

浙江大学



本科实验报告

姓名：

学院： 生物医学工程与仪器科学学院

系： 生物医学工程系

专业： 生物医学工程

学号：

指导教师： 耿晨歌

一、实验要求

1、掌握 ipconfig 命令的使用，解析网卡配置信息

- (1) 解析网卡名称、物理地址（MAC 地址）、IP 地址、子网掩码、默认网关和 DHCP 服务器信息。
- (2) 验证 DHCP 租约状态及 DNS 服务器配置。

2、理解 netstat 命令的输出，分析网络连接状态

- (1) 作为客户端（server）发起连接，识别源地址、端口及对应应用程序。
- (2) 作为服务器（client）监听端口，解析目的地址、端口及侦听进程。
- (3) 分析 TCP 连接状态（如 ESTABLISHED、SYN_SENT 等）。

3、运用 tracert 进行路由跟踪，定位网络路径

- (1) 跟踪校内、教育网及国外主机的路由路径。
- (2) 通过公共 IP 库反查路由节点地理位置。

4、掌握 arp 地址解析协议的操作方法

- (1) 查看本网段内主机的 IP 与 MAC 地址映射关系。
- (2) 验证 ARP 表动态更新机制（如访问新主机后条目变化）。
- (3) 尝试访问周围主机并列出刚访问过的主机。

5、能够通过实验数据分析网络协议栈工作机制

二、实验代码与注释

1、ipconfig 命令详解

ipconfig (Windows) 是用于查看和诊断网络配置的命令行工具。通过分析其输出内容，可以理解计算机的网络接口配置、IP 地址分配、网关设置及 DHCP 服务状态等核心网络信息。

(1) 网卡 (Network Adapter)

网卡（网络接口卡）是计算机与网络连接的硬件或虚拟设备，每个网卡对应一个网络接口（如以太网、Wi-Fi、虚拟网卡等）。其在 ipconfig 的属性主要包括：

名称（如 以太网 2、WLAN）：标识不同网络接口的逻辑名称。

物理地址（MAC 地址）（如 00-FF-0F-B0-86-4C）：网卡的唯一硬件标识符，用于数据链路层（局域网内）的通信。

媒体状态（Media State）：显示是否连接到网络（如 Media disconnected 表示未连接）。

（2）IP 地址与子网掩码（Subnet Mask）

IP 地址是逻辑地址，用于在网络层（如互联网或局域网）唯一标识设备。而子网掩码与 IP 地址配合使用，划分网络地址和主机地址（如 255.255.255.0 表示前 24 位为网络地址），其意义在于确定设备所属的本地子网范围（如 192.168.132.1/24 表示该子网可容纳 254 台主机）。

（3）默认网关（Default Gateway）

即本地网络的出口设备（通常是路由器），用于将数据转发到其他网络（如互联网）。一般而言网关 IP 地址不会为空（如 10.162.0.1）；若为空（如 VMware 虚拟网卡），表示该接口仅用于本地通信。当目标 IP 地址不在本地子网时，数据包会发送到网关进行路由。

（4）DHCP 服务（动态主机配置协议）

DHCP 服务器自动为设备分配 IP 地址、子网掩码、网关和 DNS 服务器地址，简化网络配置。DHCP 故障会导致 IP 地址分配失败（如 169.254.x.x 表示自动配置失败）。

（5）虚拟网卡（如 VMware 适配器）

由虚拟化软件（如 VMware、VirtualBox）创建的虚拟网络接口，用于虚拟机与宿主机或外部网络通信。虚拟网卡的配置（如 VMnet1 的 192.168.132.1）可与物理网络隔离，用于模拟私有网络环境；虚拟网络也通常有自己的 DHCP 服务（如 192.168.132.254）。

（6）DNS

DNS 服务器（如 10.10.0.21）用于将域名（如 www.example.com）解析为 IP 地址的服务器。

```
# 显示所有适配器的完整配置信息
ipconfig /all

# 显示所有分段（网络隔离）的配置信息
ipconfig /allcompartments

# 更新指定适配器的 IPv4 地址租约（例如 "Wi-Fi"）
ipconfig /renew "Wireless Network Connection"

# 释放指定适配器的 IPv6 地址
ipconfig /release6 "Local Area Connection"

# 清除 DNS 缓存并重新注册 DHCP 租约
ipconfig /registerdns

# 显示适配器支持的 DHCP 类 ID
ipconfig /showclassid "Wi-Fi"

# 修改适配器的 DHCP 类 ID 为 "MyClass"
ipconfig /setclassid "Wi-Fi" MyClass

# 显示详细信息
ipconfig /all

# 更新所有名称以 EL 开头的连接的 IPv4 地址租约
ipconfig /renew EL*

# 释放所有名称中包含 Con 的连接的 IPv4 地址租约
ipconfig /release *Con*

# 显示有关所有分段的详细信息
ipconfig /allcompartments /all
```

2、netstat 高级用法

```
# 显示所有连接和侦听端口
netstat -a
```

显示在创建每个连接或侦听端口时涉及的可执行程序,可能较耗时且需要足够权限,某些情况下会显示组件序列

`netstat -b`

显示以太网统计信息,可与 `-s` 选项结合使用

`netstat -e`

显示外部地址的完全限定域名 (FQDN)

`netstat -f`

以数字形式显示地址和端口号,避免地址解析

`netstat -n`

显示拥有的与每个连接关联的进程 ID,便于定位占用网络连接的进程

`netstat -o`

显示 `proto` 指定的协议的连接,`proto` 可以是 TCP、UDP、TCPv6 或 UDPv6;若与 `-s` 选项一起使用,可显示每个协议的统计,`proto` 可以是 IP、IPv6、ICMP、ICMPv6、TCP、TCPv6、UDP 或 UDPv6

`netstat -p proto`

显示路由表,可查看网络路由信息

`netstat -r`

显示每个协议的统计信息,默认显示 IP、IPv6、ICMP、ICMPv6、TCP、TCPv6、UDP 和 UDPv6 的统计,`-p` 选项可用于指定默认的子网

`netstat -s`

显示当前连接卸载状态

`netstat -t`

按指定的间隔秒数重新显示选定的统计信息,按 `CTRL+C` 停止重新显示,若省略 `interval`,则只打印当前的配置信息一次

`netstat [interval]`

显示所有连接和侦听端口,并以数字形式显示地址和端口号,同时显示关联的进程 ID

`netstat -ano`

显示 TCP 协议的连接统计信息

```
netstat -p tcp -s
```

3、tracert 路由跟踪

不将地址解析成主机名，加快跟踪速度

```
tracert -d
```

设置搜索目标的最大跃点数，避免无限循环跟踪

```
tracert -h maximum_hops
```

与主机列表一起的松散源路由（仅适用于 IPv4）

```
tracert -j host-list
```

设置等待每个回复的超时时间（以毫秒为单位）

```
tracert -w timeout
```

跟踪往返行程路径（仅适用于 IPv6）

```
tracert -R
```

指定要使用的源地址（仅适用于 IPv6）

```
tracert -S srcaddr
```

强制使用 IPv4 进行路由跟踪

```
tracert -4
```

强制使用 IPv6 进行路由跟踪

```
tracert -6
```

跟踪到目标主机的路由路径

```
tracert target_name
```

跟踪到 `zupo.zju.edu.cn` 的路由路径，不进行地址解析

```
tracert -d zupo.zju.edu.cn
```

跟踪到 `10.10.2.23` 的路由路径，最大跃点数设置为 20

```
tracert -h 20 10.10.2.23
```

4、arp 地址管理

通过询问当前协议数据，显示当前 ARP 项。若指定 `inet_addr`，则只显示指定计算机的 IP 地址和物理地址；若不止一个网络接口使用 ARP，则显示每个 ARP 表的项

```
arp -a [inet_addr]
```

