中国科学院大学

University of Chinese Academy of Sciences

# 《软件与系统安全》

## 实验 1A & 1B 讲解

助教：刘鹏

中国科学院大学 · UCAS

# Contents



- 实验内容介绍
  - 1A：嗅探器设计与实现
  - 1B：安全文件传输软件设计与实现
- 实验实施步骤
- 作业提交方式
  - 不要压缩为RAR、7Z 等格式，压缩包不要加密。
  - 命名方式见下页PPT
  - 所有代码必须开源，GitHub/Gitee/GitLab 有完整记录
- 实验考核及评分准则
  - 共有 59 人选课，最终将根据完成情况按排名打分。

# Commit Guides



1A：嗅探器设计与实现

| 文件夹 | 张三_202018008829001_EX1A |
|---|---|
| 1. 源码文件夹 | 张三_202018008829001_ex1A_src |
| 2. 演示视频 | 张三_202018008829001_ex1A_play.mp4 |
| 3. 文档 | 张三_202118008829001_ex1A_report.pdf/docx |

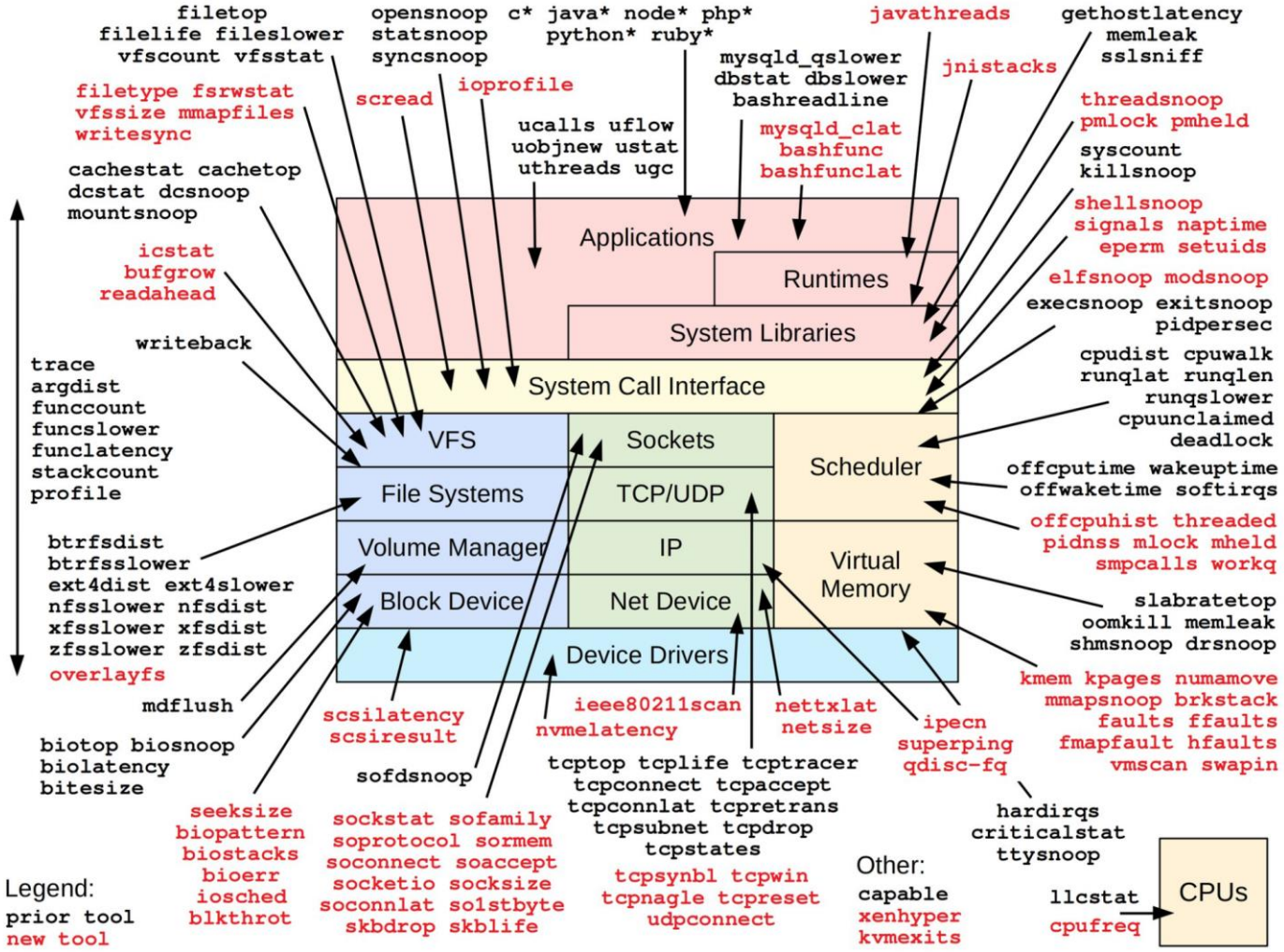
1B：安全文件传输软件设计与实现

| 文件夹 | 张三_202018008829001_EX1B |
|---|---|
| 1. 源码文件夹 | 张三_202018008829001_ex1B_src |
| 2. 演示视频 | 张三_202018008829001_ex1B_play.mp4 |
| 3. 文档 | 张三_202118008829001_ex1B_report.pdf/docx |

