包过滤器所要检查的字节数]

下面举几个常用的字节偏移和净载匹配型抓包过滤器示例。

1. 要让 Wireshark 只抓目的端口范围为 50~100 的 TCP 数据包, 抓包过滤器应如此配置: `tcp[2:2] > 50 and tcp[2:2] < 100`, 如图 2.9 所示。

中括号内的第一个数字 2 表示: 抓包过滤器应从 (Wireshark 主机网卡所收数据包的) TCP 头部的第 2 个字节起开始检查; 第二个数字 2 则指明了检查范围为 2 字节长, 即只检查 TCP 头部中目的端口号字段值。数字 50 和 100 则划定了端口范围(确定了 TCP 头部中目的端口号字段值的范围)。

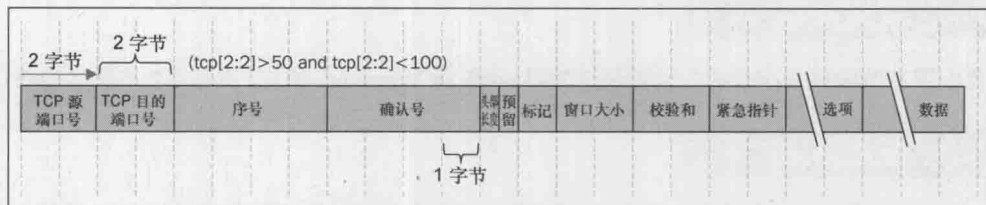


图2.9

2. 要让 Wireshark 只抓窗口大小字段值低于 8192 的 TCP 数据包, 抓包过滤器应如此配置: `tcp[14:2] < 8192`, 如图 2.10 所示。

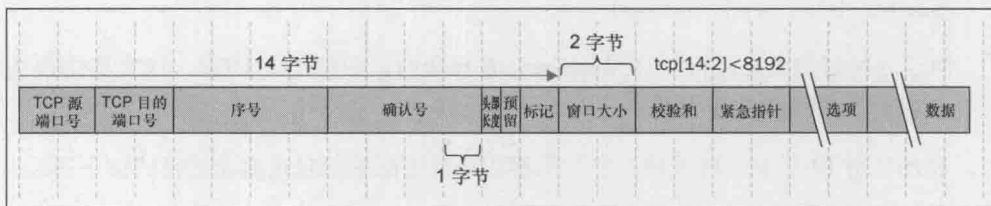


图2.10

中括号内的第一个数字 14 表示: 抓包过滤器应从 (Wireshark 主机网卡所收 TCP 数据包的) TCP 头部的第 14 个字节起开始检查; 第二个数字 2 则指明了检查范围为 2 字节长, 即只检查 TCP 头部中窗口大小字段值; `< 8192` 则指明了检查条件。

有一款 Wireshark 字节偏移和净载匹配型抓包过滤器生成工具，非常好用，请见链接：
<http://www.wireshark.org/tools/string-cf.html>。

2.7.4 拾遗补缺

下面再给几个刊载于 tcpdump 手册页中的字节偏移和净载匹配型抓包过滤器示例。

1. 要让 tcpdump（或 Wireshark）只抓取 TCP 源或目的端口号均为 80 的 HTTP 流量（其实是抓取源或目的端口号均为 80，且只包含实际 HTTP 数据的 TCP 流量。也就是说，在这批数据包的 TCP 头部的 SYN 位、FIN 位或 ACK 位中，有且只有 1 位置 1），抓包过滤器应如此配置：
`tcp port 80 and (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) != 0)。`
2. 要让 tcpdump（或 Wireshark）抓取各条 TCP 会话中的首、尾 2 个数据包，且这些数据包的源和目的 IP 地址均不隶属于抓包主机所在 IP 子网，抓包过滤器应如此配置：
`tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 and not net <local-subnet>。`
 请牢记，TCP 连接为全双工，此过滤器一配，对于每条 TCP 连接，Wireshark 都会抓到 4 个数据包，即建立连接三次握手时，客户端和服务端之间交换的第一个数据包，外加关闭连接四次握手时两者间互发的最后一个数据包。
3. 要让 tcpdump（或 Wireshark）抓取以非以太网封装方式发送的 IP 多播或广播数据包，抓包过滤器应如此配置：
`ether[0] & 1 = 0 and ip[16] >= 224。`
4. 要让 tcpdump（或 Wireshark）抓取所有类型的 ICMP 数据包，但 ICMP echo reply 和 echo request 数据包除外（即抓取所有 ICMP 流量，但由 IP ping 程序生成的流量除外），抓包过滤器应如此配置：
`icmp[icmptype] != icmp-echo and icmp[icmptype] != icmp-echo-reply。`
 请注意，并不是只有执行 ping 命令才能生成 ICMP echo reply 和 echo request 数据包，执行 traceroute 等操作也能生成这两种类型的数据包。

2.7.5 进阶阅读

- Wireshark 官网提供了一款 Wireshark 抓包过滤器生成工具，链接为 <http://www.wireshark.org/tools/string-cf.html>。

虽然该工具生成的抓包过滤器未必总能有效，但用它来练练手还是不错的。

- 读者还可以看一下这篇博文，链接为 http://www.packetlevel.ch/html/txt/byte_offsets.txt。

第3章

显示过滤器的用法

本章涵盖以下主题：

- ▶ 显示过滤器简介；
- ▶ 配置 Ethernet、ARP、主机及网络过滤器；
- ▶ 配置 TCP/UDP 过滤器；
- ▶ 配置特定协议类型的过滤器；
- ▶ 配置字节偏移型过滤器；
- ▶ 配置显示过滤器宏。

3.1 简介

本章会讲解如何操纵 Wireshark 显示过滤器。显示过滤器应用于 Wireshark 抓取数据包之后（此时，Wireshark 抓到的数据可能已经经过了抓包过滤器的过滤），使用它的目的是要让 Wireshark 按需显示已经抓取到的部分数据。

可根据以下限定规则，来配置 Wireshark 显示过滤器，对已经抓取到的数据包做进一步的精挑细选。

- 根据某些参数，比如，IP 地址、TCP/UDP 端口号、URL 或某台服务器的名称。
- 根据某些条件，像“TCP 目的端口号在 1000~2000 之间”或“数据包长度不应长于 1000 字节”这样的描述，都可以算作条件。

- 根据某些现象，比如，TCP 重传、TCP 重复确认、怪异的 TCP 确认方式、数据包中某些原本应该置 0 的标记位实际却置 1、抓包视图中出现了协议错误代码等现象。
- 根据应用程序参数，比如，短消息服务（Message Service, SMS）的始发地和目的地号码，或服务消息块（Server Message Block, SMB）服务器名称等。

通过网络传送的任何数据都可以过滤，在过滤时，还可以根据过滤条件生成相关统计信息和图形^①。

本章会介绍 Wireshark 显示过滤器的各种配置方法，包括通过预制菜单配置、从数据包显示栏内截取、在 Filter 输入栏内直接输入过滤语句等。



注意

请别忘了，捌飧显示过滤器时，所有数据都已被 Wireshark 抓获，显示出的数据只是经过显示过滤器筛选而已。也就是说，抓包文件依旧保存 Wireshark 抓到的所有原始数据，但可在应用显示过滤器之后，让 Wireshark 把经过筛选的数据单独保存为一个新的文件。

3.2 配置显示过滤器

配置显示过滤器时，可选择以下几种方式。

- 借助于显示过滤器窗口。
- 在显示过滤器工具条的 Filter 输入栏里直接输入过滤语句（与此同时，Wireshark 还可以照常抓包；此法注定会成为读者以后最常用的筛选所抓数据包的方法）。
- 在抓包主窗口的包结构显示栏中，将数据包的某个属性值选定为显示过滤器的过滤条件。
- 通过 tshark 或 wireshark 命令行来配置（详见附录）^②。

本章只介绍前三种显示过滤器的配法。

3.2.1 配置准备

每条显示过滤器通常都是由若干原词构成，原词之间通过连接符（如 and 或 or 等）连接，原词之前还可以添加 not 表示相反的意思，其语法如下所列。

[not] Expression [and|or] [not] Expression...

① 译者注：原文是“Any data that is sent over the network can be filtered, and when filtered, you can create statistics and graphs according to it”，译文直译。作者的意思应该是，显示过滤器可与 Wireshark 内置的其他工具搭配使用。

② 译者注：附录中似乎也没有这些内容。

其中, Expression 可以为任意原词形式的过滤表达式, 比如, 表示源 IP 地址的 `ip.src==192.168.1.1`、表示 TCP SYN 标记位置 1 的 `tcp.flags.syn==1`、表示发生 TCP 重传现象的 `tcp.analysis.retransmission` 等; 而连接符 `and`|`or` 则用来连接各个过滤表达式; 每个原词形式的过滤表达式则会包括任意长度的字符串以及一对或多对括号。

下表所列为显示过滤表达式中条件操作符的用途。

类似于 C 语言的操作符	简写形式	描 述	举例
<code>==</code>	eq	等于	<code>ip.addr == 192.168.1.1</code> 或 <code>ip.addr eq 192.168.1.1</code>
<code>!=</code>	ne	不等于	<code>!ip.addr==192.168.1.1</code> 、 <code>ip.addr != 192.168.1.1</code> 或 <code>ip.addr ne 192.168.1.1</code>
<code>></code>	gt	高(长、大)于	<code>frame.len > 64</code>
<code><</code>	lt	低(短、小)于	<code>frame.len < 1500</code>
<code>>=</code>	ge	不高(长、大)于	<code>frame.len >= 64</code>
<code><=</code>	le	不低(短、小)于	<code>frame.len <= 1500</code>
	is present	符合某项参数、满足某个条件, 或出现某个现象	<code>http.response</code>
	contains	包含某个(串)字符	<code>http.host contains cisco</code>
	match	某串字符匹配某个条件	<code>http.host matches www.cisco.com</code>

在参数和条件操作符之间可以不留空格, 也可以保留空格。

在过滤表达式中用条件操作符“`!=`”为 `eth.addr`、`ip.addr`、`tcp.port` 或 `udp.port` 等参数设定条件时, Wireshark 总会为其配上黄色背景色, 这表示过滤表达式语法无误, 但并不会生效, 原因如下。

当人们输入类似于 `ip.addr != 192.168.1.100` 这样的过滤表达式时, 是希望 Wireshark 过滤掉抓包文件中源和目的 IP 地址均不为 192.168.1.100 的数据包。可惜, 每个 IP 数据包必含 2 个 IP 地址, 一为源 IP 地址, 一为目的 IP 地址。Wireshark 根据上面这条过滤表达式执行显示过滤功能时, 只要发现源或目的 IP 地址至少有一个不为 192.168.1.100, 便会判定条件为真。出于这个原因, 要想让 Wireshark 显示源和目的 IP 地址均不为 192.168.1.100 的数据包, 显示过滤表达式的正确写法应该是: `!(ip.addr == 192.168.1.100)`。

下表所列为显示过滤表达式中逻辑关系操作符的用途。

类似于 C 语言的操作符	简写形式	描述	举例
&&	and	逻辑与	ip.src==10.0.0.1 and tcp.flags.syn==1 由 IP 主机 10.0.0.1 发出的所有 SYN 标记位置 1, 且只有该位置 1 的 TCP 数据包 (即 IP 主机 10.0.0.1 建立或尝试建立 TCP 连接时发出的首个数据包)
	or	逻辑或	ip.addr==10.0.0.1 or ip.addr==10.0.0.2 所有发往或源于 IP 主机 10.0.0.1 或 10.0.0.2 的数据包
!	not	逻辑非	not arp and not icmp 除 ARP 和 ICMP 数据包之外的所有数据包

3.2.2 配置方法

可选择下列途径来配置显示过滤器。

借助于显示过滤器窗口

要用此法配置显示过滤器, 请把鼠标移至过滤器工具条上的 **Expression...** 按钮, 如图 3.1 所示。

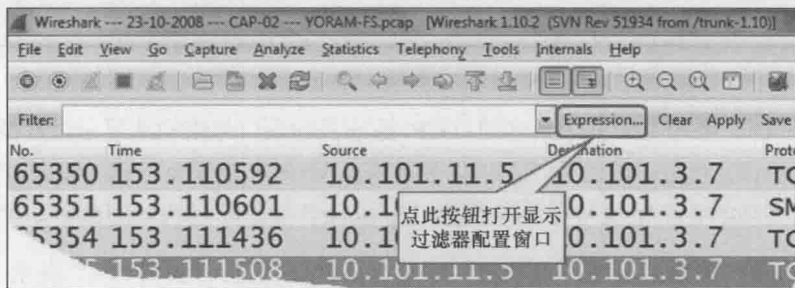


图3.1

点击此按钮, 图 3.2 所示的 Filter Expression 窗口将立刻弹出。

该窗口由 5 个重要区域构成, 如下所列。

- **Field name** (协议头部中的字段名称) 区域: 在该区域, 可利用 Wireshark 预定义的协议模板来配置显示过滤器所含各参数。点最左边的“+”号, 即可浏览到相关协议的各个属性 (或协议头部中各字段的名称), 并可选择相应的属性作为显示过滤器的参数。

试举一例, 要想基于某一具体的 IPv4 地址来构造显示过滤器, 就得先找到 IPv4 协议, 点其左边的“+”号, 便会暴露出 Wireshark 所支持的 IPv4 的各项属性 (或 IPv4 包头中的各个字段), 然后再选择 **ip.addr** 作为显示过滤器的参数即可。

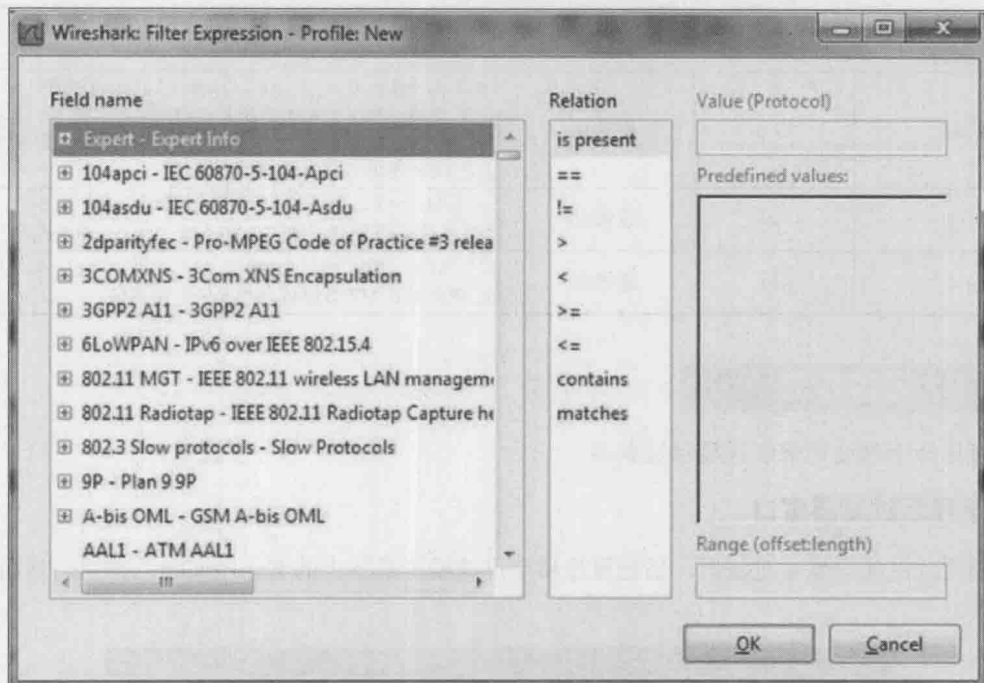


图3.2

再举一例，要想基于某个具体的 **TCP** 源或目的端口号来构造显示过滤器，需先找到 **TCP** 协议，点击左边的“+”号，将会暴露出 Wireshark 所支持的 **TCP** 的各项属性（或 **TCP** 头部的各个字段），然后再选择 **tcp.port** 作为显示过滤器的参数即可。

- **Relation（关系）区域：**可从该区域选择条件操作符。选择“=”表示“等于”，选择“!=”表示“不等于”，依次类推。

试举一例，要想让 Wireshark 只显示包含 **SIP INVITE** 方法的数据包，需先在 **Field name** 找到 **SIP** 协议，点其左边的“+”号，在暴露出的 Wireshark 所支持的 **SIP** 协议的各项属性中选择 **sip.Method**；然后在 **Relation** 区域中选择“=”；最后在 **Value** 区域中输入 **invite**。

- **Value（值）区域：**可在该区域的输入栏内输入事先从 **Field name** 区域中选择的协议头部字段（或协议属性）的字段值（或属性值）。

试举一例，要想让 Wireshark 只显示 **TCP** 头部中 **SYN** 标记位置 1 的数据包，需先在 **Field name** 找到 **TCP** 协议，点击左边的“+”号，然后在暴露出的 Wireshark 所支持的 **TCP** 的各项属性（或 **TCP** 头部中的各个字段）中选择 **tcp.flags.syn**，最后在 **Value**

区域的输入栏中输入 1。

- **Predefined values**（预定义值或预定义选项）区域：该区域是否有效，取决于定义显示过滤器时，在 **Field name** 区域中所选择的协议类型或协议属性。该区域的内容既有可能是布尔值（True 或 False），也有可能是 Wireshark 为某种协议或某种协议的某项属性预先定义的一系列选项。

试举一例，在 **Field name** 区域的 **TCP** 协议中，包含了一个名为 **tcp.option_kind** 的属性，此属性与 TCP 头部选项有关（欲知更多与 TCP 头部选项有关的信息，请阅读本书第 9 章）。若在配置显示过滤器时，选择了该属性，则只要点选 **Relation** 区域内的相关条件操作符，在 **Predefined values**（预定义值）区域内便会出现 Wireshark 为该属性预先定义的某些选项。

- **Range (offset: length)**（范围（偏移：长度））区域：可利用该区域中的输入栏，按照“偏移字节数：过滤器检测长度”的格式，来构造字节偏移型显示过滤器。

在显示过滤器工具条的 Filter 输入栏内直接输入显示过滤语句

只要掌握了显示过滤器的配置语法，在显示过滤器工具条的 Filter 输入栏内直接输入显示过滤语句，可谓是一种最为方便的配置显示过滤器的方法（见图 3.3）。

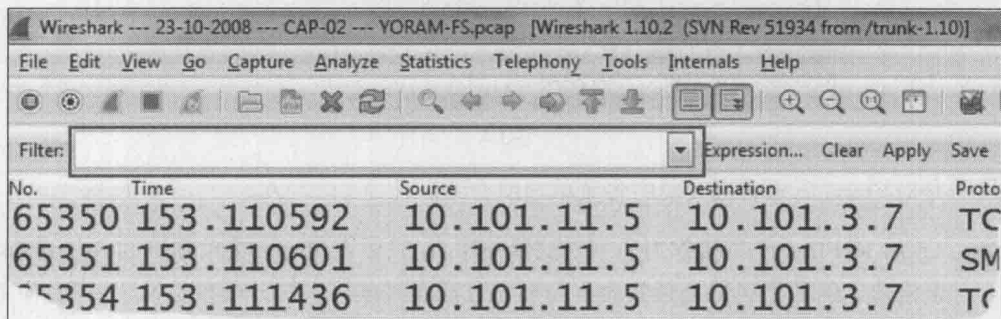


图3.3

向 Filter 输入栏内输入显示过滤语句所包含的字符时，输入栏的背景色可能会呈以下三种颜色之一。

- **绿色**：这表示输入的过滤语句正确，可应用于抓包文件。
- **红色**：这表示输入的过滤语句有误，在应用于抓包文件之前必须修改。
- **黄色**：只要过滤语句中包含了操作符“!=”，Filter 输入栏的背景色就会呈黄色，这并不表示过滤语句有误，只是提醒用户，过滤语句在应用于抓包文件之后，可

能不能生效。

在抓包主窗口的数据包结构区域内，将数据包的某个属性值指定为显示过滤器的过滤条件

这是一种定义显示过滤器的快捷方法。可在抓包主窗口中的数据包结构区域内，把数据包的某个属性（特征或协议头部字段值）指定为显示过滤器。为此，请在该区域内选中相关数据包的某个属性，单击右键。在右键菜单中，包含了几个与显示过滤器有关的菜单项，如图 3.4 所示。

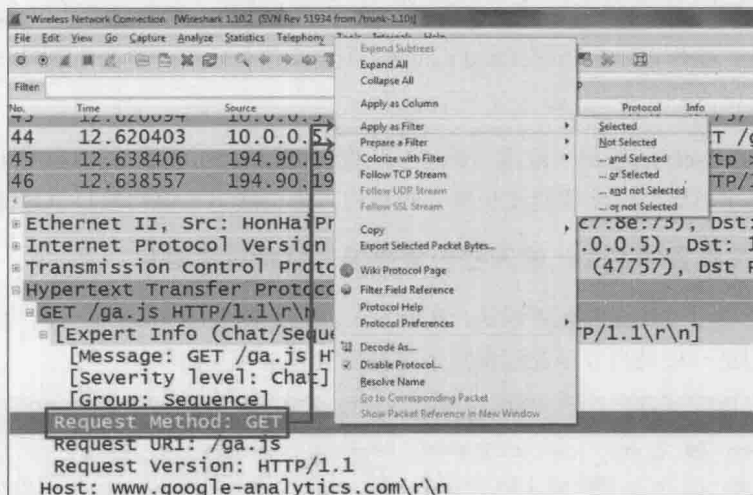


图 3.4

以下是对图 3.4 所示右键菜单中各菜单项的介绍。

- **Apply as Filter**（直接作为显示过滤器使用）：只要点选了该菜单项下的各子菜单项，事先选定的数据包的属性将会作为显示过滤器（或其中的一项参数），并同时作用于抓包文件。
- **Prepare a Filter**（作为有待应用的显示过滤器）：只要点选了该菜单项下的各子菜单项，事先选定的数据包的属性将会成为有待应用的显示过滤器（或其中的一项参数）（选定后，需点下 **Apply** 按钮才能生效）。

以下所列为上述两个右键菜单项中都包含的两个子菜单项。

- **Selected**：将选择选定的字段或参数作为显示过滤参数。
- **Not Selected**：以逻辑非的方式将选定的字段或参数作为显示过滤参数。

现举例对以上两个子菜单项的作用加以说明。若在某个 HTTP 数据包的 `hypertext transfer protocol` 下选中 `request.method: GET`，同时单击右键，并在右键菜单中选择了 **Apply as Filter** 菜单项下的 **Selected** 子菜单项，则 Wireshark 将会在显示过滤工具栏的 Filter 输入栏内自动生成显示过滤表达式 `http.request.method == GET`；若选择了 **Apply as Filter** 菜单项下的 **Not Selected** 子菜单项，则 Wireshark 会在显示过滤工具栏的 Filter 输入栏内自动生成显示过滤表达式 `!(http.request.method == "GET")`。这也正是这两个子菜单项的区别所在。

此外，还可以使用 **Apply as Filter** 和 **Prepare a Filter** 菜单项所包含的“... and selected”、“... or selected”、“... and not selected”或“or ... or not selected”子菜单项来构造显示过滤表达式。

3.2.3 幕后原理

显示过滤器为 Wireshark 软件所独有。用 Wireshark 执行抓包分析任务时，有很多地方都会用到显示过滤器，相关内容会在本书后续章节随文讲解。

在显示过滤工具条的 Filter 输入栏内输入显示过滤器时，可借助于自动补齐特性，来完成过滤器的构造。比如，若在 Filter 输入栏内输入 `tcp.f` 时，自动补齐特性将会生效，会使 Wireshark 在该输入栏下自动列出所有以 `tcp.f` 打头的显示过滤器参数（即 TCP 数据包的属性或 TCP 头部中的字段），如图 3.5 所示。对于本例，以 `tcp.f` 打头的显示过滤器参数是 `tcp.flag`（可利用该参数来引用 TCP 头部中的各标记位字段值）。

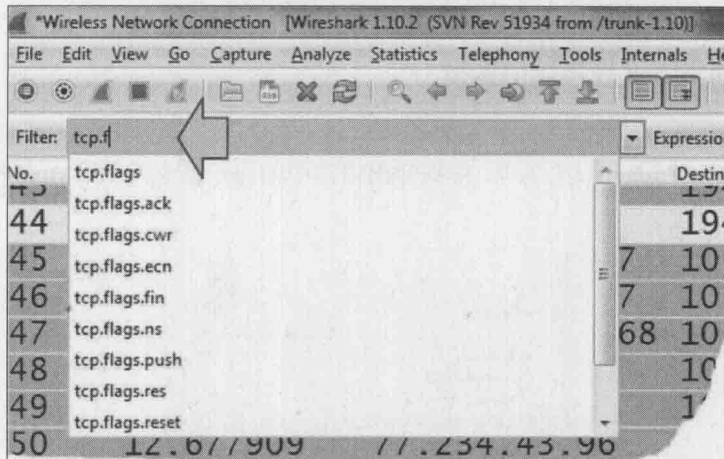


图3.5

3.2.4 拾遗补缺

本节将介绍几个与 Wireshark 显示过滤器有关的操作技巧。

如何获悉显示过滤器所包含的参数

在 Wireshark 抓包主窗口的数据包结构区域中,只要选中了任何一种协议头部的某个字段,与该字段相对应的显示过滤参数将会出现在抓包主窗口底部状态栏的左侧,如图 3.6 所示。

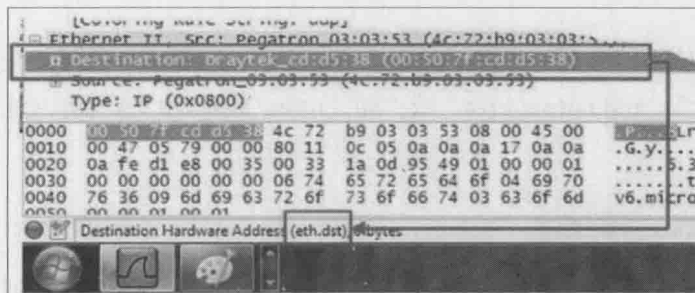


图3.6

如何在数据包列表区域中添加新列

可在 Wireshark 抓包主窗口的数据包结构区域中,把数据包的某个属性(或协议头部中的某个字段),作为数据包列表区域中的新列。具体的操作方法是,选中数据包的某个属性(或协议头部中的某个字段),点击右键,在右键菜单中选择 **Apply as Column** 列。比方说,可把 `tcp.window_size_value` 属性作为数据包列表区域中的新列,以便在抓包时同步观察 TCP 窗口大小。TCP 的性能与窗口大小息息相关,第 9 章会对此展开深入讨论。

如何保存经过显示过滤器筛选的数据

要想把应用了显示过滤器之后所看见的数据包另存为一个抓包文件,请点击 **File** 菜单下的 **Export Specified Packets...** 菜单项,根据弹出窗口中的提示信息来执行相应的操作,如图 3.7 所示。

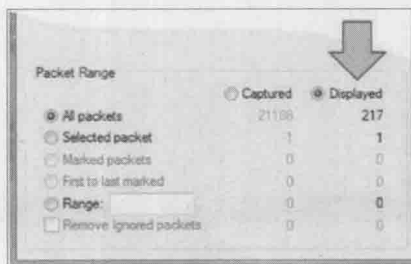


图3.7

3.3 配置Ethernet、ARP、主机和网络过滤器

本章会介绍如何配置第二层过滤器（基于 Ethernet 地址或 Ethernet 帧的某些属性来进行过滤）和第三层过滤器（基于 IP 地址或某 IP 数据包的某些属性来进行过滤）。此外，还会讲解如何配置地址解析协议（ARP）过滤器。

3.3.1 配置准备

配置 Ethernet 显示过滤器的目的，是要让 Wireshark 只显示相关的第二层以太网帧；配置 IP 显示过滤器的目的，则是让 Wireshark 只显示必要的第三层 IP 数据包。第一种过滤器所依据的是 MAC 地址或 Ethernet 帧的某些属性，第二种过滤器则要仰仗 IP 地址或 IP 数据包的某些属性。

以下两个显示过滤参数经常会在以“帧间间隔时间”为条件来行使过滤功能的显示过滤器中使用。

- `frame.time_delta`: 该参数是指当前帧与 Wireshark 所抓上一帧之间的（接收或抓取）时间间隔，即 Wireshark 在抓到了上一帧之后隔了多久，收到了当前帧。本书第 5 章会介绍其用法。
- `frame.time_delta_displayed`: 该参数是指当前帧与 Wireshark 显示出的上一帧之间的（抓取或接收）时间间隔，即 Wireshark 抓到了已显示出的上一帧（已抓到但未予显示的帧不算）后隔了多久，收到了当前帧。该参数的用法也将在本书第 5 章介绍。

通过分析 Wireshark 所抓数据帧之间的时间间隔，可对解决 TCP 性能问题起到很大的帮助。可在 Wireshark IO Graphs 中利用以上两个参数，来监控 TCP 的性能。

以下所列为实战中常用的 L2（Ethernet）显示过滤器。

- `eth.addr == <MAC Address>`: 让 Wireshark 只显示具有指定 MAC 地址的数据帧。
- `eth.src == <MAC Address>`: 让 Wireshark 只显示具有指定源 MAC 地址的数据帧。
- `eth.dst == <MAC Address>`: 让 Wireshark 只显示具有指定目的 MAC 地址的数据帧。
- `eth.type == <Protocol Type（十六进制数，格式为 0xNNNN）>`: 让 Wireshark 只显示指定以太网类型的流量。

以下所列为实战中常用的 ARP 显示过滤器。

- `arp.opcode == <value>`: 让 Wireshark 只显示指定类型的 ARP 帧（ARP 帧按其

所含操作代码字段值，可分为：ARP 应答帧、ARP 响应帧、RARP 应答帧、RARP 响应帧）^①。

- `arp.src.hw_mac == <MAC Address>`: 让 Wireshark 只显示由指定 MAC 地址的主机发出的 ARP 帧^②。

以下所列均为实战中常用的 IP 显示过滤器。

- `ip.addr == <IP Address>`: 让 Wireshark 只显示发往或源自设有指定 IP 地址的主机的数据包。
- `ip.src == <IP Address>`: 让 Wireshark 只显示由设有指定 IP 地址的主机发出的数据包。
- `ip.dst == <IP Address>`: 让 Wireshark 只显示发往设有指定 IP 地址的主机的数据包。
- `ip.ttl == <value>`、`ip.ttl < value`或`ip.ttl > <value>`: 让 Wireshark 只显示 IP 包头中 TTL 字段值为指定值的数据包。
- `ip.len = <value>`、`ip.len > <value>`或`ip.len < <value>`: 让 Wireshark 只显示指定长度的 IP 数据包（IP 包头中有一个 2 字节的总长度字段）。
- `ip.version == <4/6>`: 让 Wireshark 只显示具有指定 IP 版本号的 IP 数据包（IP 包头，不论 IPv4 还是 IPv6，都包含了一个 1 字节的版本号字段）。

3.3.2 配置方法

下表所列均为若干常用 L2 和 L3 Wireshark 显示过滤器的举例。

地址格式	语法	举例
MAC (以太网) 地址	<code>eth.addr == xx:xx:xx:xx:xx:xx</code> <code>eth.addr == xx-xx-xx-xx-xx-xx</code> <code>eth.addr == xxxx.xxxx.xxxx</code> 在以上过滤表达式中，x 为十六进制数 0~f	<code>eth.addr == 00:50:7f:cd:d5:38</code> <code>eth.addr == 00-50-7f-cd-d5-38</code> <code>eth.addr == 0050.7fcd.d538</code>
以太网广播地址	<code>Eth.addr == ffff.ffff.ffff</code>	
IPv4 主机地址	<code>ip.addr == x.x.x.x</code> 其中，x 为 0~255	<code>ip.addr == 192.168.1.1</code>

① 译者注：原文为“arp.opcode == <value>: This is used for ARP requests/responses”，译文酌改。

② 译者注：原文是“This is used to capture the ARP address of the sender”。是显示过滤器，不是抓包过滤器，作者你要 capture 什么啊？译者无语，译文酌改。

续表

地址格式	语法	举例
IPv4 网络地址	ip.addr == x.x.x.x/y 其中, x 为 0~255, y 为 0~32	ip.addr == 192.168.200.0/24 该过滤表达式涵盖了 C 类网络 192.168.200.0/24 中的所有 IP 地址
IPv6 主机地址	ipv6.addr == x:x:x:x:x:x:x:x ipv6.addr == x::x:x:x:x 在以上过滤表达式中, x 为十六进制数 0~f	ipv6.addr == fe80::85ab:dc2e:ab12:e6c7
IPv6 网络地址	ipv6.addr == x::/y 其中, x 为十六进制数 0~f, y 为 0~128	ipv6.addr == fe80::/16 该过滤表达式涵盖了 IPv6 前缀 fe80::/16 所隶属的全部 IPv6 地址

上表给出了 IPv4 和 IPv6 地址与显示过滤器参数 ip.addr 和 ipv6.addr 配搭使用时的表示方法。只要 Wireshark 显示过滤器语句中包含有 IPv4 或 IPv6 地址, 都可以采用与上表相同的表示方法。

Ethernet 过滤器

Ethernet 过滤器分为以下两类。

- 要让 Wireshark 只显示发往或源于具有某 MAC 地址的主机的数据帧, 显示过滤器的写法应类似于 eth.src == 10:0b:a9:33:64:18 或 eth.dst == 10:0b:a9:33:64:18。
- 要让 Wireshark 只显示以太网广播帧, 显示过滤器的写法为: Eth.dst == ffff.ffff.ffff。

ARP 过滤器

ARP 过滤器也分以下两类。

- 要让 Wireshark 只显示 ARP 请求帧, 显示过滤器的写法为 arp.opcode == 1。
- 要让 Wireshark 只显示 ARP 应答帧, 显示过滤器的写法为 arp.opcode == 2^①。

IP 和 ICMP 过滤器

- 要让 Wireshark 只显示由设有指定 IP 地址的主机发出的 IP 数据包, 显示过滤器的写法应类似于 ip.src == 10.1.1.254。
- 要让 Wireshark 只显示所有 IP 数据包, 但设有某指定 IP 地址的主机发出的 IP 数据包除外, 显示过滤器的写法应类似于 !ip.src == 64.23.1.1。

① 译者注: 其实 RARP 帧也属于 ARP 帧, 只是作者不知道而已。

- 要让 Wireshark 只显示交换于某一对 IP 主机之间的所有 IP 数据包, 显示过滤器的写法应类似于 `ip.addr == 200.1.1.1 and ip.addr == 192.168.1.1`。
- 要让 Wireshark 只显示发往 IP 多播目的地址的所有数据包, 显示过滤器的写法应类似于 `ip.dst == 224.0.0.0/4`。
- 要让 Wireshark 只显示发源于 IP 子网 192.168.1.0/24 的所有 IP 数据包, 显示过滤器的写法为 `ip.src == 192.168.1.0/24`。
- 要让 Wireshark 只显示发往或源于设有某个 (或某些) IPv6 地址的主机的 IPv6 数据包, 显示过滤器的写法应类似于:

- `ipv6.addr == ::1`
- `ipv6.addr == 2008:0:130F:0:0:09d0:666A:13ab`
- `ipv6.addr == 2006:0:130f::9c2:876a:130b`
- `ipv6.addr == ::`

复杂的过滤器

- 要让 Wireshark 只显示由隶属于指定 IP 子网 (比如 10.0.0.0/24) 的主机, 发往域名中包含指定字符串的网站 (比如“sohu”) 的所有 IP 流量, 显示过滤器的写法为: `ip.src == 10.0.0.0/24 and http.host contains "sohu"`。
- 要让 Wireshark 只显示由隶属于指定 IP 子网 (比如 10.0.0.0/24) 的主机, 访问域名以 .com 结尾的网站的所有 IP 流量, 显示过滤器的写法为: `ip.addr == 10.0.0.0/24 or http.host matches "\.com$"`。
- 要让 Wireshark 只显示发源于指定 IP 子网 (比如 10.0.0.0/24) 的所有 IP 广播流量, 显示过滤器的写法为: `ip.src == 10.0.0.0/24 and eth.dst == ffff.ffff.ffff`。
- 要让 Wireshark 只显示所有广播包, 但主机在执行 ARP 请求操作时所触发的广播包除外, 显示过滤器的写法为: `not arp and eth.dst == ffff.ffff.ffff`。
- 要让 Wireshark 显示除 ICMP 包和 ARP 帧以外的所有流量, 显示过滤器的写法为: `not arp && not icmp` 或 `not arp and not icmp`^①。

① 译者注: 原文是 “To check for all the packets that are not ICMP, use `!arp || !icmp`, and to check for all the packets that are not ARP, use `not arp or not icmp`”, 译文改。

3.3.3 幕后原理

本节是对上一节所举显示过滤器示例的幕后原理解释。

以太网广播

以太网广播帧是指目的 MAC 地址为全 1 的以太网帧，正因如此，要让 Wireshark 只显示已抓取的所有以太网广播帧，显示过滤器应写成：`eth.dst == ffff.ffff.ffff`（十六进制数 F 等于二进制数 1111）。

IPv4 多播

IPv4 多播数据包的目的 IP 地址范围介于 224.0.0.0~239.255.255.255 之间，若转换为二进制，则介于 11100000.00000000.00000000.00000000~11101111.11111111.11111111.11111111 之间。

仔细观察 IPv4 多播地址的二进制表示方式，应不难发现，IPv4 多播地址一定是以 1110 打头。因此，要让 Wireshark 只显示已抓取的所有 IPv4 多播数据包，显示过滤器应写成：`ip.dst == 224.0.0.0/4`。也就是说，首字节的头 4 位为 1110（11100000，224），掩码长度为 4 位的 IP 地址都属于 IPv4 多播地址范围，此类 IPv4 地址的首字节总是介于 224~239 之间。

IPv6 多播

IPv6 多播地址的首字节总是 ff，随后的一个字节由 4 位标记字段和 4 位范围字段组成。因此，要筛选出 IPv6 多播数据包，显示过滤器就应该写成 `ipv6.dst == ff00::/8`。ff00::/8 表示以 ff 打头的所有 IPv6 地址，即 IPv6 多播地址。

3.3.4 进阶阅读

➤ 欲知更多有关以太网（LAN）的内容，请参阅本书第 7 章。

3.4 配置 TCP/UDP 过滤器

TCP 和 UDP 是 IP 协议族中的两种主要协议，都可供驻留在不同主机上的应用程序“互通有无”。只要在某台主机上执行了某款网络应用程序的客户端程序，便拉开了从某个 TCP/UDP 源端口（具体的端口号为操作系统随机选择，但通常都高于 1024）向（早已监听多时的）该应用程序服务器端目的 TCP/UDP 端口（端口号一般为提前预设或已登记在案的固定端口号）建立 TCP/UDP 会话的序幕。上述源端口号和目的端口号，外加客户端主机和服务器端主机的 IP 地址，可唯一地标识某特定主机与服务器之间运行此款应用程序所建立的 TCP/UDP 会话。TCP 和 UDP 头部自然也会包含源端口号字段和目的端口号字段。

TCP 和 UDP 头部还包含其他字段。UDP 头的结构非常简单，而 TCP 头的结构要复杂许多。因此，在配置显示过滤器时，TCP 过滤器所涉及的参数也会多得多。

本节会介绍各种类型的 TCP/UDP 显示过滤器的配置方法。

3.4.1 配置准备

配置显示过滤器之前，需明确应让 Wireshark 从抓包文件里筛选出哪些数据包，并基于此来精确编制显示过滤语句。

要想根据 TCP/UDP 端口号来筛选数据包，可用以下显示过滤器。

- `tcp.port == <value>` 或 `udp.port == <value>`: 让 Wireshark 在显示数据包时，根据指定的 TCP/UDP 源、目端口号来筛选。
- `tcp.dstport == <value>` 或 `udp.dstport == <value>`: 让 Wireshark 在显示数据包时，根据指定的 TCP/UDP 目的端口号来筛选。
- `tcp.srcport == <value>` 或 `udp.srcport == <value>`: 让 Wireshark 在显示数据包时，根据指定的 TCP/UDP 源端口号来筛选。

UDP 头的结构非常简单，只包含源/目端口号字段、数据包长度字段，以及校验和字段。因此，对 UDP 数据包而言，最重要的特征就是源、目端口号。

TCP 头部则截然不同。因为 TCP 是一种面向连接的协议，内置有可靠的传输机制，所以 TCP 头部要比 UDP 头部复杂得多。不过，Wireshark 完全能够理解 TCP 所具备的面向连接以及可靠性保证等机制。Wireshark 提供了 `tcp.flags`、`tcp.analysis` 等诸多功能强大的涉及 TCP 的显示过滤参数，只要运用得当，发现并解决 TCP 性能问题（比如，TCP 重传、重复确认、零窗口等问题）或运作问题（TCP 半开连接、会话重置等问题）自然不在话下。

以下所列为实战中常用的有关 TCP 的显示过滤参数。

- `tcp.analysis`: 可用该参数来作为分析与 TCP 重传、重复确认、窗口大小有关的网络性能问题的参照物。在这一过滤参数名下，还包含多个子参数（可在 Filter 输入栏内，借助自动补齐特性，来获取该参数名下完整的子参数列表），如下所列。
 - `tcp.analysis.retransmission`: 用来让 Wireshark 显示重传的 TCP 数据包。
 - `tcp.analysis.duplicate_ack`: 用来让 Wireshark 显示确认多次的 TCP 数据包。

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



注意

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

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



注意

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

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

3.4.2 配置方法

先举几个 TCP/UDP 显示过滤器的配置实例。

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

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

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

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

导致 TCP 重传的原因有很多，本书第 9 章会对此展开深入讨论。



注意

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

再举几个与 TCP/UDP 有关的显示过滤器配置实例。

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

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

3.4.3 幕后原理

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

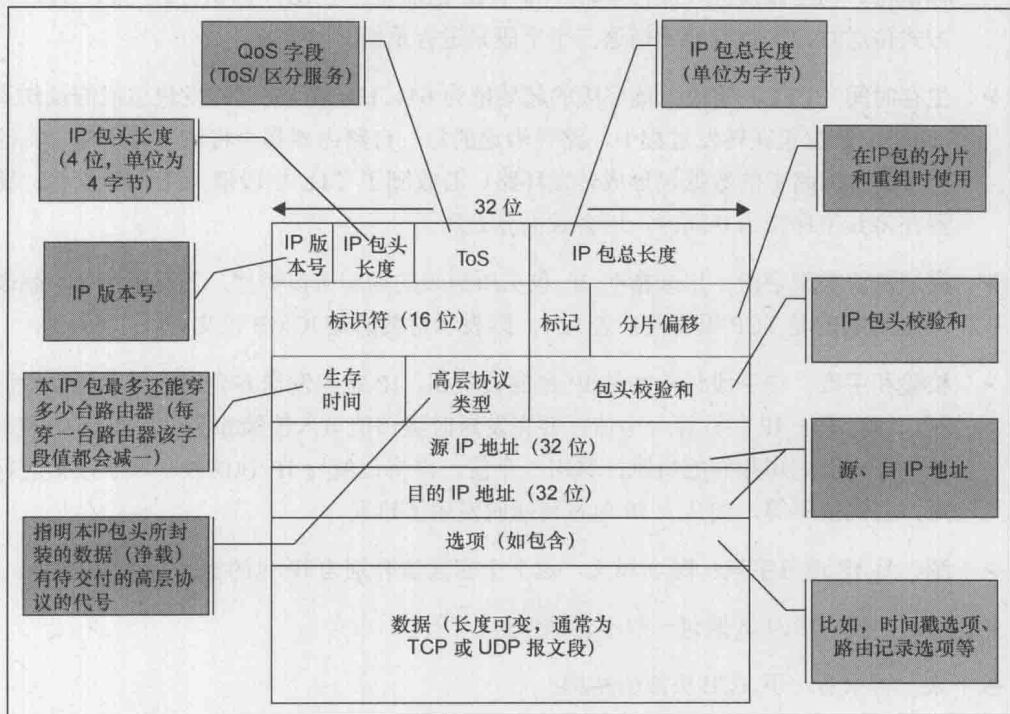


图 3.8

下面简单介绍一下 IPv4 包头中的若干重要字段。

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



注意

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

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

接下来，再来看一下 TCP 头部的结构。

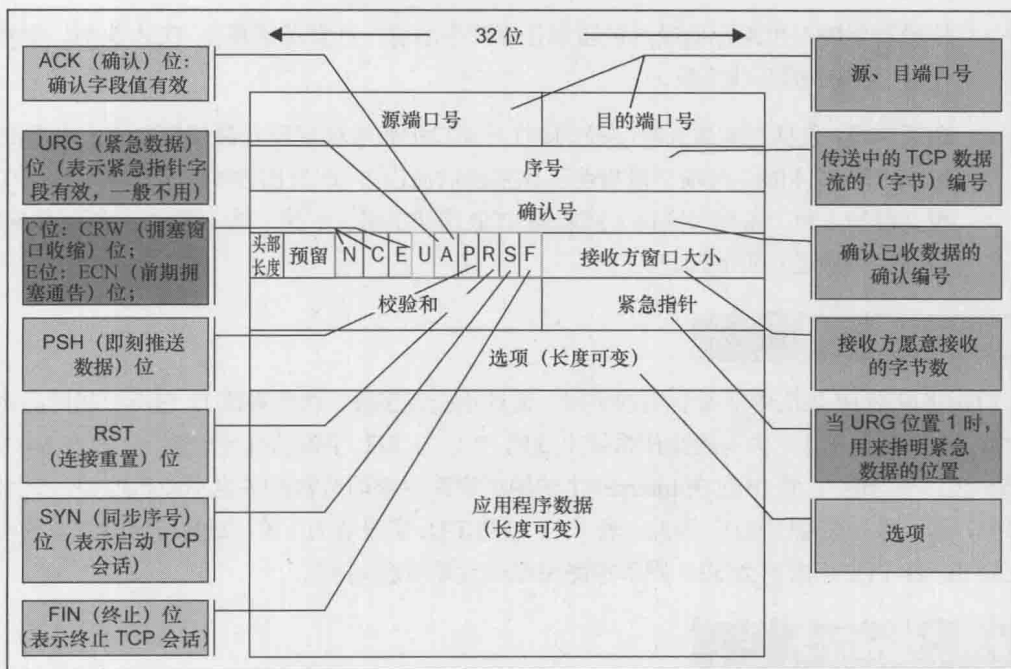


图3.9

下面来介绍一下 TCP 头部中的若干重要字段。

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

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

3.4.4 拾遗补缺

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

3.4.5 进阶阅读

- 欲进一步了解与 TCP/IP 协议栈有关的内容，请参阅第 9 章。

3.5 配置协议所独有的显示过滤器

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

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

3.5.1 配置准备

要配置显示过滤器，只需运行 Wireshark 软件，无需其他任何准备。

3.5.2 配置方法

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

HTTP 显示过滤器

以下所列为一实战中常用的 HTTP 显示过滤器。

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

DNS 显示过滤器

此处来举几个 DNS 显示过滤器示例。

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

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

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

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

FTP 显示过滤器

以下所列为实战中常用的 FTP 显示过滤器。

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

3.5.3 幕后原理

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

以下所列为正则表达式中元字符的含义。

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

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

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

可利用上述元字符来配置非常复杂的显示过滤器，下面例举若干示例。

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

3.5.4 进阶阅读

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

3.6 配置字节偏移型过滤器

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

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

3.6.1 配置准备

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

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

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

3.6.2 配置方法

先举几个字节偏移型显示过滤器的例子，如下所示。

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

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

3.6.3 幕后原理

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

3.7 配置显示过滤器宏

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

3.7.1 配置准备

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

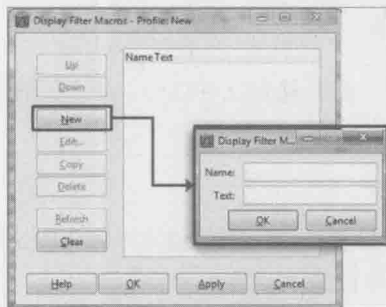


图3.10

3.7.2 配置方法

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

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

3.7.3 幕后原理

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

第4章

基本信息统计工具的法

本章涵盖以下主题:

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

4.1 简介

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

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

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

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

4.2 Statistics菜单中Summary工具的法

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

4.2.1 准备工作

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

4.2.2 使用方法

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

Summary 窗口会立刻弹出, 如图 4.2 所示。

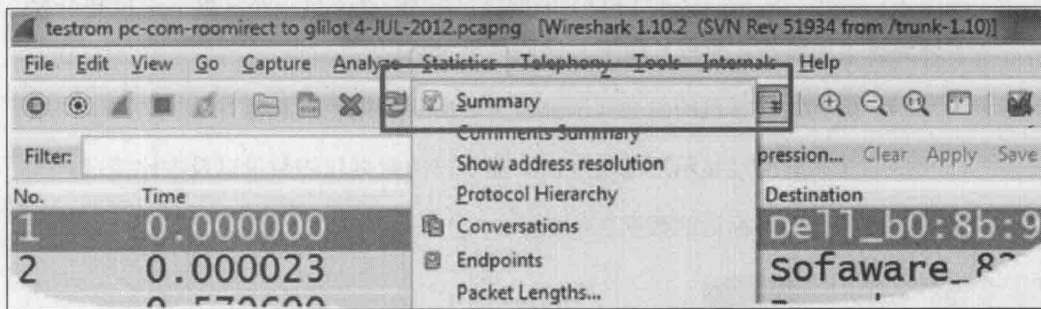


图4.1

以下是对 **Summary** 窗口中重要信息的解读。

1. 由图 4.2 可知, **Summary** 窗口的上半部分包括下述区域。

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

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

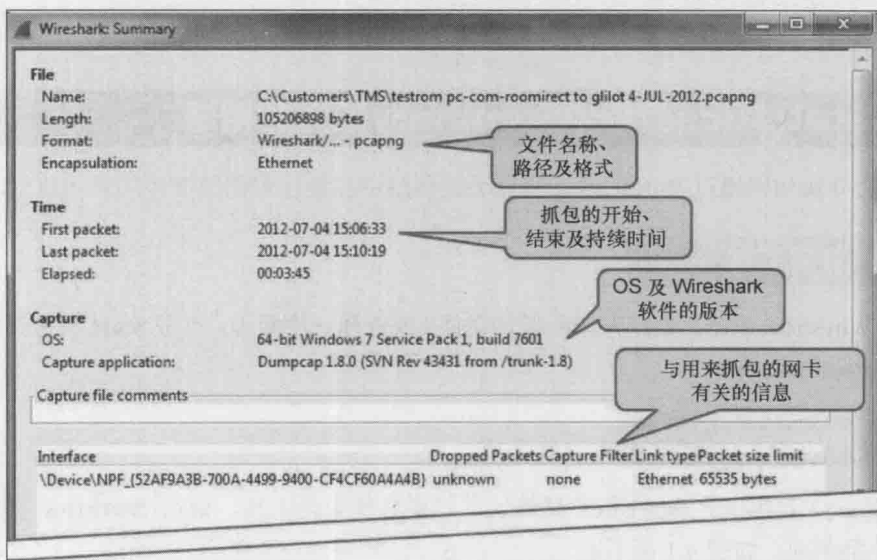


图4.2

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

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

4.2.3 幕后原理

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

4.2.4 拾遗补缺

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

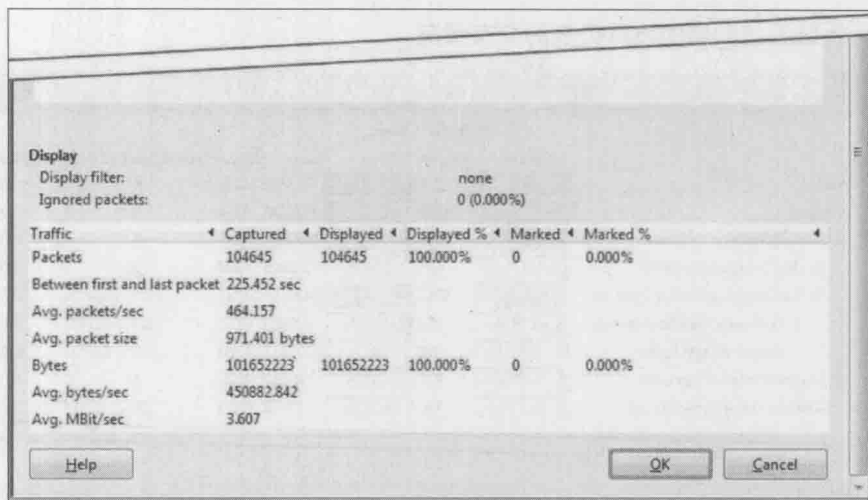


图4.3

4.3 Statistics菜单中Protocol Hierarchy工具的用法

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

4.3.1 准备工作

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

4.3.2 使用方法

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

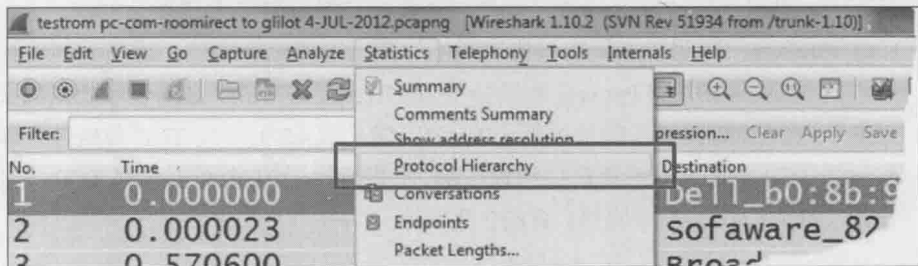


图4.4

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

抓包文件所含数据包归属的协议类型的分布情况。

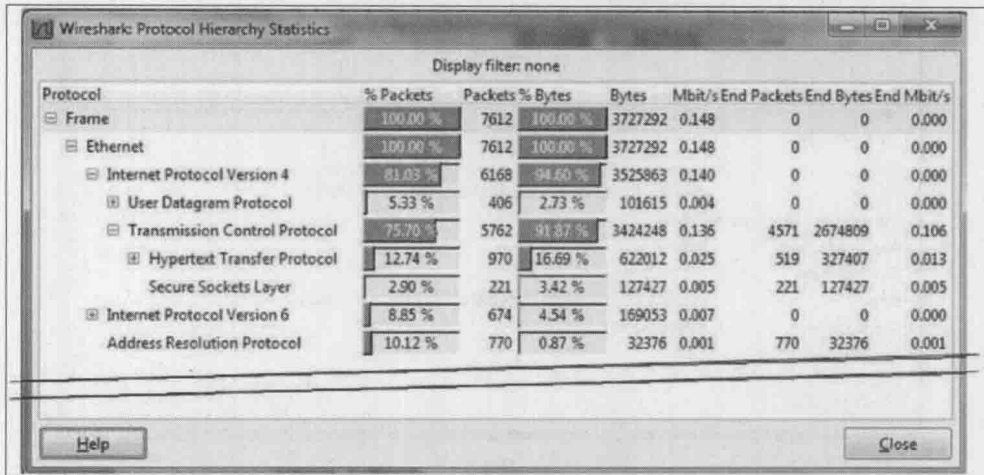


图4.5

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

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



注意

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

4.3.3 幕后原理

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

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

4.3.4 拾遗补缺

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

4.4 Statistics菜单中Conversation工具的用法

本节会介绍如何获取网络中的设备间的对话信息。

4.4.1 准备工作

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

4.4.2 使用方法

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

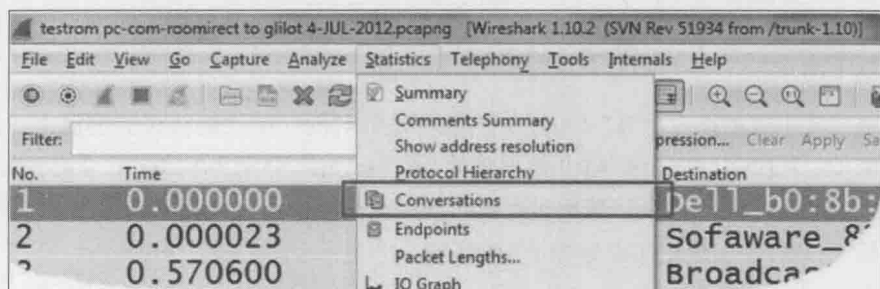


图4.6

Conversation 窗口会立刻弹出, 如图 4.7 所示。

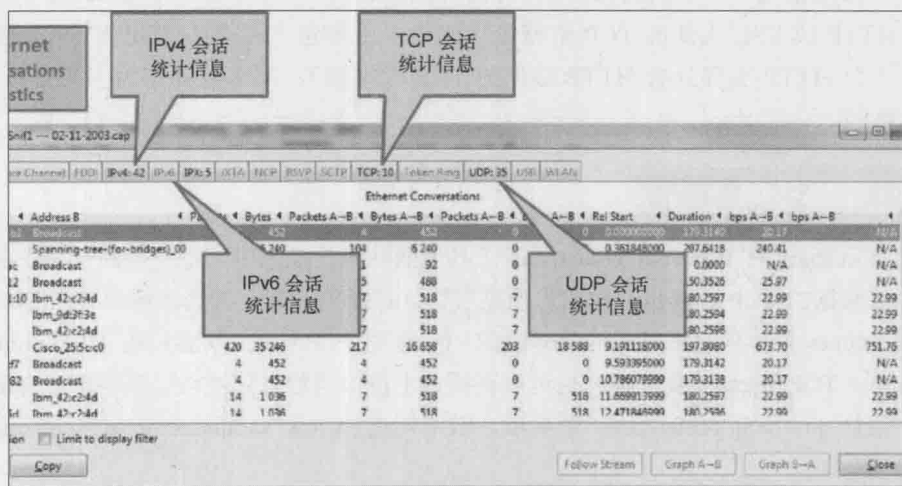
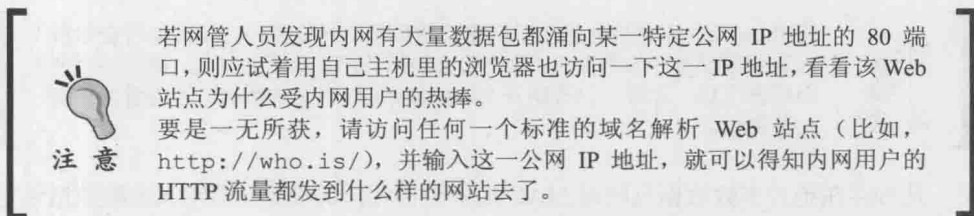


图4.7

可点选图 4.7 中相应的标签来观看网络中的主机之间在第二、三、四层上的对话。可点选：

- **Ethernet 标签**：来观察具有不同 MAC 地址的主机间发生过什么样的“交流”。
- **IPv4 标签**：来观察具有不同 IPv4 地址的主机间有过什么样的“沟通”。
- **TCP 或 UDP 标签**：来观察具有不同 IPv4 地址的主机间所建立的各种 TCP（或 UDP）会话。

通过观察主机之间在第二、三、四层上的对话，既可以发现发生在第二层的广播风暴，也可以查明发生在第三、四层的 Internet 链路负载过高等问题。



要是勾选了 **Conversation** 窗口左下角的 **Limit to display filter** 复选框，那么该窗口将会显示经过显示过滤器过滤的信息。

要想用相应的名称来取代 **Conversation** 窗口显示的 MAC 地址、IP 地址，以及 TCP/UDP 端口号，需勾选左下角的 **Name resolution**。但在此之前，还应在主窗口的 **View** 菜单的 **Name Resolution** 菜单项下点选相应的子菜单项。

要是只想观看某单一的 TCP 或 UDP 对话（conversation），需执行两个步骤。

1. 在抓包主窗口的数据包列表区域中选择隶属于该 TCP 或 UDP 对话（即 TCP 或 UDP 流）的某个数据包，再点击右键，于右键菜单中选择 **Follow TCP Stream** 或 **Follow UDP Stream** 菜单项。这便定义好了一个已应用于抓包文件的显示过滤器。
2. 选择 **Statistics** 菜单，点击 **Conversation** 菜单项，打开 **Conversation** 窗口，勾选左下角的 **Limit to display filter** 复选框。

4.4.3 幕后原理

一次对话（conversation）是指发生于一对特定端点（主机、服务器或网络设备）间的所有流量。比方说，一次 IP 对话是指交换于具有不同 IP 地址的两台主机间的所有流量；而一次 TCP 或 UDP 对话则包括了 4 大特征（源、目 IP 地址外加源、目端口号）全都匹配的数据包。

4.4.4 拾遗补缺

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

Ethernet 对话统计信息

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

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



注意

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

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



注意

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

IP 对话统计信息

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

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

络)。通过图 4.8, 可以清楚地看见有台主机正在扫描网络。

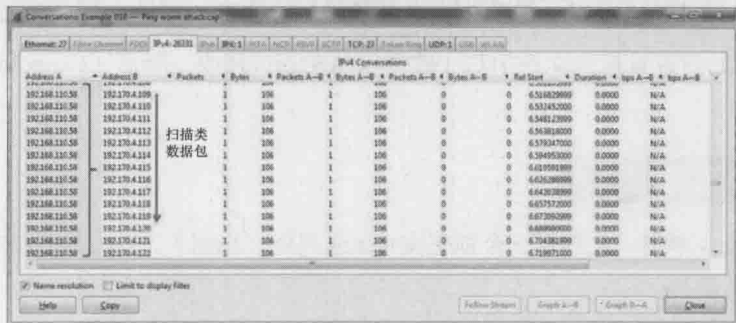


图 4.8

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

TCP/UDP 对话统计信息

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

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

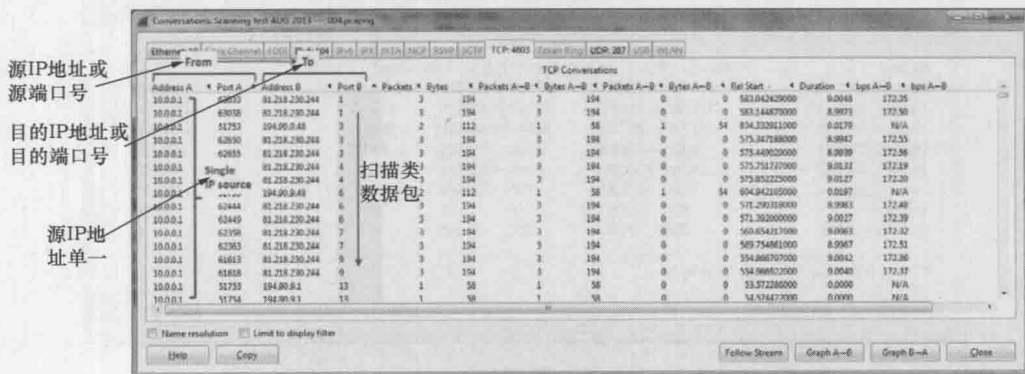


图 4.9

4.5 Statistics菜单中Endpoints工具的使用法

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

4.5.1 准备工作

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

4.5.2 使用方法

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

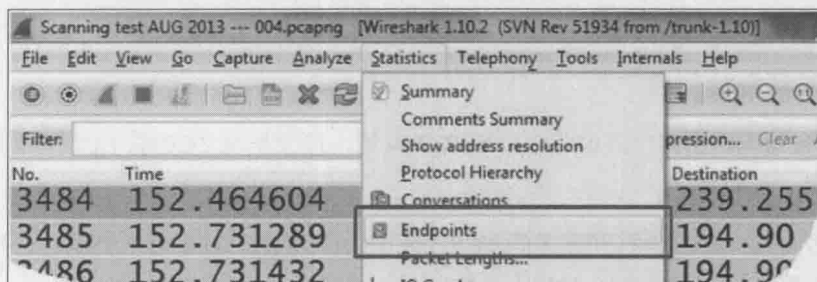


图4.10

Endpoints 窗口将立刻弹出，如图 4.11 所示。

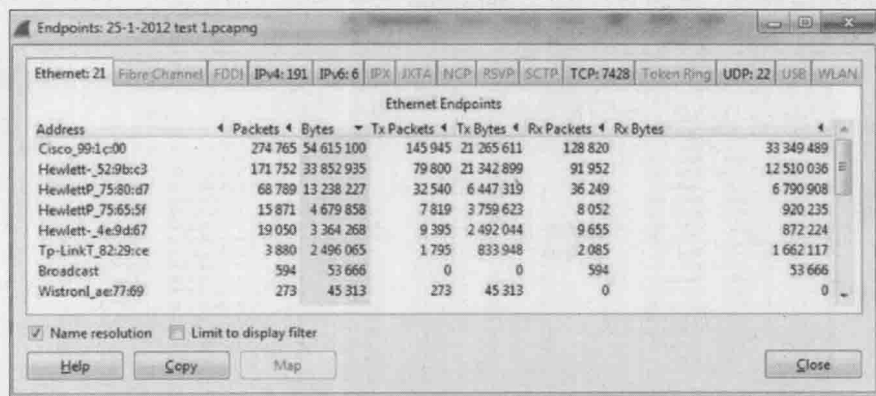


图4.11

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

4.5.3 幕后原理

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

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

4.5.4 拾遗补缺

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

如图 4.12 所示, 通过点击 **Ethernet: 21** 标签, 可判断出在这一网络中心的内网之中应部署有 4 台 HP 服务器外加一台 Cisco 路由器。判断依据是, 在勾选了 **Name resolution** 复选框之后, **Address** 一系列下的 MAC 地址中出现了网卡 (网络设备) 芯片制造商的 ID。

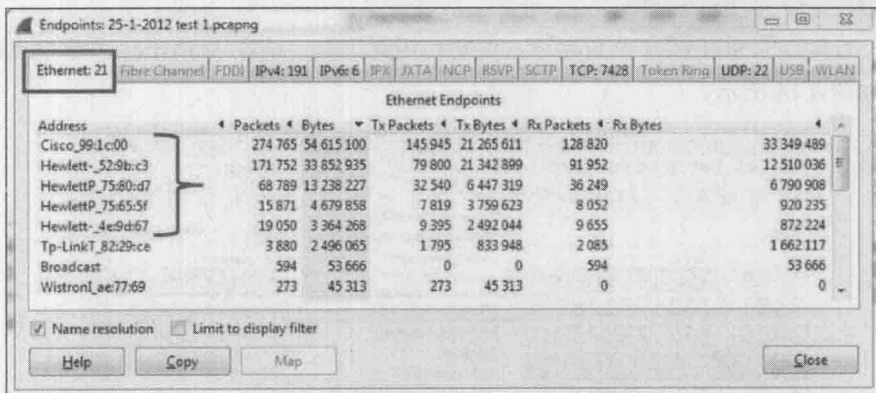


图4.12

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

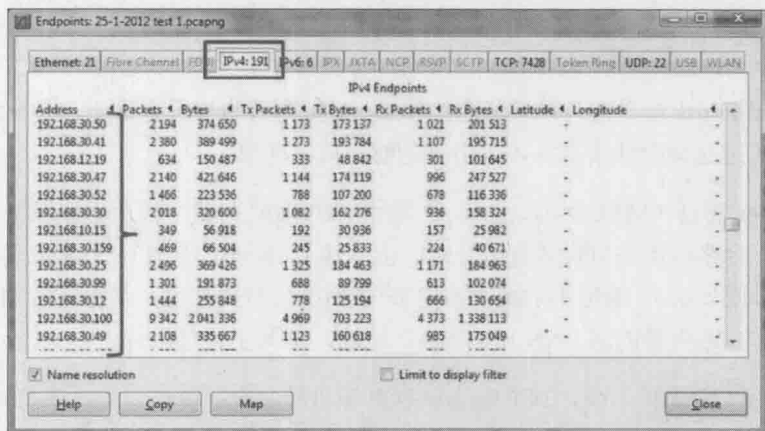


图4.13

4.6 Statistics菜单中HTTP工具的用法

本节会介绍如何查看抓包文件中有关 HTTP 流量的统计信息。

4.6.1 准备工作

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

4.6.2 使用方法

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

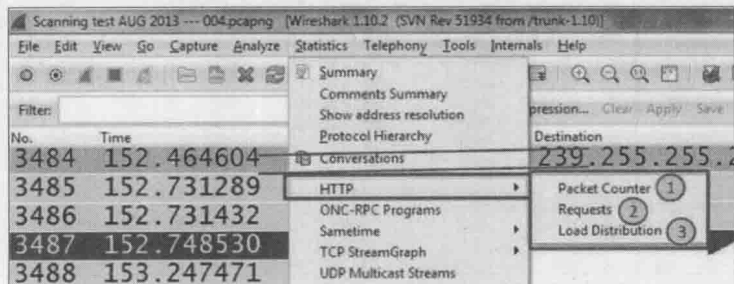


图4.14

HTTP 菜单项名下有以下三个子菜单项。

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

首先, 来看一下 Packet Counter 子菜单项的用法

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

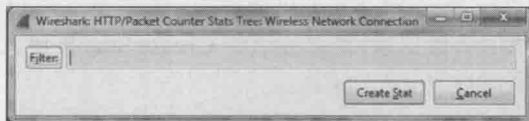


图4.15

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

Topic / Item	Count	Rate (ms)	Percent
Total HTTP Packets	3461	0.015396	
HTTP Request Packets	468	0.002082	13.52%
POST	25	0.000111	5.34%
GET	360	0.001601	76.92%
NOTIFY	71	0.000316	15.17%
SEARCH	12	0.000053	2.56%
HTTP Response Packets	340	0.001512	9.82%
??? broken	0	0.000000	0.00%
1xx: Informational	0	0.000000	0.00%
2xx: Success	250	0.001112	73.53%
3xx: Redirection	88	0.000391	25.88%
4xx: Client Error	2	0.000009	0.59%
5xx: Server Error	0	0.000000	0.00%
Other HTTP Packets	2653	0.011802	76.65%

