



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

异步社区Kensuke(18319066421) 专享 请尊重版权



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

异步社区Kensuke(18319066421)专享 请尊重版权

异步社区电子书

感谢您购买异步社区电子书！异步社区已上架电子书 500 余种，社区还会经常发布福利信息，对社区有贡献的读者赠送免费样书券、优惠码、积分等等，希望您在阅读过程中，把您的阅读体验传递给我们，让我们了解读者心声，有问题我们会及时修正。

社区网址：<http://www.epubit.com.cn/>

反馈邮箱：contact@epubit.com.cn

异步社区里有什么？

图书、电子书（[半价电子书](#)）、优秀作译者、访谈、技术会议播报、赠书活动、下载资源。

异步社区特色：

纸书、电子书同步上架、纸电捆绑超值优惠购买。

最新精品技术图书全网首发预售。

晒单有意外惊喜！

异步社区里可以做什么？

博客式写作发表文章，提交勘误赚取积分，积分兑换样书，写书评赢样书券等。

联系我们：

微博：

@ 人邮异步社区

@ 人民邮电出版社 - 信息技术分社

微信公众号：

人邮 IT 书坊

异步社区

QQ 群：368449889

图书在版编目 (C I P) 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 2015 年 2 月第 1 版
印数: 1-3 000 册 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 毕业于以色列技术学院 (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，感谢二位在那个重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、东亚以及世界其他各地的客户。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比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。

