# Computer Network Practicum

## Home-brew Protocol Stack

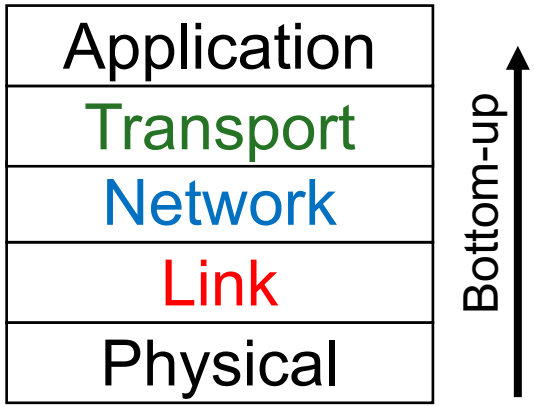
### Lab 2 Assignment

**TA: 王诚科**
**Email: chengke@pku.edu.cn**

# Overview

- 纸上得来终觉浅，绝知此事要躬行。
  - 《冬夜读书示子聿》

- Goal:
  - Build a protocol stack from bottom to top.
- This lab contains three parts!
  - Sending/receiving Ethernet frames
  - Routing algorithm
  - A simplified version of TCP
- What you can get from this lab
  - In-depth understanding of TCP/IP classic standards
  - Experience of building a computer system
  - Familiarization with Linux network tools



# Toolkits

- vnetUtils
  - Construct a controllable virtual network
- mperf
  - Test the TCP throughput and delay
- Libpcap
  - A portable C/C++ library for network traffic capture
- Wireshark / tcpdump
  - Capture and inspect network traffic on the wire
- Handy scripts by yourself
  - In C/C++, Lua, Python, Shell, …

# Workspace

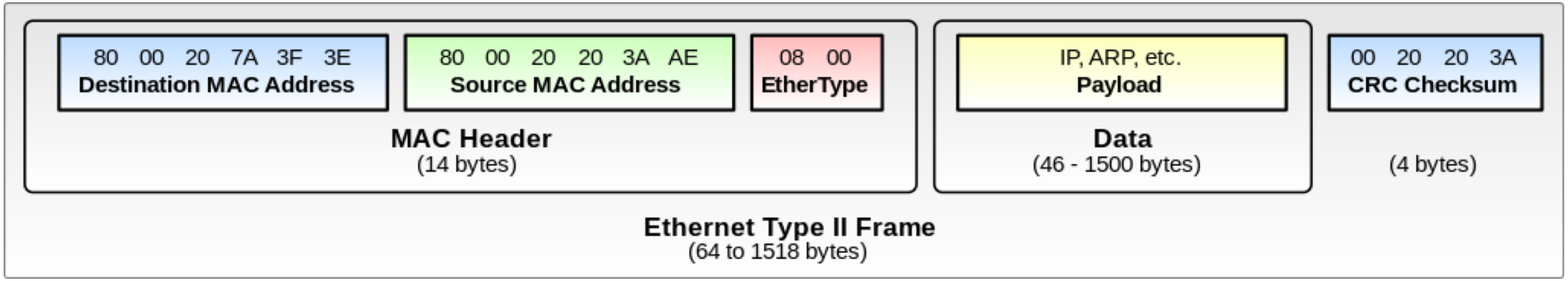
- Programming languages
  - C/C++, Rust
- WSL1 (Windows Subsystem for Linux) is not recommended
  - It doesn’t support Linux network namespace
  - WSL 2 may work
- Build automation
  - Makefile, CMake, Cargo
  - …
- Development tools
  - Valgrind: memory leak detection
  - Helgrind: data race detection
  - Google testing (gtest): unit testing and mocking framework for C++
  - Git: version control system
  - …

# Documentations

- Man page of PCAP
  - https://www.tcpdump.org/manpages/pcap.3pcap.html
- RFC 791: INTERNET PROTOCOL
  - https://datatracker.ietf.org/doc/html/rfc791
- RFC 793: TRANSMISSION CONTROL PROTOCOL
  - https://datatracker.ietf.org/doc/html/rfc793
- POSIX.1-2017: OS interface and environment standard
  - http://pubs.opengroup.org/onlinepubs/9699919799