图4.16

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

其次,来介绍一下 **Requests** 子菜单项的用法。

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

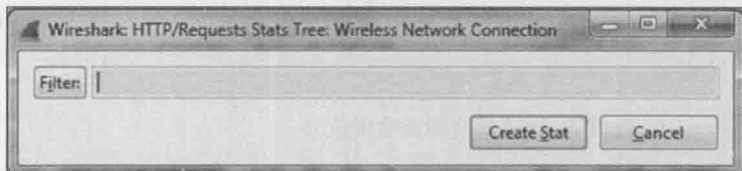


图4.17

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

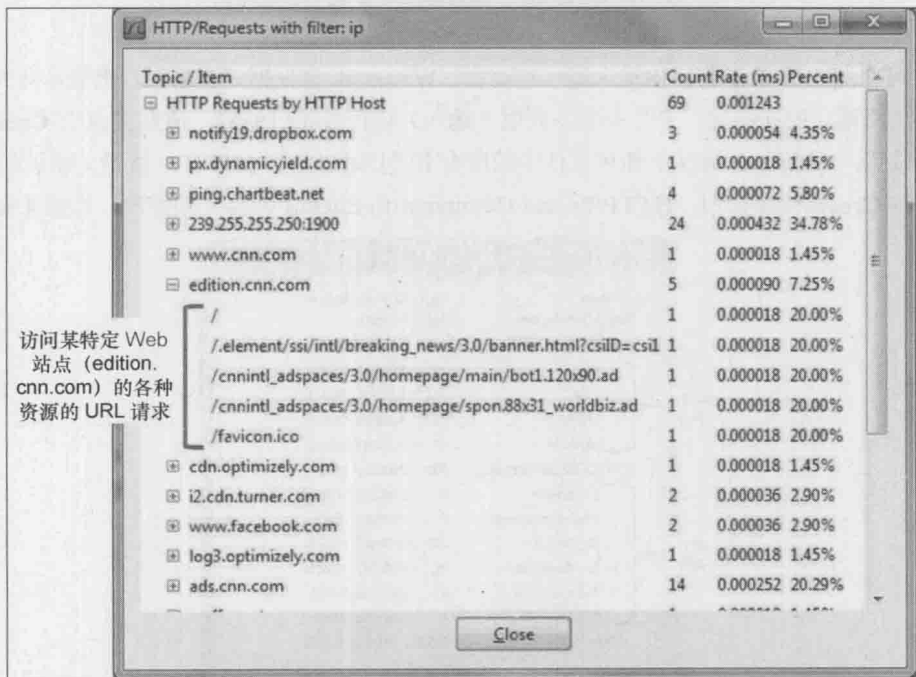


图4.18

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

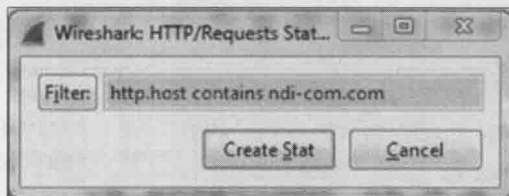


图4.19

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

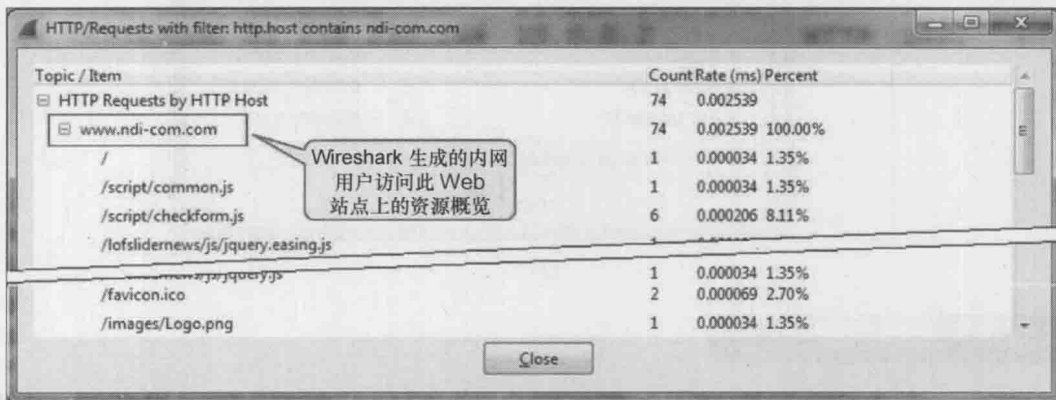


图4.20

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

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

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

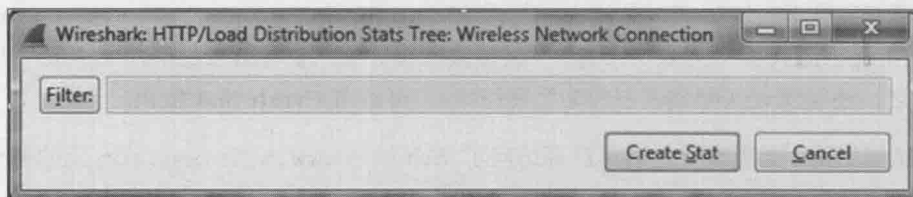


图4.21

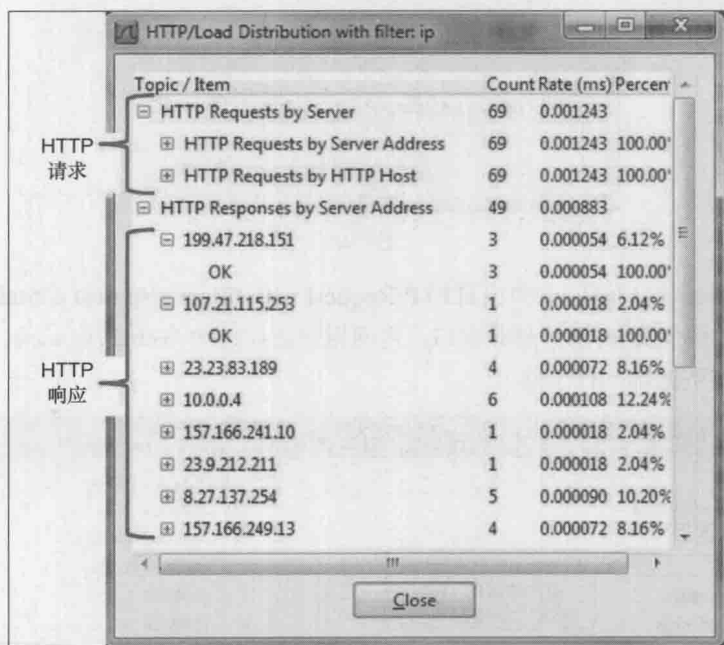


图4.22

4.6.3 幕后原理

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

4.6.4 拾遗补缺

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

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

4.7 配置 Flow Graph（数据流图），来查看 TCP 流

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

4.7.1 配置准备

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

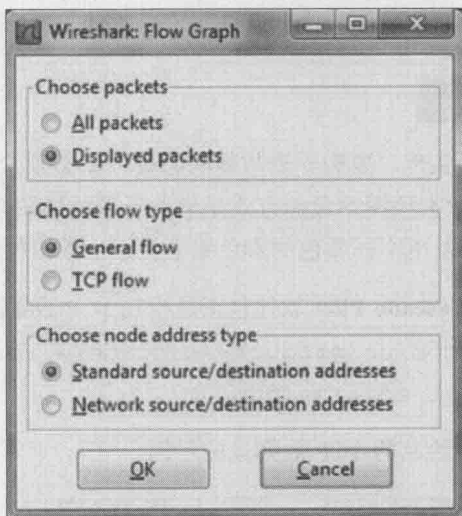


图4.23

4.7.2 配置方法

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

1. Choose packet（选择数据包）区域内的单选框。

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

2. Choose flow type（选择数据流类型）区域内的单选框。

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

4.7.3 幕后原理

只是根据抓包文件生成简单的统计信息而已，幕后原理无需赘言^①。

4.7.4 拾遗补缺

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

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

图 4.24 所示为作者手工绘制的 TCP 数据流图示例。

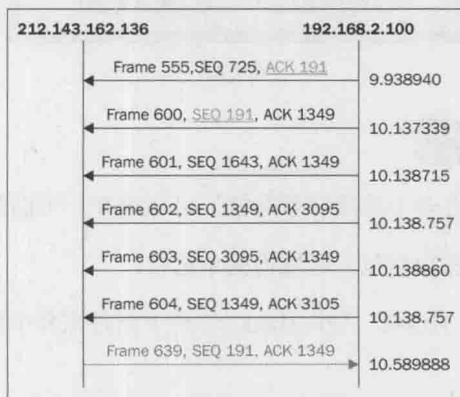


图4.24

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

万事开头难，TCP 数据流图只要多画几次，就会熟能生巧了。

4.8 生成与IP属性有关的统计信息

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

- IP Addresses;
- IP Destinations;
- IP Protocols Types。

4.8.1 准备工作

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

4.8.2 操作方法

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

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

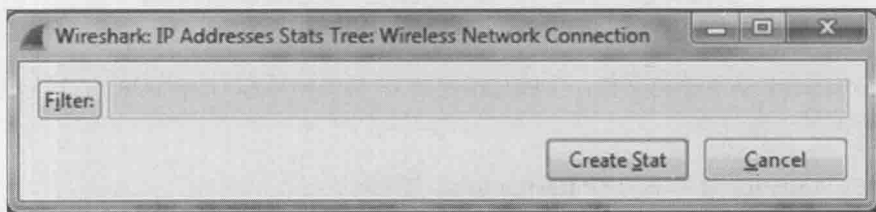


图4.25

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

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

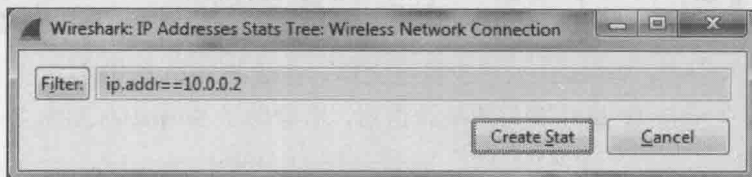


图4.26

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

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

图4.27

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

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

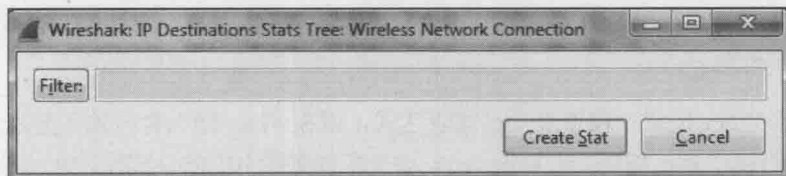


图4.28

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

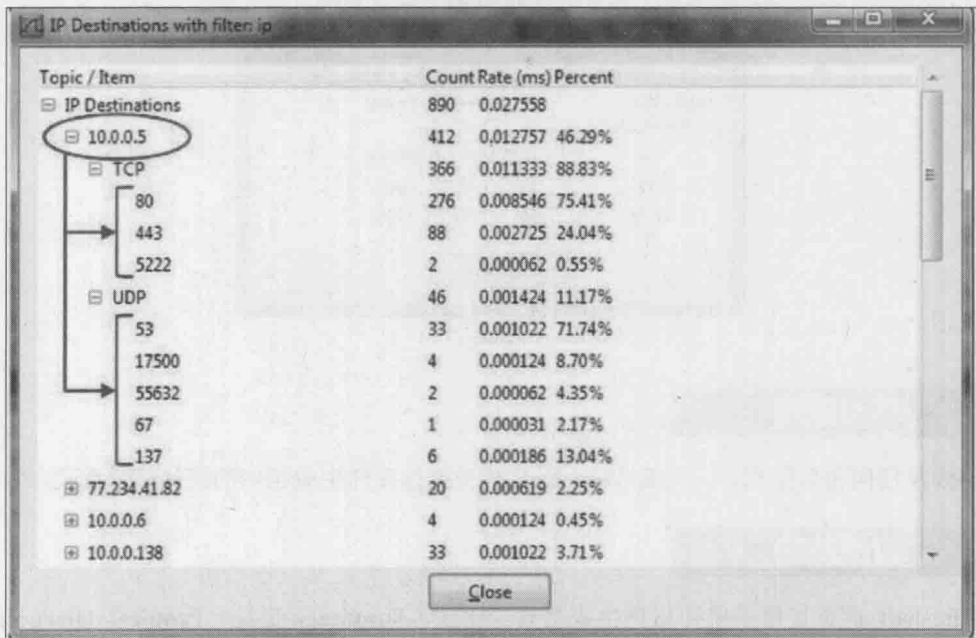


图4.29

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

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

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

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

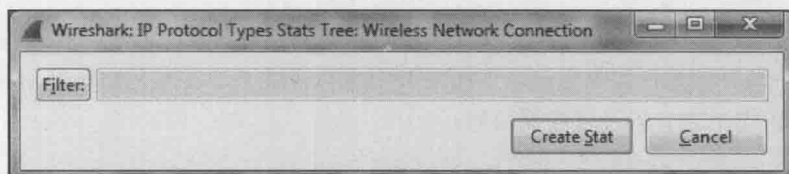


图4.30

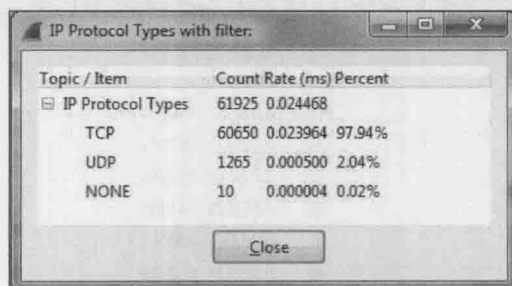


图4.31

4.8.3 幕后原理

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

4.8.4 拾遗补缺

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

第5章

高级信息统计工具的使用法

本章涵盖以下主题：

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

5.1 简介

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

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

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

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

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

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

5.2.1 配置准备

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

5.2.2 配置方法

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

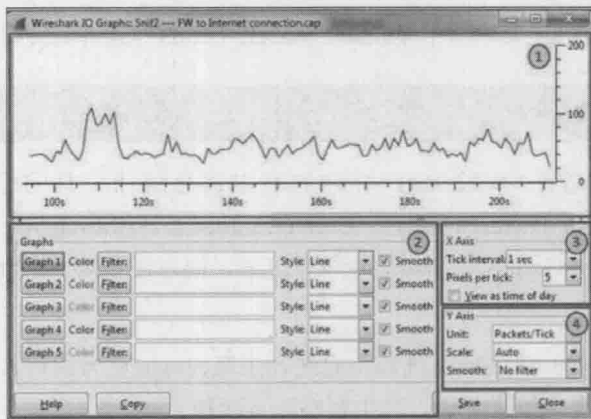


图5.1

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

显示过滤器的配置

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

X 轴（时间轴）参数的配置

X 轴参数全都在区域 3 中配置。

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



注意

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

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

Y 轴（速率轴）参数的配置

Y 轴参数全都在区域 4 中配置。

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

5.2.3 幕后原理

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

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

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

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

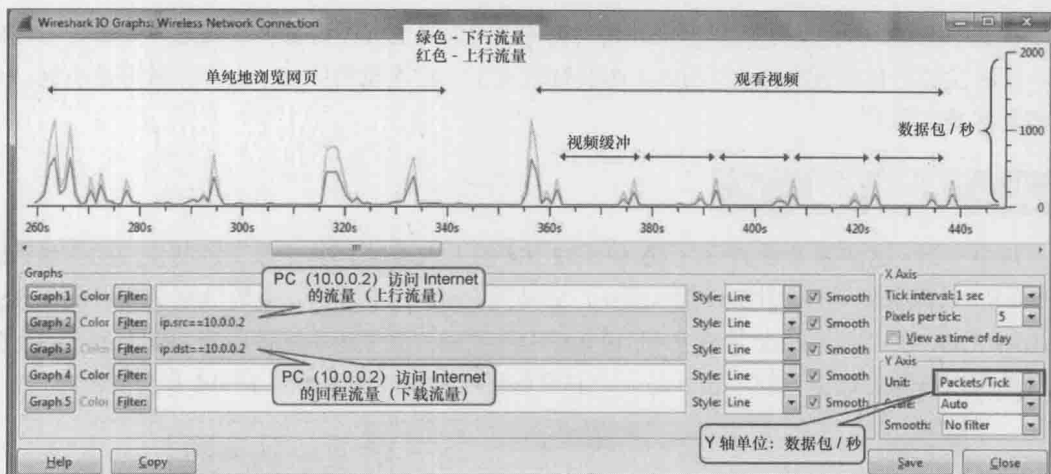


图5.2

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

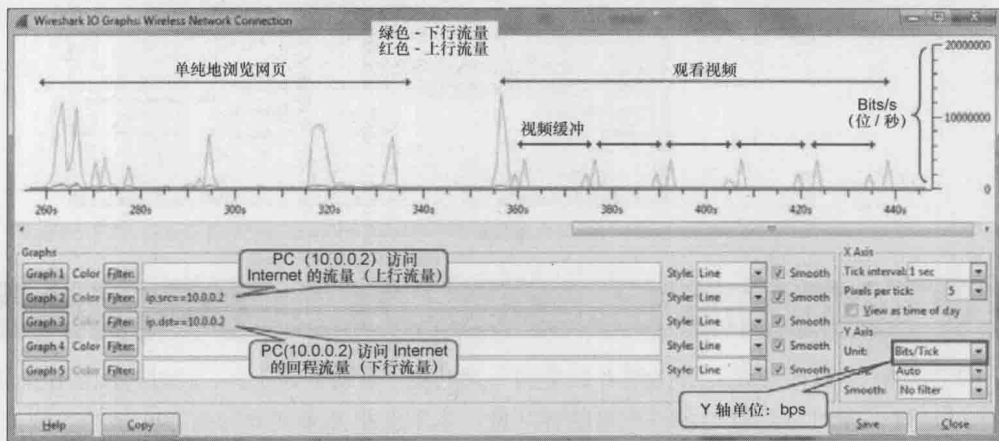


图5.3

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

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

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

5.2.4 拾遗补缺

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

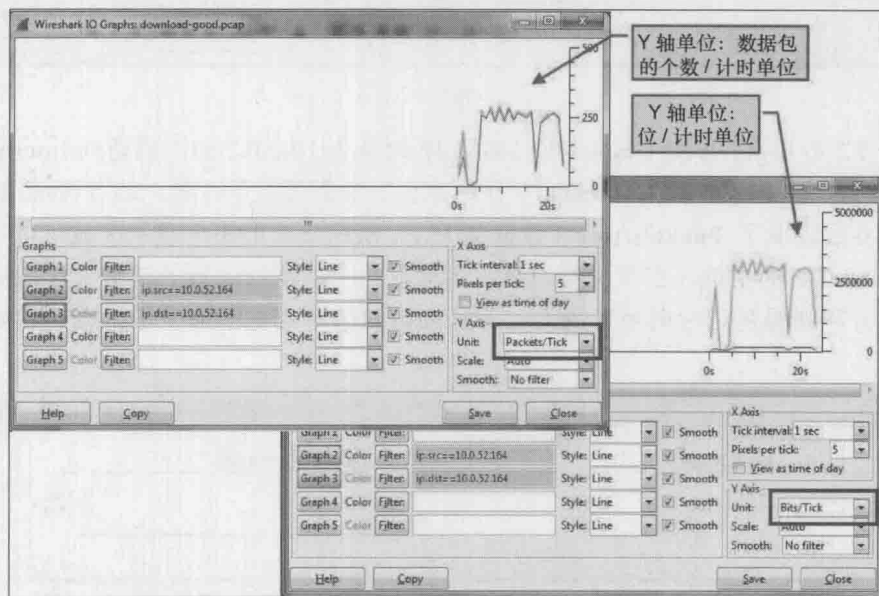


图5.4

5.3 用IO Graphs工具测算 (链路的) 吞吐量

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

5.3.1 准备工作

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

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

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

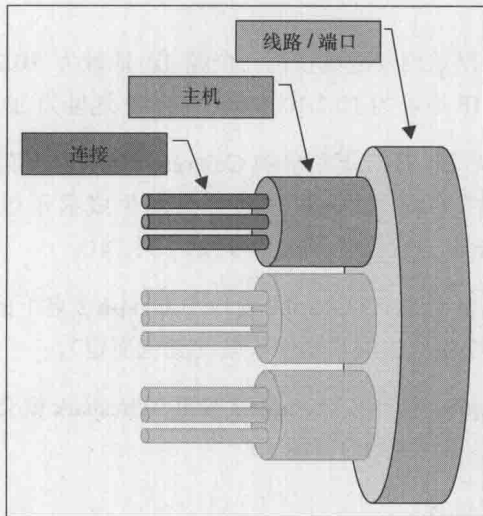


图5.5

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

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

5.3.2 测算方法

本节会给出若干测算网络流量时常用的显示过滤器^③。

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

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

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

测算端设备之间的吞吐量

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

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

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

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

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

图 5.6 显示了上述操作过程。

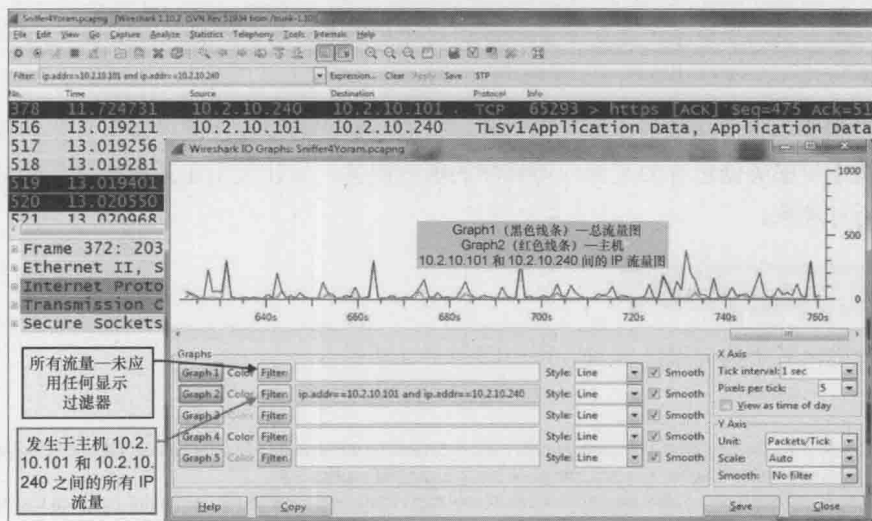


图 5.6

测算应用程序的吞吐量

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

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

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

图 5.7 显示了上述操作过程。

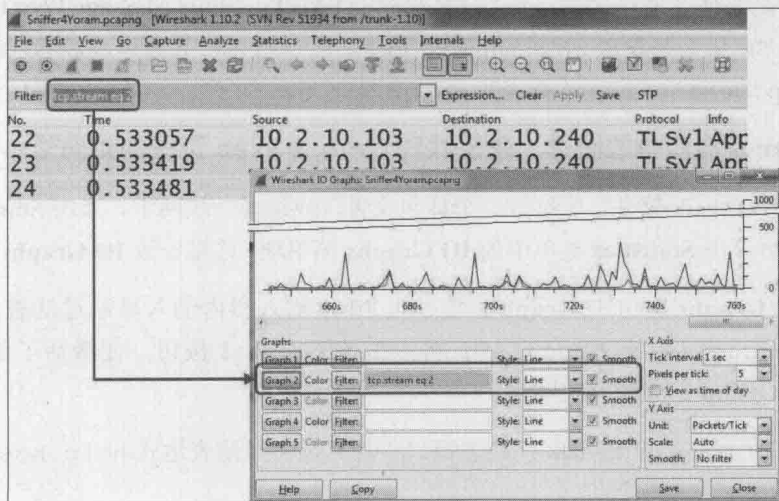


图5.7

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

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

5.3.3 幕后原理

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

5.3.4 拾遗补缺

本节会再举几个使用显示过滤器,来筛选具有某种特征的数据包的示例。

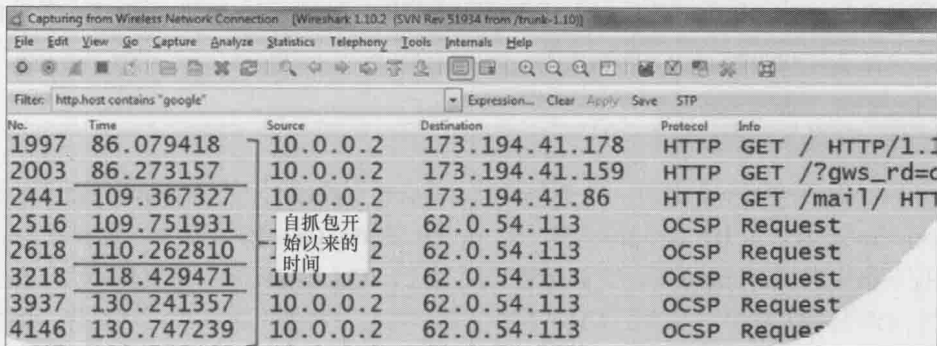
筛选由特定用户发出的 SMS 消息

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

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

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

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



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

图5.8

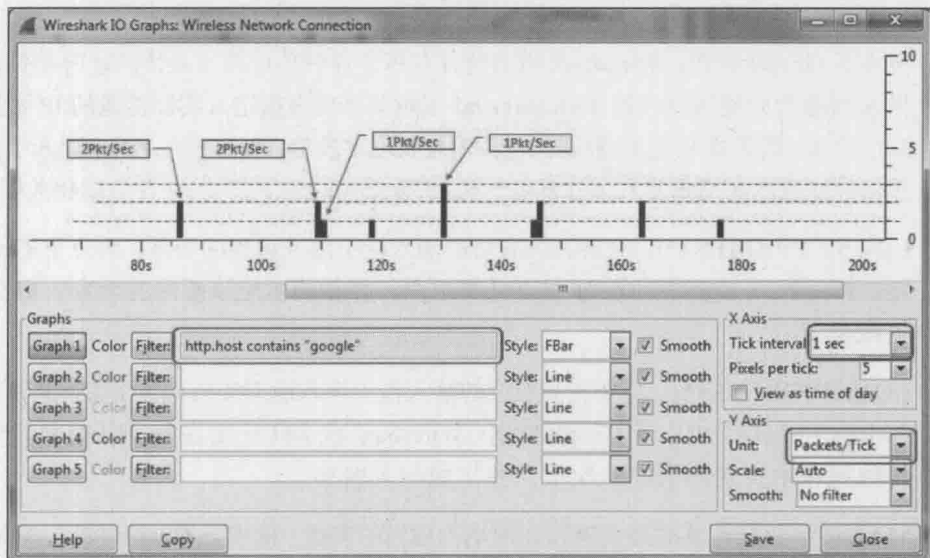


图5.9

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

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

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

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

5.4.1 配置准备

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

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

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

➤ **LOAD (*)**: 用来生成与响应时间有关的图形。

5.4.2 配置方法

要启用 **IO Graphs** 工具的高级功能, 请按以下步骤行事。

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

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

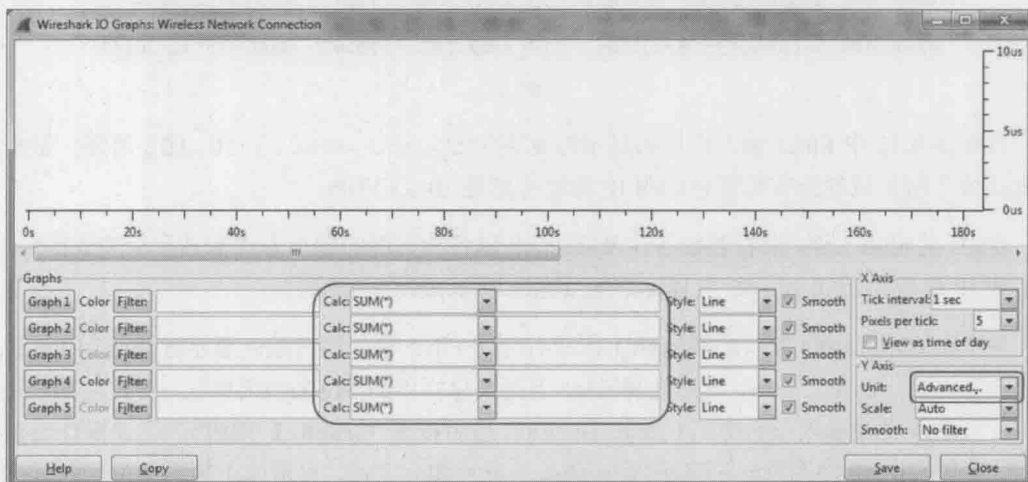


图5.10

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

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

如何生成数据帧接收 (抓取) 时间间隔统计图形

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

TCP 性能的方法。

请看图 5.11 所示的抓包文件。

No.	Time	Source	Destination	Protocol	Info
6520	116.974756	10.2.10.105	10.2.10.240	TCP	[TCP segment of
6521	116.974798	10.2.10.105	10.2.10.240	TCP	[TCP segment of
6522	116.974819	10.2.10.105	10.2.10.240	TCP	[TCP segment of
6532	116.975254	10.2.10.105	10.2.10.240	TLSv1	Application Data
6540	117.015011	数据包的	10.2.10.240	TCP	https > 65277 [A
6542	117.015590	抓取时间	10.2.10.240	TCP	https > 65277 [A
7031	130.915192	10.2.10.105	10.2.10.240	TCP	https > 65295 [A
7033	130.915744	10.2.10.105	10.2.10.240	TCP	https > 65295 [A
7034	130.919789	10.2.10.105	10.2.10.240	TLSv1	Application Data
7035	130.919834	10.2.10.105	10.2.10.240	TLSv1	Application Data

图5.11

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

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

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

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

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

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

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

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

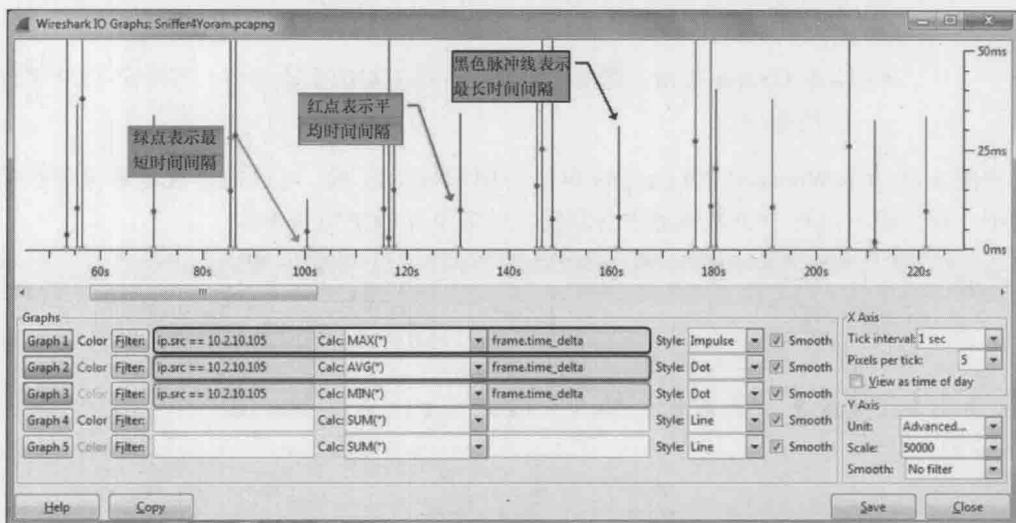


图5.12

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

如何获悉某股 TCP 数据流中发生了多少次 TCP 重传事件

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

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

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

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

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

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

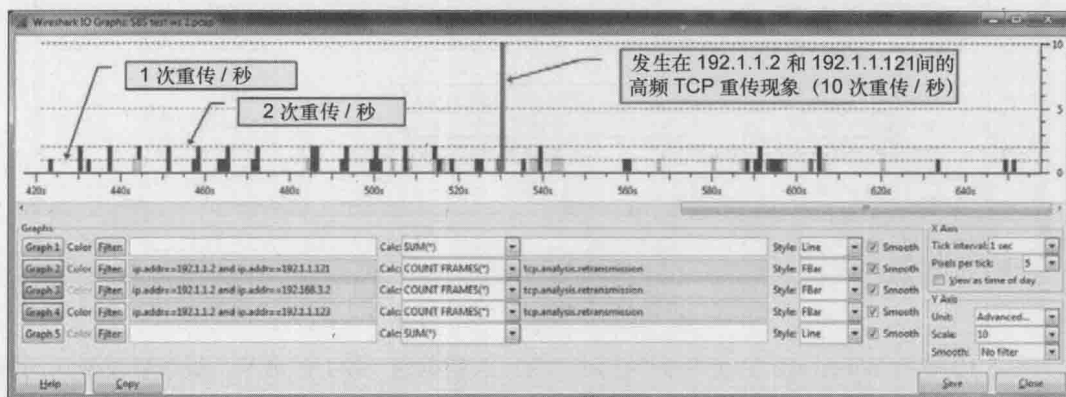


图5.13

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

如何获悉某股 TCP 数据流中数据包的抖动情况^①

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

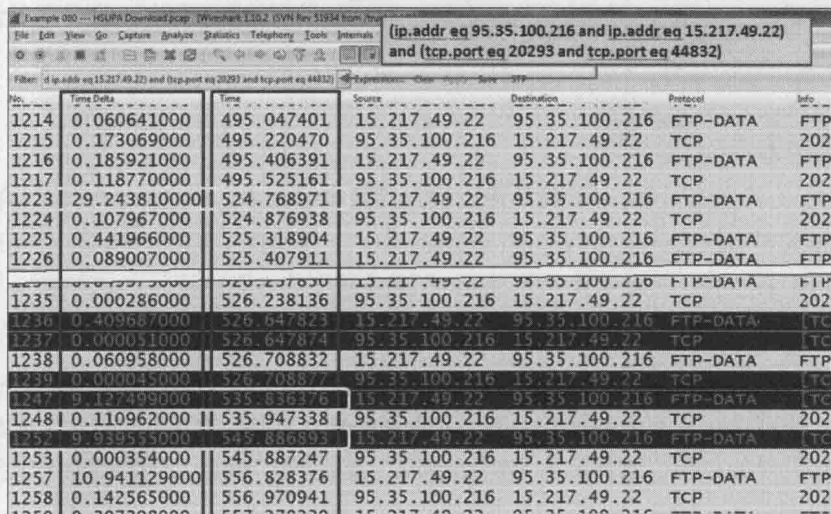
- 启动 Wireshark 软件，先打开一个抓包文件（或选择一块网卡，点击 **Start** 按钮开始抓包），在抓包主窗口中任选一个隶属于某股有待观察的 TCP 数据流的 TCP 数据包。然

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

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

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

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



The screenshot shows the Wireshark interface with a packet list table. The table has columns for No., Time Delta, Time, Source, Destination, Protocol, and Info. The filter bar at the top shows the expression: (ip.addr eq 95.35.100.216 and ip.addr eq 15.217.49.22) and (tcp.port eq 20293 and tcp.port eq 44832).

No.	Time Delta	Time	Source	Destination	Protocol	Info
1214	0.060641000	495.047401	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1215	0.173069000	495.220470	95.35.100.216	15.217.49.22	TCP	202
1216	0.185921000	495.406391	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1217	0.118770000	495.525161	95.35.100.216	15.217.49.22	TCP	202
1223	29.243810000	524.768971	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1224	0.107967000	524.876938	95.35.100.216	15.217.49.22	TCP	202
1225	0.441966000	525.318904	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1226	0.089007000	525.407911	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1235	0.000286000	526.238136	95.35.100.216	15.217.49.22	TCP	202
1236	0.409687000	526.647823	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1237	0.000051000	526.647874	95.35.100.216	15.217.49.22	TCP	202
1238	0.060958000	526.708832	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1239	0.000045000	526.708877	95.35.100.216	15.217.49.22	TCP	202
1247	9.127499000	535.836176	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1248	0.110962000	535.947338	95.35.100.216	15.217.49.22	TCP	202
1252	9.939555000	545.886893	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1253	0.000354000	545.887247	95.35.100.216	15.217.49.22	TCP	202
1257	10.941129000	556.828376	15.217.49.22	95.35.100.216	FTP-DATA	FTP
1258	0.142565000	556.970941	95.35.100.216	15.217.49.22	TCP	202

图5.14

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

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

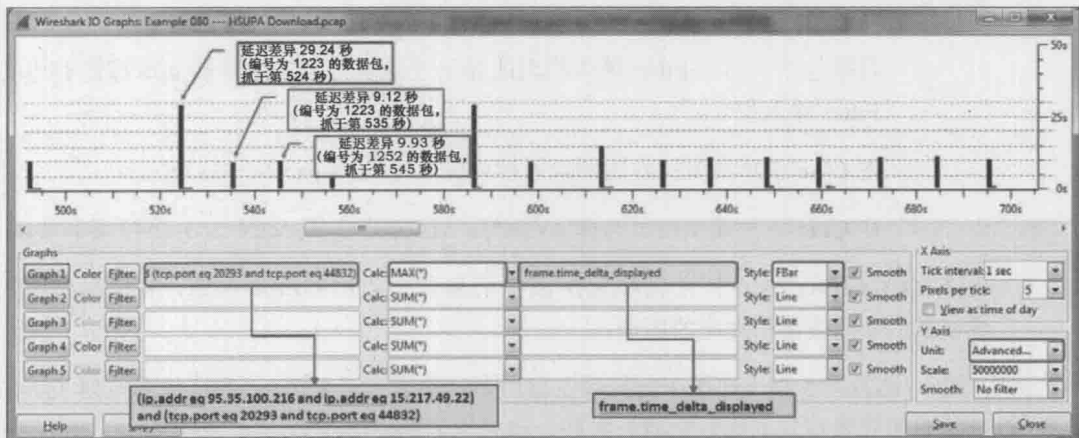


图5.15

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

5.4.3 幕后原理

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

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

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

5.4.4 拾遗补缺

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

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

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

5.5.1 使用准备

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

5.5.2 使用方法

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

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



注意

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

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

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

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

反映出了一次常规的文件下载行为。

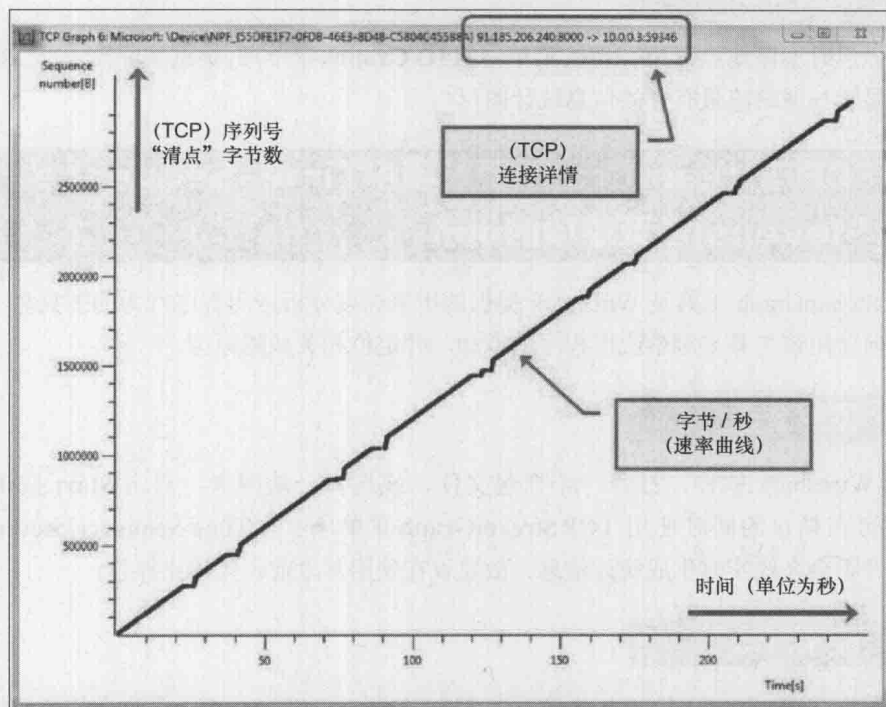


图5.16

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

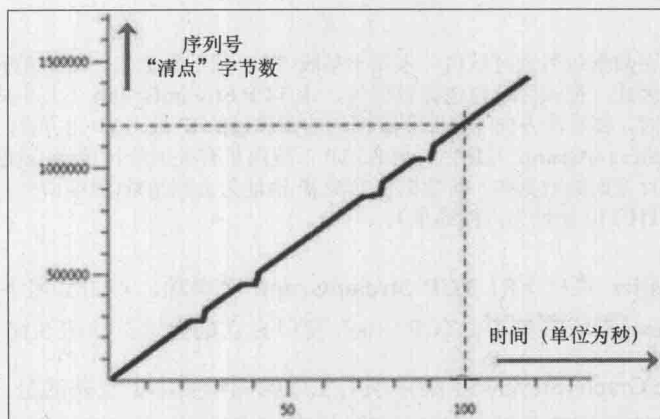


图5.17

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

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

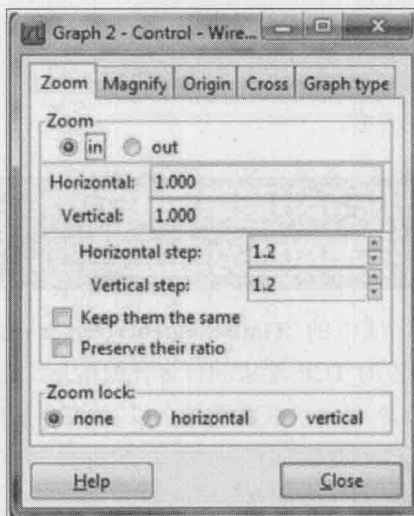


图 5.18

5.5.3 幕后原理

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

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

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

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

5.5.4 拾遗补缺

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

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

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

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

5.6.1 使用准备

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

5.6.2 使用方法

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

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

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



注意

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

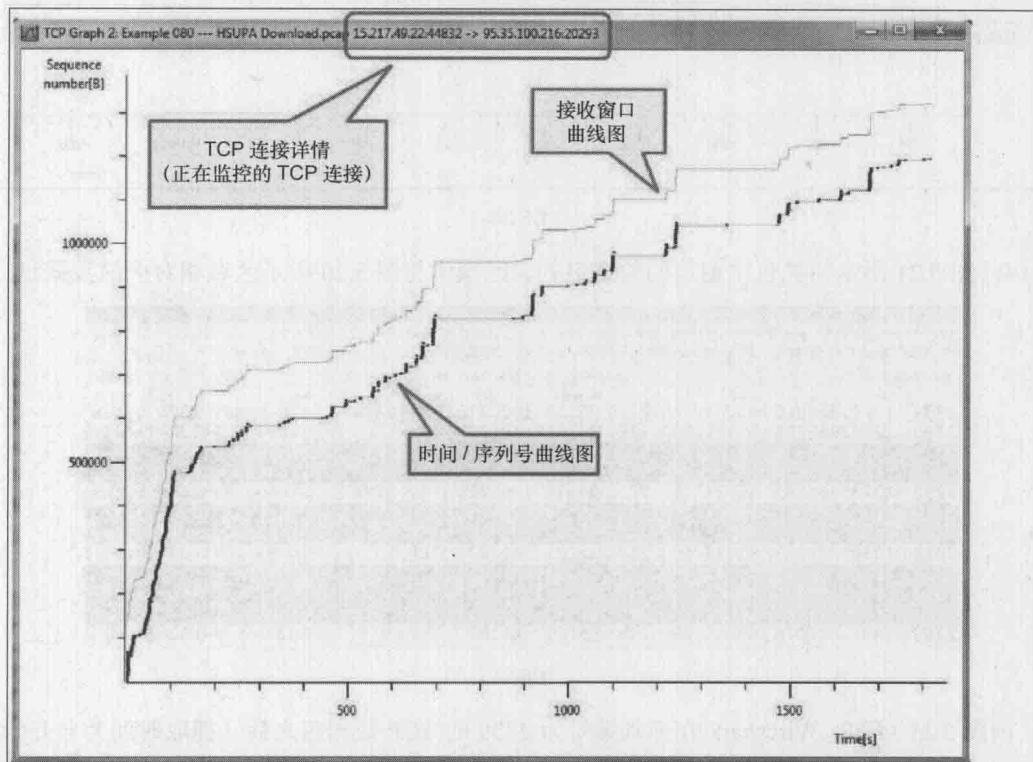


图5.19

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

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

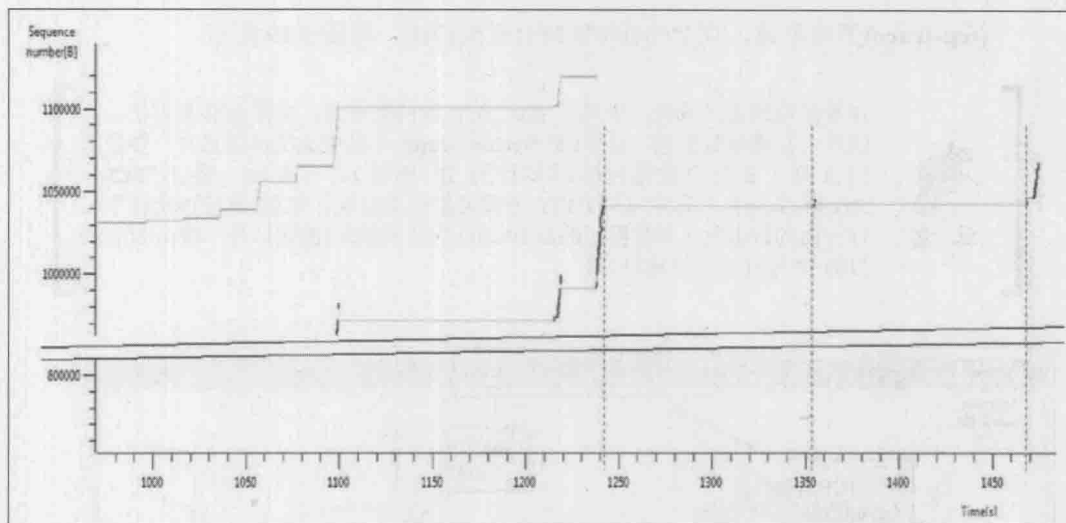


图5.20

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

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

图5.21

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

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

5.6.3 幕后原理

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

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

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

5.6.4 拾遗补缺

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

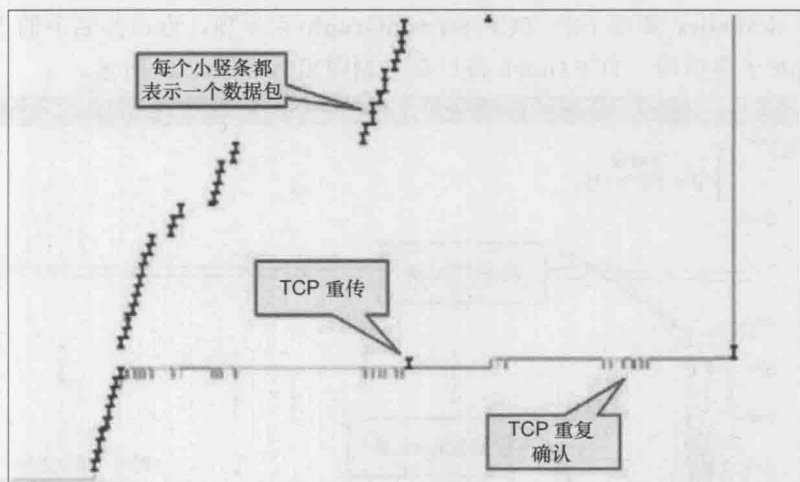


图5.22

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

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

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

5.7.1 使用准备

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

5.7.2 使用方法

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

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

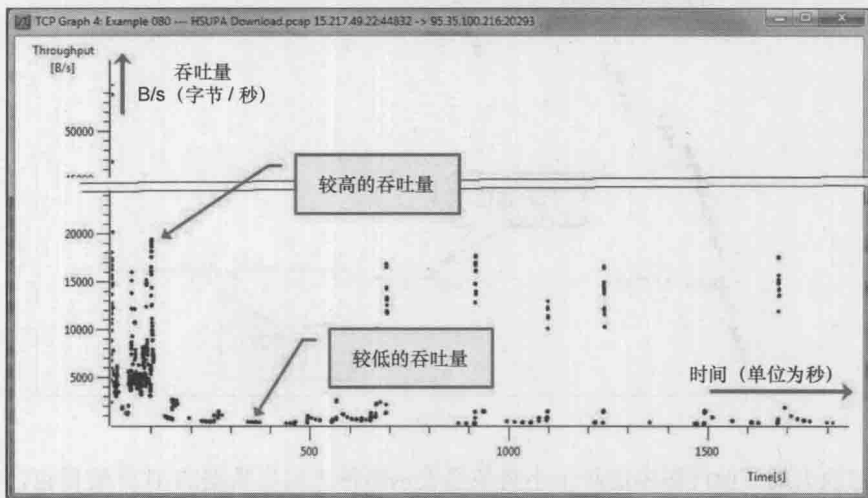


图5.23

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

5.7.3 幕后原理

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

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

5.7.4 拾遗补缺

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

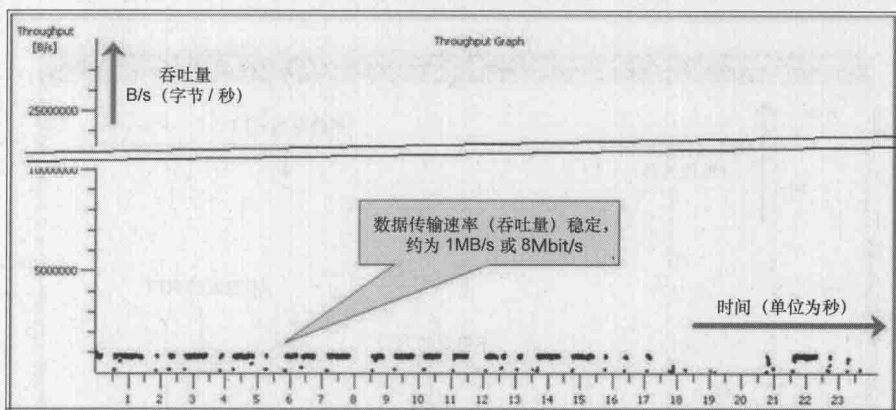


图5.24

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

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

5.8.1 使用准备

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

5.8.2 使用方法

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

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

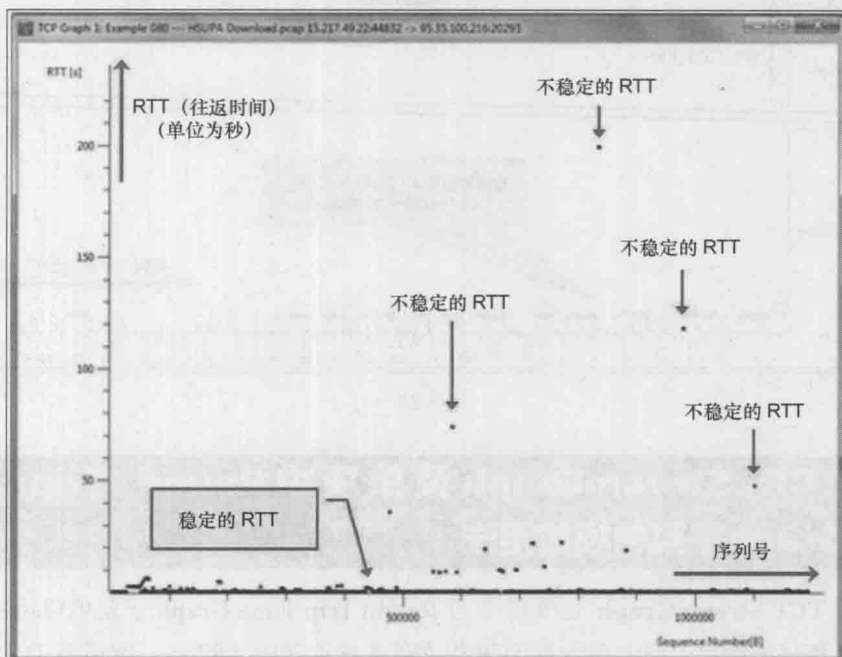


图5.25

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

5.8.3 幕后原理

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

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

5.8.4 拾遗补缺

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

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

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

5.9.1 使用准备

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

5.9.2 使用方法

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

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

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

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

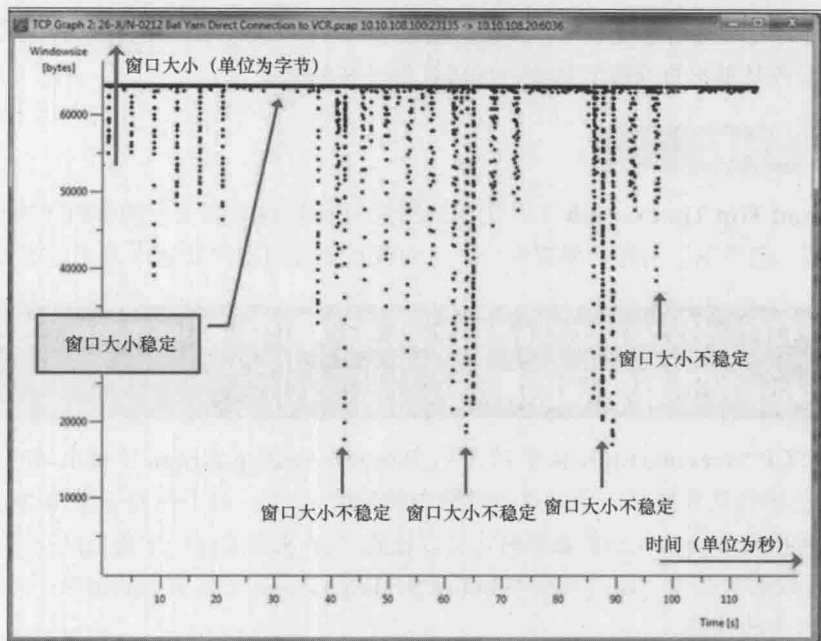


图5.26

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

5.9.3 幕后原理

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

5.9.4 拾遗补缺

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

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

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

第6章

Expert Info工具的法

本章涵盖以下主题：

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

6.1 简介

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

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

6.2 如何使用Expert Info工具执行排障任务

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

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

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

6.2.1 使用准备

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

6.2.2 使用方法

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

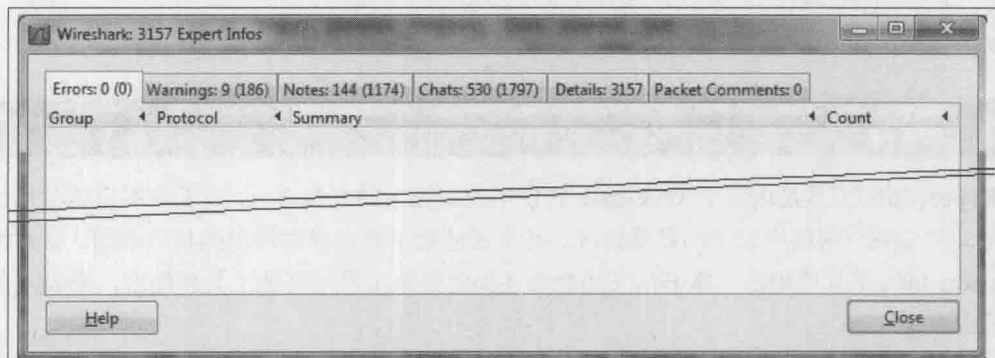


图6.1

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



在每个标签栏的事件名称的右边都有一个数字，用来表示该类事件有多少子类事件。

下面是对 Expert Infos 窗口中各标签栏的介绍。

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

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

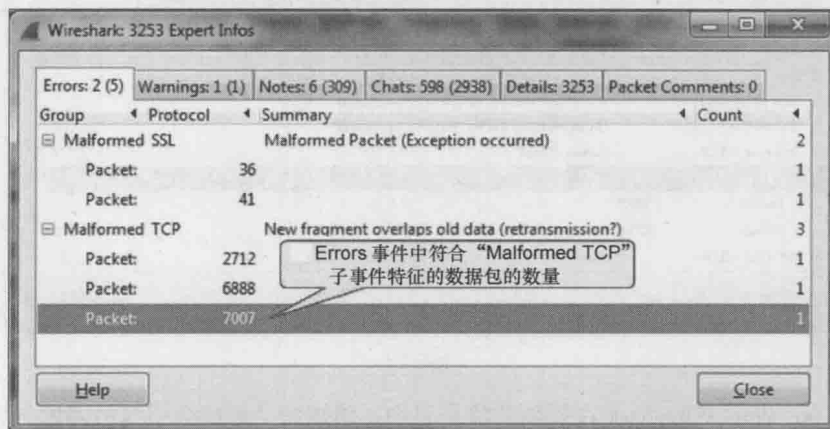


图6.2

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

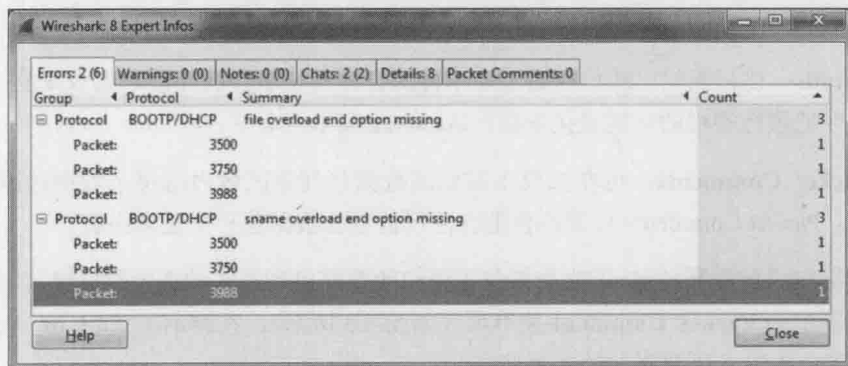


图6.3

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

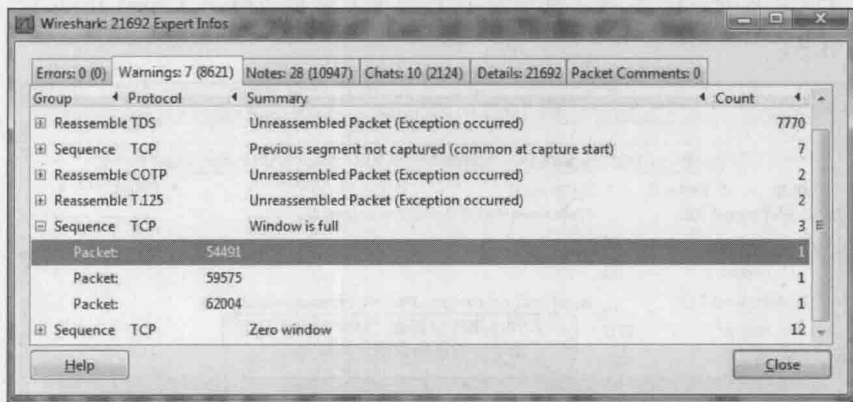


图6.4

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

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

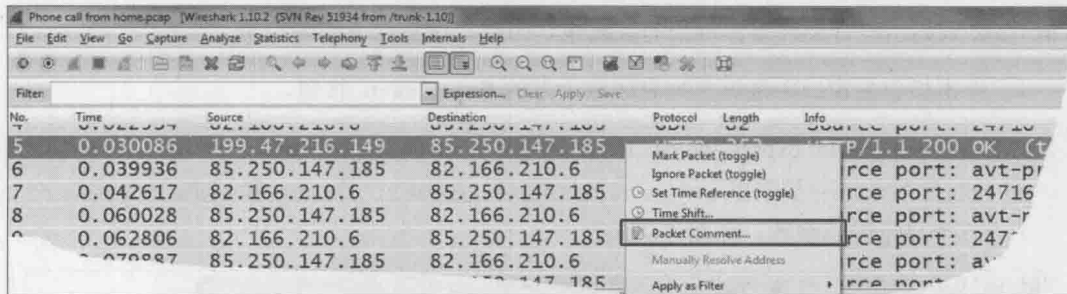


图6.5

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



注意

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

6.2.3 幕后原理

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

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



注意

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

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

6.2.4 拾遗补缺

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

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

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

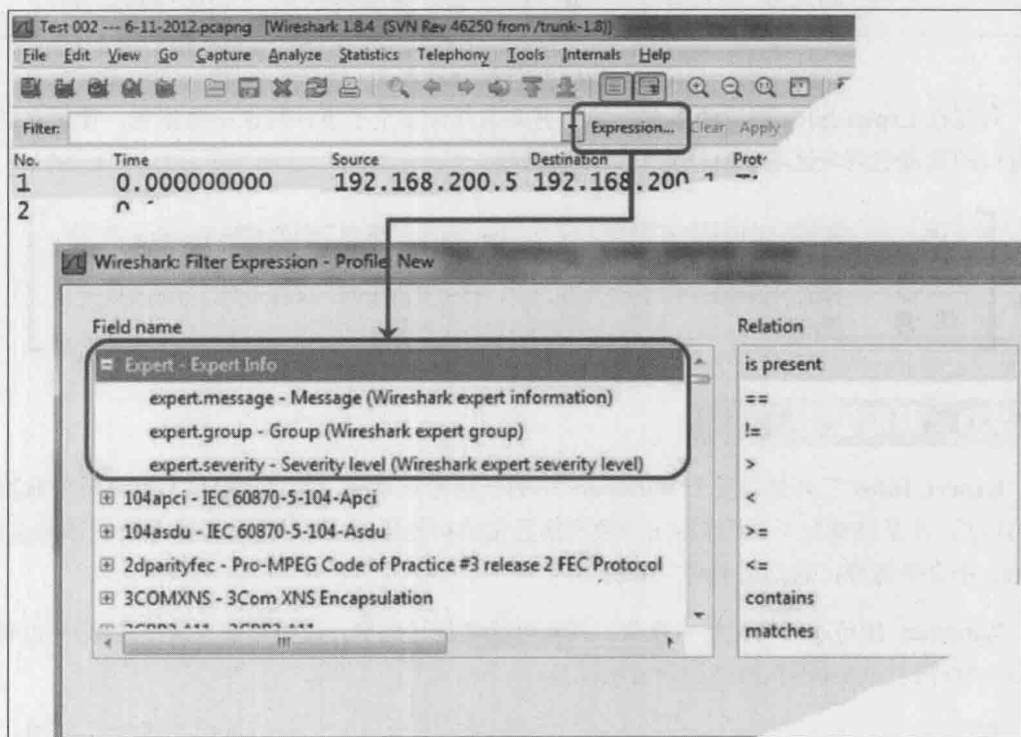


图6.6

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

下面是对 `expert.group` 参数名下各类专家消息的介绍。

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

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

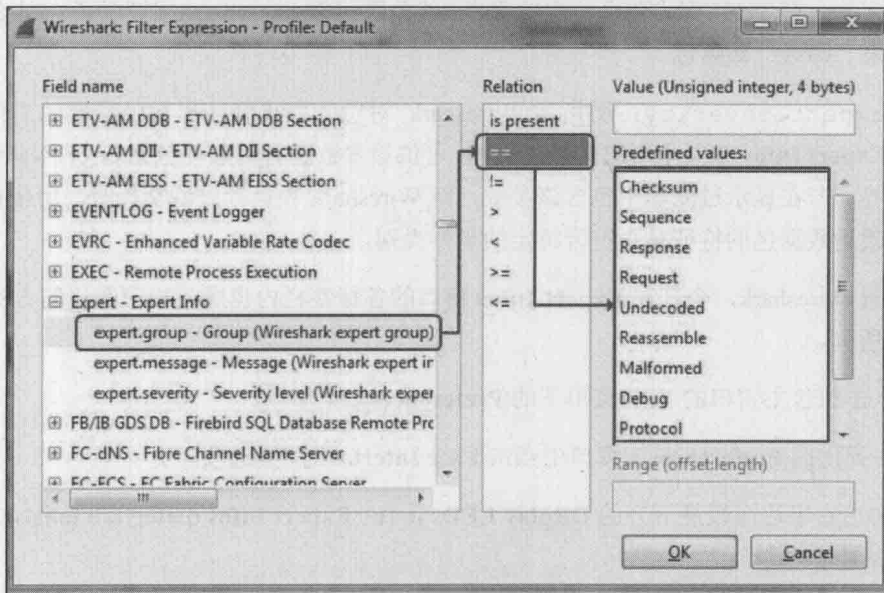


图6.7

- ▣ **Request** (请求类): 匹配供某种应用程序用来执行相关请求功能的数据包。
- ▣ **Undecoded** (不能识别类): 匹配 Wireshark 协议解析器不能识别 (无法解码) 的数据包。
- ▣ **Reassemble** (重组问题类): 匹配在重组 (经过分片的数据包) 时发生问题的数据包 (一般都是指遗失了某个数据包的某一片)。
- ▣ **Malformed** (畸形数据包类): 匹配畸形数据包, 或因 Wireshark 协议解析器故障而终止解析的数据包。
- ▣ **Debug**: 在 Wireshark 发布版中不应出现匹配此类特征的数据包。
- ▣ **Protocol** (协议类): 匹配违背协议规范的数据包 (比如, 数据包中的字段值有违标准, 或缺少了某些原本应该存在的字段), Wireshark 协议解析器有可能会继续解析此类数据包。
- ▣ **Comment** (含注释信息类): 匹配包含注释信息的数据包。至于如何给数据包添加注释, 前文已做过详细介绍, 在此不加赘述。

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

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

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

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

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

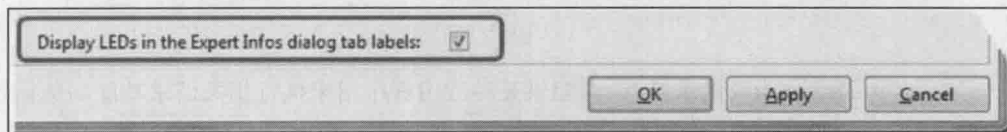


图6.8

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

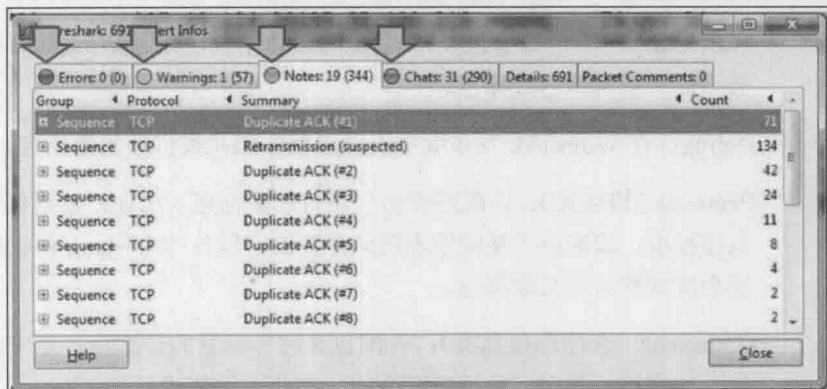


图6.9

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

6.2.5 进阶阅读

- 第9章。

6.3 认识Errors事件

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

6.3.1 准备工作

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

6.3.2 操作方法

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

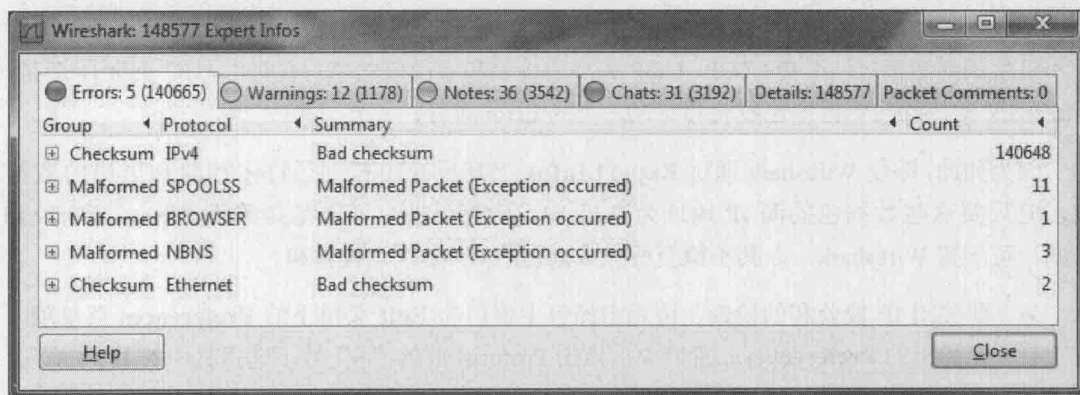


图6.10

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

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

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

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

6.3.3 幕后原理

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

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

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

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

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

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

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

6.3.4 拾遗补缺

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



注意

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

6.3.5 进阶阅读

➤ 第9章。

6.4 认识Warnings事件

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

6.4.1 准备工作

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

6.4.2 操作方法

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

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

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

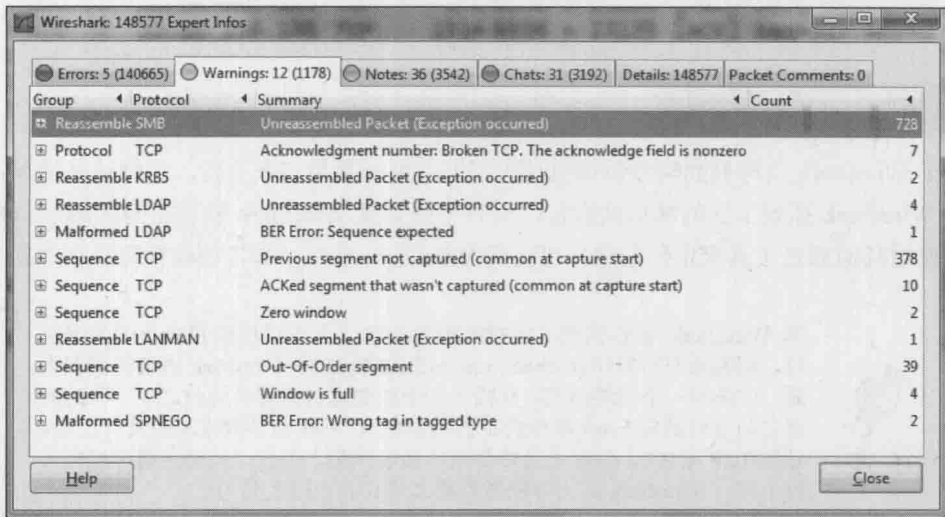


图6.11

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

6.4.3 幕后原理

Wireshark 能感知到其所抓数据包的种种特征，比如：

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

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

6.4.4 拾遗补缺

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

6.4.5 进阶阅读

➤ 第 9 章。

6.5 认识 Notes 事件

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

6.5.1 准备工作

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

6.5.2 操作方法

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

下面是对图 6.12 所示 Notes 事件的几种子事件的简单介绍。

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

预示着路由问题。

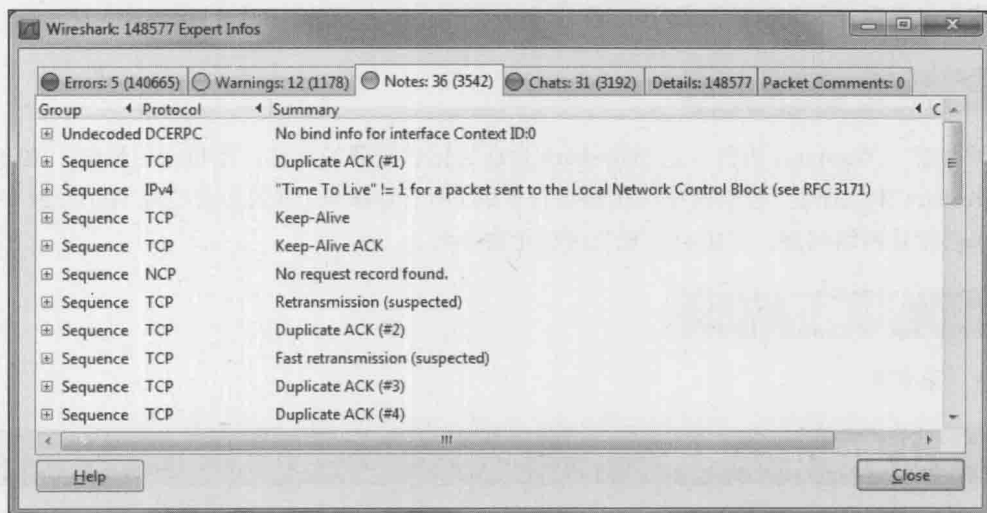


图6.12



注意

第9章到第12章会讨论 Notes 事件名下的其他几类子事件。

6.5.3 幕后原理

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

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

6.5.4 拾遗补缺

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

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

6.5.5 进阶阅读

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

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

第7章

Ethernet、LAN交换及无线LAN

本章涵盖以下主题：

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

7.1 简介

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

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

7.2 发现广播及错包风暴

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

本节会介绍几个发现、分类以及解决此类故障的秘诀。



注意

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

7.2.1 准备工作

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

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



注意

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

7.2.2 操作方法

要查明导致广播（或错包）风暴的原因，请按以下步骤行事。

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

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

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



注意

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

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

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



注意

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

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

生成树问题

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

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

某台设备（主机）触发了广播风暴

当广播风暴是由某台设备（主机）所引起时，通常具有以下典型特征。

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

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

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

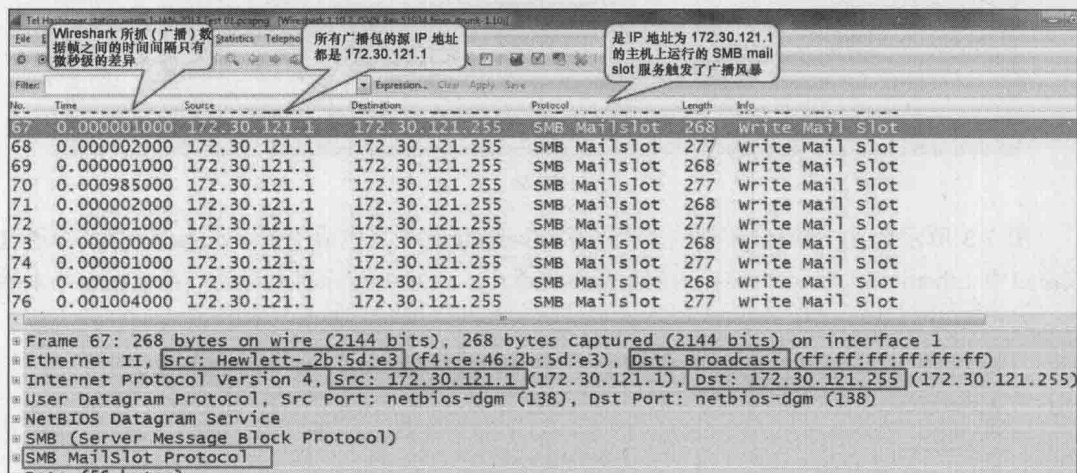


图 7.1

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

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

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

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

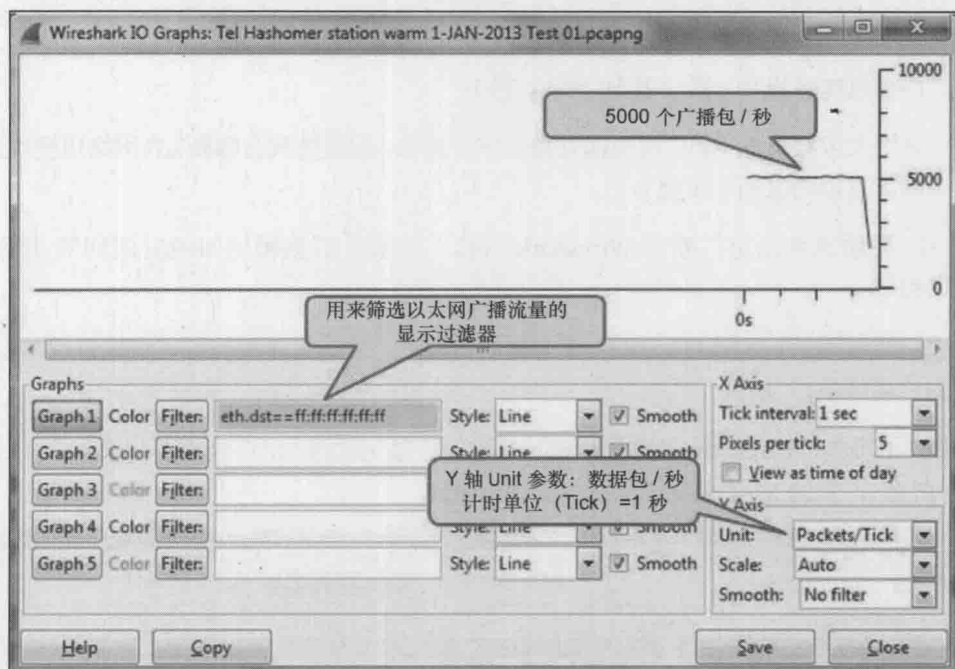


图 7.2

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

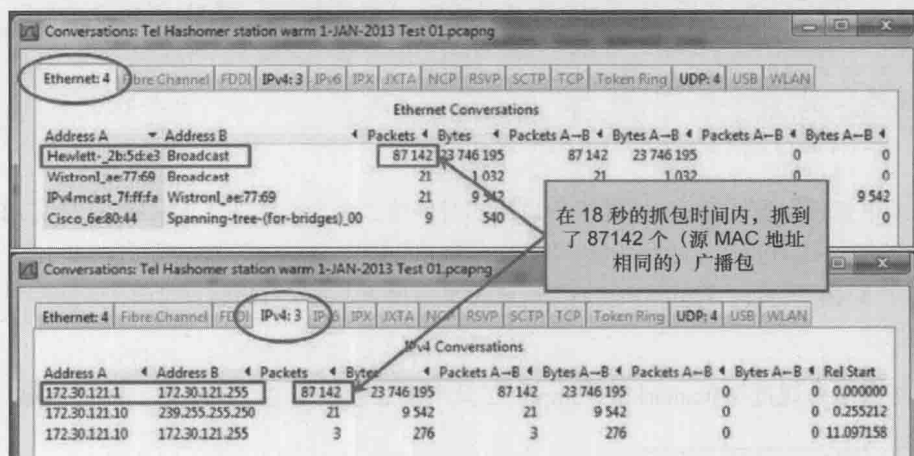
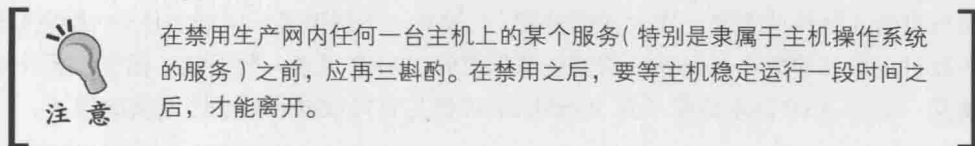


图 7.3

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



有时间规律可循的广播风暴

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

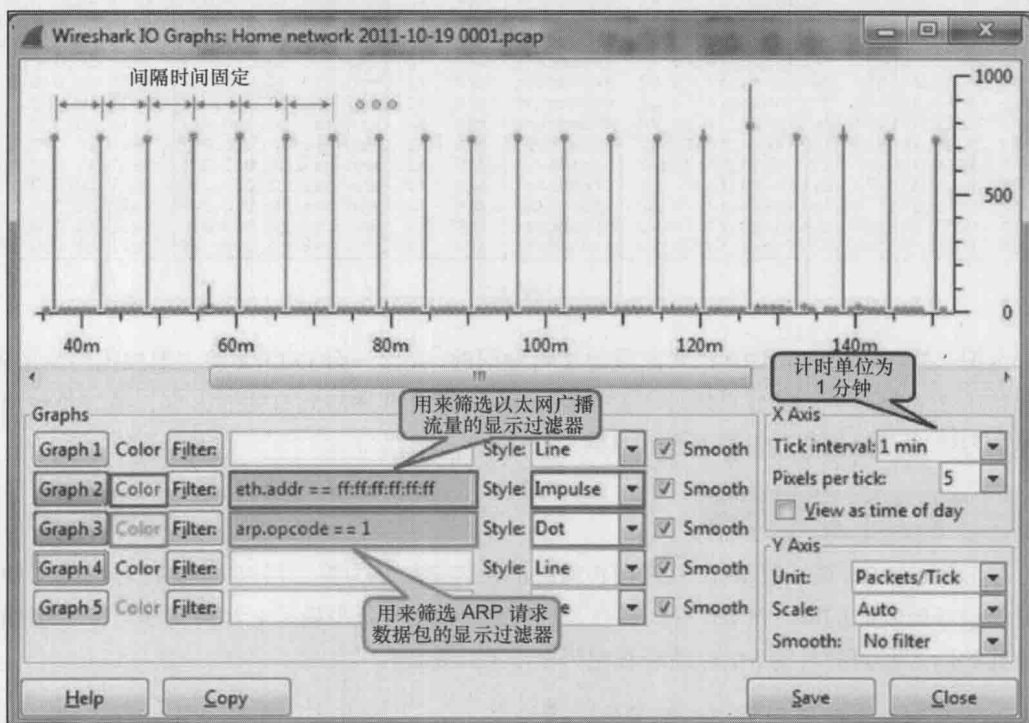


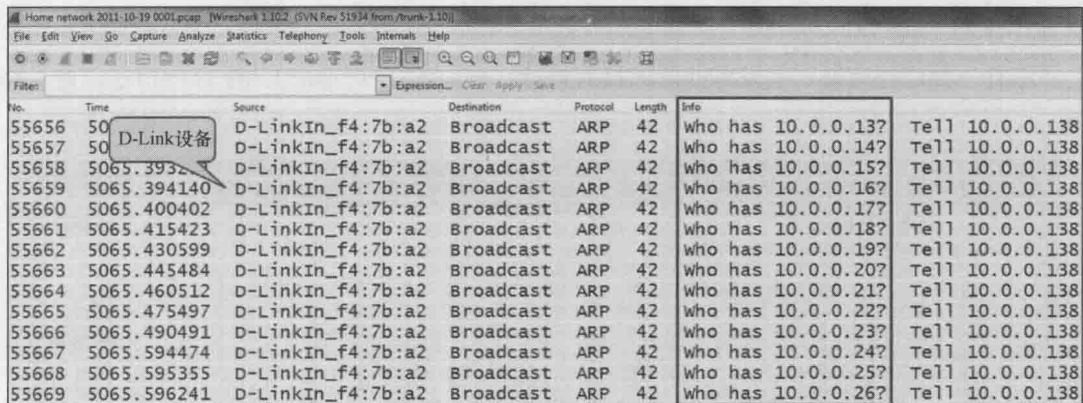
图 7.4

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

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

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

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



No.	Time	Source	Destination	Protocol	Length	Info
55656	50.0000000	D-Link设备	Broadcast	ARP	42	Who has 10.0.0.138? Tell 10.0.0.138
55657	50.0000000	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.142? Tell 10.0.0.138
55658	5065.3932000	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.152? Tell 10.0.0.138
55659	5065.3941400	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.162? Tell 10.0.0.138
55660	5065.4004020	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.172? Tell 10.0.0.138
55661	5065.4154230	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.182? Tell 10.0.0.138
55662	5065.4305990	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.192? Tell 10.0.0.138
55663	5065.4454840	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.202? Tell 10.0.0.138
55664	5065.4605120	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.212? Tell 10.0.0.138
55665	5065.4754970	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.222? Tell 10.0.0.138
55666	5065.4904910	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.232? Tell 10.0.0.138
55667	5065.5944740	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.242? Tell 10.0.0.138
55668	5065.5953550	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.252? Tell 10.0.0.138
55669	5065.5962410	D-LinkIn_f4:7b:a2	Broadcast	ARP	42	Who has 10.0.0.262? Tell 10.0.0.138

图 7.5

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

7.2.3 幕后原理

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

以下所列为几种在 IP 网络中常见的广播包。

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

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

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

7.2.4 拾遗补缺

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

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

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

7.2.5 进阶阅读

- 第 8 章。

7.3 生成树协议分析

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

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

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

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

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

7.3.1 准备工作

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

本节的主旨是如何利用 Wireshark 来协助排除 STP 故障。

请打开笔记本，启动 Wireshark，开始在 LAN 里抓包吧。

7.3.2 分析方法

要解决 STP 故障，先得回答网络中与 STP 有关的以下两个问题。

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

网络中运行的是哪个版本的 STP

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

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



注意

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

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

发生过网络拓扑变更事件吗？

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

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

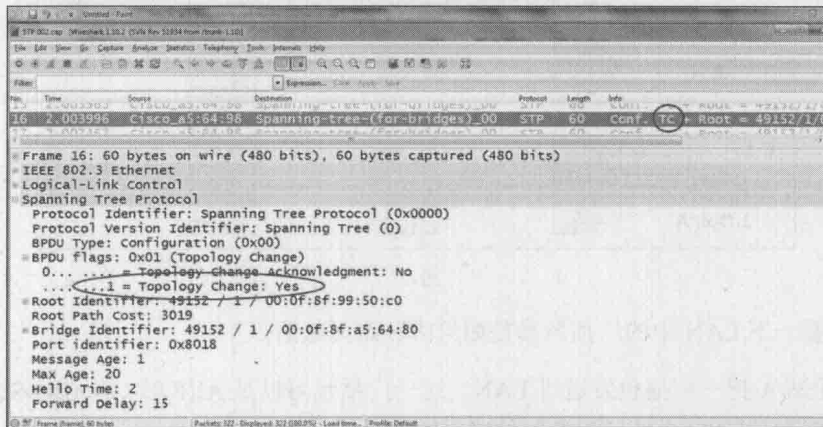


图 7.6

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



注意

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

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

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

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

7.3.3 幕后原理

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

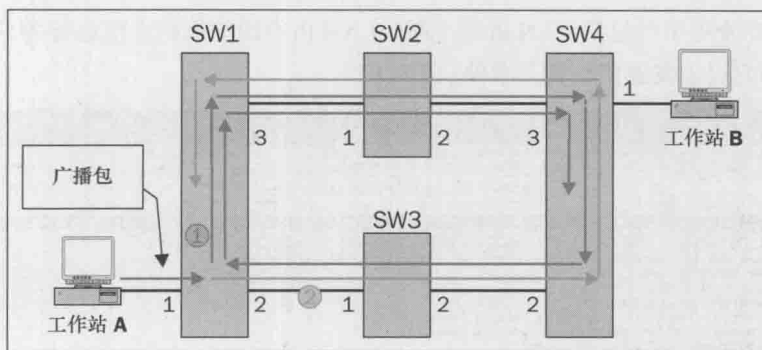


图7.7

现在来看一下 LAN 中的广播风暴是如何因环路而起的。

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

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

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

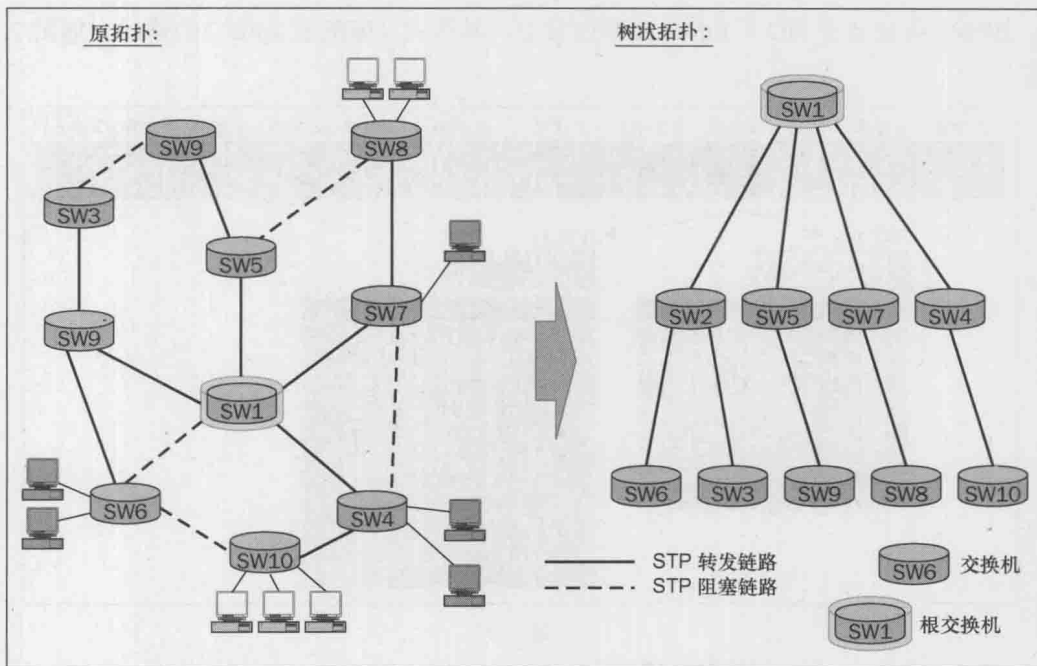


图7.8

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。

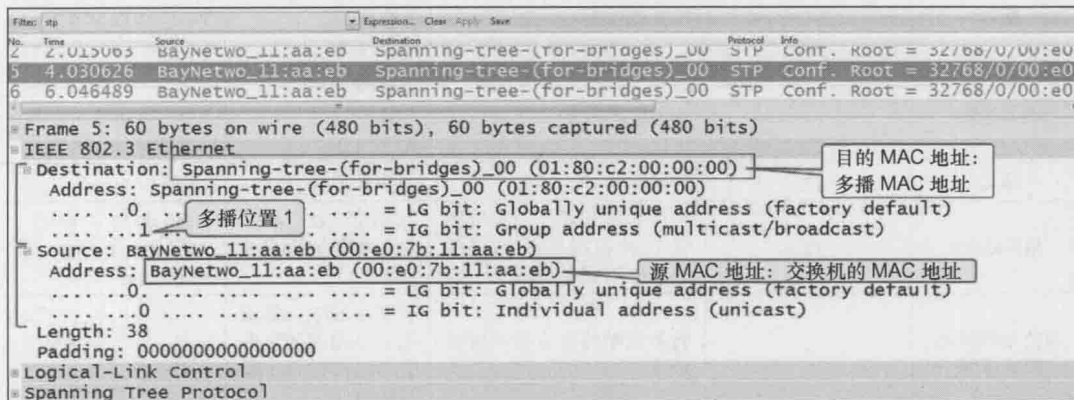


图7.9

^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。

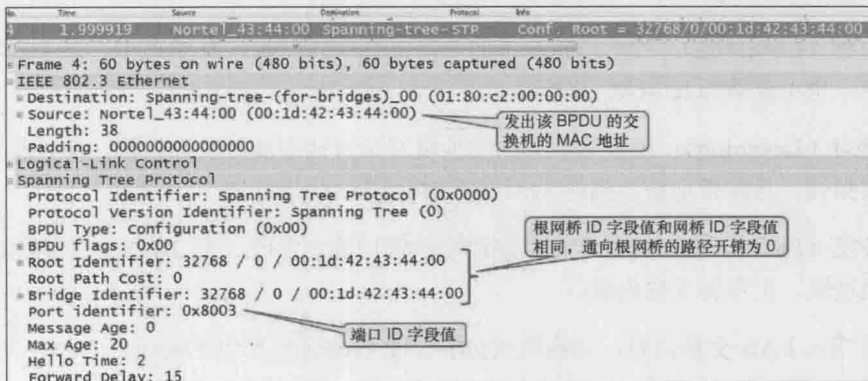


图 7.11

- 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。

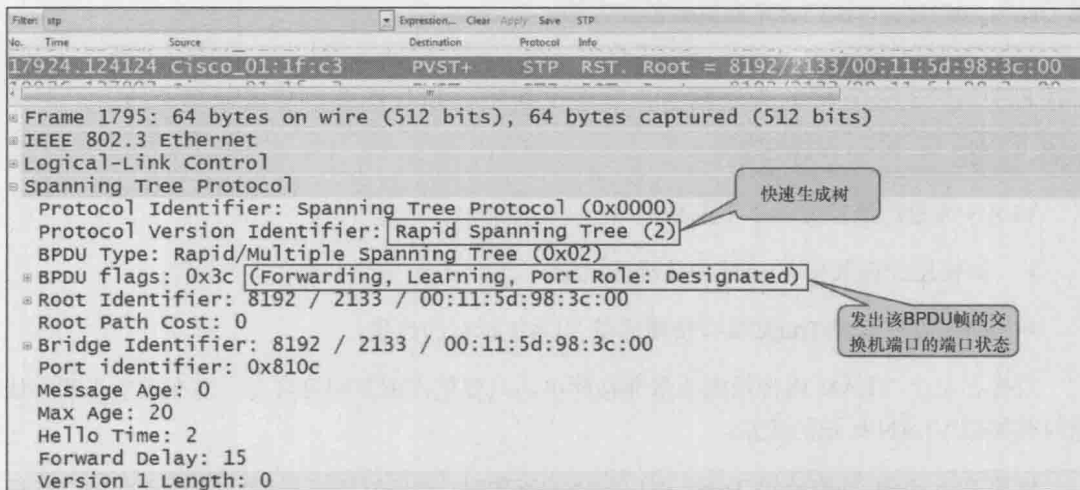


图 7.12

- 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。

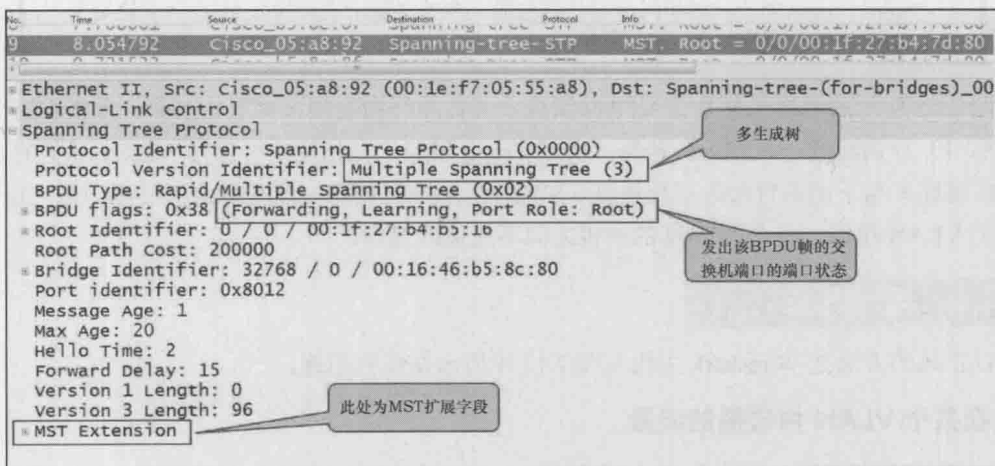


图 7.13

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。

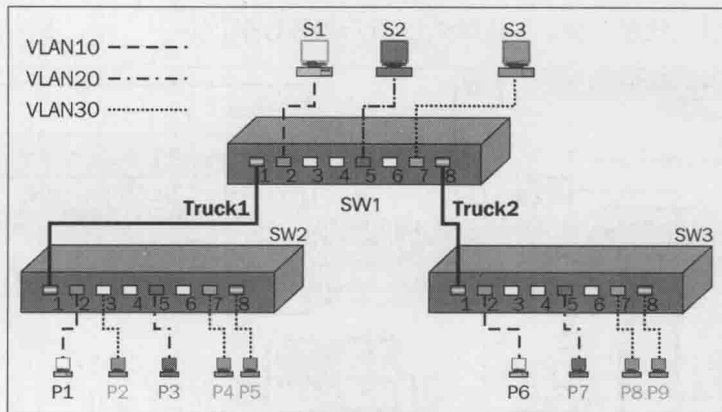


图7.14

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。

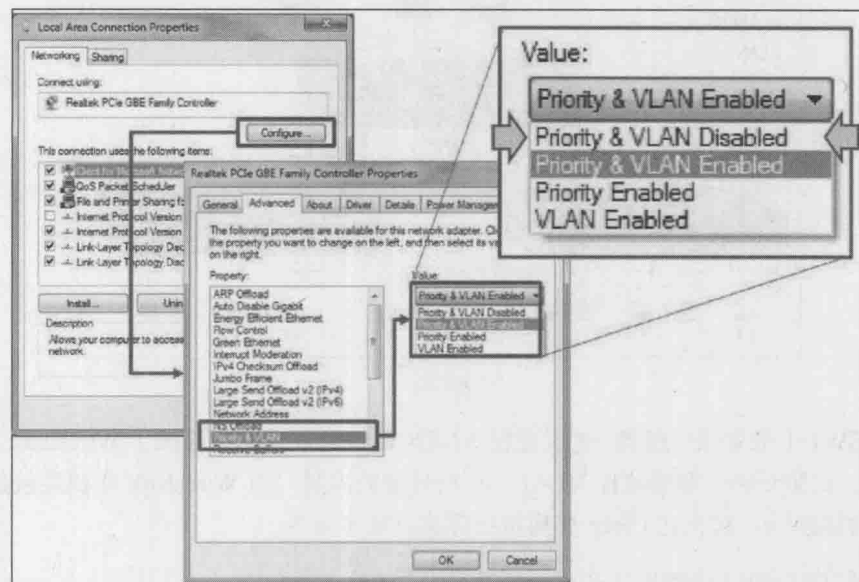


图 7.15

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的 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。

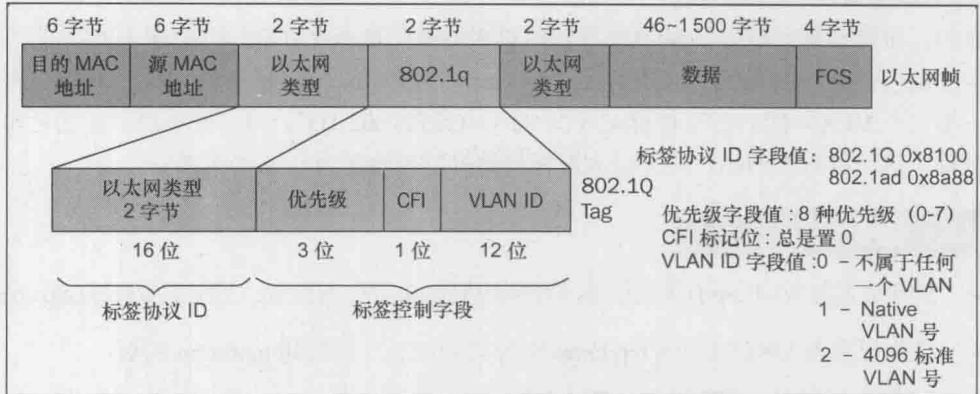
7.4.3 幕后原理

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

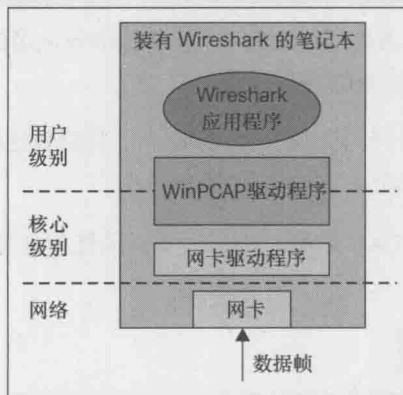


图 7.17

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)。

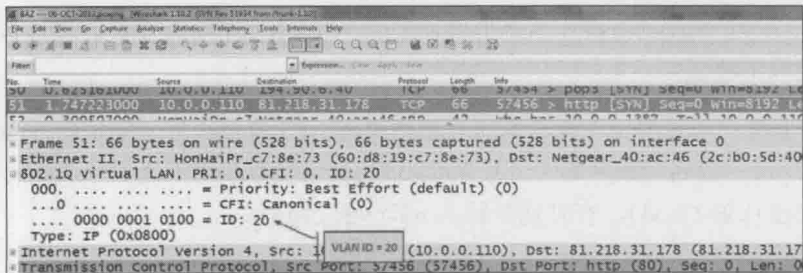


图 7.18

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，通过该工具可以获得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**）。

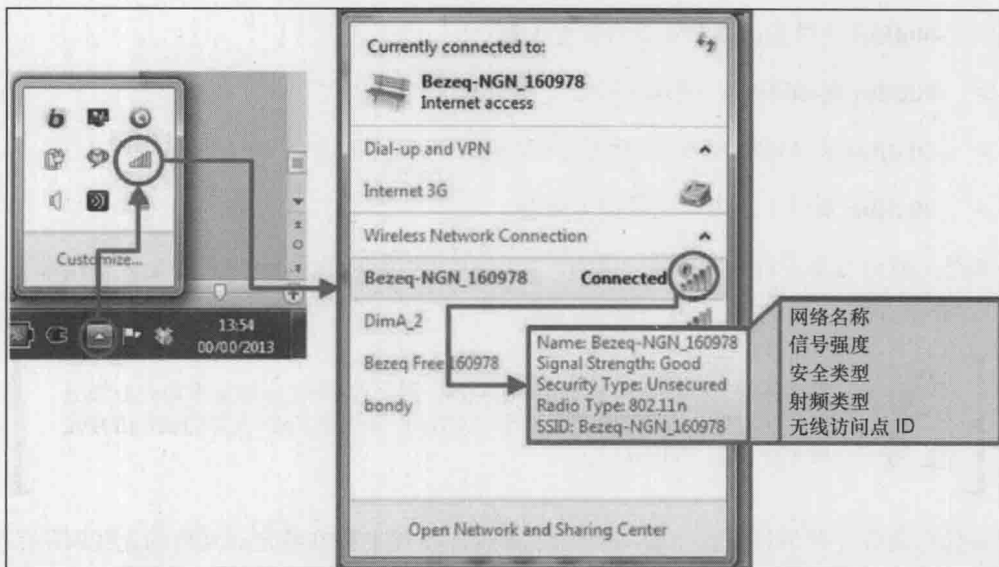


图7.19

- 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。

Network SSID	Freq. (KHz)	Channel	Rssi (dBm)	LinkQuality (%)	Mac Address	Dot11PhyType	Network type
bondy	2437000	6	-86	18	C8-BE-19-10-23-FE	7	infrastructure
Bezeq-NGN_160978	2437000	6	-60	70	34-08-04-16-09-79	7	infrastructure
Bezeq Free 160978	2437000	6	-65	60	34-08-04-16-09-7C	7	infrastructure
DimA_2	2437000	6	-73	44	FC-75-16-53-00-88	7	infrastructure