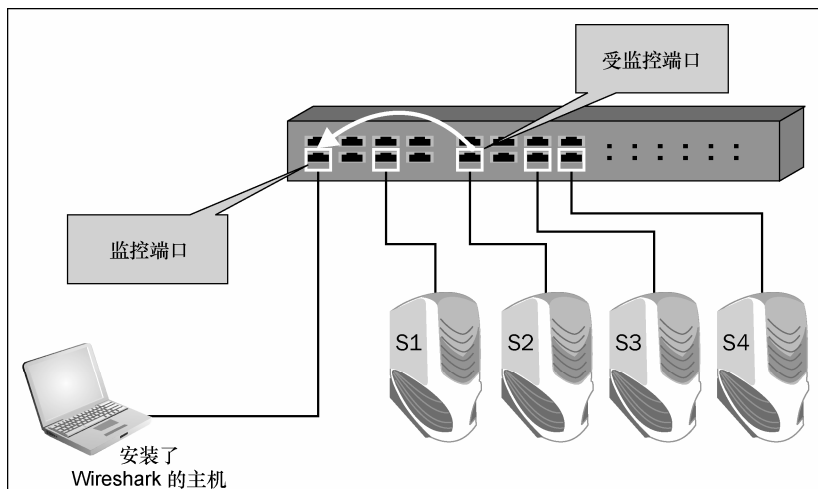


图1.1

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。

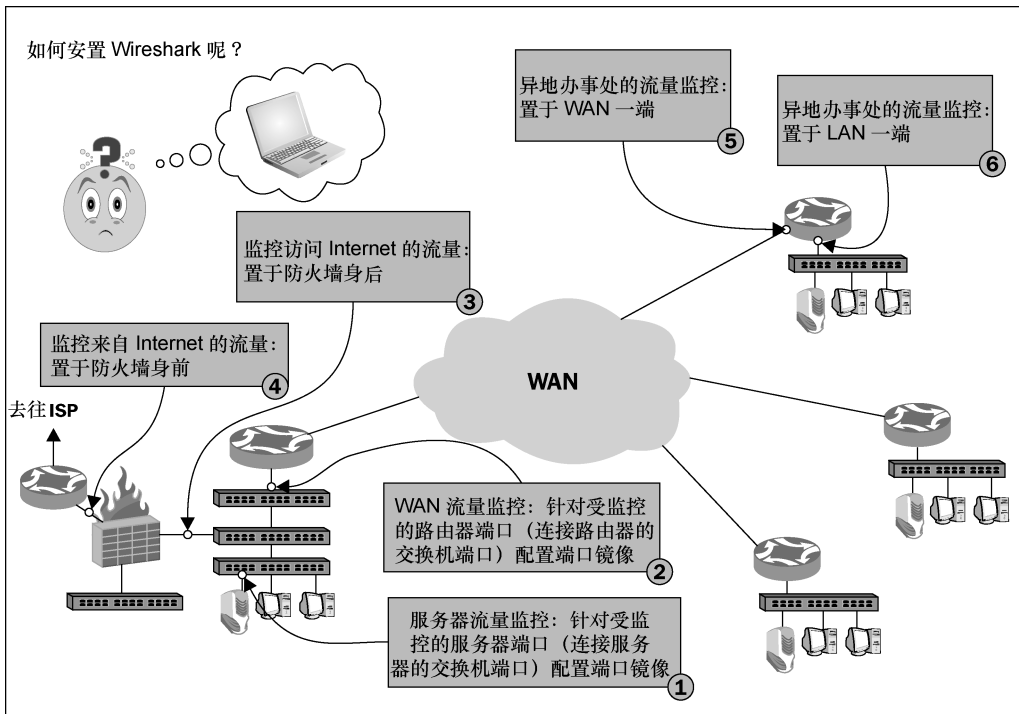


图 1.2

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。

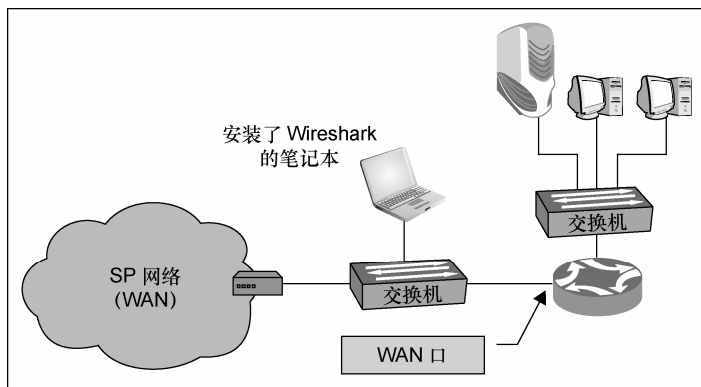


图1.3

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 主机还会再抓取一次。这么一来，便采集到了重复的流量。

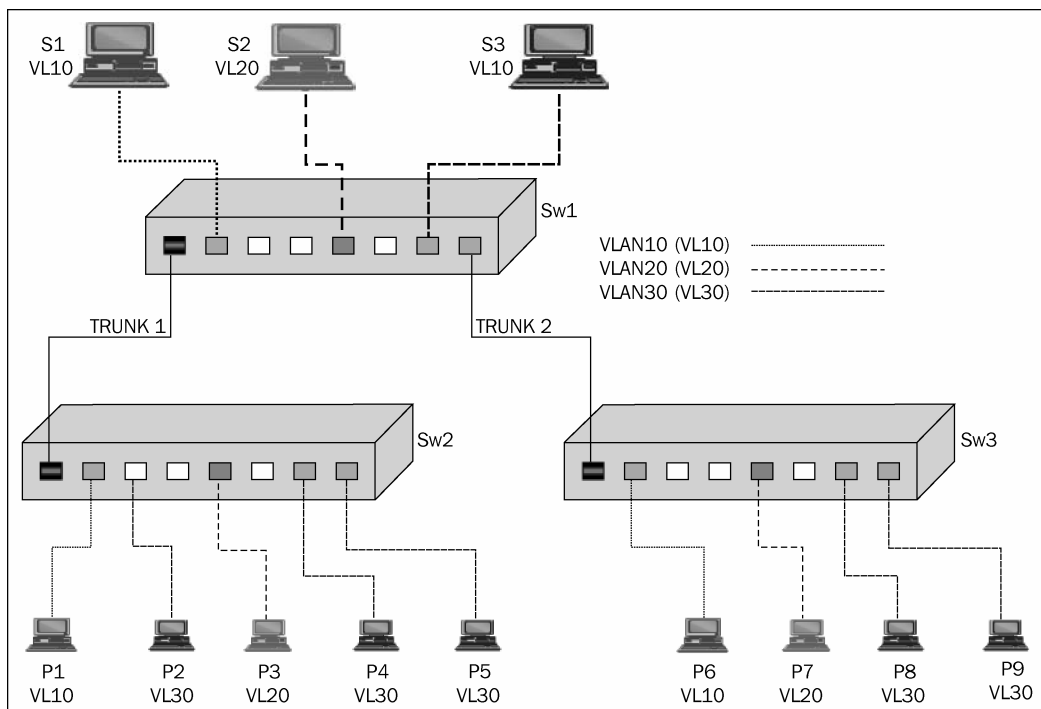


图1.4

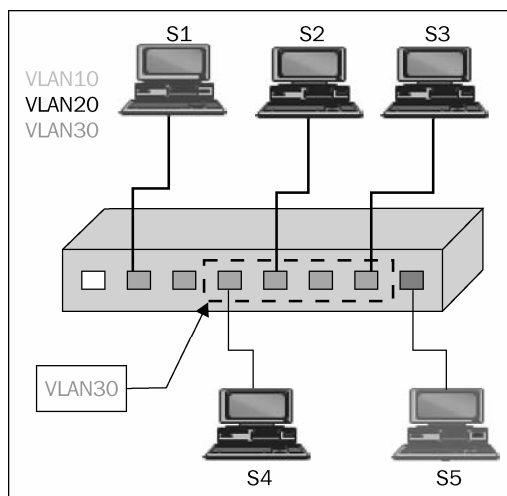


图1.5

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）。

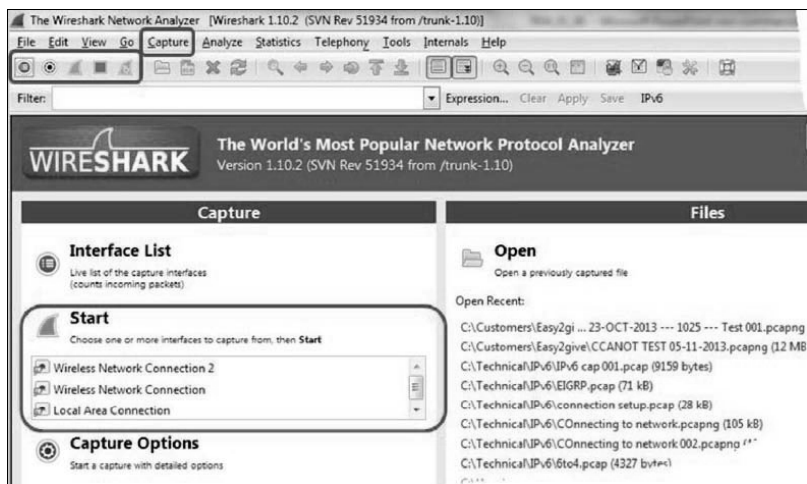


图 1.6

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。

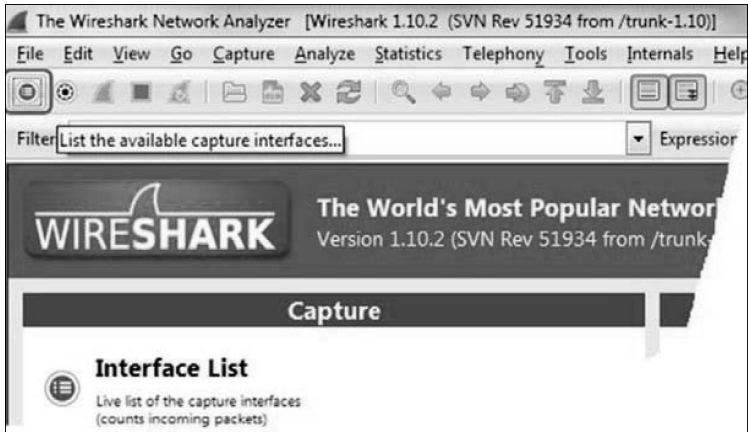


图1.7

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）。

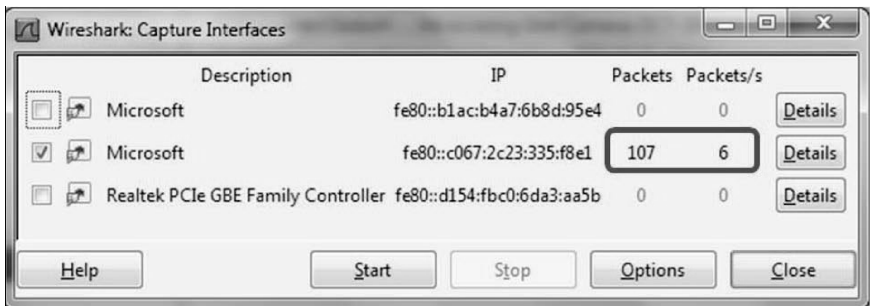


图1.8

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。

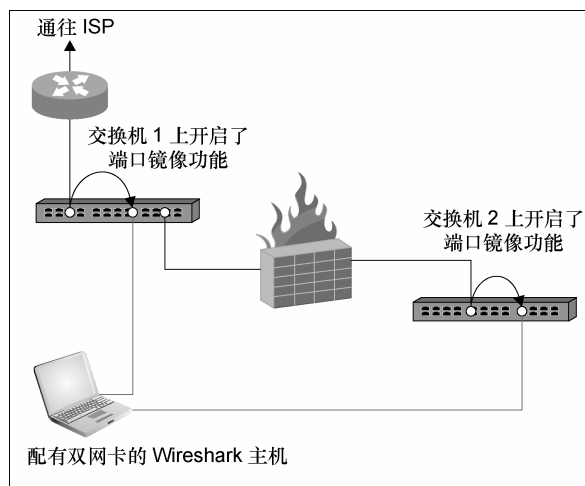


图 1.9

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。

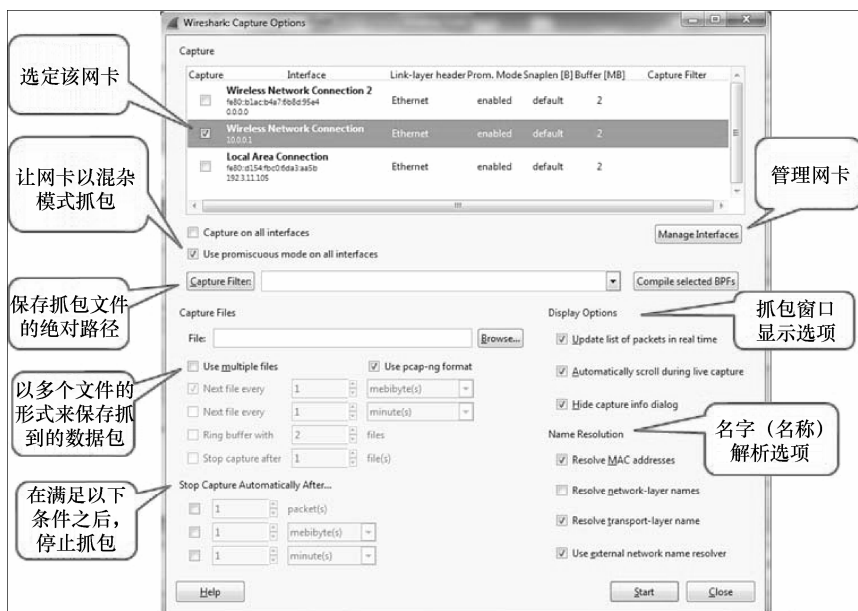


图 1.10

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“_xxxxx_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,行使用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。