与 `-a` 选项功能相同

```
arp -g
```

在详细模式下显示当前 ARP 项，所有无效项和环回接口上的项都将显示

```
arp -v
```

删除指定主机的 ARP 表项，`inet_addr` 可以是通配符 `*`，以删除所有主机

```
arp -d inet_addr [if_addr]
```

添加主机并将 Internet 地址 `inet_addr` 与物理地址 `eth_addr` 相关联，该项是永久的。若指定 `if_addr`，则修改该接口的地址转换表

```
arp -s inet_addr eth_addr [if_addr]
```

显示当前 ARP 表

```
arp -a
```

添加静态 ARP 表项，将 IP 地址 `157.55.85.212` 与物理地址 `00-aa-00-62-c6-09` 关联

```
arp -s 157.55.85.212 00-aa-00-62-c6-09
```

删除 IP 地址为 `157.55.85.212` 的 ARP 表项

```
arp -d 157.55.85.212
```

显示当前 ARP 表

```
arp -a
```

添加静态 ARP 表项，将 IP 地址 `157.55.85.212` 与物理地址 `00-aa-00-62-c6-09` 关联

```
arp -s 157.55.85.212 00-aa-00-62-c6-09
```

删除 IP 地址为 `157.55.85.212` 的 ARP 表项

```
arp -d 157.55.85.212
```

三、实验结果与分析

1、ipconfig 网络配置分析

目前的所有网卡包括：

网卡类型	名称	物理地址
Unknown adapter 本地连接	本地连接	00-FF-0F-B0-86-4C
Ethernet adapter 以太网 2	以太网 2	00-FF-47-3D-81-D1
Wireless LAN adapter 本地连接 * 1	本地连接 * 1	58-1C-F8-2E-5B-1D
Wireless LAN adapter 本地连接 * 2	本地连接 * 2	5A-1C-F8-2E-5B-1C
Ethernet adapter VMware Network Adapter VMnet1	VMware Network Adapter VMnet1	00-50-56-C0-00-01
Ethernet adapter VMware Network Adapter VMnet8	VMware Network Adapter VMnet8	00-50-56-C0-00-08
Wireless LAN adapter WLAN	WLAN	58-1C-F8-2E-5B-1C
Ethernet adapter 蓝牙网络连接	蓝牙网络连接	58-1C-F8-2E-5B-20

Ethernet adapter 以太网	以太网	E8-80-88-B9-55-2D
----------------------	-----	-------------------

在其中已连接网卡的 IP 地址和子网掩码为：

(1) VMware Network Adapter VMnet1

IPv4 地址：192.168.132.1

子网掩码：255.255.255.0

(2) VMware Network Adapter VMnet8

IPv4 地址：192.168.111.1

子网掩码：255.255.255.0

(3) Wireless LAN adapter WLAN

IPv4 地址：10.162.13.233

子网掩码：255.255.0.0

默认网关：10.162.0.1（IPv4）、fe80::763a:20ff:feb9:e802（IPv6）

其对应的 DHCP 服务器及相关信息为：

(1) VMware Network Adapter VMnet1

DHCP 已启用：Yes

DHCP 服务器：192.168.132.254

(2) VMware Network Adapter VMnet8

DHCP 已启用：Yes

DHCP 服务器：192.168.111.254

(3) Wireless LAN adapter WLAN

DHCP 已启用: Yes

DHCP 服务器: 10.162.0.1

Windows IP Configuration

Host Name : Bubbles
Primary Dns Suffix :
Node Type : Hybrid
IP Routing Enabled. : No
WINS Proxy Enabled. : No

Unknown adapter 本地连接:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : TAP-Windows Adapter V9
Physical Address. : 00-FF-0F-B0-86-4C
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes

Ethernet adapter 以太网 2:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : Sangfor SSL VPN CS Support System VNIC
Physical Address. : 00-FF-47-3D-81-D1
DHCP Enabled. : No
Autoconfiguration Enabled : Yes

Wireless LAN adapter 本地连接* 1:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : Microsoft Wi-Fi Direct Virtual Adapter
Physical Address. : 58-1C-F8-2E-5B-1D
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes

Wireless LAN adapter 本地连接* 2:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. : 5A-1C-F8-2E-5B-1C
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes

Ethernet adapter VMware Network Adapter VMnet1:

Connection-specific DNS Suffix . :
Description : VMware Virtual Ethernet Adapter for VMnet1
Physical Address. : 00-50-56-C0-00-01
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes
Link-local IPv6 Address : fe80::85c2:7d80:79b9:e05%18(Preferred)
IPv4 Address. : 192.168.132.1(Preferred)
Subnet Mask : 255.255.255.0
Lease Obtained. : 2025年3月22日 19:16:54
Lease Expires : 2025年3月22日 19:46:54
Default Gateway :
DHCP Server : 192.168.132.254
DHCPv6 IAID : 822104150
DHCPv6 Client DUID. : 00-01-00-01-2C-30-02-90-E8-80-88-B9-55-2D
NetBIOS over Tcpip. : Enabled

Ethernet adapter VMware Network Adapter VMnet8:

```
Connection-specific DNS Suffix . : 
Description . . . . . : VMware Virtual Ethernet Adapter for VMnet8
Physical Address. . . . . : 00-50-56-C0-00-08
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::e825:5c19:4402:e3fa%24(Preferred)
IPv4 Address. . . . . : 192.168.111.1(Preferred)
Subnet Mask . . . . . : 255.255.255.0
Lease Obtained. . . . . : 2025年3月22日 19:16:54
Lease Expires . . . . . : 2025年3月22日 19:46:54
Default Gateway . . . . . : 
DHCP Server . . . . . : 192.168.111.254
DHCPv6 IAID . . . . . : 838881366
DHCPv6 Client DUID. . . . . : 00-01-00-01-2C-30-02-90-E8-80-88-B9-55-2D
Primary WINS Server . . . . . : 192.168.111.2
NetBIOS over Tcpip. . . . . : Enabled
```

Wireless LAN adapter WLAN:

```
Connection-specific DNS Suffix . : 
Description . . . . . : Intel(R) Wi-Fi 6E AX211 160MHz
Physical Address. . . . . : 58-1C-F8-2E-5B-1C
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . : 2408:8642:893:e745:7958:f396:d26a:8ae4(Preferred)
Temporary IPv6 Address. . . . . : 2408:8642:893:e745:a892:2043:f81d:e103(Preferred)
Link-local IPv6 Address . . . . . : fe80::f2f8:fd2c:7fc7:67e4%23(Preferred)
IPv4 Address. . . . . : 10.162.13.233(Preferred)
Subnet Mask . . . . . : 255.255.0.0
Lease Obtained. . . . . : 2025年3月22日 19:16:58
Lease Expires . . . . . : 2025年3月23日 19:17:30
Default Gateway . . . . . : fe80::763a:20ff:feb9:e802%23
                             10.162.0.1
DHCP Server . . . . . : 10.162.0.1
DHCPv6 IAID . . . . . : 290987256
DHCPv6 Client DUID. . . . . : 00-01-00-01-2C-30-02-90-E8-80-88-B9-55-2D
DNS Servers . . . . . : 10.10.0.21
                             10.10.2.21
NetBIOS over Tcpip. . . . . : Enabled
```