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**：表示无线信号质量上佳。
- **-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。

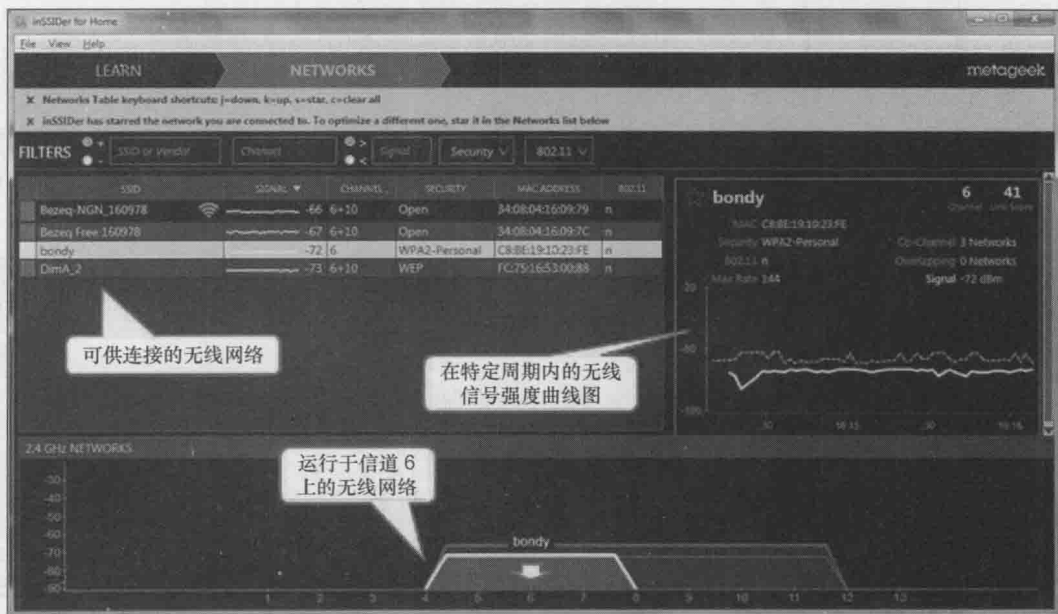


图7.21

^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。

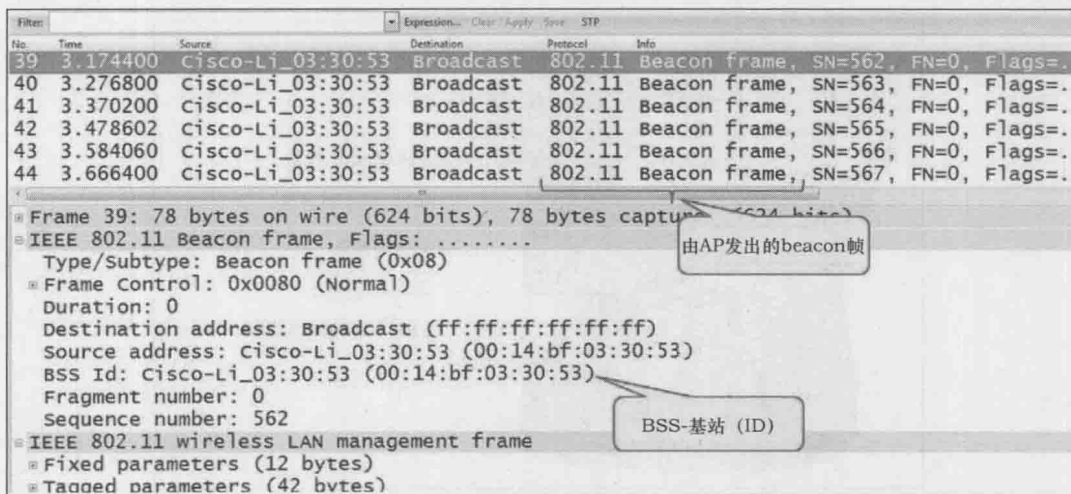


图 7.22



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。

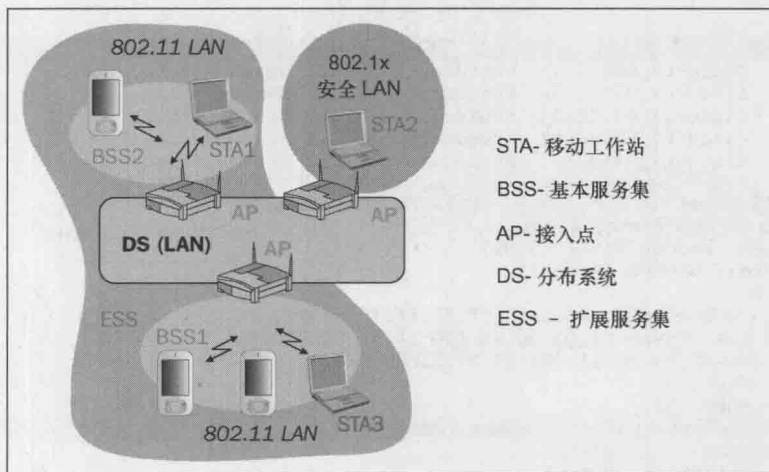


图7.23

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。

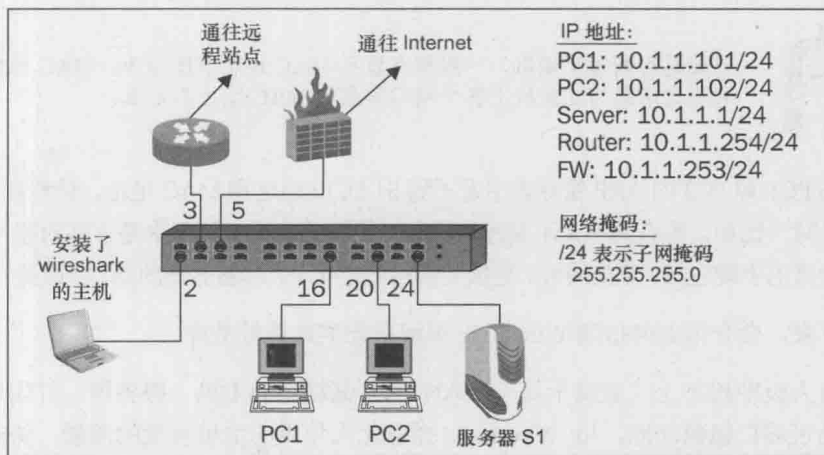


图8.1

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:::09	broadcast	ARP	172.30.116.100 正尝试解析
330	444.547	172.30.116.100	239.255.255.250	SSDP	172.30.116.254 的 MAC 地址
331	444.890	WistronI_ae:77:69	Broadcast	ARP	M-SEARCH * HTTP/1.1
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
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。

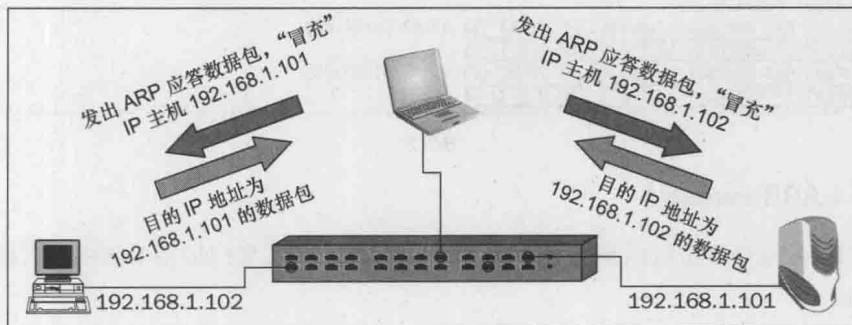


图8.3

^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）。

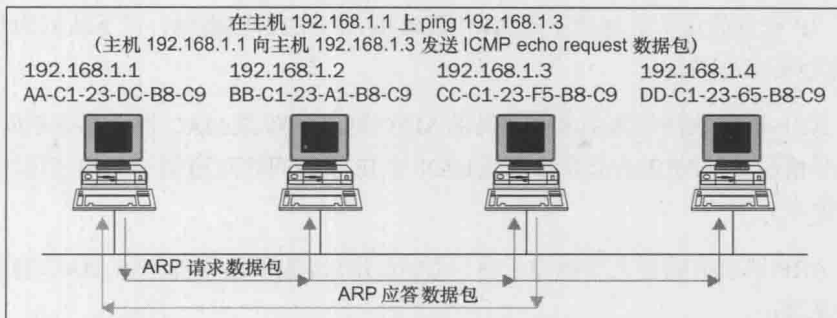


图 8.6

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。

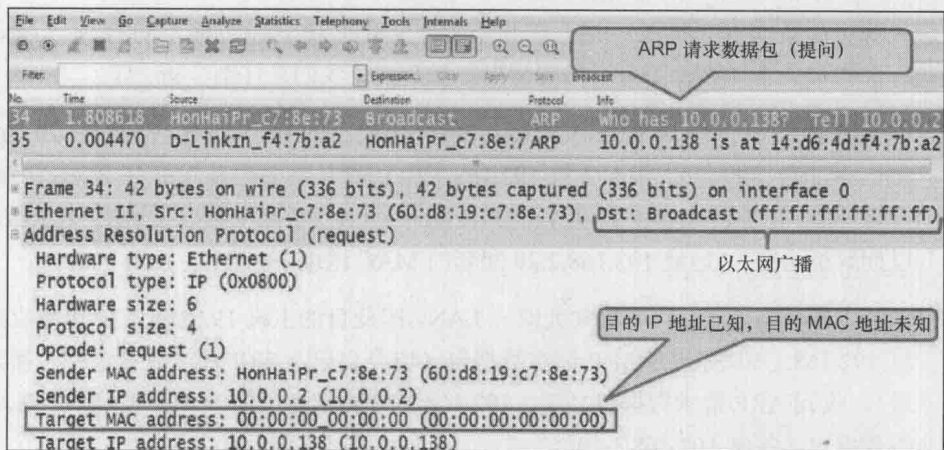


图8.7

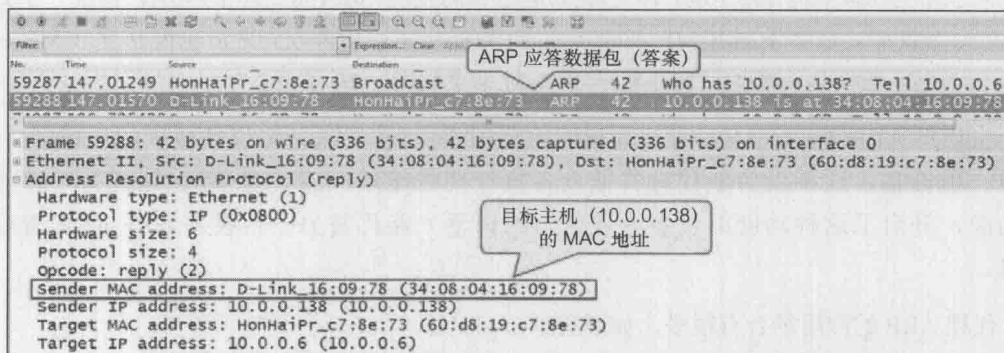


图8.8

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？

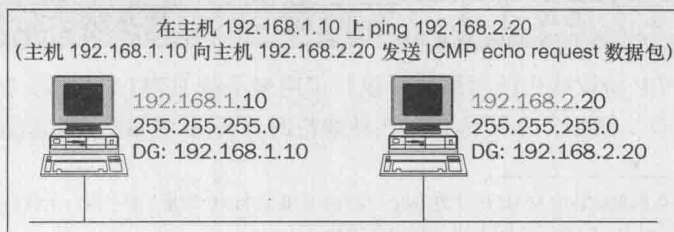


图8.9

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“当你在监控一条通信线路,连接到服务器,流量到 Internet,或任何其他类型的流量时,有几个工具可以帮助你监控源和目的 IP 地址”。

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。

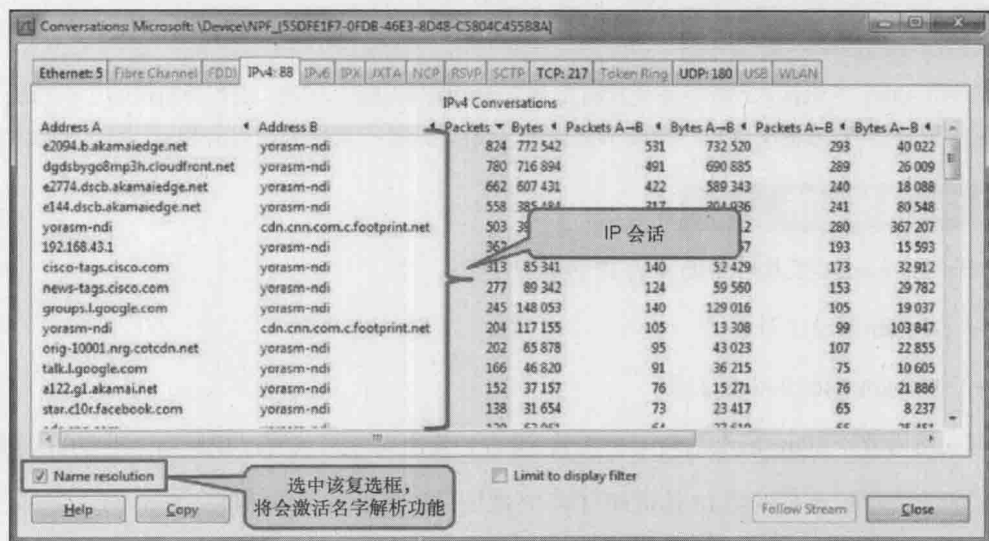


图8.11

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，把所抓数据包的源、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：

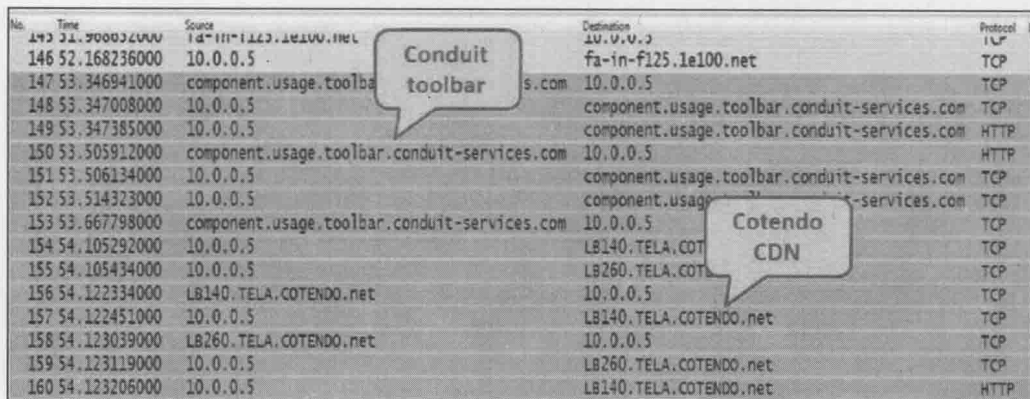
- 访问某特定 Web 站点的流量，而根据公司制度，该 Web 站点不允许访问；
- 软件自动升级所触发的流量，比如，访问杀毒软件 Web 站点或 Microsoft update 站点的流量。为使 Internet 链路得到充分利用，可部署一台中央服务器专门负责从 Internet 下载各种升级包，让所有内网 PC 都从这台服务器下载本机所需的升级包；
- 安装在内网主机的浏览器上的工具栏插件所触发的汹涌流量（请考虑一下公司内网里的每一台主机除了常规的上网流量之外，再打开约 50~100 条连接所触发的访问 Internet 的流量吧！）^①。

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
145	52.900032000	10.0.0.5	10.0.0.5	TCP
146	52.168236000	10.0.0.5	fa-in-f125.1e100.net	TCP
147	53.346941000	component.usage.toolbar.s.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	LB140.TELA.COTENDO.net	TCP
155	54.105434000	10.0.0.5	LB260.TELA.COTENDO.net	TCP
156	54.122334000	LB140.TELA.COTENDO.net	10.0.0.5	TCP
157	54.122451000	10.0.0.5	LB140.TELA.COTENDO.net	TCP
158	54.123039000	LB260.TELA.COTENDO.net	10.0.0.5	TCP
159	54.123119000	10.0.0.5	LB260.TELA.COTENDO.net	TCP
160	54.123206000	10.0.0.5	LB140.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（需配搭相应的 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** 窗口中选中 **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)。

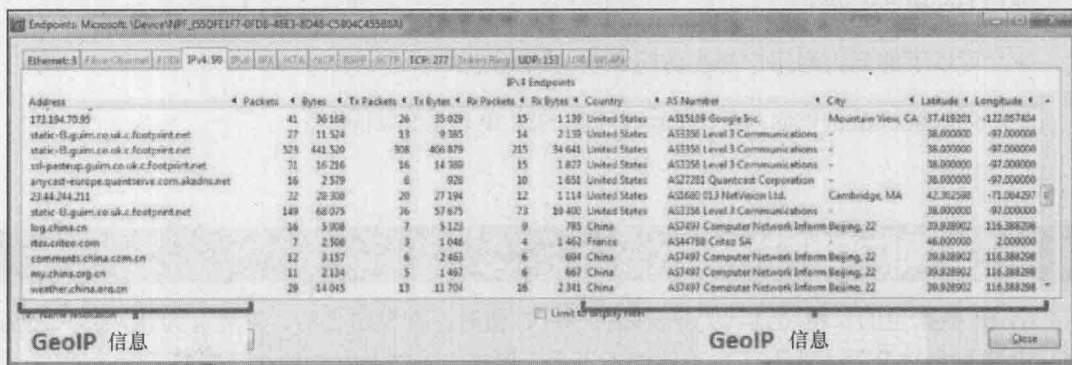


图8.14

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。

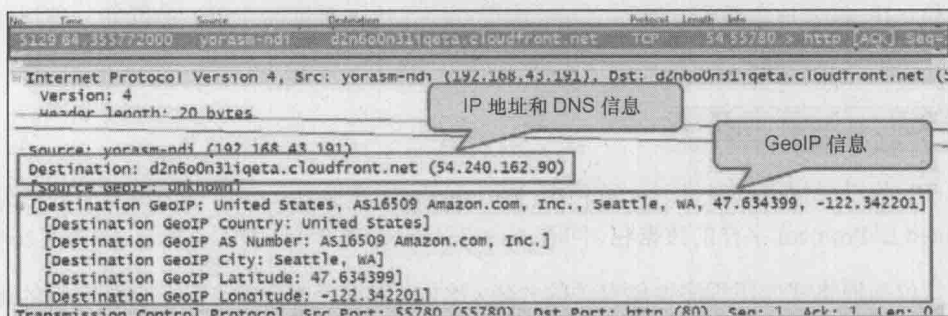


图8.15

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。

GeoLite 文件是 IP 归属地数据库的免费版本,每个月更新一次,下载链接为 http://dev.maxmind.com/geoip/geoip/geoip#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 数据包时的景象。
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。

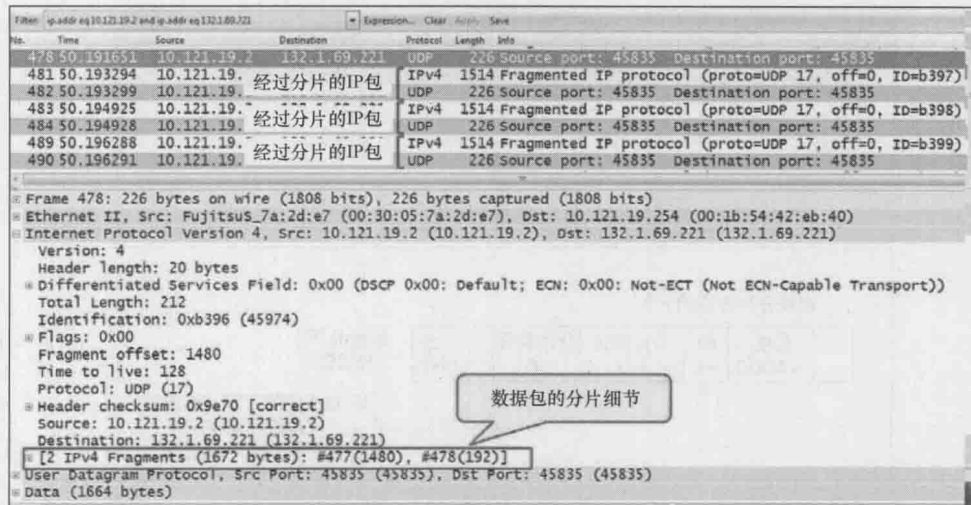


图8.16

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，作者的网路理论知识水平奇差，他给 MTU 下的定义是错误的。

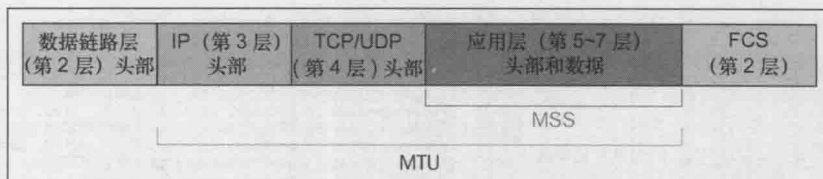


图8.17

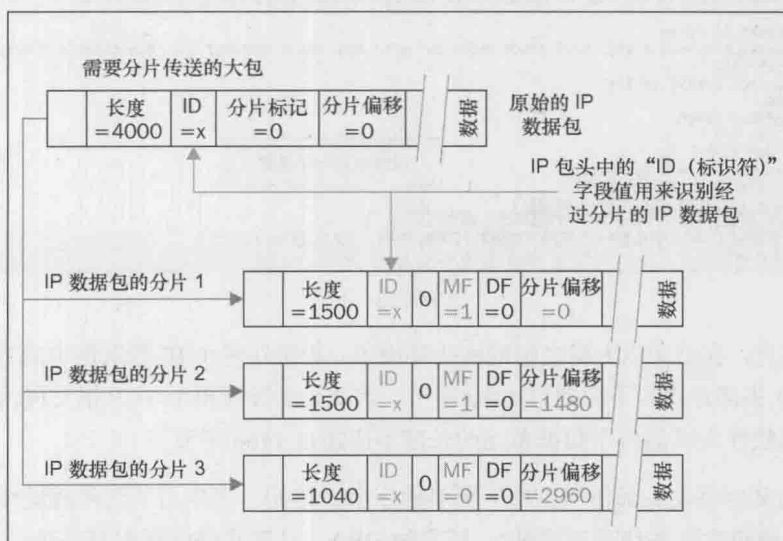


图8.18

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。

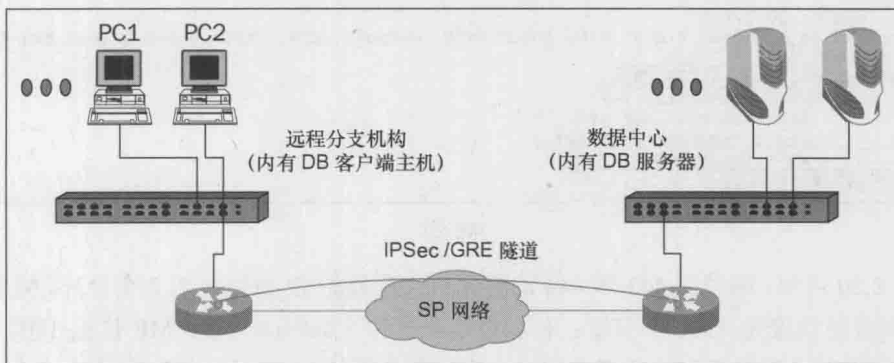


图8.19

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。

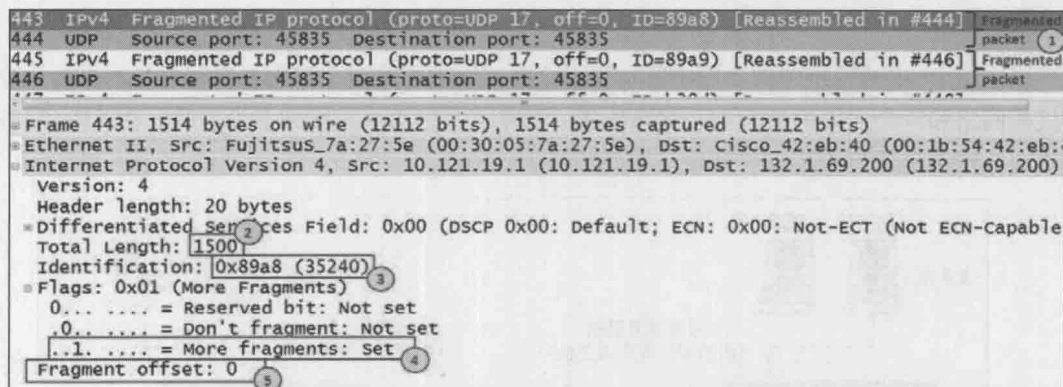


图8.20

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。

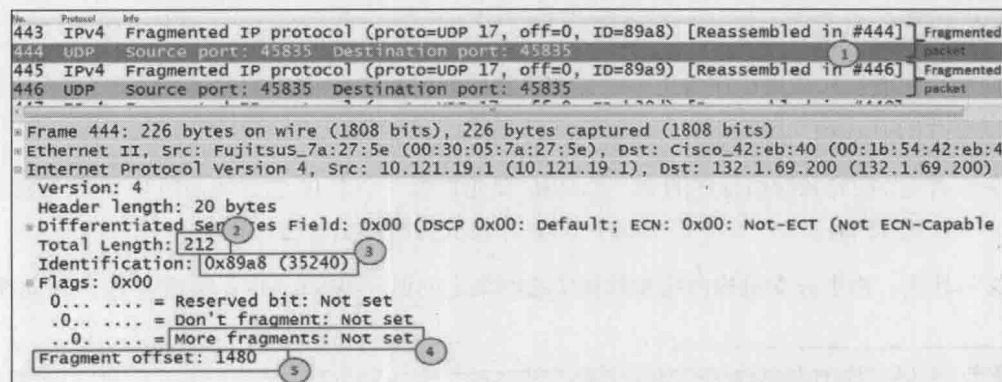


图8.21

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。

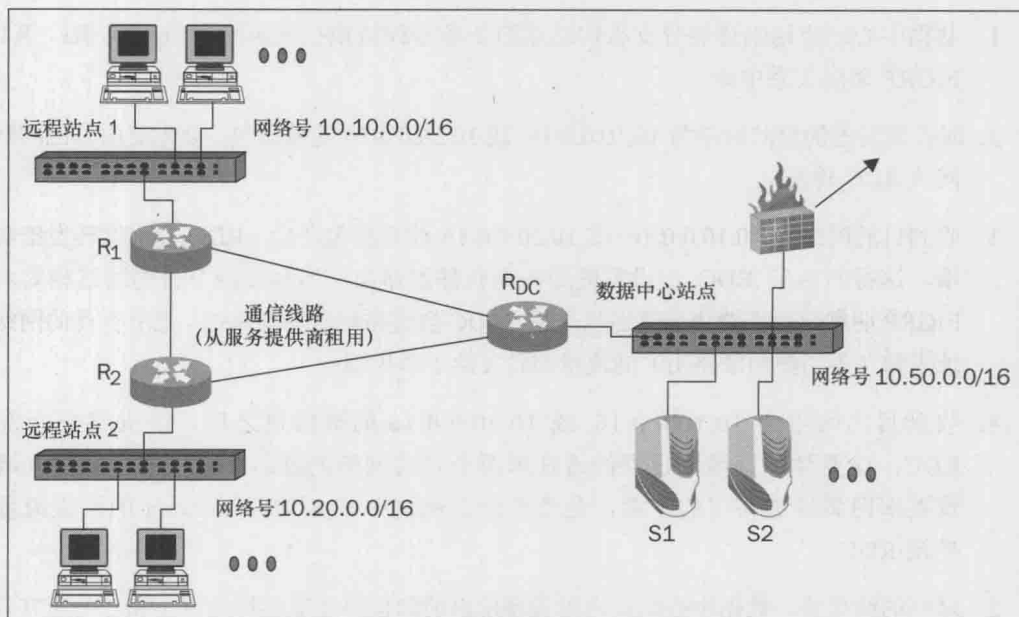


图8.22

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。

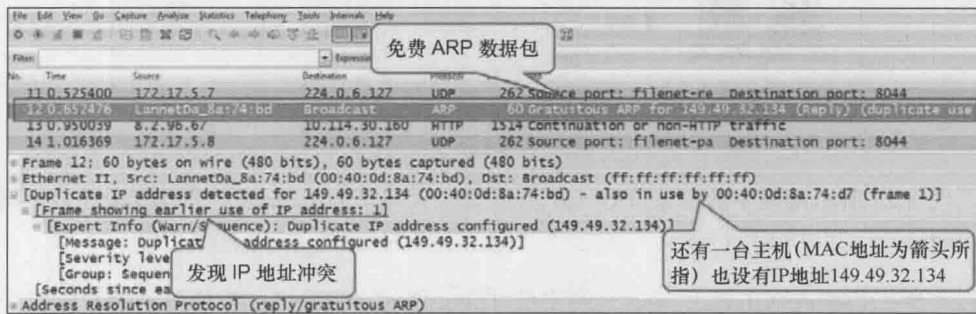


图8.24

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。

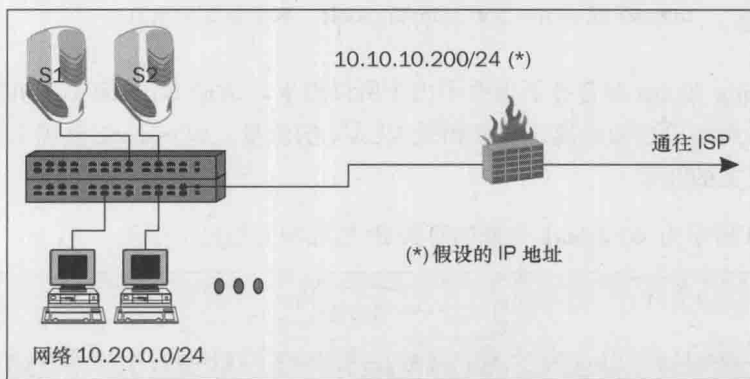


图 8.25

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。

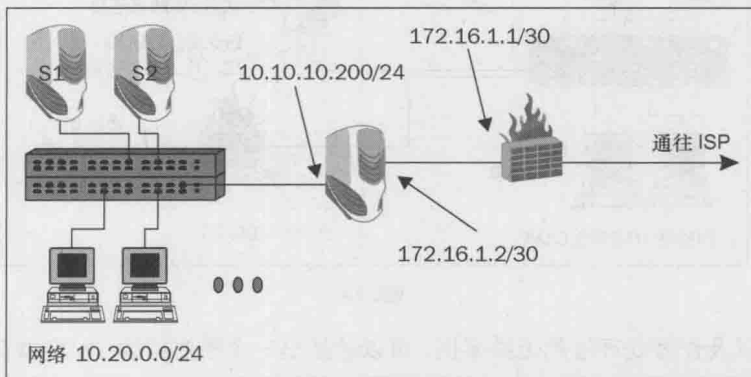


图8.26

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。

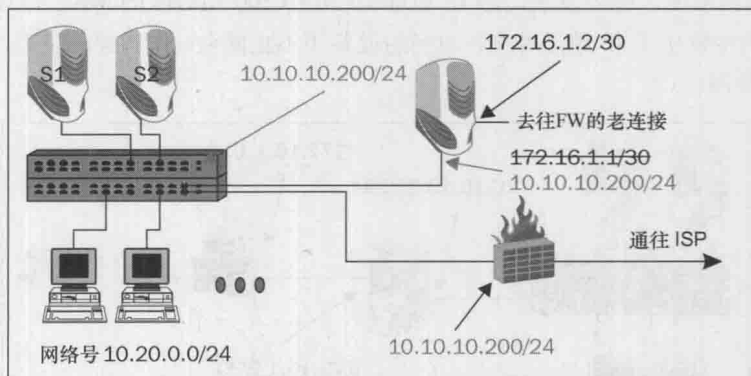


图8.27

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 Wireshark 得知)主机收到了 **DHCP Offer** 数据包,然后发出了 **DHCP Request** 数据包(问 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 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。

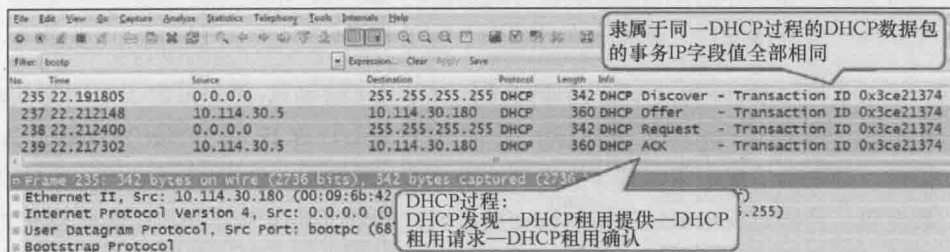


图8.28

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）。

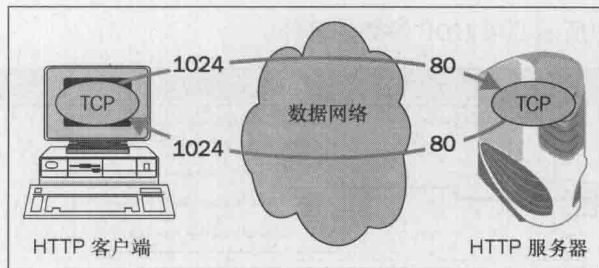


图9.1

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。

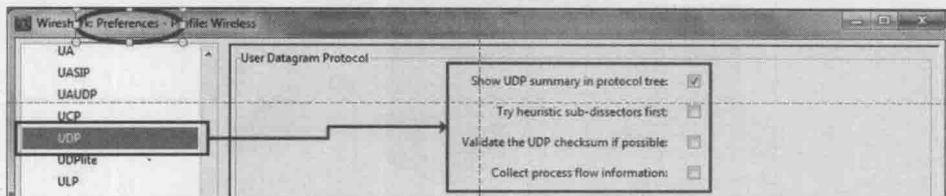


图9.2

- **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。

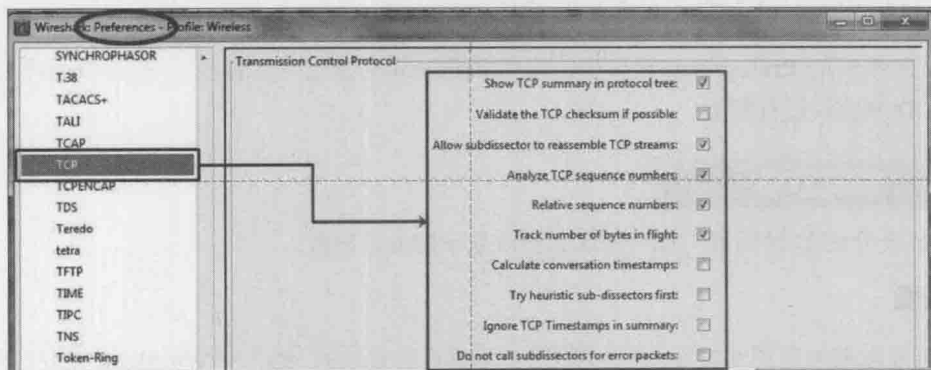


图9.3

- **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 将就在调用“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。

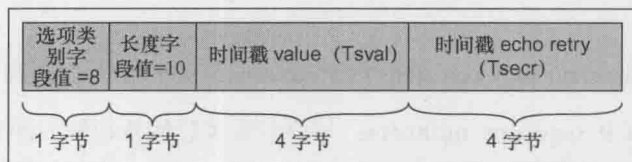


图9.4

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，既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 4.994751000	10.0.0.138			115 Standard query response 0xb38 PT
61 5.088214000	10.0.0.3	81.218.31.171	TCP	62 51910 > http [SYN] Seq=0 win=8192
62 5.090244000	10.0.0.3	81.218.31.171	TCP	62 51909 > http [SYN] Seq=0 win=8192
63 5.178158000	10.0.0.3	81.218.31.171	TCP	62 51912 > http [SYN] Seq=0 win=8192
64 6.247500000	10.0.0.3	173.194.78.125	TCP	66 51919 > http [SYN] Seq=0 win=8192
65 6.449442000	10.0.0.3	108.160.163.43	TCP	66 51921 > http [SYN] Seq=0 win=8192
66 6.480809000	108.160.163.43	10.0.0.3	TCP	66 http > 51921 [SYN, ACK] Seq=0 Ack=
67 6.480936000	10.0.0.3	108.160.163.43	TCP	54 51921 > http [ACK] Seq=1 Ack=1 win
68 6.481512000	10.0.0.3	108.160.163.43	TCP	543 GET /subscribe?host=10816016343
69 6.512241000	108.160.163.43			http > 51921 [ACK] Seq=1 Ack=290 W
70 6.512988000	108.160.163.43			HTTP/1.1 200 OK (text/html)

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)

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				
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	HTTP 316 GET / HTTP/1.1
2130	17.862143	135.82.12.1	10.0.0.3	54 http > 62423 [ACK] Seq=1 Ack=2
2189	18.736301	135.82.12.1	10.0.0.3	[TCP segment of a reassembled
2191	18.767301	135.82.12.1	10.0.0.3	[TCP segment of a reassembled

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

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 Win=0
2622	36.423135	135.82.12.1	10.0.0.3	TCP	54	6036 > 62442 [RST, ACK] Seq=915
2623	36.423135	fe80::c0b7:2c23:335:ff02::c	10.0.0.3	SSDP	208	M-SEARCH * HTTP/1.1
2624	36.423135	10.0.0.3	194.90.1.5	ICMP	74	Echo (ping) request id=0x0
2625	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 Win=0
2627	37.369547	135.82.12.1	10.0.0.3	TCP	54	6036 > 62442 [RST, ACK] Seq=915
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）和服务器端（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。

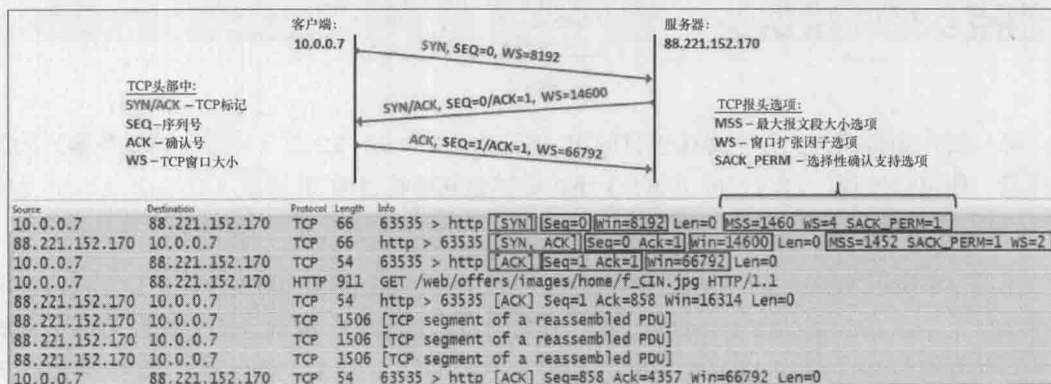


图 9.8

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。WSOPT 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。这既有可能是因为主机 10.0.0.5 发出的 TCP 报文段在 Internet 链路上丢失, 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 链路上丢失, 主机 10.0.0.5 收不到确认报文段, 误以为自己发出的 TCP 报文段丢失, 而发起了 TCP 重传。



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。

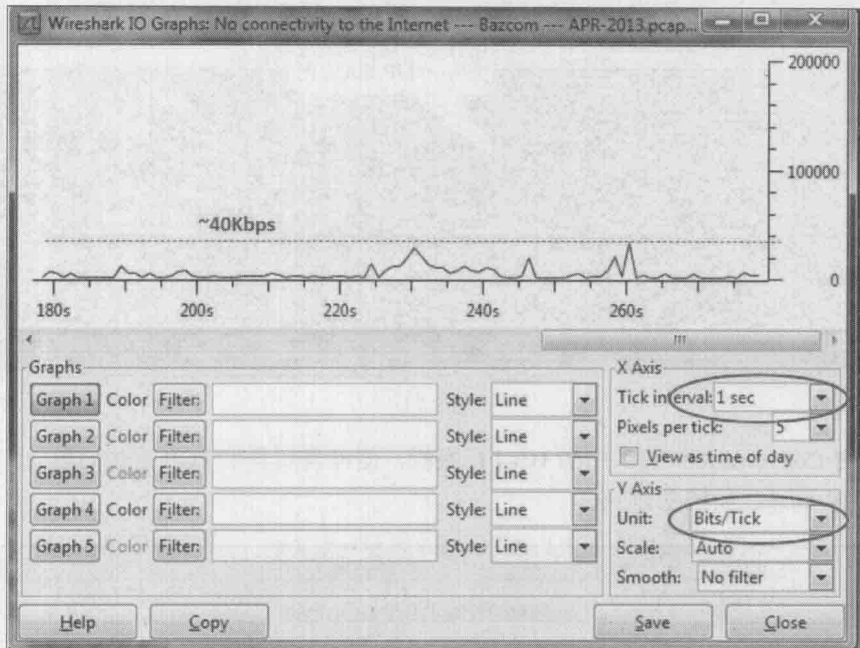


图9.10

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。见图 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。

Filter: expert.message == "Retransmission (suspected)"

所有 TCP 重传均发生在同一对 IP 地址之间

No.	Time	Source	Destination	Protocol	Length	Info
239	19.426155	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
240	19.427305	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
241	19.441252	10.1.1.200	10.90.30.12	TCP	121	TCP Retransmission
252	20.319262	10.90.30.12	10.1.1.200	TCP	153	TCP Retransmission
259	20.420397	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
269	21.322252	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
270	21.276400	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
288	22.323076	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
290	22.488386	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
291	22.489537	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
292	22.522244	10.90.30.12	10.1.1.200	TCP	148	TCP Retransmission
301	23.124370	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
303	23.363588	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
314	24.025662	10.90.30.12	10.1.1.200	TCP	60	TCP Retransmission
315	24.124575	10.90.30.12	10.1.1.200	TCP	159	TCP Retransmission

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。

Conversations: Sniff3 --- 10-11-2003 --- 1706.cap

Ethernet:1 Fibre Channel FDDI IPv4:1 IPv6 IPX JXTA NCP RSVP SCTP TCP:6 Token Ring UDP USB WLAN

IPv4 Conversations - Filter: expert.message == "Retransmission (suspected)"

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	127.90	24292.21
------------	-------------	-----	---------	----	-------	-----	---------	-------------	---------	--------	----------

所有 TCP 重传均发生在同一对 IP 地址之间

☐ Name resolution ☒ Limit to display filter

Help Copy Stream Graph A-B Graph B-A Close

拜图9.11所示 Wireshark 抓包主窗口的 Filter 输入栏内的显示过滤器所赐, 只要勾选了该复选框, Conversations 窗口将只显示存在 TCP 重传问题的会话

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。

Conversations: Sniff3 --- 10-11-2003 --- 1706.cap

Ethernet:1 Fibre Channel FDDI IPv4:1 IPv6 IPX JXTA NCP RSVP SCTP TCP:6 Token Ring UDP USB WLAN

TCP Conversations - Filter: expert.message == "Retransmission (suspected)"

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	2350	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	371	5	1 027	77 301798001	12 3181	240.95	666.99	941.75	1055.04	29.37
10.1.1.200	1972	10.90.30.12	1182	121	4	613	18.917254000	5.2973	N/A	1055.04	29.37		
10.1.1.200	1972	10.90.30.12	1186	157	4	700	52.6658855000	5.3078	N/A	1055.04	29.37		
10.90.30.12	1178	10.1.1.200	21	60	3	324	0.962677000	88.2566	N/A	24292.19	87.4711	N/A	24292.19

TCP 重传均发生在同一对 IP 地址及这些 TCP 端口号之间

☐ Name resolution ☒ Limit to display filter

Help Copy Follow Stream Graph A-B Graph B-A Close

拜图9.11所示 Wireshark 抓包主窗口的 Filter 输入栏内的显示过滤器所赐, 只要勾选了该复选框, Conversations 窗口将只显示存在 TCP 重传问题的会话

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。



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（NebBIOS 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”。

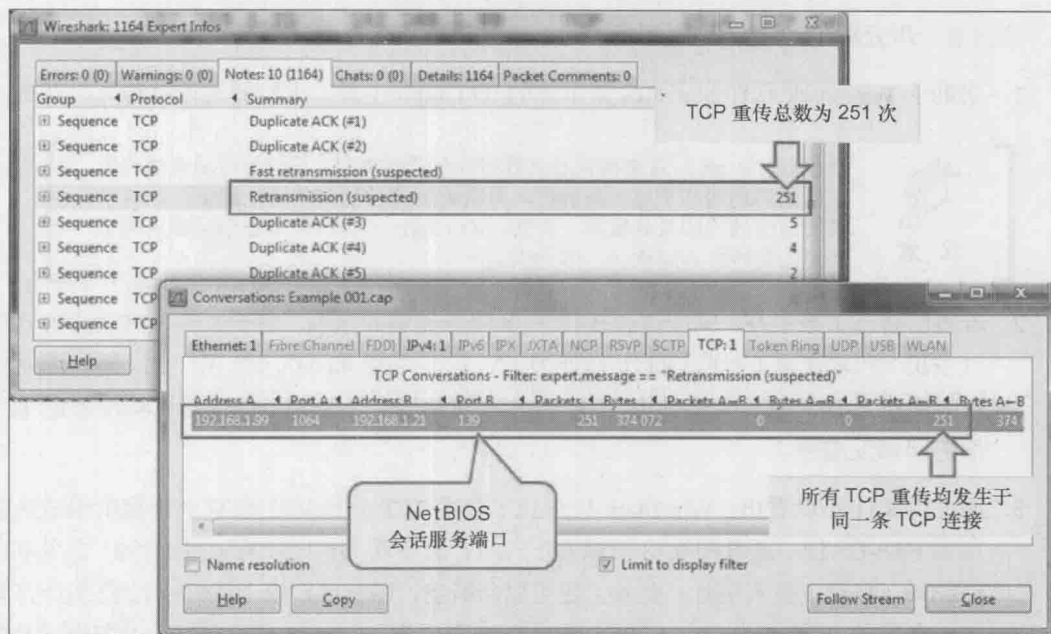


图9.14

No.	Time	Source	Destination	Protocol	Length	Info
1477	0.040695	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1505	0.032809	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1536	0.036683	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1563	0.030141	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1594	0.036375	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1627	0.039216	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1655	0.033171	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1686	0.036725	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1713	0.029351	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1744	0.036509	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1768	0.421340	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]
1790	0.023597	192.168.1.21	192.168.1.99	TCP	1514	[TCP Retransmission]

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器端发生故障时，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情之一。

- 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=29205
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 Len=0
1167	0.000033	192.168.201.93	192.168.3.50	TCP	54	agentview > http [ACK] Seq=1 Ack=1 win=65535
1168	0.000164	192.168.201.93	192.168.3.50	TCP	590	[TCP segment of a reassembled PDU]
1169	0.000030	192.168.201.93	192.168.3.50	TCP	54	[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)。

3. 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。)

4. 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
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

1. 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)。
2. 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。

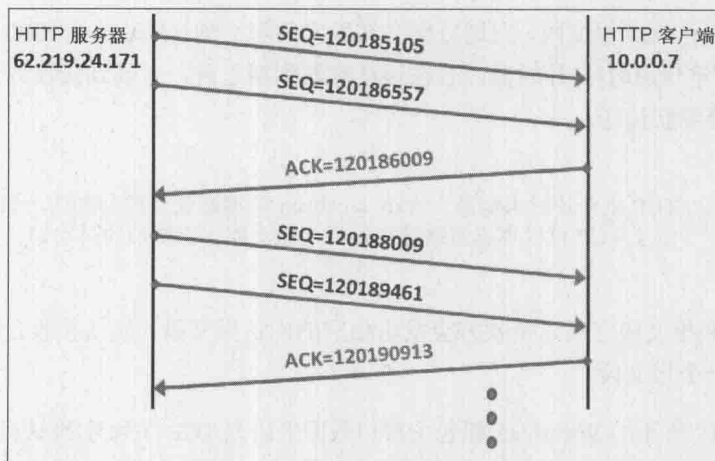


图9.18

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]）。

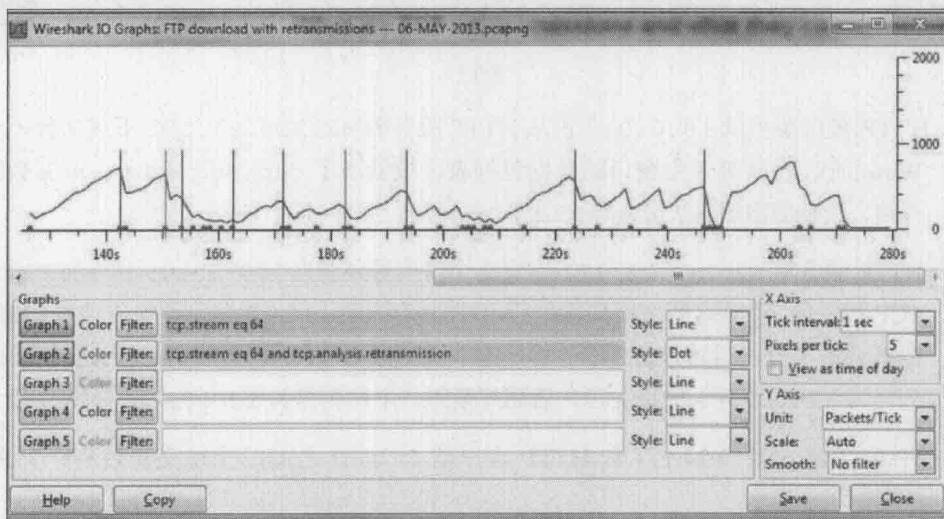


图9.19

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 I (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。

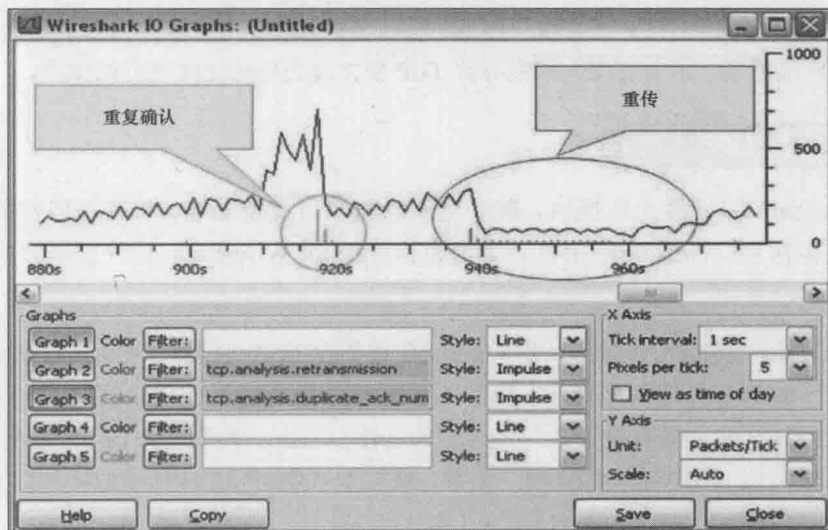


图9.21

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	Size
331	10.0.0.6	62.90.90.210	TCP	90	TCP Dup ACK 221454 [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 221455 [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 221456 [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 221457 [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 221458 [57999 > http	369
340	62.90.90.210	10.0.0.6	TCP	1474	TCP Previous segment not captured	332261
341	10.0.0.6	62.90.90.210	TCP	90	TCP Dup ACK 221459 [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 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。

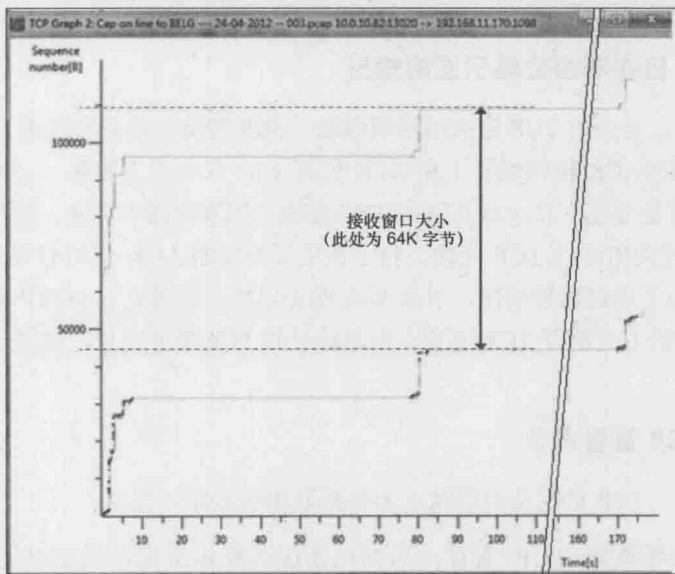


图9.25

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	Tcp-data > 39342 [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，这会导致该 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。

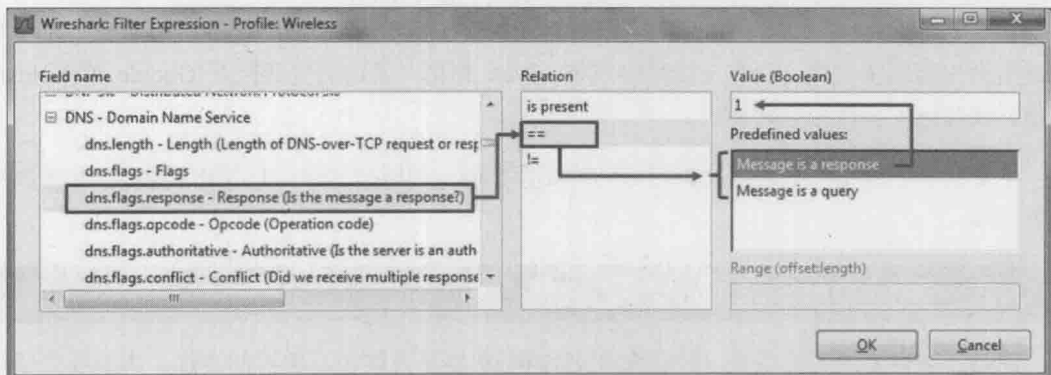


图10.1

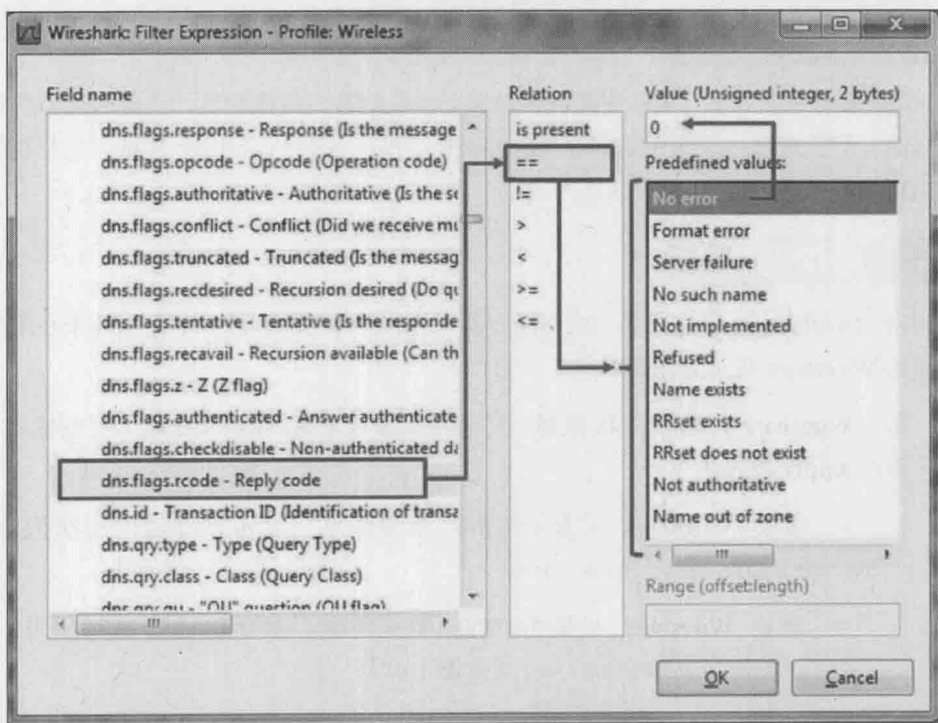


图10.2

- 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）。

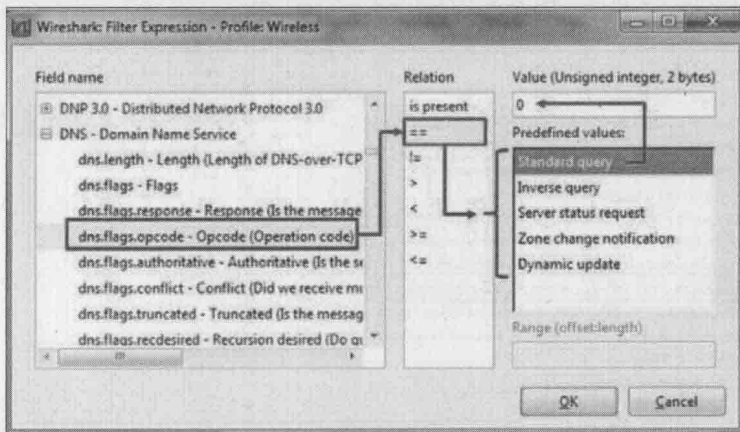


图10.3

- 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）。

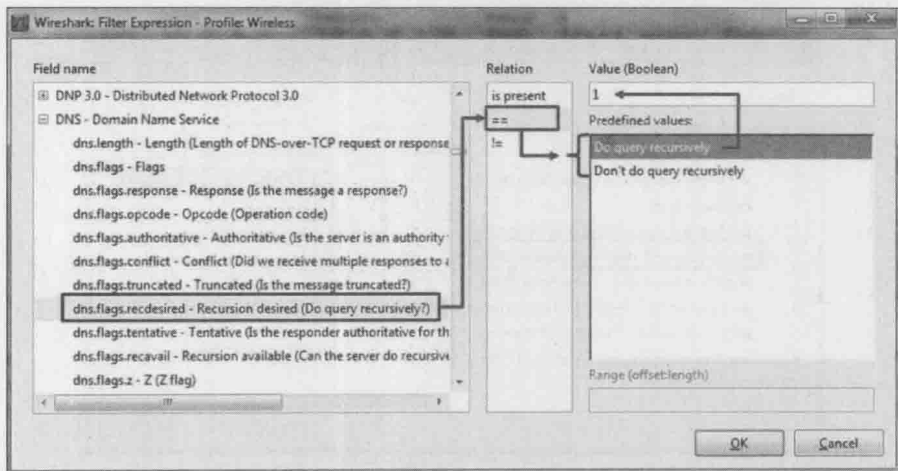


图10.4

- 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，在上网冲浪时会用到。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。

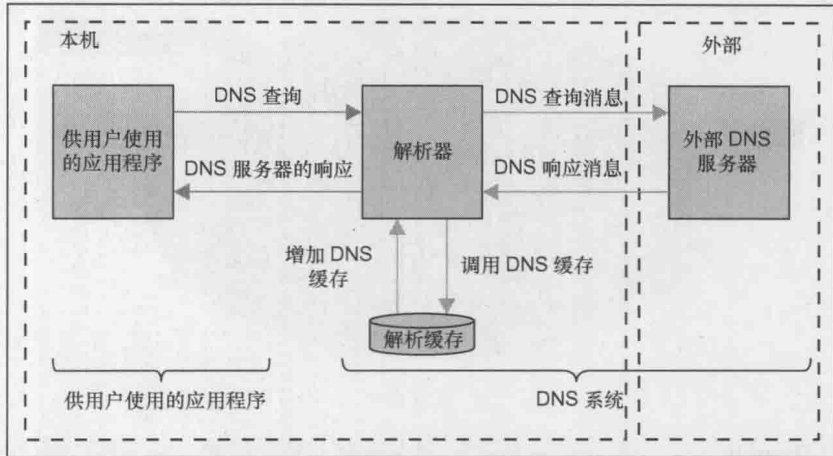


图10.5

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。

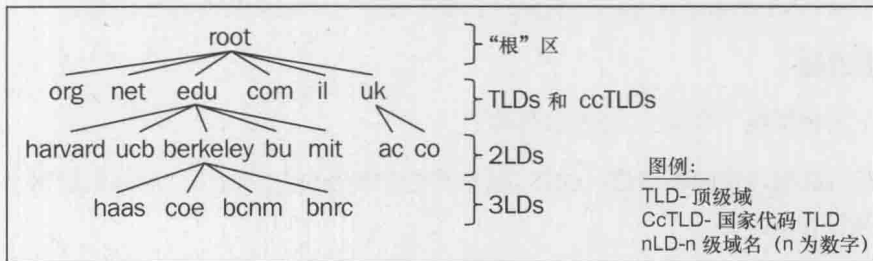


图10.6

- 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，因此他在这本章里对许多 DNS 概念的解释都是错误的。

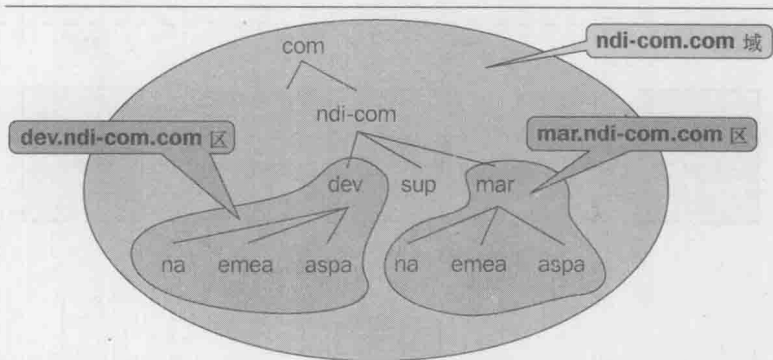


图10.7

- **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 服务器的数据库中沒有待查的域名记录时，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。

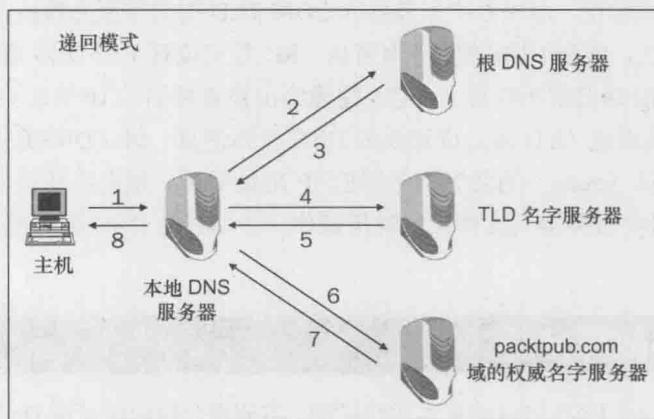


图 10.8

若本地 DNS 服务器也不知道域名 www.packtpub.com 的 IP 地址，且对 [packtpub.com](http://www.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。

10.4.2 操作方法

DNS 故障主要分为两类：

- 无法解析域名；
- 解析缓慢。

若上述两类故障现象只存在于个别用户主机，需在 LAN 交换机上开启端口镜像功能，并设法将该主机生成的流量重定向至 Wireshark 主机。此时，只需关注由这台主机生成的流量。

若上述两类故障现象普遍存在于网络，需在 LAN 交换机上开启端口镜像功能，并设法将接 DNS 服务器（或连接 Internet 线路）的端口的流量重定向至 Wireshark 主机。

无法解析域名

如何定位问题？

1. 当无法访问 Internet（无法浏览网页、收发 Email 或执行与上网有关的任何一项操作）时，若发现电脑的“本地连接”并未打叉，请 ping 一下有待访问的 Web 站点，看看能否 ping 通^①。
2. 如能 ping 通，则表示网络状况正常。
3. 要是 ping 不通，则可能是因为：
 - Web 站点“禁 ping”；
 - DNS 服务器无法解析有待访问的 Web 站点的域名。
4. 要确定上不了网（无法访问 Internet）是否拜 DNS 故障所赐，需动用 Wireshark，并配置显示过滤器，筛选出 DNS 流量。通过观察筛选出的 DNS 流量，可以发现 DNS 域名无法解析无非以下两种原因：
 - 有待访问的 Web 站点不存在；
 - DNS 服务器不做回应。



注意

也可以使用内置于主机操作系统的 nslookup 工具，来尝试解析有待访问的 Web 站点的域名。

^① 译者注：作者也太心急了，至少也得先 ping 一下本机配置的默认网关的 IP 地址和 DNS 服务器的 IP 地址吧？

5. 当有待访问的 Web 站点不存在时（具体示例详见图 10.9），请仔细观察经过过滤的 Wireshark 抓包文件，将会发现下述情况^①。

- 抓包文件中存在事务 ID 字段值相同的 DNS 查询和响应消息：DNS 查询消息和 DNS 响应消息的事务 ID 字段值相同，便意味着这两种消息是（主机与 DNS 服务器之间的）一问一答。由图 10.9 可知，Wireshark 抓到了事务 ID 字段值同为 0x971e 的一对 DNS 查询/响应消息。
- 一问一答（即 Wireshark 抓到的成对的 DNS 查询和响应消息之间）有一定的延迟：这说明本地 DNS 服务器在执行递归解析操作时经过了长途跋涉。由图 10.9 可知，一问一答的间隔时间为 346ms，之所以会出现这么长的延迟，是因为位于欧洲的本地 DNS 服务器在执行递归解析操作时，一直问到了位于中国台湾的权威名字服务器。
- 抓包文件中存在身背 No such name 字样的 DNS 响应消息。由图 10.9 可知，Wireshark 抓到了本地 DNS 服务器传回的身背 No such name 字样的 DNS 响应消息。仔细观察这条 DNS 响应消息的内容，不难发现，是位于中国台湾的权威名字服务器 a.dns.tw 做出了权威应答“有待访问的 Web 站点的域名不存在”。这充分说明了整个 DNS 系统运转正常，但有待访问的 Web 站点的域名不存在。

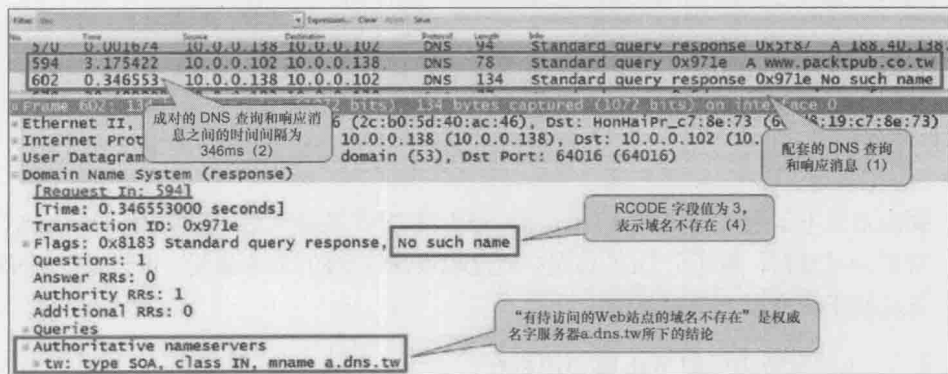


图10.9

6. 当 DNS 服务器不做回应时，仔细观察 Wireshark 抓包文件中经过筛选的数据包，将会看到以下两图所示场景。

- Wireshark 抓到了身背 refused 字样的 DNS 应答消息：这表示查询请求遭到了 DNS 服务器的拒绝，如图 10.10 所示（为什么会遭到拒绝，下一节详述）。

① 译者注：原文不成体统，译文酌改。

No.	Time	Source	Destination	Protocol	Length	Info
1936	0.329869	10.0.0.102	10.0.0.138	DNS	86	Standard query 0x1c2a A suggest.search.conduit.com
1937	0.008601	10.0.0.138	10.0.0.102	DNS	86	Standard query response 0x1c2a Refused
1941	0.081956	10.0.0.102	10.0.0.138	DNS	86	Standard query 0x90b7 A suggest.search.conduit.com
1942	0.001716	10.0.0.138	10.0.0.102	DNS	86	Standard query response 0x90b7 Refused
1944	0.234208	10.0.0.102	10.0.0.138	DNS	86	Standard query 0xe979 A suggest.search.conduit.com
1945	0.002137	10.0.0.138	10.0.0.102	DNS	86	Standard query response 0xe979 Refused
1947	0.280145	10.0.0.102	10.0.0.138	DNS	86	Standard query 0xcc35 A suggest.search.conduit.com
1948	0.001729	10.0.0.138	10.0.0.102	DNS	86	Standard query response 0xcc35 Refused
1950	0.231956	10.0.0.102	10.0.0.138	DNS	86	Standard query 0x2a5d A suggest.search.conduit.com
1951	0.001640	10.0.0.138	10.0.0.102	DNS	86	Standard query response 0x2a5d Refused

Flags: 0x8185 Standard query response, Refused	
1...	= Response: Message is a response
.000 0...	= Opcode: Standard query (0)
....0...	= Authoritative: Server is not an authority for domain
....0...	= Truncated: Message is not truncated
....1...	= Recursion desired: Do query recursively
....1...	= Recursion available: Server can do recursive queries
....0...	= Z: reserved (0)
....0...	= Answer authenticated: Answer/authority portion was not authenticated by the server
....0...	= Non-authenticated data: Unacceptable
0101	= Reply code: Refused (5)

RCODE 字段值为 5 的 DNS 应答消息

图 10.10

- Wireshark 连续抓到了多条事务 ID 字段值相同的 DNS 查询消息（如图 10.11 所示），但却未抓到一条配套的 DNS 应答消息；这表示 DNS 服务器因某种原因没有回应查询请求。

No.	Time	Source	Destination	Protocol	Length	Info
1811	11.142885	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1813	1.000038	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1814	0.227675	10.0.0.102	10.0.0.138	DNS	75	Standard query 0xc56f A su.ffa.avast.com
1820	0.772330	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1821	0.227037	10.0.0.102	10.0.0.138	DNS	75	Standard query 0xc56f A su.ffa.avast.com
1822	0.999979	10.0.0.102	10.0.0.138	DNS	75	Standard query 0xc56f A su.ffa.avast.com
1824	0.772945	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1827	1.227121	10.0.0.102	10.0.0.138	DNS	75	Standard query 0xc56f A su.ffa.avast.com
1837	2.773089	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com

DNS 查询消息：

- 1) 查询域名 s.gateway.messenger.live.com（事务 ID 字段值 0x5614）
- 2) 查询域名 su.ffa.avast.com（事务 ID 字段值 0xc56f）

DNS 消息的事务 ID 字段值 查询消息的内容

图 10.11

要想看得更仔细一点，请随便选中一条 DNS 查询消息，单击右键，点击右键菜单里的 **Follow UDP Stream** 菜单项，其结果如图 10.12 所示。由图 10.12 可知，主机 10.0.0.102 向本地 DNS 服务器 10.0.0.138 连续发送了 5 条事务 ID 字段值相同的 DNS 查询消息（每条 DNS 查询消息间的时间间隔逐渐由短变长），但未收到任何答复。

No.	Time	Source	Destination	Protocol	Length	Info
1811	0.000000	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1813	1.000038	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1820	1.000005	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1824	1.999961	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com
1837	4.000210	10.0.0.102	10.0.0.138	DNS	88	Standard query 0x5614 A s.gateway.messenger.live.com

DNS 查询消息间的时间间隔逐渐增加 所有 DNS 消息的事务 ID 字段值相同 有待请求查询的域名相同

图 10.12

DNS 解析缓慢

如何定位问题？

1. 上网冲浪时，若网页打开缓慢（或使用内网应用程序时，应用程序反应迟钝），请按以下步骤行事。

- 在交换机上开启端口镜像功能，将其连接 Internet 链路的端口的流量重定向至 Wireshark 主机。使用第 5 章介绍过的 IO Graph 工具，测量过往于该端口的流量的速率，以此来核实 Internet 链路是否过载。
- 检查 Wireshark 是否抓到了数量惊人的身背 TCP retransmissions 或 TCP duplicate ACK 字样的数据包。若是，则表示存在 TCP 传输问题。
- 检查 Wireshark 是否抓到了数量惊人的身背 TCP window 字样的数据包。若是，则表示存在 TCP 窗口问题。

2. 若不存在上述问题，则需检查是否存在 DNS 解析问题。

3. 对于有待解决的 DNS 解析问题，要按以下两种情况来处理。

- 若故障表象为内网应用程序反应迟钝，则需开启端口镜像功能，将交换机上连接内网 DNS 服务器的端口的流量重定向至 Wireshark 主机。
- 若故障表象为（上网冲浪时）网页打开缓慢（或互联网应用程序反应迟钝），则需将交换机上连接 Internet 链路的端口的流量重定向至 Wireshark 主机。

4. 获悉 DNS 解析所花费的时间，找出解析缓慢的原因。有数种获悉 DNS 解析所耗时间的方法，如下所列。

- 最简单的方法是，在 Wireshark 数据包列表区域内，选中一个隶属于 DNS 数据流的 DNS 数据包，单击右键，点击 **Follow UDP Stream** 菜单项。然后，观察 Wireshark 数据包列表区域中的 Time 列，就可以获悉相关域名的 DNS 解析时间了。
- 另一种方法是，借助于内置于 Wireshark 的 IO Graph 工具。在 IO Graphs 窗口中，要先在 Y 轴坐标区域的 Unit 选项的下拉菜单里选择 **Advanced** 选项；再到陡然增大的 IO Graphs 窗口中，选择 **Clac:** 下拉菜单栏里的 **AVG(*)** 菜单项，同时在其右边的 Filter 输入栏内填入显示过滤参数 `dns. time`，选择 **Clac:** 下拉菜单栏里的 **Dot** 菜单项；最后点击相关 **Graph** 按钮，激活 IO Graphs 窗口中与该 graph 相对应的图形，于是便得到了在整个抓包时段内的 DNS 响应时间的汇总图，如图 10.13 所示。

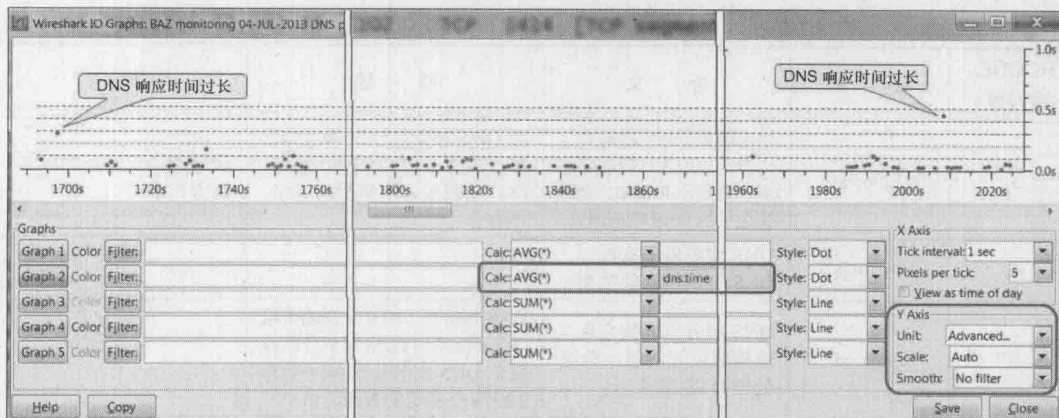


图10.13

由图 10.13 可知，在抓包时段内，绝大多数 DNS 解析的响应时间都在 100 毫秒以内，这完全可以接受；但也可以看到，有两次 DNS 解析响应过慢，一次为 300ms，另一次为 450ms，分别发生在抓包的开始和收尾阶段。

注意 在组织机构的内部网络中，合理的 DNS 解析响应时间应在几十毫秒以内；上网冲浪时，合理的响应时间不应超过 100ms，200ms 之内的响应时间则勉强可以接受。

10.4.3 幕后原理

RFC 1035 定义了 DNS 响应消息中响应代码(RCODE)字段的前 6 个值(0~5)。其他的 RCODE 字段值(6~21)则在后续标准(RFC 2136、RFC 2671、RFC 2845、RFC 2930)中定义。所有已定义的 RCODE 字段值详见 <http://tools.ietf.org/html/rfc2929#section-2.3>。

下表所列为最为常见的 RCODE 字段值及含义：

RCODE 字段值	名 称	含 义	原 因	如何处理
0	NoError	未发生错误	DNS 解析正常	无需采取任何行动
1	FormErr	DNS 服务器无法解析查询请求	当 DNS 服务器收到了格式错误的 DNS 查询消息，或收到了携带某些扩展功能字段的 DNS 查询消息，但无法识别时，便会回复 RCODE 字段值为 1 的 DNS 响应消息	可试着更换一台 DNS 服务器，重新执行 DNS 扩展功能查询请求
2	ServFail	DNS 服务器出现故障，无法处理 DNS 查询请求	当 DNS 服务器回复 RCODE 字段值为 1 的 DNS 响应消息时，便表示其发生了故障	可试着更换一台 DNS 服务器，重新发送查询请求

续表

RCODE 字段值	名 称	含 义	原 因	如何处理
3	NXDomain	请求解析的域名不存在 (界定标准为: 权威名字 服务器发出 RCODE 字段 值为 1 的 DNS 响应消息)	当 DNS 服务器回复 RCODE 字段值为 1 的 DNS 响应消息 时, 便表示请求解析的域名 不存在	请检查请求解析的域名 是否正确
4	NotImp	DNS 服务器不支持 DNS 查询消息的类型		
5	Refused	DNS 服务器因安全性 或策略设置问题, 拒绝 处理 DNS 请求	DNS 服务器(因安全性或策略 设置问题)不愿意为某些设备 提供 DNS 查询服务, 或不愿意 回应某些特定的查询请求	

10.4.4 拾遗补缺

DNS 服务器的 IP 地址如何配置(即在本机 TCP/IP 配置的 DNS 服务器配置项里应该填哪一个 IP 地址), 是很多人常问的问题。作者的答案非常简单——应填入离本机最近、解析速度最快的 DNS 服务器的 IP 地址, 这一 IP 地址一般都是本地电信运营商的 DNS 服务器的 IP 地址。

图 10.14 所示为由 Google Namebench 软件(开源软件)测算出的各 DNS 服务器的平均解析时间。

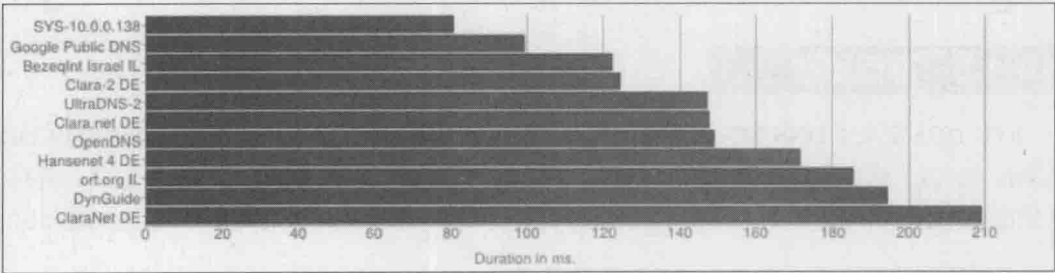


图10.14

- 内网 DNS 服务器的平均解析时间为 80 毫秒, 其 IP 地址为私网地址 10.0.0.138。
- ISP DNS 服务器 (Google public DNS) 的平均解析时间为 100 毫秒。
- 所有位居海外的 DNS 服务器的平均解析时间都高于 120 毫秒。

总而言之, 只要 DNS 解析时间在 100 毫秒之内, 则一切正常; 在绝大多数情况下, 解析时间在 150~200 毫秒之间, 也算正常; 即便是偶而出现几次 DNS 解析缓慢的情况, 也无需担心, 因为这极有可能是本地 DNS 服务器在等待位于地球另一端 (待请求域名所在区域) 的权威名字服务器的应答。

上网冲浪时,若浏览器所要加载的内容过多,则本机 DNS 解析器有可能会执行数十次 DNS 解析操作。图 10.15 所示为作者通过 Web 浏览器访问 www.cisco.com 时 Wireshark 抓取到的 DNS 流量。

No.	Time	Source	Destination	Protocol	Info
2346	67.113672	10.0.0.2	10.0.0.138	DNS	Standard query 0x744f A www.cisco.com ①
2347	67.113670	10.0.0.138	10.0.0.2	DNS	Standard query response 0x7444 CNAME www.cisco.com.akadns.net
2361	69.122842	10.0.0.2	10.0.0.138	DNS	Standard query 0x0df4 A api.webrep.avast.com
2362	69.123955	10.0.0.2	10.0.0.138	DNS	Standard query 0xd485 A ap.ff.avast.com ②
2364	69.142771	10.0.0.138	10.0.0.2	DNS	Standard query response 0x0df4 A 77.234.43.95 A 77.234.43.9
2365	69.142819	10.0.0.138	10.0.0.2	DNS	Standard query response 0xd485 A 109.123.117.68 A 77.234.44
2369	69.201093	10.0.0.2	10.0.0.138	DNS	Standard query 0x4528 A www.static-cisco.com ③
2397	69.272365	10.0.0.2	10.0.0.138	DNS	Standard query 0xdf38 A www.cisco.com
2398	69.294623	10.0.0.138	10.0.0.2	DNS	Standard query response 0xdf38 CNAME www.cisco.com.akadns.net
2501	69.388773	10.0.0.138	10.0.0.2	DNS	Standard query response 0x4528 CNAME static-cisco.cisco.com
2662	69.822286	10.0.0.2	10.0.0.138	DNS	Standard query 0x7cf8 A ciscosystemsinc.tt.omtrdc.net ④
2776	69.911815	10.0.0.138	10.0.0.2	DNS	Standard query response 0x7cf8 A 70.42.13.100 A 66.117.23.1
3056	70.547976	10.0.0.2	10.0.0.138	DNS	Standard query 0x010f A news-tags.cisoc.com ⑤
3058	70.566150	10.0.0.2	10.0.0.138	DNS	Standard query 0x80ac A cisco-tags.cisco.com
3059	70.567814	10.0.0.138	10.0.0.2	DNS	Standard query response 0x010f A 72.163.10.14
3060	70.576567	10.0.0.2	10.0.0.138	DNS	Standard query 0x4468 A cisco.112.2o7.net
3066	70.584723	10.0.0.138	10.0.0.2	DNS	Standard query response 0x80ac A 72.163.10.10
3073	70.595827	10.0.0.138	10.0.0.2	DNS	Standard query response 0x4468 A 66.235.139.110 A 66.235.13
3122	71.153544	10.0.0.2	10.0.0.138	DNS	Standard query 0xfc41 A tools.cisco.com
3128	71.218031	10.0.0.2	10.0.0.138	DNS	Standard query 0x9d0c A products.mcisco.com ⑥
3129	71.218284	10.0.0.2	10.0.0.138	DNS	Standard query 0x130c A newsroom.cisco.com ⑦
3138	71.348653	10.0.0.138	10.0.0.2	DNS	Standard query response 0xfc41 A 72.163.4.38
3140	71.376504	10.0.0.2	10.0.0.138	DNS	Standard query 0xd077 A ciscocommunities.jive-mobile.com

图10.15

由图 10.15 可知,作者的计算机首先发出了一条针对域名 www.cisco.com 的 DNS A 记录查询请求(在图中被标记为①)。然后,又依次针对域名 ap.ff.avast.com、www.static-cisco.com、ciscosystemsinc.tt.omtrdc.net、news-tags.cisoc.com、products.mcisco.com、newsroom.cisco.com,各发出了一条 DNS 查询数据包(这些 DNS 查询数据包在图中被标记为②~⑦,其中, ap.ff.avast.com 是 Avast antivirus 的 Web 防护服务器)。

借助于 IO Graph 工具(如图 10.16 所示),可以了解到本次打开网站 www.cisco.com 时,发生的上述 7 次 DNS 解析时间最长为 600 毫秒,这也印证了为了什么每次访问 Cisco 官网主页时,浏览器总要花费好几秒钟才能加载完所有内容。

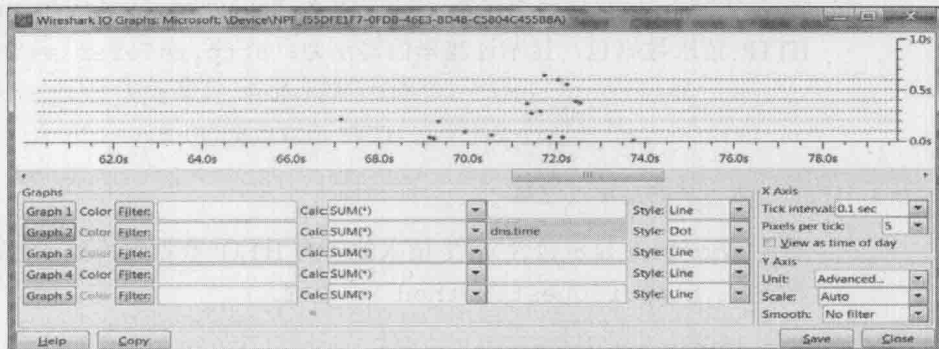


图10.16

10.5 筛选 HTTP 流量

本节将介绍如何配置 Wireshark 显示过滤器以筛选 HTTP 流量。

10.5.1 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（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。