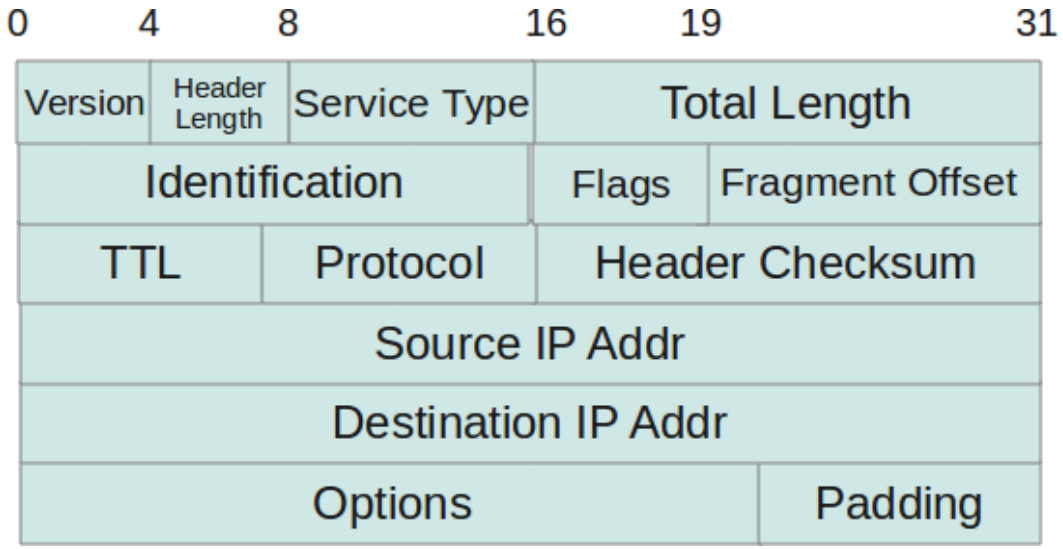
# Part A: Link Layer

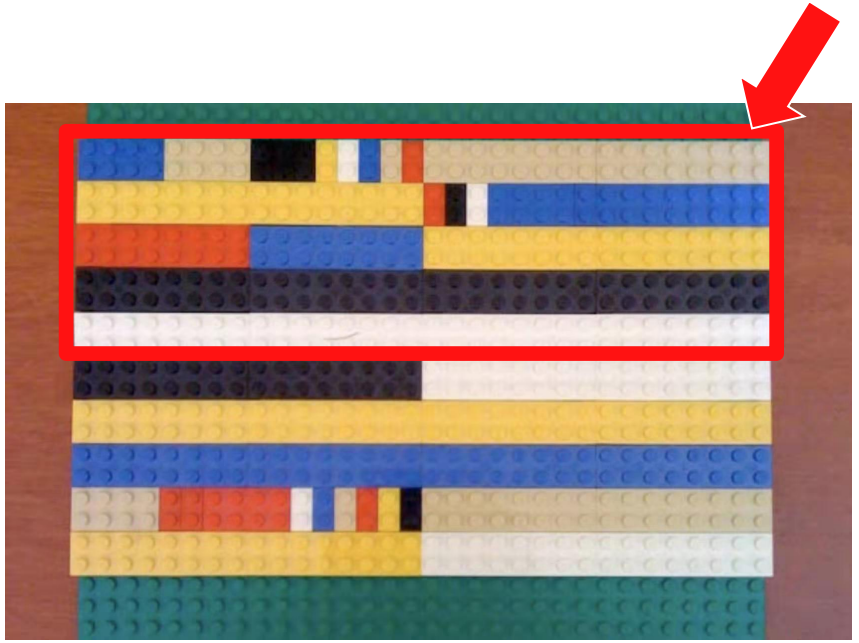
- Make use of libpcap to support sending/receiving Ethernet II frames.
- Bottom: libpcap functions, e.g. sending and receiving “framed” bitstreams.
- Top: device management, sending/receiving Ethernet II frames.