Ethernet adapter 蓝牙网络连接:

```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Bluetooth Device (Personal Area Network)
Physical Address. . . . . : 58-1C-F8-2E-5B-20
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
```

Ethernet adapter 以太网:

```
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Description . . . . . : Realtek PCIe GbE Family Controller
Physical Address. . . . . : E8-80-88-B9-55-2D
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
```

2、netstat 网络连接分析

(1) As Client

```
C:\Windows\System32>netstat -ano | findstr ESTABLISHED
```

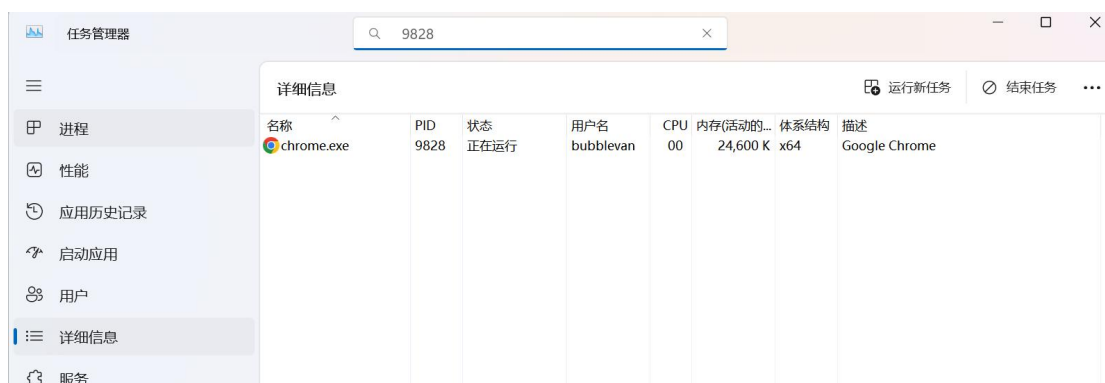
TCP	10.162.208.50:50128	111.62.37.139:443	ESTABLISHED	9828
TCP	10.162.208.50:50612	101.126.37.72:443	ESTABLISHED	16576
TCP	10.162.208.50:50615	101.126.37.72:443	ESTABLISHED	16576
TCP	10.162.208.50:51465	10.204.4.14:443	ESTABLISHED	9828
TCP	10.162.208.50:51470	104.18.19.125:443	ESTABLISHED	17352
TCP	10.162.208.50:53041	52.45.169.216:443	ESTABLISHED	39880
TCP	10.162.208.50:53082	3.214.5.159:443	ESTABLISHED	39880
TCP	10.162.208.50:53476	117.149.155.251:443	ESTABLISHED	20728
TCP	10.162.208.50:53557	52.45.169.216:443	ESTABLISHED	39880
TCP	10.162.208.50:53561	111.62.37.139:443	ESTABLISHED	9828
TCP	10.162.208.50:53933	118.178.181.20:443	ESTABLISHED	67376
TCP	10.162.208.50:54544	111.3.79.213:443	ESTABLISHED	9828
TCP	10.162.208.50:55016	10.204.4.15:443	ESTABLISHED	9828
TCP	10.162.208.50:55102	10.204.4.15:443	ESTABLISHED	9828
TCP	10.162.208.50:55441	139.224.56.191:443	ESTABLISHED	51152
TCP	10.162.208.50:55445	64.233.188.188:5228	ESTABLISHED	9828
TCP	10.162.208.50:55446	20.243.74.193:443	ESTABLISHED	5404
TCP	10.162.208.50:55449	111.62.37.139:443	ESTABLISHED	51152
TCP	10.162.208.50:55732	54.146.55.63:443	ESTABLISHED	39880
TCP	10.162.208.50:55842	183.194.189.242:443	ESTABLISHED	20572
TCP	10.162.208.50:56050	3.214.5.159:443	ESTABLISHED	17352
TCP	10.162.208.50:56987	146.56.252.202:443	ESTABLISHED	15076
TCP	10.162.208.50:57520	120.27.153.100:443	ESTABLISHED	20348
TCP	10.162.208.50:59617	140.82.114.25:443	ESTABLISHED	9828
TCP	10.162.208.50:59973	183.201.125.167:443	ESTABLISHED	9828

```
C:\Windows\System32>tasklist /fi "PID eq 9828"
```

映像名称	PID	会话名	会话#	内存使用
chrome.exe	9828	Console	1	64,696 K

以第一条为例，其输出的连接状态是 ESTABLISHED，表示连接已建立，数据正在传输。对应的进程名称是 chrome.exe，目标端口是 443（HTTPS），由谷歌浏览器发起。

其源地址和源端口为 10.162.208.50:50128，目的地址、端口是 111.62.37.139:443。



```
C:\Windows\System32>netstat -bano | findstr CLOSE_WAIT
TCP    10.162.208.50:49830    203.107.1.35:80      CLOSE_WAIT    39548
TCP    10.162.208.50:50123    117.149.155.251:443  CLOSE_WAIT    20728
TCP    10.162.208.50:53465    109.244.145.199:443  CLOSE_WAIT    32552
TCP    10.162.208.50:54029    23.50.82.235:443     CLOSE_WAIT    46328
TCP    10.162.208.50:56819    54.213.26.180:443    CLOSE_WAIT    63016
TCP    10.162.208.50:57521    203.107.1.33:80      CLOSE_WAIT    39548
TCP    10.162.208.50:59281    1.92.89.108:443      CLOSE_WAIT    6828
TCP    [2408:8642:893:e745:4818:8641:e70d:bea8]:59278 [2409:8c28:5460:203:1::f3]:443 CLOSE_WAIT    6828
TCP    [2408:8642:893:e745:9095:c853:86fc:d19d]:58423 [2401:1d40:f21:2900::57c]:443 CLOSE_WAIT    6828
TCP    [2408:8642:893:e745:9095:c853:86fc:d19d]:58649 [2409:8c28:5460:203:1::f3]:443 CLOSE_WAIT    9988
TCP    [2408:8642:893:e745:9095:c853:86fc:d19d]:59072 [2401:1d40:f21:2900::57c]:443 CLOSE_WAIT    7184
TCP    [2408:8642:893:e745:9095:c853:86fc:d19d]:63996 [2409:8c28:5460:203:1::f3]:443 CLOSE_WAIT    16576
TCP    [2408:8642:893:e745:9095:c853:86fc:d19d]:64641 [2401:1d40:f21:2900::57c]:443 CLOSE_WAIT    384

C:\Windows\System32>tasklist /fi "PID eq 39548"

映像名称                PID 会话名                会话#    内存使用
=====
DingTalk.exe             39548 Console                1        682,740 K
```