10.5.2 操作方法

配置 HTTP 显示过滤器的方法包括：直接在抓包主窗口的 Filter 输入栏内输入 HTTP 显示过滤表达式；在抓包主窗口的数据包结构区域内，将 HTTP 数据包的某个 HTTP 属性值指定为显示过滤器的过滤条件；点击抓包主窗口中过滤器工具条上的 **Expression** 按钮，在弹出的 Filter Expression 窗口中构造 HTTP 显示过滤器（具体做法详见本书第 3 章）。

用来筛选 HTTP 流量的显示过滤器分为以下几类。

➤ 基于名称的 HTTP 显示过滤器。

- 要让 Wireshark 只显示访问某指定域名（www.packtpub.com）的 HTTP 请求数据包，显示过滤器的写法为：`http.host == "www.packtpub.com"`^①。
- 要让 Wireshark 只显示访问包含了指定字符串的域名（比如，包含 PacktPub 的域名）的 HTTP 请求数据包，显示过滤器的写法为：`http.host contains "packtpub"`。
- 要让 Wireshark 只显示 Referer 头部内容为“http://www.packtpub.com/”的 HTTP 请求数据包，显示过滤器的写法为：`http.referer == "http://www.packtpub.com/"`（亦即让 Wireshark 只显示从 http://www.packtpub.com/ 链接过来的所有 HTTP 请求数据包）^②。

➤ 基于 HTTP 请求方法的显示过滤器。

- 要让 Wireshark 显示包含 GET 请求的所有 HTTP 数据包，显示过滤器的写法为：`http.request.method == GET`。

① 译者注：原文是“Requests to a specific website: `http.host == "www.packtpub.com"`”，译文酌改。

② 译者注：原文是“Requests that were forwarded from PacktPub: `http.referer == "http://www.packtpub.com/"`”。

- ▣ 要让 Wireshark 显示所有 HTTP 请求数据包，显示过滤器的写法为：`http.request`。
- ▣ 要让 Wireshark 显示所有 HTTP 响应数据包，显示过滤器的写法为：`http.Response`。
- ▣ 要让 Wireshark 显示包含所有 HTTP 数据包，但包含 GET 方法的 HTTP 请求数据包除外，显示过滤器的写法为：`http.request and not http.request.method == GET`。

➤ 基于 HTTP 状态码的显示过滤器。

- ▣ 要让 Wireshark 显示包含 HTTP 错误状态码的 HTTP 响应数据包，显示过滤器的写法为：`http.response.code >= 400`。
- ▣ 要让 Wireshark 只显示包含 HTTP 客户端错误状态码的 HTTP 响应数据包，显示过滤器的写法为：`http.response.code >= 400 and http.response.code <= 499`。
- ▣ 要让 Wireshark 只显示包含 HTTP 服务器端错误状态码的 HTTP 响应数据包，显示过滤器的写法为：`http.response.code >= 500 and http.response.code <= 599`。
- ▣ 要让 Wireshark 只显示状态码为 404 的 HTTP 响应数据包，显示过滤器的写法为：`http.response.code == 404`。



注意

在配置由非特殊字符作为参数值的 HTTP 显示过滤器（比如，`http.host == packtpub`）时，可以不用为参数值加“”。但在配置由特殊字符作为参数值的 HTTP 显示过滤器（比如，`http.host == "packtpub\r\n"`）时，就必须为参数值加“”。

10.5.3 幕后原理

本小节将引领读者浏览一遍 HTTP 协议的细节。

HTTP 方法

RFC 2616 定义了几种主要的 HTTP 请求方法。在 RFC 2616 发布之后，又有数种 HTTP 请求方法问世，这些 HTTP 请求方法刊登在了 RFC 2616 的升级版（RFC 2817、RFC 5785、RFC 6266 和 RFC 6585）以及其他的 HTTP 标准（RFC 2518、3252 和 5789）中。

以下所列为定义于 RFC 2616 的几种基本的 HTTP 请求方法。

- **OPTIONS:** HTTP 客户端可使用该方法让 Web 服务器告知其所支持的功能。
- **GET:** HTTP 客户端可使用该方法请求 Web 服务器发送某个资源。
- **HEAD:** 与 GET 方法类似。借助于该方法, HTTP 客户端可在不获取实际资源的情况下, 让 Web 服务器发送资源的概况信息。
- **POST:** HTTP 客户端可利用该方法向 Web 服务器传送数据。比方说, 当使用 webmail 时, 就会调用该方法去传送邮件操作命令。
- **DELETE:** 借助于该方法, HTTP 客户端可请求 Web 服务器删除由 URL 指定的资源。
- **PUT:** HTTP 客户端可利用该方法向 Web 服务器写入数据。有些 Web 服务器允许用户通过 PUT 方法, 在其上创建页面, 但在如此操作之前, 一般要通过用户名/密码认证。
- **TRACE:** 当 HTTP 客户端发起 HTTP 请求时, HTTP 请求报文可能会穿越防火墙、代理服务器或网关等设备, 这些设备可能会修改原始的 HTTP 请求数据包中的内容。借助于 TRACE 方法, HTTP 客户端就能让 Web 服务器弹回 (loopback) 一条 TRACE 响应报文, 其中会携带后者实际收到的 HTTP 请求报文。HTTP 客户端可借此了解原始 HTTP 请求报文是否被损坏或修改过。
- **CONNECT:** 用来连接代理设备。

HTTP 状态码

HTTP 状态码共分 5 类, 如下所列。

类 别	名 称	含 义
1xx	信息状态码	用来表示一般性信息
2xx	成功状态码	用来表示由 HTTP 客户端所请求执行的动作已被 Web 服务器成功接收、接受或处理
3xx	重定向状态码	用来告知 HTTP 客户端使用其他位置来访问其所感兴趣的资源
4xx	客户端错误状态码	用来表示 HTTP 客户端错误
5xx	服务器端错误状态码	用来表示 HTTP 错误

HTTP 状态码的完整列表详见以下链接: <http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml>。

10.5.4 拾遗补缺

使用 Wireshark 分析 HTTP 流量时, 在抓包主窗口的数据包结构区域内, 经常可以看见某

些 HTTP 数据包的 Hypertext Transfer Protocol 树状区域下会多一个名为 **Line-based text data: text/html** 的结构, 如图 10.17 所示。

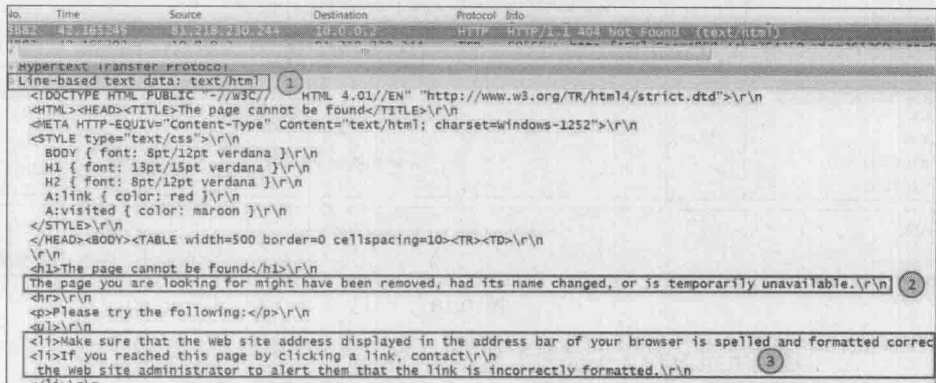


图 10.17

该结构的名称 **Line-based text data: text/html** 在图 10.17 中被标记为①, 点击其前面的“+”号, 将会暴露出由 Web 服务器返回的错误信息 (图 10.17 中的②和③)。

10.6 配置 Preferences 窗口中 protocol 选项下的 HTTP 协议参数

使用 Wireshark 抓取并分析 HTTP 协议流量之前, 可调整 Preferences 窗口中 Protocol 选项下 HTTP 协议的某些参数。对这些参数的调整, 会影响到 Wireshark 对 HTTP 协议流量的解析, 本节将讲解如何调整这些参数。

10.6.1 准备工作

启动 Wireshark 软件, 按下面内容的指示行事。

10.6.2 操作方法

1. 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, 选择 HTTP 协议, 如图 10.18 所示。
 - 在图 10.18 所示的 Preference 窗口中, 那 4 个复选框是默认勾选的。只要勾选了那个 4 个复选框, 即便 HTTP 协议数据包在 IP 层被分片传送, Wireshark

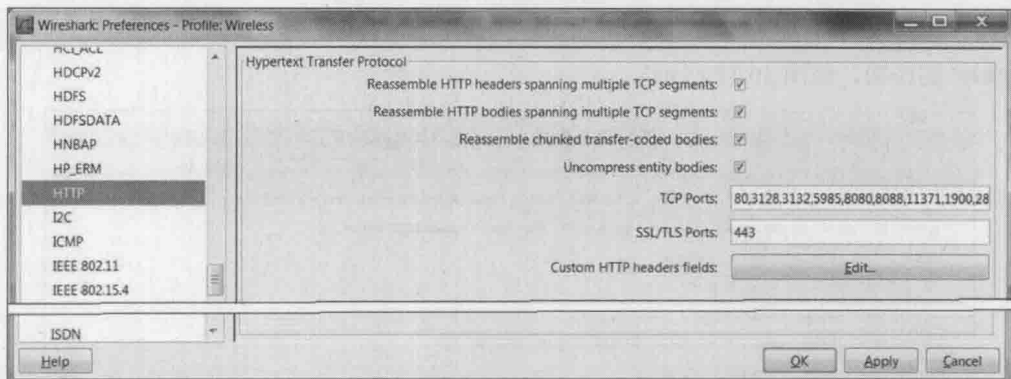


图10.18

也会重组 HTTP 头部和主体。

- 图 10.18 所示的 Preference 窗口中 **TCP Ports** 字样旁，有一个输入栏，其中默认列出了若干 TCP 端口号，Wireshark 会把发往/源于这些端口号的 TCP 流量当做 HTTP 流量来解码。在某些情况下，要想用其他 TCP 端口来跑 HTTP 流量，且希望 Wireshark 将发往/源于此端口号的 TCP 流量视为 HTTP 流量，请将该 TCP 端口号添加进 **TCP Ports** 之后的输入栏。
- 在 **SSL/TLS Ports** 字样旁的输入栏内，默认只有 TCP 443 端口，该输入栏的作用等同于 **TCP Ports** 字样旁的输入栏，要想用其他 TCP 端口来跑 HTTPS 流量，且希望 Wireshark 将发往/源于此端口号的 TCP 流量视为 HTTPS 流量，需将该 TCP 端口号填入此输入栏。

Custom HTTP headers fields 身旁的 Edit 按钮

在图 10.18 所示的 Preference 窗口中 **Custom HTTP headers fields** 字样旁，有一个 **Edit** 按钮，可借助于该按钮功能来创建 http.header 旗下的 HTTP 显示过滤参数。

试举一例，请看图 10.19。

要想让 Wireshark 直接将 HTTP 响应数据包中的 Age 头部作为 HTTP 显示过滤参数来引用，请按以下步骤行事。

- 点击 HTTP preference 窗口中 **Custom HTTP headers fields** 之后的 **Edit** 按钮（图 10.20 中被标记为①的区域）。
- 在弹出的 Custom HTTP headers fields Table 窗口中，点击 **New** 按钮（图 10.20 中被标记为②的区域）。

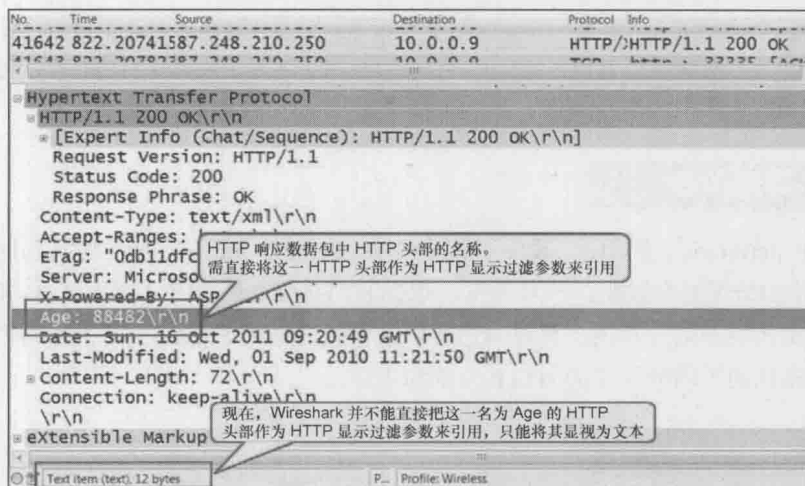


图 10.19

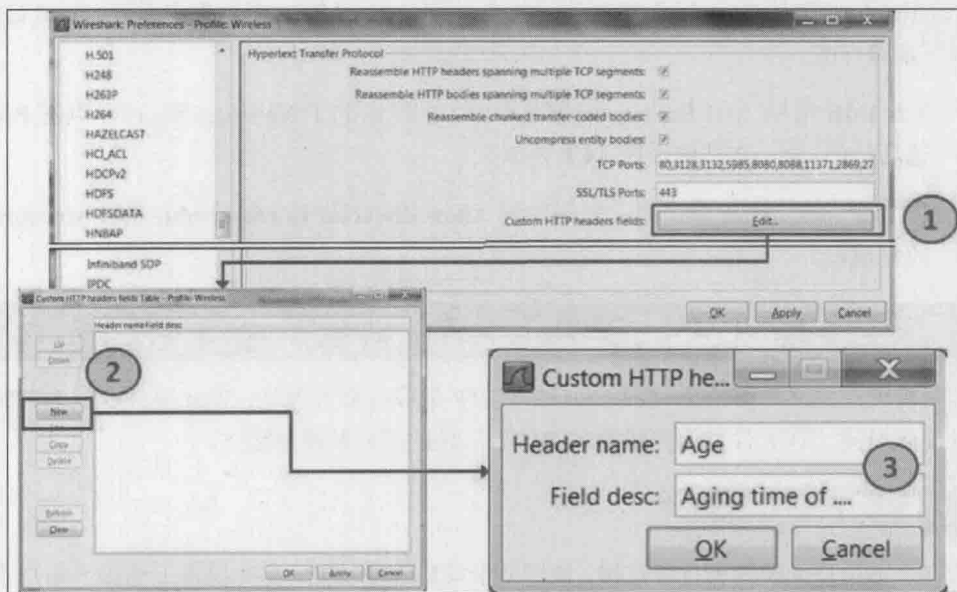


图 10.20

3. 在弹出的小窗口的 **Header name** 输入栏内（图中被标记为③的区域）输入 HTTP 头部的名称 Age(区分大小写); 在 **Field desc** 字段内输入相关注释信息, 作者输入的是 Aging time of ...。
4. 先点击小窗口中的 **OK** 按钮, 再点击 Custom HTTP header fields Table 窗口中的 **OK** 按钮。
5. 现在, 在 Wireshark 抓包主窗口的 Filter 输入栏内, 就可以用显示过滤参数 `http.header.Age` 来直接引用 Age 头部了。



注意

若 HTTP 数据包中包含有私自定义的头部, 且需要根据此类 HTTP 头部来筛选 HTTP 流量时, 便可以使用上述方法来构造 HTTP 显示过滤参数。

10.6.3 幕后原理

在 HTTP preference 窗口中, 设立了几个含 reassemble (重组) 字样的复选框。设立的原因是, IP 分片的情况时有发生, 一旦发生, 承载 HTTP 流量的 TCP 报文段也会分布在隶属于同一个 IP 数据包的不同分片内。只要勾选了那几个复选框, Wireshark (在解码时) 会重组分布在各 IP 数据包的不同分片内的 HTTP 头部和主体。

10.6.4 拾遗补缺

通常, Wireshark 只会把目的端口号为 80, 且包含有效 HTTP 头部的 TCP 数据包识别为 HTTP 数据包^①。要想让 Wireshark 把所有目的端口号为 80 的 TCP 流量都识别为 HTTP 数据包, 请按以下步骤行事。

1. 点击 **Edit** 菜单下的 **Preferences** 菜单项, 在弹出的 Preferences 窗口中, 点击 **Protocol** 选项前的+号, 在其中选择 **TCP** 协议。
2. 在 TCP preference 窗口中, 取消勾选 **Allow dissector to reassemble TCP streams** 之后的复选框。

10.7 HTTP故障分析

本章将重点关注如何分析 HTTP 故障。HTTP 故障的原因包括: Web 服务器或 HTTP 客户端主机反应迟钝、TCP 性能问题以及本章将要呈现的某些其他原因。

10.7.1 准备工作

若有个别用户反映网页打开过慢, 请借助于端口镜像功能, 将交换机上连接该用户主机的端口的流量重定向给 Wireshark 主机; 若用户普遍反映网页打开过慢, 则需要将交换机上连接 Internet 链路的端口的流量重定向给 Wireshark 主机。

10.7.2 操作方法

网页打开过慢的原因有很多, 请按以下思路来查找原因。

^① 译者注: “Usually Wireshark shows dissected packets with port 80 as HTTP only if it sees a valid HTTP header”。对于这种文字, 译者直接无语。

1. 首先, 要确保 Internet 链路没有过载。其次, 应观察连接 Internet 链路的交换机端口的统计信息, 确保未发生任何与通信链路有关的问题 (比如, 收到了错包或存在丢包)。
2. 要判断网页打开过慢是否与 TCP 性能问题有关, 请按以下步骤行。
 - 检查 Wireshark 抓包文件, 看看是否抓到了数量众多的身背 duplicate ACK 或 retransmissions 等字样的 TCP 报文段 (在正常情况下, 身负上述字样的 TCP 报文段的数量不应超过 TCP 报文段总量的 0.5%~0.8%)。
 - 检查 Wireshark 抓包文件, 看看是否抓到了重置 HTTP 连接的 TCP 报文段 (RST 标记位置 1 的 TCP 报文段)。若有, 则有可能是防火墙在捣鬼或是目标 Web 站点有访问限制。
3. 要判断网页打开过慢是否归因于 DNS 故障, 可能存在的 DNS 故障如下所列。
 - DNS 解析缓慢。
 - DNS 服务器无法解析域名, 或有待查询的域名不正确。
4. 经过一翻排查之后, 若发现网页打开过慢与上述问题无关, 请将排障目标对准 HTTP 协议。

**注意**

排除网络故障时, 应将视野放宽到整个网络环境甚至是整个 IT 环境。HTTP 与 TCP 息息相关, 而 DNS 解析速度则与用户对互联网应用的体验密不可分。因 Web 服务器反应迟钝而导致的 TCP 重传, 会使得网页打开过慢; 而 DNS 服务器反应迟钝, 也会使得网页打开看似变慢。因此, 需遵循上述思路, 按部就班地解决问题。

若用户普遍反应网页打开过慢, 应检查如下事宜。

1. 检查通信链路 (Internet 链路) 是否过载。
2. 检查通信链路上的传输延迟 (在用户主机上 ping Internet 上的 Web 站点)。
3. 设法查明 Web 站点传回的 HTTP (响应数据包中包含的) 信息状态码。当使用浏览器打开网页时若发生了 HTTP 访问错误, 浏览器一般都会显示出由 Web 服务器传回的 HTTP 错误状态码, 但并非总是如此。
4. 在 Wireshark 抓包主窗口的 Filter 输入栏内输入显示过滤表达式 `http.response >= 400`, 看看 Wireshark 抓到了多少含有 HTTP 错误状态码的数据包。在本章后续内容中, 会举几个与此有关的重要示例。

HTTP 信息状态码

状态码	原因短语	解 释	应对措施
100	Continue	表示 HTTP 请求成功执行完毕，会话可以继续	
101	Switching protocols	表示 Web 服务器根据 HTTP 客户端的指定，将协议切换为 Upgrade 头部中列出的协议	

HTTP 成功状态码

状态码	原因短语	解 释	应对措施
200	OK	HTTP 请求正常	
201	Created	表示 HTTP 请求已经执行，新的资源已经创建	
202	Accepted	表示 HTTP 请求已被接受，仍在处理	
203	Non-Authoritative information	表示实体首部所含信息并非来自于源端服务器，而是来自资源的一份副本	
204	No content	表示 HTTP 响应数据包中只包含若干 HTTP 头部和一状态行，不含实体的主体部分	
205	Reset content	Web 服务器会用该状态码告知浏览器清除当前页面中所有表单元素	
206	Partial content	表明 HTTP 请求部分成功执行	

HTTP 重定向状态码

状态码	原因短语	解 释	应对措施
300	Multiple choices	当 Web 服务器返回该状态码时，便表示(HTTP 客户端所请求的) URL 实际指向多个资源(譬如，所请求的 HTML 文件有英语和法语两个版本)。Web 服务器在返回该状态码的同时，还会提供一个选项列表，让用户选择	
301	Moved permanently	表示 HTTP 客户端所请求的资源被永久性转移，Web 服务器应在 HTTP 响应数据包的 Location 头部中包含资源目前所处的位置(URL)	
302	Moved temporarily (found)	表示 HTTP 客户端所请求的资源被临时转移，Web 服务器应在 HTTP 响应数据包的 Location 头部中包含资源目前所处的位置(URL)，但将来对资源的请求应使用原先的 URL	在遇到这种情况时，在 Web 浏览器中通常只会显示原因短语 Found。然后，Web 浏览器会另发一个 HTTP 请求数据包，其 GET 方法所引用的 URL 将指向资源目前所处的位置
303	See other	Web 服务器会用此代码告知 HTTP 客户端：请用另一 URL 来获取资源。新的 URL 位于 HTTP 响应数据包中的 Location 头部	
304	Not modified	若 HTTP 客户端发起的是条件 GET 请求，且所请求的资源最近未被修改，Web 服务器便会返回此状态码	
305	Use proxy	Web 服务器用此代码来表示必须通过某一代理设备来访问所请求的资源，代理的位置位于 HTTP 响应数据包中的 Location 头部	请检查相关代理设备的情况

HTTP 客户端错误状态码

状态码	原因短语	解释	应对措施
400	Bad request	表示 Web 服务器无法识别 HTTP 请求数据包中的语法。HTTP 客户端在重发 HTTP 请求之前, 应做修改	请检查所要访问的 Web 站点的域名或 IP 地址是否正确
401	Authorization required	HTTP 客户端对资源的访问遭 Web 服务器拒绝, 因为前者未能通过认证	请检查用户名/密码
402	Payment required	预留供将来使用	
403	Forbidden	表明 HTTP 请求遭 Web 服务器拒绝, 这可能是因为 Web 服务器有访问限制	请检查访问凭证。另外, 也有可能是因为 Web 服务器负载过重
404	Not found	表示所请求的资源不存在 (于 Web 服务器)	这既有可能是因为资源遭到了删除, 或原本就不存在, 也有可能是因为 URL 拼写错误
405	Method not allowed	表示 Web 服务器不支持或不允许 HTTP 客户端用来请求资源的 HTTP 方法	
406	Not acceptable	HTTP 客户端可通过参数来说明自己愿意接受何种类型的实体。当 Web 服务器上的资源与 HTTP 客户端可接受的 URL 不匹配时, 便会返回此状态码	请检查/升级 Web 浏览器
407	Proxy authentication	类似于状态码 401, 但为客户端代理服务器所用, 该服务器的作用是对 HTTP 客户端所要访问的资源进行认证	HTTP 客户端在访问资源之前需先通过代理服务器的认证
408	Request timed out	Web 服务器在处理 HTTP 请求时若超出了所允许的时间, 便会返回此状态码	检查响应时间及网络负载状态
409	Conflict	表示由 HTTP 客户端所提交的 HTTP 请求因与某些既定的规则相冲突, 而未能完成	可能是通过 HTTP 上传的文件旧于 Web 服务器所存, 或存在与之类似的问题。请检查 HTTP 客户端的所作所为
410	Gone	类似于状态码 404, 只是 HTTP 客户端通过 URL 所请求的资源以前确实存在	一般都是 Web 服务器问题, 既有可能是文件被删除, 也有可能是文件存放位置发生了改变
411	Content length required	表示 HTTP 客户端发出的 HTTP 请求数据包缺少 Content-Length 头部, 但 Web 服务器要求在 HTTP 请求数据包中包含这一头部	Web 浏览器与 Web 服务器不兼容, 请升级 Web 浏览器
412	Precondition failed	若 HTTP 客户端发起的是条件 GET 请求, 且启用某一条件无法满足时, Web 服务器会返回此状态码	WEB 浏览器与 Web 服务器不兼容, 请升级 Web 浏览器
413	Request entity too long	当 HTTP 客户端所发送的 HTTP 请求数据包中的实体的主体部分过长, Web 服务器无法处理时, 便会返回此状态码	Web 服务器限制

续表

状态码	原因短语	解释	应对措施
414	Request URI too long	当 HTTP 客户端所发送的 HTTP 请求数据包中的 URL 过长, Web 服务器无法处理时, 便会返回此状态码	Web 服务器限制
415	Unsupported media type	若 Web 服务器不支持或无法识别 HTTP 客户端所发 (HTTP 请求数据包中的) 实体内容的类型, 则会返回此状态码	Web 服务器限制

图 10.21 呈现了 Wireshark 识别出的 HTTP 客户端错误的简单示例。要想让 Wireshark 生成图 10.21 所示内容, 请按以下步骤行事。

1. 在 Wireshark 抓包主窗口的数据包列表区域中, 选中一个身背 HTTP 4xx 状态码的数据包, 点击右键。
2. 在弹出的右键菜单中点击 **Follow TCP Stream** 菜单项, 将会看到图 10.21 所示的窗口。

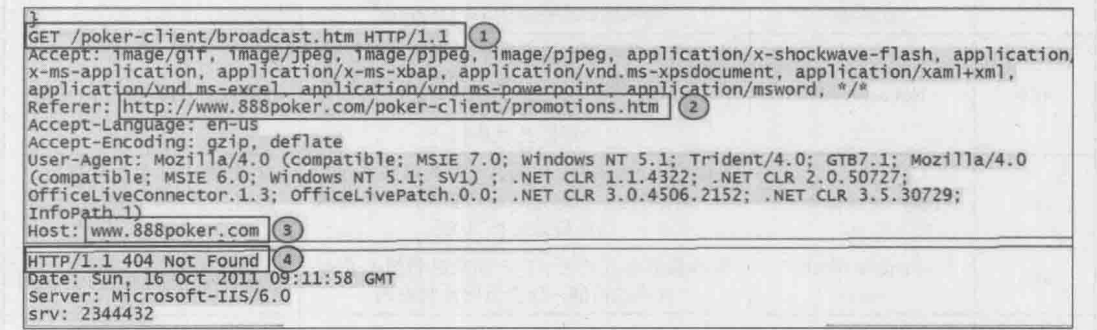


图10.21

3. 现在, 来解读一下图中所示内容。
 - HTTP 客户端所要访问的网址为 `www.888poker.com//poker-client/broadcast.htm` (在图 10.21 中分别被标记为①和③)。
 - 访问此网址的 HTTP 请求经由网址 `http://www.888poker.com/poker-client/promotions.htm` 链接而来 (在图 10.21 中被标记为②)。
 - Web 服务器返回的 HTTP 状态码为 **404 Not Found** (在图 10.21 中被标记为④)。

需要澄清的是, 作者从不赌博, 只是在用 Wireshark 执行排障任务时顺手截了个屏 (`www.888poker.com` 是一个赌博网站)。

HTTP 服务器错误状态码

状态码	原因短语	解 释	应对措施
500	Internal server error	Web 服务器遭遇意外情况时，便会返回此状态码。所谓意外情况，是指服务器因出错而无法完成 HTTP 客户端对某一 URL 的访问请求	当 Web 服务器上运行有 CGI 程序时，Perl 代码中的错误会导致其返回此状态码
501	Not implemented	表示 Web 服务器未能执行 HTTP 客户端的访问请求	Web 服务器故障
502	Bad gateway	充当代理或网关的服务器从请求响应链中的下一个环节收到无效响应（比如，该服务器无法连接到其父网关）时，便会返回此代码	服务器故障
503	Service unavailable	表示有待访问的服务或请求访问的资源当前处于失效状态	Web 服务器故障
504	Gateway timeout	类似于状态码 408，但返回此状态码的是网关或代理设备	服务器宕机或停止响应
505	HTTP version not supported	表示 Web 服务器不支持（HTTP 客户端用来）与其通信的 HTTP 协议的版本	Web 服务器不支持（HTTP 客户端使用的）HTTP 协议的版本

浏览网页时，有诸多原因会导致 Web 浏览器提示“service unavailable (code 503)”错误。现在来举一个用 Wireshark 排除此类故障的示例。有一小型办公网络，里面总有人反映：用 Web 浏览器打开 Facebook 的主页，一点问题都没有，但打开之后，只要点主页里任何一个链接，就会弹出一个提示“service unavailable”的新页面^①。由图 10.22 所示的 Wireshark 截屏可知，是网络中的一台防火墙在捣鬼。

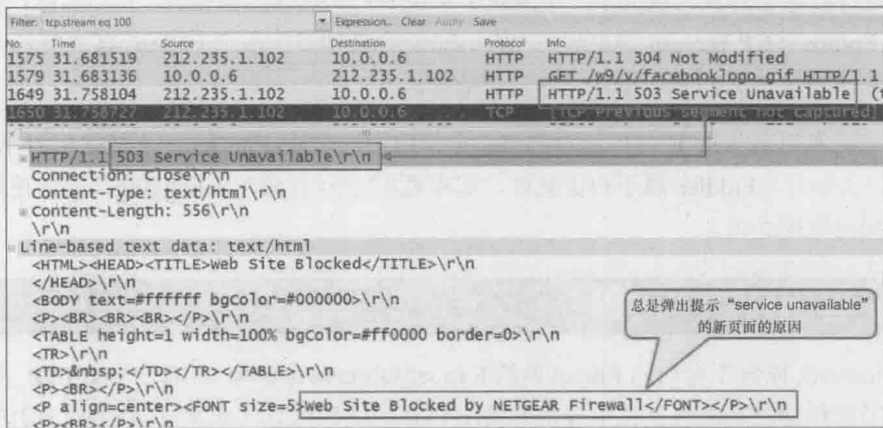


图10.22

① 译者注：原文是“In the following example there is a small office that has the following complaint: they can browse Facebook, but the moment they click on a link on this site, they get the new page as blocked”。真不敢想象，连故障现象都描述不清的人是怎么处理故障的？

10.7.3 幕后原理

通过 Web 浏览器，以 HTTP 方式打开网页的同时，用 Wireshark 抓包，能观察到下述情况。

1. TCP 连接的建立过程（TCP 三次握手过程）。
2. 身负 GET 字样的 HTTP 请求数据包。
3. 用来承载（下载）数据的 TCP 数据包。



注意

在大多数情况下，用 Web 浏览器访问某个网站的主页时，可能会建立 10 条以上的 HTTP 连接。比方说，在访问某些新闻站点的主页（比如，www.cnn.com、www.foxnews.com、www.bbc.co.uk）时，Web 浏览器会同时打开多条 HTTP 连接，分别用来加载其经济、体育、生活、天气等各个“频道”的数据。甚至会出现访问某个页面时，Web 浏览器打开上百条 HTTP 连接加载数据的情况。

对于一个需要通过多条 HTTP 连接才能加载完毕的 Web 页面来说，在打开每条 HTTP 连接时，都会衍生出 DNS 查询/响应数据包、TCP “三次握手”报文段、包含 HTTP GET 方法的 HTTP 请求数据包，但 Web 浏览器中显示的内容都是由后续的 TCP 数据包承载。

10.7.4 拾遗补缺

用 Wireshark 分析抓到的 HTTP 数据包时，若在抓包主窗口的数据包内容区域看不到有用的信息，请在数据包列表区域选中一个隶属于某条 HTTP 连接的数据包，点击右键，选择右键菜单中的 **Follow TCP Stream** 菜单项^①。Wireshark 会立刻弹出一个 Follow TCP Stream 窗口，其中会显示出通过该 HTTP 连接传递的全部数据。

还有一款常用来分析 HTTP 流量的流量分析攻击，名叫 **Fiddler**，下载链接为 <http://fiddler2.com/>。Fiddler 属于自由软件，其主要用途为 HTTP debugging，如何使用该工具不在本书探讨范围之内。

10.8 导出 HTTP 对象

在 Wireshark 抓包主窗口的 **File** 菜单的 **Export Objects** 菜单项下，有一个 **HTTP** 子菜单项，可利用该子菜单功能从抓包文件中导出有关 HTTP 的统计信息（即通过 HTTP 访问的 Web 站点信息和资源信息）^②。

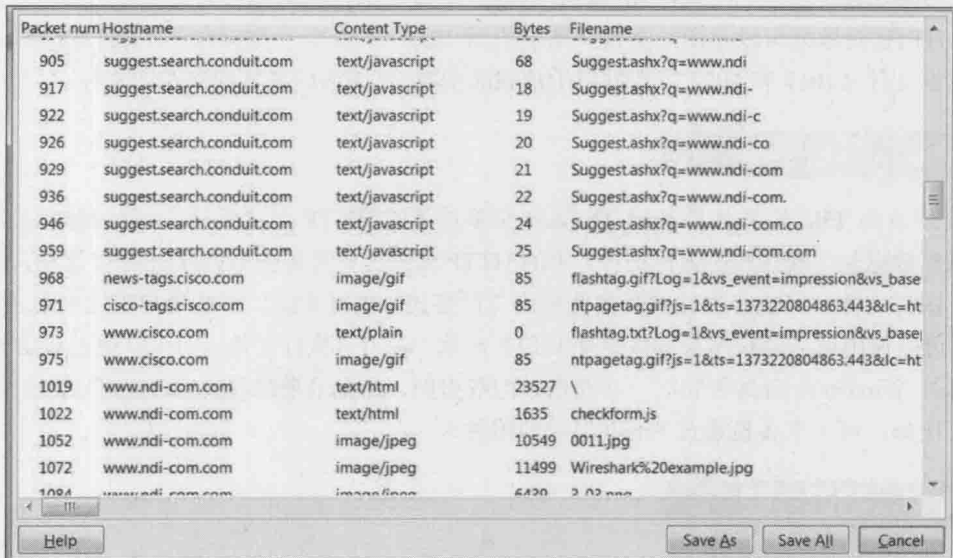
^① 译者注：原文是“When you don't see anything in the packet details pane, right-click on a packet and choose Follow TCP stream。”作者的表达能力真让人着急。

^② 译者注：原文是“Exporting HTTP objects is a simple feature for exporting HTTP statistics—websites and files accessed by HTTP”。

10.8.1 操作方法

要想从 Wireshark 抓包文件中导出 HTTP 对象，请按以下步骤行事。

1. 请在使用 Wireshark 抓包的同时（或打开一个之前保存的抓包文件），选择抓包主窗口的 **File** 菜单，点击其 **Export Objects** 菜单项下的 **HTTP** 子菜单项。**HTTP Object List** 窗口会立刻弹出，如图 10.23 所示。



Packet num	Hostname	Content Type	Bytes	Filename
905	suggest.search.conduit.com	text/javascript	68	Suggest.ashx?q=www.ndi
917	suggest.search.conduit.com	text/javascript	18	Suggest.ashx?q=www.ndi-
922	suggest.search.conduit.com	text/javascript	19	Suggest.ashx?q=www.ndi-c
926	suggest.search.conduit.com	text/javascript	20	Suggest.ashx?q=www.ndi-co
929	suggest.search.conduit.com	text/javascript	21	Suggest.ashx?q=www.ndi-com
936	suggest.search.conduit.com	text/javascript	22	Suggest.ashx?q=www.ndi-com.
946	suggest.search.conduit.com	text/javascript	24	Suggest.ashx?q=www.ndi-com.co
959	suggest.search.conduit.com	text/javascript	25	Suggest.ashx?q=www.ndi-com.com
968	news-tags.cisco.com	image/gif	85	flashtag.gif?Log=1&vs_event=impression&vs_base
971	cisco-tags.cisco.com	image/gif	85	ntpagetag.gif?js=1&ts=1373220804863.443&lc=ht
973	www.cisco.com	text/plain	0	flashtag.txt?Log=1&vs_event=impression&vs_base
975	www.cisco.com	image/gif	85	ntpagetag.gif?js=1&ts=1373220804863.443&lc=ht
1019	www.ndi-com.com	text/html	23527	\
1022	www.ndi-com.com	text/html	1635	checkform.js
1052	www.ndi-com.com	image/jpeg	10549	0011.jpg
1072	www.ndi-com.com	image/jpeg	11499	Wireshark%20example.jpg
1094	www.ndi-com.com	image/jpeg	6420	2_02.png

图 10.23

2. 在 HTTP Object List 窗口中，会列出被访问过的 Web 站点的名称，以及各 Web 站点上被访问过的文件信息（包括每个文件的类型、大小、名称等）。
3. 可点击 HTTP Object List 窗口底部的 **Save As** 或 **Save All** 按钮，来保存文件数据。
4. 在 HTTP Object List 窗口的 **Content Type** 一栏中，可能会出现的内容如下所示。
 - 文件类型 **text/plain**、**text/html**、**text/javascript**：表示通过 HTTP 访问的文件类型为文本文件，若为 **text/javascript**，则需仔细检查，因为可能存在安全隐患。
 - 文件类型 **image/jpeg**、**image/gif**：表示通过 HTTP 访问的文件类型为图像，可用图片浏览工具来打开此类文件。
 - 文件类型 **application/json**、**application/javascript** 等：表示通过 HTTP 访问的文件类型为应用程序。

- Wireshark 可识别的其他文件类型。



注意

要想让 **File** 菜单中 **Export Objects** 菜单项下的 **HTTP** 子菜单项功能生效, 需先在 **TCP** 首选项设置窗口 (点击 **Edit** 菜单中的 **Preferences** 菜单项, 在弹出的 **Preferences** 窗口中, 点击左侧 **Protocol** 之前的+号, 选中 **TCP** 协议) 中点选 **allow subdissector to reassemble TCP streams** 之后的复选框, 激活 Wireshark 的 **TCP** 数据包重组功能。

还可点击图 10.23 中的 **Save All** 按钮, 选择一个目录来存储从抓包文件中导出的所有 **HTTP** 对象。**HTTP** 对象既可以是图片文件 (图中所示 1052 和 1072 号数据包所承载的内容), 也可以是文本文件 (1019 和 1022 号数据包所承载的内容), 还可以是其他格式的文件。

10.8.2 幕后原理

只要点击 **File** 菜单中 **Export Objects** 菜单项下的 **HTTP** 子菜单项, Wireshark 就会扫描当前所抓数据包 (或打开的抓包文件) 中的 **HTTP** 数据流, 对各种 **HTML** 对象 (比如, **HTML** 文档、图片文件、可执行文件以及其他可读文件格式) 进行重组, 好让用户将这些对象存盘。稍后, 可以使用适当的程序来读取这些 **HTTP** 对象 (若为可执行文件, 则可以通过双击的方式来执行)。Wireshark 所具备的这一功能是一把双刃剑, 既能用来窃取机密, 也可以起到备份的作用 (比如, 可用来备份通过 **Email** 发出的附件)。

10.8.3 拾遗补缺

下列软件也可以以图形化的方式来执行 Wireshark 所具备的上述功能:

- **Xplico** (<http://www.xplico.org/>);
- **NetworkMiner** (<http://www.netresec.com/?page=NetworkMiner>).



注意

用 **File** 菜单的 **Export Objects** 菜单项下的 **HTTP** 子菜单项功能监控 **HTTP** 流量时, 要是发现有人从路数不正的网站下载了可疑的应用程序, 且应用程序的文件名也十分可疑时, 请“百度 (Google)”一下, 这可能预示着危险 (第 14 章会对此展开深入探讨)。

10.9 HTTP数据流分析及Follow TCP Stream窗口

本书前文已经简单介绍了 Wireshark 软件的 **Follow TCP Stream** 特性, 该特性对网络监控非常有用, 能让网管人员窥探到所抓 **TCP** 数据流的内在。本节会讲解如何借助于 **Follow TCP Stream** 特性分析 **HTTP** 流量。

10.9.1 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，把受监控端口的流量重定向至 Wireshark 主机；启动 Wireshark 软件，选择正确的网卡，开始抓包。

10.9.2 操作方法

要想在 Wireshark 软件中打开 Follow TCP Stream 窗口，请按以下步骤行事。

1. 在 Wireshark 抓包主窗口的数据包列表区域中，从有待分析的那股 HTTP 数据流中选择一个数据包，单击右键。
2. 在弹出的右键菜单中点击 **Follow TCP Stream** 菜单项，即可让 Wireshark 只显示隶属于这股 HTTP 数据流的数据包了。在 Wireshark 抓包主窗口的 Filter 输入栏内，不但可以看见这股 HTTP 数据流在抓包文件中的编号，还会弹出图 10.24 所示的窗口。

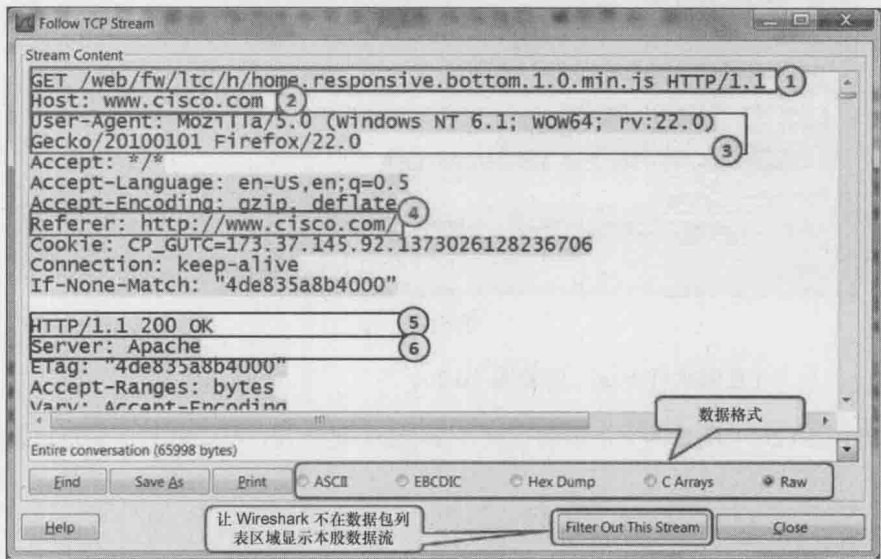


图 10.24

3. 借助于 Follow TCP Stream 窗口中的内容，可窥探到 HTTP 数据流的内在。由图 10.24 可知，这股 HTTP 数据流包含了以下内容。

- HTTP 客户端在执行 HTTP 请求操作时使用了 GET 方法（图 10.24 中的①）。
- HTTP 客户端所访问的主机名为 `www.cisco.com`（图 10.24 中的②）。
- 执行 HTTP 请求操作的浏览器为 Mozilla Firefox（图 10.24 中的③）。

- HTTP 请求数据包中 `referrer` 头部字段值为 `http://www.cisco.com` (图 10.24 中的④)。
- 在 HTTP 响应数据包中返回的信息状态码为 220 OK (图 10.24 中的⑤)。
- Web 服务器为 Apache Web 服务器 (图 10.24 中的⑥)。

4. 借助于 Follow TCP Stream 窗口，定位网络故障非常简单，再举两个例子。

- 可发现偷偷用 Kazaa (类似于 BT) 客户端“做种”的内网用户，如图 10.25 所示。

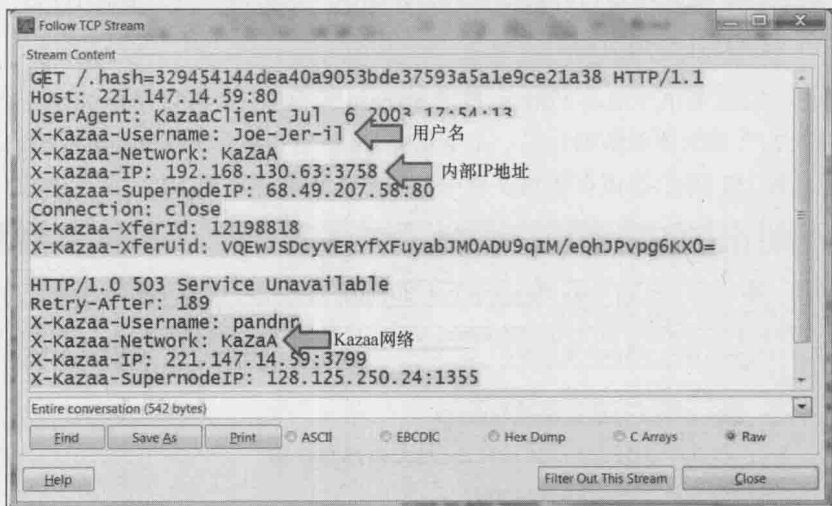


图10.25

- 可发现软件 bug，请看图 10.26。

只需把图 10.26 中圈里的文字放到百度 (Google) 里一搜，马上就可以知道这属于历史遗留性 bug。

5. 借助于 Follow TCP Stream 窗口，还可以发现：

- 错误或 bug 消息^①；
- 病毒或蠕虫——若在 Follow TCP Stream 窗口中看见了 `blast`、`probe` 或 `Xprobe` 之类的单词，则十有八九与病毒或蠕虫有关。要是看见以上述单词命名的可执行文件 (扩展名为 `.exe`)，则更要保持十二分地警惕 (与此有关的内容详见第 14 章)。

① 译者注：原文是“Error and bugs messages”，直译。

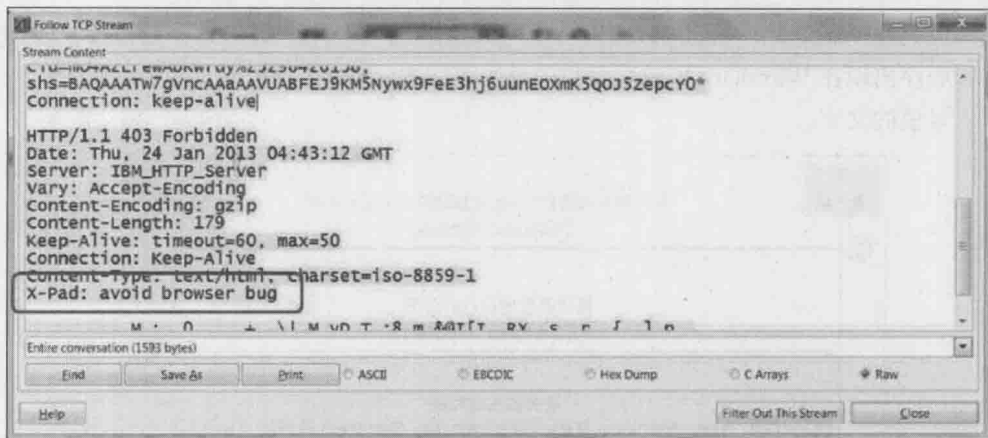


图10.26

10.9.3 幕后原理

通过 Follow TCP Stream 功能筛选出的 HTTP 数据流囊括了 HTTP 客户端与 Web 服务器之间从建立 HTTP TCP 连接的三次握手，到关闭 HTTP TCP 连接的四次握手之间的所有 TCP 报文段。该特性能够将 HTTP 数据流一股股的相互隔开，对排除网络故障大有裨益。

10.10 HTTPS协议流量分析——SSL/TLS基础

HTTPS 协议是 HTTP 协议的安全版本，HTTPS 中的那个 S 表示：HTTP 协议传输的安全性由 SSL/TLS（安全套接字层/传输层安全）来保证。在使用网银、Web 邮箱或基于 HTTP 协议但需安全性保障的应用时，就会用到 HTTPS 协议。

本节将介绍 HTTPS 协议的运作方式及其排障方法。

10.10.1 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，将受监控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。HTTPS 协议通过 TCP 443 端口来通信，应关注 Wireshark 抓到的源或目的 TCP 端号为 443 的所有流量。

10.10.2 操作方法

要使用 Wireshark 分析 HTTPS 协议流量，请按以下步骤行事。

1. 需掌握 HTTPS 握手阶段的步骤，与此有关的内容详见 10.10.3 节。

2. 应关注 Wireshark 抓到的 HTTPS 握手消息的顺序，应确保此类数据包按图 10.27 所示顺序出现在 Wireshark 抓包主窗口的数据包列表区域，这些数据包应身背图 10.27 中括号里的文字。



图10.27

3. 以下所列为 RFC 2246 中定义的各种常见的 HTTPS 告警 (alert) 消息。

- **close_notify**: 用来通知接收方，发送方将不会在该 HTTPS 会话上传输任何消息。在此之后，接收方将忽略收到的任何数据。
- **unexpected_message**: 当 HTTPS 客户端或服务器收到不应收到的消息时，便会返回此消息。这属于关键性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **bad_record_mac**: 当 HTTPS 客户端或服务器收到了包含错误 MAC (消息认证码) 的记录时，便会返回此消息^①。这属于关键性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **decryption_failed**: 当 HTTPS 客户端或服务器收到了以错误的方法加密的

① 译者注：所谓“记录” (record)，是指通过 HTTPS 会话传输的数据，每一条记录都是由记录头部+经过加密的数据+MAC 构成。

TLS 密文时，便会返回此消息。这属于关键性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。

- **record_overflow**: 当 HTTPS 客户端或服务器收到的 TLS 密文记录的长度超出所允许的范围时，便会返回此消息。这属于致命性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **decompression_failure**: 当 HTTPS 客户端或服务器无法对收到的记录进行解压缩时，便会返回此消息。这属于关键性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **handshake_failure**: 表示 HTTPS 客户端和服务器在握手时无法就使用的安全参数达成一致意见。这属于关键性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **bad_certificate**: 表示证书损坏、签名不正确或相关类似错误。
- **unsupported_certificate**: 表示接收方不支持发送方发出的证书类型。
- **certificate_revoked**: 表示证书被其签发者撤销。
- **certificate_expired**: 表示证书过期。
- **certificate_unknown**: 表示证书不被接受，但原因不明。
- **illegal_parameter**: 表示收到的握手消息中有字段越界或与别的字段不一致。这属于关键性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **unknown_ca**: 表示收到了有效证书，但因与已知的受信 CA 不匹配而不被接受。这属于致命性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **access_denied**: 表示收到了有效证书，但证书中的身份通不过访问控制检查。
- **decode_error**: 当 HTTPS 客户端或服务器收到的 SSL/TLS 消息过长或其中有字段越界，以至于无法解码时，便会返回此消息。这属于关键性错误告警消息，表示 HTTPS 客户端或服务器在实现上存在重大缺陷。
- **decrypt_error**: 表示握手加密操作失败，可能的情况有：签名验证失败、密钥交换失败、finished 消息验证失败等。

- **export_restriction**: 表示 SSL/TLS 实现违反出口限制。
- **protocol_version**: 此告警消息由 HTTPS 服务器发送, 表示客户端使用了无法识别的协议版本。
- **insufficient_security**: 此告警消息由 HTTPS 服务器发送, 表示客户端使用的加密套件的比服务器所要求的要弱。
- **internal_error**: 表示此告警消息的发送方遭遇内存分配或硬件故障等内部错误。
- **user_canceled**: 表示了 HTTPS 握手操作被用户取消, 并非协议故障。
- **no_renegotiation**: HTTPS 客户端或服务器都可以发出此消息, 来响应初始握手之后的 hello 请求消息。

只要遭遇上面提及的任何一种 HTTPS 告警信息, HTTPS 连接都无法建立。

10.10.3 幕后原理

SSL 和 TLS 协议都能用来保证某些特殊应用程序 (比如, HTTP、SNMP、Telnet 等) 的安全性。SSL 版本 1、2、3 是 Netscape 公司于 20 世纪 90 年代中期为其 Navigator 浏览器开发的, 而 TLS 协议则是 IETF 的标准协议, 先后定义于 RFC 2246、RFC 4492、RFC 5246 和 RFC 6176。TLS1.0 于 1999 年 1 月以 SSL 3.0 升级版的形式首次在 RFC 2246 中露面。

TLS 握手协议用来建立 TLS 连接, 其规程如下所列。

1. 服务器与客户端相互交换 Hello 消息, 就用来保护数据的加密算法的选定达成一致意见, 同时交换用来生成密钥的随机数。
2. 服务器与客户端相互交换加密参数, 就预主密钥的选用达成一致意见。
3. 服务器与客户端相互交换证书及加密信息, 彼此认证。
4. 服务器与客户端根据预主密钥生成主密钥, 同时交换随机数。
5. 服务器与客户端验证对方是否计算出了相同的安全参数, 同时验证 TLS 握手本身有未被攻击者所乘。

上述规程按以下次序执行。

1. 服务器与客户端选择加密算法。
 - 客户端发出 **Client Hello** 消息 (图 10.28 中用①来标记的数据包)。

- 服务器发出 **Server Hello** 消息（图 10.28 中用②来标记的数据包）。
- 2. 服务器与客户端相互认证，交换密钥（图 10.28 中用③和④来标记的数据包）。
- 3. 握手完成（图 10.28 中用⑤来标记的数据包）。

No.	Time	Source	Destination	Protocol	Info
157	16.866912	10.0.0.3	173.194.34.86	TCP	62900 > https [SYN] Seq=0 win=8192 Len=0 MSS=14
158	16.953453	173.194.34.86	10.0.0.3	TCP	https > 62900 [SYN, ACK] Seq=0 Ack=1 win=62920
159	16.953528	10.0.0.3	173.194.34.86	TCP	62900 > https [ACK] Seq=1 Ack=1 win=66792 Len=0
160	16.954763	10.0.0.3	173.194.34.86	① TLSv1	Client Hello
161	17.040545	173.194.34.86	10.0.0.3	TCP	https > 62900 [ACK] Seq=1 Ack=173 win=64000 Len=
162	17.043587	173.194.34.86	10.0.0.3	② TLSv1	Server Hello
163	17.043715	173.194.34.86	10.0.0.3	③ TLSv1	Certificate, Server Key Exchange, Server Hello
164	17.043790	10.0.0.3	173.194.34.86	TCP	62900 > https [ACK] Seq=173 Ack=1936 win=66792
165	17.066539	10.0.0.3	173.194.34.86	④ TLSv1	Client Key Exchange, Change Cipher Spec, Encrypt
166	17.152661	173.194.34.86	10.0.0.3	⑤ TLSv1	New Session Ticket, Change Cipher Spec, Encrypt
167	17.154064	10.0.0.3	173.194.34.86	TLSv1	Application Data
168	17.154412	10.0.0.3	173.194.34.86	TCP	[TCP segment of a reassembled PDU]
169	17.154416	10.0.0.3	173.194.34.86	TLSv1	Application Data
170	17.154515	173.194.34.86	10.0.0.3	TLSv1	Application Data

图10.28

来看一下 HTTPS 协议的运作原理。由图 10.28 可知，要建立 SSL/TLS 连接，需先完成标准的 TCP 三次握手（图 10.28 中编号为 157~159 的数据包），本次 TLS 握手则始于编号为 160 的数据包。现在，来研究一下本次 TLS 握手的细节。

1. 客户端通过编号为 160 的数据包发出 **Client Hello** 消息，开始 TLS 握手协商。
2. 服务器回之以 **Server Hello** 消息（编号为 162 的数据包）。
3. 服务器向客户端发送证书（编号为 163 的数据包）。
4. 客户端接受证书，生成预主密钥。
5. 服务器创建主密钥，TLS 会话开始（可选消息，编号为 166 的数据包）。



注意

上面提到的步骤 5 是指 RFC 4507 中定义的一种机制，其作用是让 TLS 服务器在无需保存每客户端会话状态的同时，能迅速恢复会话。TLS 服务器可藉此机制，将会话状态封装进 ticket，并通过编号为 166 的数据包转发给客户端。客户端随后可使用自己获取到的 ticket 来恢复 TLS 会话。比方说，当使用浏览器恢复一条通往 webmail（Gmail、hotmail）账户的 TLS 连接时，就会用到该机制，这种情况在实际的使用中非常常见。

在步骤 4 和 5 之后，服务器与客户端将开始实际的数据通信^①。

现通过 Wireshark 来观察 TLS 握手过程的每一步。

① 译者注：上述对 SSL/TLS 的介绍大多都是作者东拼西凑，抄袭而来，全无逻辑可言（其实整本书都是如此）。由于 SSL/TLS 协议太过复杂，译者只能大致按照原文字面意思翻译，虽会在译文上略作改动，但无力作大篇幅的调整。

在步骤 1 中，客户端发出 **Client Hello** 消息，这也是 TLS 握手过程中生成的首个数据包。图 10.29 示出了 Client Hello 消息（其数据包编号为 160）所包含的内容。

No.	Time	Source	Destination	Protocol	Info
160	16.954763	10.0.0.3	173.194.34.86	TLSv1	Client Hello
Secure Sockets Layer					
TLSv1 Record Layer: Handshake Protocol: Client Hello					
Content Type: Handshake (22) ①					
Version: TLS 1.0 (0x0301)					
Length: 167					
Handshake Protocol: Client Hello ②					
Handshake Type: Client Hello (1)					
Length: 163					
Version: TLS 1.0 (0x0301) ③					
Random					
gmt_unix_time: Jul 9, 2013 07:28:40.000000000 Jerusalem Daylight Time ④					
random_bytes: 3c08aa71b98a6e6e6d339b12cc3fe5531647ec10020c65b5... ⑤					
Session ID Length: 0					
Cipher Suites Length: 72					
Cipher Suites (36 suites) ⑥					
Compression Methods Length: 1					
Compression Methods (1 method) ⑦					
Extensions Length: 50					
Extension: server_name					
Extension: elliptic_curves					
Extension: ec_point_formats					
Extension: SessionTicket TLS					

图10.29

- 图 10.29 所示区域①显示的是 TLS 数据包记录层头部的内容类型字段值。该字段值为 22，即表明此 TLS 数据包为握手协议消息（与之对应的显示过滤表达式为 `ssl.record.content_type == 22`）。
- 图 10.29 所示区域②显示的是 TLS 数据包中记录的类型字段值。该字段值为 1，即表明此 TLS 数据包包含由客户端发往服务器的 **Client Hello** 握手协议消息。
- 图 10.29 所示区域③显示的是客户端所支持的 TLS 协议的版本。
- 图 10.29 所示区域④显示的是将会在密钥生成过程中所使用的客户端时间。
- 图 10.29 所示区域⑤显示的是由客户端生成，将会在密钥生成过程中使用的随机数。
- 图 10.29 所示区域⑥显示的是客户端所支持的各种加密算法，优选使用的加密算法排列在先。
- 图 10.29 所示区域⑦显示的是客户端所支持的各种数据压缩方法。

在步骤 2 中，服务器将回复 **Server Hello** 消息。图 10.30 示出了 **Server Hello** 消息（其数据包编号为 162）的内容。

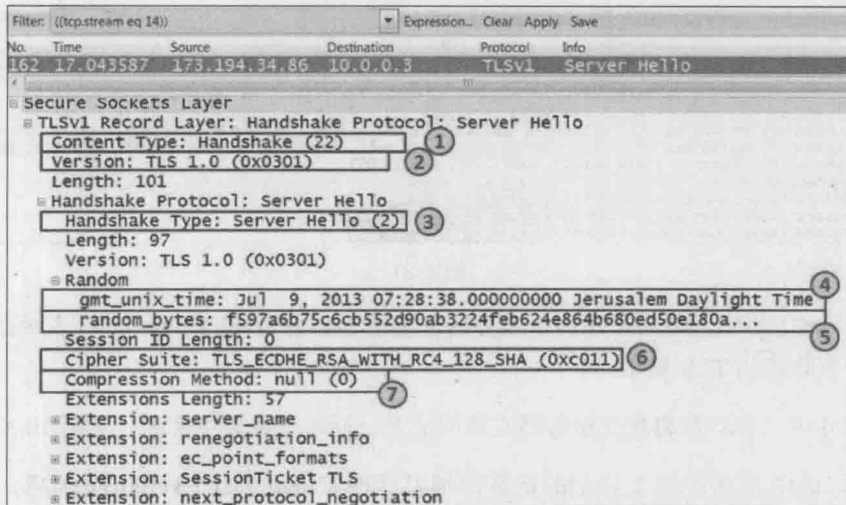


图10.30

- 图 10.30 所示区域①显示的是 TLS 数据包记录层头部的内容类型字段值。该字段值为 22，即表明此 TLS 数据包为握手协议消息（与之对应的显示过滤表达式为 `ssl.record.content_type == 22`）。
- 图 10.30 所示区域②显示的是本次会话所使用的 TLS 协议的版本。
- 图 10.30 所示区域③显示的是 TLS 数据包中记录数据的类型字段值。该字段值为 2，即表明此 TLS 数据包包含由服务器发往客户端的 **Server Hello** 握手协议消息。
- 图 10.30 所示区域④显示的是将会在密钥生成过程中所使用的服务器端时间。
- 图 10.30 所示区域⑤显示的是由服务器生成，将会在密钥生成过程中使用的随机数。
- 图 10.30 所示区域⑥显示的是本次 TLS 会话所采用的加密算法，从客户端发出的加密算法列表中选择。
- 图 10.30 所示区域⑦显示的是本次 TLS 会话所采用的数据压缩方法。

在步骤 3 中，服务器将会向客户端发送证书（见图 10.31）。

- 图 10.31 所示区域 1 显示的是服务器发送的 **Certificate** 命令，其中包括了服务器的证书。点击左边的“+”号，可以了解到证书的发布者、有效期以及其他信息。
- 图 10.31 所示区域 2 显示的是服务器发送的 **Server Key Exchange** 消息（通常都使用 Diffie-Hellman 算法来完成密钥的交换），其中会包括必要的参数（公钥、签名等信息）。

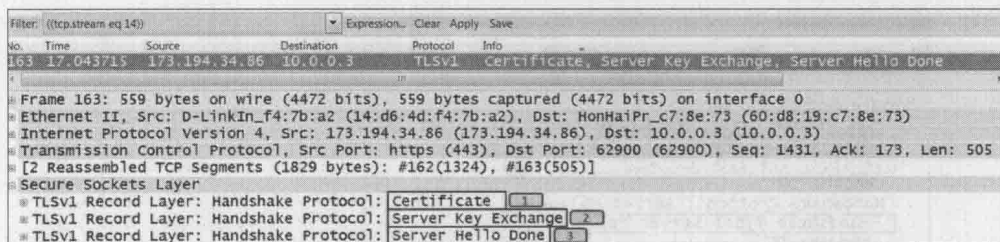


图10.31

- 图 10.31 所示区域 3 显示的是服务器发送的 **Server Hello Done** 消息, 表示服务器完成了本阶段的 TLS 握手。

在步骤 4 中, 客户端将接受服务器发送的证书, 同时生成预主密钥 (见图 10.32)。

- 图 10.32 所示区域 1 显示的是客户端发送的 **Client Key Exchange** 消息, 其中包含有由客户端创建的预主密钥, 该预主密钥在发送时会以服务器发送的公钥加密。服务器和客户端会根据 **Client Hello** 和 **Server Hello** 消息中的数据, 来生成对称加密密钥。
- 图 10.32 所示区域 2 显示的是客户端向服务器发出的 **Change Cipher Spec** 消息。这表示客户端要求服务器在后续通信中启用加密模式。

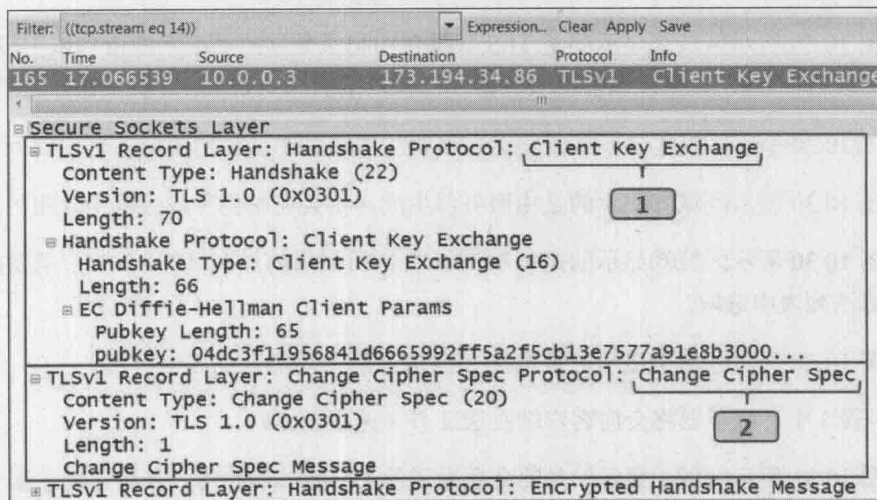


图10.32

在步骤 5 中(这一步为可选), 服务器向客户端发送 **New Session Ticket** 消息, 如图 10.33 所示。

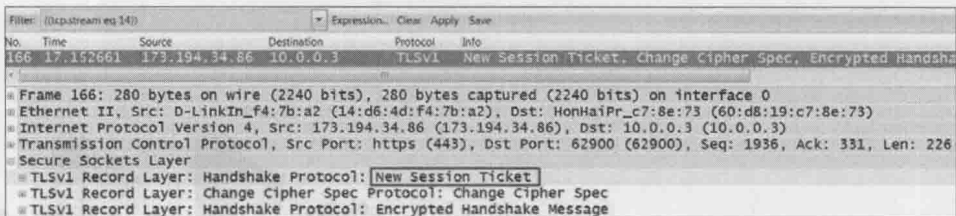


图10.33

10.10.4 拾遗补缺

有个问题作者曾被多次问及，该问题是：用 SSL/TLS 加密的信息有被破解的可能吗？只要握有目标服务器所提供的私钥，当然可以破解通过 SSL/TLS 加密的信息，不过，要想得到这一私钥却不太容易。

窃取这一私钥的方法有很多，在某些情况下，这些方法还非常奏效。显而易见，与此有关的内容不在本书探讨范围之列。如真能获取这一私钥，请点击 **Edit** 菜单中的 **Preferences** 菜单项，在弹出的 **Preferences** 窗口中，点击左边的 **Protocol** 配置选项前的“+”号，单击 **SSL** 协议，在右边的 **Pre-Shared-Key** 对话框中输入该密钥，就能解密 Wireshark 抓到的经过 SSL/TLS 加密的数据包了。与此有关的内容详见链接 <http://wiki.wireshark.org/SSL> 及其他 Web 站点和 blog。

第 11 章

企业网应用程序行为分析

本章涵盖以下主题：

- ▶ 摸清流淌于网络中的流量的类型；
- ▶ FTP 故障分析；
- ▶ Email 协议（POP、IMAP 及 SMTP）故障分析；
- ▶ MS-TS 和 Citrix 故障分析；
- ▶ NetBIOS 协议故障分析；
- ▶ 数据库流量及常见故障分析。

11.1 简介

Wireshark 存在的最大价值之一，就是能用它来分析流淌于网络中的各种应用程序的流量，从而为排除应用程序故障提供依据。当应用程序运行缓慢时，可能的原因包括：LAN 问题（一般都与无线 LAN 有关）、WAN 问题（WAN 链路带宽不足或延迟过高）、安装应用程序服务器端/客户机端软件的主机运行缓慢（TCP 窗口问题），以及应用程序自身问题等。

本章将深入探讨企业网内常用的应用程序的运作方式，同时会介绍如何定位及解决与这些应用程序有关的故障。首先，会介绍如何探查及归类流淌于网络中的流量的类型。其次，会分析各种企业网应用程序的运作方式，介绍不同的网络状况对那些应用程序的影响。

本章还会介绍如何使用 Wireshark 解决常见诸于企业网内的各种应用程序的故障。这些应

用程序包括：FTP 应用、各种 Email 应用、数据库、NetBIOS 应用、Microsoft 终端服务器和 Citrix 等。

11.2 摸清流淌于网络中的流量的类型

在一个全新的网络环境中处理故障时，第一件事就是要摸清流淌于网络中的流量的类型。运行于网络中的应用程序五花八门，支撑应用程序的协议类型多种多样，它们之间还会彼此影响，相互干扰。

在某些情况下，网络中还会有多个 VLAN，存在多个虚拟转发实例（VRF），部署多座刀箱（Bladeserver），每个刀箱里还会配备连接到虚拟背板交换机端口的刀片服务器。在同一网络基础设施里运行的每一样东西都有可能相互影响。



注意

很多人都分不清 VLAN 和 VRF 的区别。两者的用途虽大致相同，但使用场合却全然不同。VLAN 用在 LAN 中行使第一、二层的隔离任务；VRF 则是指共存于同一台路由器上的多个路由表实例，常用在 SP 网络中行使第三层的隔离任务。在 SP 网络中，VRF 一般都会跟多协议标签交换（MPLS）技术结合使用，用来提供同一客户不同站点间的 IP（第三层）连通性，同时还能实现不同客户间的网络隔离。

本节会探讨如何借助 Wireshark，来洞察流淌于网络中的流量的类型。



注意

术语“刀箱”（Bladeserver）是指刀片服务器的机箱，其正面安装的是刀片服务器，后面则配有 LAN 交换机。这一称谓随厂商而异，IBM 称其为刀片中心（Bladecenter），HP 则将其命名为刀片系统（Bladesystem）。

11.2.1 准备工作

在一个全新的网络环境中处理故障时，首先应把 Wireshark 主机接入网络，通过抓包来弄清网络中究竟流淌着哪些流量。在执行上述操作时，请按以下原则行事。

- 若故障涉及一台服务器，则应在 LAN 交换机上开启端口镜像功能，将连接该服务器的端口的所有流量重定向给 Wireshark 主机，以此来弄清该服务器都接收或发送了何种类型的流量。
- 若故障涉及一远程分支机构网络，则应设法将路由器上连接该分支机构网络的 WAN 端口的流量重定向给 Wireshark 主机，以此来弄清进、出该分支机构网络的流量的类型。

- 若故障涉及 Internet 链路（比如，用户普遍反应网页打开缓慢），则应设法将奔流于该 Internet 链路的所有流量重定向给 Wireshark 主机，以此来弄清该 Internet 链路究竟承载了何种类型的流量。

本节会介绍如何借助 Wireshark 软件，来摸清流淌于网络中的流量的类型，以及如何排除相应的网络故障。

11.2.2 操作方法

要借助 Wireshark 软件来分析流淌于网络中的流量的类型，请按以下步骤行事。

1. 按照之前提及的三原则之一，将相关流量重定向给 Wireshark 主机。
2. 可利用 Wireshark 软件自带的下列工具，来分析所抓流量。
 - 利用 **Statistics** 菜单下的 **Protocol Hierarchy** 工具，来观察并了解所抓流量的协议类型及占比情况。
 - 利用 **Statistics** 菜单下的 **Conversations** 工具，来了解所抓流量在各个层级上的源头和归宿。
3. 凭借 **Protocol Hierarchy** 工具所生成的信息，能获知流淌于网络中的流量的类型，如图 11.1 所示。

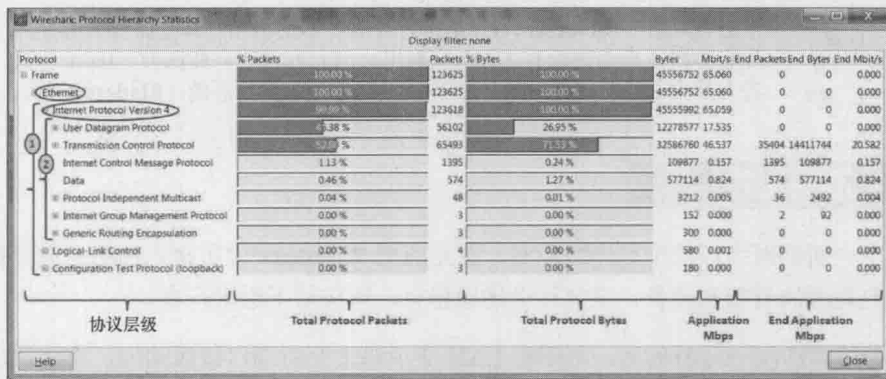


图 11.1

4. 通过图 11.1，可了解到穿梭于网络中的各种流量的分布情况。
 - 以太网帧：其麾下的子协议流量包括：IPv4 流量、逻辑链路控制协议（Logical-Link Control）流量，以及配置测试协议流量（Configuration Test Protocol [loopback]）。

- ▣ **IPv4 (Internet Protocol Version 4) 数据包**: 其麾下的子协议流量包括: UDP 流量、TCP 流量、PIM 流量、IGMP 以及 GRE 流量等。

- 若点击各 **Protocol** 左边的“+”号, 则会显示出其麾下所有类型的上层协议流量。
- 要想了解某种协议流量的吞吐量, 请点击其“父协议”左边的“+”号, 直至该协议暴露为止, 如图 11.2 所示。通过图 11.2, 可获知 HTTP 流量在本次抓包时间段内的平均传输速率。

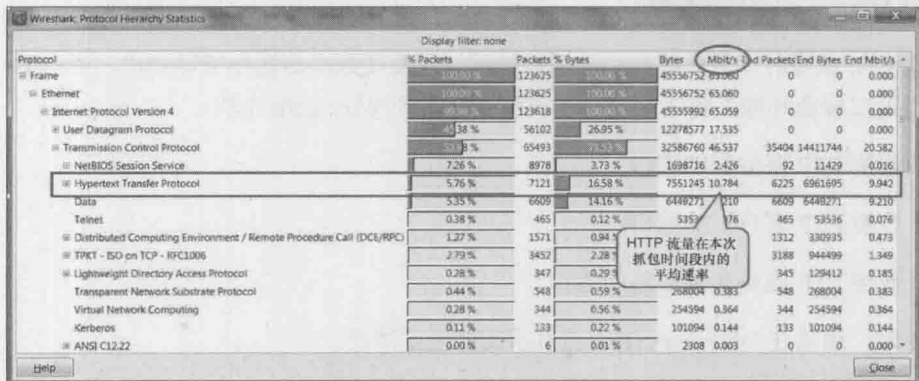


图11.2

- 若再点击 Hypertext Transfer Protocol 协议左边的“+”号, 则可了解到经由 HTTP 封装的各种数据 (XML、MIME、JavaScripts 等) 在本次抓包时间段内的平均传输速率。

11.2.3 拾遗补缺

在某些情况下 (比如, 准备网络统计报表, 向领导汇报时), 需要以图形方式来呈现与网络的各种性能指标有关的统计信息。可使用以下工具来完成这项工作。

- **Etherape (用于 Linux)**: <http://etherape.sourceforge.net/>。
- **Compass (用于 Windows)**: http://download.cnet.com/Compass-Free/3000-2085_4_-75447541.html?tag=mncol;1 (来自 Wildpackets)。

11.3 FTP故障分析

文件传输协议 (FTP) 的主要用途是跨 TCP/IP 网络传输文件。它是一种运行于 TCP 20 和 21 端口的协议, 这两个端口分别用来传输数据和发送控制命令。

FTP 有以下两种操作模式。

- **Active 模式 (ACTV):** 当 FTP 以此模式运行时, FTP 客户端会先发起一条通往 FTP 服务器端的控制连接, 后者随后会主动发起一条通往前者的数据连接。
- **Passive 模式 (PASV):** 当 FTP 以此模式运行时, FTP 客户端会先后发起通往 FTP 服务器端的控制和数据连接各一。

FTP 的这两种模式都很常用, 11.3.4 节将细述各自的原理。

11.3.1 准备工作

处理 FTP 故障时, 若怀疑存在连通性故障, 或发现数据传输缓慢等迹象, 请在 LAN 交换机上启用端口镜像功能, 将下列端口的流量重定向至 Wireshark 主机:

- 连接 FTP 服务器的端口;
- 连接 FTP 客户机的端口;
- 连接 FTP 流量穿越链路的端口。

必要时, 还需在 Wireshark 中应用显示及抓包过滤器。

11.3.2 分析方法

排查 FTP 性能问题时, 应遵循以下排障思路。

1. 首先, 应确保在第二、三、四层方面 (以太网、IP、TCP) 没有故障, 前面几章已经介绍过了解决第二、三、四层故障之法。在许多情况下, 数据传输缓慢都与第二、三、四层故障有关, 跟 FTP 应用本身无关。
2. 检查 Wireshark 抓包文件中是否有身背 TCP retransmissions 和 duplicate ACKs 字样的数据包。若有, 则需进一步查明这样的数据包是只涉及 FTP 流量, 还是涉及所有 TCP 协议流量^①。若涉及所有 TCP 协议流量, 则说明不单是 FTP 流量传输缓慢, 而是整个网络都有故障。若身背 TCP retransmissions 和 duplicate ACKs 字样的数据包只涉及 FTP 流量, 且那些数据包的源或目的 IP 地址相同, 则说明 FTP 流量传输缓慢很可能是拜 FTP 服务器或客户机反应慢所赐。
3. 通过 FTP 传输文件时, 若使用 Wireshark 抓包, 则相应的流量在 IO Graphs 窗口中将会呈直边梯形的趋势, 在 TCP Graph (time-sequence) 窗口中将会出现一条笔直的斜线。

^① 译者注: 原文是 “Check for TCP retransmissions and duplicate ACKs. Check if they are on the entire traffic or only on the FTP connection”。

4. 图 11.3 (TCP graph[time-sequence]窗口) 呈现了一次有问题的 FTP 传输过程。

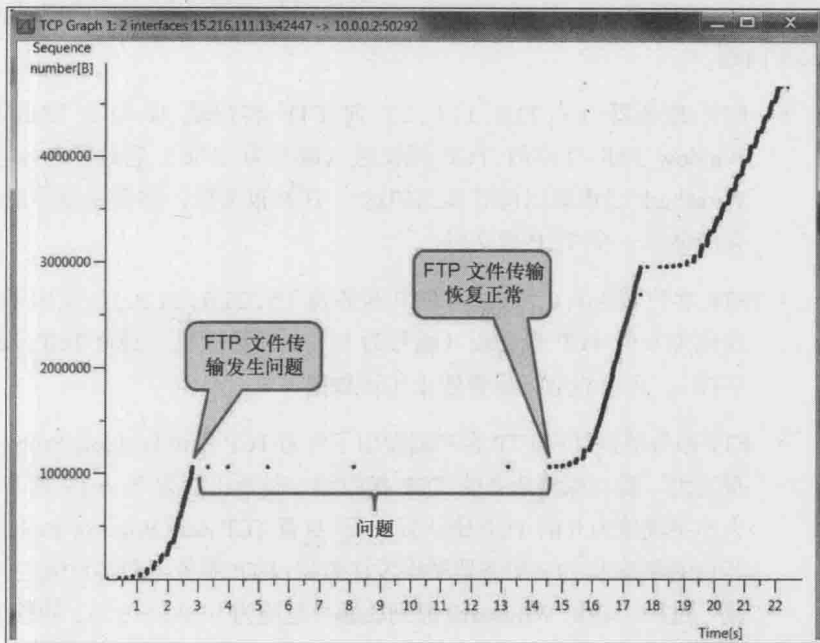


图11.3

5. 现在来看看这次有问题的 FTP 文件传输在应用了适当的显示过滤器的 IO Graphs 窗口中呈现出的图形 (见图 11.4)。

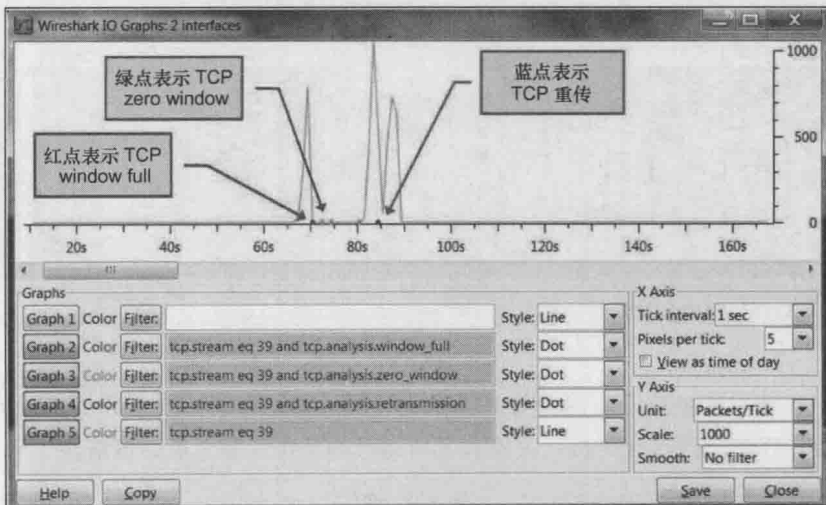


图11.4

6. 图 11.5 所示为由这次 FTP 文件传输所生成的数据包在 Wireshark 抓包主窗口的数据包列表区域中的真面目。由图 11.5 可知, 这些数据包被 Wireshark 识别为存在以下 TCP window 问题。

- FTP 服务器 15.216.111.13 向 FTP 客户端 10.0.0.2 发出了身背 TCP Window Full 字样的 TCP 报文段 (编号为 5763 的数据包), 也就是说, Wireshark 已识别出前者发出的这一 TCP 报文段, 将会是填满后者的接收缓存的最后一个 TCP 报文段。
- FTP 客户端 10.0.0.2 向 FTP 服务器 15.216.111.13 发出了窗口大小字段值为 0 的 TCP 报文段 (编号为 5778 的数据包, 身背 TCP Zero Window 字样), 其目的是让后者停止传送数据。
- FTP 服务器持续向 FTP 客户端发出了身背 TCP Zero Window Probe 字样的 TCP 报文段, 意在探测后者的 TCP 接收窗口是否仍然为 0。FTP 客户端发出窗口大小字段值为 0 的 TCP 确认报文段 (身背 TCP Zero Window Probe Ack 字样), 其目的是请求 FTP 服务器暂停发送数据。FTP 服务器和客户端之间的上述“交流”过程, 详见 Wireshark 抓到的编号范围为 5793~5931 的数据包。
- 片刻之后, FTP 客户端向 FTP 服务器发出了窗口大小字段值不为 0 的 TCP 确认报文段 (编号为 5939 的数据包, 身背 TCP Window Update 字样), 让后者继续传输数据。

No.	Time	Source	Destination	Protocol	Length	Info
5759	69.888792	10.0.0.2	15.216.111.13	TCP	54	50292 > 42447 [ACK] Seq=1 Ack=1055837
5763	70.127149	15.216.111.13	10.0.0.2	TCP	62	[TCP window Full] FTP Data: 668 bytes
5778	70.388829	10.0.0.2	15.216.111.13	TCP	54	[TCP zerowindow] 50292 > 42447 [ACK] Seq=1
5793	70.785945	15.216.111.13	10.0.0.2	FTP-DATA	55	[TCP ZerowindowProbe] FTP Data: 1 byte
5794	70.785986	10.0.0.2	15.216.111.13	TCP	54	[TCP ZerowindowProbeAck] [TCP ZerowindowProbeAck]
5801	72.105701	15.216.111.13	10.0.0.2	FTP-DATA	55	[TCP ZerowindowProbe] FTP Data: 1 byte
5802	72.105735	10.0.0.2	15.216.111.13	TCP	54	[TCP ZerowindowProbeAck] [TCP ZerowindowProbeAck]
5805	74.939558	15.216.111.13	10.0.0.2	FTP-DATA	55	[TCP ZerowindowProbe] FTP Data: 1 byte
5806	74.939596	10.0.0.2	15.216.111.13	TCP	54	[TCP ZerowindowProbeAck] [TCP ZerowindowProbeAck]
5930	80.025402	15.216.111.13	10.0.0.2	FTP-DATA	55	[TCP ZerowindowProbe] FTP Data: 1 byte
5931	80.025435	10.0.0.2	15.216.111.13	TCP	54	[TCP ZerowindowProbeAck] [TCP ZerowindowProbeAck]
5939	81.171193	10.0.0.2	15.216.111.13	TCP	54	[TCP Window Update] 50292 > 42447 [ACK] Seq=1
5940	81.3904	15.216.111.13	10.0.0.2	FTP-DATA	838	FTP Data: 784 bytes

图11.5

7. 图 11.5 所描绘的这一有问题的 FTP 数据传输过程, 是因 FTP 客户端主机“反应迟钝”所致。仔细检查那台主机, 关停其上所运行的某些进程之后, 便解决了问题。

导致 FTP 连通性故障的原因包括: FTP 服务器无法提供服务 (比如, 主机宕机或 FTP 服务未能正常启动等)、防火墙封锁了 FTP 连接、安装在 FTP 服务器或客户端主机上的某些软件

破坏了 FTP 协议的正常运作等。排查此类故障时，应遵循以下思路。

1. 核实 FTP 文件传输所依赖的 TCP 连接能否正常建立，为此，应检查 Wireshark 是否抓全了与此 FTP 连接配套的 TCP “三次握手” (SYN/SYN-ACK/ACK) 报文段。若与 FTP 文件传输配套的 TCP 连接未能正常建立，则不出以下原因。
 - 防火墙封锁了支撑 FTP 文件传输的 TCP 连接。此时，应赶紧去找管理防火墙的网络管理员。
 - FTP 服务器无法提供服务，需登录安装 FTP 服务器端软件的主机，检查主机自身及 FTP 服务器端软件的运行情况。
 - 安装在 FTP 服务器（主机）上的某个软件，干扰 FTP 协议的正常运作。杀毒软件、VPN 客户端软件、防火墙软件或其他安全防护软件都有可能阻挡来自 FTP 客户端的 FTP 连接。
 - 检查安装 FTP 客户端软件的主机，看看是不是 VPN 客户端软件或防火墙软件阻碍其发起 FTP 连接。
2. 当 FTP 运行于 active 模式时，客户端会先向服务器发起一条控制连接，后者再向前者发起一条数据连接。因此，需配置防火墙，令其在两个方向上放行必要的 FTP 流量，或将 FTP 的运行模式改为 passive 模式。