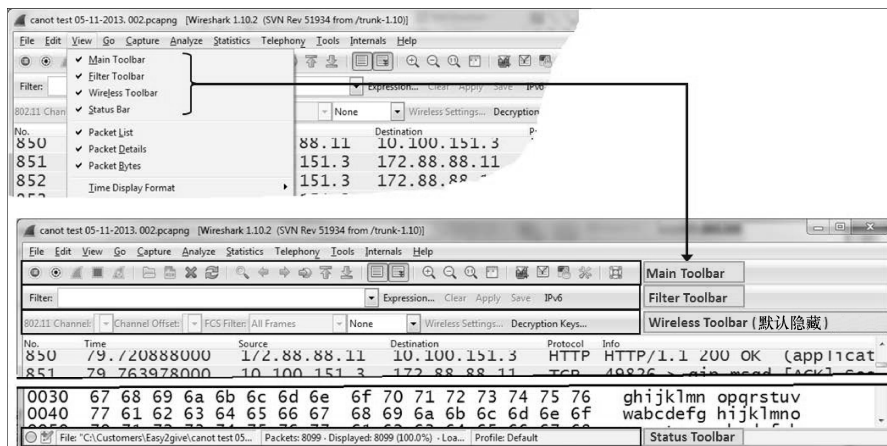


图 1.11

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。

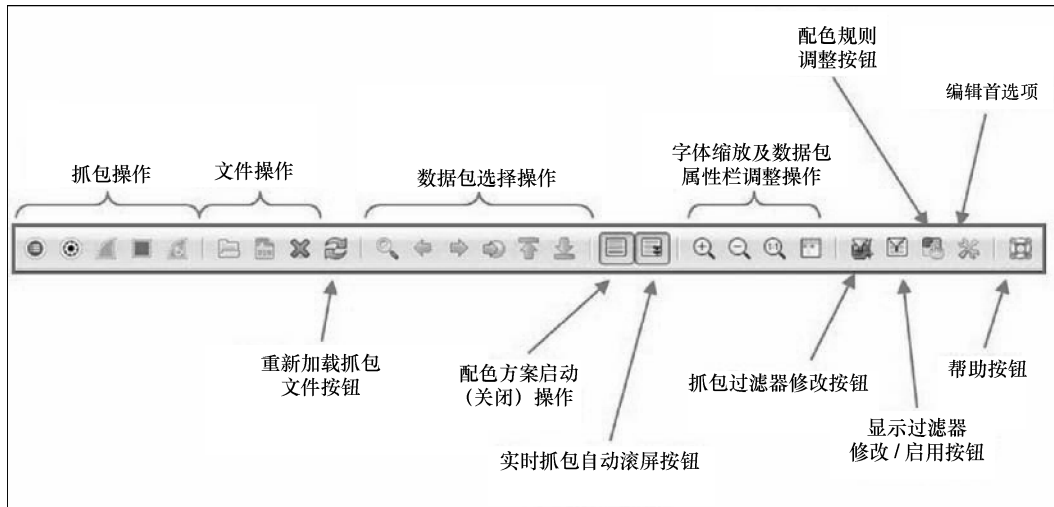


图1.12

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。

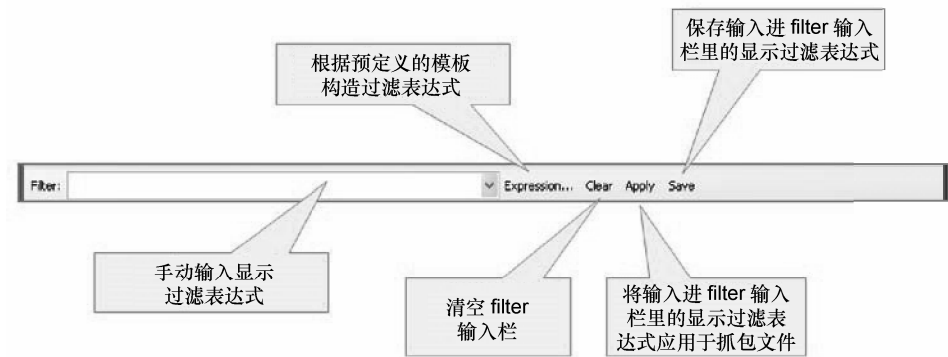


图1.13

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。

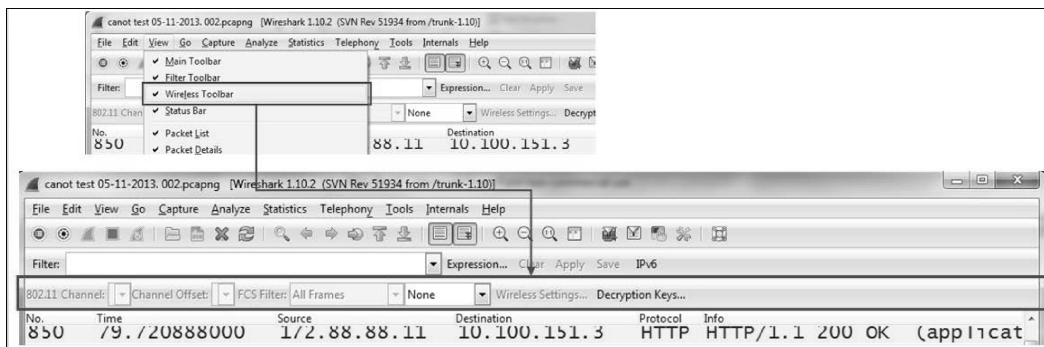


图 1.15

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。

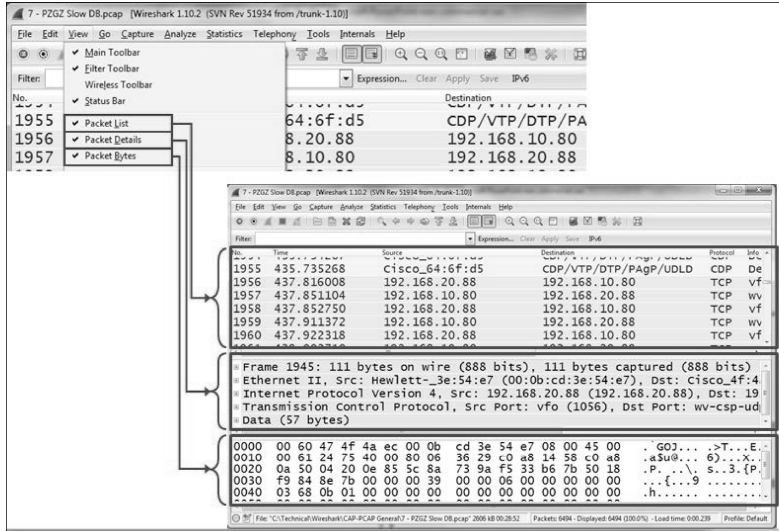


图 1.16

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，也有可能需要取消勾选 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。