然后是 CLOSE_WAIT 状态，仍然以第一条为例，是钉钉的服务，我这里已经将其最小化到托盘。

源地址、源端口：10.162.208.50:49830（本地 Client 的 IP 和动态端口）

目的地址、目的端口：203.107.1.35:80（远程 Server 的 HTTP 端口）

应用程序：DingTalk.exe（钉钉作为 Client 发起连接，未正确关闭导致 CLOSE_WAIT）

状态解析：远程 Server 已关闭连接（发送 FIN），钉钉回复 ACK 后未主动关闭，处于等待释放状态。

```
C:\Windows\System32>tasklist /fi "PID eq 39548"

映像名称                PID 会话名                会话#    内存使用
=====
DingTalk.exe             39548 Console                1        682,740 K

C:\Windows\System32>netstat -bano | findstr SYN_SENT
TCP    10.162.208.50:54605    142.250.198.78:443    SYN_SENT      9828
TCP    10.162.208.50:54606    142.250.198.78:443    SYN_SENT      9828
TCP    10.162.208.50:64755    192.168.111.128:8481  SYN_SENT      5404
TCP    10.162.208.50:65418    142.250.66.74:443     SYN_SENT      9828
TCP    10.162.208.50:65419    142.250.66.74:443     SYN_SENT      9828
TCP    127.0.0.1:64754        127.0.0.1:9229        SYN_SENT      36448

C:\Windows\System32>tasklist /fi "PID eq 9828"

映像名称                PID 会话名                会话#    内存使用
=====
chrome.exe               9828 Console                1        64,724 K

C:\Windows\System32>tasklist /fi "PID eq 5404"

映像名称                PID 会话名                会话#    内存使用
=====
msedge.exe               5404 Console                1        81,028 K
```

Chrome/Edge 打开新页面时，会并发发起多条 SYN 请求，短暂处于 SYN_SENT 状态后会转为 ESTABLISHED。目前的 SYN_WAIT 表面本地程序(Client)正在尝试与远程服务器建立 TCP 连接，已发送 SYN（同步请求）包，但未收到

服务器的 SYN-ACK（同步确认）响应。本质上处于 TCP 三次握手的第一阶段（Client → Server: SYN）。

(2) As Server

```
C:\Windows\System32>netstat -bano | findstr LISTENING
TCP    0.0.0.0:135          0.0.0.0:0          LISTENING        1764
TCP    0.0.0.0:445          0.0.0.0:0          LISTENING         4
TCP    0.0.0.0:902          0.0.0.0:0          LISTENING       13464
TCP    0.0.0.0:912          0.0.0.0:0          LISTENING       13464
TCP    0.0.0.0:2343         0.0.0.0:0          LISTENING       7800
TCP    0.0.0.0:3240         0.0.0.0:0          LISTENING      13796
TCP    0.0.0.0:3306         0.0.0.0:0          LISTENING       8376
TCP    0.0.0.0:3580         0.0.0.0:0          LISTENING       7036
TCP    0.0.0.0:3582         0.0.0.0:0          LISTENING      11072
TCP    0.0.0.0:4523         0.0.0.0:0          LISTENING      26456
TCP    0.0.0.0:5040         0.0.0.0:0          LISTENING      13812
TCP    0.0.0.0:7680         0.0.0.0:0          LISTENING       4268
TCP    0.0.0.0:8080         0.0.0.0:0          LISTENING      13140
TCP    0.0.0.0:8082         0.0.0.0:0          LISTENING      80548
TCP    0.0.0.0:8090         0.0.0.0:0          LISTENING      20348
TCP    0.0.0.0:13473        0.0.0.0:0          LISTENING      70176
TCP    0.0.0.0:19991        0.0.0.0:0          LISTENING       7248
TCP    0.0.0.0:32123        0.0.0.0:0          LISTENING      25308
TCP    0.0.0.0:33060        0.0.0.0:0          LISTENING       8376
TCP    0.0.0.0:48080        0.0.0.0:0          LISTENING       8796
TCP    0.0.0.0:49664        0.0.0.0:0          LISTENING       1508
TCP    0.0.0.0:49665        0.0.0.0:0          LISTENING       1392
TCP    0.0.0.0:49670        0.0.0.0:0          LISTENING       3096
TCP    0.0.0.0:49671        0.0.0.0:0          LISTENING       4196
TCP    0.0.0.0:49678        0.0.0.0:0          LISTENING       6372
TCP    0.0.0.0:49723        0.0.0.0:0          LISTENING       9152
TCP    0.0.0.0:49796        0.0.0.0:0          LISTENING       1472

C:\Windows\System32>tasklist /fi "PID eq 1704"

映像名称                PID 会话名                会话#    内存使用
=====
WUDFHost.exe            1704 Services                0        8,496 K
```

同样以第一条为例：

源地址、源端口：0.0.0.0:135（监听所有 IP 地址的 135 端口）

目的地址、端口：0.0.0.0:0（表示接受所有目的地址的连接）

连接状态：LISTENING（服务器正在等待连接）

接受侦听的应用程序：PID 1764 对应 WUDFHost.exe（Windows 用户模式驱动框架服务）

TCP	0.0.0.0:59113	0.0.0.0:0	LISTENING	13268
TCP	10.10.10.6:139	0.0.0.0:0	LISTENING	4
TCP	10.162.208.50:139	0.0.0.0:0	LISTENING	4
TCP	127.0.0.1:1234	0.0.0.0:0	LISTENING	9008
TCP	127.0.0.1:4709	0.0.0.0:0	LISTENING	16576
TCP	127.0.0.1:5283	0.0.0.0:0	LISTENING	20572
TCP	127.0.0.1:5354	0.0.0.0:0	LISTENING	7004
TCP	127.0.0.1:8440	0.0.0.0:0	LISTENING	39548
TCP	127.0.0.1:8680	0.0.0.0:0	LISTENING	11068
TCP	127.0.0.1:9080	0.0.0.0:0	LISTENING	7156
TCP	127.0.0.1:9876	0.0.0.0:0	LISTENING	9008
TCP	127.0.0.1:10000	0.0.0.0:0	LISTENING	13532
TCP	127.0.0.1:10001	0.0.0.0:0	LISTENING	55488
TCP	127.0.0.1:13013	0.0.0.0:0	LISTENING	11068
TCP	127.0.0.1:13016	0.0.0.0:0	LISTENING	11068
TCP	127.0.0.1:14456	0.0.0.0:0	LISTENING	25308
TCP	127.0.0.1:15644	0.0.0.0:0	LISTENING	9152
TCP	127.0.0.1:27015	0.0.0.0:0	LISTENING	7164
TCP	127.0.0.1:27017	0.0.0.0:0	LISTENING	7268
TCP	127.0.0.1:35600	0.0.0.0:0	LISTENING	15076
TCP	127.0.0.1:42950	0.0.0.0:0	LISTENING	72512
TCP	127.0.0.1:45413	0.0.0.0:0	LISTENING	25308
TCP	127.0.0.1:45819	0.0.0.0:0	LISTENING	70176
TCP	127.0.0.1:49687	0.0.0.0:0	LISTENING	8816
TCP	127.0.0.1:49690	0.0.0.0:0	LISTENING	8572
TCP	127.0.0.1:49691	0.0.0.0:0	LISTENING	8196
TCP	127.0.0.1:49743	0.0.0.0:0	LISTENING	8188
TCP	127.0.0.1:49838	0.0.0.0:0	LISTENING	7748
TCP	127.0.0.1:49942	0.0.0.0:0	LISTENING	16828
TCP	127.0.0.1:53484	0.0.0.0:0	LISTENING	70176

其余有源地址为 127.0.0.1 的，表明仅监听本地环回地址（仅限本机访问，外部无法连接）；具体 IP（如 10.162.208.50:139）则表示监听指定网卡的 IP 地址，这里 System 进程（PID 4）在监听多个端口。