11.3.3 幕后原理

FTP 有两种运行模式：active 模式和 passive 模式。以 active 模式运行时，控制连接建立之后，FTP 服务器会主动打开一条通往 FTP 客户端的数据连接；运行于 passive 模式时，数据连接则会由后者向前者主动发起。现在来了解一下上述两种 FTP 模式的原理。

图 11.6 所示为 FTP passive 模式的原理。

1. FTP 客户机发起通往 FTP 服务器的 FTP 控制连接，这条 TCP 连接的源端口号为 FTP 客户端随机选定（本例为 TCP 1024 端口），目的端口号为众所周知的 TCP 21 端口。
2. FTP 服务器以 TCP 21 端口进行回应，与 FTP 客户机的 TCP 1024 端口建立 FTP 控制连接。
3. 控制连接成功建立之后，需要传递数据时，FTP 客户机将再次发起通往 FTP 服务器的 FTP 数据连接，这条 TCP 连接的源端口号为“P(FTP 控制端口号)+1”（本例为 1025），目的端口号为 FTP 服务器指定的高于 1024 的任一 TCP 端口号（本例为 TCP 2000 端口），这一端口号会通过先前建立的控制连接通告给 FTP 客户机。

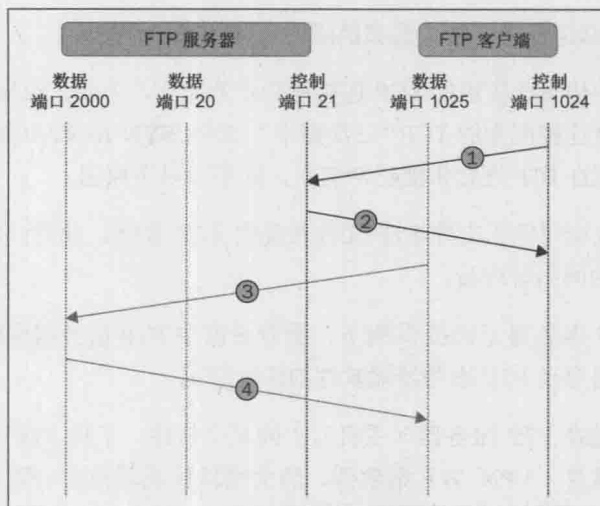


图11.6

4. FTP 服务器以 TCP 2000 端口进行回应，与 FTP 客户机的 TCP1025 端口建立 FTP 数据连接。

当 FTP 以 active 模式运行时，情况会略有不同。

1. FTP 客户机发起通往 FTP 服务器的 FTP 控制连接，这条 TCP 连接的源端口号为 FTP 客户端随机选定（本例为 TCP 1024 端口），目的端口号为众所周知的 TCP 21 端口（见图 11.7）。

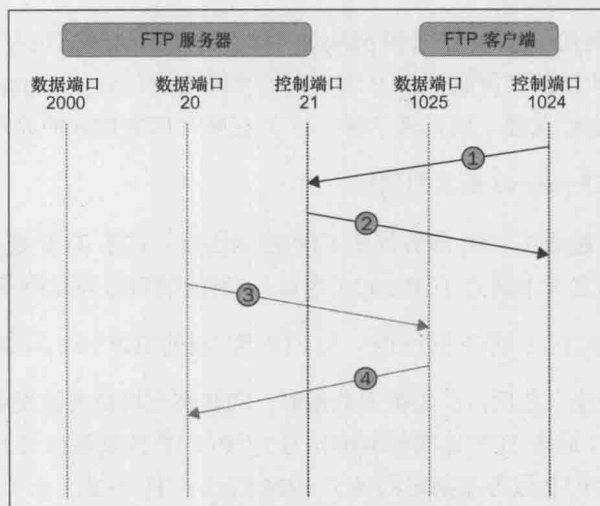


图11.7

2. FTP 服务器以 TCP 21 端口进行回应, 与 FTP 客户机的 TCP 1024 端口建立 FTP 控制连接。
3. 控制连接成功建立之后, 需要传递数据时, FTP 服务器将主动发起通往 FTP 客户机的 FTP 数据连接, 这条 TCP 连接的源端口号为众所周知的 TCP 20 端口, 目的端口号为“P (FTP 控制端口号) +1” (本例为 1025)。
4. FTP 服务器以 TCP 20 端口进行回应, 与 FTP 客户机的 TCP 1025 端口建立 FTP 数据连接。

11.3.4 拾遗补缺

FTP 是一种比较简单的应用层协议, 在大多数情况下, FTP 故障都不难排查。下面举几个作者参与排除过的 FTP 故障的实例:

- **故障实例 1:** FTP 服务器和客户端之间通过一条国际专线互连, 用户普遍感觉 FTP 文件传输缓慢, 并指责提供专线的运营商不给力。但据运营商反映, 那条国际专线的带宽利用率只有不到 20%。经作者核查, 确认了这一点。作者使用 Wireshark 进行抓包分析, 分析结果表明不存在任何与 TCP 有关的问题 (比如, TCP 重传或窗口问题)。无奈之下, 作者登录进安装 FTP 服务器端软件的主机, 卸载掉了原有的 FTP 服务器端软件, 安装了另一款软件 (免费的 FTP 服务器端软件遍地皆是), FTP 文件传输的性能就得到了极大地改善。看来, 这只不过是 FTP 服务器端软件效率低下所引发的 FTP 文件传输缓慢问题。
- **故障实例 2:** 有一用户反映, 他在连 FTP 服务器时, 每次都是尝试了 5、6 次之后, 遭到了 FTP 服务器的拒绝。动用 Wireshark 之后, 作者抓到了很多身背 FTP connection refused 字样的数据包 (用户投诉的原因也正在于此), 这看起来像是 FTP 服务器不响应。作者登录进安装 FTP 服务器端软件的主机, 重启了 FTP 服务之后, 问题便无影无踪。经过检查, 原来是主机上安装的杀毒软件干扰了 FTP 服务器端软件的运行。

作者通过以上两个实例是想传达这样一个理念, 那就是排除网络故障时, 即便有 Wireshark (或其他软件) 助一臂之力, 也得具备最基本的 IT 常识。

11.4 Email 协议 (POP、IMAP、SMTP) 流量及故障分析

常用的 Email 协议包括 POP3 (邮局协议版本 3)、简单邮件传送协议 (SMTP) 以及 Internet 消息访问协议版本 4 (IMAP4), 这几种协议不是运行于 Email 服务器与客户端之间, 就是运

行于 Email 服务器之间。

还有一种收取邮件的方法是，通过 Web 方式来访问邮箱，像 Gmail、Yahoo!以及 Hotmail 之类的邮件服务器都可以通过 Web 方式来访问。

本节将重点关注最常用的 Email 客户端/服务器协议和 Email 服务器/服务器协议：POP3 和 SMTP。此外，还将讨论在使用其他方法访问邮件服务器时，如何处理遇到的问题。

11.4.1 准备工作

当有用户反映存在邮件收发故障时，应首先核实故障是否因弱智级错误而起。所谓的弱智级错误包括：用户名/密码错误、认证协议未配置等。若未发现上述弱智级错误，请把 Wireshark 主机接入交换机；然后，在交换机上启用端口镜像功能，把连接了存在邮件收发问题的用户主机流量重定向至 Wireshark 主机。若遭遇大面积用户投诉，则需要将通往邮件服务器的通信链路的流量重定向至 Wireshark 主机。

11.4.2 操作方法

POP3 是一种运行于 Email 客户端与服务器之间的协议；SMTP 则是运行于 Email 服务器之间的协议。

POP3 协议运作机制

POP3 协议的主要作用是让 Email 客户端从 Email 服务器收取邮件。当 Email 客户端无法从 Email 服务器收取邮件时，请按以下步骤行事。

1. 检查 Email 客户端的用户名/密码是否配置正确。
2. 检查 Email 客户端能否顺利通过 Email 服务器的验证。由图 11.8 所示的 Wireshark 截屏可知，该 Email 账户的用户名和密码分别以 doronn@和 u6F 打头（用户名和密码的后半部分未予显示）。
3. 要想看到图 11.8 所示的内容，需先在 Wireshark 抓包主窗口的数据包列表区域内选择一个隶属于 POP3 数据流的数据包，点击右键，在弹出的右键菜单中点击 **Follow TCP Stream** 菜单项。
4. 在 POP 用户验证阶段，只要用户名/密码有误，POP3 连接便会中断。图 11.9 所示的 Wireshark 截屏展示了一次 POP3 认证失败的例子。由图可知，收到服务器的 **Logon failure** 提示之后，客户端主动断开了相应的 TCP 连接。



图 11.8

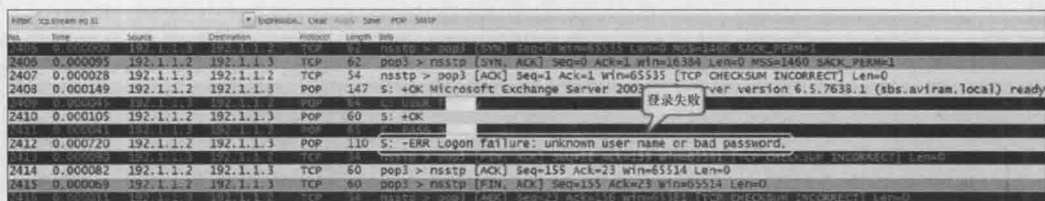


图 11.9

5. 客户端从邮件服务器下载邮件时，很容易导致通信链路过载，这一点请务必关注。可借助于之前介绍过的 IO Graphs 工具，并搭配显示过滤表达式“POP”，来检查通信链路的带宽利用率。
6. 应关注 Wireshark 是否抓到了身背 TCP retransmissions、TCP zero-window、TCP window-full 等字样的数据包。若身负上述字样的数据包全都隶属于 POP3 数据流，则表示存在通信线路过载、服务器反应慢等问题。这些问题有可能导致 Email 客户端无法从 Email 服务器收取邮件。

SMTP 协议运作机制

SMTP 协议的用途如下所列。

- SMTP 是一种运行于服务器之间的邮件传输协议。
- 对某些邮件客户端软件而言，可被配置为用 SMTP 来发送邮件（把待发邮件从客户端交付给服务器），用 POP3 或 IMAP4 从服务器收取邮件。

当感觉到邮件服务器之间沟通缓慢时，请按以下步骤行事。

1. 核实两台邮件服务器是否都位于同一站点。

- 若位于同一站点，则很可能是邮件服务器反应慢或应用程序问题。一般不太可能是网络传输问题，若两台邮件服务器都部署在通过高速 LAN 链路互连的数据中心内，则更是如此。
- 若位于不同的站点（比如，两台邮件服务器分别位于通过低速 WAN 链路互连的异地站点内），请检查 WAN 链路的负载状况。当发送包含大附件的邮件时，极有可能会造成低速 WAN 链路的拥塞。

2. 检查 Wireshark 抓包文件，看看是否抓到了身背 TCP retransmissions、TCP zero-window、TCP window-full 等字样，且隶属于 SMTP 数据流的数据包。图 11.10 所示为 Wireshark 抓到了众多身负 TCP retransmissions 字样，且隶属于 SMTP 数据流的数据包时的情形。

No.	Time	Source	Destination	Protocol	Length	Info
4125	0.001469	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4127	0.000001	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4129	0.000754	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4131	0.000001	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4143	0.001782	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4145	0.000001	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4147	0.000001	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4152	0.000738	192.5.11.98	172.16.30.243	SMTP	130	TCP Retransmission
4154	0.000001	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4156	0.000001	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4158	0.000001	172.16.30.113	192.5.11.73	SMTP	566	TCP Retransmission
4160	0.000001	172.16.30.243	192.5.11.98	SMTP	142	TCP Retransmission

图 11.10

3. 核实问题是否是出在 Email(SMTP)服务器反应慢上面。当读者看见图 11.11 所示 Wireshark 截屏时，应该知道作者使用了 **Statistics** 菜单下 **Conversation** 工具中的 **TCP** 标签功能。在勾选 **Limit to display filter** 复选框（让显示过滤参数 tcp.analysis.retransmission 生效），点击过 **Conversation** 窗口中的 **Packets** 一列（按数据包数量的多少重新排序）之后，可以看到有 793 个重传的 SMTP（TCP 25 端口）数据包，而 172.16.30.247 和 172.16.30.2 TCP 445 端口（Microsoft DS）之间、172.16.30.180 和 192.5.11.198 TCP 80 端口（HTTP）之间的 TCP 重传数据包则分别高达 9014 和 2139 个。由此可以判断出 SMTP 协议本身并无问题，邮件发送不畅只是受到了恶劣网络环境的影响。

4. 检查 Wireshark 抓包文件，看看是否抓到了身背 SMTP 状态代码的数据包。由图 11.12 可知，Wireshark 抓到了身背 SMTP 状态代码 451 的数据包。状态代码 451 表示邮件（SMTP）服务器发生了 local error in processing 错误。

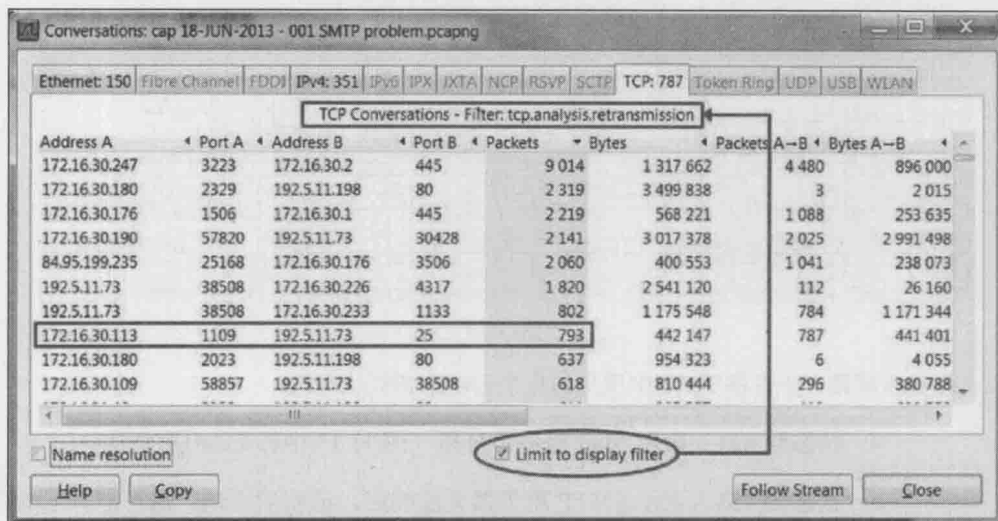


图 11.11

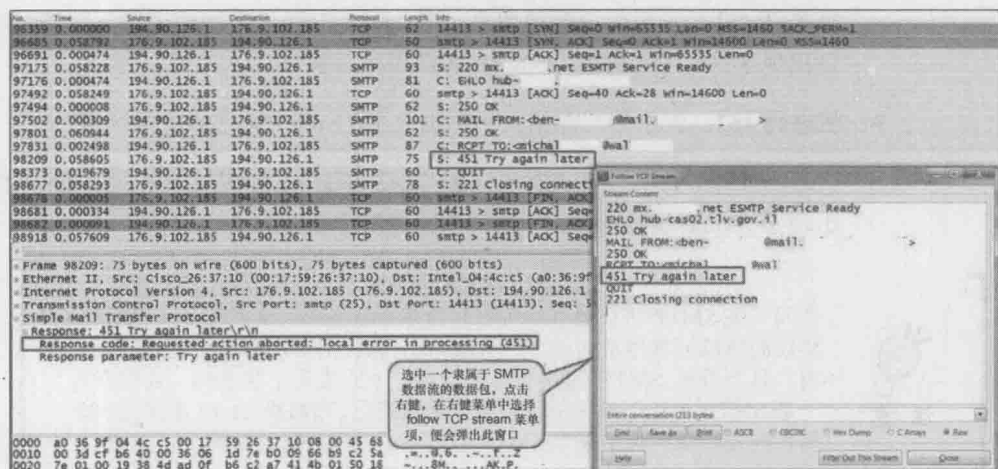


图 11.12



注意

当某种应用（层协议）发生故障时，其客户端或服务端主机一般都会生成与故障有关的日志信息。通过阅读这些信息或基于这些信息执行 Google 搜索，一般都能得出导致故障的原因。本文后文会举几个这方面的例子。

SMTP 状态码列表详见 RFC 1893 (<http://www.ietf.org/rfc/rfc1893.txt>)。

1. 要想弄清 SMTP 对话双方所发数据包是否包含 SMTP 错误状态码，请配置图 11.13 所示的 Wireshark 显示过滤器。

No.	Source	Destination	Protocol	Info
230881	194.90.126.3	212.179.113.105	SMTP	S: 500 Syntax error, I cannot recognize this command
233930	46.4.167.9	194.90.126.1	SMTP	S: 421 mx... .net Service unavailable (1)
236492	176.9.102.185	194.90.126.1	SMTP	S: 451 Try again later (2)
240907	194.90.126.3	212.29.221.217	SMTP	S: 500 Syntax error, I cannot recognize this command (3)
249262	80.179.55.150	194.90.126.1	SMTP	S: 250 2.5.0 Address and options OK, I 451 4.2.2 user over quot
264923	194.90.126.3	209.85.212.175	SMTP	S: 500 Syntax error, I cannot recognize this command
298264	207.232.39.169	194.90.126.1	SMTP	S: 452 <noreply@...> Mailbox size limit exceeded (4)
307959	194.90.126.3	192.114.23.26	SMTP	S: 500 Syntax error, I cannot recognize this command
325389	194.90.126.3	115.73.233.142	SMTP	S: 550 Invalid recipient/sender mailing address (5)
335006	192.115.106.20	194.90.126.1	SMTP	S: 250 OK / 450 <hub-...@tlv.gov.il>: Hello command rejected:
343706	194.90.126.3	62.219.19.49	SMTP	S: 500 Syntax error, I cannot recognize this command
368771	194.90.126.3	80.179.55.166	SMTP	S: 500 Syntax error, I cannot recognize this command
374990	194.90.126.3	212.29.221.217	SMTP	S: 500 Syntax error, I cannot recognize this command

图11.13

2. 现在来解读一下在图 11.13 中现身的几个 SMTP 错误状态码。

- 状态码 421: 表示 Mail 服务不可用 (图 11.13 中标记为①的区域)。
- 状态码 451: 表示 SMTP 服务器无法响应, 请稍后再试, 这可能预示着服务器负载过高或服务器故障 (图 11.13 中标记为②的区域)。
- 状态码 451: 表示用户配额超限 (图 11.13 中标记为③的区域, 在该区域内还出现了 SMTP 状态码 250, 具体原因稍后解释)。
- 状态码 452: 表示用户邮箱大小超限 (图 11.13 中标记为④的区域)。
- 状态码 450: 表示主机未发现 (图 11.13 中标记为⑤的区域, 在该区域内还出现了 SMTP 状态码 250, 具体原因稍后解释)。



注意

在同一条 SMTP 协议消息中可以包含多个状态码, 某些其他 (应用层) 协议的协议消息也是如此。在 Wireshark 抓包主窗口的数据包列表区域内, 只能看见 SMTP 协议消息中的第一个或头几个状态码。要想查看一条 SMTP 协议消息中所包含的全部状态码, 请按图 11.14 那样先在数据包列表区域内选中某个 SMTP 数据包, 再到数据包结构区域内查看。

Source	Destination	Protocol	Length	Info
194.90.126.3	192.114.23.26	SMTP	117	S: 500 Syntax error, I cannot recognize this command
194.90.126.3	115.73.233.142	SMTP	100	S: 550 Invalid recipient/sender mailing address
192.115.106.20	194.90.126.1	SMTP	179	S: 250 OK / 450 <hub-cas02.tlv.gov.il>: Hello command rejected: Host not found
194.90.126.3	62.219.19.49	SMTP	117	S: 500 Syntax error, I cannot recognize this command
194.90.126.3	80.179.55.166	SMTP	105	S: 500 Syntax error, I cannot recognize this command
194.90.126.3	212.29.221.217	SMTP	117	S: 500 Syntax error, I cannot recognize this command

Frame 335006: 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits)	
Ethernet II, Src: Cisco_26:37:10 (00:17:59:26:37:10), Dst: Intel_04:4c:c5 (a0:36:9f:04:4c:c5)	
Internet Protocol Version 4, Src: 192.115.106.20 (192.115.106.20), Dst: 194.90.126.1 (194.90.126.1)	
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 16419 (16419), Seq: 168, Ack: 123, Len: 75	
Simple Mail Transfer Protocol	
Response: 250 OK\r\n	SMTP 消息
Response code: Requested mail action okay, completed (250)	
Response parameter: OK	
Response: 450 <hub-cas02.tlv.gov.il>: Hello command rejected: Host not found\r\n	
Response code: Requested mail action not taken: mailbox unavailable (450)	
Response parameter: <hub-cas02.tlv.gov.il>: Hello command rejected: Host not found	

图11.14

若发现有很多条 SMTP 消息都包含不止一个状态码,则表示 Mail 服务器不可用。此时,需联系 Mail 服务器管理员。

其他收取邮件（访问邮箱）的方法和问题

下面简单介绍一下之前提到过的其他收取邮件（访问邮箱）的常用方法,这包括 Web Mail 和基于 HTTP 的 RPC。

- 使用 Web Mail 时,实际上是通过 HTTPS 连接到 Web 服务器,HTTPS 协议在第 10 章已做详述。以 Web Mail 的方式登录邮箱时,如遇到问题,一般都是 HTTPS 问题。
- 基于 HTTP 的 RPC 应用也与上面这种方法相仿。RPC 协议流量可轻松导致网络拥塞,对网络的带宽要求很高,因此基于 HTTP 的 RPC 应用对网络的延迟和抖动非常敏感。Microsoft Outlook 应用运行于 HTTPS 之上,并非基于标准的 RPC 协议。所以说此类应用一旦发生问题,多半为 HTTPS 问题。

11.4.3 幕后原理

邮件客户端软件与邮件服务器沟通时,大多都是借助于 POP3 协议,偶尔也会使用 SMTP 协议。当需要对邮件服务器执行某些操作时(比如,需要在客户端软件界面中查看远程邮件服务器上邮件的主题,但并不下载那些邮件时),将会用到 IMAP4 协议。邮件服务器之间的“沟通”大都使用 SMTP 协议。



注意

IMAP4 和 POP3 协议的区别是,前者允许用户直接通过邮件客户端软件对服务器上的邮件进行操作,而不用将邮件事先全部下载;当使用后者收取邮件时,所有邮件都会从邮件服务器上清除,并下载至本机客户端软件。也就是说,IMAP4 协议提供的是邮件客户端与服务器之间的双向通信,在邮件客户端软件内执行的操作,都会被反馈至邮件服务器;而使用邮件客户端软件操纵通过 POP3 协议收取到邮件时,操作结果不会被反馈至邮件服务器。

SMTP 状态码由三个部分构成,其构造方式使得 SMTP 消息中的错误一目了然。SMTP 状态码的构造方法和细节将在 11.4.4 节讨论。

POP3 协议

POP 是一种邮件客户端用来从服务器收取邮件的应用层协议。图 11.15 中的 Wireshark 截屏展示了一次典型的 POP3 会话建立过程。

1. 客户端主机与服务器之间以三次握手的方式建立 TCP 连接。

No.	Time	Source	Destination	Protocol	Info
460	0.000000	212.150.83.94	212.150.49.3	TCP	53797 > pop3 [SYN] Seq=0 win=5840 Len=0
545	0.256698	212.150.49.3	212.150.83.94	TCP	pop3 > 53797 [SYN, ACK] Seq=0 Ack=1 win=5840
546	0.000011	212.150.83.94	212.150.49.3	TCP	53797 > pop3 [ACK] Seq=1 Ack=1 win=5840
643	0.275490	212.150.49.3	212.150.83.94	POP3	S: +OK Messaging Multiplexor (Sun Java(tm) 1.3.1_01)
645	0.001145	212.150.83.94	212.150.49.3	TCP	53797 > pop3 [ACK] Seq=1 Ack=102 win=5840
652	0.018639	212.150.83.94	212.150.49.3	POP3	C: USER ddron@shtil.com
745	0.276816	212.150.49.3	212.150.83.94	TCP	pop3 > 53797 [ACK] Seq=102 Ack=26 win=666
746	0.000006	212.150.49.3	212.150.83.94	POP3	S: +OK password required for user doronn@
752	0.019747	212.150.83.94	212.150.49.3	POP3	C: PASS U6FU6F
844	0.272635	212.150.49.3	212.150.83.94	TCP	pop3 > 53797 [ACK] Seq=153 Ack=39 win=666
848	0.010938	212.150.49.3	212.150.83.94	POP3	S: +OK Maildrop ready
850	0.000745	212.150.83.94	212.150.49.3	POP3	C: NOOP
944	0.273535	212.150.49.3	212.150.83.94	POP3	S: +OK
946	0.000479	212.150.83.94	212.150.49.3	POP3	C: STAT
1042	0.288746	212.150.49.3	212.150.83.94	POP3	S: +OK 0 0
1048	0.017165	212.150.83.94	212.150.49.3	POP3	C: QUIT
1136	0.273700	212.150.49.3	212.150.83.94	POP3	S: +OK
1137	0.000006	212.150.49.3	212.150.83.94	TCP	pop3 > 53797 [FIN, ACK] Seq=192 Ack=58 Win=0 Len=0
1138	0.000239	212.150.83.94	212.150.49.3	TCP	53797 > pop3 [FIN, ACK] Seq=58 Ack=193 Win=0 Len=0
1227	0.274037	212.150.49.3	212.150.83.94	TCP	pop3 > 53797 [ACK] Seq=193 Ack=59 Win=666

图11.15

2. 邮件服务器向客户端主机发送身负 **OK Messaging Multiplexor** 字样的 POP3 数据包。
3. 操作客户端主机的用户输入邮箱的用户名/密码。
4. POP3 协议运作的帷幕正式拉开。邮件客户端主机发出身背 **NOOP** (no operation) 字样的 POP3 数据包，测试邮件服务器是否已经打开了 POP3 连接。邮件客户端主机发出身负 **STAT** (status) 字样的 POP3 数据包，要求邮件服务器回发有关邮箱的统计信息，比如，邮件数和总字节数。邮件服务器发出身背 **OK 00** 字样的 POP3 数据包（编号为 1042）作为应答，表示邮件数和总字节数均为 0。
5. 当发现服务器上的邮箱无邮件可供收取时，邮件客户端主机会发出身背 **QUIT** 字样的 POP3 数据包（编号为 1048），邮件服务器会发出身负 **OK** 字样的 POP3 数据包（编号为 1136）进行确认。随后，邮件客户端主机会拆除先前建立起的 TCP 连接（编号为 1137、1138 和 1227 的数据包）。

在启用了加密连接的情况下，邮件客户端主机和服务端之间的交流过程应如图 11.16 中的 Wireshark 截屏所示。以三次握手方式建立起 TCP 连接之后 (①)，邮件客户端主机和服务端之间会彼此交换几个 POP 数据包 (②)，然后双方会建立起 TLS 连接 (③)，以加密方式传输数据。

SMTP 协议和 SMTP 状态码

SMTP 状态码的构造方式如下所列。

```
class. subject. detail
```

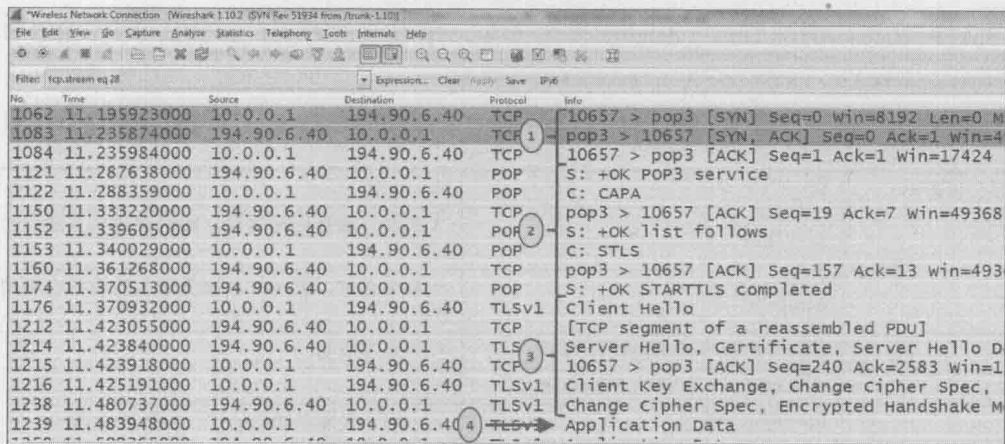


图11.16

举个例子，一条包含状态码 450 的 SMTP 消息具有如下含义。

- **class 4:** 表示存在临时性问题。
- **subject 5:** 指明了邮件递送协议的状态。
- **detail 0:** 表示其他或未定义的协议状态（RFC 3463 3.6 节）

下表所列 SMTP 状态码的各种类别码（class）。

SMTP 状态码	含 义	描 述
2.x.xxx	成功	操作成功
4.x.xxx	持续出现暂时性的故障	临时情况（暂时性的故障）导致邮件服务器无法发送邮件。暂时性的故障是因服务器负载过重或网络拥塞所致。通常，（在邮件发送失败之后，）重新发送将会取得成功
5.x.xxx	持久性的故障	持久性的故障导致邮件服务器无法发送邮件。所谓持久性的故障一般是指服务器故障或兼容性问题

下表所列 SMTP 状态码的各种主题码（subject）。

SMTP 状态码	含 义	原 因
x.0.xxx	其他或未定义状态	
x.1.xxx	地址状态	
x.2.xxx	邮箱状态	
x.3.xxx	邮件系统状态	
x.4.xxx	网络及路由状态	
x.5.xxx	邮件递送协议状态	
x.6.xxx	邮件内容或媒体状态	
x.7.xxx	安全或策略状态	

SMTP 状态码的细节码 (detail) 着实太多, 难以全部罗列, 详情请参考 RFC 3463 (<http://tools.ietf.org/html/rfc3463>)。

下表所列为常见的 SMTP 状态码。

SMTPu 状态码	含 义	原 因
220	服务就绪	(邮件) 服务正在运行, 随时准备执行邮件“投递”任务
221	服务关闭	一般为正常, 表示邮件服务器上的服务在不用时关闭
250	对邮件的操作请求已执行完毕	邮件发送成功
251	非本地用户, 邮件将会被转发	一切正常
252	用户通不过验证	用户通不过服务器的验证, 邮件无法递送
421	服务不可用	邮件发送服务不可用, 服务器无法处理收到的邮件, 这可能是因为服务器问题 (邮件递送服务未开启) 或服务器限制问题
422	邮件大小问题	收件人邮箱已满或邮件服务器对所收邮件的大小有限制
431	内存溢出或磁盘空间已满	邮件服务器内存溢出或磁盘空间已满
432	邮件接收队列被停止	服务器故障 (邮件递送服务未开启)
441	收件服务器无响应	发件服务器表示收件服务器不响应
442	连接故障	通往收件服务器的连接 (链路) 存在故障
444	无法路由	服务器无法确定递送邮件的下一跳
445	邮件服务器拥塞	邮件服务器临时性拥塞
447	邮件递送逾期	收件系统认为邮件太“旧”, 这通常都表示队列和传输问题
450	操作请求未被接受	邮件可能未被递送, 这通常都归咎于远程邮件服务器上的邮件递送服务有问题
451	无效命令	表示命令乱序或不被支持。收件服务器终止执行操作请求, 主要原因是其负载过重
452	操作请求未被接收	收件服务器存储空间不足
500	语法错误	由服务器发出的命令未被识别为有效的 SMTP 或 ESMTP 命令
512	DNS 错误	无法定位收件服务器
530	认证问题	收件服务器要求认证, 或已将发件服务器录入了黑名单
542	收件地址遭拒	表示收件人地址遭到收件服务器的拒绝。这通常是因为收件人地址通不过反垃圾邮件系统、IDS/IPS、智能防火墙或其他安全设备的检查

11.4.4 拾遗补缺

Email 应用有时也会成为网络中的杀手级应用, 对使用非对称线路接入 Internet 的小型

企业网络来说则更是如此。在发送文本格式的邮件时，并不会消耗多少网络带宽；可是在发送邮件时携带几兆甚至几十兆的附件，且 Internet 链路的上行带宽又过窄，网络中的其他用户就会在相当长的一段时间内感觉上网卡顿。这样的事儿在小型企业网络中可谓屡见不鲜。

另一个问题是，有些用户会把邮件客户端软件配置为在启动时自动从邮件服务器下载新邮件。在长假（或双休日）后的第一个工作日，要是有人一来上班就反映上网卡顿，则很有可能是公司所有员工正同时打开邮件客户端软件从邮件服务器收取邮件。

11.5 MS-TS 和 Citrix 故障分析

微软终端服务器（Microsoft Terminal Server, MS-TS）基于远程桌面协议（Remote Desktop Protocol, RDP）和 Citrix Metaframe 独立计算架构（Independent Computing Architecture, ICA），是一种在本地和远程 PC（或瘦客户端）之间实施桌面控制的应用程序。需要注意的是，这种桌面控制型应用程序会通过网络来传递远程主机的屏幕的变化情况。若屏幕变化不大，则 MS-TS 应用基本不会占用多少带宽，否则，便会吃掉大量带宽。

还有一事值得关注，那就是由安装桌面控制型应用程序的主机生成的流量是完全不对称的。对主控端而言，下行流量速率少则数十 K（kbit/s），多则数 M（Mbit/s）；而上行流量速率则最多只有几 K（kbit/s）。在设计运行此类应用程序的网络时，切莫遗忘这一点。

本节会举几个与桌面控制型应用有关的典型故障案例，同时会细述故障解决方法。为求表述简洁，作者会把上述此类应用统称为 Microsoft TS，简写为 MS-TS。

11.5.1 准备工作

当有用户反应 MS-TS 应用程序运行缓慢时，应首先问清楚慢在什么地方。然后，再到连接客户主机或受控端主机的 LAN 交换机上开启端口镜像功能，将相关端口的流量重定向至 Wireshark 主机。

11.5.2 操作方法

要想准定位 MS-TS 相关故障，首先应向一线用户询问具体遇到的是什么问题，并按以下步骤行事。

1. 当用户反应与 MS-TS 有关的应用程序运行速度缓慢时，应问他们：是屏幕中的数据呈现速度慢，还是窗口切换速度慢？
2. 若得到的回答是窗口切换速度不慢，则说明并非 MS-TS 问题。要是 MS-TS 出了问题，

就会出现窗口切换速度缓慢、鼠标键盘操作结果回显迟钝等现象。

3. 若得到的回答是基于 MS-TS 运行的应用程序生成报表的速度变慢, 则很有可能是该应用程序的后台数据库问题, 并非 MS-TS 或 Citrix 问题。
4. 若通信链路负载较高, 则使用 MS-TS 相关应用的最终用户势必会反映, 输入回显延迟较高。这是因为 MS-TS 应用的主要用途就是传输远程主机的屏幕变化情况, 只要用户打字速度过快, 在通信链路拥塞的情况下, 就会感觉输入回显迟钝。
5. 要想利用 Wireshark 获知通信链路的负载状况, 请按以下步骤行事。

- 启用 Wireshark 自带的 IO Graphs 工具。
- 在 IO Graphs 窗口中, 输入相关 Wireshark 显示过滤器, 监控 (MS-TS 主控端的) 上、下行链路的负载状况。
- 在 IO Graphs 窗口中的 Y 轴坐标区域内, 选择 Unit 下拉菜单中的 Bits/Tick 菜单项。

6. 按照上述设置, Wireshark IO Graphs 窗口看起来应该如图 11.17 所示。

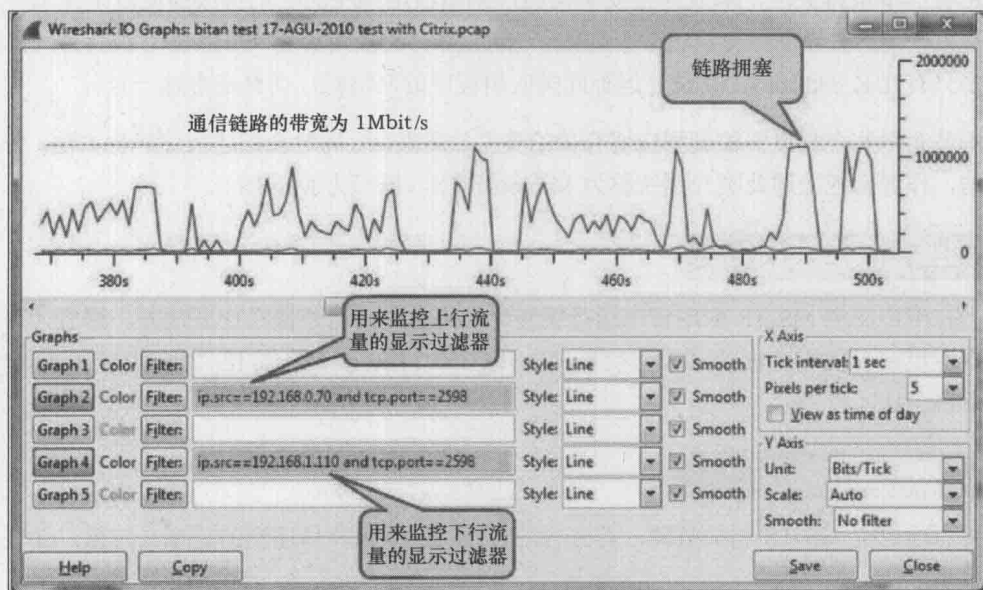


图11.17

7. 由图 11.17 所示流量模式可知, 运行 MS-TS 主控端程序的主机会生成极高的下行流量, 但却不怎么生成上行流量。请注意, 在图 11.17 所示的 IO Graphs 中, Y 轴坐标的单位为

Bits/Tick, 即比特/秒 (在 X 轴区域内的 Tick interval 下拉菜单中, 选择的是 “1 sec”)。在抓包开始后的 485~500 秒那段时间内, 通信链路上的流量吞吐量达到了顶峰。也就是在那段时间内, 用户感觉到了 MS-TS 应用运行缓慢 (比如, 鼠标或键盘操作结果回显缓慢等)。



注意

Citrix ICA 客户端连接应用服务器 (presentation server) 时, 所使用的目的端口号是 TCP 2598 或 1494 端口。

8. 请别忘了, 用户通过 MS-TS 主控端连接到 MS-TS 服务器之后, 操作的软件其实是安装在 MS-TS 服务器上的某种应用的客户端软件, 也就是说 MS-TS 服务器还得作为客户端去连接别的应用服务器。因此, 若 MS-TS 服务器不给力, 也会让用户产生 MS-TS 应用运行缓慢的错觉。
9. 若确诊为 MS-TS 故障, 则有必要弄清故障具体出在网络层面还是系统层面, 此时应:
 - 借助于 Wireshark 自带的 IO Graph 工具来了解通信链路的负载状况。如 MS-TS 应用运行缓慢确实是因为通信链路负载过高 (如上图所示), 则可以通过扩容链路带宽来解决;
 - 检查 MS-TS 服务器的性能状况。运行 MS-TS 应用的服务器的内存消耗量都颇为可观, 因此需检查其内存 (RAM) 使用情况。

11.5.3 幕后原理

MS-TS、Citrix Metaframe 等应用在运行时, 需要通过网络来传输 (远程服务器的) 屏幕变化情况。用户要先从客户端 (即安装了 MS-TS 客户端软件的主机或瘦客户端) 连接到终端服务器, 再使用安装在终端服务器上的各种应用程序客户端软件, 来连接各应用服务器。图 11.18 所示为终端服务器的运作原理。

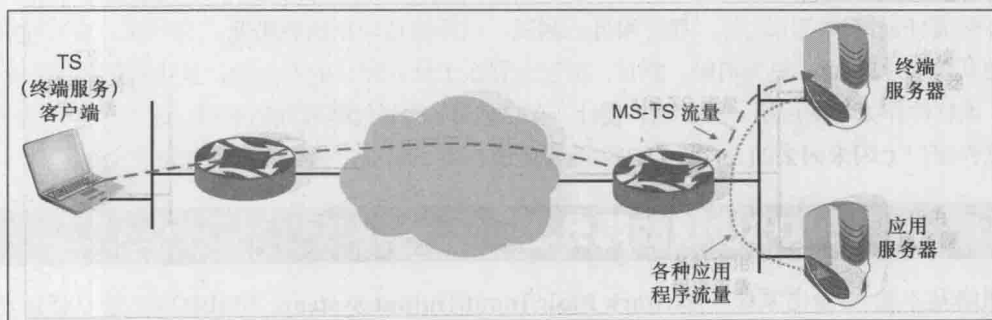


图 11.18

11.5.4 拾遗补缺

许多终端服务器软件的生产厂商都吹嘘自己的产品能在两个方面提升企业的生成率。厂商的销售人员首先会说他们的产品能方便 IT 人员对用户终端和各种软件客户端的管理,理由是各种应用程序的客户端软件都只需在终端服务器上安装,这样一来,就不用管理每个用户的 PC 以及其上所安装的各种软件了。其次,那些销售人员还会说,他们的产品一经部署,网络流量就会大大降低。

对于一种说法,作者倒也不准备抬杠,而且该话题与本书的内容无关。不过,作者绝不认可第二种说法。在有多个用户同时连接终端服务器的网络环境中,网络流量的高低要取决于用户操作什么样的应用程序。

1. 在操作基于文本或字符的应用程序(比如,ERP[企业资源规划]类应用程序)时,用户都是先输入数据,再点击某个按钮,最后查看结果。对于使用终端客户端的用户而言,要先连接到终端服务器,才能操作安装在终端服务器上的 ERP 客户端软件。此时,终端服务器会连接 ERP 程序的后台数据库。上述第二种说法能否成立,将取决于应用程序开发人员对后台数据库的设计,预期的流量(终端服务器与数据库服务器之间的流量)约为几十至几百 K (kbit/s) 之间。
2. 在终端客户端上操作像 Word、PowerPoint 这样的办公软件时,到底会生成多少流量则完全取决于用户的操作行为。若只是编辑普通的 Word 文档,几十到几百 K (kbit/s) 的流量也就够了。可要是编辑 PowerPoint 文档的话,则至少需要几百 K (kbit/s) 甚至几 M (Mbit/s) 的流量。要是以全屏方式预览 PowerPoint 文档(打开 PPT 文件后,再按一下 F5 键),吞吐量将高达 8~10M (Mbit/s)。
3. 通过终端客户端上网冲浪时,所耗带宽将会在几百 K (kbit/s) 到几 M (Mbit/s) 之间,具体的数字要取决于用户的上网行为。要是在线观看高清视频呢?还是别这么干为妙。

在部署任何终端应用之前,都必须再三测试,以弄清该应用所耗带宽。有一次,有家软件公司为其软件上的 logo 更为清晰、醒目,把它安置在了软件窗口的右上角,且让它每秒闪烁个 10 来次。该软件作为终端应用一经部署,便让一条 2Mbit/s 的通信链路拥堵不堪,这当然要归功于那个在软件窗口上闪来闪去的 logo。有些东西要是没有事先测试,你就永远不知道它会怎样。

11.6 NetBIOS协议故障分析

网络基本输入/输出系统(Network Basic Input/Output System, NetBIOS)是一套开发于 20 世纪 80 年代初期,用于 LAN 通信的协议。若干年前,这套协议被 Microsoft 公司相中,将

它用于 Windows 操作系统在 LAN 内的网络通信协议。随后, Microsoft 又把它与 TCP/IP 相融合在了一起(RFC 1101 和 RFC 1102, “NetBIOS over TCP/IP”)。

如今, NetBIOS 协议能提供以下三项服务。

- 名字服务(端口号 137): 用于名称注册以及名称/IP 地址间的解析。
- 数据报发布服务(端口号 138): 用于客户端和服务器的服务发布。
- 会话服务(端口号 139): 用于主机间的会话协商、文件访问和打开目录等。

本节将讨论 NetBIOS 协议族的常见问题以及解决问题的方法。NetBIOS 协议族极其复杂, 故障点极多, 本节只介绍该协议容易在什么地方出故障, 以及解决常见故障的方法。

11.6.1 准备工作

NetBIOS 协议主要用于 Windows 主机间的通信, 也可以用于 MAC/Linux 主机与 Windows 主机间的通信。如果在部署了上述机型的网络环境中出现断网、网速慢等问题, NetBIOS 协议故障也有可能是原因之一。Wireshark 是定位上述问题的绝佳工具, 它能显示出奔流于网络中的流量, 而 Windows 自带的某些工具则能显示出客户机与服务器间的操作情况^①。

11.6.2 操作方法

要解决 NetBIOS 协议故障, 请在 LAN 交换机上开启端口镜像功能, 将连接了相关客户机或服务器的端口的流量重定向至 Wireshark 主机。接下来, 作者会举几个 NetBIOS 协议故障场景的例子。

Wireshark 内置有若干显示过滤表达式, 可专门用来筛选 NetBIOS 协议流量, 在 Filter Expression 窗口中找到这些显示过滤表达式(在 Wireshark 抓包主窗口中点击 Filter 输入栏右边的 Expression 按钮, 就会弹出 Filter Expression 窗口)。

1. 以 netbios 打头的显示过滤表达式, 作用于包含 NetBIOS 命令的流量。
2. 以 nbns 打头的显示过滤表达式, 作用于 NetBIOS 名字服务流量。
3. 以 nbds 打头的显示过滤表达式, 作用于 NetBIOS 数据报服务流量。
4. 以 nbss 打头的显示过滤表达式, 作用于 NetBIOS 会话服务流量。
5. 以 smb 打头的显示过滤表达式, 作用于 SMB 流量。

^① 译者注: 原文是 “It will show you what runs over the network, and Windows tools will show you what runs in the clients and servers”。

常规测试

首先，要了解一下网络流量的整体状况；然后，在其中寻找异常流量。

1. 将 Wireshark 主机与 LAN 交换机相连，连哪个端口都可以，只要与连接了故障主机的端口隶属同一 VLAN（同一广播域）。
2. 在 Wireshark 抓包主窗口的 Filter 输入栏内输入显示过滤表达式 `nbns.flags.response == 0`，点击 **Apply** 按钮。所有 NetBIOS 名字服务请求数据包会立刻浮出水面，此类数据包都以广播方式发送，如图 11.19 所示。
3. 在图 11.19 所示的 Wireshark 截屏中，能看见以下几种 NetBIOS 名字服务请求数据包。

No.	Time	Source	Destination	Protocol	Length	Info
4994	0.000002	10.0.0.103	10.0.0.255	① NBNS	110	Registration NB WORKGROUP<1e>
4997	0.749962	10.0.0.103	10.0.0.255	NBNS	110	Registration NB ETTI<20>
4998	0.000002	10.0.0.103	10.0.0.255	NBNS	110	Registration NB WORKGROUP<1e>
5057	10.255261	10.0.0.102	10.0.0.255	② NBNS	92	Name query NB WPAD<00>
5075	0.763927	10.0.0.102	10.0.0.255	NBNS	92	Name query NB WPAD<00>
5088	0.512027	10.0.0.138	10.0.0.105	NBNS	92	Name query NBSTAT *<00><00><00><00><00>
5091	0.252327	10.0.0.102	10.0.0.255	NBNS	92	Name query NB WPAD<00>
5141	16.912745	10.0.0.105	10.0.0.255	③ NBNS	92	Name query NB YORASM-NDI<1c>
5144	0.749377	10.0.0.105	10.0.0.255	NBNS	92	Name query NB YORASM-NDI<1c>
5147	0.750008	10.0.0.105	10.0.0.255	NBNS	92	Name query NB YORASM-NDI<1c>
5265	4.215402	169.254.26.83	169.254.255.255	④ NBNS	92	Name query NB UM23.ESET.COM<00>
5287	0.744925	169.254.26.83	169.254.255.255	NBNS	92	Name query NB UM23.ESET.COM<00>
5297	0.749979	169.254.26.83	169.254.255.255	NBNS	92	Name query NB UM23.ESET.COM<00>
5298	0.751111	169.254.26.83	169.254.255.255	NBNS	92	Name query NB UM23.ESET.COM<00>

图 11.19

- 主机 10.0.0.3 发出的包含用名称 WORKGROUP 和 ETTI 来注册的名称 registration request 数据包(①)。NetBIOS 名字服务(NBNS)服务器会发出 positive name registration response 或 negative name registration response 数据包对发出 name registration request 数据包的主机进行回应(拒绝或接受注册)。主机发出 name registration request 数据包若收不到任何回应，将会假定注册成功。
- 用来查询(具有)指定名称(的主机)的名称 query request 数据包(②、③、④)。若网络中部署了 NBNS 服务器(域控制器)，Wireshark 还将抓到以下两种 NBNS 响应数据包中的一种。
 - 代码字段值为 3，表示所查名称不存在的 negative name registration response 数据包。
 - 包含所查名称的 positive name registration response 数据包，其代码字段值为 0。
- 4. 不应抓到由源 IP 地址为 169.254 打头的主机发出的任何 NBNS 请求数据包(⑤)。

以 169.245 打头的 IP 地址是自动私有 IP 地址 (APIPA)。当 (Windows) 主机被配置为以 DHCP 方式获取 IP 地址, 但获取不到任何 IP 地址时, 便会自行配置一个 APIPA 地址。

5. 在 Wireshark 抓包主窗口的 Filter 输入栏内输入显示过滤表达式 `tcp.port == 138 or udp.port == 138`, 点击 **Apply** 按钮, 所有 NetBIOS 数据报服务数据包会立刻浮出水面^①。NetBIOS 数据报服务数据包的目的 IP 地址一定是广播地址, 源、目端口则是 TCP 或 UDP 138 端口。图 11.20 所示为 Wireshark 抓到的 NetBIOS 数据报服务数据包, 主机发出此类数据包的目的是, 通告本机所行使的功能 (是工作站、数据库服务器还是打印机服务器等)。通过图 11.20, 可以看见:

- IP 主机 172.16.100.10 的主机名是 FILE-SRV, 其所行使的功能包括: 工作站、服务器和 SQL server (workstation, server, SQL server) (①)。
- IP 主机 172.16.100.204 的主机名是 GOLF, 其所行使的功能包括: 工作站、服务器和打印排队服务器 (workstation, server, print queue server) (②)。

No.	Source	Destination	Protocol	Length	Info
10119	172.16.100.176	172.16.100.255	BROWSER	243	Host Announcement ZIVAK, Workstation, Server, windows for w
10179	172.16.100.119	172.16.100.255	BROWSER	243	Host Announcement MERAVTI, Workstation, Server, NT workstat
10332	172.16.100.16	172.16.100.255	BROWSER	244	Host Announcement GNETAPP, Workstation, Server, NT workstat
10424	172.16.100.96	172.16.100.255	BROWSER	243	Host Announcement HAGITA, Workstation, Server, NT workstat
10471	172.16.100.10	172.16.100.255	BROWSER	243	Host Announcement FILE-SRV, Workstation, Server, SQL Server
10542	172.16.100.94	172.16.100.255	BROWSER	243	Host Announcement ORNAPI, Workstation, Server, NT Workstat
10721	172.16.100.204	172.16.100.255	BROWSER	264	Host Announcement GOLF, Workstation, Server, Print Queue se
10721	172.16.100.204	172.16.100.255	BROWSER	264	Host Announcement GOLF, Workstation, Server, Print Queue se
10766	172.16.100.124	172.16.100.255	BROWSER	243	Host Announcement ADIP, Workstation, Server, NT Workstation
10768	172.16.100.170	172.16.100.255	BROWSER	243	Host Announcement MICHALA, Workstation, Server, NT workstat
10929	172.16.100.106	172.16.100.255	BROWSER	258	Domain/workgroup Announcement NDI, NT Workstation, Domain E
10930	172.16.100.204	172.16.100.255	BROWSER	264	Host Announcement GOLF, Workstation, Server, Print Queue se

Frame 10119: 243 bytes on wire (1944 bits), 243 bytes captured (1944 bits)	
Ethernet II, Src: 3com_82:9a:c7 (00:50:da:82:9a:c7), Dst: Broadcast (ff:ff:ff:ff:ff:ff)	
Internet Protocol Version 4, Src: 172.16.100.176 (172.16.100.176), Dst: 172.16.100.255 (172.16.100.255)	
User Datagram Protocol, Src Port: netbios-dgm (138), Dst Port: netbios-dgm (138)	
NetBIOS Datagram Service	
SMB (Server Message Block Protocol)	协议结构
SMB Mailslot Protocol	
Microsoft Windows Browser Protocol	

图 11.20

6. 某些蠕虫或病毒也会设法让主机发出 NetBIOS 名字服务数据包, 来达成扫描网络的目的。要想得知网络中是否有主机感染了蠕虫或病毒, 需要仔细分析抓包文件, 看看 NetBIOS 名字服务流量是否高的离谱。
7. 仔细分析抓包文件, 关注广播数据包的数量。正常情况下, 一台主机每分钟发出广播数据包不应超过 10 个, 若超过了这一数字, 就要找到发包主机, 对其仔细检查。

① 译者注: 原文只有短短一句 “There are many announcement packets as well”。译文酌改。



分析 Wireshark 抓包文件时，对异常流量模式的判断并无定式可循，需借助于 Wireshark 专家系统（Expert system）工具、百度（Google）和常识来加以判断。

具体问题

在日常运维过程中，可能会碰到以下问题。

1. 需要使用 SMB（Server Message Block，服务器消息块）协议的网络中，Wireshark 可能会抓到包含 SMB 错误代码（完整的 SMB 错误代码列表详见 Microsoft MSDN: <http://msdn.microsoft.com/en-us/library/ee441884.aspx>）的 NetBIOS 会话服务数据包。SMB 协议是一种能让主机跨网络浏览（其他主机的）目录，（从其他主机）复制文件或执行其他操作的协议^①。

2. 若 SMB 错误代码为 0，则表示 SMB 协议或 SMB 应用程序的状态正常。在检查 Wireshark 抓到的 NetBIOS 会话服务数据包时，如发现其（SMB 头部中的）错误代码不为 0，就表示 SMB 协议或 SMB 应用程序存在问题，应仔细检查相关应用^②。

3. 在图 11.21 所示的 Wireshark 截屏中，可以看到一个 SMB 错误代码为 0xC0000022（STATUS_ACCESS_DENIED）的 NetBIOS 会话服务数据包^③。这一 SMB 错误代码是网管人员理应关注的众多 SMB 错误代码之一。由图 11.21 所示的 Wireshark 截屏可知，一台主机（IP 地址以 203 打头，IP 地址的后 3 个字节因故省略）在访问一台服务器（IP 地址为 10.1.70.95）上的目录\\NAS01\HOMEDIR 时，遭到了拒绝。

4. 使用基于 NetBIOS 协议的应用程序以远程访问的方式，浏览一台 Windows 主机的主目录时，弹出了标有“拒绝访问”（ACCESS DENIED）或类似字样的对话框。当然，该应用程序也可能会有自己的报错方式。执行上述操作的同时，若使用 Wireshark 抓包，在抓到的相关 NetBIOS 会话服务数据包中必能看见精确的 SMB 错误状态码，只要将 NT Status: 后的内容复制进 Google 里进行搜索，必能搜到访问遭拒的具体原因。

① 译者注：原文是“Using Server Message Block (SMB), which is the protocol that is used for browsing directories, copying files, and other operations over the network, you might see some error codes. The full list of error codes is listed in Microsoft MSDN:<http://msdn.microsoft.com/en-us/library/ee441884.aspx>”。译文酌改，如有不妥，请指正。依译者拙见，在未事先介绍 NetBIOS 协议数据包的种类和结构的情况下，本节的内容对许多初学者来说是都在“对牛弹琴”。

② 译者注：原文是“Code 0 means STATUS_OK, when everything works fine and there is no problem. Any other code should be examined”。译文酌改，如有不妥，请指正。

③ 译者注：原文是“In the following example, you can see a message STATUS_ACCESS_DENIED”。作者只会看表象，真不知道他是怎么排除网络故障的。

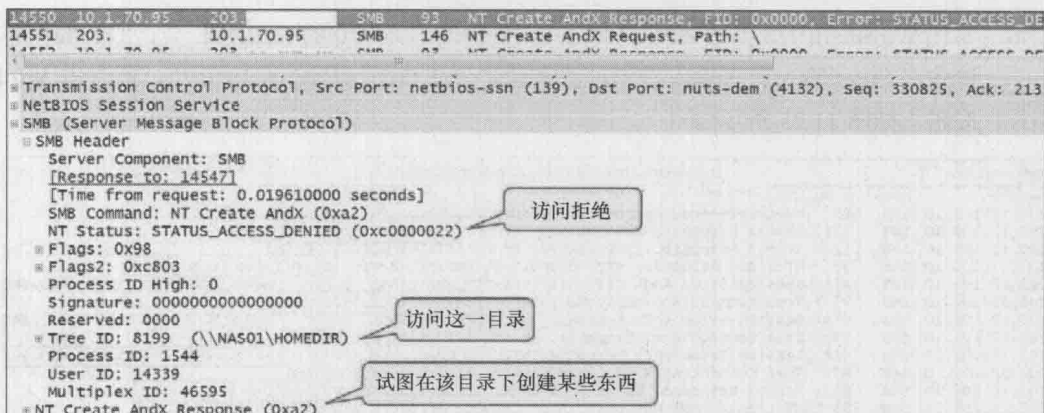


图 11.21

5. 在图 11.22 所示的 Wireshark 截屏中, 可以看到一个 SMB 错误代码为 0xc0000016 (STATUS_MORE_PROCESSING_REQUIRED) (②) 的 NetBIOS 会话服务数据包。由图 11.22 所示的 Wireshark 截屏可知, 那台 IP 地址以 203 打头的主机试图访问 (①) IP 地址为 10.1.70.95 的服务器上的目录 \\NAS01\ \SAMIM (③) 时, 发生了错误。

- 根据上述 SMB 错误代码, 查询之前提及的链接 (<http://msdn.microsoft.com/en-us/library/ee441884.aspx>), 可以获知出错的原因为 “There is more data available to read on the designated named pipe”(在指定的命名管道里无更多的数据可供读取)。
- 将 (②) 处 NT Status: 后的内容或出错的原因复制进百度 (Google), 搜索后得知是访问凭证问题, 需与系统管理员协商解决。

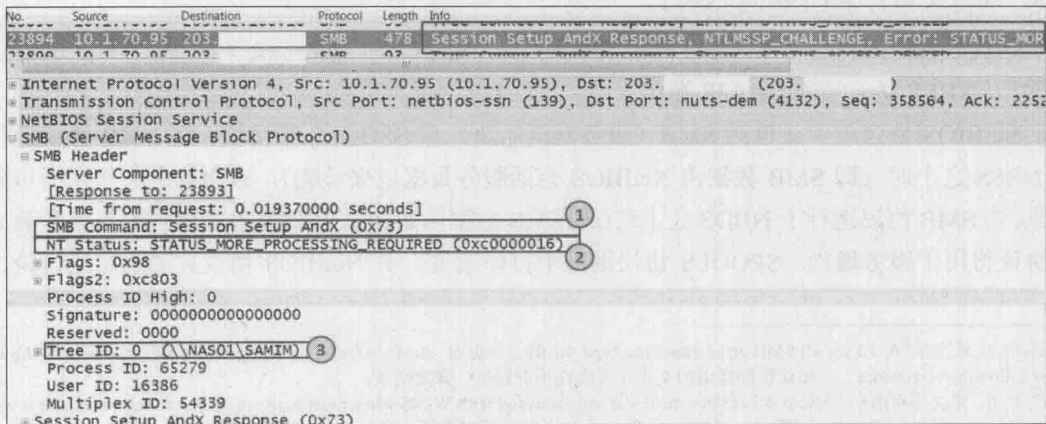
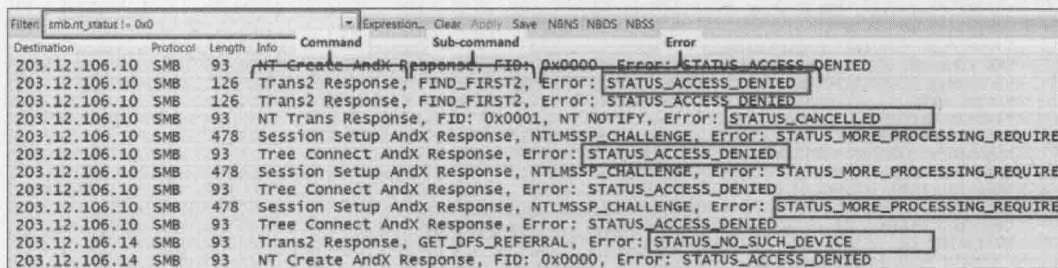


图 11.22

6. 要让 Wireshark 只显示出抓包文件中 SMB 错误码为非 0 的 NetBIOS 会话服务数据包, 请先在 Filter 输入栏内输入显示过滤表达式 `smb.nt_status != 0x0`, 再点击 **Apply** 按钮, 如图 11.23 所示^①。



Destination	Protocol	Length	Info	Command	Sub-command	Error
203.12.106.10	SMB	93	NT Create AndX Response, FID: 0x0000			STATUS_ACCESS_DENIED
203.12.106.10	SMB	126	Trans2 Response, FIND_FIRST2			STATUS_ACCESS_DENIED
203.12.106.10	SMB	126	Trans2 Response, FIND_FIRST2			STATUS_ACCESS_DENIED
203.12.106.10	SMB	93	NT Trans Response, FID: 0x0001, NT NOTIFY			STATUS_CANCELLED
203.12.106.10	SMB	478	Session Setup AndX Response, NTLMSSP_CHALLENGE			STATUS_MORE_PROCESSING_REQUIRED
203.12.106.10	SMB	93	Tree Connect AndX Response			STATUS_ACCESS_DENIED
203.12.106.10	SMB	478	Session Setup AndX Response, NTLMSSP_CHALLENGE			STATUS_MORE_PROCESSING_REQUIRED
203.12.106.10	SMB	93	Tree Connect AndX Response			STATUS_ACCESS_DENIED
203.12.106.10	SMB	478	Session Setup AndX Response, NTLMSSP_CHALLENGE			STATUS_MORE_PROCESSING_REQUIRED
203.12.106.10	SMB	93	Tree Connect AndX Response			STATUS_ACCESS_DENIED
203.12.106.14	SMB	93	Trans2 Response, GET_DFS_REFERRAL			STATUS_NO_SUCH_DEVICE
203.12.106.14	SMB	93	NT Create AndX Response, FID: 0x0000			STATUS_ACCESS_DENIED

图11.23

11.6.3 幕后原理

如 11.1 节所述, NetBIOS 协议能提供三种服务: NetBIOS 名字服务 (NBNS)、NetBIOS 数据报发布服务 (NBDS) 以及 NetBIOS 会话服务 (NBSS)。

NBNS 提供注册以及将 NetBIOS 名转换为 IP 地址的功能。所谓注册是指客户机以其 NetBIOS 名来注册域控制器。客户机在注册时, 会发出 NetBIOS 注册请求数据包 (即前面提到的 name registration request 数据包), 服务器会回复注册响应数据包 (即前面提到的 positive name registration response 或 negative name registration response 数据包) 来告知其是否注册成功^②。

客户端和服务端会利用 NBDS, 来行使服务宣告功能。网络中的主机可借助于 NBDS 来宣告: 本机 NetBIOS 名称、本机可向网络中的其他主机提供那些服务, 以及如何访问这些服务等。NBDS 运行于 UDP 138 端口。

NBSS 的作用包括: 在主机间建立会话, 跨网络打开、保存或执行远程 (主机上的) 文件^③。NBSS 运行于 UDP 138 端口。

NetBIOS 协议族中还包括 SMB (服务器消息块) 和 SPOOLS 等协议。当 SMB 协议运行于 NBSS 之上时 (即 SMB 数据由 NetBIOS 会话服务数据包来承载), 该协议所起作用为事务操纵; 当 SMB 协议运行于 NBDS 之上时 (即 SMB 数据由 NetBIOS 数据报服务数据包来承载), 该协议将用于服务通告。SPOOLS 协议则用于打印服务。对 NetBIOS 协议族的深层次讨论,

① 译者注: 原文是“6. To see all SMB error messages, type the filter `smb.nt_status != 0x0`. You will get all error responses, as shown in the following screenshot”。如此粗制滥造的文字译者实在不能忍受, 译文酌改。

② 译者注: 原文还有两句“Microsoft environment was implemented with WINS when most networks did not use it, and later it was replaced by DNS. It works over UDP port 137”。译者认为作者是在胡言乱语, 保留原文, 不予翻译。

③ 原文是“NBSS is used to establish sessions between hosts, open or save files, and execute remote files and other sessions over the network”。原文粗制滥造, 实在没法翻译。

跟本书的内容无关。要想排除 NetBIOS 协议故障，需要遵照本节内容里的指示——注意观察（Wireshark 抓到的）NetBIOS（会话服务）数据包所含的 SMB 错误代码。

11.6.4 拾遗补缺

为了让读者更好的理解 NetBIOS 协议，本节会举几个与该协议有关的抓包示例。

示例 1——应用程序卡顿

通过图 11.24 所示的 Wireshark 截屏，可发现该应用程序卡顿的原因。

仔细观察 Wireshark 截屏中的数据包，不难发现下述情况。

1. 一台主机（IP 地址以 203 打头）在请求访问一台服务器（IP 地址为 10.1.70.95）上的目录\\NAS01\\SAMIM 时（对应于身背 PATH\\NAS01\\SAMIM 字样的数据包），遭到了服务器的拒绝（对应于身背 STATUS_ACCESS_DENIED 字样的数据包）。

No.	Time	Source	Destination	Protocol	Info
26562	362.699257	203.	10.1.70.95	SMB	Tree Connect Andx Request, Path: \\NAS01\\SAMIM
26563	362.717483	10.1.70.95	203	SMB	Tree Connect Andx Response, Error: STATUS_ACCESS_DENIED
26564	362.717635	203.	10.1.70.95	SMB	Logoff Andx Request
26565	362.734572	10.1.70.95	203.	SMB	Logoff Andx Response
26572	362.853441	203.	10.1.70.95	TCP	nuts-dem > netbios-ssn [ACK] Seq=226260 Ack=359968 Win=
36000	482.813425	10.1.70.95	203.	TCP	netbios-ssn > nuts-dem [ACK] Seq=339967 Ack=226260 Win=
36001	482.813508	203.	10.1.70.95	TCP	TCP RST ACK 3600111 nuts-dem > netbios-ssn [ACK] Seq=
43569	602.799670	10.1.70.95	203	TCP	TCP Keep-Alive netbios-ssn > nuts-dem [ACK] Seq=35996
44872	802.800321	203	10.1.70.95	TCP	TCP Keep-Alive ACK nuts-dem > netbios-ssn [ACK] Seq=
55372	722.786747	10.1.70.95	203	TCP	TCP Keep-Alive netbios-ssn > nuts-dem [ACK] Seq=35996
55375	722.787380	203	10.1.70.95	TCP	TCP Keep-Alive ACK nuts-dem > netbios-ssn [ACK] Seq=
59751	798.181386	10.1.70.95	203.	NBSS	Session keep-alive
59758	798.390573	203.	10.1.70.95	TCP	nuts-dem > netbios-ssn [ACK] Seq=226260 Ack=359972 Win=
60622	816.812860	203.	10.1.70.95	SMB	Tree Disconnect Request
60623	816.829093	10.1.70.95	203.	SMB	Tree Disconnect Response
60627	816.984481	203.	10.1.70.95	TCP	nuts-dem > netbios-ssn [ACK] Seq=226299 Ack=360011 Win=
83503	936.946273	10.1.70.95	203	TCP	TCP Keep-Alive netbios-ssn > nuts-dem [ACK] Seq=35996
83508	936.949119	203	10.1.70.95	TCP	TCP Keep-Alive ACK nuts-dem > netbios-ssn [ACK] Seq=
75007	1030.936116	10.1.70.95	203	TCP	TCP Keep-Alive netbios-ssn > nuts-dem [ACK] Seq=35996
75008	1030.936568	203	10.1.70.95	TCP	TCP Keep-Alive ACK nuts-dem > netbios-ssn [ACK] Seq=
84086	1142.229579	10.1.70.95	203.	TCP	netbios-ssn > nuts-dem [RST, ACK] Seq=360011 Ack=226299

图11.24

2. 那台主机向服务器发出取消访问的请求（对应于身背 logoff andx request 字样的数据包），得到了服务器的确认（服务器发出了身背 logoff andx Response 字样的数据包）。
3. 由于（安装在那台主机上）实际发出目录访问请求的应用程序一直在等待，因此主机和服务器之间会互发身背 TCP keep-alive 字样的数据包，来保持 TCP 连接。
4. 片刻之后，那台主机向服务器发出断开连接请求（对应于身背 Tree disconnect request 字样的数据包），征得了服务器的同意（对应于服务器发出了身背 Tree disconnect Response 字样的数据包）。
- 5.（安装在那台主机上）实际发出目录访问请求的应用程序继续等待，主机和服务器之间继续互发身背 TCP keep-alive 字样的数据包，来保持 TCP 连接。

6. 服务器发出 RST 位置 1 的 TCP 数据包, 断开之前建立的 TCP 连接。

现在知道为什么最终用户反映该应用程序卡顿了吧?

示例 2——由 SMB 协议造成的广播风暴

有一次, 作者接到了一个客户的紧急求助电话。客户在电话里反映: 其公司总部 (HQ) 与某一远程站点之间通信中断。以下所列为该客户的 HQ 网络与那个远程站点网络的基本资料。

- 远程站点网络隶属于 IP 子网 172.30.121.0/24, 网关为 172.30.121.254。
- HQ 网络的 IP 地址段为 172.30.0.0/24。
- HQ 网络与远程站点网络之间通过运营商提供的 L3 MPLS IP-VPN 网络互连。

为了查明断网的原因, 作者进行了如下操作。

1. 作者在远程站点网络内的一台主机上 ping HQ 网络内的一台服务器, 但 ping 不通。
2. 作者致电提供 MPLS IP-VPN 网络的运营商; 运营商告知作者, 据他们的网管系统显示, 那条 MPLS VPN 链路负载为空。
3. 作者继续在那台主机上 ping 其默认网关 (远程站点网络内的路由器) 的 IP 地址 172.30.121.254, 同样 ping 不通。也就是说, 问题出在远程站点的本地 LAN 内, 主机与默认网关 (路由器) 之间无法通信。
4. 作者将安装了 Wireshark 的笔记本接入 LAN 交换机, 开启端口镜像功能, 抓取连接了路由器 (内部 LAN 接口) 的端口的流量。于是, 作者看到了图 11.25 所示的景象。

No.	① Time	② Source	③ Destination	④ Protocol	⑤ Info
22	0.000002	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
23	0.000001	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
24	0.000001	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
25	0.000001	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
26	0.000002	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
27	0.000910	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
28	0.000002	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
29	0.000001	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
30	0.000001	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot
31	0.000857	172.30.121.1	172.30.121.255	SMB Mailslot	Write Mail Slot

* Frame 1: 277 bytes on wire (2216 bits), 277 bytes captured (2216 bits) on interface 1 * Ethernet II, Src: Hewlett-2b:5d:e3 (f4:ce:46:2b:5d:e3), Dst: Broadcast (ff:ff:ff:ff:ff:ff) * Internet Protocol Version 4, Src: 172.30.121.1 (172.30.121.1), Dst: 172.30.121.255 (172.30.121.255) * User Datagram Protocol, Src Port: netbios-dgm (138), Dst Port: netbios-dgm (138) * NetBIOS Datagram Service * SMB (Server Message Block Protocol) * SMB Mailslot Protocol * Data (65 bytes)				SMB Mailslot 协议
---	--	--	--	-----------------------

图 11.25

5. 由图 11.25 可知, IP 主机 172.30.121.1 几乎是每隔 1 微秒 (①) 便发出一个协议属性为 SMB Mailslot (④), 身背 Write Mail Slot 字样 (⑤) 的广播数据包 (③)。

6. 为弄清身背 Write Mail Slot 字样的广播包的发送频率，作者动用了 Wireshark 自带的 IO Graph 工具，该工具给出的结果是 5000 个数据包/秒，如图 10.26 所示。而那台路由器的内部 LAN 接口只是 10Mbit/s 接口，根本就承受不了那么多蜂拥而来的广播包（即便是 100Mbit/s 或 1000Mbit/s 接口，也同样无法承受）。
7. 作者在 Google 和 Microsoft 站点上搜索了一翻，但一无所获。于是，便开始逐一停止运行于 IP 主机 172.30.121.1 上的可疑服务。在停止可疑服务的同时，保持对 Wireshark 抓包主窗口数据包列表区域的关注（见图 11.26）。当作者关停了一个名为 LS3Bcast.exe 的服务时，Wireshark 就抓不到身背 Write Mail Slot 字样的广播包了。看来这个名叫 LS3Bcast.exe 的服务才是导致断网的罪魁祸首。

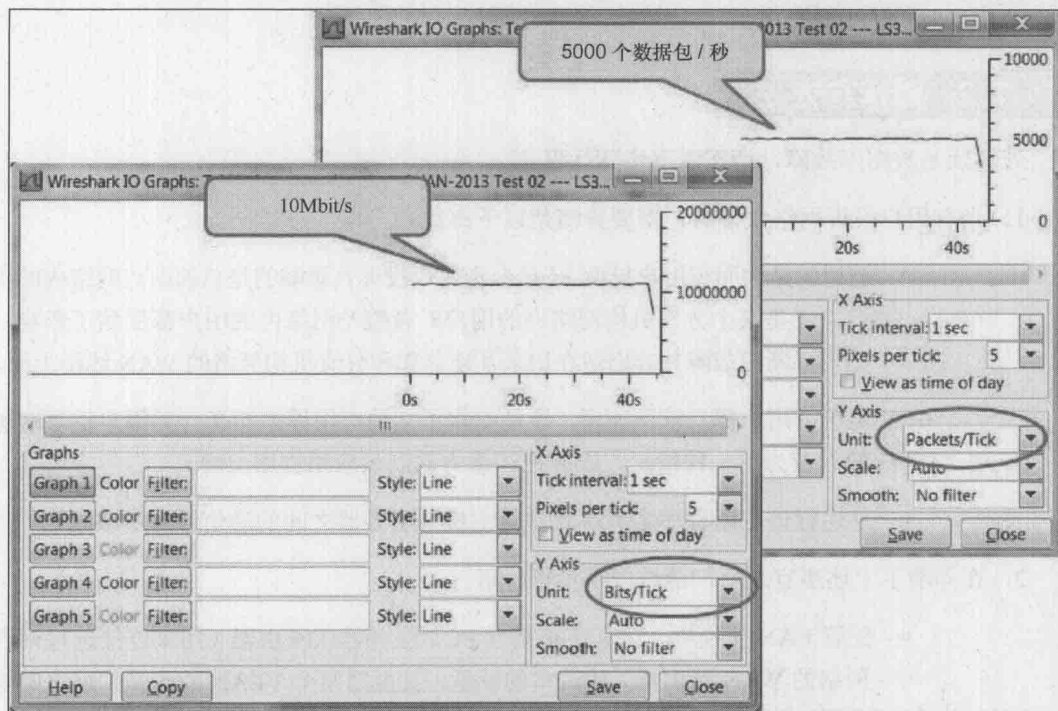


图 11.26

11.7 数据库流量及常见故障分析

有些读者一定会对作者在本章安排这么一节内容而感到不解。在 IT 领域里，网络技术与数据库技术毕竟是截然不同的两个分支。精通数据库和应用程序开发技术的人一般都不懂网络技术，反之亦然，当然少数绝世牛人除外。所以说，一般会让网络工程师去排除数据库故障，

也不可能让 DBA 去排除网络故障。不过，通过抓取并分析流淌在网络中的流量，网络工程师可以帮助 DBA 解决某些数据库故障。

在大多数情况下，当应用程序无法使用时，最终用户会首先认定是网络问题，从而对网络技术人员心存不满。因此，网络技术人员必须能准确判断故障是否真的跟网络有关。在某些特殊场合下，网络技术人员或许还要动用 Wireshark 抓取并分析网络中的流量，去配合 DBA 协查数据库故障。

11.7.1 准备工作

当有用户反映“网络太慢”的时候，网络技术人员首先需要判断网络是不是真的太慢。为此，请按下面所述步骤行事。

11.7.2 操作方法

若怀疑是数据库故障，请按以下步骤行事。

1. 当有用户投诉网络太慢时，需要弄清楚以下事宜。

- 遭遇的是大面积用户投诉，还是个别用户投诉？影响的是总部站点网络内的用户，还是某个分支机构网络内的用户？若整个网络内的用户都受到了影响，则可以断定故障并非出现在用来互连总部和分支机构网络的 WAN 线路上面。
- 是所有用户都反映网速慢，还是安装了某种应用程序的那一类用户反映网速慢？若为后一种情况，则应把注意力放到该应用程序上面。
- 是运行该应用程序客户端软件的主机与服务器之间的通信链路过载吗？

2. 在弄清了上述事宜之后，请按以下步骤行事。

- 配置 LAN 交换机，将其连接某台 PC、服务器或路由器（用来连接远程站点网络的 WAN 路由器）的端口的流量，甚至是某个 VLAN 的流量，重定向至 Wireshark 主机。打开 Wireshark 软件，开始抓包。
- 在抓包的同时，点击 **Analyze** 菜单下的 **Expert Info** 菜单项，Expert Infos 窗口会立刻弹出。要注意观察该窗口中各标签栏内与 TCP 有关的各种事件（尤其是 Errors 和 Warnings 事件），了解与这些事件相对应的数据包的源、目 IP 地址和源、目 TCP 端口号信息。这些信息将助于快速定位故障的原因。



注意

用 Wireshark 抓取并分析 Internet 链路的流量时，一定会观察到很多来自/发往 Internet Web 站点或 Email 服务器的 TCP 数据包都身背了 TCP retransmissions 或 TCP duplicate ACK 等字样。没有办法，访问 Internet 就是这样。然而，用 Wireshark 抓取并分析内网流量时，身背 TCP retransmissions 等字样的 TCP 数据包最多只应占总 TCP 流量的 0.1%~0.5%。

3. 若判断是网络中的主机、服务器或其他网络设备发生故障，请按本书之前各章所述来解决故障。不过，某些网络自身的问题也会对数据库应用的行为产生影响。在下面这个例子中，读者可以看到一客户机通过一条往返延迟为 35~40 毫秒的链路^①，与服务器通信时的“举动”^②。

- 由图 11.27 可知，Wireshark 截屏中所示数据包全都归属于编号为 8 的 TCP 数据流(①)，前三个 TCP 数据包是建立 TCP 连接的三次握手(TCP SYN/SYN-ACK/ACK) TCP 报文段。作者已将 Wireshark 显示出的第一个数据包的时间设置为参考时间(②)(选中第一个数据包，点击右键，选择右键菜单中的 **set time reference (toggle)** 菜单项)。构成这条编号为 8 的 TCP 数据流的 TCP 数据包的总数为 371(③)。

No.	Time	Source	Destination	Protocol	Info
1840	*REF*	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [SYN] Seq=0 W
1844	0.013483	192.168.10.80	192.168.20.88	TCP	wv-csp-udp-cir > vfo [SYN, ACK] Seq=
1845	0.013496	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [ACK] Seq=1 A
1846	0.015710	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [PSH, ACK] Seq=
1847	0.044857	192.168.10.80	192.168.20.88	TCP	wv-csp-udp-cir > vfo [PSH, ACK] Seq=
1848	0.045057	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [PSH, ACK] Seq=
1849	0.075752	192.168.10.80	192.168.20.88	TCP	wv-csp-udp-cir > vfo [PSH, ACK] Seq=

Frame 1930: 87 bytes on wire (696 bits), 87 bytes captured (696 bits)	
Ethernet II, Src: Hewlett-3e:54:e7 (00:0b:cd:3e:54:e7), Dst: Cisco_4f:4a:ec (00:60:47:4f:4e)	
Internet Protocol Version 4, Src: 192.168.20.88 (192.168.20.88), Dst: 192.168.10.80 (192.168.10.80)	
Transmission Control Protocol, Src Port: vfo (1056), Dst Port: wv-csp-udp-cir (3717), Seq: 1845	
Data (33 bytes)	

File: C:\Courses\Upstream Systems\PCAP Files\Example 0... Packets: 6494 Displayed: 371 Marked: 0 Load time: 0:00:307 Profile: Wireless

图 11.27

- 由图 11.28 可知，客户机从每次发出数据库查询，到收到来自服务器的查询响应所耗时间约为 35 毫秒。

① 译者注：通过后面的抓包文件可以看出，这条链路应该是 wireless 链路，这么重要的信息作者却偏未指明。

② 译者注：原文是“If you see problems in the network, solve them as we learned in previous chapters. But, there are some network issues that can influence database behavior. In the following example, we see the behavior of a client that works with the server over a communication line with a roundtrip delay of 35 to 40 ms”。作者举的这个例子漏洞百出，由于译者并未参与此次故障排除，译文只能按字面意思直译。

No.	Time	Source	Destination	Protocol	Info
1981	35.833309	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [PSH, ACK] Seq=...
1982	35.869385	192.168.20.88	192.168.10.80	TCP	wv-csp-udp-cir > vfo [PSH, ACK] Seq=...
1983	35.869930	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [PSH, ACK] Seq=...
1984	35.905654	192.168.20.88	192.168.10.80	TCP	wv-csp-udp-cir > vfo [PSH, ACK] Seq=...
1985	35.906194	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [PSH, ACK] Seq=...
1986	35.944428	192.168.20.88	192.168.10.80	TCP	wv-csp-udp-cir > vfo [PSH, ACK] Seq=...
1987	35.953804	192.168.20.88	192.168.10.80	TCP	vfo > wv-csp-udp-cir [PSH, ACK] Seq=...

图11.28

- 由于这条编号为 8 的 TCP 数据流由 371 个数据包构成,因此客户机完成数据库查询所耗时间约为 13 秒 (371×35 毫秒)^①。再加上(因无线网络质量所导致的)某些 TCP 重传,使得操纵该客户机的用户要等待 10~15 秒(或更长的时间)才能完成本次数据库查询任务。
- 在这种情况下,应与 DBA 协商如何通过改进数据库应用程序来降低过往于网络的数据包的数量。或者,可以让那台客户机不要通过无线网络去访问服务器,改用其他方式(比如,终端服务器或 Web 方式)去访问服务器^②。

4. 有时,通过 Wireshark 抓包分析,还可以发现应用程序自身的问题。在图 11.29 所示的 Wireshark 截屏中,可以看见客户机向服务器连续发送了 5 个身背 TCP retransmissions 字样的 TCP 数据包。之后,客户机与服务器之间又重新通过三次握手,建立起了 TCP 连接^③。这很像是 TCP 故障,但只有当用户在该应用程序客户端软件的特定窗口内执行某些操作时,才能在 Wireshark 抓包主窗口的数据包列表区域内看见上述景象。出现这样的问题,是由于该应用程序设计失当,每当用户执行上操作时,服务器端软件就会停止响应客户端软件发起的 TCP 连接^③。

No.	Time	Source	Destination	Protocol	Info
274	0.078889	192.168.3.50	192.168.200.227	TCP	http > vrtp [ACK] Seq=1 Ack=59884 Win=...
275	0.380166	192.168.200.227	192.168.3.50	TCP	[TCP Retransmission] TCP segment of...
276	0.983678	192.168.200.227	192.168.3.50	TCP	[TCP Retransmission] TCP segment of...
277	2.195589	192.168.200.227	192.168.3.50	TCP	[TCP Retransmission] TCP segment of...
278	4.804757	192.168.200.227	192.168.3.50	TCP	[TCP Retransmission] TCP segment of...
279	9.432867	192.168.200.227	192.168.3.50	TCP	[TCP Retransmission] TCP segment of...
280	18.989050	192.168.200.227	192.168.3.50	TCP	rcts > http [SYN] Seq=0 Win=65535 Len=...
281	18.994054	192.168.3.50	192.168.200.227	TCP	http > rcts [SYN, ACK] Seq=0 Ack=1 Win=...
282	18.994085	192.168.200.227	192.168.3.50	TCP	rcts > http [ACK] Seq=1 Ack=1 Win=655...
283	18.994264	192.168.200.227	192.168.3.50	TCP	[TCP segment of a reassembled PDU]
284	18.994280	192.168.200.227	192.168.3.50	TCP	[TCP segment of a reassembled PDU]
285	19.000271	192.168.3.50	192.168.200.227	TCP	http > rcts [ACK] Seq=1 Ack=537 Win=...