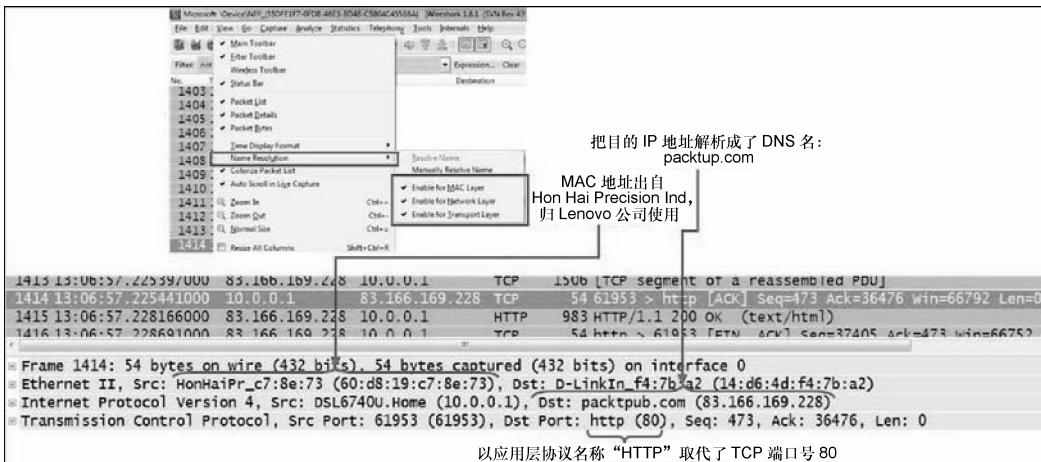


图 1.17

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。

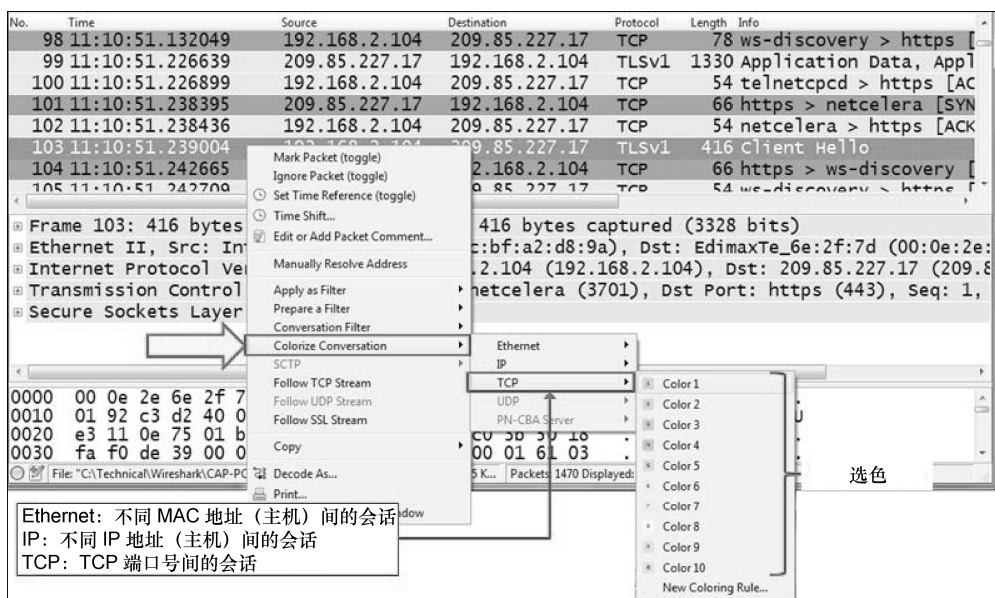


图 1.18

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。

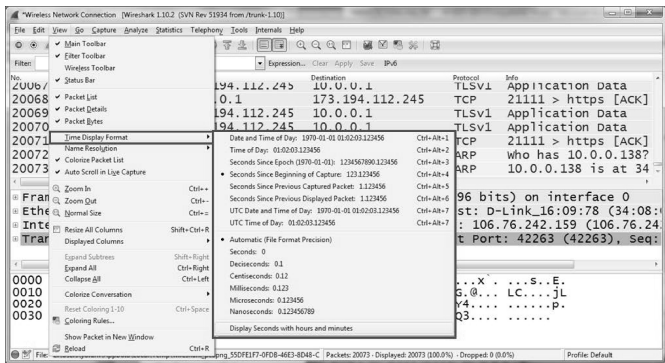


图 1.19

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。

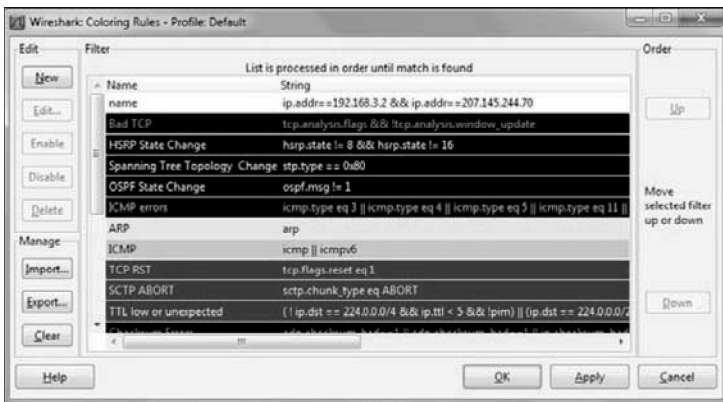


图 1.20

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。

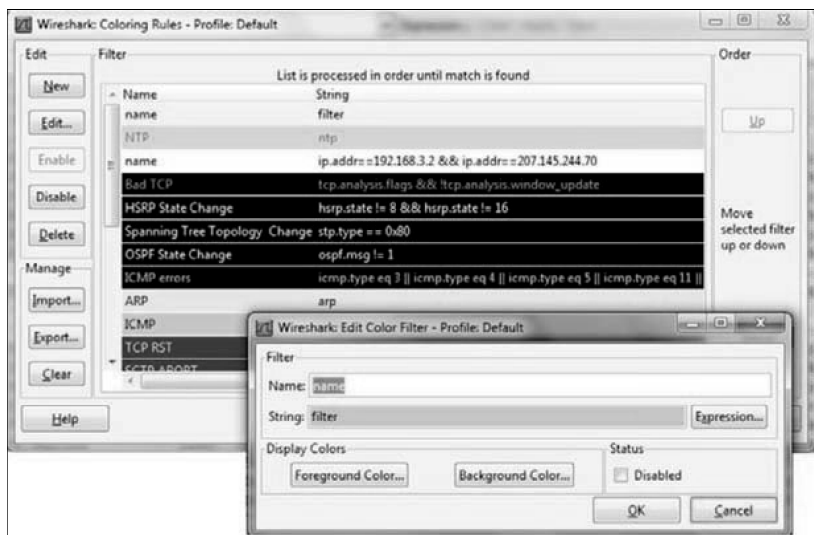


图 1.21

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。

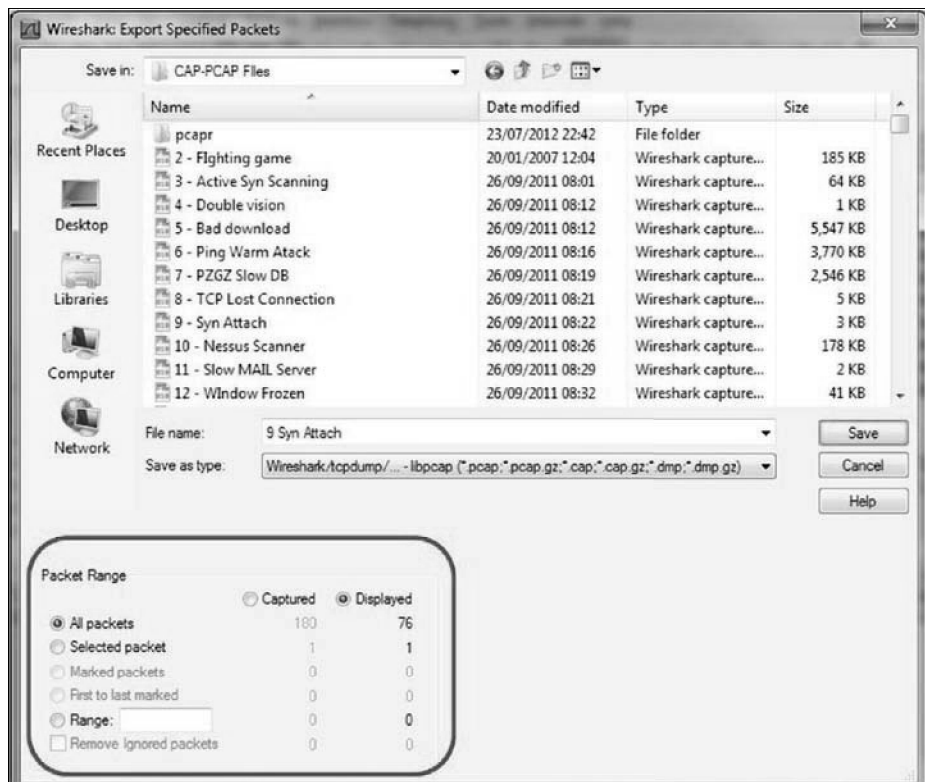


图 1.22

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。