3、tracert 路由跟踪

(1) tracert 校内主机

10.xxx.xxx.xxx 是私有 IP 地址（属于校园网内部地址），说明数据完全在浙江大学的校内网络内传输，没有经过互联网（公网）。个别跃点的 * 表示该设备未回复 ICMP 包。

```
C:\Users\bubblevan>tracert zupo.zju.edu.cn

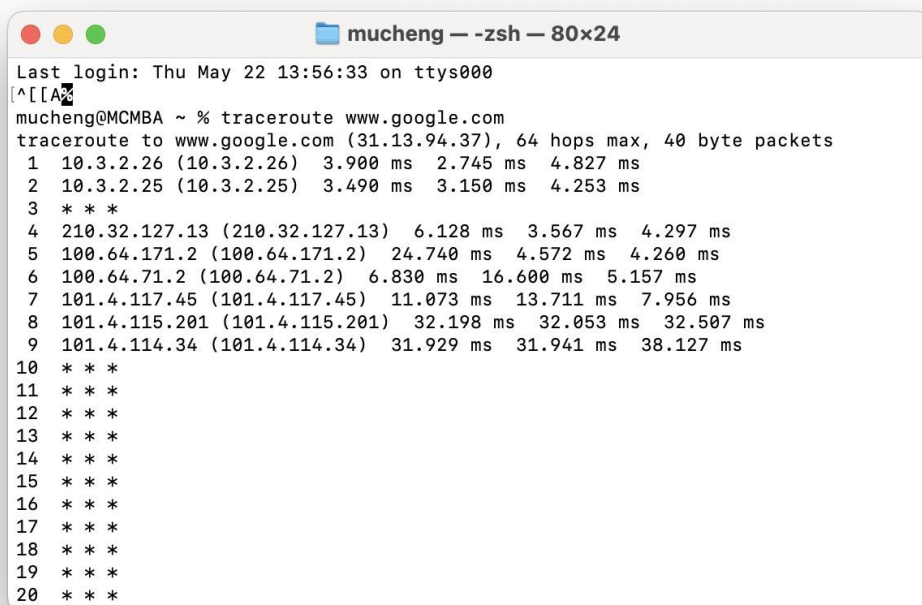
Tracing route to zupo.zju.edu.cn [10.203.4.70]
over a maximum of 30 hops:

  1      3 ms      4 ms      4 ms  10.3.2.29
  2     12 ms     5 ms     3 ms  10.3.2.30
  3      7 ms     3 ms     9 ms  10.3.1.13
  4      7 ms     5 ms     7 ms  10.3.1.22
  5     12 ms      *      *    10.3.7.74
  6     12 ms    11 ms    11 ms  10.203.4.70

Trace complete.
```

(2) V2rayN VPN

在实验过程中，我首先不使用 VPN 直接对 Google 和 Berkeley 大学网站进行 traceroute。对 Google 的追踪从本地网络开始，经过中国国内骨干网节点后，在出口处（101.4.114.34 之后）全部显示超时，无法完成追踪；



```
mucheng — zsh — 80x24
Last login: Thu May 22 13:56:33 on ttys000
mucheng@MCMBA ~ % traceroute www.google.com
traceroute to www.google.com (31.13.94.37), 64 hops max, 40 byte packets
 1  10.3.2.26 (10.3.2.26)  3.900 ms  2.745 ms  4.827 ms
 2  10.3.2.25 (10.3.2.25)  3.490 ms  3.150 ms  4.253 ms
 3  * * *
 4  210.32.127.13 (210.32.127.13)  6.128 ms  3.567 ms  4.297 ms
 5  100.64.171.2 (100.64.171.2)  24.740 ms  4.572 ms  4.260 ms
 6  100.64.71.2 (100.64.71.2)  6.830 ms  16.600 ms  5.157 ms
 7  101.4.117.45 (101.4.117.45)  11.073 ms  13.711 ms  7.956 ms
 8  101.4.115.201 (101.4.115.201)  32.198 ms  32.053 ms  32.507 ms
 9  101.4.114.34 (101.4.114.34)  31.929 ms  31.941 ms  38.127 ms
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
```

而对 Berkeley 大学的追踪则成功完成，经过 17 个跳转最终到达目标服务器（141.193.213.20）。


```

mucheng — -zsh — 80x25
traceroute: Warning: www.berkeley.edu has multiple addresses; using 141.193.213.20
traceroute to www.berkeley.edu (141.193.213.20), 64 hops max, 40 byte packets
 1  10.3.2.26 (10.3.2.26)  9.823 ms  3.043 ms  6.685 ms
 2  10.3.2.25 (10.3.2.25)  14.525 ms  3.314 ms  5.955 ms
 3  * * *
 4  210.32.127.13 (210.32.127.13)  5.167 ms  7.801 ms  4.637 ms
 5  100.64.171.2 (100.64.171.2)  4.942 ms  4.657 ms  4.825 ms
 6  100.64.71.2 (100.64.71.2)  4.544 ms  4.945 ms  5.472 ms
 7  101.4.117.45 (101.4.117.45)  8.983 ms  12.625 ms  9.353 ms
 8  * * *
 9  101.4.115.250 (101.4.115.250)  33.409 ms  33.007 ms
   101.4.115.146 (101.4.115.146)  37.028 ms
10  * * *
11  101.4.114.194 (101.4.114.194)  34.018 ms  40.984 ms  35.188 ms
12  101.4.114.182 (101.4.114.182)  69.490 ms  69.648 ms  67.659 ms
13  hnk-b4-link.ip.twelve99.net (80.239.130.172)  67.058 ms * 70.650 ms
14  * hnk-b3-link.ip.twelve99.net (62.115.143.241)  74.013 ms  71.542 ms
15  cloudflare-ic-371889.ip.twelve99-cust.net (213.248.84.113)  89.256 ms  85.33
   8 ms  71.340 ms
16  103.22.203.71 (103.22.203.71)  70.429 ms
   103.22.203.25 (103.22.203.25)  73.654 ms
   103.22.203.29 (103.22.203.29)  73.223 ms
17  141.193.213.20 (141.193.213.20)  67.868 ms  71.646 ms  67.692 ms
mucheng@MCMBA ~ %

```

随后我启用了 flyingbird VPN 的 TUN 模式再次进行测试，出乎意料的是，对 Berkeley 大学的追踪结果与未使用 VPN 时几乎完全一致，同样经过国内骨干网络，最终到达相同的目标 IP。

```

mucheng — -zsh — 80x25
mucheng@MCMBA ~ % traceroute www.berkeley.edu
traceroute: Warning: www.berkeley.edu has multiple addresses; using 141.193.213.21
traceroute to www.berkeley.edu (141.193.213.21), 64 hops max, 40 byte packets
 1  10.3.2.26 (10.3.2.26)  4.261 ms  3.444 ms *
 2  10.3.2.25 (10.3.2.25)  3.430 ms  2.890 ms  2.996 ms
 3  * * *
 4  210.32.127.13 (210.32.127.13)  5.526 ms  4.752 ms  5.102 ms
 5  100.64.171.2 (100.64.171.2)  5.667 ms  4.012 ms  3.588 ms
 6  100.64.71.2 (100.64.71.2)  4.283 ms  3.838 ms  4.467 ms
 7  101.4.117.45 (101.4.117.45)  12.761 ms  8.123 ms  8.781 ms
 8  101.4.115.201 (101.4.115.201)  32.205 ms  32.367 ms *
 9  * * *
10  * * *
11  101.4.114.194 (101.4.114.194)  37.117 ms  33.408 ms  33.838 ms
12  101.4.114.182 (101.4.114.182)  71.900 ms  72.449 ms  69.030 ms
13  hnk-b4-link.ip.twelve99.net (80.239.130.172)  69.284 ms  66.121 ms *
14  hnk-b3-link.ip.twelve99.net (62.115.143.241)  67.258 ms  67.270 ms  67.834 m
   s
15  * * cloudflare-ic-376333.ip.twelve99-cust.net (62.115.165.191)  67.999 ms
16  103.22.203.27 (103.22.203.27)  67.723 ms
   103.22.203.227 (103.22.203.227)  67.691 ms
   103.22.203.231 (103.22.203.231)  78.756 ms
17  141.193.213.21 (141.193.213.21)  66.741 ms  66.316 ms  67.111 ms
mucheng@MCMBA ~ %