图11.29

① 译者注:应该是 $371/2 \times 35$ 毫秒。

② 译者注:原文是“*In this case, you should consult with the DBA on how to significantly reduce the number of packets that run over the network; or you can move to another way of access, for example, terminal server or web access*”。

③ 译者注:整段原文是“*Another problem that can happen is that you will have a software issue that will reflect in the capture file. If you have a look at the following screenshot, you will see that there are five retransmissions (1), and then a new connection is opened from the client side (3). It looks like a TCP problem but it occurs only in a specific window in the software. It is simply a software procedure that stopped processing, and this stopped the TCP from responding to the client (2)*”。对原文无语,译者也只能译到这个份上了。还有就是,作者描述的这一场景似乎与数据库关系不大。

11.7.3 幕后原理

常言道，隔行如隔山，网络技术人员肯定玩不转数据库。不过，网络技术人员有义务配合 DBA 去排除与数据库有关的故障，这也是作者在本章安排这些内容的目的。

11.7.4 拾遗补缺

在 Wireshark 抓包主窗口的数据包列表区域中，选中一个隶属于某条数据库客户端与数据库服务器间 TCP 会话的数据包，点击右键，选择右键菜单中的 **Follow TCP Stream** 菜单项，Follow TCP Stream 窗口会立刻弹出^①。可以让 DBA 通过该窗口所示内容，来了解经由网络传输了哪些数据。

若数据库客户端与数据库服务器之间的网络延迟较高（比如，数据库客户端通过移动网络访问数据库服务器），则实际操纵应用程序的用户就会觉得卡顿。此时，需考虑让用户采用其他方式来访问数据库服务器（比如，采用终端服务器或 Web 方式）。

数据库服务器的运作方式同样重要。若某台数据库服务器需要从另一台文件服务器获取数据，即便数据库客户端与该数据库服务器之间通信正常，但只要前两者之间的通信出现问题，还是会对数据库应用产生影响。在排除数据库相关故障时，请务必弄清待查目标的各个环节^②。

对网络技术人员而言，最重要的是要跟真正有本事的 DBA 处好关系，说不定哪天就能用上他们。

① 译者注：原文是 “When you right-click on one of the packets in the database client to the server session, a window with the conversation will open”。

② 译者注：原文是 “An important issue is how the database works. If the client is accessing the database server, and the database server is using files shared from another server, it can be that the client-server works great; but the problems come from the database server to the shared files on the file server. Make sure you that know all these dependencies before starting with your tests”。

第12章

SIP、多媒体和IP电话

本章将介绍如何使用 Wireshark 去诊断并解决与 IP 电话、语音及视频有关的故障。本章涵盖以下主题：

- ▶ 内置于 Wireshark 的 IP 电话及多媒体流量专用分析工具的用法；
- ▶ SIP 故障分析；
- ▶ RTP/RTCP 故障分析；
- ▶ 视频及视频监控应用排障场景；
- ▶ IPTV 应用排障场景；
- ▶ 视频会议应用排障场景；
- ▶ 排除 RSTP 故障。

12.1 简介

包括 IP 电话、视频会议、视频监控以及在线教育在内的各类多媒体应用是现代化网络通信的重要组成部分。

这些多媒体应用对网络的要求与前面介绍的众多企业网应用有所不同。像 HTTP、E-mail 以及文件共享这样的企业网应用需要较高的带宽，而 IP 电话等多媒体应用所需带宽不高（只需要区区 100kbit/s 的带宽），但对网络的延迟及抖动（延迟的差异）却非常敏感。对于那些拥有远程分支机构的企业而言，在各分支机构网络内，绝大多数企业网应用都会引发较高的下载流量，而诸如视频监控系统之类的多媒体应用却会触发较高的上传流量，因为总部要想监控各个分支机构，就得从各分支机构网络下载视频监控图像。

本章会关注包括语音、视频在内的多媒体应用以及它们在网络中的运作方式，同时还会介绍当这些应用发生故障时，如何利用 Wireshark 排除故障。本章涉及：会话初始协议（Session Initiation Protocol, SIP）、实时协议/实时传输控制协议（Real Time Protocol /Real-time Transport Control Protocol, RTP/RTCP）、实时流协议（Real Time Streaming Protocol, RTSP）等常用多媒体协议。

本章首先会介绍内置于 Wireshark 的 IP 电话及多媒体流量专用分析工具。然后，将介绍如何解决 SIP 故障，以及如何排除 RTP/RTCP 会话故障。最后，会关注如何排除各种视频系统（比如，视频会议及视频监控系统）故障。

12.2 使用内置于Wireshark 的IP电话及多媒体流量专用分析工具

先来了解一下内置于 Wireshark 的语音、视频及多媒体专用分析工具。

12.2.1 准备工作

处理语音通话、视频电话会议或其他多媒体会话应用故障时，请按图 12.1 所示安置 Wireshark 主机并在 LAN 交换机上开启端口镜像功能。

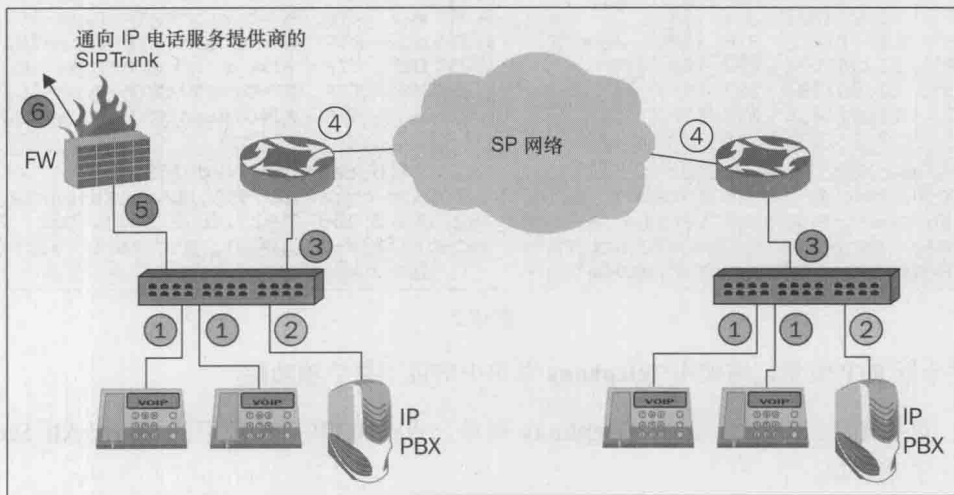


图12.1

1. 若怀疑是用户使用的 IP 电话（图 12.1 中的①处）故障，请把 Wireshark 主机接入交换机，开启交换机的端口镜像功能，将连接 IP 电话的端口的流量重定向至 Wireshark 主机。

2. 若怀疑是各站点内的整个 IP 电话网络故障（IP PBX 故障，图 12.1 中的②处），请把 Wireshark 主机接入交换机，开启交换机的端口镜像功能，将连接 IP PBX 的端口的流量重定向至 Wireshark 主机。
3. 若怀疑站点间网络存在连通性故障（WAN 线路故障，图 12.1 中的③、④处），请把路由器上 WAN 端口的流量重定向给 Wireshark 主机^①。
4. 若怀疑 SIP Trunk 故障，请设法将防火墙的内口（图 12.1 中的⑤处）或外口（图 12.1 中的⑥处）流量重定向至 Wireshark 主机。防火墙的外口所连线路既有可能是 Internet 线路，也有可能是通向 IP 电话服务提供商的专线。

12.2.2 操作方法

请在 Wireshark 主抓包窗口中点击 **Telephony** 菜单，如图 12.2 所示。

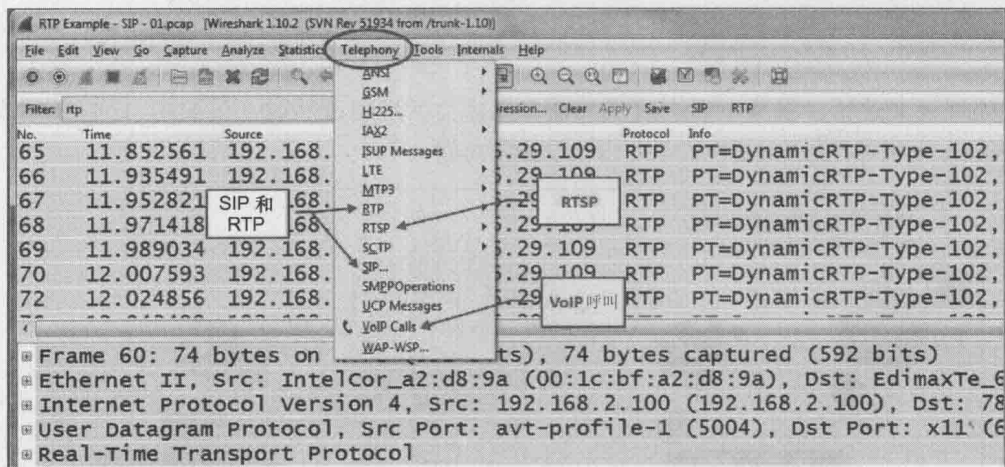


图 12.2

要分析 RTP 流量，需使用 **Telephony** 菜单中的以下菜单项功能。

1. 可按图 12.3 所示，选择 **Telephony** 菜单，点击 **RTP** 菜单项下的 **Show All Streams** 子菜单项。
2. RTP Streams 窗口会立刻弹出，如图 12.4 所示。
3. 通过 RTP Streams 窗口，可以获知以下细节。

^① 译者注：若图中的那对路由器通过串行 SDH 线路互连，请问怎么用 Wireshark 抓包？退一步来讲，即便那对路由器通过 SMTP 线路互连，如果存在连通性故障，无需使用 Wireshark 抓包应该也能判断故障所在。

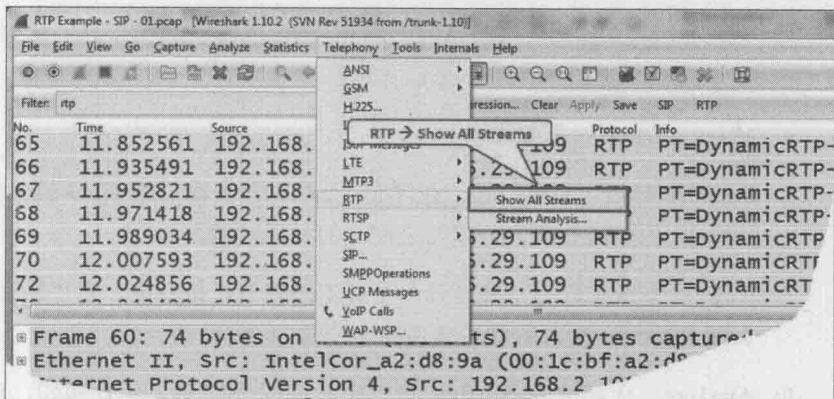


图 12.3

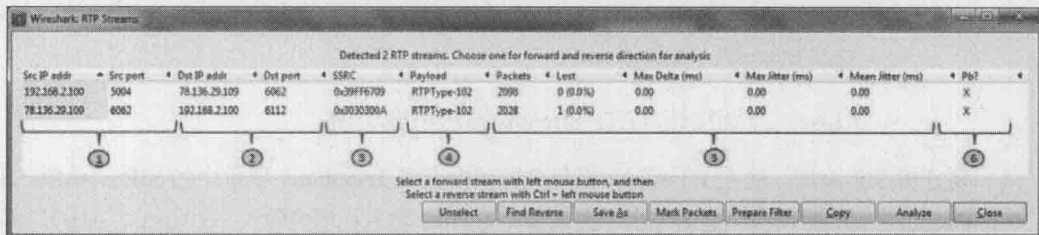


图 12.4

- RTP 流的源、目 IP 地址及源、目 UDP 端口号。
- 同步源 (SSRC) 标识符 (RTP 流的标识符)。
- RTP 流量的净载类型 (通常所指为编码类型)。
- RTP 流的属性信息, 包括: 抓到的数据包的总数、丢包数、数据包间的最长间隔时间、最长抖动时间及平均抖动时间。
- Pb?: 表示 RTP 流中发生的常见问题。

在 RTP Streams 窗口的底部, 还有如下按钮。

- **Unselect:** 选中 RTP Streams 窗口中的一条语音流之后, 点击此按钮, 便会取消之前的选择。
- **Find reverse:** 由语音通话 (或多媒体呼叫) 所生成的 RTP 数据流必然是一来一回, 选中 RTP Streams 窗口中的一条 RTP 流之后, 点击此按钮, 该 RTP 流便会与与之配套的逆向 RTP 流同时以亮灰色面目示人。
- **Save as:** 点此按钮, 可将选中的 RTP 流以 rtpdump 文件格式存盘。欲知更

多与 rtpdump 文件格式有关的内容,请访问 <http://www.cs.columbia.edu/irt/software/rtptools/>。

- **Mark packets:** 不起作用。
 - **Prepare filter:** 点此按钮,在 Wireshark 抓包主窗口的 Filter 输入栏内,便会基于选中的 RTP 流,创建与之配套的显示过滤器。
 - **Copy:** 先点此按钮,再打开一个文本编辑器,按右键,点“粘贴”,便可将 RTP Streams 窗口中的内容复制进该文本编辑器。
 - **Analyze:** 选择 RTP Streams 窗口中的 RTP 流之后,点此按钮,RTP Stream Analysis 窗口会立刻弹出。在 Wireshark 抓包主窗口中先选中一个 RTP 数据包,再点击 **Telephony** 菜单,选择 RTP 菜单项中的 **Stream** 子菜单项,也会导致相同窗口的弹出。
 - **Close:** 点此按钮,RTP Streams 窗口将会关闭。
4. 要分析 SIP 流量,请在 Wireshark 抓包主窗口内点击 **Telephony** 菜单中的 **SIP** 菜单项,**SIP Packet Counter** 窗口会立刻弹出。在该窗口的 Filter 输入栏里内输入 ip (udp、sip),再点击 **Create Stat** 按钮,**SIP statistics with filter: ip (udp/sip)** 窗口会立刻弹出,如图 12.5 所示。

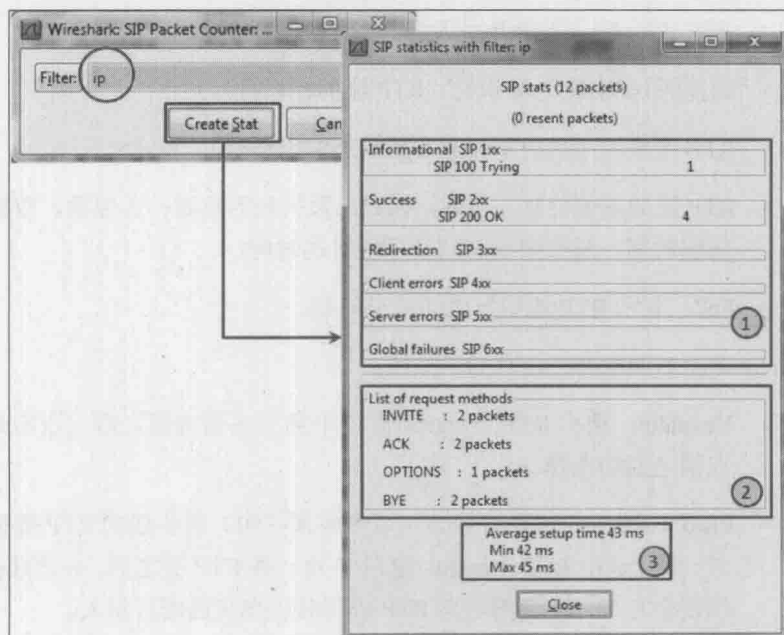


图 12.5

5. 通过 SIP statistics with filter: ip (udp/sip) 窗口, 可了解到以下与 SIP 流量有关的统计信息。

- 含各种状态码的 SIP 响应数据包的个数 (显示在图 12.5 中标以①的区域内)。
- 含各种 SIP 方法的 SIP 请求数据包的个数 (显示在图 12.5 中标以②的区域内)。
- SIP 会话的建立时间 (包含最大、最小及平均值) (显示在图 12.5 中标以③的区域内)。

6. 要分析 RTSP 流量, 请在 Wireshark 抓包主窗口内选择 **Telephony** 菜单, 点击 SIP 菜单项中的 **Packet Counter** 子菜单项, RTSP/Package Counter 窗口会立刻弹出。在该窗口的 Filter 输入栏里输入 ip (或 rtsp, 也可以什么都不输), 再点击 **Create Stat** 按钮, **RTSP/Package counter with filter: ip (rtsp)** 窗口会立刻弹出, 如图 12.6 所示。

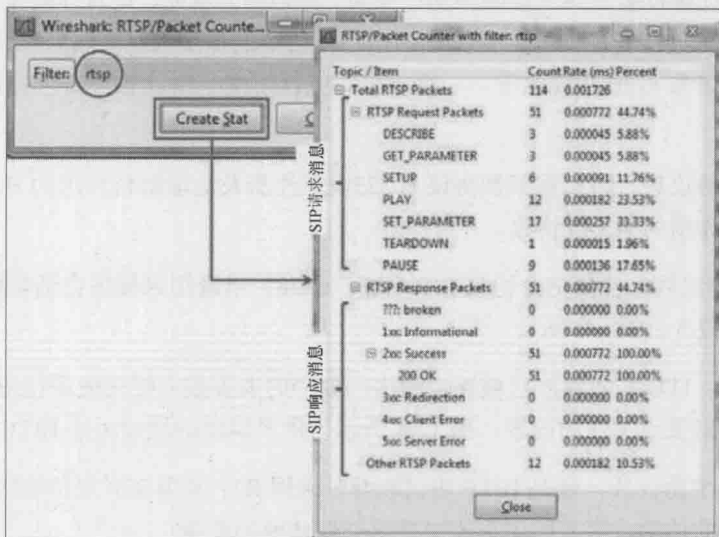


图12.6

7. 要分析已抓取的数据包中的 VoIP 通话状况, 请在 Wireshark 抓包主窗口内点击 **Telephony** 菜单中的 **VoIP Calls** 菜单项, VoIP Calls 窗口会立刻弹出, 如图 12.7 所示。

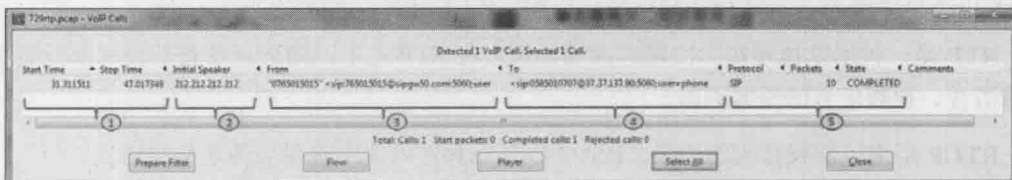


图12.7

8. 通过 VoIP Calls 窗口，可以了解到以下 VoIP 通话状况。

- VoIP 呼叫的开始和结束时间（显示在图 12.7 中标以①的区域内）。
- 发起呼叫的 IP 地址（显示在图 12.7 中标以②的区域内）。
- 发起呼叫的 SIP 地址（主叫方）（显示在图 12.7 中标以③的区域内）。
- 接收呼叫的 SIP 地址（被叫方）（显示在图 12.7 中标以④的区域内）。
- 建立呼叫的信令协议（一般都是 SIP 协议）、信令协议（SIP 协议）数据包的个数以及信令协议最终的状态。

12.2.3 幕后原理

应把语音、视频及多媒体流量的传递一分为二来看待。首先，是媒体流量（即实际的语音或视频流量）的传递；其次，是信令流量的传递，信令的作用包括：语音/视频呼叫的建立和终止、邀请参与者参与通话/视频等。一直以来，有以下两个协议族可用来提供语音或视频的信令功能。

- **ITU-T** 协议族，包括框架性协议 H.323、起注册及地址解析作用的 H.225 协议，以及起控制作用的 H.245 协议。
- **IETF** 协议族，包括信令协议 SIP（RFC 6878）以及用来描述会话参数的 SDP（会话描述协议）（RFC 4566）。

在过去几年，ITU-T 协议族已被逐步淘汰，如今的大多数多媒体应用使用的都是 IETF 协议族，这也是本章重点关注的内容。图 12.8 所示为供多媒体应用使用的 IETF 协议族的架构。

无论是 ITU-T 协议族，还是 IETF 协议族都是采用 RTP 和 RTCP 来传输实际的媒体流量，前者用于媒体流的传输，后者则用来控制并保证媒体流的质量。

图 12.8 所示为在 IP 网络中用来传递多媒体流量的 IETF 协议族的架构。

SIP 是一种作用于末端用户之间的信令协议。SDP 数据由 SIP 消息承载，用来描述会话。**RSTP** 用来行使流量（一般为视频流量，比如，IPTV 流量）控制功能。

RTP 是一种用来传递实际的媒体流的协议。在 RTP 之上，则运行着若干种用来压缩及解压缩语音、视频信号的编解码器。

RTCP 用来控制媒体流的质量，**RSVP** 则是一种在网络中确保服务质量的协议。

图 12.8 中呈现的所有协议都基于 TCP/IP 协议族，本章稍后会详尽探讨其中的大多数协议。

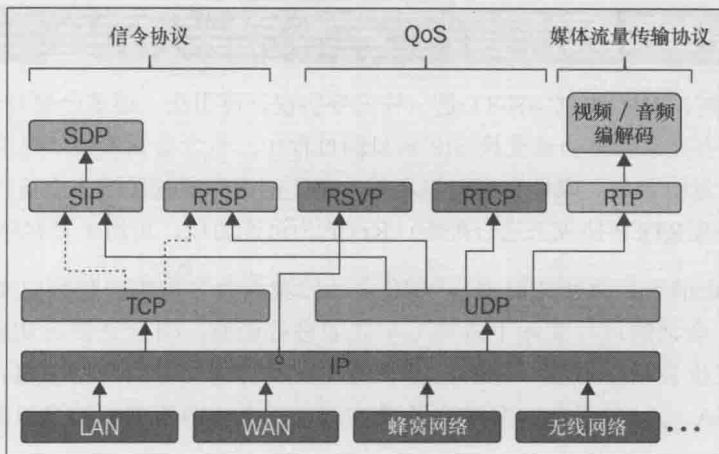


图12.8

12.2.4 拾遗补缺

由于 SIP 协议所使用的端口号固定，因此 Wireshark 总是把目的端口为 5060 (SIP) 或 5061 (SIP over TLS，即受 TLS 保护的安全 SIP) 的 TCP 或 UDP 数据包识别为 SIP 协议数据包。虽然 IETF 标准规定 SIP 协议报文可封装在 TCP 报文段或 UDP 数据报内，但在一般情况下，SIP 协议报文总是以 UDP 数据报的面目示人。

RTP 和 RTCP 协议所使用的端口号并不固定，RTP 协议流量的源和目的端口号总是为偶数，而与之对应的 RTCP 协议流量的源和目的端口号则总是分别采用下一个奇数。比方说，如 RTP 协议流量的源和目的端口为 5004 和 2006，那么与之对应的 RTCP 协议流量的源和目的端口号将会是 5005 和 2007。所以说，在默认情况下，Wireshark 是识别不出 RTP 和 RTCP 协议流量的，只能将那些流量解析为 UDP 数据包。

要想让 Wireshark 自动识别 RTP 和 RTCP 协议流量，有以下两种方法。

1. 在 Wireshark 抓包主窗口的数据包列表区域内选择一个隶属于 RTP 流的 UDP 数据包，点击右键，选择右键菜单中的 **Decode as** 菜单项，在弹出的 Decode as 窗口的右边找到 **RTP** 协议，点击 **OK** 按钮，关闭窗口。
2. 在 Wireshark 抓包主窗口中，点击 **Edit** 菜单里的 **Preferences** 菜单项。在弹出的 Preferences 窗口中，点击左边 **Protocol** 配置选项之前的+号，选择其中的 **RTP** 协议，勾选右边与 **Try to decode RTP outside of conversations** 挂钩的复选框。在大多数情况下，可让 Wireshark 以这种方式来自动解码 RTP 协议数据包。

上述方法同样适用于 RTCP 协议流量的自动识别。

12.3 SIP故障分析

如上一节所述，SIP（RFC 6878）是一种信令协议，可用在一或多个参与者之间创建、修改及终止会话。在建立媒体会话交换 SIP 消息的过程中，包含会话参数信息的 SDP 消息也会被封装在 SIP 消息中传递。媒体会话创建之后，实际的语音或视频流才会被封装进 RTP 数据包中传输，可通过 RTCP 协议来进行控制（RTCP 为可选协议，可用于多媒体应用）。

在 IETF 制定的 SIP 标准文档中，把 SIP 会话的端点命名为用户代理（User Agent, UA）。因此，创建 SIP 会话的过程实际上就是 UA 之间进行协商，对会话参数达成一致意见的过程。至于提供定位会话参与者、注册、呼叫转发以及其他服务的网络主机，则被称为服务器（server），UA 可以向服务器发送注册请求消息、会话邀请请求消息以及其他各种请求消息。

本节将介绍用来传递媒体流量的 IETF 协议族中的信令协议——SIP，同时会详述如何使用 Wireshark 来排除 SIP 故障。下一节将介绍 RTP 和 RTCP 协议。

可把 IP 上的语音和视频故障分为以下两大类。

- 呼叫建立、修改及终止类故障：诸如提起听筒后听不见拨号音、听得见拨号音但不能拨号、能拨号但被叫方电话无法振铃之类的情況都属于此类故障。此类故障一般都跟信令协议有关。
- （语音或视频）信号质量类故障：此类故障与信号质量有关，比如，语音通话过程中的信号干扰问题（有噪音）、视频画面停顿问题等。此类故障通常都是由网络问题、RTP 协议问题或各种介质问题所导致。

本节讨论的是如何解决第一类故障，第二类故障将放到下一节讨论。



以上两类 IP 上的语音和视频故障并不一定都跟网络故障有关，也有可能是由设备（比如，电话交换机或末端设备）配置问题所引发。

12.3.1 准备工作

在处理呼叫建立、修改及终止类故障时，首先应考虑信令协议有无故障。SIP 是在 IP 网络环境中用来传递语音和视频的信令协议，它有两种主要的消息类型：SIP 呼叫控制请求消息和 SIP 响应消息。在 SIP 呼叫控制请求消息中，包含的是 SIP 方法，即 SIP 会话的一方发出的命令；SIP 会话的另一方会发出 SIP 响应消息作为应答。

排除 SIP 协议故障时，应重点关注 SIP 响应消息所包含的内容。

12.3.2 操作方法

把 Wireshark 主机接入网络，在交换机上开启端口镜像功能，将必要的流量重定向至 Wireshark 主机。以下所列在排除 SIP 协议故障时，需要关注的地方。

1. 当某一 UA 希望建立多媒体会话时，将会向远端 UA 发送包含 **INVITE** 方法的 SIP 呼叫控制请求消息，后者也会发出 SIP 响应消息回应前者，两者相互交流的过程如图 12.9 所示。

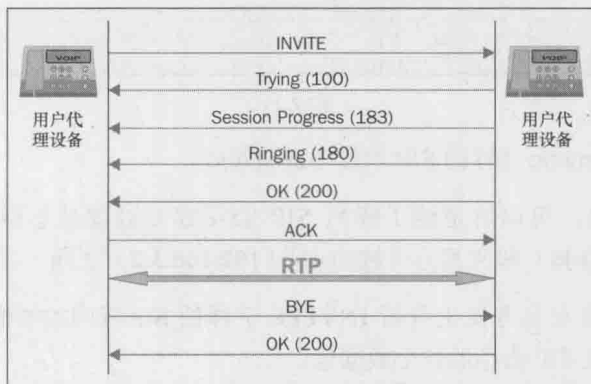


图12.9



注意

按 SIP 协议的行话来讲，末端用户设备被称为用户代理设备（**User Agent, UA**）。UA 既可以是发起呼叫也可以接收呼叫。IP 电话、摄像头、具备语音或视频功能的客户端软件，以及能参与 SIP 会话的任何软、硬件都属于 UA。

2. 图 12.10 中的 Wireshark 截屏清晰地展示了 VoIP 通话双方建立 SIP 会话时的过程。

No.	Time	Source	Destination	Protocol	Info
22	0.000000	93.172.121.146	192.168.3.2	SIP/SDP	Request: INVITE sip:90526613742@213.8.115.84:5068 , with ses
23	0.000000	192.168.3.2	93.172.121.146	SIP	Status: 100 Trying
27	0.960000	192.168.3.2	93.172.121.146	SIP/SDP	Status: 183 Session Progress , with session description
599	3.890000	192.168.3.2	93.172.121.146	SIP	Status: 180 Ringing
1834	9.950000	192.168.3.2	93.172.121.146	SIP/SDP	Status: 200 OK , with session description
1877	10.150000	93.172.121.146	192.168.3.2	SIP	Request: ACK sip:90526613742@213.8.115.84:5068;user=phone
3426	17.850000	93.172.121.146	192.168.3.2	SIP	Request: BYE sip:90526613742@213.8.115.84:5068;user=phone
3427	17.850000	192.168.3.2	93.172.121.146	SIP	Status: 200 OK

图12.10

3. 要想更为详细地了解 VoIP 通话双方相互交流的内情，请在抓包主窗口中点击 **Statistics** 菜单下的 **Flow Graph** 菜单项，Flow Graph 窗口会立刻弹出。同时点选该窗口中的 **Displayed packets**、**General flow** 和 **Standard source/destination addresses** 单选框后，再点击 **OK** 按钮，**Graph Analysis** 窗口会立刻弹出，如图 12.11 所示。
4. 对于一次正常的 SIP 会话建立过程而言，在图 12.11 所示的 Graph Analysis 窗口中，首先应能看到身负 INVITE 字样的 SIP 消息，而从对端发来的身背 Trying、Session

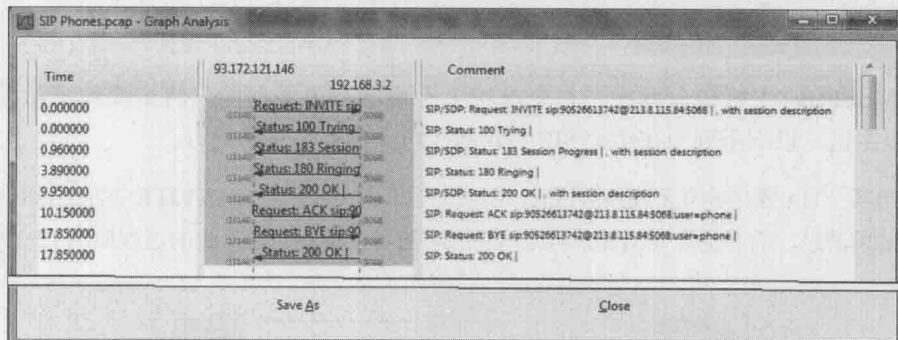


图12.11

Progress、Ringing 字样的 SIP 消息将紧随其后。

5. 通过图 12.11, 可以清楚地了解到 SIP 会话建立过程是怎样在发起方 (主叫方) (93.172.121.146) 和应答方 (被叫方) (192.168.3.2) 之间一步步进行的。

- 会话发起方发出身背 **INVITE** 字样的 SIP 呼叫控制请求数据包, 这一定是建立 SIP 会话的首个数据包。
- 会话应答方会立刻回复身背 **Trying** (状态码 100) 和 **Session progress** (状态码 183) 字样的 SIP 响应数据包, 三秒之后, 还会回复身背 **Ringing** (状态码 180) 字样的 SIP 响应数据包。此后, 当有人提起电话听筒时, 会话应答方还会继续发出身背 **OK** 字样的 SIP 响应数据包。



注意

建立 SIP 会话时, 被叫方未必会回复之前提到的全部三种 SIP 响应消息, 在某些情况下, 可能只会回复其中的一种或两种 SIP 响应消息, 但 SIP 会话仍能正常建立。本章后文会介绍当出现这种情况时, 如何辨别是否存在故障。

- 会话发起方发出身背 **ACK** 字样的 SIP 呼叫控制请求数据包, 至此, SIP 会话成功建立。

6. 在 SIP 会话建立阶段, 若抓到了身背 **ERROR** 字样的 SIP 响应消息, 则 SIP 会话将无法建立。



注意

SIP 消息通过 UDP 报文段来承载, 而 UDP 协议是一种非面向连接、不可靠的传输层协议。在网络质量堪忧的情况下, 包含 **INVITE** 方法的 SIP 呼叫控制请求消息有可能会在传输途中丢失。所以说, 如果 SIP 会话发起方收不到任何 SIP 响应消息, 则很有可能是其发出的包含 **INVITE** 方法的 SIP 呼叫控制请求消息因网络质量问题而未能抵达 SIP 会话接收方。

7. 若 UA 之间的通话要通过电话交换机来转接, 则 SIP 会话的建立过程应该向图 12.12 中 Wireshark Graph Analysis 窗口显示的那个样子。诸如 IP 电话交换机、呼叫管理器、IP PBX 之类的 VoIP 术语, 读者一定都耳熟能详, 其实它们都是指能在末端设备之间“居中调停”的电话交换机。这些 VoIP 术语所指代的设备, 在 SIP 术语系里被统称为“代理服务器”(Proxy Server), 下一节会就此展开讨论。

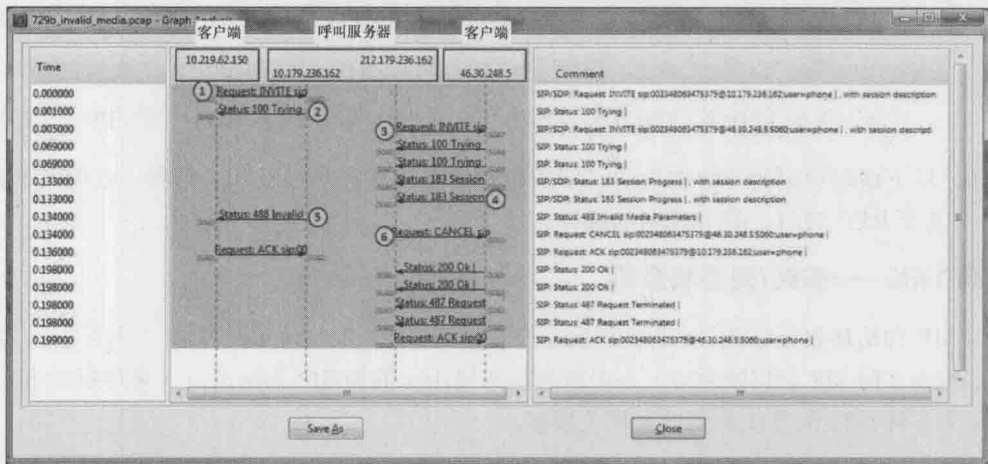


图12.12

8. 通过图 12.12 所示的 Wireshark 截屏, 可以看出, 该 IP 电话交换机有 2 个接口, 分别连接内网和外网, IP 地址分别为 212.179.236.162 和 10.179.236.162。

- 左侧的 UA (10.219.62.150) 向 IP 电话交换机发出身背 **INVITE** 字样的 SIP 控制请求消息 (在图 12.12 中被标记为①)。
- IP 电话交换机向左侧的 UA 回复身背 **Trying** 字样的 SIP 响应消息 (在图 12.12 中被标记为②)。
- IP 电话交换机向右侧的 UA 发出身背 **INVITE** 字样的 SIP 控制请求消息 (在图 12.12 中被标记为③)。
- 右侧的 UA 向 IP 电话交换机回复身背 **Trying** (状态码为 100) 和 **session progresses** (状态码为 183) 字样的 SIP 响应消息 (在图 12.12 中被标记为④)。
- 片刻之后, 当 IP 电话交换机感知到右侧的 UA 无进一步的响应, 于是发出包含状态 **488** 的 SIP 响应消息, 通知主叫方 (左侧的 UA) 被叫方 (右侧的 UA) 没有应答, 同时给出了没有应答的原因 (**invalid**) (在图 12.12 中被标记为⑤)。

- IP 电话交换机向右侧的 UA 发出身背 **Cancel** 字样的 SIP 控制请求消息（在图 12.12 中被标记为⑥）。
9. 要想定位 SIP 协议故障，请按下列步骤行事。
- 拥有一张最新的包含所有部件的网络拓扑结构图。
 - 检查 Wireshark 抓包文件，看看是否出现了身背 **ERROR** 字样的 SIP 数据包。
 - 弄清 VoIP 客户端或服务器为什么会发出身背 **ERROR** 字样的 SIP 数据包。
10. 以下诸表所列为 SIP 响应消息中所包含的各种状态码以及相关解释，它们几乎都定义于 RFC 3261，另有说明的除外。

状态码 1XX——临时/信息状态码

当 SIP 会话接收方收到（SIP 会话发起方发出的）SIP 控制请求消息，且正在做进一步处理时，便会（向 SIP 会话发起方）发出包含状态码 1xx 的 SIP 响应消息（来通报此事）。下表所列各种临时/信息状态码以及相关描述。

状态码	事件名	描 述
100	Trying	表示服务器已收到并接受了（SIP 会话发起方发出的）SIP 控制请求消息，且正在对本次呼叫做进一步的处理
180	Ringing	表示接收本次呼叫的 UA 正在（发出振铃）提醒用户。当接收本次呼叫的 UA 发出振铃时，将会向主叫端 UA 回发包含此状态码的 SIP 响应消息
181	Call forward	表示呼叫被前传至另一目的地
182	Queued	表示被叫方暂时不便接听（比如，占线），服务器会暂时接管呼叫，以便稍后再试
183	Session progress	表示接收方服务器正在处理会话。与会话处理有关的详细信息，将会由包含此状态码的 SIP 响应消息头部中的某些字段来承载

状态码 2XX——成功状态码

当 SIP 会话接收方成功接收、识别并接受了（SIP 会话发起方发出的）SIP 控制请求消息时，便会（向 SIP 会话发起方）发出包含状态码 2xx 的 SIP 响应消息（来通报此事）。下表所列为各种成功状态码以及相关描述。

状态码	事件名	描 述
200	OK	表示（SIP 会话发起方发出的）SIP 控制请求消息已被（SIP 会话接收方）接受并成功处理。
202 （定义于 RFC 3265）	Accepted	表示（SIP 会话发起方发出的）SIP 控制请求消息已被（SIP 会话接收方）接受且正在处理，但操作尚未完成

状态码 3XX——重定向状态码

当 SIP 会话接收方只有采取重定向操作才能完成 SIP 会话发起方发出的请求时，便会向后者发出包含状态码 3XX 的 SIP 响应消息（来通报此事）。下表所列为各种重定向状态码以及相关描述。

状态码	事件名	描 述
300	Multiple choices	表示接收 SIP 控制请求消息的服务器在解析过消息中所含目的地址之后，发现有很多转发选择，亦即呼叫可被转发至多个指定位置。服务器可在包含此状态码的 SIP 响应消息的消息主体中列出资源的特征及位置，好让（发出 SIP 控制请求消息的）UA 从中选择一个最合适的呼叫目的地址，并将 SIP 控制请求消息重定向至该地址
301	Moved permanently ^①	
302	Moved temporarily	
305	Use proxy ^②	
380	Alternative service	

状态码 4XX——客户端错误状态码

当（SIP 会话发起方发出的）SIP 控制请求消息包含了错误的语法，或消息不能被 SIP 会话接收方（服务器）解析时，后者便会向前者发出包含状态码 4xx 的 SIP 响应消息（来通报此事）。下表所列为各种客户端错误状态码以及相关描述。

状态码	事件名	描 述
400	Bad request	表示 SIP 会话接收方（服务器）因 SIP 控制请求消息中包含了错误的语法而无法对其进行解析
401	Unauthorized	表示 SIP 会话接收方（服务器）收到了 UA 发出的 SIP 控制请求消息，但需要对 UA 执行认证
402	Payment required	预留以供将来使用
403	Forbidden	表示 SIP 会话接收方（服务器）能识别 UA 发出的 SIP 控制请求消息，但拒绝对其进行处理。UA 不应重复发送之前曾发出过的 SIP 控制请求消息
404	Not found ^②	
405	Method not allowed	
406	Not acceptable	

① 译者注：原文对 SIP 状态码 301~380 的描述照抄自 RFC 3261。在译者看来，若未对 SIP 消息的格式做详细介绍，按字面意思翻译这些内容没有任何意义，且作者添加这些内容只是为了凑字数，跟本书主题无关，故而掠过不译。

② 译者注：原文对 SIP 状态码 404~493 的描述照抄自 RFC 3261。在译者看来，若未对 SIP 消息的格式做详细介绍，按字面意思翻译这些内容没有任何意义，且作者添加这些内容只是为了凑字数，跟本书主题无关，故而掠过不译。

续表

状态码	事件名	描 述
407	Proxy authentication required	
408	Request timeout	
410	Gone	
413	Request entity too large	
414	Request-URI too long	
415	Unsupported media type	
416	Unsupported URI scheme	
420	Bad extension	
421	Extension required	
423	Interval too brief	
424	Bad location information	
428	Use Identity header	
429	Provide referrer identity	
433	Anonymity disallowed	
436	Bad identity info	
437	Unsupported certificate	
438	Invalid identity header	
470	Consent needed	
480	Temporarily unavailable	
481	Call/transaction does not exist	
482	Loop detected	
483	Too many hops	
484	Address incomplete	
485	Ambiguous	
486	Busy here	
487	Request terminated	

续表

状态码	事件名	描 述
488	Not acceptable here	
491	Request pending	
493	Undecipherable	

状态码 5XX——服务器错误状态码

若 SIP 会话发起方收到了包含状态码 5xx 的 SIP 响应消息,则表示 SIP 会话接收方(服务器)无法解析本机发出的有效的 SIP 控制请求消息。下表所列为各种服务器错误状态码以及相关描述。

状态码	事件名	描 述
500	Server internal error	表示服务器因异常状况无法处理 UA 发出的 SIP 呼叫控制请求消息
501	Not implemented	表示(接收 SIP 控制请求消息的)服务器不支持(发出此消息的) UA 请求执行的功能
502	Bad gateway ^①	
503	Service unavailable	
504	Server time out	
505	Version not supported	
513	Message too large	

状态码 6XX——全局故障状态码

收到包含状态码 6xx 的 SIP 响应消息,则意味着 SIP 会话发起方之前(通过发出 SIP 控制请求消息)所请求的功能在任何一台服务器都无法执行。下表所列为各种全局故障状态码以及相关描述。

状态码	事件名	描 述
600	Busy everywhere	表示能成功联络上接收方系统,但此时该用户繁忙,无意接受呼叫,且无任何可替代的目标端系统(比如,语音信箱服务器)能够接收呼叫
603	Decline	表示能成功联络上接收端 UA,但该用户明确表态不愿意参与呼叫,且无任何可替代的目标端系统(比如,语音信箱服务器)愿意接收呼叫

① 译者注:原文对 SIP 状态码 502~513 的描述照抄自 RFC 3261。在译者看来,若未对 SIP 消息的格式做详细介绍,按字面意思翻译这些内容没有任何意义,且作者添加这些内容只是为了凑字数,跟本书主题无关,故而掠过不译。

续表

状态码	事件名	描 述
604	Does not exist anywhere	当 SIP 会话发起方希望联络某个用户时, 会向服务器发出 SIP 控制请求消息, 在消息中会包含该用户所在位置 (URI)。若服务器所持权威信息表明, 该用户不存在, 便会回复包含状态码 604 的 SIP 响应消息
606	Not acceptable	表示 SIP 会话发起方能成功联络上用户的代理 (agent), 但在前者发出的 SIP 控制请求消息的消息主体中, 由 SDP 描述的会话中的某些属性不被后者接受

12.3.3 幕后原理

SIP 是一种应用层协议, 可用来建立、维护及终止两个或多个端节点之间的呼叫。

参与 SIP 会话的网络大致被为两类——客户端和服务端。

- 客户端是指发送 SIP 呼叫控制请求消息的实体 (或应用程序)。
- 服务器则是发送 SIP 响应消息, 对 SIP 呼叫控制请求消息进行回应的实体 (或应用程序)。

参与 SIP 会话的另一类网络实体被称为网关 (gateway), 其作用是把 SIP 网络和公众交换电话网络 (PSTN) 连接在一起。

如图 12.13 所示, 每个 SIP 终端都是由用户代理客户端 (User Agent Client, UAC) 和用户代理服务器 (User Agent Server, UAS) 构成, 既可以发起呼叫请求 (发出 SIP 控制请求消息), 也可以对呼叫请求进行应答 (发出 SIP 响应消息, 作为对收到的 SIP 控制请求消息的回应)。

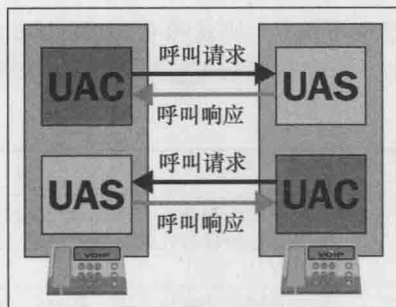


图12.13

SIP 服务器则分为以下几种类型。

- **SIP 代理服务器 (Proxy Server):** 作用是先接收来自 UA 或其他代理服务器的 SIP 控

制请求消息，再转发或代为转发至其他位置。

- **SIP 重定向服务器 (Redirect Server)**: 作用是接收来自 UA 或其他代理服务器的 SIP 控制请求消息，向它们回复包含状态码 3xx (重定向状态码) 的 SIP 响应消息，指明 SIP 控制请求消息实际的归宿。
- **SIP 注册服务器 (Registrar)**: 作用是接受来自用户的 SIP 注册请求，将用户的位置信息登记进名为“位置服务”(location service) 的数据库 (或其他数据库)。

图 12.14 所示为以上几种 SIP 服务器是如何密切合作的。

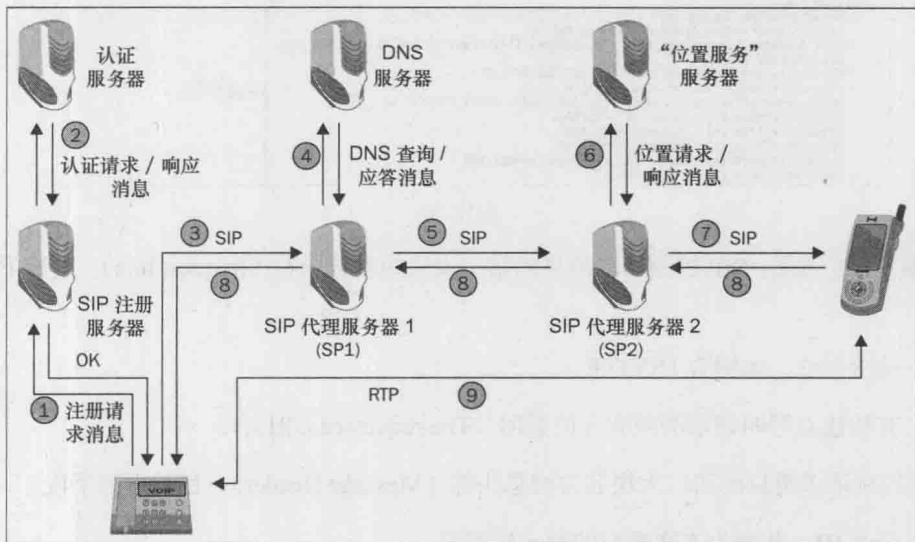


图 12.14

由图 12.14 可知，一部 IP 电话（主叫方）正在向 SIP 注册服务器注册 (①)。SIP 注册服务器向认证服务器提交认证请求 (②)，若通过认证，就会向服务提供商的 SIP 代理服务器发送 SIP 控制请求消息 (③)。服务提供商的 SIP 代理服务器与 DNS 服务器进行沟通 (④)，以解析出被叫方所在域的 IP 地址。若解析成功，则会将 SIP 控制请求消息转发至被叫方所在地 SIP 代理服务器 (⑤)。此 SIP 代理服务器无法定位被叫方，于是会找“位置服务”服务器索要被叫方的位置 (⑥)。若能获悉被叫方的位置，此 SIP 代理服务器就能将 SIP 控制请求消息转发至被叫方 (⑦)。被叫方回复 SIP 响应消息，接受主叫方的呼叫请求 (⑧)。若一切 OK，主叫方和被叫方之间将会建立起 RTP 会话 (⑨)，所使用的 UDP 端口号则取自两者之前发出的 SIP 控制请求消息/响应消息中 SDP 媒体描述部分所包含的媒体端口号字段值。

SIP 控制请求消息的格式如图 12.15 所示。

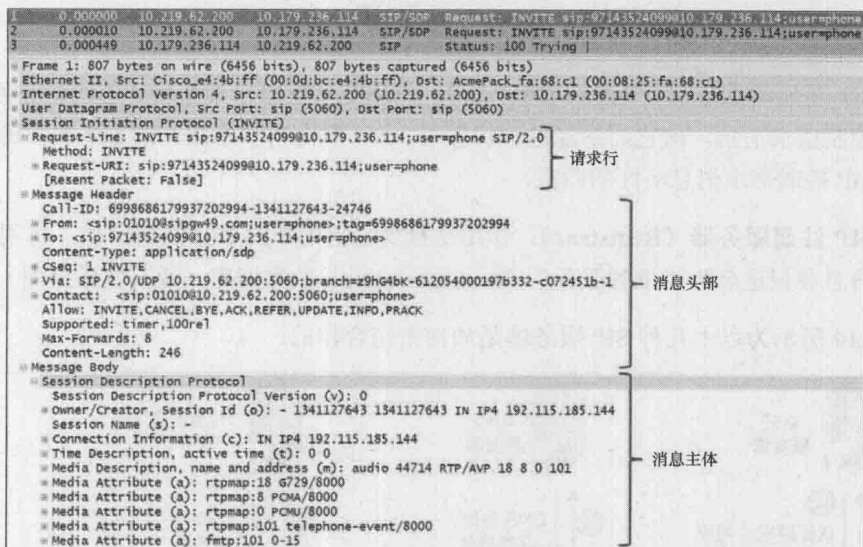


图 12.15

由图 12.15 可知，SIP 控制请求消息的第一大块叫做请求行（Request line），由以下两部分组成。

- 请求方法：本例为 **INVITE**。
- 有待建立呼叫请求的被叫方的 URI（The requested URI）。

SIP 控制请求消息的第二大块名为消息头部（Message Header），包括下列字段。

- **Call-ID**：其值为本次呼叫的唯一标识符。
- **From**：用来标识本次呼叫的主叫方。
- **To**：用来标识本次呼叫的被叫方。
- **Cseq**：其内容为一个序列号值（无符号整数）外加本消息所包含的请求方法。该序列号值根据各自的方向（主叫方和被叫方的发出控制请求消息的方向）严格单向按 1 递增，用来区分某条 SIP 控制请求消息是新的消息还是重发的消息。
- **Via**：指出了本 SIP 控制请求消息在网络中穿行的路线（由哪个 UA 发出，穿过了哪台 SIP 代理服务器），与其相对应的 SIP 响应消息也应走相同的路线。
- **Contact**：包含了一或多个 SIP URI，用来向会话接收方提供会话发起方的联络信息。
- **Allow**：用来指明所允许的请求方法。

- **Supported:** 用来指明是否支持诸如计时器之类的参数。
- **Max-Forwards:** 指明了本 SIP 控制请求消息所能途经的 SIP 实体的上限。
- **Content-Length:** 指明了消息主体的字节数。

SIP 控制请求消息的第三大块名为消息主体 (**Message Body**)，包含了主叫方所支持的媒体编码方式等信息。

12.3.4 拾遗补缺

排查 SIP 协议故障时，首先需要根据 SIP 消息头部中的 Call-ID 字段值来筛选出隶属于某次呼叫的所有 SIP 流量。为此，可按以下两种方法之一行事。

- 在 Wireshark 抓包主窗口的数据包列表区域先选择一个隶属于某个呼叫的 SIP 数据包，再到数据包结构区域中点击 **Message Header** 之前的+号，找到 Message Header 字段，点击右键，选择右键菜单中的 **Apply as Filter** 菜单项，如图 12.16 所示。

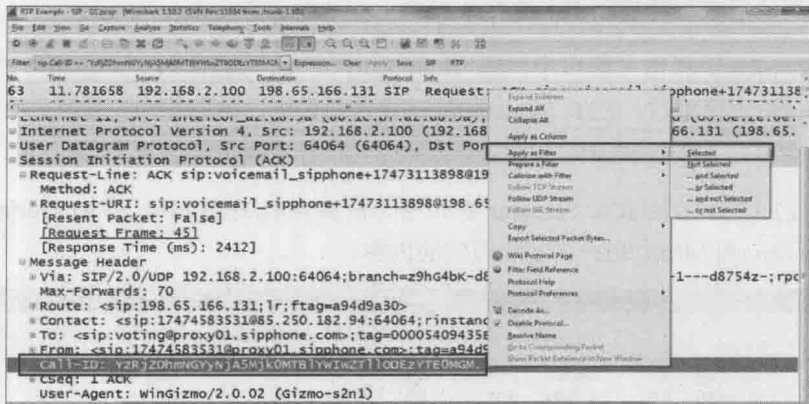


图12.16

- 借助于之前介绍过的 **Telephony** 菜单中的 **VoIP Calls** 菜单项功能。

当排除 SIP Trunk（其作用为在 IP PBX 之间传递信令）故障时，应首先判断是个别用户之间通话有问题，还是所有用户的通话都有问题。排除 VoIP 故障的最佳方法就是仔细观察抓包文件中的 SIP 协议数据包。

12.4 RTP/RTCP故障分析

上一节介绍了在 IP 上传输语音和视频时所使用的信令协议 SIP 和 SDP。本节将介绍用来传递语音和视频数据自身的 RTP 协议，以及排障之法。

对 RTP/RTCP 故障的排查，必然起始于最终用户对语音或视频通话质量的投诉。

此外，请别忘了，要是通过 Wireshark 抓包分析也看不出个所以然来的话，那就只能说需要对 IP PBX 做进一步优化，来增加语音传输容量了。

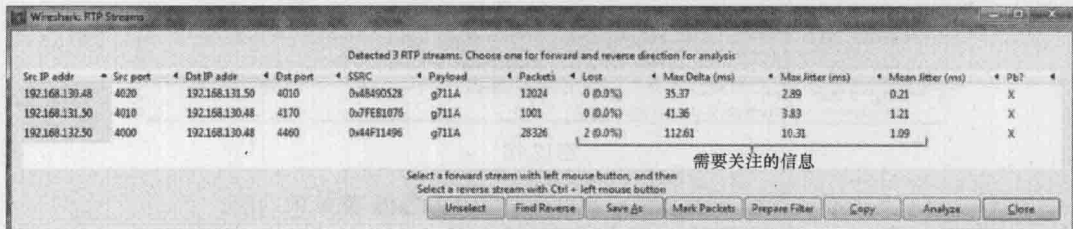
12.4.1 准备工作

当某个用户反映语音或视频通话质量不佳时，请借助于端口镜像功能，将交换机上连接该用户 UA 的端口的流量重定向至 Wireshark 主机；在处理接入相同链路的多个用户的类似问题时，亦需利用端口镜像功能设法将流淌于该链路的流量重定向至 Wireshark 主机^①。

12.4.2 操作方法

要定位跟语音或视频通话质量有关的问题，请按以下步骤行事。

1. 在交换机上开启端口镜像功能，把 Wireshark 主机连接到正确的端口之后，开始抓包。
2. 确保有语音或视频通话正在进行。
3. 在 Wireshark 抓包主窗口中点击 **Telephony** 菜单，选择 RTP 菜单项中的 **Show All Streams** 子菜单项，RTP Streams 窗口会立刻弹出，如图 12.17 所示。由图可知，该窗口会显示出 Wireshark 从受监控交换机端口抓取到的所有 RTP 流。
4. 在图 12.17 所示的 RTP Streams 窗口中，需要重点关注 Lost、Max Delta (ms)、Max Jitter (ms) 和 Mean Jitter (ms) 这四列的内容。



Src IP addr	Src port	Dst IP addr	Dst port	SSRC	Payload	Packets	Lost	Max Delta (ms)	Max Jitter (ms)	Mean Jitter (ms)	Pkts
192.168.130.48	4020	192.168.131.50	4010	0x48490528	g711A	12024	0 (0.0%)	35.37	2.89	0.21	X
192.168.131.50	4010	192.168.130.48	4170	0x7FE81076	g711A	1001	0 (0.0%)	41.36	3.83	1.21	X
192.168.132.50	4000	192.168.130.48	4460	0x44F1496	g711A	28326	2 (0.0%)	112.61	10.31	1.09	X

需要关注的信息

Select a forward stream with left mouse button, and then
Select a reverse stream with Ctrl + left mouse button

Unselect Find Reverse Save As Mark Packets Prepare Filter Copy Analyze Close

图12.17



注意

对于基于 IP 的交互式语音和视频通话而言，若网络延迟（RTT）和抖动值分别高过 300 和 50 毫秒，便预示着有可能存在通话质量问题，12.4.3 节将对此展开深入探讨。

^① 译者注：原文是 “When facing problems on a specific client, connect Wireshark to the client port with a port mirror. When facing problems with all clients connected to the same link, connect Wireshark to the link with a port mirror”。译文酌改。

若怀疑通话质量不佳是由高网络延迟所引发，请按以下思路来定位问题。

1. 用最基本的 ping 命令来了解 UA 之间的网络延迟状况。

- 若发现延迟较高，请检查是否为 UA 间通信链路所固有的正常延迟（12.4.4 节列出了每一种通信链路固有的正常延迟值）。
- 若是，无需采取任何行动。请联系 IP 电话及 PBX 厂商，让它们优化自己的设备。
- 若否，请将一台 PC 接入网络，分别 ping 主/被叫 UA，观察 ping 命令输出中的延迟情况。
- 若在 ping 某个 UA 时，发现有较高的延迟，且远高于通信链路所固有的正常延迟，则有必要在通往此 UA 的每一段链路上执行相同的 ping 命令，看看高延迟到底出在哪一段链路上。
- 同时，需借助于 Wireshark 内置的 IO Graph 工具，检查通信链路的负载状况。在大多数情况下，链路的高负载必然伴随着高延迟和高抖动。

2. 以下所列会使得网络产生高延迟的直接原因。

- **链路拥塞：**可使用 Wireshark 内置的 IO Graph 工具，来检查链路是否拥塞。
- **路由器/交换机负载过高：**可使用 SNMP 网管软件、厂商专用的工具或通过命令行来了解路由器/交换机的负载（CPU 及内存负载等）状况。
- **路由器/交换机缓存中的排队延迟：**请按设备厂商的操作手册行事。

若怀疑网络中存在高抖动现象，请按以下思路来定位问题。

1. 可沿用之前提及的排除高延迟故障的思路，来定位高抖动问题。

2. 以下所列会使得网络产生高抖动现象的直接原因。

- **链路拥塞：**可使用 Wireshark 内置的 IO Graph 工具，来检查链路是否拥塞。同时，可在链路两端执行 ping 命令，来了解链路上的抖动状况^①。
- **路由器/交换机负载过高：**可使用 SNMP 网管软件、厂商专用的工具或通过命令行来了解路由器/交换机的负载（CPU 及内存负载等）状况。

若怀疑网络中存在丢包现象，请按以下思路来定位问题。

^① 译者注：原文是“Check the line with a ping command to see if you have any problems here”。

1. 执行 ping 命令，了解 UA 之间是否存在丢包现象。
2. 若存在丢包现象，需检查 UA 间每一段链路的丢包状况。
3. 还可以在 Wireshark 抓包主窗口的数据包列表区域中先选中一个 RTP 数据包，再点击 **Telephony** 菜单，选择 **RTP** 菜单项中的 **Stream Analysis** 菜单项，RTP Stream Analysis 窗口会立刻弹出，如图 12.18 所示。

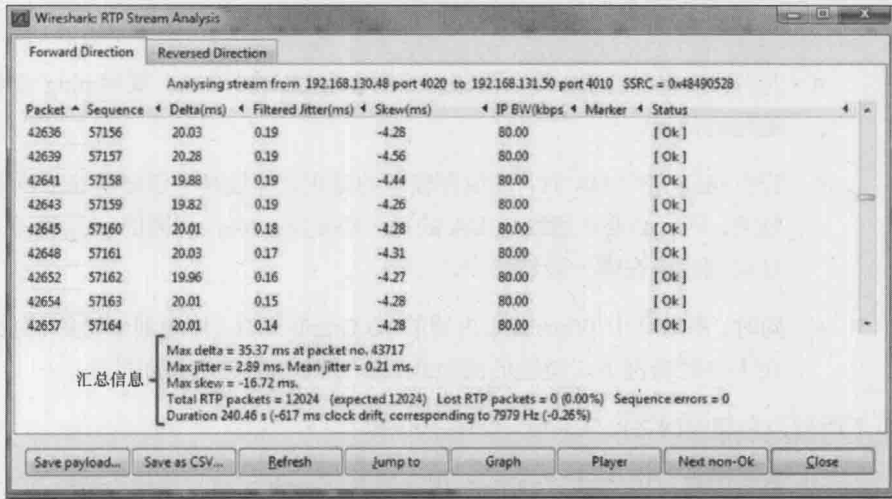


图12.18

4. 通过图 12.18 所示的 RTP Stream Analysis 窗口，可以了解到那个被选中的 RTP 数据包所隶属的 RTP 流的相关信息。以下是对该窗口中各状态栏下数值的解释。

- **Packet:** 表示 RTP 数据包在抓包文件中的编号。
- **Sequence:** 表示 RTP 序列号。
- **Delta (ms):** 表示本 RTP 数据包与前一 RTP 数据包在同一 RTP 流中的（抓取）时差，单位为毫秒。
- **Filtered Jitter (ms):** 表示 RTP 数据包的实际接收时间与 RTP 时间戳参数间的时差，单位为毫秒；其值应该越低越好，最好为 0。
- **Skew (ms):** 表示 RTP 数据包的实际到达时间与预计到达时间之差（偏早或偏晚），单位为毫秒。比方说，若 RTP 数据包的发送速率为每秒 20 个，则发送间隔时间为 50 毫秒，在这种情况下，要是第一个 RTP 数据包到达之后，过了 49 毫秒，第二个 RTP 数据包到达，则后者的 Skew 值将会被显示为 -1 毫秒 (ms)（偏早）。

- **IP BW (kbps):** 表示 (传输此) RTP 数据包所消耗的带宽, Wireshark 在计算时会把整个 RTP 数据包的 IP 包头考虑在内。
 - **Marker:** 表示 RTP 数据包的 RTP 头部中的 M (Marker) 位是置 0 还是置 1。M 位用来标记 RTP 流中的重要事件, 比如, 可在静音后第一个 RTP 数据包内将该位置 1。
 - **Status:** 表示 RTP 数据包的状态是否正常。
5. 由图 12.18 所示的 RTP Stream Analysis 窗口中的汇总信息部分, 可以了解到 (在传输这条 RTP 流时, Wireshark 检测出的) 最高抖动 (maximum Jitter) 为 2.89 毫秒。由于抖动非常之低, 因此在这条 RTP 流的传输过程中不会遇到任何麻烦。
 6. 点击 RTP Stream Analysis 窗口底部中间的 **Graph** 按钮, RTP Stream Analysis 窗口会立刻弹出, 如图 12.19 所示。

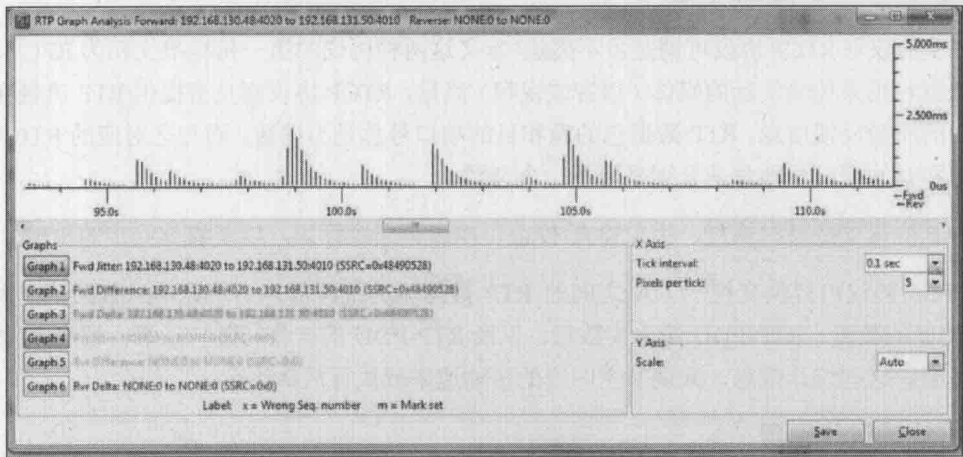


图 12.19

7. 由图 12.19 可知, 网络状况非常良好, 抖动范围在 2.500 毫秒以内 (要是只想观察抖动状况, 只需按下按钮 **Graph 1**, 同时让所有其他 Graph 按钮浮起)。
8. 经过上述 Wireshark 抓包分析, 且未发现任何问题, 可要是仍有用户投诉语音/视频质量欠佳, 请检查语音/视频设备是否存在配置问题。
9. 要是在 RTP Stream Analysis 窗口的 Status 一栏下看见了 Wrong sequences 或 Wrong timestamps 等字样 (如图 12.20 所示), 则表示在传输这条 RTP 流时, Wireshark 检测出了高延迟和高抖动。

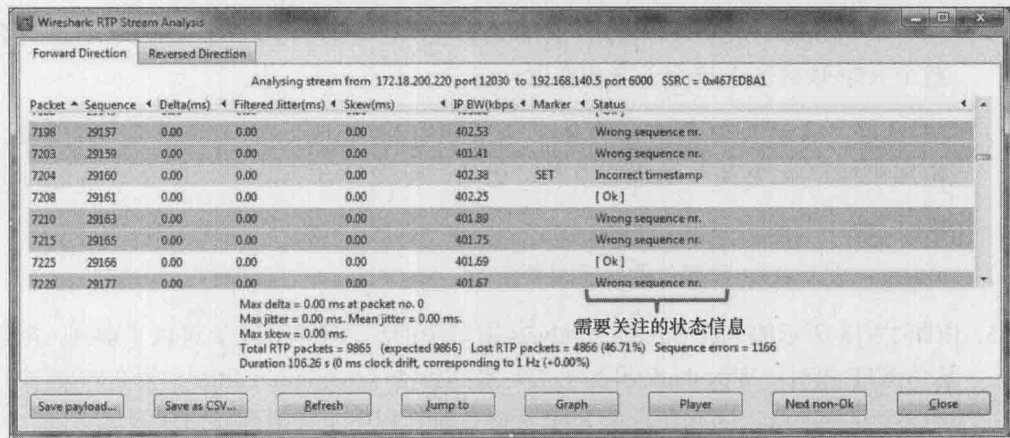


图12.20

12.4.3 幕后原理

RTP 协议与 RTCP 协议可谓是孟不离焦(定义这两种协议的第一份标准文档为 RFC 3550)。RTP 数据包用来传输实际的媒体(语音或视频)流量, RTCP 协议则负责提供 RTP 流量在传输过程中的质量反馈信息。RTP 数据包的源和目的端口号总是为偶数,而与之对应的 RTCP 数据包的源和目的端口号则总是分别采用下一个奇数。

RTP 协议支持及时恢复、丢包检查/校正、净载和来源标识,以及媒体同步等机制。

RTCP 协议可提供交换于 UA 之间的 RTP 数据流的统计信息,统计信息包括:已发送的 RTP-PDU 的数量、RTP-PDU 的丢失数量,以及 RTP-PDU 的接收间隔抖动等。语音/视频应用程序可根据这些统计信息,来调节主叫方的传输速率或执行故障诊断。

RTP 协议的运作原理

RTP 数据包在发送时要先用 UDP 头部来封装,然后再封以 IP 包头。图 12.21 所示为 RTP 数据包的格式。

下面是对 RTP 头部所含各字段的解释。

- 版本(V) 字段: 其值用来表示 RTP 协议的版本号。
- 填充(P) 位: 该位置 1 时,表示 RTP 数据包的末尾填充有一或多个不属于净载数据部分的字节。
- 扩展(X) 位: 该位置 1 时,表示 RTP 标准头部之后还紧跟着扩展头部。
- SCRC 计数字段: 其值指明了 RTP 固定头部之后 SCRC 字段的数量。

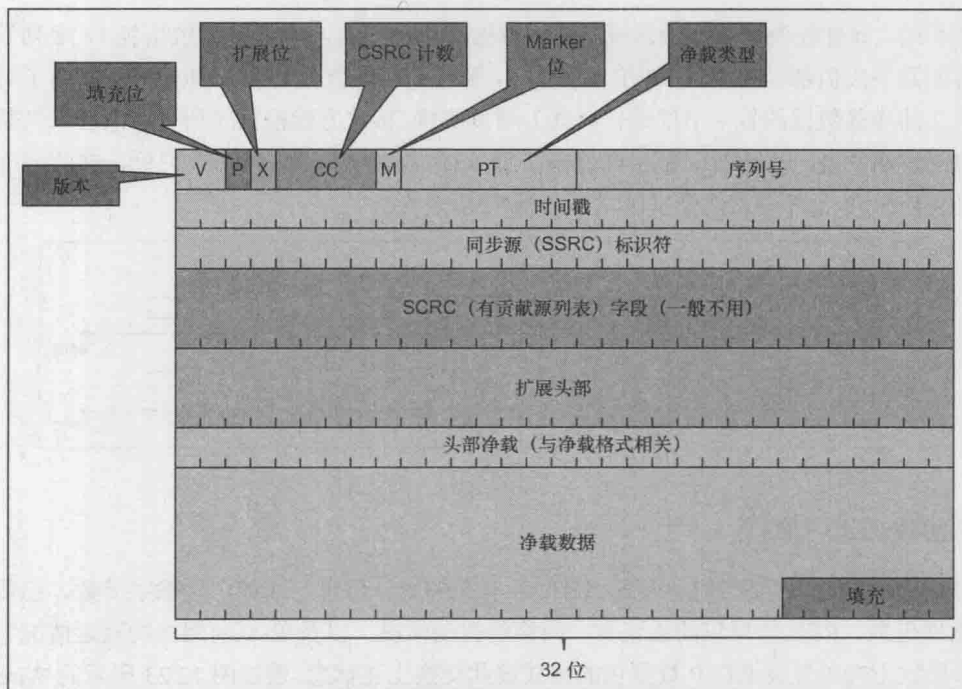


图12.21

- **Marker (M) 位**：用来标记 RTP 流中的重要事件，比如，可将该位置 1，来标识视频帧的边界。
- **净载类型字段**：用来指明 RTP 净载数据的格式，以便接收端应用程序解释。
- **序列号字段**：发送方每发出一个 RTP 数据包，便会将该字段值加 1，接收方可据此检测 RTP 数据包是否丢失。
- **时间戳字段**：其值指明了 RTP 数据包中首字节的采样时间。
- **同步源 (SSRC) 字段**：其值为随机选择，用来标识 RTP 数据流，以使同一条 RTP 会话中没有任何两个同步源具有相同的 SSRC 标识符。
- **贡献源标识符列表 (CSRC) 字段**：其值用来指明对 RTP 数据包中净载数据有贡献的贡献源（亦即 RTP 视频流的来源）。

图 12.22 所示为序列号字段和时间戳字段在 RTP 数据包所起的作用。

由图 12.22 可知，发送方每发出一个 RTP 数据包，便会将序列号字段值加 1；而时间戳字段值则为 RTP 数据包中净载数据的首字节的采样时间。时间戳字段值所表示的时间是连续的、

单调增长的，即使在没有数据输入或不发送数据时也应如此。对于 RTP 数据包 1，序列号字段值和时间戳字段值都是 1；对于 RTP 数据包 2，那两个字段值分别为 2 和 12（12 指明了对 RTP 数据包 2 的净载数据的首字节的采样时间），依此类推。接收方能根据序列号字段值来判断 RTP 数据包的发送顺序，凭借时间戳字段值来获悉 RTP 数据包离开发送方的时间。接收方应用程序正是凭借 RTP 头部中这两个字段来回放音频/视频流的。

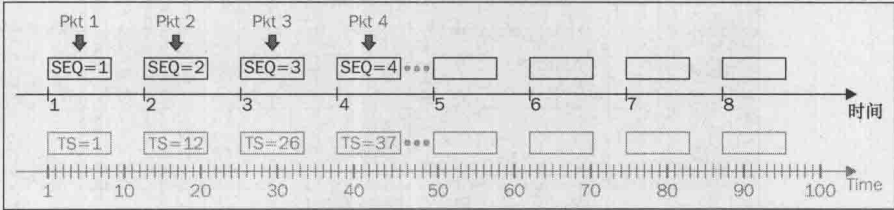


图12.22

RTCP 协议的运作原理

RTCP 协议负责收集与媒体连接（RTP）有关的统计信息，比如，传输字节数、已传输的 RTP 数据包数、RTP 数据包的丢包数、网络的抖动情况，以及单/双向网络的延迟情况等。发送方和接收方会以互发 RTCP 数据包的方式彼此交换上述信息。通过图 12.23 所示的 Wireshark 截屏，可以了解到发送发向接收方通报的已传输的数据包的个数/字节数，以及时间戳等信息。

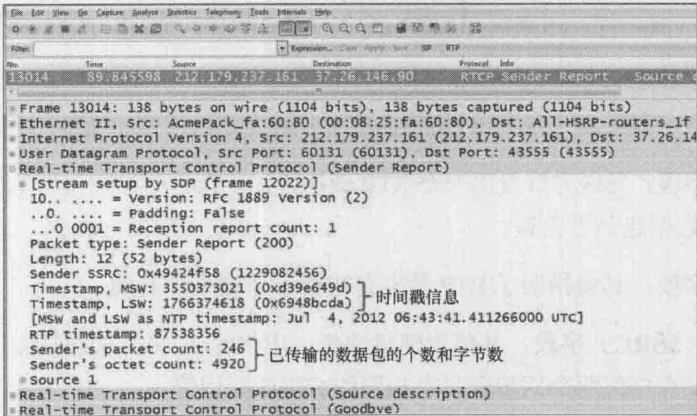


图12.23

12.4.4 拾遗补缺

以下所列为传输 RTP 数据流时，哪些地方可能会引入延迟。

- 编/解码延迟：由数/模转换所导致的延迟。
- 处理延迟（“打包”延迟）：（网络设备）构建 RTP 数据包并将语音/视频信息插入其中

所产生的延迟。

- **串行延迟**：RTP 数据包在通信链路上发送所产生的延迟，具体时长取决于 RTP 数据包的大小及通信链路的带宽（在不发生拥塞的情况下，每一种通信链路的串行延迟时长都相对固定）。

以下所列为各种通信链路（尚未过载时）所固有的串行延迟，即在通信链路一端的主机上 ping 对端主机时，ping 命令的输出显示出的 time 值。

- **LAN 链路**：延迟低于 1 毫秒。
- **WAN 链路**：同城链路的延迟为 1~2 毫秒，长途链路的延迟为 15~20 毫秒。若为老掉牙的帧中继或 X.25 链路，延迟将会高达 100 毫秒或以上。
- **家用宽带链路**（比如，xDSL 或 CaTV 链路）：延迟为 15~20 毫秒。
- **国际长途链路**：延迟为 100~200 毫秒。
- **移动蜂窝链路**：对于老式的 2.5G 网络（GPRS 或 CDMA 1X），延迟为 300~600 毫秒；对于 3G 网络（UMTS 或 EVDO），延迟为 120~150 毫秒；对于 HSDPA、HSUPA 以及 HSPA+ 网络，延迟为 60~100 毫秒；对于 LTE 网络，延迟为 20~50 毫秒。
- **卫星通信网络**：延迟为 500~600 毫秒。



注意

数据包在通信链路上（发送）所耗时长等于传播延迟外加处理延迟。传播延迟是指电（或光）信号穿越整条通信链路所耗时长；处理延迟则为通信链路沿途所有运营商设备交换或路由数据包所耗时长。自本世纪初以来，有很多运营商都在对外提供业务的网络中部署了高速路由器和交换机，同时采用新的技术（比如，MPLS 和城域以太网），来实现数据包转发的低延迟，但也有不少这样的网络仍在沿用老掉牙的 ATM 或帧中继技术，数据包在其中转发时会获得较高的延迟。

12.5 视频及视频监控应用排障场景

在过往的 10~12 年间，由安保及视频监控系统所生成的流量在通信网络（所承载的总流量）中所占比重越来越高^①。如本章将要讨论的那样，在那些（需要承载安保及视频监控系统所生成的流量的）网络中，通常会因带宽不足而导致视频信号卡顿等更为复杂的问题。本章会介绍在部署了上述系统的网络中可能会碰到的各种各样的问题，以及解决问题的方法。

^① 译者注：原文是 “In the last 10 to 12 years, security and surveillance systems have taken on a larger and more important role in communications networks”。作者词不达意，译文酌改。

12.5.1 准备工作

在部署了安保及视频监控系统的网络中,网络技术人员经常需要出马解决用户体验问题^①。在图 12.24 所示的网络中,解决上述问题时,可借助于端口镜像功能,用 Wireshark 抓取:监控探头生成的流量(①)、通往远程分支机构的通信链路的流量(②),或监控探头服务器生成的流量(③)。当然,也可以在(未部署监控探头的)中心站点抓取核心链路的流量(④),再配置显示过滤器,筛选出与远程分支机构网络交互的流量。

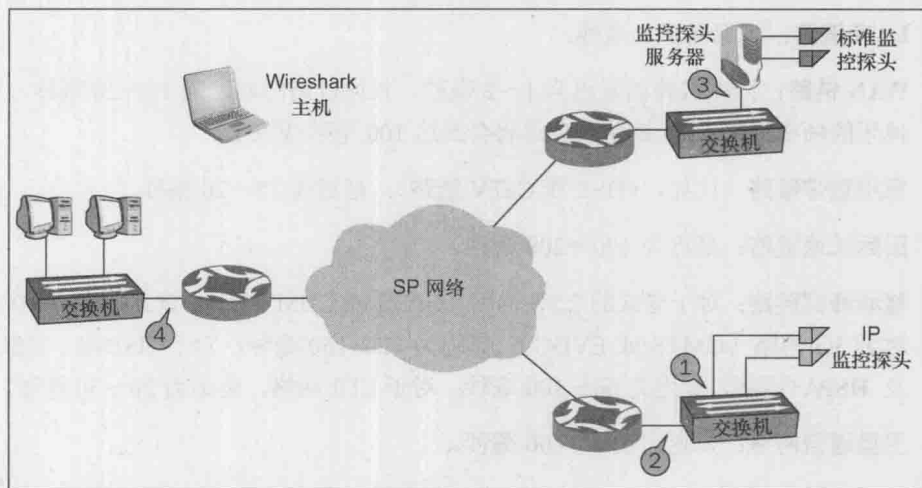


图12.24

12.5.2 操作方法

要想发现上图所示网络中的视频监控流量的传输问题,请按以下步骤行事。

1. 首先,应借助于端口镜像技术,用 Wireshark 抓取并分析 Viewer (视频观看器) 与跟其直连的 IP 监控探头之间的流量(即在 Viewer 跟 IP 监控探头共同接入的 LAN 链路上抓包)。如此行事,就能获悉视频监控图像以每一种分辨率示人时,所需要的带宽。



注意

观看视频监控时,如对图像的分辨率要求不高(比如,只要求最基本的黑白图像),所需带宽为 128 kbit/s;若要观看分辨率较高的黑白或彩色图像,平均所需带宽为 0.5~1Mbit/s;对于高清图像,所耗带宽将会高达 6~8Mbit/s。

^① 译者注:原文是“Usually, you will be called to solve problems that users experience when watching security cameras”。文字不通,译文酌改。

2. 要检查带宽是否足够。

- 当出现视频卡顿现象时, 应使用 Wireshark 内置的 IO Graph 工具, 来监控链路带宽占用率。请确保有足够的带宽来传输视频监控流量, 同时需保证链路并没有因传输其他类型的流量而导致拥塞。
- 要是只想弄清由监控探头或监控服务器生成的流量占用了多少带宽, 请在 IO Graphs 窗口中的 Filter 输入栏内, 输入以监控探头或监控服务器的 IP 地址作为参数的显示过滤器。



注意

视频流在传输时既可以基于 UDP, 也可以基于 TCP。UDP 通常用于交互式视频应用, 而 TCP 则多用于远程视频监控应用。

3. 要检查网络中是否有丢包或明显的高延迟/高抖动现象。

- 要想确定网络中是否存在丢包现象, 请登录进网络设备执行相关 (show) 命令或借助于基于 SNMP 的网管软件, 来了解详情^①。
 - 要想确定网络中是否存在高延迟/高抖动现象, 可执行 ping 命令或借助于某些图形界面的网络工具 (这些工具大多都是自由软件, 比如, Colasoft) 来了解具体情况。
4. 在观看远程监控探头传回的视频监控图像时, 若出现短暂的卡顿现象, 请使用之前介绍过的 TCP Graph 工具 (在 Wireshark 抓包主窗口的数据包列表区域内, 先选中一个隶属于该视频流的 TCP 数据包, 再点击 **Statistics** 菜单, 选择 **TCP Stream Graph** 菜单项下的 **Time Sequence (Stevens)** 子菜单项), 如图 12.25 所示。当视频监控流量能正常传送时, 相应 TCP Graph 窗口中示出的所有 I 帧和 P 帧的接收时间间隔应保持不变。
 5. 在图 12.25 所示的客户站点中, 监控探头生成的视频流量需要先传送给一台中心服务器, 这台中心服务器会将视频流量继续分发给各台监视器。由图 12.25 可知, 由于服务器推迟发送了某些 P 帧, 因此在观看视频监控图形时可以感觉到短暂的卡顿现象。经过检查, 最终发现是那台服务器上安装的软件有问题。
 6. 在尝试登录视频监控服务器 (上的视频管理系统) 时, 只要有某些端口被屏蔽, 便有可能登录不进去; 也有可能发生可以登录, 但无法实施操作的情况。要想验证所有必要的 TCP (UDP) 端口是否被屏蔽, 请执行以下检查。

^① 译者注: 原文是 “For packet losses, log in to the communications equipment or use SNMP”。译者实在无法帮作者打圆场了。

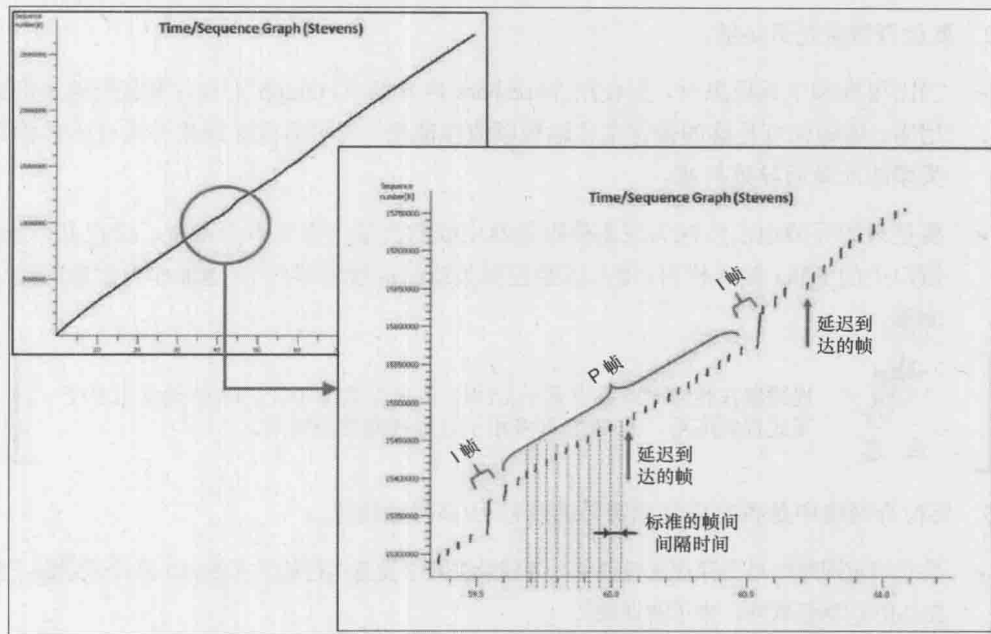


图12.25

- 若本机与视频监控服务器之间部署有防火墙，请检查它的配置，看看它是否屏蔽了必要的端口。
 - 使用 Wireshark 抓包分析，若连相关 TCP 连接（由本机发起的通往视频监控服务器的 TCP 连接）的完整的三次握手过程都观察不到，则该 TCP 连接一定是遭到了某些东西的“阻拦”。
7. 由图 12.26 所示的 Wireshark 截屏可知，在内部办公网主机 **10.0.0.3** 和公网视频监控服务器 **82.82.182.182**（在实战中，请千万不要把视频监控服务器暴露在公网之上，此处只是为了举例）之间建立起了一条 HTTP 连接（①）。此外，还可以看出，内网主机 10.0.0.3 向视频监控服务器 **82.82.182.182** 发出了目的端口号为 **6036**，SYN 标记位置 1 的 TCP 报文段（数据包编号为 **2614**），以期建立另外一条 TCP 连接。但是，后者回复了 RST 标记位置 1 的 TCP 报文段（数据包编号为 **2615**），拒绝了前者的美意（②）。这一行为又连续重复了两次（③和④）。也就是说，两台主机之间虽能建立起目的端口号为 A 的 TCP 连接，但端口号为 B 的 TCP 连接则未必能够成功建立。
8. 实际情况是，操纵内网主机 10.0.0.3 的管理员在登录视频监控服务器 82.82.182.182（的管理系统）时，能够看到 Web 管理界面（这说明 HTTP 连接可以建立），但在 Web 页面中输入用户名/密码，企图登录管理系统时，还需另行发起一条目的端口号

为 6036 的 TCP 连接。由于 TCP 连接被防火墙阻断, 在那台内网主机上, 只能通过浏览器打开视频监控管理系统的 Web 管理界面, 但却无法登录。由于无法登录, 因此也就看不到监控探头传来的视频了。