# Part B: Network Layer

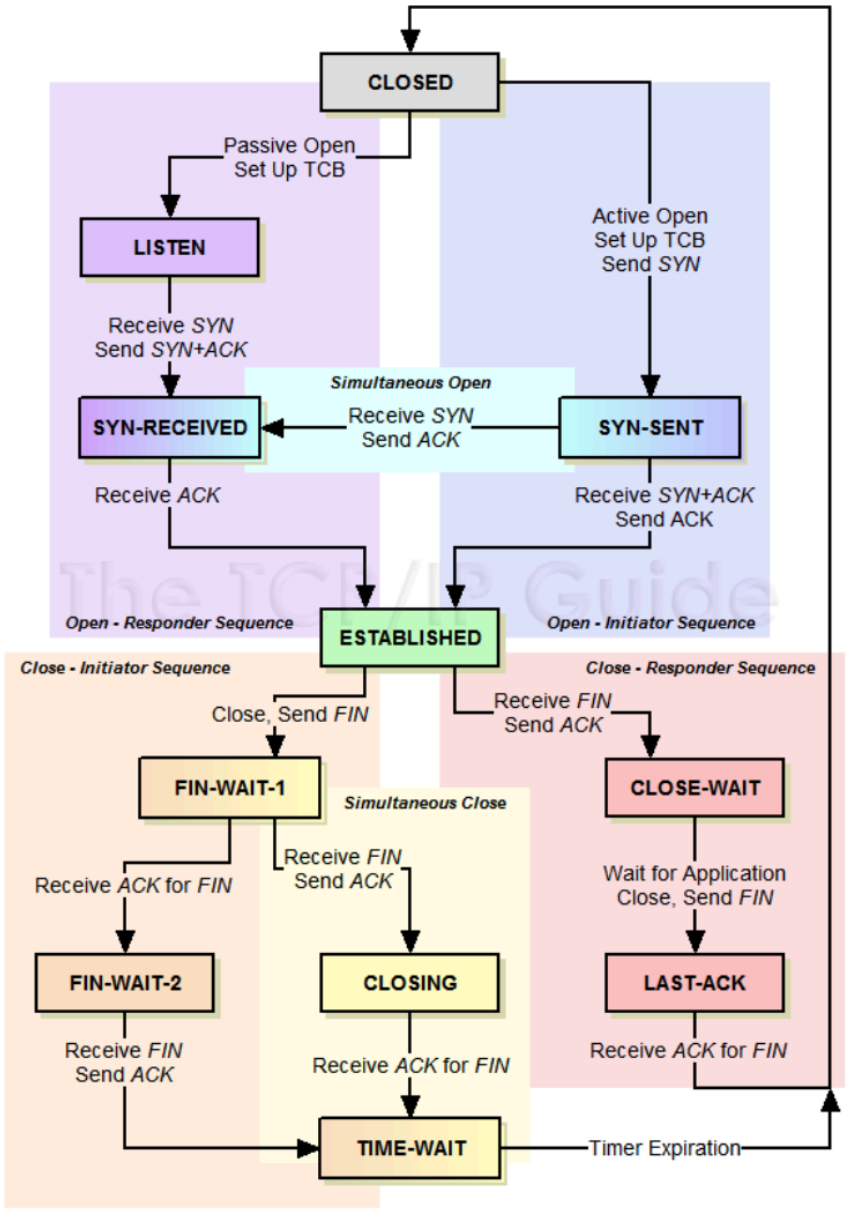
- Implement a simplified version of Internet Protocol version 4.
- Bottom: your Ethernet library.
- Inside: your routing algorithm.
- Top: sending/receiving packets over the whole Internet.