```

这一发现与预期不符——理论上 VPN 应该改变网络路由路径，但实验表明即使开启 TUN 模式，traceroute 的路径基本未发生变化。为了对比验证，我们通过海外服务器对相同目标进行 traceroute。结果显示对 Google 的追踪非常顺畅，延迟普遍较低；对 Berkeley 大学的追踪则路径更短（仅 11 跳），通过加拿大节点直接到达目标，整体网络性能明显优于国内访问。

```
mucheng — admin_qcy@SanakaServer: ~ — ssh SanakaServer — 80x24
Disk Usage: 20G / 71G
Last login: Sun Apr 27 23:14:32 2025 from 2408:8642:893:8ec8:9157:30cc:cb16:4c85
admin_qcy@SanakaServer:~$ traceroute www.google.com
traceroute to www.google.com (142.250.69.132), 30 hops max, 60 byte packets
 1 10.3.1.1 (10.3.1.1) 0.402 ms 0.373 ms 0.350 ms
 2 * * *
 3 10.161.129.124 (10.161.129.124) 1.903 ms 1.885 ms 1.863 ms
 4 10.34.97.12 (10.34.97.12) 0.604 ms 10.34.97.10 (10.34.97.10) 0.558 ms 0.646 ms
 5 10.74.8.132 (10.74.8.132) 0.342 ms 10.74.8.130 (10.74.8.130) 1.758 ms 10.74.8.66 (10.74.8.66) 0.297 ms
 6 10.95.81.10 (10.95.81.10) 1.266 ms 10.95.81.8 (10.95.81.8) 0.573 ms 10.95.81.10 (10.95.81.10) 1.009 ms
 7 * * ymq-mtl3-sbb2-8k.qc.ca (142.44.208.174) 2.532 ms
 8 10.200.3.7 (10.200.3.7) 2.265 ms 10.200.3.1 (10.200.3.1) 2.945 ms 2.984 ms
 9 * * *
10 192.178.86.251 (192.178.86.251) 1.406 ms 192.178.86.183 (192.178.86.183) 2.795 ms 192.178.86.185 (192.178.86.185) 2.745 ms
11 172.253.77.243 (172.253.77.243) 25.257 ms 172.253.77.117 (172.253.77.117) 24.591 ms 172.253.77.243 (172.253.77.243) 23.844 ms
12 tzvula-ab-in-f4.1e100.net (142.250.69.132) 1.242 ms 1.219 ms 1.253 ms
admin_qcy@SanakaServer:~$
```

```
admin_qcy@SanakaServer:~$ traceroute www.berkeley.edu
traceroute to www.berkeley.edu (141.193.213.21), 30 hops max, 60 byte packets
 1 10.3.1.1 (10.3.1.1) 0.084 ms 0.056 ms 0.041 ms
 2 54.39.49.252 (54.39.49.252) 1.268 ms 1.272 ms 1.439 ms
 3 10.161.129.124 (10.161.129.124) 0.203 ms 0.234 ms 10.161.129.125 (10.161.129.125) 0.198 ms
 4 10.34.97.14 (10.34.97.14) 0.629 ms 10.34.97.10 (10.34.97.10) 0.707 ms 10.34.97.8 (10.34.97.8) 0.504 ms
 5 10.74.8.132 (10.74.8.132) 0.289 ms 10.74.8.174 (10.74.8.174) 0.233 ms 10.74.8.66 (10.74.8.66) 0.243 ms
 6 10.95.81.10 (10.95.81.10) 1.006 ms 1.353 ms 0.897 ms
 7 be101.yto-tr1-sbb1-8k.on.ca (192.99.146.162) 9.043 ms be101.yto-tr1-sbb2-8k.on.ca (192.99.146.49) 9.454 ms be101.yto-tr1-sbb1-8k.on.ca (192.99.146.162) 9.418 ms
 8 10.200.5.5 (10.200.5.5) 7.217 ms 10.200.5.9 (10.200.5.9) 7.172 ms 10.200.5.7 (10.200.5.7) 9.049 ms
 9 * * *
10 108.162.239.28 (108.162.239.28) 7.823 ms 7.769 ms 108.162.239.18 (108.162.239.18) 7.912 ms
11 141.193.213.21 (141.193.213.21) 7.536 ms 7.557 ms 7.717 ms
admin_qcy@SanakaServer:~$
```

针对上述现象，我查询了相关技术资料（见末尾参考资料）进行深入研究。根据 V2EX 上的讨论，问题的核心在于协议层级的不匹配：ICMP 是网络层（第三层）协议，而常见的 SOCKS5 代理是传输层（第四层）协议，两者本质上不兼容。

正如用户 "Badupp" 所指出："sock5 是四层，ICMP 是三层，怎么代理？" 关于 TUN 模式的局限性，用户 "specture"："clash 的 ping 应该是直接 return 了，ping

任何地址都是<1ms", 表明许多 TUN 实现并不真正处理 ICMP 包, 而是在本地模拟返回结果。VPN 类型的差异也是关键因素, 用户 "busier" 指出只有 "真正意义上的 VPN 才可以三层转发, 例如 PPTP、L2TP、IKEv2、OpenVPN、WireGuard 等", 而许多所谓的 "VPN" 工具实际上只是代理工具的封装。

Reddit 讨论进一步解释了这一现象: 当使用 VPN 时, traceroute 命令很可能绕过 VPN 隧道直接使用物理网络接口, 除非 VPN 客户端特别配置为处理 ICMP 流量。这解释了为什么实验中 TUN 模式 VPN 对 traceroute 结果几乎没有影响。

基于调研结果, 我尝试了几种可能的解决方案。首先考虑更换为真正工作在网络层的 VPN 协议 WireGuard, 这些协议理论上能够完全接管网络流量, 包括 ICMP 包。但是部署完毕后却完全连不上网了, 最终失败; 另一种思路是特殊处理 ICMP 包, 确保虚拟网卡正确配置为处理所有网络层流量, 但是依然是同样的结果。

不过我相信直接通过海外服务器去 tracert 一样能达到老师最开始想要我们通过 VPN 学习 tracert 的效果。

(3) 校外上网

通过手机热点实验。

```
C:\Windows\System32>tracert www.baidu.com

通过最多 30 个跃点跟踪
到 www.a.shifen.com [2409:8c20:6:1794:0:ff:b080:87f0] 的路由:

 1    15 ms    13 ms     8 ms    2409:8929:96e:4557:4b87:6c68:b7dd:ac28
 2     *       *         *       请求超时。
 3     *       *       1495 ms  fc00:1000::161
 4    28 ms    48 ms     67 ms    2409:8028:0:ffffd::71
 5   208 ms    53 ms     55 ms    2409:8028:0:ffffd::301
 6    46 ms    41 ms     *       2409:8028:0:ffffd::600
 7    35 ms    *         45 ms    2409:8080:0:2:1105:1151:300:0
 8    56 ms    37 ms     40 ms    2409:8080:0:1:405:1105::
 9   748 ms    39 ms     32 ms    2409:8080:0:2:405:463:1800:1
10    49 ms    46 ms     61 ms    2409:8020:3015:604::1
11    74 ms    73 ms     53 ms    2409:8020:3015:813::3
12   180 ms    59 ms     49 ms    240c:4001:2130::ec2:eb1:2
13    62 ms    40 ms     38 ms    240c:4051:1229:1eb:leaf:1:eb01:8
14    43 ms    68 ms     64 ms    240c:4051:1229:108:leaf:1:1b08:3
15   474 ms    63 ms     36 ms    2409:8c20:6:1794:0:ff:b080:87f0

跟踪完成。
```