No.	Time	Source	Destination	Protocol	Info
2608	36.075818	10.0.0.3	82.82.182.182	TCP	62418 > http [ACK] Seq=915 Ack=41863 Win=66792 Len=0 [TCP segment of a reassembled PDU]
2609	36.120139	82.82.182.182	10.0.0.3	TCP	62438 > http [ACK] Seq=915 Ack=45687 Win=66792 Len=0 [TCP segment of a reassembled PDU]
2610	36.120284	10.0.0.3	82.82.182.182	TCP	62438 > http [ACK] Seq=915 Ack=47511 Win=66792 Len=0 [TCP segment of a reassembled PDU]
2611	36.193607	82.82.182.182	10.0.0.3	TCP	62438 > http [ACK] Seq=915 Ack=47511 Win=66792 Len=0 [TCP segment of a reassembled PDU]
2612	36.214472	82.82.182.182	10.0.0.3	TCP	62438 > http [ACK] Seq=915 Ack=47511 Win=66792 Len=0 [TCP segment of a reassembled PDU]
2613	36.222803	82.82.182.182	10.0.0.3	HTTP/DL	unknown (0x0f)
2614	36.249612	10.0.0.3	82.82.182.182	TCP	62442 > 8036 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1
2615	36.251104	82.82.182.182	10.0.0.3	TCP	8036 > 62442 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2620	36.421135	10.0.0.3	82.82.182.182	TCP	62438 > http [ACK] Seq=915 Ack=45444 Win=65856 Len=0
2621	36.789037	10.0.0.3	82.82.182.182	TCP	62442 > 8036 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1
2622	36.830139	82.82.182.182	10.0.0.3	TCP	8036 > 62442 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
2623	37.008763	fe80::c067:2c23:ff02::c	ssdp	SSDP	M-SEARCH * HTTP/1.1
2626	37.325129	10.0.0.3	82.82.182.182	TCP	62442 > 6036 [SYN] Seq=0 Win=0 Len=0 MSS=1460 SACK_PERM=1
2627	37.363547	82.82.182.182	10.0.0.3	TCP	6036 > 62442 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

图12.26

12.5.3 幕后原理

视频流是由 I 帧 (Intra-coded frames, 内部编码帧)、P 帧 (Predicted frames, 可预测帧) 和 B 帧 (Bi-predictive frames, 双向可预测帧) 构成。I 帧是指包含全画面的帧; P 帧也叫差别帧, 不含完整的画面数据, 只含与前一帧的画面有差别的数据。P 帧记录的是本帧与前、后帧的差别。

采用 TCP 传输视频流时, 一个 I 帧、P 帧或 B 帧会被分割为由若干个 TCP 报文段来承载。因此, 若网络中存在 TCP 性能问题 (比如, TCP 重传等问题), 就会直接影响到视频流的传输。

12.5.4 拾遗补缺

视频监控流量的传输质量取决于多种因素, 这些因素涉及: 所采用的编解码器、每秒传输的帧的数量、帧间间隔时间, 以及监控探头或视频监控服务器的各种配置参数。要想看到优质的监控画面, 需确保所有参数配置无误。

12.6 IPTV应用排障场景

在过去几年里, IPTV 应用变得越来越受欢迎, 通过互联网提供服务的电视台也日渐增多。与此同时, 各类基于互联网的电子化教育 (E-learning) 应用也如星火燎原之势蔓延开来。

IPTV 应用基本上都是基于 TCP, 与该应用有关的故障大多数都是 TCP 故障 (比如, TCP 重传等), 本节会举几个这方面的例子。

12.6.1 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, 将连接了受监控设备的端口的流量重定向至 Wireshark 主机。

12.6.2 操作方法

启动 Wireshark 软件，开始抓包，同时按以下步骤行事。

1. 在 Wireshark 抓包主窗口中选择 **Statistics** 菜单中的 **IO Graph** 菜单项，启动 **IO Graph** 工具，查看 IPTV 流量的缓冲情况，如图 12.27 所示。

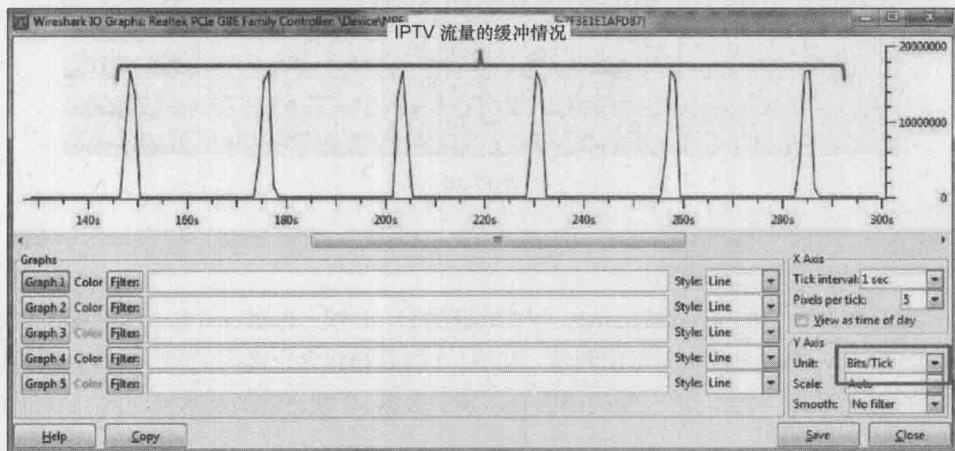


图12.27

2. 若 (Internet) 链路带宽不足，I/O Graph 窗口中图形显示区域的波浪会在顶端被截断。此时，需调整 IPTV 播放器，降低 IPTV 的播放画质（一般的 IPTV 播放器都会在带宽不足时做自动调整）。
3. 检查 Wireshark 是否抓到了众多身背 TCP retransmissions、duplicate ACK 以及 TCP window Full 等字样的数据包，若是，请先核实是否是 TCP 故障。

12.6.3 幕后原理

从网络的角度来看，IPTV 不过是一种基于 TCP 的简单应用。

12.6.4 拾遗补缺

IPTV 故障往往都是 TCP 故障，TCP 故障的排障思路详见本书第 9 章。

12.7 视频会议应用排障场景

视频会议应用和 VoIP 应用在运行时所涉及的网络协议完全相同，但在流量传输的方法上却有所区别：拨打 IP 电话时，会在每个方向上各生成一条语音流；使用视频会议系统时，不

但在每个方向上要各生成一条语音流,还会再分别多生成一条视频流。若在参与视频会议的终端上抓包,将会抓到 4 条数据流:两条是此终端发给对端设备的语音流和视频流,另外两条则是对端设备发来的语音流和视频流。

还有一点值得注意的是,有些视频会议应用仍旧基于 H.323 协议族。因此,在排除信令协议故障时,只懂 SIP 协议可就玩不转了,还需掌握 H.225 和 H.245 协议。不过,绝大多数视频会议应用都基于 IETF 协议族,所以本章将聚焦于 IETF 协议族。

12.7.1 准备工作

在交换机上开启端口镜像功能,把连接了出问题的视频会议终端设备的端口的流量重定向给 Wireshark 主机。

12.7.2 操作方法

在 Wireshark 抓包主窗口中,选择 **Telephony** 菜单中 **RTP** 菜单项下的 **Show All Stream** 子菜单项时,会弹出 RTP Streams 窗口。对于 Wireshark 所感知到的每一次视频会议通话,在 RTP Streams 窗口中都会显示出 4 条与之对应的数据流,如图 12.28 所示。

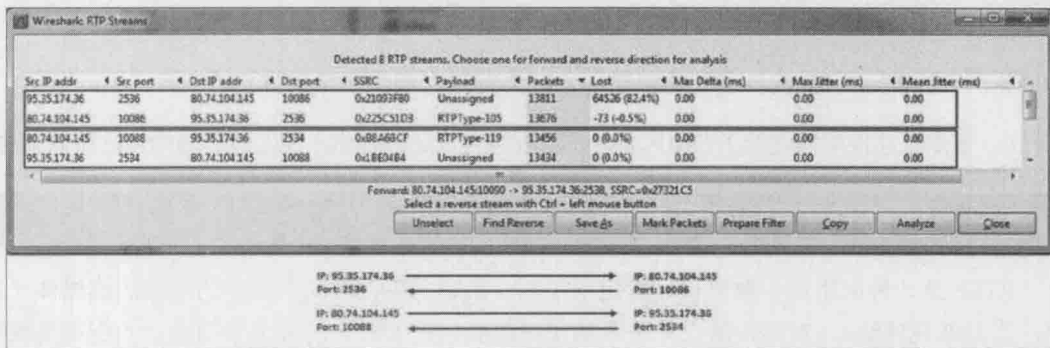


图 12.28

1. 由图 12.28 可知,在每个方向上都各诞生了两条数据流。
2. 图中所示第一条数据流存在很明显的性能问题。
3. 欲知更多详情,请选中该数据流,再点击 **Analyze** 按钮。
4. RTP Stream Analysis 窗口会立刻弹出,如图 12.29 所示。
5. 由图 12.29 可知,在传输第一条数据流时,存在网络(传输速率)不稳定等若干问题, Wireshark 报告了以下错误。

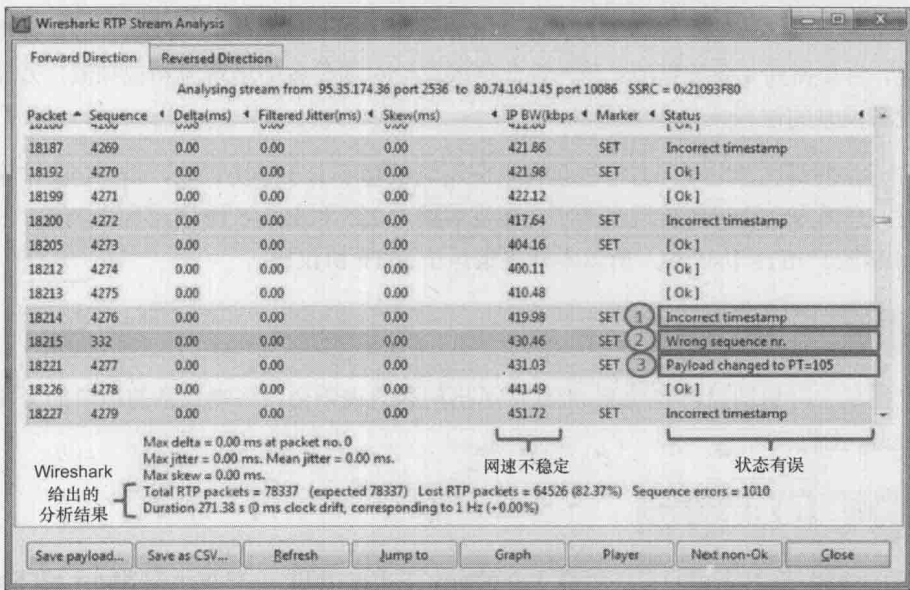


图12.29

- “Incorrect timestamp (时间戳有误)” (①) 和 “Wrong sequence number (序列号错误)” (②)，报错的原因要归咎于通信线路存在高抖动现象。
- “Payload change” (③)，报错的原因为发送方调优了编码方式。

6. 催生上述性能问题的主要原因是，那次视频会议通话是由不稳定的移动蜂窝链路承载的。

12.8 排除RTSP协议故障

RTSP 是一种应用层控制协议，可用来控制一条或多条在时间上同步的连续的流媒体（比如，音频或视频流）。对远程的媒体服务器实施控制，是 RSTP 的最主要用途。当点击流媒体播放软件中的“播放”或“暂停”按钮时，就会用到 RSTP 协议。此外，还可以使用 RSTP 协议来邀请一新的媒体服务器参加正在举行的视频会议。RTSP 只是一种控制协议，流媒体数据本身要由 RTP 协议来承载（也就是说，RTP 协议为数据平面协议），而 RTCP 协议则负责监控流媒体的传输。

在定义 RTSP 协议的标准 RFC 2326 里，并未对承载 RSTP 流量的传输层协议的类型（UDP 或 TCP）有任何硬性规定，但绝大多数 RSTP 实现都采用 TCP 协议作为传输层协议。RTSP 协议经常会在观看 IPTV 时用到。本节会讲解如何使用 Wireshark 抓取并分析 RSTP 协议消息，同时会介绍如何排除 RSTP 相关故障。

12.8.1 准备工作

当流媒体在播放过程中出现问题（比如，媒体播放器的控制功能失效，或运行媒体播放器的客户端主机与媒体服务器之间的连通性出现问题）时，应使用 Wireshark 来抓取并分析 RTSP 协议消息^①。RTSP 协议的运作方式如图 12.30 所示。

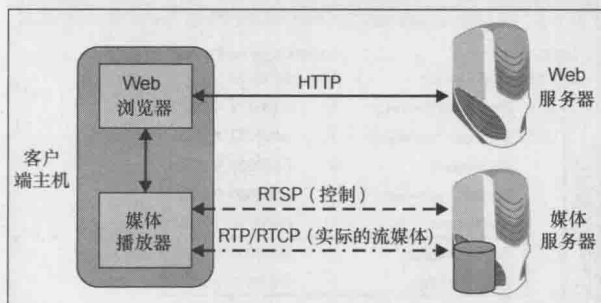


图 12.30

当一条音频或视频流在播放过程中出现问题时，问题既有可能出在 RTP/RTCP 协议上面，也有可能出在 HTTP 协议身上，甚至都有可能是 TCP 协议故障。本章前文以及第 9 章和第 10 章已经介绍过了如何借助于 Wireshark 来排除上述协议故障，本节将讨论如何排除 RTSP 协议（图 12.30 中中间的那道虚线）故障。RTSP 协议出现故障时，同样会影响音频或视频流的正常播放。

图 12.30 所示 Web 服务器和媒体服务器既可以部署于相同或不同的物理机器上，也可以部署于不同的虚拟机上。无论怎么部署，RTSP 协议的运作方式都如图 12.30 所示。

12.8.2 操作方法

把 Wireshark 主机接入网络，在交换机上开启端口镜像功能，将连接了出问题的客户端主机的交换机端口的流量重定向至 Wireshark 主机。若为大面积故障，则需设法将连接了 Web 服务器和媒体服务器的交换机端口的流量重定向至 Wireshark 主机。

1. 要让 Wireshark 只显示 RTSP 流量，需筛选出所有目的端口号为 554 的 TCP 流量，显示过滤器的写法为：`tcp.port==554`。



注意

显示过滤语句 `tcp.port==554` 和 `rtsp` 之间的差别在于，前者会让 Wireshark 显示出所有目的端口号为 554 的 TCP 数据包，而后者会让 Wireshark 显示出所有包含 RTSP 头部的数据包。

^① 译者注：原文是“RTSP monitoring should be used in cases in which you experience transmission disturbances; for example, problems with the media player control or cases with connectivity problems to a server”。译文酌改。

2. 要观看 Wireshark 抓到的有关 RTSP 数据包的汇总信息，请在 Wireshark 抓包主窗口中，点击 Telephony 菜单下 RTSP 菜单项里的 **Packet Counter** 子菜单项。等 RTSP/Packet Counter Stats 窗口弹出之后，点击 **Create Stat** 按钮，RTSP/Packet Counter with filter: ip 窗口会立刻弹出，如图 12.31 所示。包含 RTSP 错误状态码的 RTSP 响应数据包身负的字样不是 4xx 就是 5xx。

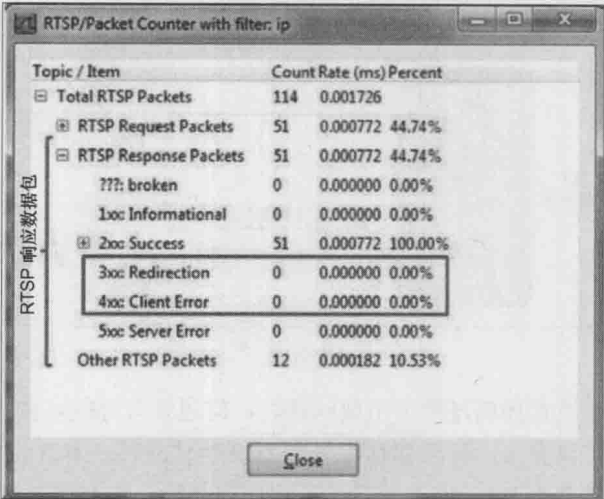


图12.31

3. 要让 Wireshark 只显示身负 RTSP 错误状态码的 RTSP 数据包，请在抓包主窗口的 Filter 输入栏里输入显示过滤表达式 `rtsp.status >= 400`，再点击 **Apply** 按钮。

12.8.3 幕后原理

跟受控于 SIP 的 VoIP 流量一样，受 RTSP 控制的音频或视频流量同样要用 RTP 协议来承载。RTP 协议在运作方式上似乎是在刻意模仿 HTTP 协议，所使用的语法也几乎相同。

下表所列常见的 RTSP 方法（命令），其中 C 表示客户端，S 表示服务器。

方 法	方 向	作 用
OPTIONS	C to S 或 S to C	确定服务器/客户端所具备的可选功能
DESCRIBE	C to S	获取对流媒体的描述
ANNOUNCE	C to S 或 S to C	宣告并描述一条新会话
SETUP	C to S	创建一条流媒体会话
PLAY	C to S	开始交付流媒体
RECORD	C to S	开始记录流媒体
PAUSE	C to S	暂停交付流媒体
REDIRECT	S to C	重定向至另一台服务器
TEARDOWN	C to S	停止交付流媒体，释放相关资源

下表所列为 RTSP 响应状态码的分类情况。

状态码编号	类 型	含 义
1xx	信息	表示服务器收到 RTSP 请求消息，正在进行处理
2xx	成功	表示服务器已接受、识别并认可了客户端通过 RTSP 请求消息所传达的操作请求
3xx	重定向	服务器告知客户端，要想完成（通过 RTSP 请求消息所传达的）操作请求，还需采取进一步的行动
4xx	客户端错误	表示客户端发出的 RTSP 消息中包含了错误的语法，或服务器未执行客户端通过 RTSP 消息传达的操作请求
5xx	服务器错误	表示服务器未能执行客户端通过 RTSP 消息传达的明显有效的操作请求

12.8.4 拾遗补缺

图 12.32 所示为 Wireshark 抓到的典型的 RTSP 数据流。

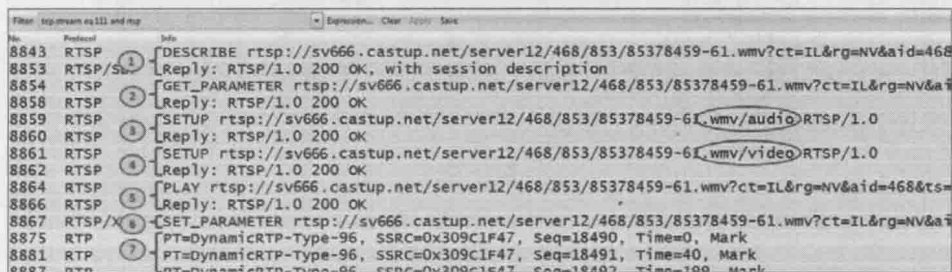


图12.32

现在，将围绕图 12.32 来说明客户端与服务器之间是如何以交换 RTSP 数据包的方式，对所传递的流媒体进行控制的。

1. 客户端向服务器发出身背 **DESCRIBE** 字样的 RTSP 请求消息，意在从服务器获取由（包含在请求消息中的）URL 指明的对表示（presentation）或对媒体对象的描述信息。服务器向客户端发出身负 **200 OK** 字样的 RTSP 响应消息，作为对之前 RTSP 请求消息的回应^①。
2. 客户端向服务器发出身背 **SET_PARAMETER** 字样的 RTSP 请求消息，目的是向服务器获取由（包含在请求消息中的）URI 指明的表示（presentation）或流的参数信息^②。

① 译者注：原文是“A DESCRIBE request is sent to the server, asking to retrieve the description of a presentation or media object identified by the request URL from that server, and the server replies with 200 OK”。译文酌改。想问问作者，“presentation”是 RTSP 标准中定义的术语，你知不知道？你知道的话，干吗不解释？

② 译者注：原文是“A SET_PARAMETER request retrieves the parameter value of a presentation or stream specified in the URI”。作者只能看见表象，知其然，不知所以然。

- 3. 客户端向服务器发出身背 **SETUP** 字样的 RTSP 请求消息，意在打开音频流；服务器回之以身背 **220 OK** 字样的 RTSP 响应消息。
- 4. 客户端向服务器发出身背 **SETUP** 字样的 RTSP 请求消息，目的是打开视频流；服务器回之以身背 **220 OK** 字样的 RTSP 响应消息。
- 5. 客户端向服务器发出身背 **PLAY** 字样的 RTSP 请求消息，意在播放流媒体。
- 6. 客户端向服务器发出身背 **SET_PARAMETER** 字样的 RTSP 请求消息，目的是设置由（包含在请求消息中的）URI 指明的表示（presentation）或流的参数。
- 7. 实际的流媒体数据开始通过 RTP 协议传送。

图 12.33 展示了如何借助于 RTSP 协议，来停止流媒体的传送。

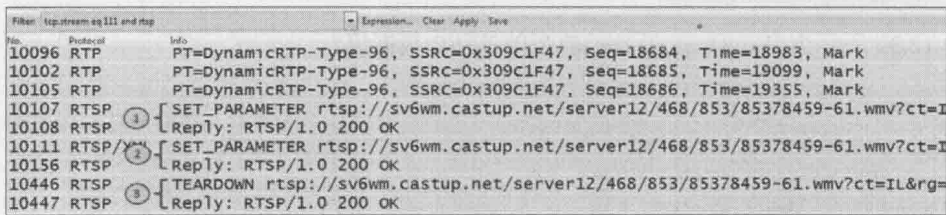


图12.33

以下所列为关闭流媒体的过程。

- 1. 客户端向服务器发出身背 **SET_PARAMETER** 字样的 RTSP 请求消息，目的是设置由（包含在请求消息中的）URI 指明的表示（presentation）或流的参数。
- 2. 客户端向服务器发出第二条身背 **SET_PARAMETER** 字样的 RTSP 请求消息。
- 3. 客户端向服务器发出身背 **TEARDOWN** 字样（包含 **TEARDOWN** 方法）的 RTSP 请求消息，目的是停止流媒体的传送。

第13章

排除由低带宽或高延迟所引发的故障

本章涵盖以下主题：

- ▶ 测量通信链路的总带宽；
- ▶ 测量每个用户及每种应用所占用的通信链路的带宽；
- ▶ 借助 Wireshark，来获知链路上的延迟及抖动状况；
- ▶ 发现因高延迟/高抖动所引发的应用程序故障。

13.1 简介

读者应该知道，通信链路质量的高低取决于 4 个重要参数：带宽、延迟、抖动以及丢包率。有些应用程序在运行时需要高带宽，而另外一些则对延迟和抖动非常敏感。至于丢包，将会影响到所有应用程序的运行，只是影响程度有所不同而已。

本章将探讨如何测量通信链路的这 4 项参数，同时会介绍如何定位并解决由低带宽、高延迟、高抖动或高丢包率所导致的网络故障^①。

13.2 测量通信链路的总带宽

本节会介绍如何测量通信链路的总带宽。测量链路带宽之前，应会同服务提供商仔细核对

^① 译者注：原文是 “In this chapter we will learn how to measure these parameters, how to check for network problems caused by it, and how to solve them when possible”。

通信链路的资料, 确定通信链路是属于对称链路还是非对称链路, 若为非对称链路, 得分别弄清其上、下行速率(带宽)。

13.2.1 测量准备

在测量之前, 需做好如下准备工作。

- 如要测量两个站点网络间通信链路的带宽, 请提前在本地和远程站点分别部署好测试用机(台式机、笔记本或服务器均可)^①。
- 如要测量互联网链路的带宽, 请事先在 ISP (Internet 服务提供商) 网络一侧安排一台测试用服务器。

13.2.2 测量方法

要测量一条通信链路的带宽, 请按以下步骤行事。

1. 向 SP 或 ISP 询问通信链路或 Internet 链路的带宽。
2. 在通信链路的对端安置好测试用机。



注意

用笔记本电脑或 PC 作为测试用机时, 请确保其足够强劲, 能经得起高网络流量的冲击。一台安装了 Win7 OS 的 PC 能通过每条 TCP 连接生成约 200Mbit/s 的流量, 但若开启的连接数过多, 则有可能会遭遇磁盘性能等方面的限制。因此, 建议先在几乎没有带宽限制的 LAN 内传输文件, 弄清测试用机的性能, 然后才能用其去测量 SP 或 ISP 提供的通信线路。若以 FTP 上传或下载的方式来测量通信链路的带宽, 请安装最高效的 FTP 服务器软件(比如, FileZilla)。当然, 最好还是使用专门的线路带宽测量设备来进行测量, 可从 VeEX、Fluke Networks、IXIA 等厂商订购这些设备。

- 在图 13.1 所示的网络中, 要想测量两个站点网络间通信链路的带宽, 可在图中①和②之间或①和③之间传输文件。所传输的文件不能太小, 最起码也应该在链路“满负荷”情况下能支撑 1 分钟以上。比方说, 若通信链路的带宽为 10Mbit/s, 所传输的文件大小至少应为 $10\text{Mbit} \times 60\text{秒} / 8 = 75\text{MB}$ ^②。

① 译者注: 原文是“When you measure a communication line between two offices: in this case connect your laptop (or any PC on the network) to the LAN, and verify whether you have a server or another PC on the other side of the line”。如此简单的事情, 给整那么复杂, 服了。

② 译者注: 以上两句的原文是“A file big enough should load the line for a significant amount of time, that is, a minute or more. For example, if you want to test a 10 Mbps (Megabits per second) line, use a file of at least $10/8 = 1.25\text{ MB}$ (Megabytes)”。译文酌改。

- 提供互联网链路的 ISP 一般都会部署专门的测速服务器，测试互连网链路的带宽（速率）时，只需在本方网络内的测试用机和 ISP 的测速服务器之间（图中的①和⑤之间）传输文件即可。



注意

为测量链路带宽，而执行文件传输操作时，传输机制应尽量采用 TCP 或 UDP。若传输机制采用 TCP，则可以在测量带宽时，顺带检查是否存在 TCP 性能问题。为此，请使用像 **Iperf** 这样能生成 TCP 或 UDP 流量的测试软件。

在图 13.1 所示网络中，那两个站点网络通过 SP 提供的 WAN 链路互连。在图中左侧的站点网络内，部署有一台防火墙，该防火墙通过一条 ISP 提供的互联网链路让整个站点网络接入 Internet。

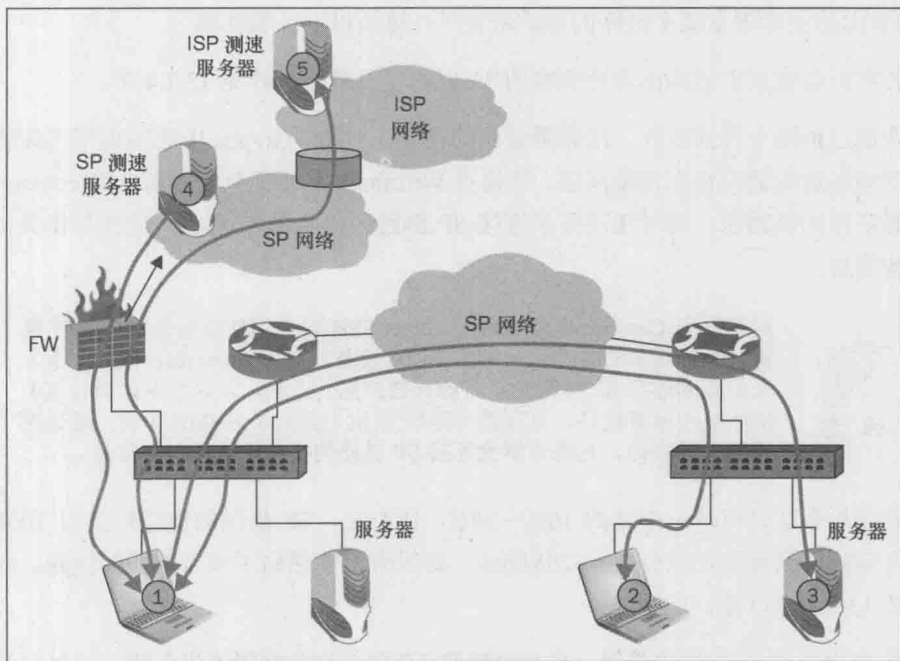


图13.1

要测量图 13.1 所示各条通信链路的带宽，请按以下步骤行事。

1. 使用 Wireshark 自带的 IO Graphs 工具。



注意

测量高带宽链路时，Wireshark 可能会力有不逮。作者本人更喜欢用其他工具（比如 **Omnipeek**）来测量带宽为 200Mbit/s 以上的链路。

2. 安装并配置像 Iperf (<http://sourceforge.net/projects/iperf/>) 这样能生成 TCP 或 UDP 流量的软件。以下所列采用 Iperf 时, 测量链路带宽的步骤。

- 在链路两端的测试用机上安装 Iperf。
- 把链路两端的测试用机分别配置为 Iperf 服务器和 Iperf 客户端。
- 让 Iperf 生成流量, 同时用 Wireshark 自带的 IO Graphs 工具来观察流量传输速率是否稳定。



注意

在传输文件时, 应传输一个大文件, 不要传输同一目录下的多个小文件。在传输多个小文件时, 测试用机会因不停地建立或关闭 TCP 连接, 而花费额外的时间。这样一来, 将起不到很好的测量效果。

当测量出的实际带宽低于链路的理论带宽时, 请按以下步骤行事。

1. 若实际带宽低于过高出理论带宽的 5% 以内, 具体原因详见 13.2.4 节。
2. 若通过传输文件的方式, 来测量链路的带宽, 且 IO Graphs 中的图形呈锯齿状, 则有可能是通信链路存在传输问题。请检查 Wireshark 是否抓到了身背 TCP retransmissions 等字样的数据包, 同时还需登录连接 SP 链路的路由器/交换机, 检查其相关接口的计数信息。



注意

可借助于 Console 电缆或通过 Telnet/SSH 客户端软件登录路由器/交换机, 执行相关 CLI 命令 (若为 Cisco 设备, 则为 **show interface** 命令), 来检查其连接 SP 链路的接口的计数信息。当然, 若网络中部署有基于 SNMP 的网管软件, 且在路由器/交换机上开启了 SNMP 功能, 通过网管软件的界面, 也能了解到连接 SP 链路的接口的计数统计信息。

3. 若测量带宽只有理论带宽的 10%~20%, 比方说, SP 提供的链路带宽为 100Mbit/s, 但实际的测量带宽只有 10~20Mbit/s, 则很有可能遇到了双工不匹配问题。对此, 将在 13.2.3 节讨论。

链路的实际带宽低于理论带宽, 也有可能拜 SP 网络设备配置不当所赐, 只是这种情况并不多见。若测量带宽远低于理论带宽, 且原因不在以上所列, 就应该与 SP 仔细沟通, 令其检查自己的网络设备配置。

13.2.3 幕后原理

读者需理解以下两个术语的定义, 尤其应掌握两者之间的区别。

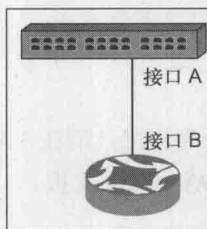
- **带宽 (Bandwidth)**: 指一条通信链路每秒所能传输的总的比特数。

- **吞吐量 (Throughput):** 指一条通信链路的两个端点之间每秒所能传输的应用程序的有效字节数。

要想获悉一条通信链路的带宽,除了询问提供链路的 SP 以外,还可以在传输大文件的同时,借助于 Wireshark 自带的 IO Graphs 工具来进行测量。

若通信链路为 100Mbit/s (或 1000Mbit/s) 以太网链路,只要链路两端的交换机/路由器接口被分别配置为自动协商和 100Mbit/s 全双工,则势必会出现双工不匹配问题。此时,通信链路的实际测量带宽将会远低于其理论带宽。

像图 13.2 那样将一台网络设备 (图中所示为一台路由器) 接入 LAN 交换机时,若两设备间的互连接口 (的双工模式和速率) 都是手工配置,比如,都配置为 100Mbit/s 全双工 (Full Duplex, FDX),则一般不太可能会出现双工不匹配问题 (图 13.2 表格中的情形 1)。



情形	配置值		实际值		状态
	Port A	Port B	Port A	Port B	
1	100FD	100FD	100FD	100FD	V
2	100FD	AUTO	100FD	100HD	X
3	AUTO	100FD	100HD	100FD	X
4	AUTO	AUTO	1000FD	1000FD	

V: 正常, X: 双工不匹配

图13.2

若两端互连接口 (假设交换机与路由器的接口均为吉比特接口) 都启用默认配置 (自动协商), 一般也不会出现问题 (图 13.2 表格中的情形 4)。

若把两设备间的互连接口分别配置为 100Mbit/s 全双工和自动协商, 或 100Mbit/s 全双工和 100Mbit/s 半双工, 则势必会出现双工不匹配问题 (图 13.2 表格中的情形 2、3)。此时, 通信线路将会出现丢包或速率骤降等问题。

13.2.4 拾遗补缺

测量订购自 SP 的通信链路的带宽时, 一般都会碰到实际测量值要略高或略低于理论值的情况。比方说, 若所测链路的理论带宽为 10Mbit/s, 但由于支撑该链路的基础设施采用的是 SDH 或 SONET 技术 (该链路实际上是由 5 条 VC-12 逻辑链路捆绑而成), 因此其理论带宽应为 $5 \times 2.176 \text{ Mbit/s}$, 即 10.88 Mbit/s。

换言之, 若两站点网络之间都采用 10Mbit/s 的互联网链路, 以 site-to-site VPN 方式互连, 即便这两条链路都由同一家 ISP 提供, 但拜赐于 VPN 自身的加密机制, 应用程序流量的实际传输速度会打 5%~10% 的折扣。也就是说, 在这种网络环境中传输文件时, 即便互联网链路

的带宽利用率达到了 100%，文件的实际传输速率（吞吐量）最多也只有 9~9.5Mbit/s。

13.3 测量每个用户及每种应用所占用的通信链路的带宽

在很多情况下，不仅需要知道一条通信链路（或连接服务器的交换机端口）的总带宽，还需要知道是哪台主机（对应于具体的 IP 地址）或哪种应用（对应于具体的 IP 地址+端口号）具体占用了多少带宽。本节会介绍如何测量每个用户（对应于具体的 IP 主机）及每种应用（对应于具体的 IP 地址+端口号）所占用的通信链路的带宽。

要想得到测量结果，既可以使用某些专用工具（比如，用于交换机的 **RMON1**、**RMON2** 和 **sFlow**，用于路由器的 **Cisco Netflow** 或 **Juniper Jflow**）从网络设备采集相关数据，也可以在交换机上开启端口镜像功能，通过 **Wireshark** 来获悉连接受监控链路的端口的流量分布情况。本机将重点关注后一种测量方法。

13.3.1 准备工作

要想使用 **Wireshark** 来获悉某条通信链路所承载的流量的分布情况，请在 LAN 交换机上开启端口镜像功能，将连接受监控链路的端口的流量重定向至 **Wireshark** 主机。

13.3.2 操作方法

要想获悉受监控链路所承载的流量的基本分布情况，请按以下步骤行事。

➤ 如何获悉常规的流量分布情况。

1. 在 **Wireshark** 抓包主窗口中，点击 **Statistics** 菜单下的 **Conversations** 菜单项。
2. 在弹出的 **Conversations** 窗口中，会显示出自开始抓包以来由每台 IP 主机和每种应用所生成的数据包个数、字节数等统计信息。
3. 还可以使用第 11 章介绍过的诸如 **Compass** 之类的图形化专用工具。

➤ 如何使用 **IO Graphs** 工具，同时配搭显示过滤器，基于 IP 地址和/或（TCP/UDP）端口号，来了解特定 IP 主机或应用所消耗的链路带宽。

1. 在 **Wireshark** 抓包主窗口中，点击 **Statistics** 菜单下的 **IO Graphs** 菜单项。
2. 在弹出的 **IO Graphs** 窗口中配置并应用显示过滤器，筛选出指定 IP 主机或具体应用程序生成的流量。第 5 章已经介绍过了 **IO Graphs** 工具的用法。

➤ 要想用 **Wireshark** 连续监控某条链路的流量分布情况，请在抓包之前先点击 **Capture**

菜单下的 **Option** 菜单项，在弹出的 Capture Options 窗口中，除了要点选用来抓包的网卡以外，还应勾选 **use multiple files** 和 **ring buffer with** 复选框。当然，也可以使用 Netflow 或 Jflow 这样的工具来监控路由器端口的流量分布情况^①。

13.3.3 幕后原理

在测量一条链路所承载的流量的分布情况时，若选用的工具是 Wireshark，请按第 1 章所述方法进行抓包并分析。

若选用的是 Netflow 和 Jflow 等直接从路由器采集数据的工具，则路由器会定期把（被采集的）数据发送到安装了相应管理控制台软件的服务器。

若采用 RMON1 和 RMON2 这样的工具，且受监控交换机支持这两种工具，则可以从能读取 RMON1/RMON2 MIB 的 SNMP 网管软件获取受监控链路所承载的流量的分布情况。RMON1 可提供第 1、2 层的统计信息；RMON2 则能提供第 3、4 层的统计信息。定义 RMON 的 RFC 文档有：RFC 2613、2819、3577 和 4502。

至于运行在第三层以上的网络设备，比如，防火墙、入侵检测系统、深度包检测系统、WAN 加速器等，都能提供自身各端口所承载的流量的分布统计信息。

13.3.4 进阶阅读

欲了解前文提及的众多工具的更多详情，请访问以下链接：

Cisco Netflow: http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html

<http://www.ietf.org/rfc/rfc3954.txt>

Juniper Jflow:

<http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ipsservices/html/ip-jflow-stats-config2.html>

sFlow:

<http://www.sflow.org/index.php>

<http://tools.ietf.org/html/rfc3176>

13.4 借助Wireshark，获悉链路上的延迟及抖动状况

但凡网络应用，其性能都要受延迟和抖动的制约。要想监控一条链路的延迟和抖动状况，可在链路一端的主机上长 ping 链路对端的主机，同时仔细观察 ping 命令的输出。而 Wireshark 并不能直接测量出端到端的网络延迟，但能通过帧间延迟（inter-frame delay）状况来反映延迟

^① 译者注：原文是“*For continuous monitoring, use Wireshark with multiple files with ring buffer, or use tools such as Netflow or Jflow for router monitoring*”。碰到这种文字和作者，真是没话好说。

对网络流量乃至各种网络应用的影响。

本节将介绍如何利用 Wireshark 来获悉通信链路上的延迟及抖动状况，下一节会讲解如何发现由延迟和抖动所导致的故障。

13.4.1 准备工作

要想得知一条通信线路的延迟状况，首先，应在链路一端的主机上 ping 链路对端的主机，同时需仔细观察 ping 命令的输出；然后，再在 LAN 交换机上开启端口镜像功能，将连接该通信链路的端口的流量重定向至 Wireshark 主机。

13.4.2 操作方法

要利用 Wireshark 来获悉某股 TCP 或 UDP 数据流中的帧间延迟状况，请按以下步骤行事。

1. 筛选出该股数据流。

- 在 Wireshark 抓包主窗口的数据包列表区域内选中一个隶属于该股 TCP 或 UDP 数据流的数据包。
- 点击右键，在弹出的右键菜单中选择 **Follow TCP Stream** 或 **Follow UDP Stream** 菜单项。
- 把在 Wireshark 抓包主窗口的 Filter 输入栏内自动生成的显示过滤表达式（图 13.3 中的位置①）复制进缓冲区。

2. 点击 Wireshark 抓包主窗口中 **Statistics** 菜单下的 **IO Graph** 菜单项。

3. 在弹出的 IO Graph 窗口中，点击 Y 轴区域中的 **Unit** 下列菜单，选择 **Advanced...** 菜单项（图 13.3 中的位置②）。

4. 在陡然增大的 IO Graph 窗口中，将已复制进缓冲区的显示过滤表达式，粘贴进 **Graph 1** 按钮右边的 Filter 输入栏（图 13.3 中的位置③）。

5. 点击 Filter 输入栏右边的 **Calc** 下拉菜单，选择 **AVG(*)** 菜单项（图 13.3 中的位置④）。

6. 在 Calc 下拉菜单右边的输入栏（图 13.3 中的位置⑤）里输入显示过滤表达式 `frame.time_delta_displayed`。

7. 按下 Graph 按钮，在 IO Graph 窗口上半部分的图形显示区域中（图 13.3 中的位置⑥），将会显示出该股数据流中各数据包之间的平均延迟时间（单位为毫秒）。

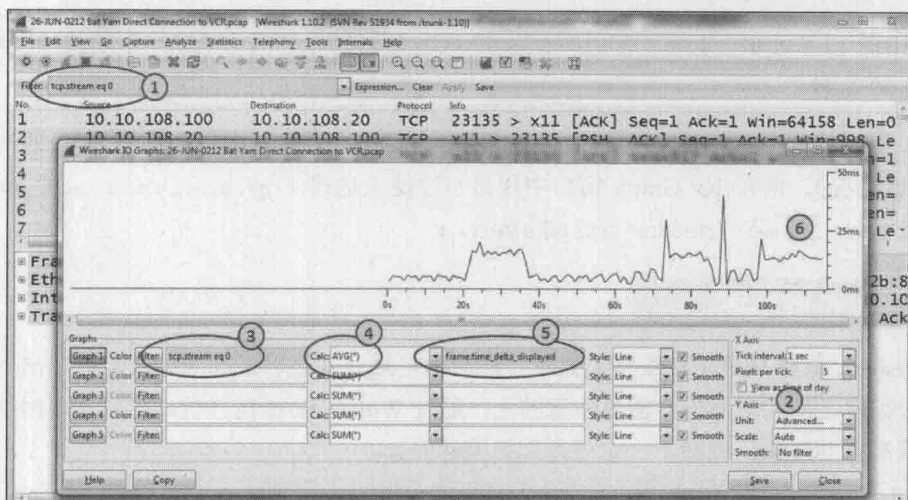


图13.3

8. 在 Wireshark 抓包主窗口中, 点击 **Statistics** 菜单中 **TCP Stream Graph** 菜单项下的 **Round Trip Time Graph** 子菜单项, 会弹出 TCP Graph 窗口。通过该窗口中观察到的结果与图 13.4 所示相同。

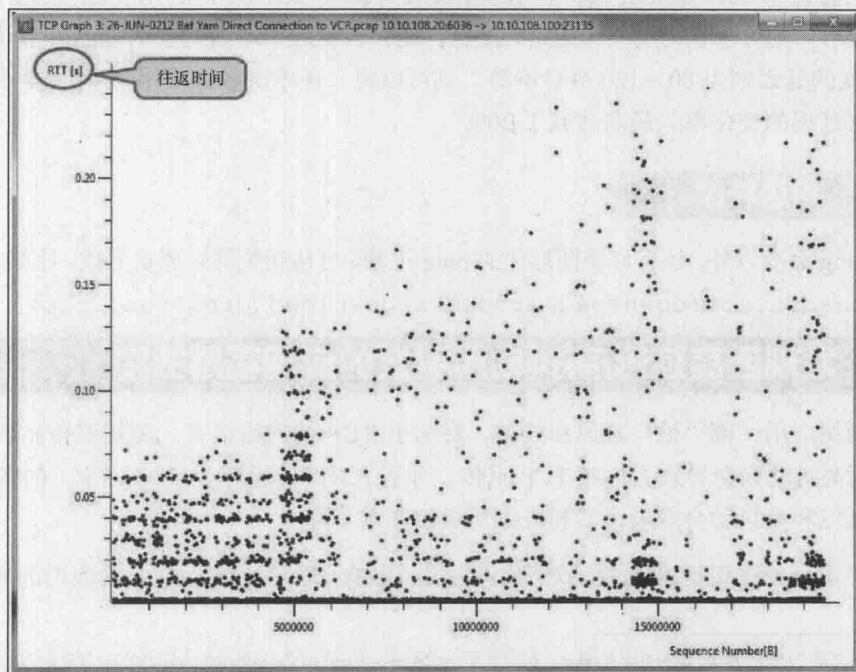


图13.4

9. 由图 13.4 可知, 隶属于那股数据流的 TCP 数据包的 RTT 值短则 10 毫秒, 长则 200~300 毫秒。
10. 要想获悉发送方从发出 TCP 报文段到得到接收方的确认所花费的时间 (即传输层延迟状况), 请在 IO Graph 窗口中用显示过滤表达式 `tcp.analysis.ack_rtt` 替换 `frame.time_delta_displayed`。

13.4.3 幕后原理

Wireshark 软件只是先从线路上抓包, 然后再记录并 (能以不同的格式) 显示出 (先后抓到的) 数据包之间的时间差。需要注意的是, 通过 Wireshark 抓包分析, 可以发现网络中存在高延迟或高抖动现象, 但却不一定能得知是因何而起。

延迟 (delay) 是指源端主机从发出数据包到收到 (目的端主机发回的) 相应的回馈数据包所花费的时间, 这通常也被称为 RTT^①。只需借助于 ping 命令之类的工具, 便可以测量出网络 (链路) 的延迟。延迟的单位可以是秒、毫秒或微秒等。

在 IP 网络中, 抖动 (Jitter) 是指延迟的变化程度。比方说, 在主机 A 上 ping 主机 B 100 次 (即让主机 A 向主机 B 发送 100 个 ICMP echo request 数据包), 若每 ping 一次的延迟都固定为 100 毫秒, 则可以认为网络 (链路) 没有抖动; 若 ping 100 次的平均延迟为 100 毫秒, 但每 ping 一次的延迟则为 80~120 毫秒不等, 则可以说 (在本次 ping 操作期间), 网络 (链路) 的抖动率 (延迟的变化率) 最高达到了 20%^②。

13.4.4 拾遗补缺

除了 ping 命令以外, 还有许多图形化的 ping 工具, 可从相关网站免费下载, 比如 http://www.colasoft.com/download/products/download_ping_tool.php。

13.5 发现因高延迟/高抖动所引发的应用程序故障

但凡网络应用, 都“怕”延迟和抖动。对基于 TCP 的应用而言, 高延迟会显著降低其传输效率, 而高抖动则会导致丢包或 TCP 重传。对基于 RTP (运行于 UDP 之上) 的多媒体应用而言, 高延迟和高抖动会造成通话和视频质量的严重下降。

本节将重点关注高延迟/高抖动对 TCP 协议的影响, 以及对基于 TCP 的应用的影响。高延

^① 译者注: 原文是 “Delay is the time that it takes a packet to get from one end of the network to the other. It is usually referred to as RTT”。译文酌改。

^② 译者注: 原文是 “Jitter in IP networks measure the variations in delay. For example, if we have an average delay of 100 ms, and it varies between 80 ms and 120 ms, the jitter is 20 percent”。译文如有不妥, 请指正。

迟/高抖动对基于 UDP 的 RTP 协议的影响详见第 12 章。

13.5.1 准备工作

要是在 ping 远程站点内的主机的时候，在 ping 命令的输出中观察到“time”值较高，而与此同时，Wireshark 也抓到了很多身背 retransmissions 字样的 TCP 数据包，那么网络中就有可能存在与通信链路或与应用程序交付有关的高延迟现象^①。要想查明原委，仍需借助端口镜像功能，将交换机上的“关键”端口的流量重定向至 Wireshark 主机。

当网络中频繁出现 TCP 重传及 TCP 重复确认现象时（即 Wireshark 抓到了超多身背 retransmissions 和 duplicate Acks 字样的数据包时），判断罪魁祸首是否为高延迟或高抖动是本节所关注的重点。

13.5.2 操作方法

若 Wireshark 抓到了很多身背 TCP retransmissions 字样的数据包，请按以下思路来判断罪魁祸首是否为高延迟或高抖动。

1. 检查身背 TCP retransmissions 字样的数据包是否都涉及同一 IP 地址或同一种应用程序。若是，则罪魁祸首很有可能是该应用程序的服务器端/客户端软件故障，或者是安装服务器端/客户端软件的主机不给力，不太可能是高延迟/高抖动问题。

若身背 TCP retransmissions 字样的数据包涉及多个 IP 地址或多种应用程序，则多半要拜高延迟/高抖动问题所赐，比如，因 WAN 链路质量不佳而引发的高延迟/高抖动问题。

2. 在 Wireshark 抓包主窗口的 Filter 输入栏内，输入显示过滤表达式 `tcp.analysis.retransmissions`（图 13.5 中的位置①），点击 **Apply** 按钮，让所有身背 TCP retransmissions 字样的数据包单独露面。
3. 在 Wireshark 抓包主窗口的数据包列表其内，先选中一个身背 TCP retransmissions 字样的数据包，到数据包结构区域中点击 `transmission control protocol` 之前的+号，找到 `SEQ/ACK analysis` 部分，再点击之前的+号，直至发现图 13.5 中位置②、③所示信息。

- 图 13.5 中位置②所示信息的含义是：（TCP 发送方）在原始 TCP 报文段发出之后（由于没有收到接收方的确认，过了）**0.225003000** 秒，重新发送了本 TCP 报文段。

^① 译者注：原文是“When you ping a remote site and get high delays, and in the Wireshark you see many retransmissions, it can be because of high network or applications delay”。原文不入流，译者给改了改，如有不妥，请指正。

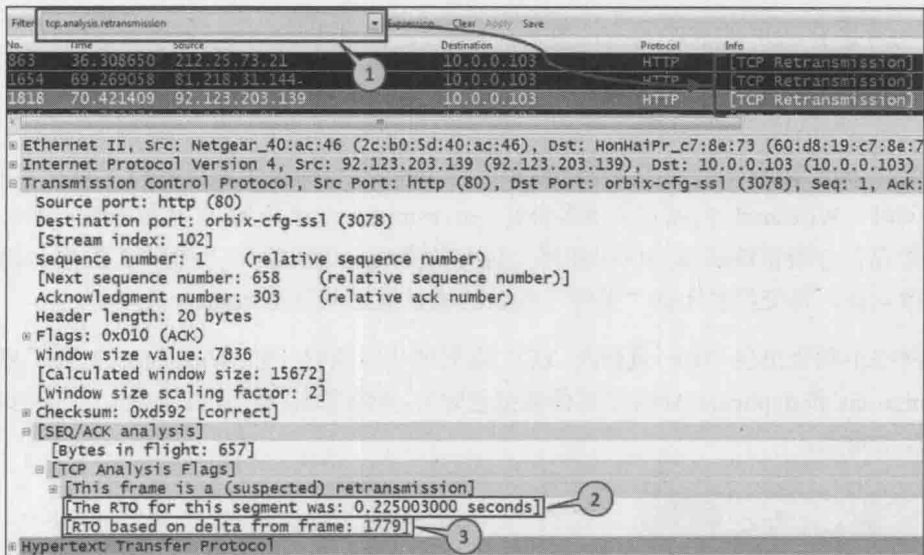


图13.5

- 图 13.5 中位置③所示信息的含义是：本数据包所含 TCP 报文段是编号为 1779 的数据包所含 TCP 报文段的重传 TCP 报文段。
4. 对 LAN 链路而言，TCP 超时重传（Retransmission Time Out, RTO）计时器值约为 0.2 秒；对 WAN 链路而言，则为 0.3~0.4 秒，但最初会假设为 0.2 秒。对 TCP RTO 机制的解释详见 13.5.3 节。
 5. 要想获悉隶属于某条 TCP 数据流的各数据帧（TCP 报文段）之间的延迟状况，可借助于 **IO Graphs** 窗口，同时在其中配置并应用以下显示过滤表达式，如图 13.6 所示。
 - `tcp.stream eq <the stream number>`：作用是筛选出感兴趣的 TCP 数据流。该显示过滤表达式可由 Wireshark 自动生成，生成方法是：1. 在 Wireshark 抓包主窗口的数据包列表区域内，选中一个隶属于某股 TCP 数据流的数据包；2. 点击右键，在弹出的右键菜单中选择 **Follow TCP Stream** 菜单项。这么一点，该显示过滤表达式就会在 Wireshark 抓包主窗口的 Filter 输入栏内自动现身了。
 - `frame.time_delta`^①：在筛选出了感兴趣 TCP 数据流之后，需结合使用该显示过滤表达式，让 Wireshark 显示出隶属于该 TCP 数据流的各数据帧（TCP 报文段）之间的延迟（抓取时间间隔）。

① 译者注：译者认为应该是 `frame.time_delta_displayed`，用 `frame.time_delta` 一点意义都没有。

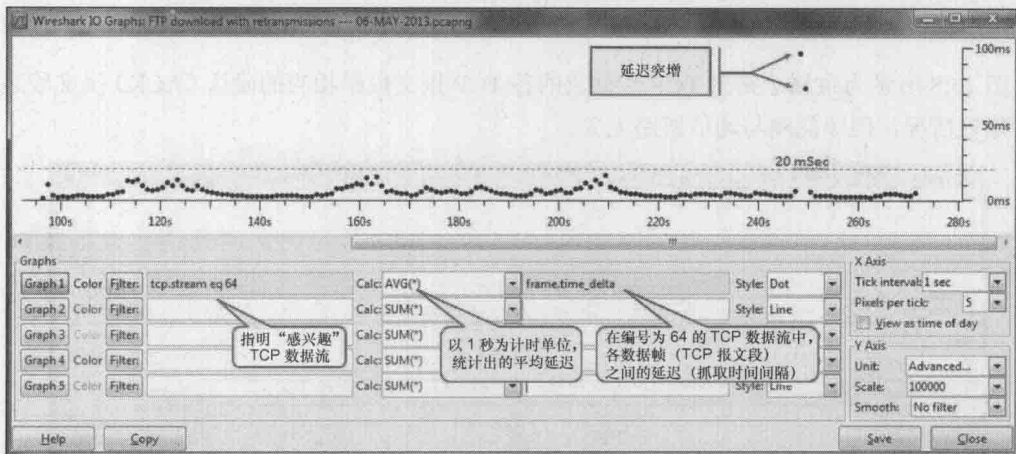


图 13.6

由图 13.7 可知, 该 TCP 数据流 (在受监控链路上) 传输状况稳定, 其麾下的各数据帧 (TCP 报文段) 之间的延迟基本保持在 20 毫秒以内, 只有在抓包开始后的第 250 秒处, 出现 2 次延迟突增的情况。这 2 次延迟突增, 导致了 TCP 重传。

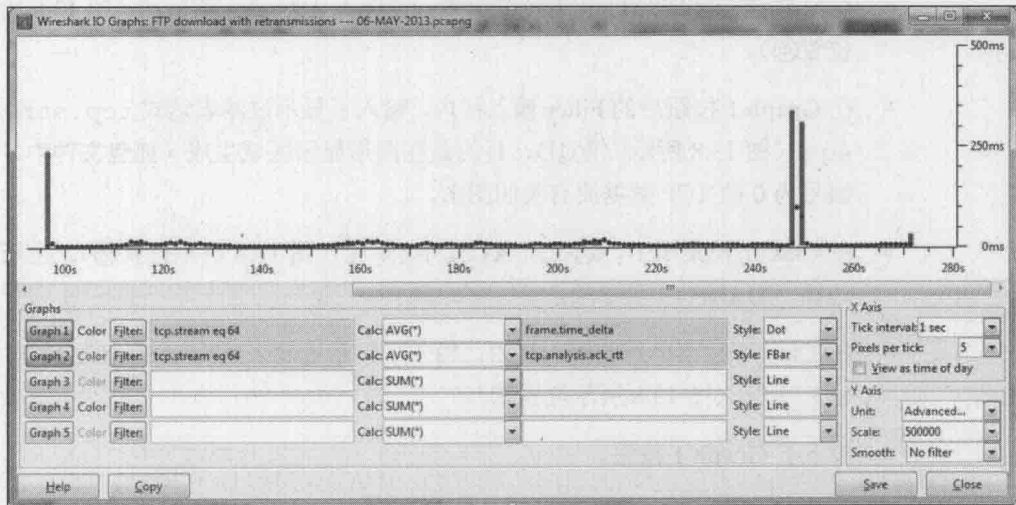


图 13.7

6. 可在 IO Graphs 窗口中结合使用显示过滤表达式 `tcp.stream eq <the stream number>` 与 `tcp.analysis.ack_rtt`, 来显示隶属于某条 TCP 数据流的各 TCP 报文段跟相应的确认 (ACK) 报文段之间的延迟情况。图 13.8 同时显示出了在那条编号为 64 的 TCP 数据流中, 各数据帧 (TCP 报文段) 之间的延迟, 以及各 TCP 报文段跟

相应的确认 (ACK) 报文段之间的延迟情况。

图 13.8 所示为隶属于某条 TCP 数据流的各 TCP 报文段跟相应的确认 (ACK) 报文段之间的高延迟情况, 但该问题与通信链路无关。

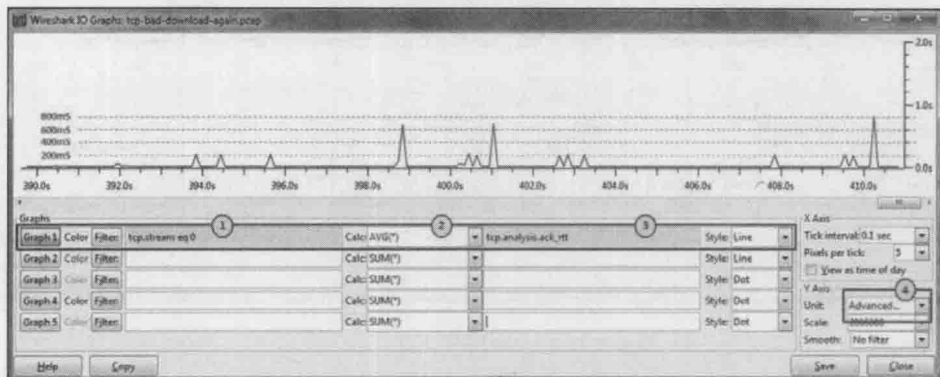


图13.8

7. 先来介绍一下图 13.8 所示 IO Graphs 窗口的设置情况。

- 在 Y 轴坐标区域的 Unit 下拉菜单中, 点选了 **Advance** 菜单项 (图 13.8 所示位置④)。
- 在 **Graph 1** 按钮后的 Filter 输入栏内, 输入了显示过滤表达式 `tcp.stream eq 0` (图 13.8 所示位置①), 目的是在图形显示区域生成 (抓包文件中) 与编号为 0 的 TCP 数据流有关的图形。
- 在 **Clac** 下拉菜单中, 选择了 **AVG(*)** 菜单项 (图 13.8 所示位置②), 也可以选择 “**MAX(*)**” 菜单项。
- 在 **Clac** 下拉菜单后的输入栏内, 输入了显示过滤表达式 `tcp.analysis.ack_rtt` (图 13.8 所示位置③)。
- 按下了 **Graph 1** 按钮。

按照上述设置, 通过观察 IO Graphs 窗口的图形显示区域, 即可了解到在那条编号为 0 的 TCP 数据流中, 各 TCP 报文段跟相应的确认 (ACK) 报文段之间的延迟情况。

8. 要想得知那条编号为 0 的 TCP 数据流在传输过程中, 是否发生过 TCP 重传, 还需做如下设置, 详情请见图 13.9。

- 在 **Graph 2** 按钮后的 Filter 输入栏内, 输入了显示过滤表达式 `tcp.stream`

eq 0 (图 13.9 所示位置①), 目的是在图形显示区域生成 (抓包文件中) 与编号为 0 的 TCP 数据流有关的图形。

- 在 **Clac** 下拉菜单中, 选择了 **COUNT FRAMES(*)** 菜单项 (图 13.9 所示位置②)。
- 在 **Clac** 下拉菜单后的输入栏内, 输入了显示过滤表达式 `tcp.analysis.ack_rtt` (图 13.9 所示位置③)。
- 按下了 **Graph 2** 按钮。

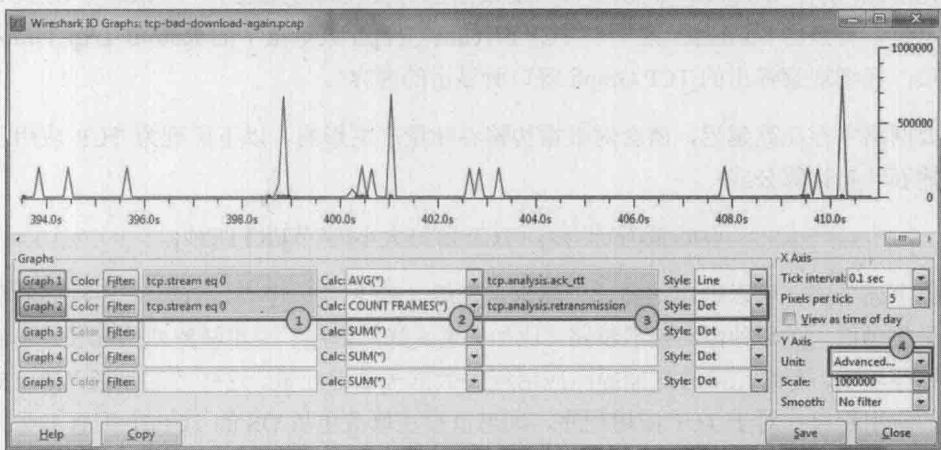


图 13.9

9. 通过图 13.9 所示的 Wireshark IO Graphs 窗口截屏, 可以很清楚地观察到在那条编号为 0 的 TCP 数据流中, TCP 重传与 TCP 确认的高延迟紧密挂钩。通过 IO Graphs 窗口所示图形, 还判断不出导致 TCP 重传的具体原因。也就是说, 出现了那么多次 TCP 重传既有可能是通信链路问题, 也有可能是应用程序问题, 还有可能是安装了应用程序的主机问题。要想找出具体原因, 请阅读第 9 章。

10. 除了 TCP 报文段跟与之配套的确认 (ACK) 报文段之间的高延迟以外, 丢包也会导致 TCP 重传。

13.5.3 幕后原理

TCP 发送方在发送数据 (发出 TCP 报文段) 的同时, 会启动一个计时器, 若该计时器到期, 仍未得到数据接收方的 (TCP) 确认 (报文段), 便会重新传送之前所发数据 (TCP 报文段)。该计时器所指定的时长被称为超时重传时间 (**Retransmission Time Out, RTO**)。定义这一 TCP 超时重传机制的第一份标准文档为 RFC 1122。在该 RFC 中, 明文规定了 TCP 超时

重传时间的计算方法，该计算方法刊载在 Jacobson V.和 M. Karels 于 1998 年发表的论文“Congestion Avoidance and Control”中。RFC 1122 在 2000 年被更新为 RFC 2998，到了 2011 年，又被更新为当前使用的 RFC 6298。

13.5.4 拾遗补缺

要想弄清隶属于某条 TCP 数据流中的各 TCP 报文段跟相应的确认 (ACK) 报文段之间的延迟变化情况，请在 Wireshark 抓包主窗口的数据包列表区域内先选中一个隶属于该 TCP 数据流的数据包，再点击 **Statistics** 菜单中 **TCP Stream Graph** 菜单项下的 **Round Trip Time Graph** 子菜单项，仔细观察弹出的 TCP Graph 窗口所显示的内容^①。

只要网络中存在高延迟，就会对数据传输吞吐量产生影响。以下所列为 TCP 应用程序的数据传输吞吐量计算公式：

$$\text{吞吐量[字节/秒]} = \text{TCP 窗口大小[字节]} / \text{RTT [秒]}$$

根据上述公式，可以得出这样一个结论：延迟 (RTT 值) 越高，TCP 应用程序的数据传输吞吐量将越低。在那种由高延迟链路（比如，老式蜂窝链路、卫星链路或国际长途链路）构成的网络环境中，有好几种方法能提高应用程序数据传输吞吐量。这些方法包括：使用支持并发连接的应用程序；对于 TCP 应用程序，则可以设法修改主机 OS 的 TCP/IP 配置参数，增加 TCP 窗口的大小（Window Vista 及更高版本的 Windows OS，外加多种版本的 LINUX OS 都默认自带能调整 TCP 窗口大小的配置工具）^②。

① 译者注：原文只有一句“For delay variations, you can also navigate to Statistics | TCP Stream Graph | Round Trip Time Graph”。作者把英文的简洁性发挥到了极致。

② 译者注：整段原文为“From here we can see that the higher the delay is, the lower the throughput becomes. In networks with high delays, for example, old cellular networks, satellite lines, and long distance international lines, we have several methods to improve the application's throughput. Among these methods are applications that use multiple connections per application, usage of the TCP increases the window size (comes as default in Window Vista and the higher versions, along with various Linux versions)”。文字不通，译文如有不妥，请指正。

第14章

认识网络安全

本章涵盖以下主题：

- ▶ 发现异常流量模式；
- ▶ 发现基于 MAC 地址和基于 ARP 的攻击；
- ▶ 发现 ICMP 扫描和 TCP SYN 端口扫描；
- ▶ 定位高级 TCP 攻击；
- ▶ 发现针对某些应用层协议的暴力攻击。

14.1 简介

信息安全是 IT 领域中的重要环节，其主要目标是为了加固组织机构的信息系统，使之能对抗来自内部或外部的形形色色的攻击。那些来自内部或外部的攻击，都得通过网络来发动，无非是源于内/外网而已，因此同样可以被 Wireshark（或稍后提及的其他工具）所感知。

要想使得网络免受恶意流量的攻击，首先需要分清什么是正常流量。然后，才能识别出恶意流量是如何伪装成正常流量的。可能会形成攻击的恶意或异常流量包括：ARP、IP 或 TCP 扫描流量，单方面的 DNS 响应流量，TCP 标记位置位方式古怪的 TCP 流量，源/目的 IP 地址或源/目的（TCP/UDP）端口号怪异的流量等。

能分清何为网络问题、何为安全问题，同样十分重要。比方说，若网络中出现了 ICMP 扫描行为，则既有可能是恶意软件所为，也有可能是网管软件在探索网络设备时的必要之举；而 TCP 扫描行为也未必只有蠕虫病毒才能引发，软件 bug 同样可以触发。本章会对此展开深入探讨。

本章首先会讲解正常流量与异常流量之间的差别，然后会介绍形形色色的攻击手段，以及如何发现并制止这些攻击。

14.2 发现异常流量模式

本节会讲解何为正常流量，何为异常流量，同时会介绍两者之间的差别。

14.2.1 准备工作

首先，应安置好 Wireshark 主机。现以图 14.1 为例，来介绍如何安置 Wireshark 主机。

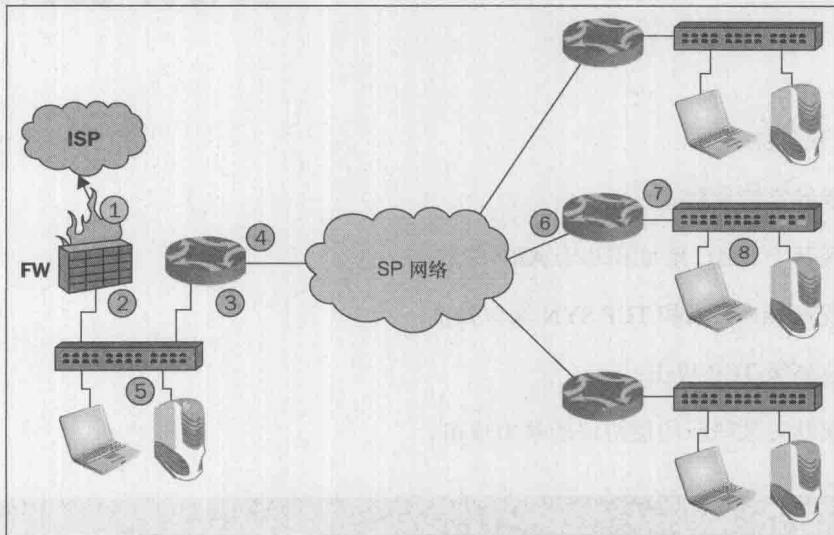


图14.1

1. 若怀疑攻击流量来自于互联网，请将 Wireshark 主机置于防火墙之前（图 14.1 中所示位置①）；若怀疑来自于互联网的攻击流量已被防火墙放行，请将 Wireshark 主机置于防火墙之后（图 14.1 所示位置②）^①。
2. 若怀疑恶意流量发源于远程站点（分支机构）网络，请将 Wireshark 主机至于图 14.1 所示位置④或位置③处（需在主站点网络的核心交换机上开启端口镜像功能）。安置好 Wireshark 主机之后，可根据分配给各远程站点的 IP 网络号来配置显示过滤器，逐一比对发源于各远程站点网络的流量，以求定位出恶意流量的来源。
3. 若判断出恶意流量发源于某一具体的远程站点，请将 Wireshark 主机至于图 14.1 所示

^① 译者注：原文是“When you suspect an attack that comes from the Internet, locate Wireshark after the firewall (1), and when you suspect that it crosses the firewall, locate it before (2)”。译者认为作者前（before）后（after）不分，译文酌改。

位置⑦或位置⑧处（需在远程站点网络的核心交换机上开启端口镜像功能），亦即远程站点 WAN 路由器的 WAN 侧或 LAN 侧。

4. 若怀疑攻击流量发源于某台 PC 机或服务器，请将 Wireshark 主机至于图 14.1 所示位置⑧处，同时在（连接 Wireshark 主机以及可能产生恶意流量的 PC 机或服务器的）交换机上开启端口镜像功能，将相应端口的流量重定向至 Wireshark 主机

其次，应仔细观察抓取到的各类流量，尝试判断哪些是正常流量，哪些是值得关注的异常流量。

在执行上述操作之前，请确保手头能有一份最新的网络拓扑结构详图，这份网络拓扑图应包括以下内容。

- IP 编址详情（比如，服务器的 IP 地址、各站点 LAN 的 IP 网络号等）。
- 网络设备（路由器、交换机等网络通信设备）的 IP 地址及拓扑连接状况。
- 网络安全设备（防火墙、IPS/IDS、WAF、数据库防火墙、应用防火墙、防病毒系统以及能生成、过滤或转发流量的其他网络安全设备）的 IP 地址及拓扑连接状况。
- 网络所承载的应用程序类型、各类应用程序所使用的 UDP/TCP 端口号，以及安装各类应用程序服务器端软件的服务器主机的 IP 地址等。

14.2.2 操作方法

若怀疑恶意流量发源于内网，对内网流量抓包分析时，应关注下列事宜。

1. 流量的源、目 IP 地址是否异常^①。

- 正常：流量的源或目的 IP 地址均在合法的 IP 地址范围之内（隶属于已分配的内网 IP 地址范围）。
- 异常：流量的源或目的 IP 地址不在合法的 IP 地址范围之内（不隶属于已分配的内网 IP 地址范围）。

2. 流量的源、目 TCP/UDP 端口号是否异常。

- 正常：流量的源、目端口号与运行于网络中的应用层协议相匹配，比方说，若在内网中运行的应用层协议包括 HTTP、NetBIOS、RDP、FTP、POP3、SMTP、DNS 等，则 Wireshark 所抓到的源、目（TCP/UDP）端口号为 80、137/138/139、3389、20/21、25、110、53 的流量都属于正常流量。

^① 译者注：原文是“Traffic that is generated from known addresses (in the organization)”。作者绕来绕去还讲不到点子上，译文酌改。

- 异常：流量的源、目端口号与应用层协议不匹配，譬如，若某台 Web 服务器明明未开启远程桌面服务，但却总是莫名其妙地收到 RTP 流量，则发往该服务器的 RTP 流量就属于异常流量。

3. TCP 流量的标记位的置位方式是否异常。

- 正常：在建立 TCP 连接时，Wireshark 应抓到相关的三次握手 TCP 报文段（SYN 位置 1/SYN 和 ACK 位同时置 1/ACK 位置 1）；在快速拆除 TCP 连接时，Wireshark 应抓到相应的 RST 位置 1 的 TCP 报文段；在关闭 TCP 连接时，Wireshark 应抓到四次握手报文段（双方各自发出 FIN 位置 1/FIN 位和 ACK 位同时置 1 的 TCP 报文段）。
- 异常：若通过 Wireshark 抓包了解到，有大量 SYN 位置 1 的 TCP 数据包涌向一或多台主机，而且这些 TCP 数据包的源 IP 地址还千奇百怪，这就属于异常流量（通常，这意味着有人在执行 TCP 端口扫描，详情请见后文）。此外，标记位置位方式怪异（比如，RST 或 FIN 位与 URG 位同时置 1）的 TCP 流量也属于异常流量。

4. 某台（或某几台）主机的上、下行流量的速率是否总是固定不变。

- 正常：一般而言，对任何一台主机来说，由其生成的上、下行流量的速率都不应该总是固定不变，这是因为用户使用网络的行为总是在不停的变化。下载文件、浏览网页、收发邮件、使用远程桌面时所产生的流量的速率肯定各不相同。
- 异常：若发现由一台主机所生成的上、下行流量的速率总是固定不变，则使用主机的用户不是在观看视频，就是在偷着 P2P 下载，或是有人对其进行扫描。此时，需仔细检查。图 14.2 所示为通过 Wireshark IO Graph 工具观察到的上述异常情况。

5. 广播流量是否异常。

- 正常：网络中每台主机在特定时间段内发出的广播包的数量应该在合理范围之内。
- 异常：单台主机每秒发出数十、数百乃至数千个广播包。

6. DNS 流量是否异常。

- 正常：对于执行标准 DNS 查询的主机来说，每秒最多也只会生成几十个 DNS 查询数据包。
- 异常：有巨量 DNS 查询或响应数据包在网络中泛滥；Wireshark 抓到了很多 DNS 响应数据包，但却未抓到“配套”的 DNS 查询数据包。

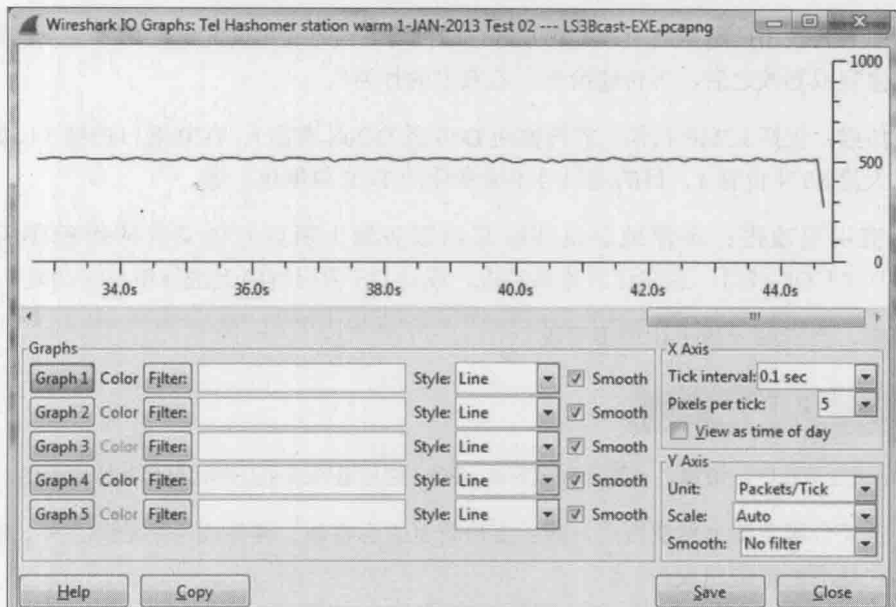


图14.2

14.2.3 幕后原理

网络取证跟电视里的刑侦剧所描述的场景颇为类似。网络工程师跟警察一样，都是先接到“报案”（用户申告），再抵达“犯罪现场”（网络机房），然后开始查找“罪犯”留下的蛛丝马迹。

不同之处在于，警察要查找的是罪犯遗留的指纹和 DNA，而网络工程师则要抓包分析流量是否异常，观察异常流量所具有的特征（模式）。

本节会深入探讨各种网络攻击手段，同时会描述当网络遭受攻击时将会出现的异常状况。此外，还将详述如何发现并化解攻击。

以下所列为几种最为常见的网络攻击手段。

- **病毒**：是指能对计算机软、硬件造成伤害的应用程序。杀毒软件可发现并清除病毒。
- **蠕虫**：是指能自我复制并自动通过网络来传播的应用程序。蠕虫爆发时，会导致网络带宽资源、计算机 CPU 资源的极度消耗。在杀灭蠕虫的那一刻，所有的一切都将风平浪静。
- **DoS（拒绝服务）和 DDoS（分布式拒绝服务）攻击**：发动这两种攻击的目的，都是要让合法用户无法正常访问网络资源。由于这两种攻击的破坏力惊人，加之发动的方法截然不同，因此不但很容易发现，区别也比较明显。

- 中间人攻击：是指当甲、乙双方进行通信时，丙方在中间先截取两者互发的消息，在偷窥或篡改之后，再传递给甲、乙双方的行为^①。
- 扫描：包括 ICMP 扫描（能转换为 DoS 或 DDoS 攻击）、TCP 端口扫描（比如，发出大量 SYN 位置 1、目的端口号不断变化的 TCP 数据包）等。
- 应用层攻击：不停地尝试连接某台服务器上所运行的某种应用程序所监听的 TCP/UDP 端口，意在让服务器宕机，或让 OS/应用程序无法行使正常功能。

接下来，会详细介绍上述攻击手段。

14.2.4 拾遗补缺

当网络遭受攻击时，迹象之一就是服务器、PC、通信链路或某些网元莫名其妙地变慢、变卡。

- 当某台服务器突然变慢时，除了要检查是否存在软、硬件或网络故障之外，还得考虑它是不是正遭受攻击。
- 若中心站点网络与某分支机构网络间的互连链路突然变慢，或许是因为链路（在传输正常流量时）拥塞，也有可能是有人正在发动 DOS/DDoS 攻击，在链路上传送垃圾流量，导致链路不畅。
- 若一台 PC 突然变慢，或许是在执行常规任务，但也不能排除已被攻陷，正尽其所有处理能力执行黑客下达的指令。

可部署下列（软、硬件）系统以使网络免受攻击。

- 防火墙：用途是防止未经授权的流量进、出指定的网络区域，可部署在 Internet 边界、内网服务器之前、两个网络区域之间，甚至能以软件的形式安装在用户的 PC 里。
- NAC（网络访问控制）系统：用途是防止未经授权的用户或设备接入网络^②。
- IDS/IPS（入侵检测/入侵防护系统）：用途是识别具有攻击属性的流量，生成告警记录或对那些流量加以拦截（视配置而定）^③。IDS/IPS 既可以是一台独立的物理设备，部署于防火墙和 Internet 链路接入交换机之间，也可以以软件或功能模块的形式驻留于硬件防火墙之内。

① 译者注：原文是 “These are attacks in which the attacker intercepts messages and then retransmits them. In this way, the attacker can eavesdrop on the traffic or change it before it gets to the destination”。译文酌改。

② 译者注：原文还有一句 “When connecting an unauthorized device to the network, you will see that the link on the device will be turned on and immediately off, and the unauthorized device will be blocked on the MAC layer”。此句太过幼稚，掠过不翻。

③ 译者注：原文是 “These systems can identify intrusion patterns and block them”。幼稚至极，酌改；后面还有一句 “There are usually two lines of defense here—one at the ISP network and one at the customer premises”，同样幼稚，掠过不翻。

- **Web 应用防火墙 (WAF)**、应用防火墙、数据库防火墙以及其他应用防护设备：这些第 7 层设备可窥得数据包的内在，能防止针对应用层的攻击。
- **Web 和 Email 过滤器**：可扫描并检测 HTTP 和 Email 流量的内容，并根据检测结果，来决定是否放行流量。

上述系统既可以是一台单独的物理设备，也可以分别以软件的形式安装在虚拟机 (VM) 上，还可以以功能模块的方式集成进同一台物理设备^①。

14.2.5 进阶阅读

14.3.4 节介绍了几种网络安全系统，现给出相关厂商的产品链接。

- **防火墙**：Checkpoint (www.checkpoint.com)、Juniper SSG 系列防火墙 (<http://www.juniper.net/us/en/products-services/security/ssg-series/>)、Cisco ASA 系列防火墙 (http://www.cisco.com/en/US/products/ps5708/Products_Sub_Category_Home.html) 等。
- **NAC**：Fore Scout (<http://www.forescout.com/solutions/network-access-control/>)、Enterasys (<http://www.enterasys.com/company/literature/nac-ds.pdf>)。
- **IDS/IPS**：Juniper IDP 系列 (<http://www.juniper.net/us/en/products-services/security/idpseries/>)、Check IPS Software Blade (<http://www.checkpoint.com/products/ips-software-blade/>)。
- **WAF**：Imperva (http://www.imperva.com/products/wsc_web-application-firewall.html)、F5 (<http://www.f5.com/glossary/web-application-firewall/>)。
- **数据库防火墙**：Imperva (http://www.imperva.com/products/dsc_database-firewall.html)、Oracle (<http://www.oracle.com/us/products/database/security/audit-vault-database-firewall/overview/index.html>)。
- **Web 和 E-mail 过滤器**：McAfee (<http://www.mcafee.com/au/products/E-mail-and-web-security/index.aspx>)、Blue Coat (<http://www.bluecoat.com/security-policy-enforcement-center>)、Websense (<http://www.websense.com>)。

^① 译者注：原文是“The features mentioned above can come as different devices, software on Virtual Machines (VMs), or features on the same device”。

//www.websense.com/content/Home.aspx)。

14.3 发现基于MAC地址和基于ARP的攻击

借助于 Wireshark, 可以很容易地发现基于 MAC 地址 (第 2 层) 和基于 ARP (介于第 2、3 层之间) 的攻击。此类攻击通常都以扫描攻击 (详见本节) 或中间人攻击 (详见第 8 章) 的面目示人。本节将介绍几种常见的攻击, 以及发动攻击的具体手段。

14.3.1 准备工作

若 ARP 请求数据包在网络中泛滥成灾, 或观察到以太网帧帧头中包含了非常规的 MAC 地址时, 请在 LAN 交换机上开启端口镜像功能, 将相关端口流量重定向至 Wireshark 主机, 进行抓包分析^①。

14.3.2 操作方法

要想发现基于 MAC 地址和基于 ARP 的攻击, 请按以下步骤行事。

1. 若网络中未划分任何 VLAN, 请将 Wireshark 主机连接到 LAN 交换机上的任一端口, 开始抓包分析^②。
2. 观察 Wireshark 抓包主窗口, 看看 ARP 数据包是否层出不穷。由于 ARP 请求数据包为广播包, 因此在网络未划分 VLAN 的前提下, 可把 Wireshark 主机连接到 LAN 交换机的任一端口, 且无需开启端口镜像功能。图 14.3 所示的 Wireshark 截屏清楚地展示了一次典型的 ARP 扫描行为。需要注意的是, ARP 扫描也可以用来办正事。比如, 某些安装了 SNMP 网管软件的主机会通过 ARP 扫描, 来发现同一 LAN 内的网络设备; 某些宽带路由器也会通过内网接口 (LAN 口) 不停地发送免费 (gratuitous) ARP 数据包。但若层出不穷的 ARP 数据包来路不正, 则应仔细查明原因。
3. 通过图 14.3 所示的 ARP 数据包, 可判断出网络中可能存在以下攻击行为。
 - Wireshark 抓到的所有 ARP 数据包均来源于同一台主机, 发包主机所设 IP 地址应该是 192.168.43.191, 但其 (通过 ARP 数据包) 查询的主机身处的 IP 子网却是 10.0.0.0/24^③。之所以会出现这种情况, 原因之一是这台主机的同

① 译者注: 原文是 “When viewing too many ARP requests on a network or when seeing non-standard MAC addresses in the network, connect Wireshark with port mirror to their source and start the capture”。想问作者, 你单凭肉眼都能得知 ARP 数据包泛滥成灾, 甚至还能看到以太网帧的内部, Wireshark 真的对你有帮助吗?

② 译者注: 原文是 “Connect Wireshark to any port on the network”, 直译为 “将 Wireshark 连接到网络中的任何端口”。请问作者, 把 Wireshark 主机连接到路由器、防火墙端口可不可以?