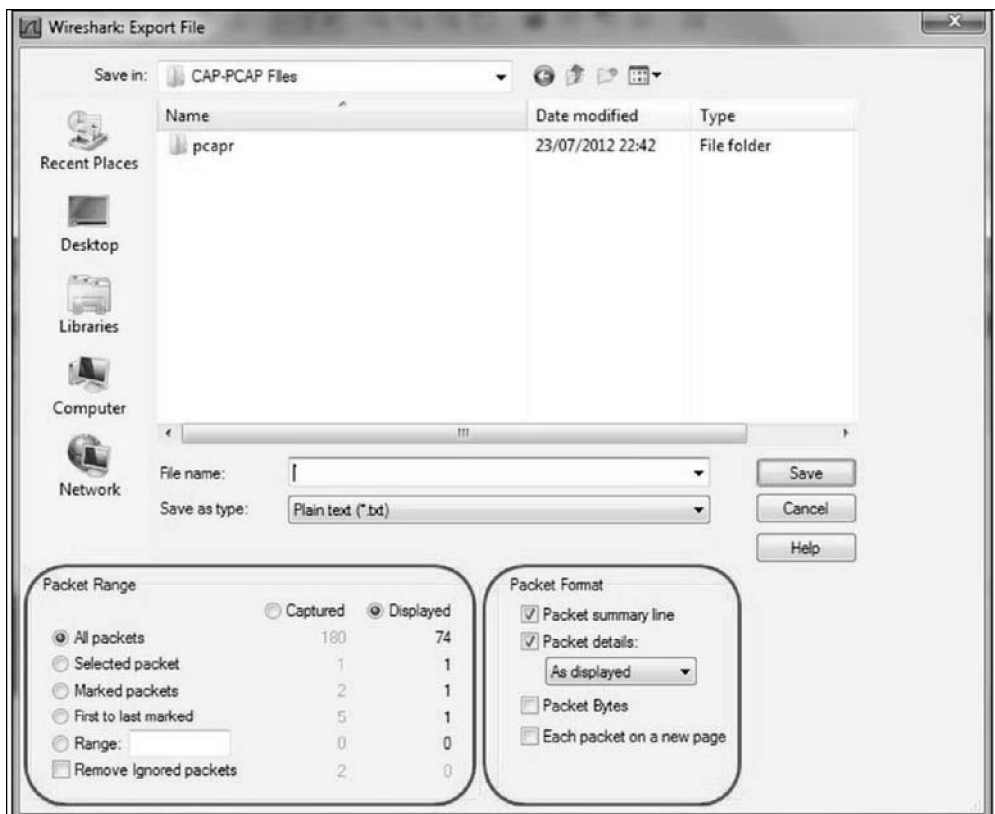


图1.23

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**）。

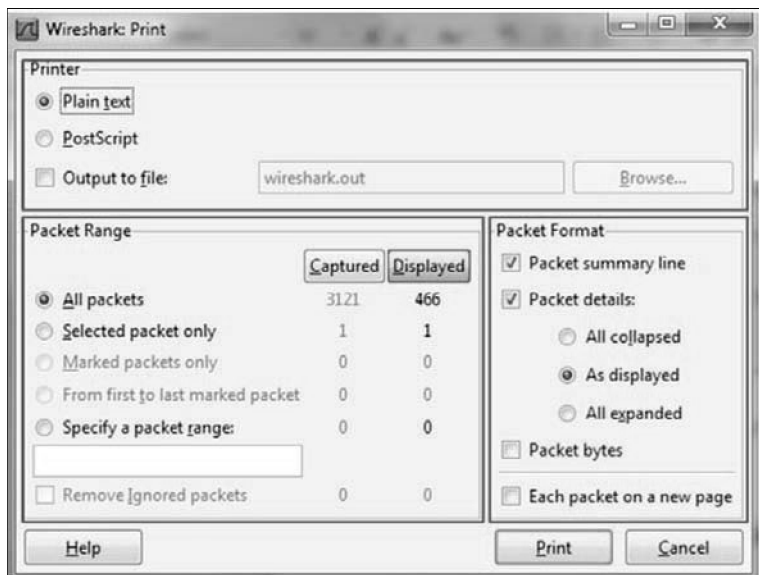


图1.24

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。

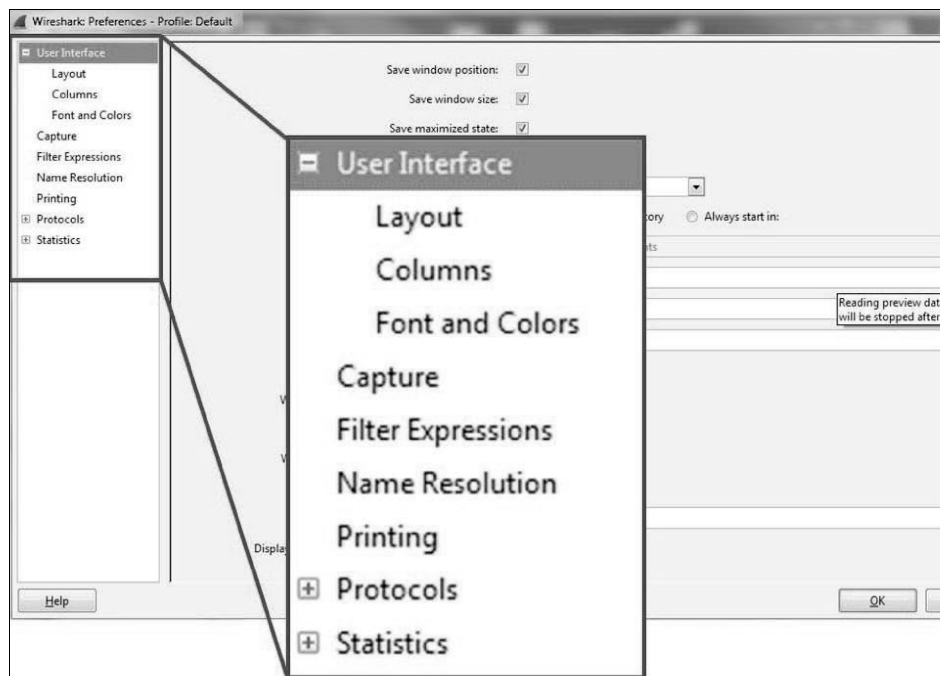


图1.25

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。

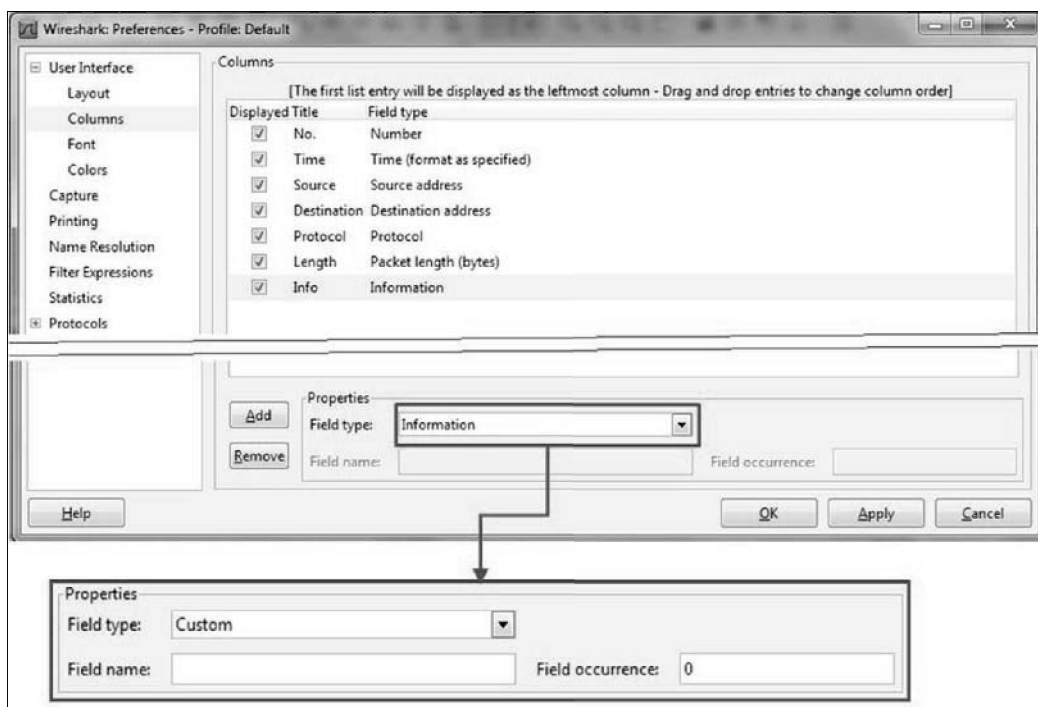


图 1.26

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。

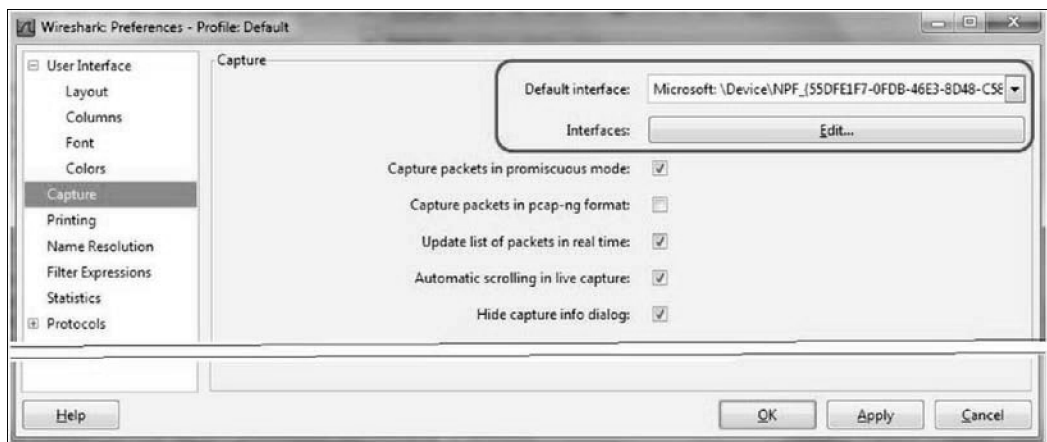


图 1.27

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。

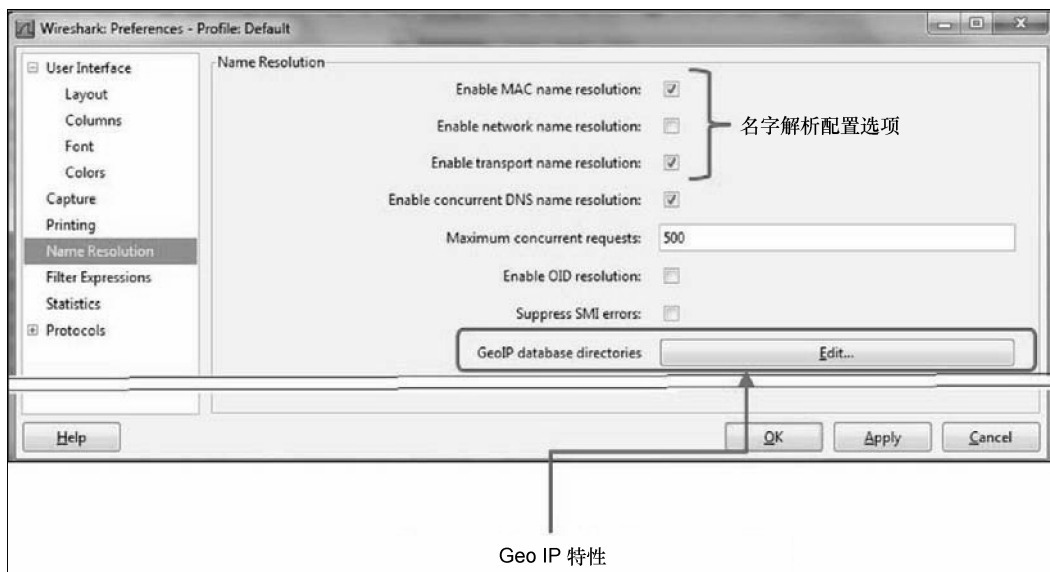


图1.28