**无需 README，想说的全放在文档里即可。Git 链接必须写进去。**

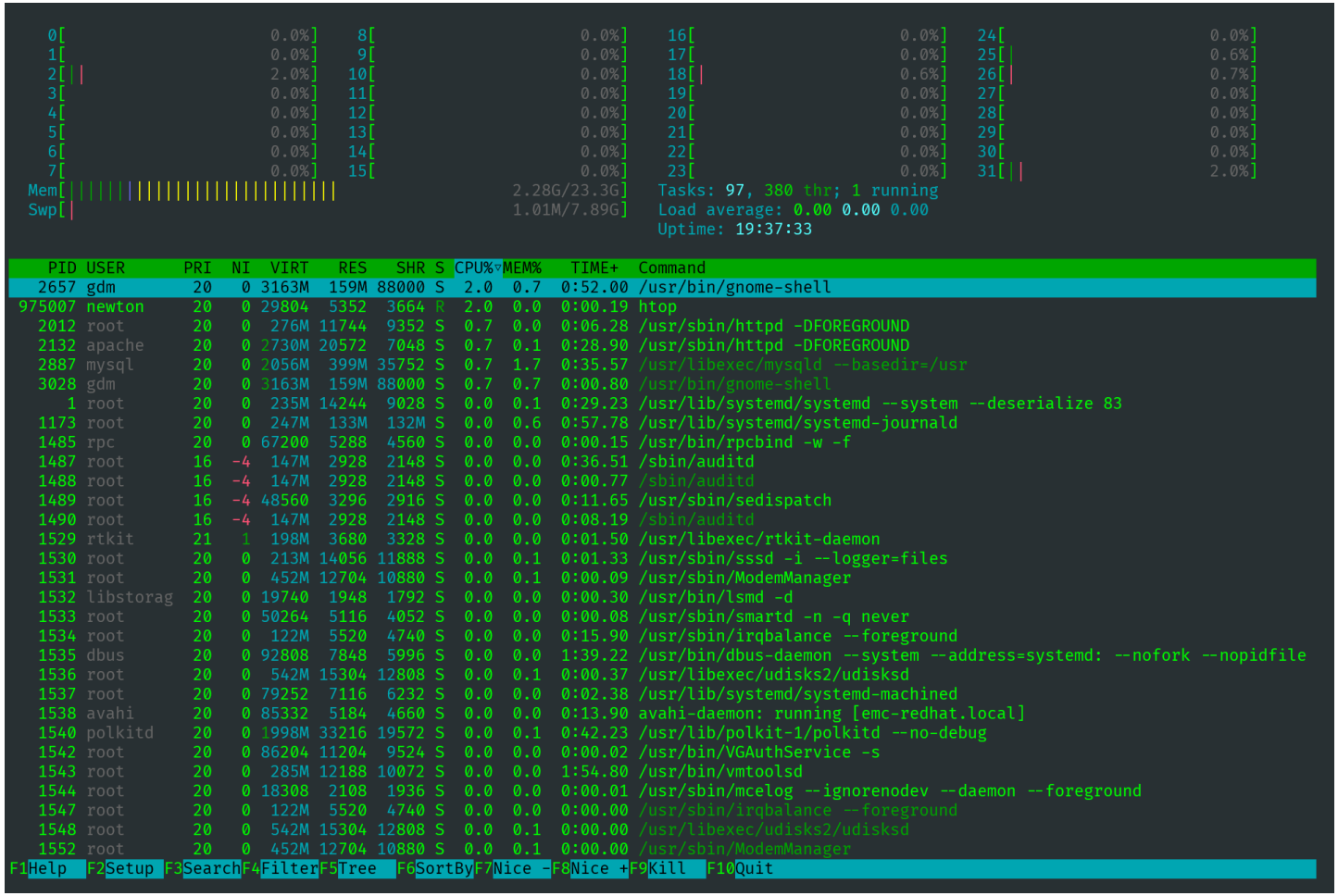
# 实验 1A：Introduction





# 实验 1A：Introduction






Linux htop 命令输出

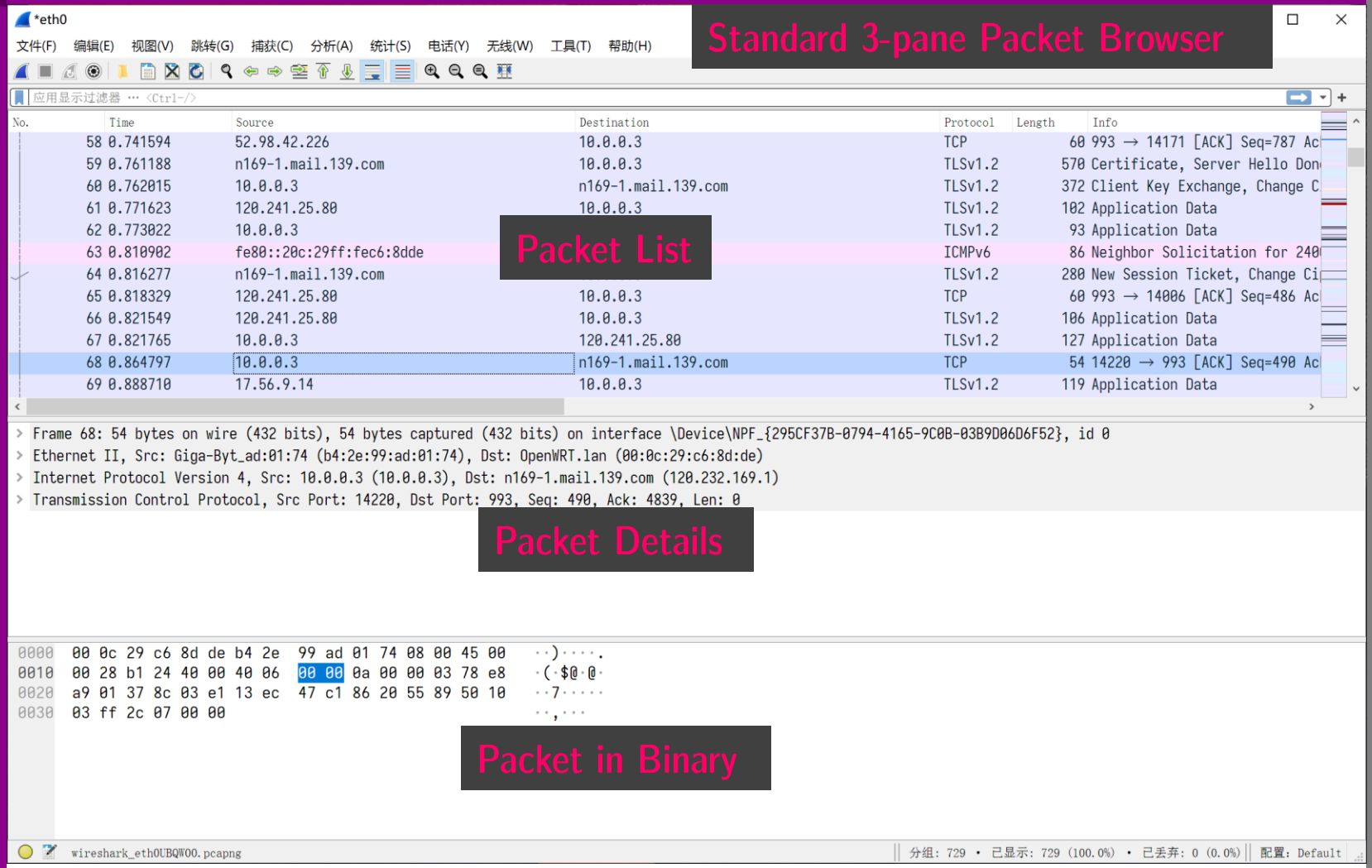
# 实验 1A：嗅探器设计与实现



- **网络嗅探器**
  - **介绍**：网络嗅探器，Network Packet Sniffer (NPS)，是一种专门用来进行网络流量侦听的工具。数据包嗅探器是研究网络行为学的基础工具。
  - 本次实验中的嗅探器与常见的 Wireshark 等工具不同
    - **Wireshark 等工具实现了强大的数据包协议分析功能，属于 Network Protocol Analyzer (NPA).**
    - **请各位思考，如何识别某种特定类型的应用层报文。**
- **温馨提示**
  - 实验中必须实现 NPS 功能（否则不及格）
  - 实验中尽量完善 NPA 功能（否则分不高）
  - 如果有同学可以基于 Wireshark 做良好的二次开发，也会有加分。