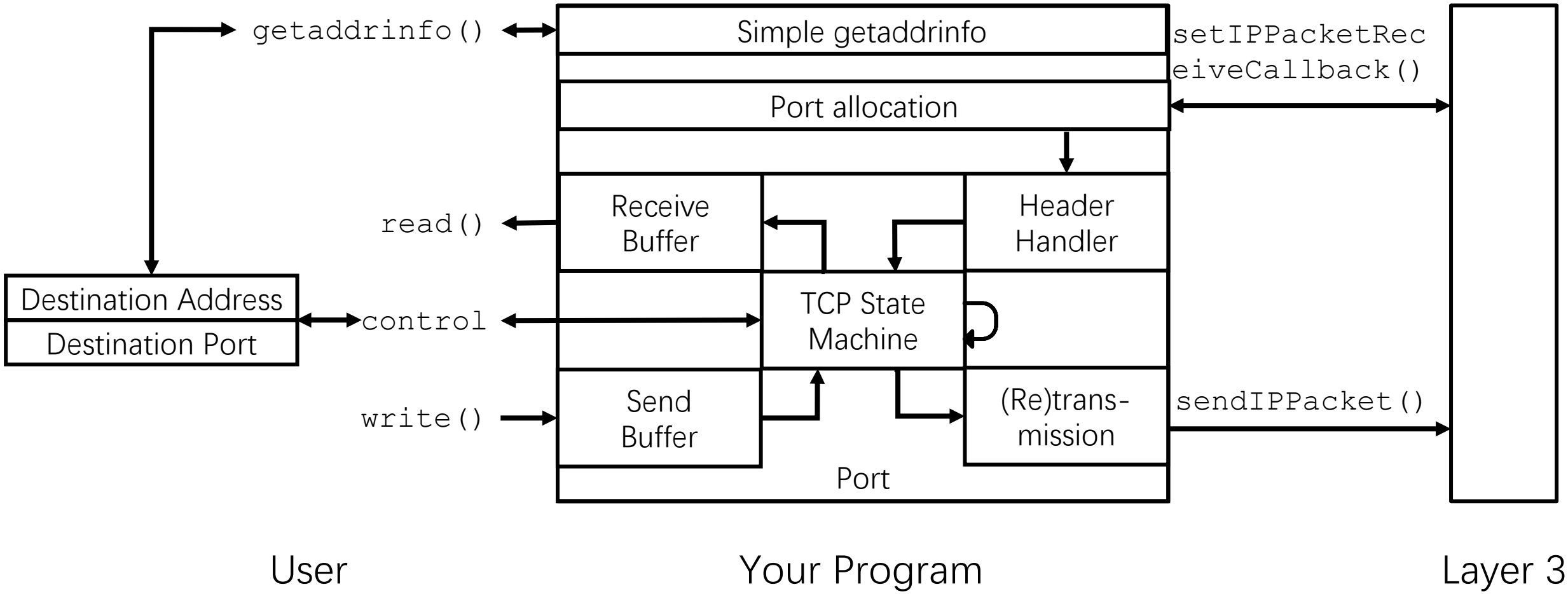



# Part C: Transport Layer

- Implement a simplified version of TCP protocol, providing a subset of socket interfaces.
- Bottom: your IP library.
- Inside: connection state machine.
- Top: reliable connection via network.



# Example


getaddrinfo()
Simple getaddrinfo
setIPPacketReceiveCallback()
Port allocation
read()
Receive Buffer
Header Handler
Destination Address
Destination Port
control
TCP State Machine
write()
Send Buffer
(Re)trans-mission
sendIPPacket()
Port
User
Your Program
Layer 3

# Grading

- Tasks:
  - Programming Task (PT) [PT5 is optional]
  - Writing Task (WT) [WT4 is optional]
  - Checkpoint (CP)
  - Challenge (CL) [Optional]

| | pt | Total pt |
|---|---|---|
| PT1, PT2 | Each task 5 pts | 10 |
| PT3, PT4 | Each task 8 pts | 16 |
| CP1-CP8 | Each checkpoint 5 pts | 40 |
| CP9, CP10 | Each checkpoint 7 pts | 14 |
| WT1 | | 4 |
| WT2, WT3 | Each task 8 pts | 16 |
| PT5, WT4 | Extra points for bonus! | +20 |
| CL1-CL11 | Each task 8 pts. No more than 30 pts. | +30 |

# Submission

- Due:
  - Part A: Oct. 20, 2021 (23:59:59 UTC+8)
  - Part B: Nov. 3, 2021 (23:59:59 UTC+8)
  - Part C: Nov. 29, 2021 (23:59:59 UTC+8)
- Submit the zipped file named “lab2-Name-StudentID” to chengke@pku.edu.cn
- Grading: 20% of the course
- Start early and good luck
- Q&A on Slack