对于 baidu, 网络请求从本地网络出发, 经过多个移动网络的节点, 最终到

达目标服务器。

第 1 跃点是手机热点分配的本地 IPv6 网关，跳数 2 请求超时，可能是因为该节点的路由器配置了丢弃 ICMP 数据包，或者网络存在拥堵等。后续经过 2409:8028、2409:8080 等运营商 IPv6 骨干节点，最终到达百度服务器。

```
C:\Windows\System32>tracert www.facebook.com

通过最多 30 个跃点跟踪
到 www.facebook.com [2a03:2880:f11c:8183:face:b00c:0:25de] 的路由:

 1      7 ms      3 ms      3 ms  2409:8929:96e:4557:4b87:6c68:b7dd:ac28
 2      *        *        *      请求超时。
 3     51 ms     38 ms     27 ms  fc00:1000::162
 4     57 ms     57 ms     34 ms  2409:8028:0:ffffd::73
 5     57 ms     47 ms     80 ms  2409:8028:0:ffffd::303
 6      *       236 ms     34 ms  2409:8028:0:ffffd::602
 7     65 ms     52 ms     40 ms  2409:8028:0:1::7094
 8     44 ms     33 ms     49 ms  2409:8080:0:2:1105:1171:1:0
 9    188 ms     51 ms     33 ms  2409:8080:0:1:1105:2c2::
10      *        *        *      请求超时。
11      *        *        *      请求超时。
12      *        *        *      请求超时。
13      *        *        *      请求超时。
14      *        *        *      请求超时。
15      *        *        *      请求超时。
16      *        *        *      请求超时。
17      *        *        *      请求超时。
18      *        *        *      请求超时。
19      *        *        *      请求超时。
20      *        *        *      请求超时。
21      *        *        *      请求超时。
22      *        *        *      请求超时。
23      *        *        *      请求超时。
24      *        *        *      请求超时。
25      *        *        *      请求超时。
26      *        *        *      请求超时。
27      *        *        *      请求超时。
28      *        *        *      请求超时。
29      *        *        *      请求超时。
30      *        *        *      请求超时。

跟踪完成。
```

facebook 前 6 跃点与百度类似（经过本地网关和运营商 IPv6 节点），但从第 10 跃点开始持续超时，最终未到达目标。推测 Facebook 在国内属于被屏蔽的网站，流量可能在出境节点（如运营商国际出口）被防火墙拦截，导致后续路由中断。

```
C:\Windows\System32>tracert yandex.com

通过最多 30 个跃点跟踪
到 yandex.com [2a02:6b8:a::a] 的路由:

 1      5 ms      9 ms      8 ms  2409:8929:96e:4557:4b87:6c68:b7dd:ac28
 2      *        *        *      请求超时。
 3      *        *        *      请求超时。
 4     66 ms     46 ms     54 ms  2409:8028:0:ffffd::71
 5     93 ms     54 ms     58 ms  2409:8028:0:ffffd::301
 6      *        86 ms     45 ms  2409:8028:0:ffffd::600
 7     61 ms     66 ms     44 ms  2409:8028:0:3::7020
 8     60 ms      *        69 ms  2409:8080:0:2:1106:1173:1:0
 9     64 ms     47 ms     63 ms  2409:8080:0:1:1106:2c1::
10    243 ms     47 ms     62 ms  2409:8080:0:4:2c5:2f5:2:0
11      *        *        *      请求超时。
12   202 ms    248 ms    213 ms  2402:4f00:1000:100::221
13      *        *        *      请求超时。
14      *        *        *      请求超时。
15      *        *        *      请求超时。
16   318 ms    354 ms    659 ms  2400:8800:f000:2::aa
17   334 ms    577 ms    336 ms  2a0e:fd80:2:4::1
18      *        *        *      请求超时。
19      *        *        *      请求超时。
20      *        *        *      请求超时。
21   359 ms    346 ms    372 ms  yandex.ru [2a02:6b8:a::a]

跟踪完成。
```

yandex 前 9 跃点与百度、Facebook 类似，经过运营商 IPv6 骨干网。第 10-19 跃点多次超时，但最终在第 21 跃点到达目标（延迟较高，300-700ms）。Yandex 服务器位于俄罗斯，路由需经过更多国际节点，部分节点（尤其是跨国链路）可能禁用 ICMP 响应，导致中间跃点超时。

(4) WSL

使用 Ubuntu 24.04.1 进行测试。

```

(base) bubblevan@Bubbles:~$ traceroute zupo.zju.edu.cn
traceroute to zupo.zju.edu.cn (10.203.4.70), 30 hops max, 60 byte packets
 1 Bubbles.mshome.net (172.24.160.1) 0.223 ms 0.212 ms 0.210 ms
 2 10.3.2.29 (10.3.2.29) 14.452 ms 14.449 ms 14.446 ms
 3 10.3.2.30 (10.3.2.30) 15.308 ms 15.305 ms 15.303 ms
 4 10.3.1.13 (10.3.1.13) 14.415 ms 14.389 ms 14.387 ms
 5 10.3.1.18 (10.3.1.18) 14.400 ms 14.399 ms 10.3.1.22 (10.3.1.22) 14.397 ms
 6 10.3.7.74 (10.3.7.74) 22.382 ms 21.407 ms *
 7 * * *
 8 * * *
 9 * * *
10 * * *
11 * * *
12 * * *
13 * * *
14 * * *
15 * * *
16 * * *
17 * * *
18 * * *
19 * * *
20 * * *
21 * * *
22 * * *
23 * * *
24 * * *
25 * * *
26 * * *
27 * * *
28 * * *
29 * * *
30 * * *

```

对比下来，Windows tracert 完整追踪到了目标，共 6 跳，路径：10.3.2.29 → 10.3.2.30 → 10.3.1.13 → 10.3.1.22 → 10.3.7.74 → 10.203.4.70

但是 Ubuntu traceroute 起始 IP 为 172.24.160.1（WSL 虚拟网络地址），部分 IP 与 Windows 追踪不同，延迟明显较高，无法完成追踪，第 7 跳后全部超时。

主要原因是 Ubuntu 是在 WSL 环境下运行，通过虚拟网络适配器连接，而 WSL 使用 NAT 连接到主机网络，导致网络路径不同。

4、ARP 地址解析

```
C:\Users\bubblevan>arp -a
```

```
接口: 10.10.10.6 --- 0x6
```

Internet 地址	物理地址	类型
10.10.10.5	00-ff-10-b0-86-4c	动态
10.10.10.7	ff-ff-ff-ff-ff-ff	静态
224.0.0.22	01-00-5e-00-00-16	静态
224.0.0.251	01-00-5e-00-00-fb	静态
224.0.0.252	01-00-5e-00-00-fc	静态
239.255.255.250	01-00-5e-7f-ff-fa	静态
255.255.255.255	ff-ff-ff-ff-ff-ff	静态

```
接口: 192.168.132.1 --- 0x12
```