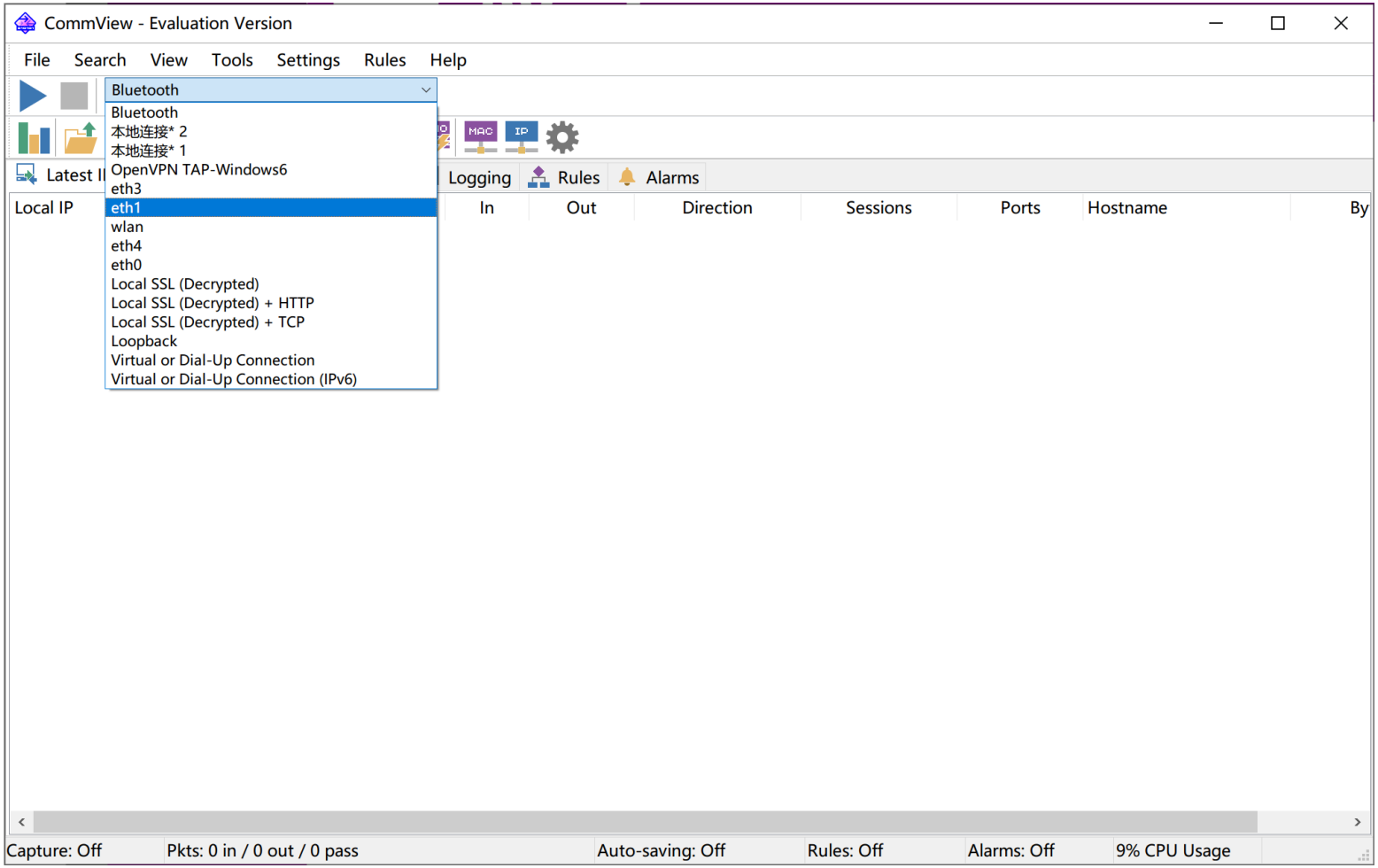
# Demo: Wireshark





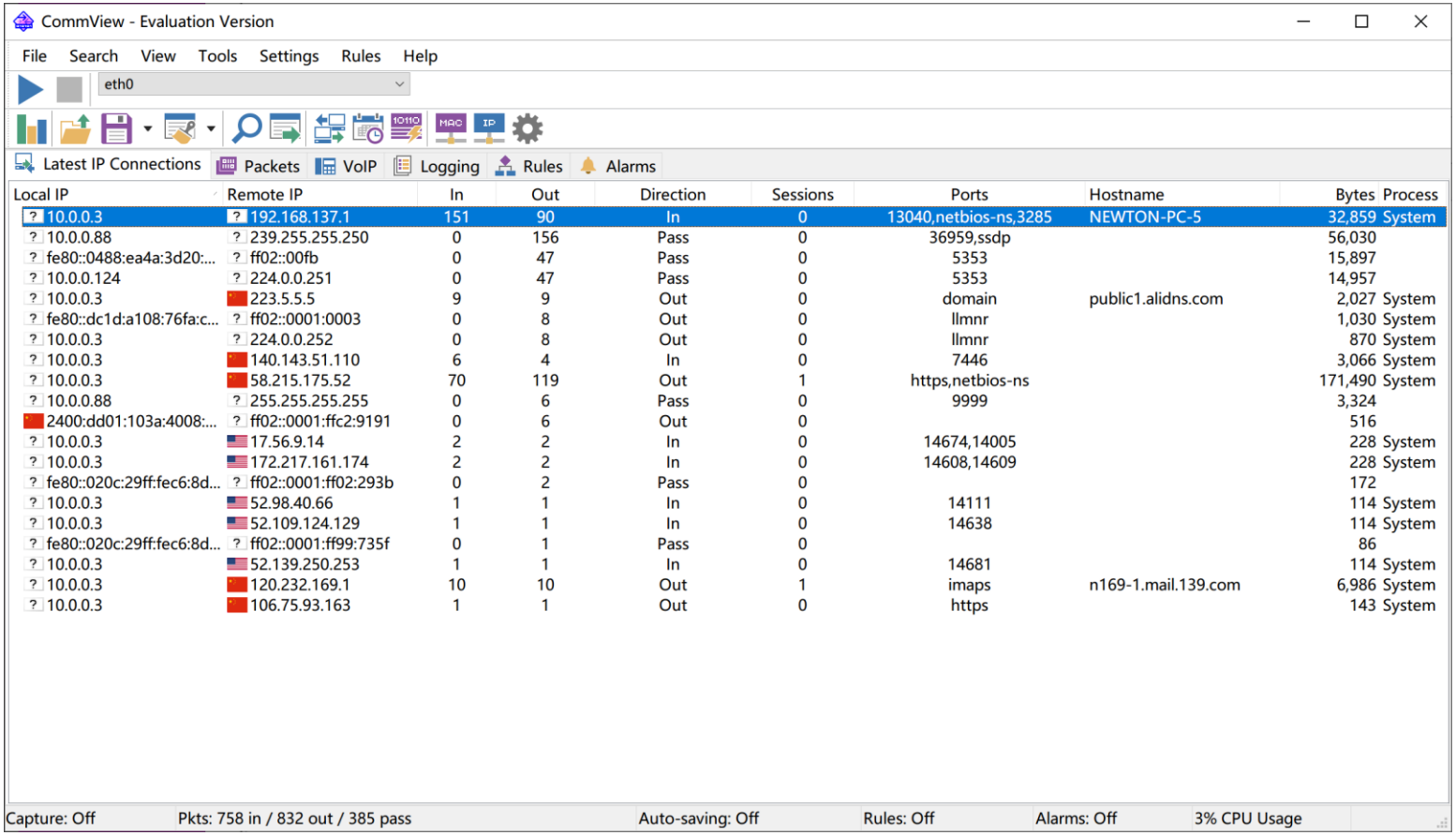
# Demo: CommView





# Demo: CommView





# 实验 1A 要求



- **务必 OSI 五层模型下的全部抓包**
  - **深入理解抓包层次与分析层次**
    - iOS 的 Thor App 可以在安装了系统描述文件之后，抓取 HTTP/HTTPS 的应用层包。同样因为安装了根证书性质的描述文件，故 Thor 能解析 HTTPS 中的内容。但具体到某些传输控制协议的内容，Thor 无能为力。这告诉了我们什么道理?
  - **要有一定的协议过滤能力**
    - Wireshark 等软件的协议过滤器支持逻辑演算，所以该软件里含有逻辑推导的组件。基本的协议过滤需要支持筛选 HTTP、TCP/UDP、IPv4/v6、ICMP 等不同类型、层次的数据包。libpcap 提供了数据包筛选功能。
  - **有一定的流追踪能力（加分项）**
    - 基于 IP+Port 的 TCP 流
    - 某进程产生的所有 TCP 流

# 实验 1A 提示



- **基于图形化编程的重要性**
  - 现代化的软件基本都提供图形化交互界面。
  - 图形化的界面对于分析流的时序图具有重要帮助作用。
  - 在现代图形 API 的帮助下开发图形化程序并不难。
  - 有关现代图形库的帮助，参见 Prompt II.
- **抓包的系统 API**
  - libpcap
  - winPcap: 基于 Windows NT 内核定制的 libpcap
  - **请大家自行学习上述 lib/dll 的用法，然后在代码中实现调用。**