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。

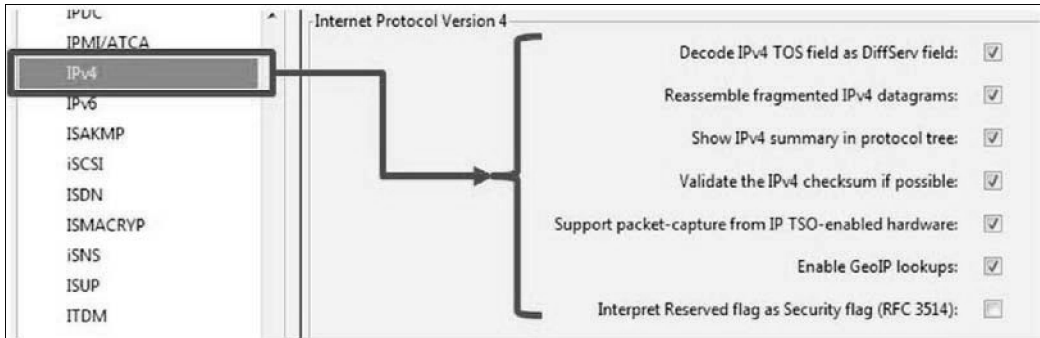


图1.30

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。

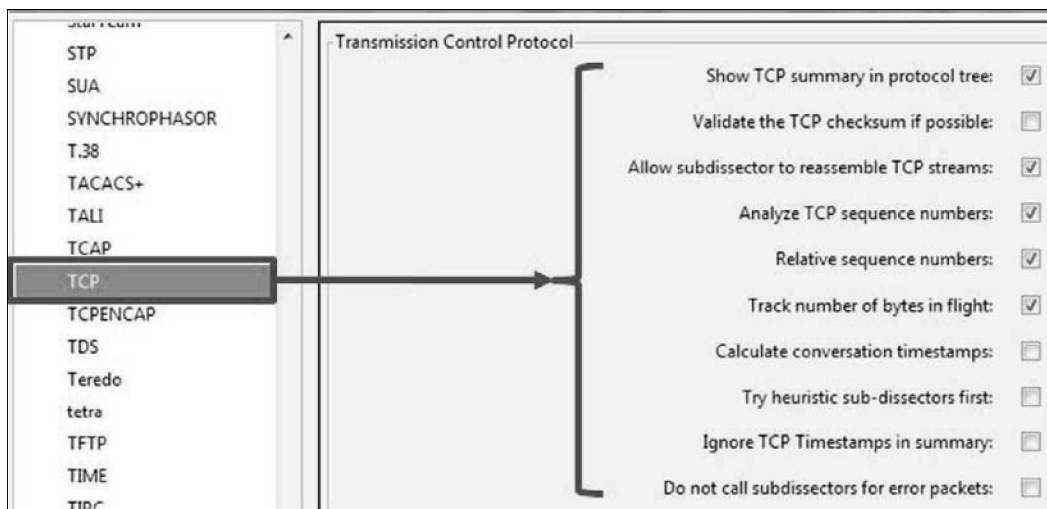


图1.31

- **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。

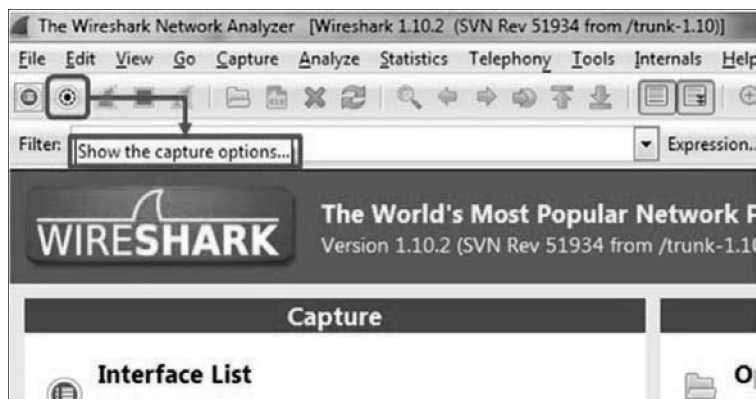


图2.1

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

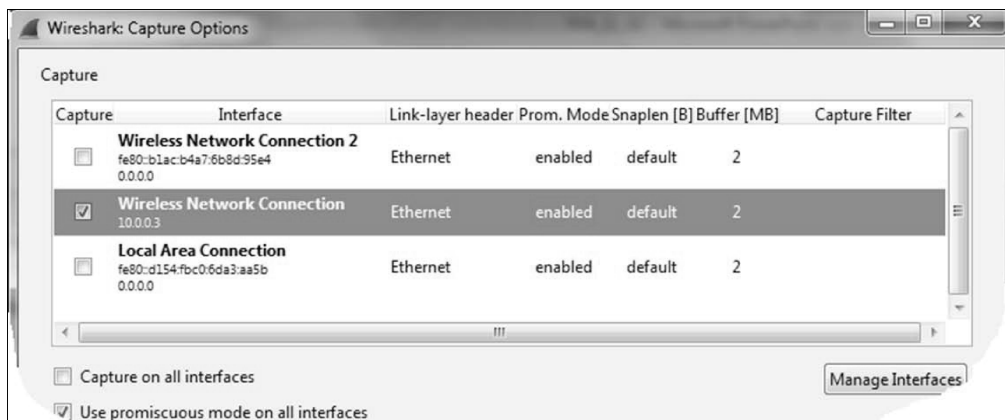


图2.2

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。

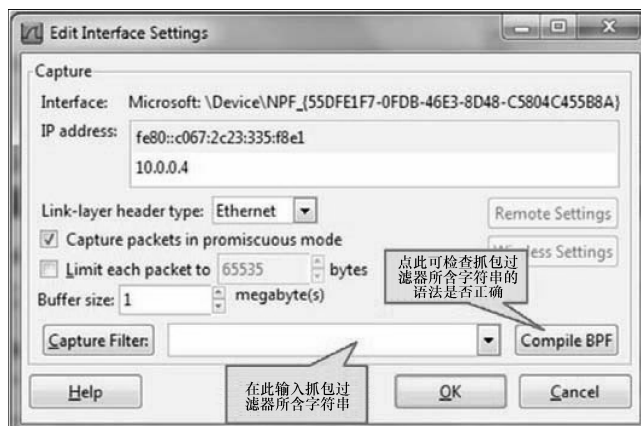


图2.3

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 **Compile 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。