Internet 地址	物理地址	类型
192.168.132.254	00-50-56-f5-d8-36	动态
192.168.132.255	ff-ff-ff-ff-ff-ff	静态
224.0.0.22	01-00-5e-00-00-16	静态
224.0.0.251	01-00-5e-00-00-fb	静态
224.0.0.252	01-00-5e-00-00-fc	静态
239.255.255.250	01-00-5e-7f-ff-fa	静态
255.255.255.255	ff-ff-ff-ff-ff-ff	静态

```
接口: 10.162.208.50 --- 0x17
```

Internet 地址	物理地址	类型
10.162.0.1	74-3a-20-b9-e8-02	动态
10.162.255.255	ff-ff-ff-ff-ff-ff	静态
224.0.0.22	01-00-5e-00-00-16	静态
224.0.0.251	01-00-5e-00-00-fb	静态
224.0.0.252	01-00-5e-00-00-fc	静态
239.255.255.250	01-00-5e-7f-ff-fa	静态
255.255.255.255	ff-ff-ff-ff-ff-ff	静态

```
接口: 192.168.111.1 --- 0x18
```

Internet 地址	物理地址	类型
192.168.111.254	00-50-56-f0-c2-29	动态
192.168.111.255	ff-ff-ff-ff-ff-ff	静态
224.0.0.22	01-00-5e-00-00-16	静态
224.0.0.251	01-00-5e-00-00-fb	静态
224.0.0.252	01-00-5e-00-00-fc	静态
239.255.255.250	01-00-5e-7f-ff-fa	静态
255.255.255.255	ff-ff-ff-ff-ff-ff	静态

动态项: 通过 ARP 协议自动获取的地址 (如网关), 一段时间后会过期。

静态项: 手动添加或系统固定的地址 (如环回地址), 永久存在。

抓住自习的同桌同学，获取其 ipconfig 的 IP 地址后：

```
C:\Users\bubblevan>ping 10.162.52.19

正在 Ping 10.162.52.19 具有 32 字节的数据:
来自 10.162.52.19 的回复: 字节=32 时间=50ms TTL=63
来自 10.162.52.19 的回复: 字节=32 时间=46ms TTL=63
来自 10.162.52.19 的回复: 字节=32 时间=27ms TTL=63
来自 10.162.52.19 的回复: 字节=32 时间=27ms TTL=63

10.162.52.19 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 27ms, 最长 = 50ms, 平均 = 37ms
```

接口: 10.162.208.50 --- 0x17		
Internet 地址	物理地址	类型
10.162.0.1	74-3a-20-b9-e8-02	动态
10.162.52.19	74-3a-20-b9-e8-02	动态
10.162.255.255	ff-ff-ff-ff-ff-ff	静态
224.0.0.22	01-00-5e-00-00-16	静态
224.0.0.251	01-00-5e-00-00-fb	静态
224.0.0.252	01-00-5e-00-00-fc	静态
239.255.255.250	01-00-5e-7f-ff-fa	静态
255.255.255.255	ff-ff-ff-ff-ff-ff	静态

可见 ping 通了目标主机，触发了 ARP 请求，获取了对方的 MAC 地址。在真实网络接口（10.162.208.50）的表中，明确出现了 10.162.52.19 的动态项，证明访问有效。

四、讨论与心得

（一）实验心得

通过本次网络诊断工具实验，我对网络协议栈的工作机制有了更为深入的理解。ipconfig 命令展示了网络接口配置的复杂性，不仅仅是简单的 IP 地址，还包括物理地址、子网划分和网关配置等关键要素。这些配置共同构成了设备与网络通信的基础框架，尤其是通过分析虚拟网卡与物理网卡的配置差异，让我认识到现代网络架构中虚拟化技术的重要性。

netstat 命令的使用让我能够深入观察网络连接状态,理解 TCP/IP 协议栈的运行机制。通过分析不同的连接状态(如 ESTABLISHED、SYN_SENT、CLOSE_WAIT 等),我切实感受到了 TCP 协议的状态机模型在实际网络通信中的应用。尤其是当观察到钉钉客户端产生的 CLOSE_WAIT 状态时,更加理解了网络应用程序需要正确处理连接的关闭过程,否则可能导致资源浪费或连接泄漏。这种实践经验对于理解网络编程中的错误处理与资源管理至关重要。

tracert 实验部分带来了最出乎意料的发现。在使用 TUN 模式 VPN 进行路由跟踪时,发现其对 ICMP 协议的处理存在局限性,导致 VPN 开启前后的 tracert 结果几乎一致。通过查阅技术资料,我理解了网络协议分层机制的重要性——ICMP 属于网络层协议,而常见的 SOCKS5 代理工作在传输层,两者存在本质的层级不匹配。这解释了为什么许多轻量级"VPN"工具实际上无法改变 tracert 的路由路径,因为它们并未真正接管网络层流量。对比从海外服务器发起的 tracert 结果,更清晰地展示了不同网络环境下的路由优化差异,这种实际观察对于理解全球互联网的拓扑结构具有重要意义。

ARP 协议的实验让我亲身体验了网络层与数据链路层之间的交互机制。当 ping 一个此前未通信过的主机时,系统自动触发 ARP 请求获取物理地址,这一过程展示了协议栈各层次之间协同工作的精妙之处。随着新增 ARP 表项的出现,我认识到网络通信不仅需要正确的 IP 寻址,还依赖于准确的 MAC 地址解析,这种双层寻址机制保证了数据包从源到目的地的准确传递。通过多次实验观察 ARP 表的动态变化,我更深入地理解了该协议的工作原理及其在局域网通信中的核心作用。

在 VPN 与 tracert 的实验中遇到的困难促使我深入研究网络协议的分层设计。据 V2EX 和 Reddit 上的技术讨论,我了解到只有真正工作在网络层的 VPN 协议(如 WireGuard、OpenVPN 等)才能完全接管包括 ICMP 在内的所有网络流量。这种协议分层的严格边界既是计算机网络设计的精髓,也是使用工具时需要注意的技术限制。虽然尝试多种解决方案未能成功让 ICMP 通过代理,但这个"失败"本身就是一次宝贵的学习机会,它揭示了网络工具的底层实现细节,而不仅仅是表面的功能呈现。

总体而言,这次实验不只是简单地学习网络诊断工具的使用方法,更是通过

实践验证和探索网络协议栈的工作原理。从物理层的 MAC 地址，到网络层的 IP 寻址和路由选择，再到传输层的连接状态管理，每一层都以其独特方式发挥作用，共同支撑起复杂而高效的网络通信系统。通过亲手操作和分析数据，这些原本抽象的概念变得具体可感，加深了我对计算机网络从理论到实践的综合理解。这种实验驱动的学习方式不仅丰富了我的技术知识，也培养了解决实际网络问题的能力。

（二）参考资料

[怎么让 ICMP 协议走代理 - V2EX](#)

https://www.reddit.com/r/HomeNetworking/comments/rpjlba/how_does_traceroute_work_when_connected_to_a_vpn/?rdt=37942

<https://www.purewl.com/traceroute-vpn/#How Traceroute Works Over a VPN Tunnel>

<https://github.com/hwanz/SSR-V2ray-Trojan>