# Linux 协议栈



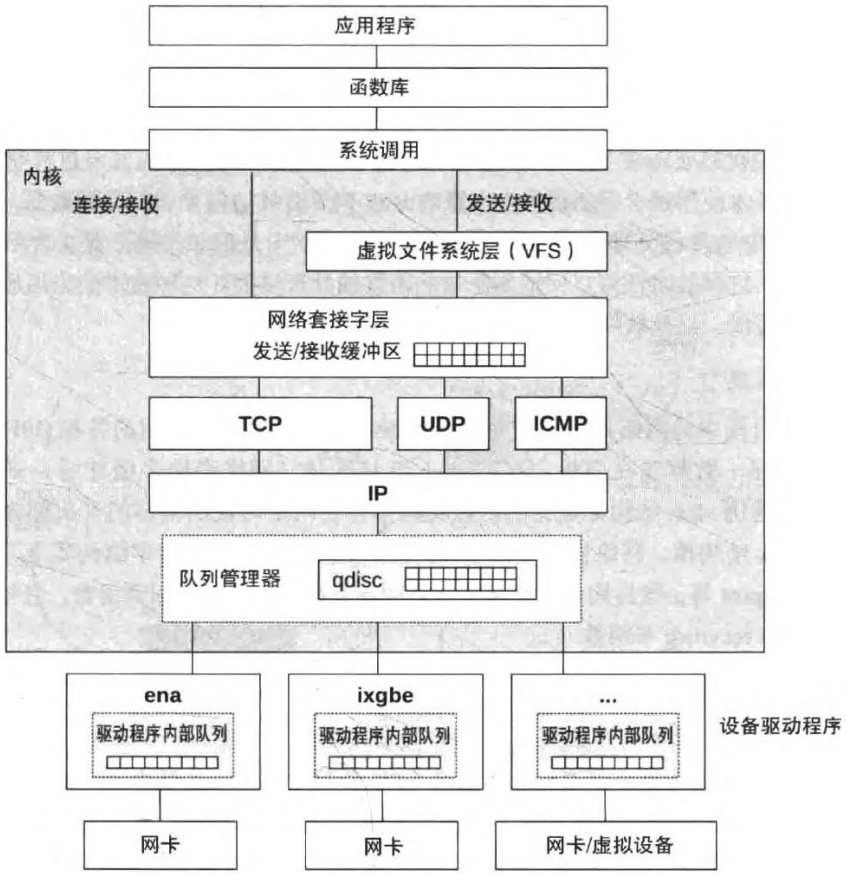



图 1. Linux 网络协议栈

# Linux BPF Internals



- 网络传输速度不断提升，以太网已经实现 100Gbps 甚至是 400Gbps 的速率，如何让抓包工具能匹配网速呢？性能是个关键问题。
- BPF/eBPF (2022) 已经是一个技术名称综合，与 LLVM 不是底层虚拟机一样，不可“望文生义”。利用 BPF 可以在内核态编程，并且有以下主要优势：
  - User-defined programs 用户可编程
  - Limited and secure kernel access 内核依旧安全
  - A new type of software 与 VM、Docker 一样已经为一种新技术