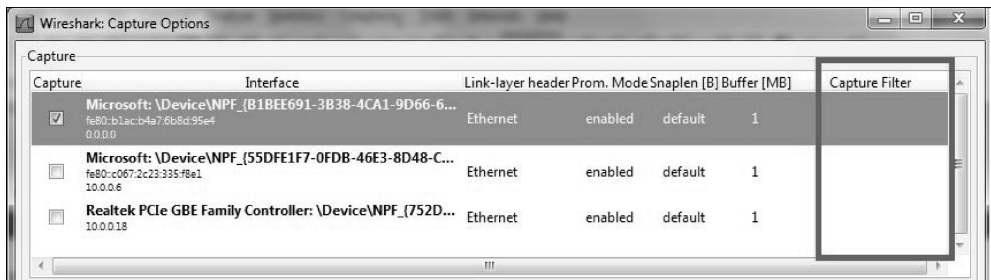


图2.5

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。

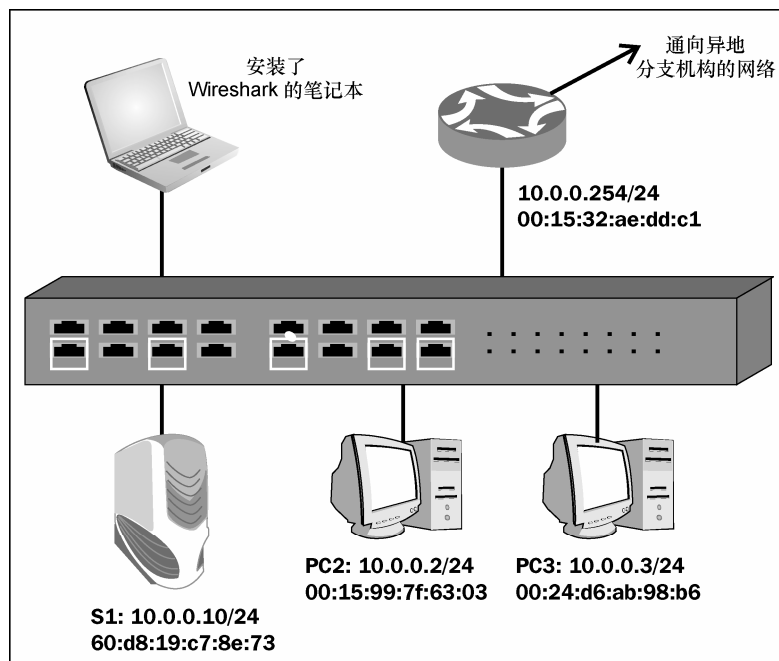


图2.6

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了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

以下所列是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。

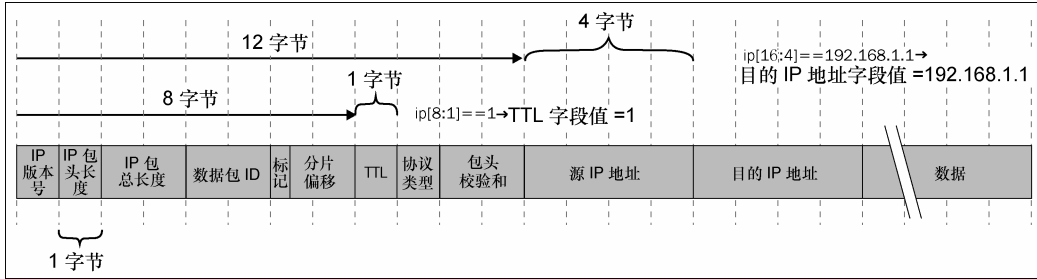


图2.7

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称为（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。

- `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）。
- `tcp[13] & 0x01 = 1`: 用来抓取 **FIN** 位置 1，但 **ACK** 位置 0 的 TCP 流量。
- `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。

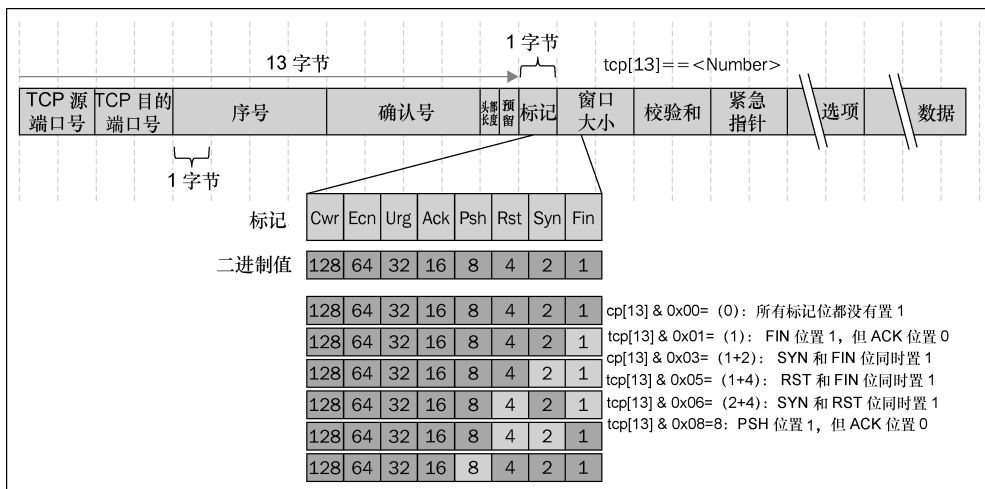


图2.8

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）。

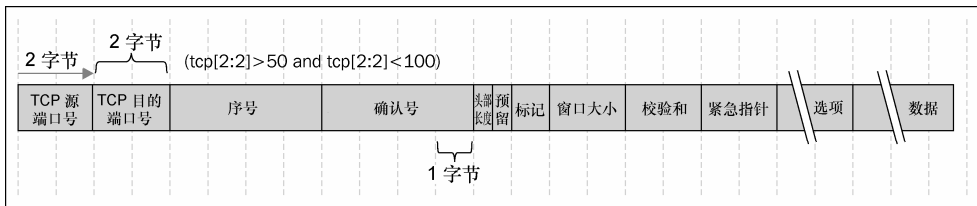


图2.9

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。

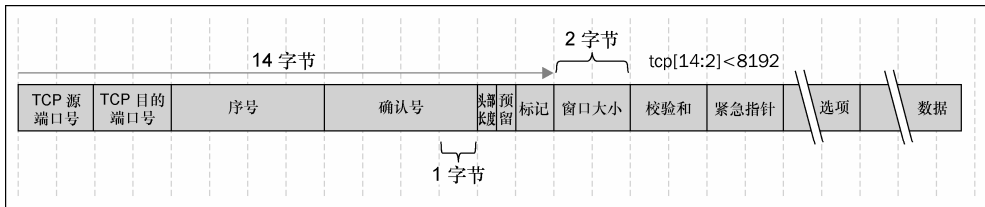


图2.10

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，捌飭显示过滤器时，所有数据都已被 Wireshark 捕获，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==10.0.02 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。