③ 译者注: 原文只有一句 “Two identical MAC addresses with different IP addresses”。译文酌改。

一块网卡设有两个 IP 地址，这在实战中非常常见。第二个原因是，有人在获悉了网络中一台服务器的 MAC 地址（图中所示 ARP 数据包的源 MAC 地址）之后，将自己所用主机（的网卡的）MAC 地址改成了那台服务器的 MAC 地址（每块网卡的 MAC 地址都是可以修改的）。

Time	Source	Destination	Protocol	Info	Length	Interface
0.000217	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.239?	Tell	192.168.43.191
0.000194	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.242?	Tell	192.168.43.191
0.000184	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.243?	Tell	192.168.43.191
0.000194	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.246?	Tell	192.168.43.191
0.000183	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.247?	Tell	192.168.43.191
0.000412	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.240?	Tell	192.168.43.191
0.000067	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.241?	Tell	192.168.43.191
0.000116	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.244?	Tell	192.168.43.191
0.000385	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.250?	Tell	192.168.43.191
0.000092	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.245?	Tell	192.168.43.191
0.000044	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.248?	Tell	192.168.43.191
0.000264	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.249?	Tell	192.168.43.191
0.496923	NonHaIPr_c7:8e:73	Broadcast	ARP	who has 10.0.0.212?	Tell	192.168.43.191

图 14.3

- 有人正在网络中发动中间人攻击。对中间人攻击的介绍详见 8.2.2 节。

14.3.3 幕后原理

ARP 请求数据包的目的 MAC 地址一定是广播地址，主机发出 ARP 请求数据包是为查询指定 IP 地址的 MAC 地址。若网络中 ARP 请求数据包泛滥成灾，且 ARP 数据包的源、目 IP 地址或内容异常，则可认为网络中有 ARP 攻击的征兆^①。

14.3.4 拾遗补缺

某些安装了 SNMP 网管软件的主机会不断发出 ARP 请求数据包，来探索同一 LAN 内的网络设备（执行 ARP 扫描）；还有些宽带路由器也会通过内网接口（LAN 口）不停地发送免费 ARP 数据包。因此，若 Wireshark 抓包结果表明网络中存在 ARP 扫描现象，也未必是有人在发动攻击，关键是要弄清 ARP 请求数据包的源头。更多与 ARP 扫描有关的内容详见第 8 章。

14.4 发现 ICMP 和 TCP SYN/端口扫描

扫描是指不停的针对某个网络发送探测类数据包的行为，目的是查明网络中主机的 IP 地址、服务器所侦听的端口号，以及其中部署的系统和应用程序的资源类型。

14.4.1 准备工作

当网络中有用户反应网速过慢，或网管系统检测出服务器或通信链路的负载过高时，往往

^① 译者注：原文是“ARP sends broadcasts to the network asking for the MAC address of a specific IP destination. Anything that is not according to this pattern should be considered malicious”。这完全是外行的写法，译文酌改。

就预示着有人正在对网络进行扫描。若网络中部署有安全信息及事件管理系统 (Security Information and Event Management Systems, SIEM), 可能会在有人发动扫描攻击时, 生成告警信息。如怀疑有人对网络进行扫描, 则应尽量在离攻击的起源处最近的地方安置 Wireshark 主机, 进行抓包分析。

14.4.2 操作方法

要想发现扫描攻击, 请按以下步骤行事。

1. 在离攻击的起源处最近的地方安装 Wireshark, 进行抓包分析。
 - 若 Internet 链路拥塞, 且流量来历不明, 请在 Internet 链路接入交换机上开启端口镜像功能, 将连接 Internet 链路的端口的流量重定向至 Wireshark 主机。
 - 若 (用户反映) 对某台服务器的访问速度突然变慢, 请在服务器接入交换机上开启端口镜像功能, 将连接该服务器的端口的流量重定向至 Wireshark 主机。
 - 若 (用户反映) 对某一远程站点网络的访问速度突然变慢, 请设法将连接该远程站点网络的 WAN 链路的流量重定向至 Wireshark 主机^①。
2. 在抓包时, 若 Wireshark 主机或 Wireshark 软件自身毫无反应, 则很有可能是攻击流量太过“汹涌”, 以至于它们难以招架。此时, 可先关闭 Wireshark 软件 (若抓包主机的 OS 为 Windows, 请按 Ctrl+Alt+Del 组合键, 点击“任务管理器”按钮, 在弹出的“Windows 任务管理器”窗口中结束 Wireshark.exe 进程; 若为 UNIX, 请执行 kill 命令, 终结 Wireshark 软件的进程), 再重启该软件, 将其配置为以多个文件的形式来保存抓到的数据包 (具体配置方法详见第 1 章)。
3. 扫描攻击的手段五花八门, 但都需要发出大量的 ICMP 或 TCP 数据包 (进行探测)。因此, 当网络遭受扫描攻击时, 若使用 Wireshark 抓包, 必能抓到超多来历不明的 ICMP 和/或 TCP 数据包。现举几个例子来加以说明。
4. 图 14.4 所示为一个饱受扫描攻击之苦的网络。位于远程分支机构网络中的所有用户都在反映, 当他们访问中心站点网络内的服务器 (图 14.4 中的左下角) 时, 感觉网速极慢。
 - 当作者在某个远程站点网络内安置好 Wireshark 主机 (如图 14.4 所示), 抓取并分析奔流在通往中心站点网络的 WAN 链路上的流量时, 发现其中“夹杂”着大量异常的 ICMP echo request 数据包 (③)。之所是说这些 ICMP 数

^① 译者注: 原文是 “If remote offices become slow, port mirror the lines to them”, 直译为 “若远程办事处变慢, 端口镜像连接其的线路”。

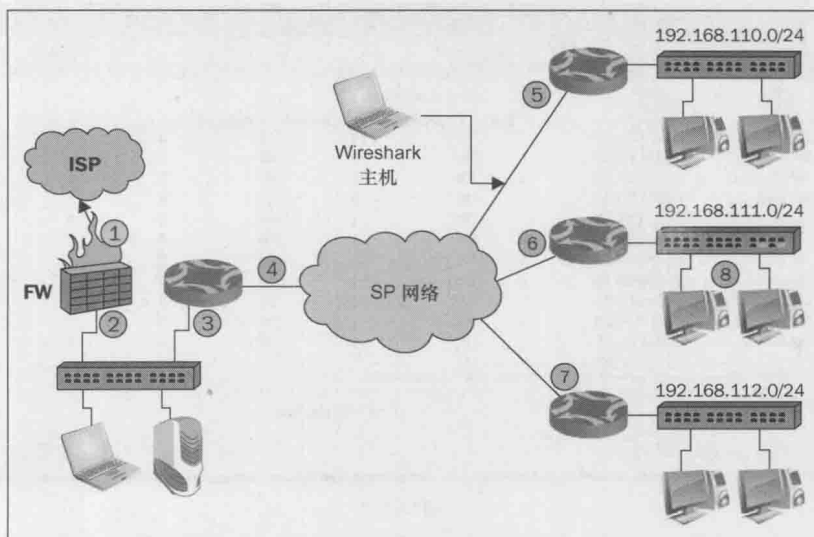


图14.4

据包“异常”，是因为它们的源 IP 地址全都隶属于 IP 网络 192.168.110.0/24 (①)，它们的目的 IP 地址则是五花八门 (②)。至于这些 ICMP 数据包的目的 IP 地址怎么个五花八门，详见图 14.5 所示的 Wireshark 抓包截屏。

Filter: 发源于 IP 子网 192.168.110.0/24 (①)		目的 IP 地址 五花八门 (②)		ICMP echo request 数据包 (③)	
No.	Time	Source	Destination	Protocol	Info
1	0.00000	192.168.110.5	192.170.2.218	ICMP	Echo (ping) request id=0x0200,
2	0.001541	192.168.110.58	192.170.2.203	ICMP	Echo (ping) request id=0x0200,
3	0.000458	192.168.110.189	192.170.0.230	ICMP	Echo (ping) request id=0x0200,
4	0.000013	192.168.110.69	218.88.25.157	ICMP	Echo (ping) request id=0x0200,
5	0.001561	192.168.110.76	192.168.37.86	ICMP	Echo (ping) request id=0x0200,
6	0.000015	192.168.110.12	192.169.204.227	ICMP	Echo (ping) request id=0x0400,
7	0.000376	192.168.110.5	192.169.254.156	ICMP	Echo (ping) request id=0x0200,
8	0.000639	192.168.110.67	192.170.6.10	ICMP	Echo (ping) request id=0x0200,
9	0.000912	192.168.110.56	192.170.2.185	ICMP	Echo (ping) request id=0x0200,
10	0.002516	192.168.110.63	192.166.52.9	ICMP	Echo (ping) request id=0x0200,
11	0.002113	192.168.110.70	192.166.2.180	ICMP	Echo (ping) request id=0x0200,
12	0.001146	192.168.110.68	192.166.2.165	ICMP	Echo (ping) request id=0x0200,
13	0.000826	192.168.110.60	192.166.252.185	ICMP	Echo (ping) request id=0x0200,
14	0.000411	192.168.110.66	192.169.214.11	ICMP	Echo (ping) request id=0x0200,
15	0.000631	192.168.110.82	192.169.221.203	ICMP	Echo (ping) request id=0x0200,

图14.5

- 仔细观察图 14.5 所示 Wireshark 抓包主窗口中数据包列表区域里的 Time 列，不难发现，Wireshark 抓取到的 ICMP echo request 数据包之间的时间间隔很短。当网络遭受扫描攻击时，则势必会在短时间内收到密集的探测流量。
- 点击 Wireshark 抓包主窗口中的 **Statistics** 菜单，选择 **Conversations** 菜单项，Conversations 窗口会立刻弹出，点击 IPv4 标签 (④)，可以观察到图 14.6 所示的景象。

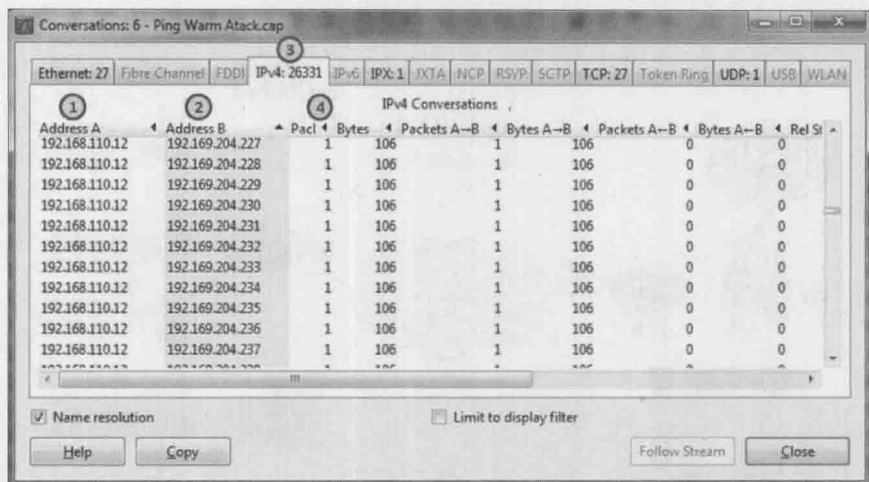


图 14.6

- 在 Conversations 窗口的 IPv4 标签页内，点击 Address A 列 (①)，让该标签页中的 IP 会话 (IPv4 Conversations) 按源 IP 地址排序。用鼠标滑轮进行翻滚，可以发现，那些异常的 ICMP echo request 数据包的源 IP 地址全都隶属于 IP 子网 192.168.110.0/24，目的 IP 地址则五花八门 (图 14.6 只是显示了众多异常的 ICMP echo request 数据包的一小部分，它们的源 IP 地址都是 192.168.110.12，目的 IP 地址虽然五花八门，但都隶属于 IP 子网 192.169.204.0/24。也就是说，主机 192.168.110.12 正在以发送 ICMP echo request 数据包的方式，扫描目的 IP 子网 192.169.204.0/24)。
- 上述局面由蠕虫病毒造成。感染了蠕虫病毒的计算机会发出 ICMP echo request 数据包，来扫描整个网络，只要有主机对这些数据包进行回应，便会感染上蠕虫病毒，然后会发起新一轮的 ICMP 扫描。中心站点网络和各分支机构网络间的 WAN 链路正是被那些进进出出的 ICMP echo request 数据包弄得拥堵不堪^①。



结论

通过 Wireshark 抓包分析，发现某条链路 (或通信信道) 上充斥着大量 ICMP echo request 数据包时，切勿掉以轻心。这有可能是安装了 SNMP 网管软件的主机在探索网络设备，也有可能是感染了蠕虫病毒的主机或服务器在扫描整个网络。

- TCP SYN 扫描是另外一种常见的扫描攻击。具体的攻击手段是，攻击者不断发出 SYN 标记位置 1、目的端口号五花八门的 TCP 探测类数据包，等待 (监听相应端口的) 主

^① 译者注：原文是 “This worm simply scans the network with ICMP requests. The moment someone answers, the worm infects him/her also, and after a few minutes, all communication lines are blocked with ICMP requests going out of the remote offices”，译文酌改。

机回复 SYN 和 ACK 标记位同时置 1 的 TCP 数据包。一旦等到了回复（扫描成功），攻击者可以进一步实施以下攻击手段。

- 攻击者可继续发出 SYN 标记位置 1 的 TCP 数据包，诱使（监听目标端口的）受攻击主机回复 SYN 和 ACK 标记位同时置 1 的 TCP 数据包，但攻击者绝不会回复 ACK 标记位置 1 的 TCP 数据包，这会使得攻击主机与受攻击主机之间建立起多条 TCP 半开连接（half-open connection）。如此行事的目的是，让受攻击主机的资源消耗殆尽，此乃 DoS 攻击的一种形式。
- 持续回复 ACK 标记位置 1 的 TCP 数据包，以正常的三次握手方式，与受攻击主机建立多条无用的 TCP 连接，目的同样是消耗受攻击主机的资源，起到 DoS 攻击的效果。

6. 当网络遭受 TCP-SYN 扫描攻击时，用 Wireshark 抓包，在抓包主窗口中的数据包列表区域所见到的数据包的特征将会与下面几图所示一致。

- 在 Wireshark 抓包主窗口的数据包列表区域中，可能会看见攻击主机向受攻击主机（被扫描的主机）发出多个 SYN 标记位置 1，且源端口号几乎一致，但目的端口号各不相同的 TCP 扫描类数据包。只因受攻击主机未监听相应的 TCP 端口，故那些 TCP 扫描类数据包多半得不到回应，如图 14.7 所示^①。

No.	Time	Source	Destination	Protocol	Info
17984	0.000061	192.168.43.191	173.194.66.116	TCP	50991 > 714 [SYN] Seq=0 win=1024 Len=0 M
17985	0.000083	192.168.43.191	173.194.66.116	TCP	50990 > 11110 [SYN] Seq=0 win=1024 Len=0
17986	0.000064	192.168.43.191	173.194.66.116	TCP	50990 > 1198 [SYN] Seq=0 win=1024 Len=0
17987	0.000071	192.168.43.191	173.194.66.116	TCP	50990 > 50300 [SYN] Seq=0 win=1024 Len=0
17988	0.000067	192.168.43.191	173.194.66.116	TCP	50990 > 5002 [SYN] Seq=0 win=1024 Len=0
17989	0.000070	192.168.43.191	173.194.66.116	TCP	50990 > 6002 [SYN] Seq=0 win=1024 Len=0
17990	0.000063	192.168.43.191	173.194.66.116	TCP	50990 > 9081 [SYN] Seq=0 win=1024 Len=0
18109	0.794487	192.168.43.191	173.194.66.116	TCP	50991 > 6788 [SYN] Seq=0 win=1024 Len=0
18110	0.000160	192.168.43.191	173.194.66.116	TCP	50990 > 15742 [SYN] Seq=0 win=1024 Len=0
18111	0.001761	192.168.43.191	173.194.66.116	TCP	50991 > 79 [SYN] Seq=0 win=1024 Len=0 M
18112	0.001911	192.168.43.191	173.194.66.116	TCP	50991 > 1805 [SYN] Seq=0 win=1024 Len=0

图 14.7

- 在 Wireshark 抓包主窗口的数据包列表区域中，可能会看见两类 TCP 数据包：第一类是攻击主机向受攻击主机（被扫描的主机）发出多个 SYN 标记位置 1，且源端口号不断变化，但目的端口号几乎一致的 TCP 扫描类数据包；第二类是受攻击主机分别回复的 RST 标记位置 1 的 TCP 数据包，如图 14.8 所示。之所以会存在这种情况，是因为受攻击主机的“身前”部署有防火墙（有可能是安装在受攻击主机 OS 里的软件防火墙，也有可能是安置在网络边

① 译者注：原文是“You will see many SYN packets without any response from the node under attack”。译文酌改。

界的硬件防火墙)，是防火墙代替受攻击主机回复了 RST 标记位置 1 的 TCP 数据包，以达到快速拆除 TCP 连接的目的^①。

No.	Time	Source	Destination	Protocol	Info
1284	0.178676	192.168.43.191	194.90.1.105	TCP	6910 > 445 [SYN] Seq=0 Win=0 Len=0 MSS=1460
1285	0.001176	192.168.43.191	194.90.1.105	TCP	6911 > 139 [SYN] Seq=0 Win=0 Len=0 MSS=1460
1295	0.058867	192.168.43.191	194.90.1.89	TCP	6907 > 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
1296	0.090965	192.168.43.191	194.90.1.89	TCP	6908 > 139 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
1297	0.007015	192.168.43.191	194.90.1.105	TCP	6904 > 445 [SYN] Seq=0 Win=0 Len=0 MSS=1460
1298	0.097377	194.90.1.105	192.168.43.191	TCP	445 > 6910 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1299	0.011296	194.90.1.105	192.168.43.191	TCP	139 > 6911 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1300	0.000453	194.90.1.89	192.168.43.191	TCP	445 > 6907 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1301	0.017741	194.90.1.105	192.168.43.191	TCP	445 > 6904 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1302	0.001921	194.90.1.89	192.168.43.191	TCP	139 > 6908 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
1303	0.004212	192.168.43.191	194.90.1.37	TCP	6866 > 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460

图14.8

- 在 Wireshark 抓包主窗口的数据包列表区域中，可能会看见单一源 IP 地址（攻击主机的 IP 地址）向单一目的 IP 地址（受攻击主机的 IP 地址）发出多个 SYN 标记位置 1，但目的端口号各不相同的 TCP 数据包，如图 14.9 所示。不过，受攻击主机会针对某些端口号（比如，图 14.9 中的 TCP 111、113、118、1421、1422、1423 端口）回复 RST 位置 1 的 TCP 数据包，快速拆除连接；会针对另外一些端口号（比如，图 14.9 中的 TCP 135、139 端口）回复 SYN 和 ACK 标记位同时置 1 的 TCP 数据包，同意与攻击主机继续建立 TCP 连接。前一种情况是因为受攻击主机未监听 TCP 111、113、118、1421、1422、1423 端口（或这些端口不在防火墙的开放范围之列）；后一种情况是因为受攻击主机监听了 TCP 135、139 端口，且防火墙开放了对这些目的端口的访问^②。

No.	Time	Source	Destination	Protocol	Info
100	0.001171	192.168.1.101	192.168.1.103	TCP	1410 > 111 [SYN] Seq=0 Win=0 Len=0 MSS=1460
187	0.000153	192.168.1.103	192.168.1.101	TCP	111 > 1416 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
188	0.000486	192.168.1.101	192.168.1.103	TCP	1417 > 113 [SYN] Seq=0 Win=0 Len=0 MSS=1460
189	0.000141	192.168.1.103	192.168.1.101	TCP	113 > 1417 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
190	0.001459	192.168.1.101	192.168.1.103	TCP	1418 > 118 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
191	0.000161	192.168.1.103	192.168.1.101	TCP	118 > 1418 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
192	0.001194	192.168.1.101	192.168.1.103	TCP	1419 > 135 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
193	0.000179	192.168.1.103	192.168.1.101	TCP	135 > 1419 [SYN, ACK] Seq=1 Ack=1 Win=0 Len=0
194	0.000024	192.168.1.101	192.168.1.103	TCP	1419 > 135 [ACK] Seq=1 Ack=1 Win=1/520 Len=0
195	0.000608	192.168.1.101	192.168.1.103	TCP	1420 > 139 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
196	0.000170	192.168.1.103	192.168.1.101	TCP	139 > 1420 [SYN, ACK] Seq=1 Ack=1 Win=1/520 Len=0
197	0.000020	192.168.1.101	192.168.1.103	TCP	1420 > 139 [ACK] Seq=1 Ack=1 Win=0 Len=0
198	0.000955	192.168.1.101	192.168.1.103	TCP	1421 > 156 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
199	0.000195	192.168.1.103	192.168.1.101	TCP	156 > 1421 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
200	0.001017	192.168.1.101	192.168.1.103	TCP	1422 > 179 [SYN] Seq=0 Win=0 Len=0 MSS=1460
201	0.000147	192.168.1.103	192.168.1.101	TCP	179 > 1422 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
202	0.000446	192.168.1.101	192.168.1.103	TCP	1423 > 371 [SYN] Seq=0 Win=16384 Len=0 MSS=1460
203	0.000139	192.168.1.103	192.168.1.101	TCP	371 > 1423 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
204	0.001253	192.168.1.101	192.168.1.103	TCP	1424 > 443 [SYN] Seq=0 Win=16384 Len=0 MSS=1460

图14.9

① 译者注：原文是“You will see many SYN packets when a TCP RST packet is sent as a response to each one of them. This is usually when you have a firewall on the device that is under attack or will be attacked”。译文酌改，如有不妥请指正。

② 译者注：原文是“You can also have consecutive SYN and RST packets. When there is a port number that is opened, you will see the complete SYN/SYN-ACK/ACK when the scanner opens connection to the victim. This is illustrated in the following screenshot”。译文酌改，如有不妥，请指正。

7. 仔细观察 Wireshark 抓包主窗口的数据包列表区域, 寻找异常的流量模式。比方说, 若 ICMP 数据包满屏皆是, 则极有可能存在 ICMP 扫描行为。此时, 需要留意 ICMP 数据包的具体类型 (ICMP 数据包根据 ICMP 头部中的类型字段值和代码字段值来分类, 由 ping 命令触发的 ICMP echo request 数据包只是 ICMP 数据包的一种, 其他常见的 ICMP 数据包还包括 ICMP echo reply 和 ICMP timestamp request 数据包等)、ICMP 数据包的源/目 IP 地址等。若发现 ICMP 数据包的源 IP 地址单一, 目的 IP 地址有规律可循 (比如, 从 194.90.15.1 按序递增至 194.90.15.254), 并且 ICMP 数据包的类型不是 ICMP echo request 就是 ICMP timestamp request, 则可认定是 ICMP 恶意扫描。

- 若怀疑网络中存在 ICMP 恶意扫描行为, 可先在 Wireshark 抓包主窗口的 Filter 输入栏内输入显示过滤表达式 icmp, 筛选出 ICMP 流量; 再到数据包列表区域中点击 Destination 列, 让 ICMP 数据包按目的 IP 地址排序。如此操作, 就可以很容易判断是否存在 ICMP 恶意扫描行为。
- 图 14.10 所示的 Wireshark 截屏展示了一次典型的 ICMP 恶意扫描行为。

No.	Time	Source	Destination	Protocol	Length	Info
21595	536.943169	10.0.0.1	194.90.15.152	ICMP	60	Timestamp request, id=0x2003, seq=0/0, ttl=41
10749	524.213695	10.0.0.1	194.90.15.153	ICMP	60	Echo (ping) request, id=0xe5cf, seq=0/0, ttl=46
11715	525.318336	10.0.0.1	194.90.15.153	ICMP	60	Echo (ping) request, id=0xec57, seq=0/0, ttl=44
25000	543.548790	10.0.0.1	194.90.15.153	ICMP	60	Timestamp request, id=0x12b7, seq=0/0, ttl=54
25112	544.107653	10.0.0.1	194.90.15.153	ICMP	60	Timestamp request, id=0xc371, seq=0/0, ttl=38
10750	524.213751	10.0.0.1	194.90.15.154	ICMP	60	Echo (ping) request, id=0xa, seq=0/0, ttl=41
11716	525.318361	10.0.0.1	194.90.15.154	ICMP	60	Echo (ping) request, id=0x0, seq=0/0, ttl=41
28770	551.905402	10.0.0.1	194.90.15.154	ICMP	60	Timestamp request, id=0x3, seq=0/0, ttl=41
28838	552.007147	10.0.0.1	194.90.15.154	ICMP	60	Timestamp request, id=0x9, seq=0/0, ttl=41
10751	524.213802	10.0.0.1	194.90.15.155	ICMP	60	Echo (ping) request, id=0x25d4, seq=0/0, ttl=43
11717	525.318387	10.0.0.1	194.90.15.155	ICMP	60	Echo (ping) request, id=0xfb5a, seq=0/0, ttl=49
31470	560.457512	10.0.0.1	194.90.15.155	ICMP	60	Timestamp request, id=0xb244, seq=0/0, ttl=42
31557	561.591617	10.0.0.1	194.90.15.155	ICMP	60	Timestamp request, id=0xd657, seq=0/0, ttl=45
10752	524.213836	10.0.0.1	194.90.15.156	ICMP	60	Echo (ping) request, id=0x289e, seq=0/0, ttl=59
11718	525.321236	10.0.0.1	194.90.15.156	ICMP	60	Echo (ping) request, id=0x289e, seq=0/0, ttl=59

图 14.10

8. 还有一种针对特定应用程序的扫描, 其手段是向某种应用层协议所使用 (监听) 的端口, 发送 TCP 扫描类数据包。如遭遇此类攻击, 在 Wireshark 抓包主窗口的数据包列表区域中会呈现以下特征^①。

- **NetBIOS 扫描**: 在 Wireshark 抓包主窗口的数据包列表区域中, 会看到大量发往 NetBIOS 相关端口的 TCP 扫描类数据包。
- **HTTP 扫描**: 在 Wireshark 抓包主窗口的数据包列表区域中, 会看到攻击者先行发出的目的端口号为 80、SYN 标记位置 1 的 TCP 扫描类数据包; 若得

^① 译者注: 原文是 “In the case of application scanning, you can have various types of scans”。译文酌改。

到了 HTTP 服务器的回应, 还将看到攻击者随后发出的 HTTP 请求数据包^①。

- **SMTP 扫描:** 在 Wireshark 抓包主窗口的数据包列表区域中, 会看见大量的目的端口号为 25, 且 SYN 标记位置 1 的 TCP 数据包。
- **SIP 扫描:** 在 Wireshark 抓包主窗口的数据包列表区域中, 会看到大量发往 5060 端口的 TCP 扫描类数据包。

若有针对其他应用层协议的扫描行为, 在 Wireshark 抓包主窗口的数据包列表区域中, 会看到大量发往相应目的端口的 TCP 扫描类数据包。

14.4.3 幕后原理

大多数扫描攻击所遵循的步骤是: 先进行 ARP 扫描, 再进行 ICMP 扫描, 最后进行 TCP 或 UDP 端口扫描。扫描攻击的原理非常简单, 如下所列。

- 攻击者通过扫描工具在 LAN 内执行 ARP 扫描 (以广播方式发出 ARP 请求数据包, 以期掌握同一 LAN 内其他设备的 IP 地址和 MAC 地址)。
- 攻击者通过扫描工具 (针对某一目的 IP 网络) 执行 ICMP 扫描 (发出 ICMP echo request 数据包, 以期获得该网络内主机的回应)。
- 若 ARP 扫描和 ICMP 扫描能得到其他设备的回应, 则攻击者会通过扫描工具向这台 (或这批) 设备的 IP 地址发出目的端口号不一的 TCP/UDP 扫描类数据包。若发现有设备监听并开放了某个 (或某些) TCP/UDP 端口, 则攻击者会继续发动 (与此端口号相对应的) 应用程序扫描。
- 执行应用程序扫描时, 攻击者会通过扫描工具发送包含特定命令 (内容) 的攻击数据包, 以此做进一步地入侵。

14.4.4 拾遗补缺

大多数新型入侵检测系统/入侵防护系统 (IDS/IPS) 都能识别和/或拦截 ICMP 扫描攻击、TCP 扫描攻击, 以及其他各种扫描攻击。这些攻击都是特征明显, 手法简单粗暴。只要在 Internet 链路上部署 IDS/IPS, 或接入部署了 IDS/IPS 的 ISP, 一般都不太可能遭受上述手法简单粗暴的扫描攻击。

IDS/IPS 按运作方式来分可分为以下两类。

^① 译者注: 原文是 “It looks for SYN requests to HTTP port 80 with HTTP requests later on”。译文如有不妥, 请指正。

- 基于 NetFlow/Jflow 的 IDS/IPS: 能识别出从多个来源涌入的扫描流量。此类 IDS/IPS 既可以直接拦截这些流量, 也可以基于流量的源 IP 地址, 调整 (路由设备的) 路由表, 阻断发源于 ISP 网络的扫描流量^①。
- 基于内容的 IDS/IPS: 先根据流量的内部特征来判断其是否属于恶意流量, 然后再决定是放行还是丢弃。

由于部署在网络边界的安全设备管不着发源于内网的攻击流量, 因此相对于外部攻击而言, 内部攻击要更为常见。14.6 节会对某些极为精巧的内部攻击展开深入探讨。

要想有效阻止发源于 Internet 的攻击, 要么在 Internet 边界部署 IDS/IPS, 要么就从部署了 IDS/IPS 的 ISP 申请 Internet 链路。要想阻止由内部发起的攻击, 除了要严格执行本单位制定的安全策略以外, 还得借助于安全防护软件 (比如, 杀毒软件和防火墙软件等)。

14.4.5 进阶阅读

上一小节提到的安全策略, 是指由各单位制定并需要严格执行的一套与安全有关的规章制度。与此有关的更多信息在 Internet 上到处可查, 现给出某些重要链接。

- http://www.cert.org/work/organizational_security.html
- <http://www.praxiom.com/iso-17799-4.htm>
- <http://www.sans.org/reading-room/whitepapers/policyissues/1331.php>
- <http://www.sans.org/security-resources/policies/>

14.5 发现DoS/DDoS攻击

发动拒绝服务 (DoS) / 分布式拒绝服务 (DDoS) 攻击的目的, 是要让用户访问不到本可以正常访问的某些网络资源。这些网络资源如下所示。

- 通信链路资源: 攻击者以生成“巨流”的方式发动 DoS/DDoS 攻击, 让垃圾流量充斥通信链路, 导致链路拥塞。
- 应用程序及网络服务资源 (Web 服务、Email 服务等): 攻击者以某种方式发动 DoS/DDoS 攻击, 让安装网络应用程序或提供网络服务的服务器宕机或资源耗尽, 无法处理用户发起的服务请求。

之前提及的各种扫描攻击, 最终可能会转化为 DoS/DDoS 攻击。若某些扫描攻击导致通信链路的拥塞或服务器处理能力的下降, 达到了拒绝用户访问的目的, 则这样的扫描攻击可被

^① 译者注: 原文是 “NetFlow/Jflow-based IDS/IPS that identifies massive traffic coming from several sources; they neutralize it by blocking it or changing the routing tables to disable these packets from getting to the ISP network”。

称为 DoS/DDoS 攻击。

本节会介绍几种常见的 DoS/DDoS 攻击，同时会讲解如何发现并阻止这些攻击。

14.5.1 准备工作

当某一网络资源（比如，某条通信链路、某种网络服务）突然无法访问或访问速度突然变得很慢时，这往往意味着有人正针对其发动 DoS/DDoS 攻击。

在确定了受攻击的网络资源之后，请设法将相关流量重定向至 Wireshark 主机，再启动 Wireshark 软件，开始抓包。本节会介绍几种常见的 DoS/DDoS 攻击，同时会细述各自的特征。

14.5.2 操作方法

请将 Wireshark 主机接入 LAN 交换机，在该交换机上开启端口镜像功能，设法将可能遭受 DoS/DDoS 攻击的网络资源所承受的流量重定向至 Wireshark 主机^①。一般而言，可能遭受 DoS/DDoS 攻击的网络资源不是一台莫名其妙变慢的服务器，就是一条负载猛然增高的链路。当然，它还可以是突然无法访问或访问速度突然变慢的其他任何网络资源。

1. 当一条通信链路（比如，一条 Internet 链路）突然变慢时，请将 Wireshark 主机接入网络，设法通过端口镜像技术，将那条链路所承载的流量重定向至 Wireshark 主机。

- 仔细观察 Wireshark 抓包主窗口数据包列表区域，对流量的源 IP 地址应格外注意。
- 图 14.11 所示为作者抓到的源于/发往一台服务器的流量^②。

No.	Time	Source	Destination	Protocol	Info
1	0.000000	1.1.164.98	94.23.71.12	TCP	44129 > 6000 [SYN] Seq=0 win=16384 Len=0
2	0.000011	1.1.164.99	94.23.71.12	TCP	44130 > 6000 [SYN] Seq=0 win=16384 Len=0
3	0.000011	1.1.164.100	94.23.71.12	TCP	44131 > 6000 [SYN] Seq=0 win=16384 Len=0
4	0.000011	1.1.164.101	94.23.71.12	TCP	44132 > 6000 [SYN] Seq=0 win=16384 Len=0
5	0.000012	1.1.164.102	94.23.71.12	TCP	44133 > 6000 [SYN] Seq=0 win=16384 Len=0
6	0.000011	1.1.164.103	94.23.71.12	TCP	44134 > 6000 [SYN] Seq=0 win=16384 Len=0
7	0.000011	1.1.164.104	94.23.71.12	TCP	44135 > 6000 [SYN] Seq=0 win=16384 Len=0
8	0.000011	1.1.164.105	94.23.71.12	TCP	44136 > 6000 [SYN] Seq=0 win=16384 Len=0
9	0.000011	1.1.164.106	94.23.71.12	TCP	44137 > 6000 [SYN] Seq=0 win=16384 Len=0
10	0.000012	1.1.164.107	94.23.71.12	TCP	44138 > 6000 [SYN] Seq=0 win=16384 Len=0
11	0.000011	1.1.164.108	94.23.71.12	TCP	44139 > 6000 [SYN] Seq=0 win=16384 Len=0
12	0.000011	1.1.164.109	94.23.71.12	TCP	44140 > 6000 [SYN] Seq=0 win=16384 Len=0

图14.11

- 根据源 IP 地址对抓到的数据包做升序排列后，可以看出，对于 Wireshark 抓

① 译者注：原文是“Connect Wireshark to the network with port mirror to the port of the resource that you suspect is exposed to DoS/DDoS”。

② 译者注：原文是“I’ve port mirrored the server, and this is what I got”。作者前面还在说通信链路，没过一会便扯上了服务器，不清楚作者葫芦里的药，译文直译。

到的发往公网 IP 地址 **94.23.71.12** 的所有数据包, 其源 IP 地址的前三个字节一模一样, 而最后一个字节居然在数字上是连续的。



注意

通过图 14.11 还可以看出, 数据包之间的时间间隔全都介于 11~12 微秒之间 (作者在 **View** 菜单的 **Time Display Format** 菜单项中点选的是 **Seconds Since Previous Displayed Packet** 子菜单项)。要是读者抓到了以这样的频率发送的 SYN 标记位置 1 的 TCP 数据包, 请务必多加小心, 这往往就意味着 DoS/DDoS 攻击。

- 由于所抓数据包的源 IP 地址非常怪异, 因此需要借助于 **Statistics** 菜单中 **Conversations** 工具, 来看看它们的 MAC 地址到底是什么, 如图 14.12 所示。
- 由图 14.12 可知, 与那批源 IP 地址相对应的 MAC 地址只有一个, 也就是作者监控的那台服务器的 MAC 地址。

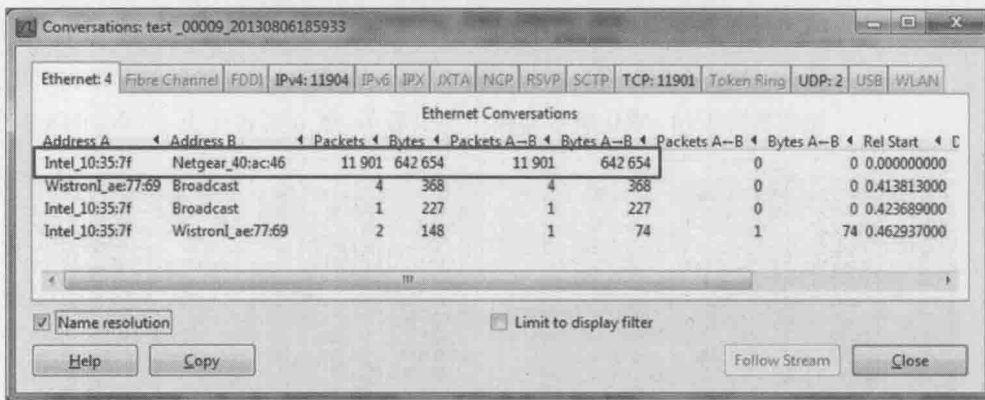


图 14.12



注意

在抓到了被怀疑是起 TCP SYN 扫描作用的数据包时, 应仔细核查数据包的源 IP 地址和源 MAC 地址。安装了某些操作系统的主机在感染了蠕虫病毒之后, 可能会批量生成源 IP 地址为非本机所设 IP 地址的数据包, 但这些数据包的源 MAC 地址却一定是其网卡的 MAC 地址^①。

2. TCP SYN 扫描也有可能从单个来源发起, 也就是说, 用作于 TCP SYN 扫描的数据包的源 IP 地址只有一个, 如图 14.13 所示。这种数据包的特征有二: TCP 头部中 SYN 标记位会置 1; 目的 TCP 端口号五花八门。此时, 使用 Wireshark 抓包, 结果将会是以下三种情况中的一种。

- 只能抓到用于 TCP SYN 扫描的数据包。

① 译者注: 原文是 “Check for SYN scans, and verify which IP and MAC addresses they are coming from. It can be that a worm is generating source addresses that are not the addresses of the host”。

- 除了起 TCP SYN 扫描作用的数据包之外，还能抓到与其相对应的 TCP 重置（reset）数据包。
- 既能抓到起 TCP SYN 扫描作用的数据包，也能抓到受攻击主机与攻击主机间建立 TCP 三次握手的其余 TCP 数据包。

3. 以下所列为上述 TCP SYN 扫描攻击可能造成的各种后果。

- 要是只能抓到起 TCP SYN 扫描作用的数据包，那么受攻击主机将一切无恙。但若抓到了受攻击主机发出的与攻击主机建立 TCP 三次握手的数据包，受攻击服务器则岌岌可危。
- 只要攻击主机与受攻击主机之间建立了过多的 TCP 连接（SYN/SYN-ACK/ACK）或半开连接（(SYN/SYN-ACK)，后者就会因为资源消耗过大，而变得越来越慢。
- 图 14.13 所示为一次典型的 TCP SYN 扫描攻击。像这样的扫描攻击只要能使通信链路拥堵，或让服务器停止提供服务，就立刻转化为了 DoS/DDoS 攻击。

No.	Time	Source	Destination	Protocol	Info
55372	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 1082 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55373	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 15003 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55374	0.000034	10.0.0.103	10.0.0.10	TCP	33928 > 6567 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55375	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 458 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55376	0.000026	10.0.0.103	10.0.0.10	TCP	33928 > 8383 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55377	0.000035	10.0.0.103	10.0.0.10	TCP	33928 > 2100 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55378	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 1721 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55379	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 8994 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55380	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 6699 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55381	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 10616 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55382	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 2381 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55383	0.000024	10.0.0.103	10.0.0.10	TCP	33928 > 55555 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55384	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 8193 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55385	0.000026	10.0.0.103	10.0.0.10	TCP	33928 > 10001 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
55386	0.000025	10.0.0.103	10.0.0.10	TCP	33928 > 5904 [SYN] Seq=0 Win=1024 Len=0 MSS=1460

图14.13

14.5.3 幕后原理

拒绝服务（DoS）攻击的目的是让（原本可以访问的）网络服务不可访问。DoS 攻击的实施手段是设法消耗受攻击的网络服务的资源（比如，网络服务所驻留的主机的 CPU、内存资源等），使其无法提供正常的服务。

DoS 攻击的源头一般只有一个，而 DDoS 攻击的源头则可以有很多。

14.5.4 拾遗补缺

DoS/DDoS 攻击有时很难觉察，因为攻击者在发动攻击时会将相应的攻击流量“伪装”为正常流量。举例如下。

1. 可以伪装为由网管系统发出的用来发现网络设备的 ICMP echo request 数据包 (ping 包)。
2. 可以伪装为可被 Web 服务器正常接收的包含 GET 方法的 HTTP 请求数据包。
3. 可以伪装为 SNMP GET 请求数据包。

因此,要想发现 DoS/DDoS 攻击,不但要查明可疑数据包的来源,还得关注其数量,甚至还需知其内在。图 14.14 所示的 Wireshark 截屏显示了作者用 **Follow TCP Stream** 右键菜单项功能,关注的一条指定的 TCP 数据流。

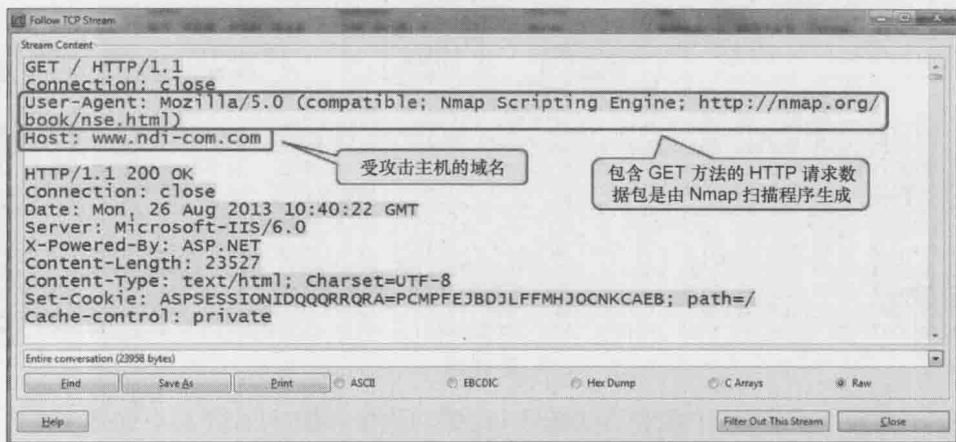


图14.14

14.6 发现高级TCP攻击

高级 TCP 攻击是指攻击者发出精心打造的 TCP 数据包,让受攻击设备不知如何处理,或者在安全设备那里蒙混过关。虽然部署在当今网络中的新型防火墙能够识别并阻止此类攻击,但作者认为仍有必要对其做简要介绍。

14.6.1 准备工作

作者只要在一个新的网络环境中执行排障任务,通常都会把安装了 Wireshark 的笔记本接入该网络,观察其所承载的流量。首先,作者会把笔记本连接到 LAN 交换机,观察抓到的广播流量。然后,会启用端口镜像功能,将交换机上连接关键服务器或重要通信链路的端口的流量重定向至 Wireshark 主机,仔细观察抓到的流量^①。

^① 译者注:后面还有一句“To look for unusual traffic, port mirror communications links and central servers, and check for unusual traffic patterns”,直译为“要发现异常流量,端口镜像通信链路及中心服务器,严查异常流量模式”。作者貌似不擅写作,还偏要那么多废话,此句不放入正文。

14.6.2 操作方法

在分析 Wireshark 抓到的流量时,若发现 TCP 流量具备以下特征,则很可能是有人在发动高级 TCP 攻击。

- 若发现了大量目的 IP 地址相同,目的端口号千奇百怪,且 ACK 标记位置 1 TCP 数据包(如图 14.15 所示),则可认定有人在发动名为 ACK 扫描的高级 TCP 攻击。这种攻击的目的是破坏已经建立的 TCP 连接。

No.	Time	Source	Destination	Protocol	Info
252	0.000032	192.168.43.191	10.0.0.138	TCP	51752 > 1503 [ACK] Seq=1 Ack=1 win=1024 Len=0
253	0.000023	192.168.43.191	10.0.0.138	TCP	51752 > 3128 [ACK] Seq=1 Ack=1 win=1024 Len=0
254	0.000023	192.168.43.191	10.0.0.138	TCP	51752 > 19315 [ACK] Seq=1 Ack=1 win=1024 Len=0
255	0.000033	192.168.43.191	10.0.0.138	TCP	51752 > 1580 [ACK] Seq=1 Ack=1 win=1024 Len=0
256	0.000026	192.168.43.191	10.0.0.138	TCP	51752 > 1066 [ACK] Seq=1 Ack=1 win=1024 Len=0
257	0.000025	192.168.43.191	10.0.0.138	TCP	51751 > 9595 [ACK] Seq=1 Ack=1 win=1024 Len=0
258	0.001317	192.168.43.191	10.0.0.138	TCP	51752 > 212 [ACK] Seq=1 Ack=1 win=1024 Len=0
259	0.000084	192.168.43.191	10.0.0.138	TCP	51752 > 512 [ACK] Seq=1 Ack=1 win=1024 Len=0
260	0.000027	192.168.43.191	10.0.0.138	TCP	51752 > 10629 [ACK] Seq=1 Ack=1 win=1024 Len=0
261	0.000027	192.168.43.191	10.0.0.138	TCP	51752 > 40193 [ACK] Seq=1 Ack=1 win=1024 Len=0
262	0.000023	192.168.43.191	10.0.0.138	TCP	51752 > 1053 [ACK] Seq=1 Ack=1 win=1024 Len=0

图 14.15

- 若发现了大量标记位置位方式怪异(比如,URG、FIN 和 RST 标记位同时置 1,或者 SYN 和 FIN 标记位同时置 1)的 TCP 数据包,则可认定有人在发动高级 TCP 攻击。

图 14.16 所示为有人在执行名为 Xmas 扫描攻击时, Wireshark 抓到的 URG、FIN 和 RST 标记位同时置 1 的 TCP 数据包。

No.	Time	Source	Destination	Protocol	Info
2190	0.026411	192.168.43.191	10.0.0.138	TCP	51889 > 1 [ACK] Seq=1 Ack=1 win=33554432 Len=0 WS
2191	0.025524	192.168.43.191	10.0.0.138	TCP	51890 > 1 [FIN, PSH, URG] Seq=1 win=2147450880 Ur
2193	0.028932	10.0.0.138	192.168.43.191	TCP	1 > 51889 [RST] Seq=1 win=0 Len=0
2195	0.048204	192.168.43.191	10.0.0.138	UDP	Source port: 51930 Destination port: 30282
2196	0.029788	192.168.43.191	10.0.0.138	TCP	51888 > 1 [SYN] Seq=0 win=31337 Len=0 WS=1024 MSS
2197	0.024198	192.168.43.191	10.0.0.138	TCP	51890 > 1 [FIN, PSH, URG] Seq=1 win=2147450880 Ur
2200	0.078040	192.168.43.191	10.0.0.138	UDP	Source port: 51930 Destination port: 30282
2201	0.026694	192.168.43.191	10.0.0.138	TCP	51888 > 1 [SYN] Seq=0 win=31337 Len=0 WS=1024 MSS
2202	0.024286	192.168.43.191	10.0.0.138	TCP	51890 > 1 [FIN, PSH, URG] Seq=1 win=2147450880 Ur
2205	0.080907	192.168.43.191	10.0.0.138	UDP	Source port: 51930 Destination port: 30282
2206	0.026051	192.168.43.191	10.0.0.138	TCP	51888 > 1 [SYN] Seq=0 win=31337 Len=0 WS=1024 MSS
2207	0.025937	192.168.43.191	10.0.0.138	TCP	51890 > 1 [FIN, PSH, URG] Seq=1 win=2147450880 Ur
2208	0.173624	192.168.43.191	10.0.0.138	TCP	21462 > 80 [ACK] Seq=1 Ack=1 win=2401 Len=0
2209	0.000169	192.168.43.191	10.0.0.138	TCP	21463 > 80 [ACK] Seq=1 Ack=1 win=18085 Len=0

图 14.16

图 14.17 所示为所有标记位全都置 0 的 TCP 数据包。在发动名为空扫描(Null scan)的高级 TCP 攻击时,便会生成这种标记位置位方式更为怪异的 TCP 数据包。

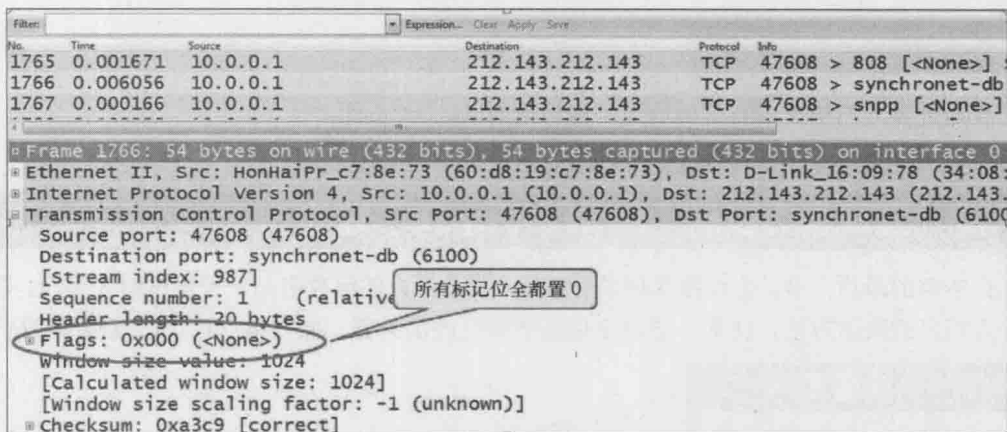


图14.17

- 若发现了大量目的 IP 地址相同，目的端口号千奇百怪，且 ACK 和 FIN 标记位同时置 1 的 TCP 数据包（如图 14.18 所示），则可认定有人在发动名为 FIN-ACK 扫描的高级 TCP 攻击。这种攻击的目的是破坏已经建立的 TCP 连接或对目的主机进行流量泛洪。

No.	Time	Source	Destination	Protocol	Info
1133	0.092435	10.0.0.1	212.143.212.143	TCP	50948 > 545 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1134	0.000199	10.0.0.1	212.143.212.143	TCP	50948 > 2005 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1135	0.000156	10.0.0.1	212.143.212.143	TCP	50948 > 57294 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1136	0.018944	10.0.0.1	212.143.212.143	TCP	50948 > 1455 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1137	0.000237	10.0.0.1	212.143.212.143	TCP	50948 > 9040 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1138	0.000125	10.0.0.1	212.143.212.143	TCP	50948 > 25734 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1139	0.000178	10.0.0.1	212.143.212.143	TCP	50948 > 20221 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1140	0.000108	10.0.0.1	212.143.212.143	TCP	50948 > 11110 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0
1141	0.000070	10.0.0.1	212.143.212.143	TCP	50948 > 45100 [FIN, ACK] Seq=1 Ack=1 Win=1024 Len=0

图14.18

14.6.3 幕后原理

上面提到的种种高级 TCP 攻击，无非遵循两个思路：发出大量 RST 或 FIN 位置 1（ACK 位置 0 或置 1 均可）的 TCP 数据包，扫描目标主机可能监听的各 TCP 端口，目的是破坏其已经（通过各监听端口）建立的 TCP 连接；发出大量标记位置方式怪异的 TCP 数据包，让目标主机忙于处理，以达到拒绝服务的目的。

新型防火墙或 IDS/IPS 能识别并阻挡绝大多数所谓的高级 TCP 攻击。

14.6.4 拾遗补缺

在怀疑遭受攻击，动用 Wireshark 进行抓包分析时，虽可利用显示过滤器来匹配各类攻击流量的特征，但最好还是彻查抓到的所有流量，从中寻找异常流量模式。

14.6.5 进阶阅读

欲知更多与扫描攻击有关的信息，请访问 Nmap.org 页面：

<http://nmap.org/book/man-port-scanning-techniques.html>

14.7 发现暴力破解（brute-force）攻击

在本章的最后一节，会介绍几种暴力破解攻击。暴力破解攻击是一种以试错为手段，窃取受害人信息的攻击方法，比如，尝试定位某些单位的服务器、探索用户目录以及破解密码^①。

14.7.1 准备工作

不论是探测攻击，还是破解攻击，一般都不会在网络中掀起大风大浪（比如，不会让流量大起大落），因此其攻击行为只有 IDS 等网络安全防护设备才能觉察，或者就得靠网管人员的直觉来发现类似的网络入侵了。

14.7.2 操作方法

若怀疑有人在网络中实施暴力破解攻击，请按以下步骤来加以定位。

1. 在 LAN 交换机上启用端口镜像功能，设法将相关端口（连接了被怀疑是遭受暴力破解攻击的服务器的端口）的流量重定向至 Wireshark 主机。
2. 要想发现 DNS 探测攻击，需要在抓包文件中筛选出 DNS 流量，仔细检查询问本单位主机（服务器）域名的 DNS 查询数据包^②。图 14.19 所示的 Wireshark 抓包文件展示了一次针对 icomm.com 域的 DNS 探测攻击。不难发现，攻击主机（IP 地址 10.0.0.1）正在执行 DNS 探测攻击，该主机向 DNS 服务器（10.0.0.138）发出了诸多 DNS 查询数据包，企图探测 icomm.com 域旗下的多台主机的 IPv4 和 IPv6 地址（查询 DNS A 记录和 AAAA 记录）。
 - 由于该攻击主机所探测的大部分主机的域名都不存在，因此 Wireshark 抓到了很多身背 no such name 字样的 DNS 响应数据包（详见图 14.19 中的 2、3、4、5）。
 - 需要注意的是，如在 Wireshark 抓包文件中发现有身背 no such name 字样的

① 译者注：原文是“A brute-force attack is a trial-and-error method used to obtain information from the victim, for example, trying to find organizational servers, user directories, and crack passwords”。作者说的“brute-force attack”，其实应该是探测攻击和破解攻击的结合体，但本节作者所列举的“brute-force attack”都是探测攻击。在面的译文中，译者会根据上下文将“brute-force attack”译为“探测攻击”或“破解攻击”。

② 译者注：原文是“For DNS brute-force attacks, look for DNS queries that are asking for common names under your domain”。作者的写作思路，译者实在不敢苟同，译文酌改。

DNS 响应数据包, 并不意味着一定有人在执行 DNS 探测攻击。此时, 需要从身背 no such name 字样的 DNS 响应数据包的数量, 以及与之对应的 DNS 查询数据包的源 IP 地址来判断是否存在 DNS 探测攻击。

No.	Time	Source	Destination	Protocol	Info
7749	0.127587	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 AAAA sip.icomm.com
7750	0.023064	10.0.0.138	10.0.0.1	DNS	Standard query response 0x0001
7751	0.128110	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 A dns.icomm.com
7752	0.026680	10.0.0.138	10.0.0.1	DNS	Standard query response 0x0001 A 81.199.199.199
7755	0.124379	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 AAAA dns.icomm.com
7756	0.023907	10.0.0.138	10.0.0.1	DNS	Standard query response 0x0001
7757	0.127113	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 A ns2.icomm.com
7758	0.023341	10.0.0.138	10.0.0.1	DNS	Standard query response 0x0001 No such name
7759	0.005137	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 AAAA corp.icomm.com
7760	0.000190	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 AAAA whois.icomm.com
7761	0.000640	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 AAAA ns2.icomm.com
7762	0.001602	10.0.0.138	10.0.0.1	DNS	Standard query 0x0001 AAAA ns2.icomm.com
7763	0.023563	10.0.0.138	10.0.0.1	DNS	Standard query response 0x0001 No such name
7764	0.088002	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 A intranet.icomm.com
7765	0.024316	10.0.0.138	10.0.0.1	DNS	Standard query response 0x0001 No such name
7766	0.134785	10.0.0.1	10.0.0.138	DNS	Standard query 0x0001 AAAA intranet.icomm.com
7767	0.023727	10.0.0.138	10.0.0.1	DNS	Standard query response 0x0001 No such name

图 14.19

3. 要想发现 HTTP 探测攻击 (发动 HTTP 探测攻击的目的, 是要探测出 Web 服务器上有哪些资源), 需要仔细检查 Wireshark 抓到的相关 HTTP 流量^①。

- 要仔细检查 HTTP 请求数据包的内容, 看看里面是否有攻击者留下的足迹, 如图 14.20 所示。

No.	Time	Source	Destination	Protocol	Info
8991	0.038179	10.0.0.1	81.218.230.244	HTTP	GET /level/15/exec/~/.configure/http HTTP/1.1
9010	0.037638	10.0.0.1	81.218.230.244	HTTP	OPTIONS / HTTP/1.1
9017	0.001234	10.0.0.1	81.218.230.244	HTTP	GET /wp-config.php~ HTTP/1.1
9019	0.004997	81.218.230.244	10.0.0.1	HTTP	HTTP/1.1 404 Not Found (text/html)
9021	0.004005	10.0.0.1	81.218.230.244	HTTP	GET /level/15/exec/~ HTTP/1.1

#	Frame 9017: 218 bytes on wire (1744 bits), 218 bytes captured (1744 bits) on interface 0
#	Ethernet II, Src: HonHaiPr_c7:8e:73 (60:d8:19:c7:8e:73), Dst: D-Link_16:09:78 (34:08:04:16:09:78)
#	Internet Protocol Version 4, Src: 10.0.0.1 (10.0.0.1), Dst: 81.218.230.244 (81.218.230.244)
#	Transmission Control Protocol, Src Port: 42573 (42573), Dst Port: http (80), Seq: 1, Ack: 1, Len: 164
#	Hypertext Transfer Protocol
#	GET /wp-config.php~ HTTP/1.1\r\n
#	Connection: close\r\n
#	User-Agent: Mozilla/5.0 (compatible; Nmap Scripting Engine; http://nmap.org/book/nse.html)\r\n
#	Host: www.ndi.co.il\r\n
#	\r\n
#	[Full request URI: http://www.ndi.co.il/wp-config.php~]

图 14.20

- 应关注身背 HTTP 错误码 4xx 的 HTTP 应答数据包的数量。为此, 请先在 Wireshark 抓包主窗口中点击 **Statistics** 菜单的 HTTP 菜单项中的 **Packet Counter** 子菜单项, 再在弹出的 HTTP/Packet Counter Stats Tree 窗口中输入 IP, 点 **Create Stat** 按钮, 如图 14.21 所示。在 HTTP/Packet Counter with filter 窗口中点击 **HTTP Response Packets** 之前的 “+” 号, 要是发现 Wireshark 抓到了太多含 HTTP 状态码 4xx 的 HTTP 响应数据包, 则表示有人在执行 HTTP 扫描攻击。

① 译者注: 原文是 “Another brute-force attack to watch out for is HTTP trying to find resources on the server”。

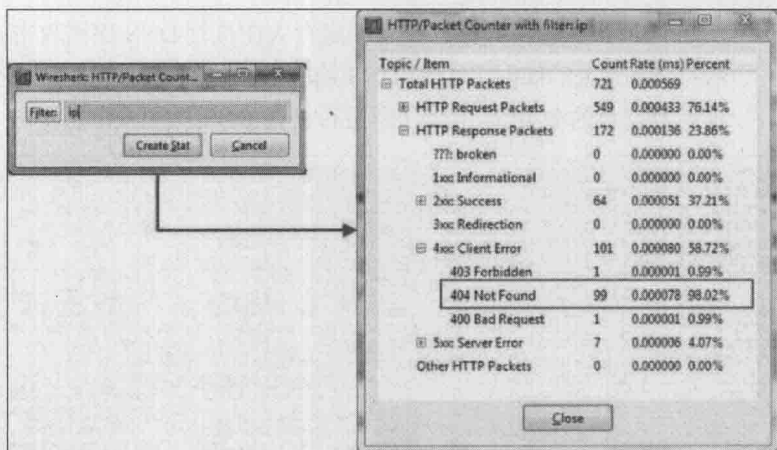


图14.21

14.7.3 幕后原理

对目标网络进行探测，向其发送探测类数据包，是暴力破解攻击的第一步。发送探测类数据包目的，当然是希望目标网络内的主机予以回应。只要网络安全措施能落实到位，加之那些探测又漫无目的，当黑客探测网络时，表现出的征兆包括：在 Wireshark 抓包文件中会发现大量身背 Not Found 或 forbidden 字样，以及包含各种错误状态码的（HTTP、SMTP、SIP）数据包。

14.7.4 拾遗补缺

要想从 Wireshark 抓包文件中筛选出含错误状态码的 HTTP 应答数据包，显示过滤器的写法是 `http.response.code >=400`^①。对于某些其他协议（比如 SIP），其协议数据包也会携带类似于 HTTP 协议的错误状态码，同样可以利用相似的显示过滤器，基于错误状态码对数据包进行筛选。要想从抓包文件中筛选出由某些知名扫描工具生成的攻击流量，请先在 Wireshark 抓包主窗口中点击 **Edit** 菜单中的 **Find Packet** 菜单项，再到弹出的 Find Packet 窗口中选择 **String** 和 **Packet bytes** 单选按钮，朝 Filter 输入栏里填入知名扫描工具的名称，最后点击 **Find** 按钮即可。图 14.22 所示为如何从 Wireshark 抓包文件中筛选出由 Nmap 生成的数据包。Nmap 是众多扫描工具中知名度最高的一个。

朝 Filter 输入栏里填入 `nmap.org` ①，点击 Find 按钮，Wireshark 抓包主窗口的数据包列表区域就只会显示由 Nmap 扫描工具生成的数据包（即包含关键字 `nmap.org` 的数据包）了，如图 14.23 所示。

① 译者注：原文是 “For discovering HTTP error codes, configure the display filter `http.response.code >=400`”。译文酌改。

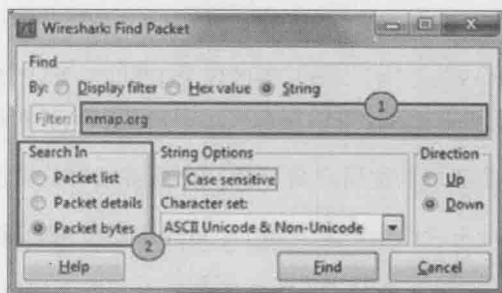


图 14.22

暴力破解攻击到了最后一步，攻击者所要做的就是企图以猜测密码的方式，来攻陷服务器或网络设备。

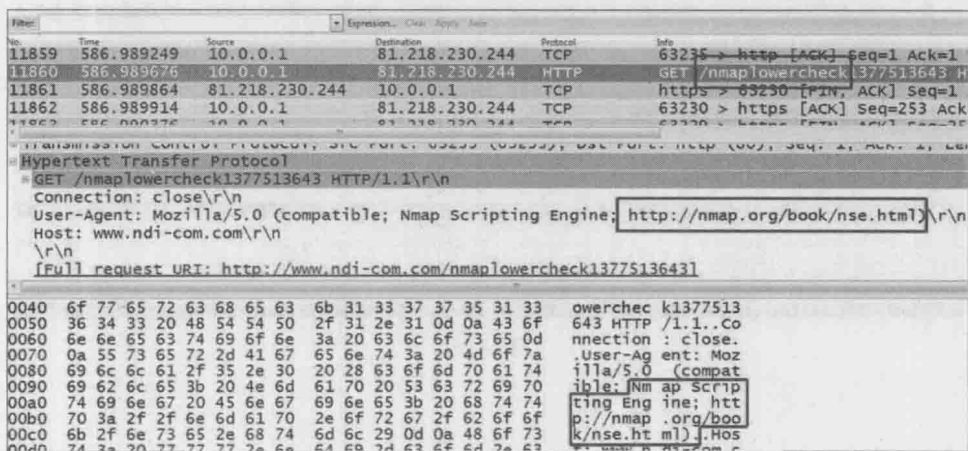


图 14.23

图 14.24 所示为攻击者尝试攻陷一台受到良好保护的 FTP 服务器时的情况。

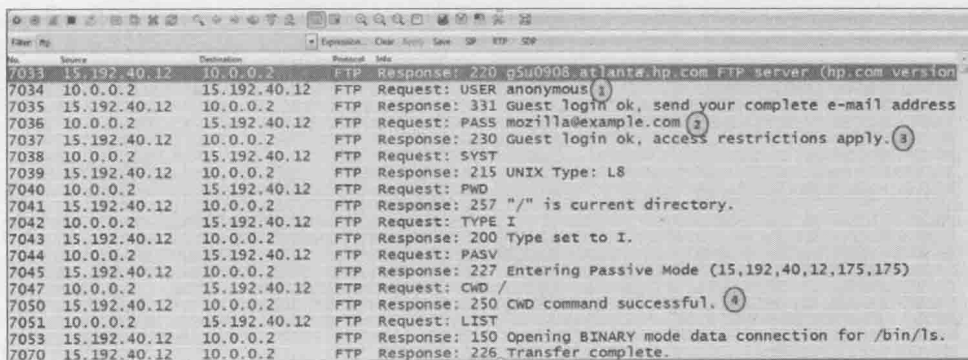


图 14.24

- 1. 由于所要攻陷的是 FTP 服务器，因此攻击者先以匿名的方式（使用用户名 anonymous 和密码 mozilla@example.com）尝试登录①和②。FTP 服务器接受了攻击者的登录请求③。攻击者登录之后就可以执行匿名用户获准执时的命令④。
- 2. 图 14.25 所示为攻击者以其他用户名登录 FTP 服务器时的情况。
- 3. 由图 14.25 可知，攻击者以用户名 root①、admin②和 administrator③分别尝试登录。

因为攻击者未能提供正确的密码，所以 FTP 服务器将其拒之门外，同时向其发出身背 TCP Zero-Window 字样，包含 you could at least say goodbye 内容的 FTP 数据包。

No.	Source	Destination	Protocol	Info
11054	15.240.238.51	10.0.0.2	FTP	Response: 530 Login incorrect.
11059	15.240.238.51	10.0.0.2	FTP	[TCP ZeroWindow] Response: 221 You could at least say goodbye.
11064	10.0.0.2	15.240.238.51	FTP	Request: USER root
11065	15.240.238.51	10.0.0.2	FTP	Response: 220 g9u0201.houston.hp.com FTP server (hp.com version
11066	10.0.0.2	15.240.238.51	FTP	Request: PASS
11068	15.240.238.51	10.0.0.2	FTP	Response: 331 Password required for root. ①
11070	15.240.238.51	10.0.0.2	FTP	Response: 530 Login incorrect.
11074	15.240.238.51	10.0.0.2	FTP	[TCP ZeroWindow] Response: 221 You could at least say goodbye.
11079	10.0.0.2	15.240.238.51	FTP	Request: USER admin
11080	15.240.238.51	10.0.0.2	FTP	Response: 220 g9u0201.houston.hp.com FTP server (hp.com version
11081	10.0.0.2	15.240.238.51	FTP	Request: PASS
11084	15.240.238.51	10.0.0.2	FTP	Response: 331 Password required for admin. ②
11086	15.240.238.51	10.0.0.2	FTP	Response: 530 Login incorrect.
11090	15.240.238.51	10.0.0.2	FTP	[TCP ZeroWindow] Response: 221 You could at least say goodbye.
11095	10.0.0.2	15.240.238.51	FTP	Request: USER administrator
11096	15.240.238.51	10.0.0.2	FTP	Response: 220 g9u0201.houston.hp.com FTP server (hp.com version
11097	10.0.0.2	15.240.238.51	FTP	Request: PASS
11099	15.240.238.51	10.0.0.2	FTP	Response: 331 Password required for administrator. ③
11101	15.240.238.51	10.0.0.2	FTP	Response: 530 Login incorrect.
11105	15.240.238.51	10.0.0.2	FTP	[TCP ZeroWindow] Response: 221 You could at least say goodbye.

图14.25

附录

链接、工具及阅读资料

在本附录里，作者会提供某些有用的链接，读者可从中获取更多与 Wireshark 有关的信息、学习资源，以及相关软件信息。

与 Wireshark 软件有关的链接

Wireshark 官网主页的链接是 <http://www.wireshark.org>，其中包含下述页面。

- Wireshark 软件下载页面: <http://www.wireshark.org/download.html>。
- Wireshark 软件学习页面: <http://www.wireshark.org/docs/>。
- 在 Wireshark 官网主页面的右上角有一个名为 **Enhance** 的区域，点击该区域便会被链接到 Riverbed 公司网站的产品介绍页面: <http://www.riverbed.com/products/performance-management-control/network-performance-management/>。Riverbed 公司（已被 CACE Technologies 公司收购）是 Wireshark 软件的主赞助商，开发并销售商用的 Wireshark 附加组件。
- Wireshark 属于开源软件，其开发资源的链接为: http://www.wireshark.org/docs/wsdg_html_chunked/ 或 <http://www.wireshark.org/develop.html>。

下面再给出一些有用的链接。

- Wireshark 软件的图形用户界面是用 GTK+或 GIMP 工具包来创建的，欲了解这一开源工具包，请访问 GTK+项目的 Web 主页面: <http://www.gtk.org/>。
- 欲了解与 WinPCaP（Windows 平台的抓包驱动程序）有关的信息，请访问 www.win

pcap.org。欲了解与远程抓包有关的信息，请访问 http://www.winpcap.org/docs/docs_40_2/html/group__remote.html#Config。

与 tcpdump 软件有关的链接

tcpdump 是一款基于 UNIX 的自由软件，通过 UNIX/Linux 命令行来运行。以下所列为与这款软件有关的有用链接。

- tcpdump 官网: <http://www.tcpdump.org/>。
- Windows 版本的 tcpdump (windump): <http://www.winpcap.org/windump/default.htm>。
- 容易上手的教程: <http://danielmiessler.com/study/tcpdump/>。
- tcpdump 官网主页: http://www.tcpdump.org/tcpdump_man.html。

用 Wireshark 也能打开 tcpdump 抓包文件。

其他工具

Wireshark 虽为当今市面上最常见的网络分析工具，但作者也常用某些其他网络分析工具来排除网络故障。在介绍这些工具之前，先来回顾一下作者在这么多年里处理过的一个最为滑稽的网络故障。故障的原因虽然比较简单，但却能给人以深刻的启迪。有家大医院药房的网络出了故障。药房里的药剂师都配有无无线终端，并根据无线终端上的指令来配药，然后再将配好的药送达医院的各个科室。故障的表象是：所有药剂师的无线网络终端的网速都非常之慢。医院请了一家系统集成公司来帮助排除故障，那家公司派出大队人马，并弄来了当时市场上能买到的所有网络排障装备，包括：Wireshark、Sniffer、无线网络分析仪、频谱分析仪等各种各样的软、硬件工具。当作者赶到那儿，看到他们乱得像一堆无头苍蝇时，叹了口气，对他们说：“你们什么工具都带齐了，就是没有带脑子。”其实那些人只要动动脑子，就应该能够发现是药房与医院主机房之间一条 50 米长的五类双绞线的 RJ45 接口出了毛病。

上面这个故事的结论是：工具总归是工具，身为网络技术人员，要是没有网络理论知识，也就不知道如何使用工具。对那些人来讲，有没有工具其实差别都不大。在本节，作者将会介绍一些其他网络排障工具，以及各自的用途。



本书前文提及的以及本节所要介绍的网络工具，作者都已使用多年，其中的某些是自由软件，而另外一些则为商业产品。值得一提的是，作者是基于个人使用体验来介绍这些工具的，并不牵涉商业或其他任何方面的利益。

SNMP 工具

排除网络故障时，作者最先使用的工具一般都是 SNMP 工具。在众多 SNMP 工具中，有些具有较强的生成网络拓扑结构图的功能，有些具备良好的信息统计功能，还有一些则具有完善的日志及事件记录功能。

要是只想监控（网络设备的）SNMP 计数器，只需使用简单的 MIB browser 工具和图形工具，这些工具大多为自由软件，如下所列^①。

厂商	软件名称	下载链接	特色	使用许可
Manage engine	MibBrowser	http://www.manageengine.com/products/mibbrowserfree tool/	配置门槛低，界面友好	自由软件
开源软件	MRTG	http://oss.oetiker.ch/mrtg/	安装及配置的门槛较高。适合作为长期的信息统计工具来用。MRTG 也是许多 ISP 提供给用户们查看网络流量状况的软件	GNU 通用公共许可证
SolarWinds	Network device monitor	http://www.solarwinds.com/products/freetools/network device-monitor/	SolarWinds 公司是业界领先的网管工具开发厂商，虽然其拳头产品都是商业软件，但也同样出品许多自由软件，Network device monitor 工具就属于自由软件	自由软件
SolarWinds	Engineer's toolset (网工工具箱)	http://www.solarwinds.com/engineers-tool set.aspx	集各种监控、故障诊断、配置、SNMP，以及 Cisco 设备管理等 60 余种工具于一体	分免费版和商业版，前者功能有限，后者功能齐备

SNMP 平台

SNMP 平台是指可控制各种 SNMP 工具的中央控制台软件，通过这个控制台软件可以：生成并显示网络拓扑结构图；收集各种 SNMP 统计信息，根据收集到的信息，按指定条件生成相关报表；收集各种 SNMP 事件，按事件的严重程度或其他条件来加以呈现。

以下所列常用的 SNMP 平台。

厂商	软件名称	下载链接	特色	使用许可
Castlerock Computing	SNMPc	http://www.castlerock.com/	界面友好，作者使用其的时间已经超过 10 年。该 SNMP 网管平台使用简单，非常适用于网络故障诊断	商业软件

^① 译者注：原文是“First, in order to just monitor SNMP counters, you can use simple free MIB browsers and graphical tools such as”。译文按字面意思直译。

续表

厂商	软件名称	下载链接	特色	使用许可
SolarWinds	各种网管软件	http://www.solarwinds.com/network-management-software.aspx	SolarWinds 公司出品各种各样的网管软件, 功能从网络监控, 到配置管理, 再到流量分析, 不一而足。该公司的软件是网络管理的不二之选, 美中不足的是价格高昂	商业软件
Manage engine	各种网管软件	http://www.manageengine.com/network-performance-management.html	Manage engine 公司同样出品各种各样的网管软件, 能与 SolarWinds 公司分庭抗礼。Manage engine 公司的网管软件除了价格高大上以外, 几乎没有缺点	商业软件
HP	IMC	http://h17007.www1.hp.com/us/en/networking/solutions/networkmanagement/index.aspx#.UkgqGT8YhyI	极其出色, 在界面友好程度上要优于其前身 NNM (Network Node Manager) 软件, 绝对值得为其付费	商业软件
OpenNMS	OpenNMS	http://www.opennms.org/	开源软件, 配置门槛较高	自由软件
Nagios	Nagios	http://www.nagios.org/	开源软件, 配置门槛较高	自由软件

常用的 SNMP 平台有下面几个。

- 开源软件 Cacti (<http://www.cacti.net/>)。
- Zabbix (<http://www.zabbix.com/>)。
- MRTG (<http://oss.oetiker.ch/mrtg/>)。
- 遵循 GNU 通用公共许可证的其他 SNMP 平台软件。

以下所列为土豪级 SNMP 平台。

- HP OpenView 管理应用套件 (The HP OpenView suite of management applications) (<http://www8.hp.com/us/en/software-solutions/software.html?compURI=1174702#tab=TAB1>)。
- CA Unicenter (<http://www.ca.com/us/network-performance-management.aspx>)。

还有某些中产级 SNMP 平台, 比如: Plixer 公司推出的各种工具 (<http://www.plixer.com/>) 等。

对网管人员而言, 监控网络、排除网络故障需要使用很多工具, 但 SNMP 平台软件只要一套也就够了。SNMP 平台软件的选择标准是: 能满足网管需求; 使用起来得心应手; 性价比高。对上述网管平台各个方面的比较详见 <http://en.wikipedia.org/wiki/Comparison>

_of_network_monitoring_systems.

NetFlow、Jflow 和 Sflow 分析工具

Cisco 公司的 NetFlow(<http://www.cisco.com/go/netflow>)和 Juniper 公司的 Jflow (<http://www.juniper.net/techpubs/software/erx/junose82/swconfig-ip-services/html/ip-jflow-stats-config2.html>) 都可用来采集路由设备的 TCP/IP 流量统计信息。

Sflow (<http://en.wikipedia.org/wiki/Sflow> and <http://www.sflow.org/index.php>) 是用来监控高速交换网络的业界标准技术

以上各种技术的区别如下所列。

- **NetFlow** 适用于 Cisco 路由器和 Cisco 三层交换机。并非所有的 Cisco 三层交换机都支持 NetFlow, 具体情况要视软件版本和硬件型号而定。有些 Cisco 三层交换机可能需要安装额外的软、硬件才能支持 NetFlow。定义 NetFlow 的 IETF 标准是 RFC 3954 (<http://www.ietf.org/rfc/rfc3954.txt>)。
- **Jflow** 适用于 Juniper 路由器和 Juniper 三层交换机。
- **Sflow** 是监控 LAN 流量的标准, 定义于 RFC 3176 (<http://tools.ietf.org/html/rfc3176>)。
- **IPFIX** (RFC 5101 和 5102) 基于 NetFlow v9 来开发, 是 IETF 制定的标准。

要想使用 NetFlow、Jflow、Sflow 和 IPFIX, 需要事先在交换机或路由器上激活相关特性, 然后这些网络设备才会把采集自本机接口的数据流发送给流量采集工作站。当然, 只有在该工作stations 安装了 Xflow (Xflow 表示 NetFlow、Jflow、Sflow) 分析软件之后, 才能从网络设备正确收集并呈现数据流。

通过安装在流量采集工作站上的分析软件, 即可了解到是哪台主机 (以 IP 地址或 DNS 名称来示人) 生成了何种流量 (流量类型用 TCP/UDP 端口号来表示) 导致了网络拥塞。Wireshark 虽然也能办同样的事儿 (需借助于 **Statistics** 菜单中的 **Conversations** 菜单项功能), 但它只适合做短期流量监控, 而 Xflow 配搭分析软件则适合执行长时期的流量监控任务。

以下所列为作者常用的 Xflow 分析软件。

- Plixer 公司的 scrutinizer flow analyzer (<http://www.plixer.com/Scrutinizer-Netflow-Sflow/scrutinizer-flow-analyzer.html>)。
- SevOne 公司的 NetFlow Analysis (<http://www.sevone.com/technologies/>)

NetFlow-analysis)。

在众多 Xflow 分析软件中,既有免费软件,也有商业软件,有些商业软件可免费使用,但功能受限(对受监控网络设备的接口数有限制,上限一般为 5 个)。

HTTP 流量分析工具

顾名思义,HTTP 流量分析工具只能用来分析 HTTP 流量。下表所列为常见的 HTTP 流量分析工具。

厂 商	软件名称	下载链接	特 色
Eric Lawrence and Telerik	Fiddler	http://fiddler2.com/	是最常用的 HTTP 流量分析工具,而且还是自由软件,能独立抓取 HTTP 流量,并对其加以分析
Simtec Limited	HTTP Watch	http://www.httpwatch.com/	分为商业版和功能受限的基本版。可集成进 Firefox 或 Internet Explorer 浏览器,甚至还支持 iPhone iOS。 HttpWatch 分为两部分: 第一部分是集成进 IE 或 Firefox 中用来收集、查看、保存 HTTP 信息的插件;第二部分是独立运行的日志查看器,名为 HttpWatch Studio

通过 HTTP 流量分析工具可以了解到与 HTTP 有关的统计信息和性能状况,比如,打开某个页面需要多长时间、页面加载过慢的原因以及相关错误信息等。

Syslog 工具

Syslog (<https://tools.ietf.org/html/rfc5424>) 是一种用来记录(网络设备或服务器的)系统日志的协议^①。可在网络中部署 Syslog 服务器(并安装 Syslog 服务器软件),同时在网络设备上开启 Syslog 发送功能,令其在出现变故的情况下,将相关日志信息发送给 Syslog 服务器^②。所谓的变故是指发生在网络设备上的软、件问题,而这些问题可能并不为 SNMP 协议所感知。

Syslog 服务器软件的作用是接收网络设备发送的日志信息,并正确加以呈现。Kiwi Syslog Server (<http://www.kiwisyslog.com/free-edition.aspx>) 就是一款极为好用的 Syslog 服务器软件。前面提到的某些 SNMP 网管平台也会自带 Syslog 服务器软件。

杂项工具

在排除网络故障时,有可能还会用到以下工具。

① 译者注:原文是“Syslog (<https://tools.ietf.org/html/rfc5424>) is a protocol for message logging”。译文酌改。

② 译者注:原文是“‘There are many parameters on communication devices that can be configured, so in cases where a problem occurs, a message will be sent to the Syslog server’”。作者啊,你懂不懂什么是 Syslog?

- **Xplico** 自由软件, 用来从抓包文件中提取应用数据的神器 (<http://www.xplico.org/about>)。
- **Nmap** 安全扫描工具 (<http://nmap.org/>)。
- **Netcat (nc) for Linux** (<http://nc110.sourceforge.net/>)。

网络分析工具

Wireshark 虽然是如今最为常用的网络分析工具, 但有一些其他的网络分析工具也非常好用, 如下所示。

- **Riverbed 公司的 Cascade 套件**: 由 Wireshark 软件的开发者开发, 提供了一套图形界面来分析 wireshark 抓包文件, 链接为 <http://www.riverbed.com/products-solutions/products/performance-management/>。
- **WildPackets 公司的 OmniPeak**: 作者的笔记本里同时安装有 OmniPeak 和 Wireshark, 当流量大到 Wireshark 无法处理时, 作者就会用 OmniPeak 来抓包。该工具内置有极为强大的信息统计工具, 适合在大流量网络中执行抓包分析任务, 链接为 http://www.wildpackets.com/products/omnipeek_network_analyzer。

网络分析工具还有很多, 但作者只用这么两三种。

在某些 Cisco 设备的 IOS 中也内置有一种简易的网络分析工具, Cisco 将其命名为 **Mini Protocol Analyzer** (<https://www.cisco.com/en/US/docs/routers/7600/ios/12.2SR/configuration/guide/mpa.html>)。

Web 站点

以下所列为作者经常访问的与网络技术有关的 Web 站点。

- <http://www.cisco.com> 是作者最常访问的 Web 站点, 可在此学习 Cisco 设备的配置技术。
- <http://www.honeynet.org/> 里面有许多 Wireshark 抓包文件、练习和难题。
- http://packetlife.net/media/library/13/Wireshark_Display_Filters.pdf^① 是 Wireshark 显示过滤参数汇总表。
- <http://wiki.wireshark.org/SampleCaptures> 里面有许多抓包文件样本。
- <http://www.packetlevel.ch/html/tcpdumpf.html> 另一张 Wireshark 显示过滤参数汇总表。

① 译者注: 作者啊, 这是 Web 站点吗?

- <http://perldoc.perl.org/perlre.html> 可找到许多与 Perl 有关的东西; <http://perldoc.perl.org/perlre.html#Regular-Expressions> 和 <http://www.regular-expressions.info/perl.html> 里面有与 Perl 正则表达式有关的内容。

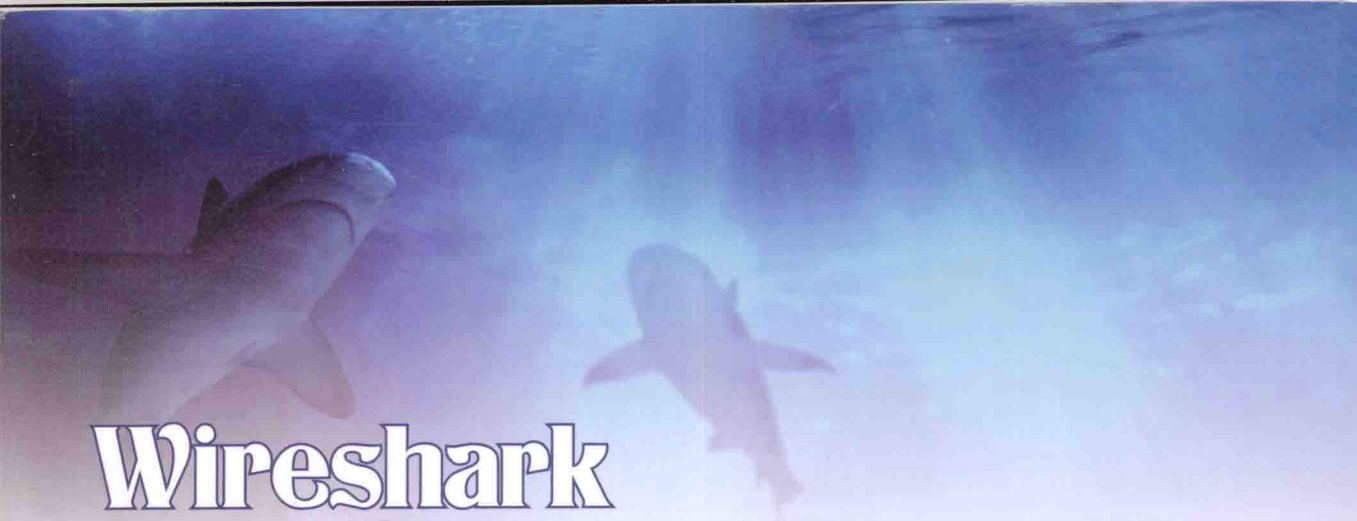
书籍

以下所列为作者研读多年的计算机网络书籍以及经常参阅的各种技术文档的链接。

- 《TCP/IP Illustrated, Volume 1: The Protocols (*Second Edition*)》(*Addison-Wesley Professional Computing Series*) TCP/IP 经典书籍。
- 《The TCP/IP Guide》全面分析了 Internet 协议, 可免费下载, 下载链接为 <http://www.tcpipguide.com/>。
- Cisco Press 出版的众多网络技术书籍、网络专业认证书籍都值得阅读, 该出版社的官网链接为 <http://www.ciscopress.com/>。

Cisco 公司发布的各种网络设计指南: 请到 Cisco 公司官网上输入 “Cisco design guides” 自行搜索^①。

^① 译者注: 原文是 “Cisco design guides: Just Google the subject you're looking for”。Google 是 “非法” 网站, 而百度又很难搜到, 译文酌改。



Wireshark

网络分析实战 Network Analysis Using Wireshark Cookbook

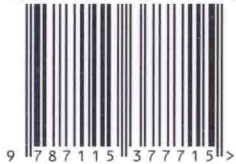
本书深入解析了一系列实用秘诀，读者可用这些秘诀来按部就班地解决发生在网络中的任何故障。

本书以 OSI 参考模型的各个层次为基础，讲解如何解决网络故障。首先，讲解如何解决以太网和 LAN 交换故障、IP 故障以及 TCP/UDP 故障，同时关注 WLAN 故障。TCP 性能问题是 TCP/UDP 故障的重点。其次，介绍如何排除各种常见的应用层协议故障，涉及的应用层协议包括 HTTP、Mail 以及 DNS 等。最后，关注网络取证以及如何发现并定位可能会对网络造成危害的安全性问题。

- 如何在网络中安置并配置 Wireshark，使之能有效执行网络分析。
- 如何配置抓包过滤器和显示过滤器，来获取必要的数据库。
- 如何使用内置于 Wireshark 的强大的信息统计工具和专家系统 (expert system) 工具，来执行网络分析，查明网络故障。
- 如何执行 WiFi 测试，如何解决与无线 LAN 有关的故障。
- 如何查明与 TCP 有关的性能问题。
- 掌握 IO Graphs 工具的标准和高级功能。
- 如何发现并解决与带宽、吞吐量、延迟及丢包有关的网络问题。
- 如何发现并解决包括语音、视频以及多媒体应用在内的各种 VoIP 故障。



ISBN 978-7-115-37771-5



9 787115 377715 >

封面设计：董志桢

分类建议：计算机 / 网络技术 / 网络安全
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-37771-5

定价：79.00 元