*https://www.usenix.org/conference/lisa21/presentation/gregg-bpf*

# 利用 Linux BPF 开发监控软件



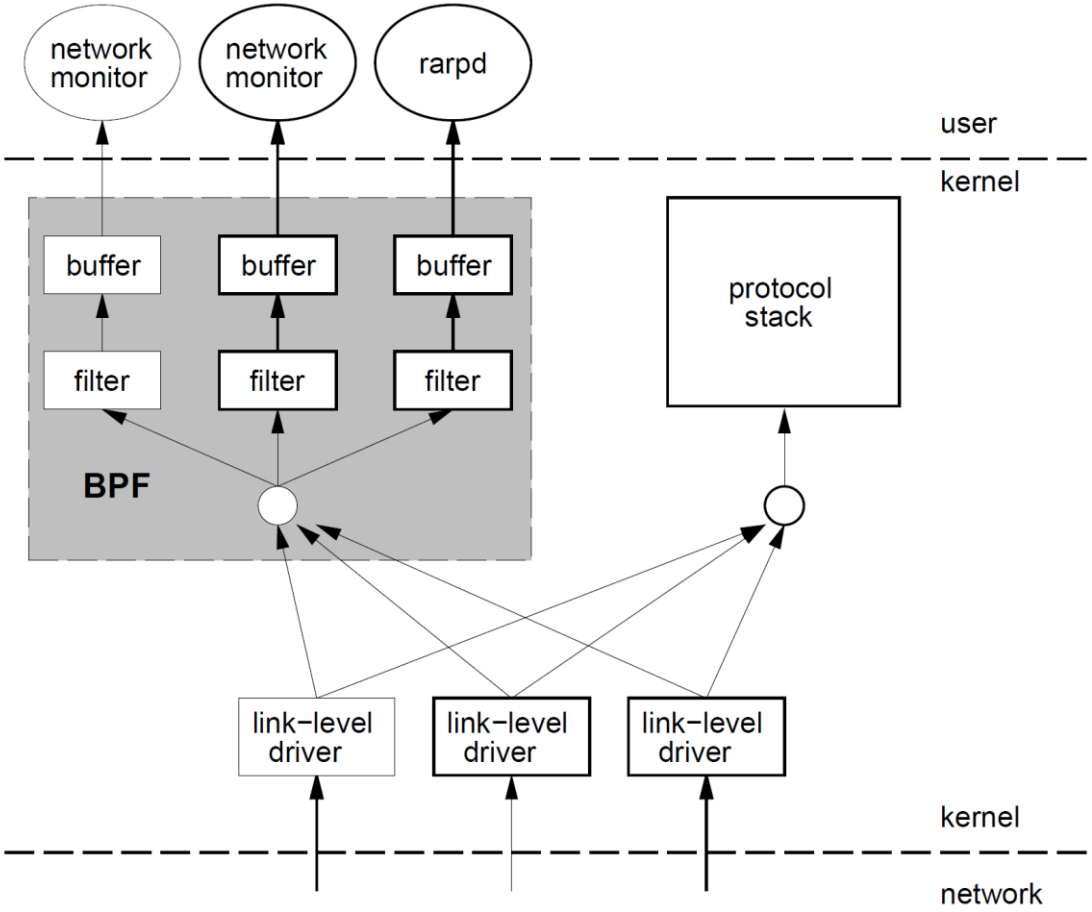



图 2. BPF 概览

# BPF 与 传统编程对比



| | Execution model | User defined | Compil-ation | Security | Failure mode | Resource access |
|---|---|---|---|---|---|---|
| **User** | task | yes | any | user based | abort | syscall, fault |
| **Kernel** | task | no | static | none | panic | direct |
| **BPF** | event | yes | JIT, CO-RE | verified, JIT | error message | restricted helpers |

图 3. 用户态、内核态与 BPF 编程对比

# BPF kernel Verifier



```
check_subprogs                  check_helper_mem_access
check_reg_arg                   check_func_arg
check_stack_write               check_map_func_compatibility
check_stack_read                check_func_proto
check_stack_access              check_func_call
check_map_access_type           check_reference_leak
check_mem_region_access         check_helper_call
check_map_access                check_alu_op
check_packet_access             check_cond_jmp_op
check_ctx_access                check_ld_imm
check_flow_keys_access          check_ld_abs
check_sock_access               check_return_code
check_pkt_ptr_alignment         check_cfg
check_generic_ptr_alignment     check_btf_func
check_ptr_alignment             check_btf_line
check_max_stack_depth           check_btf_info
check_tp_buffer_access          check_map_prealloc
check_ptr_to_btf_access         check_map_prog_compatibility
check_mem_access                check_struct_ops_btf_id
check_xadd                      check_attach_modify_return
check_stack_boundary            check_attach_btf_id
```

# Reference 1A



- https://www.youtube.com/watch?v=_5Z2AU7QTH4&ab_channel=USENIX
- libpcap官方仓库： https://github.com/the-tcpdump-group/libpcap

# 实验 1B：安全文件传输软件



- **Transfer File over Network with Security**
  - 安全地通过网络传输文件，简称 FTPS
  - 不要求写一个带身份验证的类 SMB 协议
  - 不要求利用内核 RDMA 等高级功能
- **温馨提示**
  - 建议使用 Visual Studio 2019 实现图形化部分
    - Windows 的 WinForm、WPF 图形库
    - macOS 的 Cocoa 图形库
    - Linux 的各种图形库（Wxwidgets，xxQT）
  - 由于用来判卷的电脑是 Windows 系统，因此用其他 OS 的同学，需要提供编译成功、运行成功的视频。

# 现代 TCP 报头



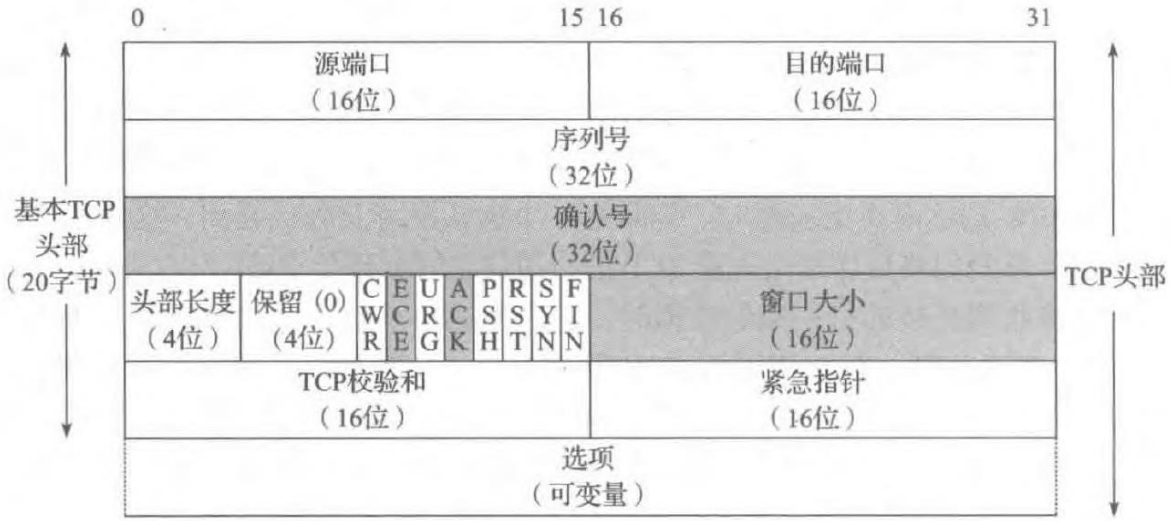



图 12-3 TCP 头部。它的标准长度是 20 字节，除非出现选项。头部长度（Header Length）字段以 32 位字为单位给出头部的大小（最小值是 5）。带阴影的字段（确认号（Acknowledgment Number）、窗口大小（Window Size）以及 ECE 位和 ACK 位）用于与该报文段的发送方关联的相反方向上的数据流