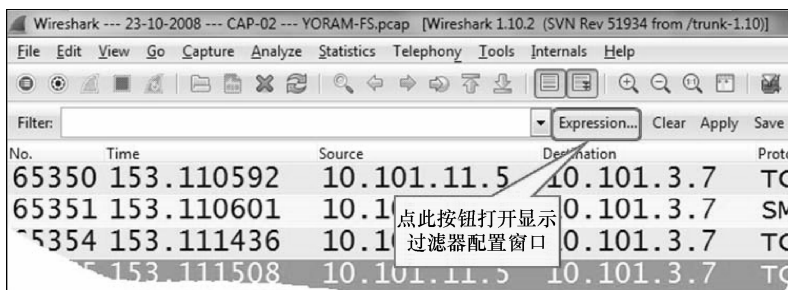


图3.1

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。

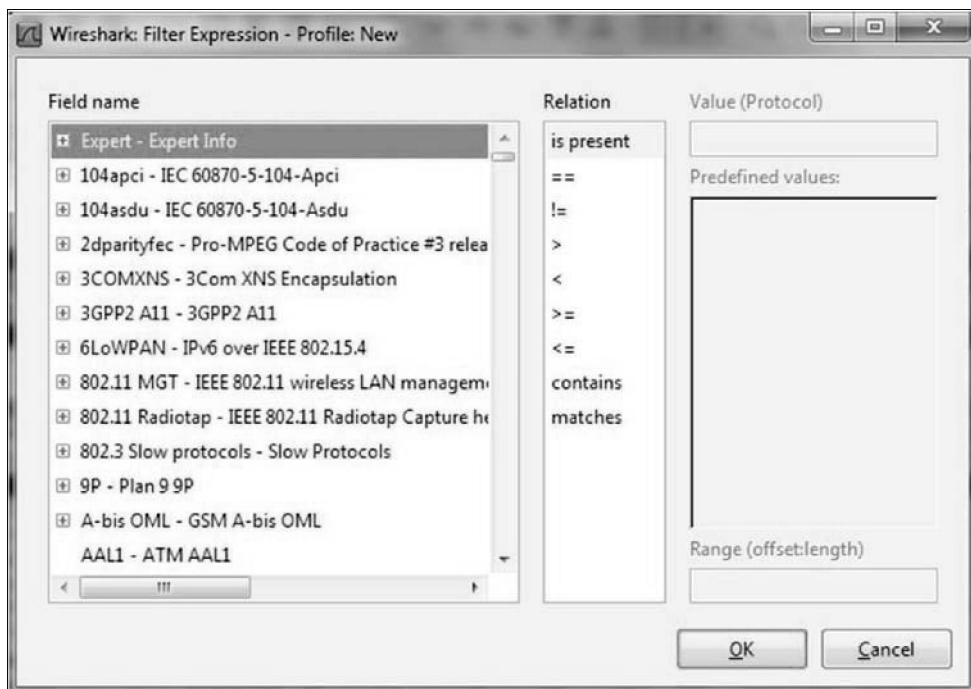


图3.2

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 Filed 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，则只要點選 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）。

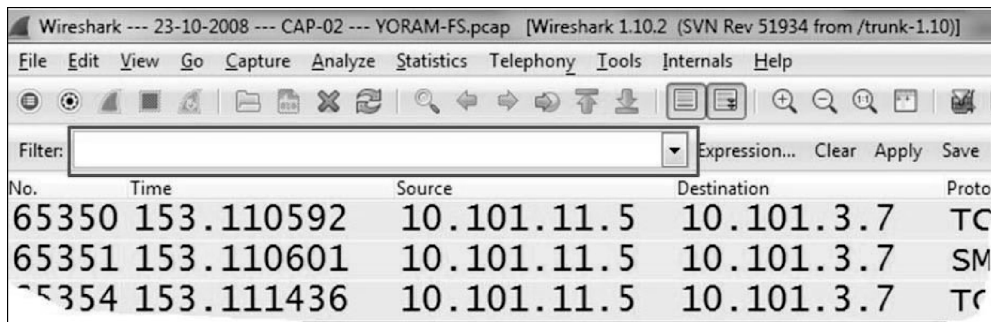


图3.3

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。

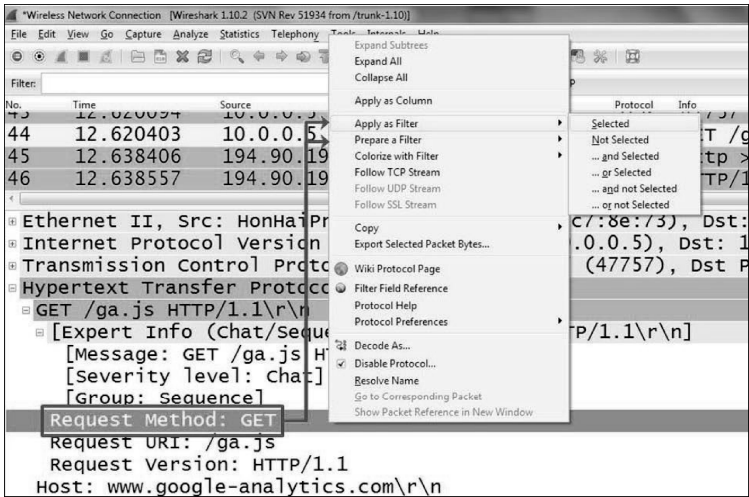


图3.4

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）。

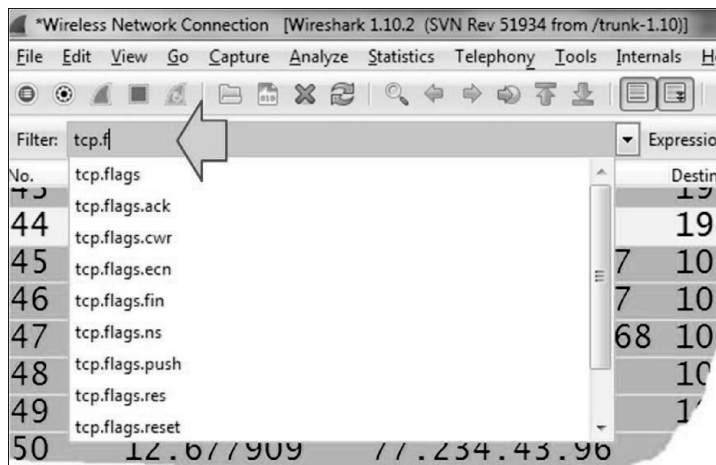


图3.5

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。

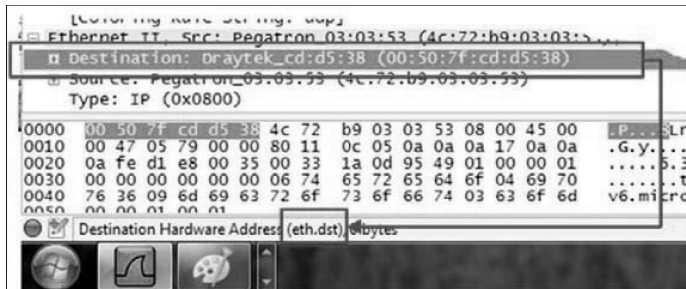


图3.6

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。

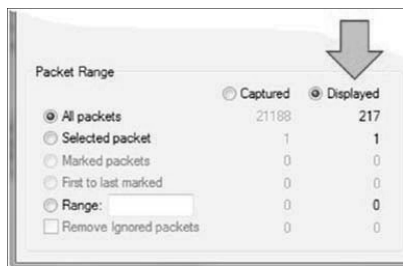


图3.7

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。
- `rame.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	<code>ip.addr == x.x.x.x/y</code> 其中, x 为 0~255, y 为 0~32	<code>ip.addr == 192.168.200.0/24</code> 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	<code>ipv6.addr == x:x:x:x:x:x:x:x</code> <code>ipv6.addr == x::x:x:x:x</code> 在以上过滤表达式中, x 为十六进制数 0~f	<code>ipv6.addr == fe80::85ab:dc2e:ab12:e6c7</code>
IPv6 网络地址	<code>ipv6.addr == x::/y</code> 其中, x 为十六进制数 0~f, y 为 0~128	<code>ipv6.addr == fe80::/16</code> 该过滤表达式涵盖了 IPv6 前缀 <code>fe80::/16</code> 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 == 2①`。

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。