# 下知地理：TCP 可靠字节流



- TCP 协议本质是带累积正向确认的滑动窗口协议
  - TCP 虽然会给网络层提供 Segment，即报文段，但 TCP 本身是面向字节流的，并非是传输片段。
  - TCP 会自动分片
  - TCP 利用反馈、重传机制，在有损信道上实现可靠传输。
- 如何利用 TCP 传输一个固定大小的文件?
  - 建立连接
  - 发送方直接用 Socket 发送文件，比如以 4KB 为单位读取本地文件在读了 N 个 4KB 之后，终于读取到了 EOL 标志，则最后发送 $(\mathrm{FileSize} - 4 * \mathrm{N})$ KB的数据，然后发送 FIN，关闭连接
  - 接收方只需要傻傻接收，然后把数据包组合起来即可恢复出原文件。

# 如何在应用层做文章?



- 上述场景的局限性在于传输的终结由发送方控制
  - 在传输完成之前，只有发送方知道文件的具体大小。
  - 文件是否接收完是由发送方关闭连接所确定的。
- 如何克服上述弊端?
  - 发送方通过自定义应用层协议，提前告知接收方需准备多少磁盘空间（比如 N kB）来存储要接收的数据。
  - 接收方根据 N 的值，从某一个包开始，截取包内容，并往本地的文件描述符写入数据，一旦接收到了 N kB 的数据，就完成接受，并回复接收完成的确认。此后也不需要关闭连接。
  - 接收方回复已完成接收后，发送方关闭打开的文件，并对其文件描述符引用计数减一。

# Server 端 ACL 配置



- 自定义服务端业务代码
  - 业务端进程需具备严格的访问控制。服务端只允许某个目录被 Client 读取、写入。
  - ACL 主要针对以下项目进行权限控制
    - 用户 User
    - 用户组 Group
    - 默认属性 Mask
  - 另有 setfacl 之外的方法可实现服务器端的访问控制。
  - Server 端的可读写目录配置需以 shell 脚本或其他形式给出明确体现。

# 应用层协议设计



- 需要建立长连接，代码中要明确指示 keep alive
  - 以下行为是不允许的：~~Client 发起一次请求并得到响应之后，即关闭 TCP 连接。~~需要有守护进程维护连接，直至 Client 被手动关闭。
- 需要在 Report 明确写出应用层协议的设计原理
  - 图文并茂地叙述协议设计的思路
    - 时序图
    - 协议头部信息描述图（类似 TCP 头部）
  - 解释协议能准确运行的合理性

# 加分项



- 文件断点续传
- 基于公钥的身份验证
- 支持同时上传、下载多个文件
- 支持服务端根据用户身份，控制传输带宽

# 提示：协议的实现方法



- 仿照 TCP 报头的样式进行实现
  - 划分字段，不同的值代表不同的行为
  - 收到报头，按照 struct 进行解析，然后按照对应的逻辑完成对应的任务。
  - 特别注意，统一
- 用 JSON 编码/解析器完成
  - 这种方法与 struct 报头的方案在理论上是等价的，但是更加现代化。
  - 优势：JSON 是服务器端和客户端都理解的描述语言，因此只要定义了双方的通信解析方法，就可以通过 Json 格式的数据来传输控制报文，无需做字节级的包头解析。
  - 劣势：用 JSON 做控制平面的数据传输固然方便，但实现控制平面与数据平面的分离是颇具挑战的。

# 仿 TCP 等报文头的设计



| 包类型名称 | 类型解释 | 包类型定义 |
| --- | --- | --- |
| REGISTER_REQUEST | 注册请求 | 00 |
| REGISTER_RESPONSE | 注册响应 | 01 |
| LOGIN_REQUEST | 登陆请求 | 11 |
| LOGIN_RESPONSE | 登陆响应 | 12 |
| CATALOG_REQUEST | 远程文件目录请求 | 21 |
| CATALOG_RESPONSE | 远程文件目录响应 | 22 |
| FILE_REQUEST | 文件下载请求 | 31 |
| FILE_METADATA | 文件元数据通知 | 32 |
| FILE_CONTENT | 文件内容 | 33 |

# 技术分享建议



- 先描述好一个问题。
  - 描述一个能让大家感同身受的编程、设计问题，不要一上来就讲你怎么做的。
- How 比 What 重要。
  - 要有不同技术的比较。
- 一定要有 Best Practice 或方法论总结

# Thanks

- **Thank you for your listening!**
- **contact information**
  - Email: liupeng19@mails.ucas.edu.cn