

冲刺网络开发计划

目标岗位：云原生网络研发工程师 / Service Mesh / eBPF 方向

项目方向：C++ P2P/NAT 网络库 + Kubernetes eBPF 网络观测系统

个人背景：

- 1年C语言协议栈开发经验 (ARP/ICMP/TCP/UDP/DHCP/IPsec)
- OPPO云连接项目：多重NAT穿透、STUN、虚拟网卡方案
- Linux网络编程有实战 (epoll、socket)
- 当前Rust网络开发在职
- 目标：云原生网络 / Service Mesh / eBPF 方向

时间安排：

- 工作日：19:00-22:00 (3h)
- 休息日：全天可用 (8h)
- 每周总计：约31h

| 时段 (工作日) | 内容 | 时长 |
|-------------|------------|-------|
| 19:00-20:00 | C++/eBPF学习 | 1h |
| 20:00-21:00 | 项目编码 | 1h |
| 21:00-22:00 | LeetCode刷题 | 1h |
| 22:00-22:20 | 技术书籍阅读 | 20min |

| 时段 (休息日) | 内容 | 时长 |
|-------------|------------|-------|
| 09:00-12:00 | 项目开发 | 3h |
| 14:00-18:00 | 项目开发 | 4h |
| 20:00-21:30 | LeetCode刷题 | 1.5h |
| 21:30-22:00 | 技术书籍阅读 | 30min |

阶段一：C++ P2P/NAT 项目 (第1-7周)

阶段目标：使用原生 epoll + socket 实现 P2P/NAT 穿透 demo，展示现代 C++ 能力和网络编程底层功力

第1周：项目骨架 + epoll 事件循环

| 任务 | 具体内容 | 产出 |
|-------------|---|----------------------|
| 项目初始化 | CMake 构建、目录结构、编码规范 | 项目骨架搭建完成 |
| epoll 事件循环 | 封装 EventLoop 类，支持添加/删除 fd、事件回调 | event_loop.cpp/h |
| 基础 TCP 封装 | 非阻塞 socket、TcpListener、TcpConnection (RAII) | tcp_connection.cpp/h |
| Echo Server | 用事件循环实现简单的 TCP Echo Server | 验证事件循环正确性 |
| LeetCode | 6题 (数组/链表) | - |

C++ 重点：智能指针 (shared_ptr/unique_ptr) 、 RAII、 std::function 回调

第2周：UDP 封装 + 信令服务器

| 任务 | 具体内容 | 产出 |
|----------|---------------------------|----------------------|
| UDP 封装 | UdpSocket 类，支持非阻塞收发、绑定端口 | udp_socket.cpp/h |
| 信令协议设计 | 定义 JSON 消息格式 (注册、候选交换、心跳) | protocol.h |
| 信令服务器 | 接受客户端注册、存储地址信息、转发候选地址 | signaling_server.cpp |
| 单元测试 | 信令消息解析测试 | tests/ |
| LeetCode | 6题 (哈希表/字符串) | - |

C++ 重点：lambda 表达式、 std::map/unordered_map、 JSON 解析 (nlohmann/json)

第3周：NAT 类型探测

| 任务 | 具体内容 | 产出 |
|-----------|--|------------------------|
| STUN 协议基础 | 理解 STUN Binding Request/Response | - |
| NAT 探测逻辑 | 实现简化版 NAT 类型探测 (Full Cone / Symmetric 等) | nat_detector.cpp/h |
| 信令客户端 | 连接信令服务器、注册、接收对端候选 | signaling_client.cpp/h |
| 集成测试 | 两个客户端通过信令交换地址 | - |
| LeetCode | 6题 (双指针/滑动窗口) | - |

C++ 重点：std::thread、 std::mutex、条件变量

第4周：UDP 打洞 + P2P 连接

| 任务 | 具体内容 | 产出 |
|----------|---|------------------------|
| 打洞逻辑 | 双方同时向对方发送 UDP 包，尝试建立直连 | hole_puncher.cpp/h |
| 连接状态机 | Init → Signaling → Probing → Connected / Fallback | connection_state.cpp/h |
| 超时重试 | 打洞超时后重试，多次失败后回退 | - |
| 端到端测试 | 两个客户端在同一 NAT 下成功打洞 | - |
| LeetCode | 6题 (二叉树/BFS) | - |

C++ 重点：std::atomic、内存模型基础、状态机设计

第5周：中继回退 + 心跳保活

| 任务 | 具体内容 | 产出 |
|----------|---------------------|------------------|
| 中继服务 | 信令服务器兼任中继，转发打洞失败的流量 | relay_server.cpp |
| 回退逻辑 | 打洞失败后自动切换到中继模式 | - |
| 心跳保活 | 定时发送心跳包，超时断开 | heartbeat.cpp/h |
| 断线重连 | 检测连接断开，触发重新打洞/重连 | - |
| LeetCode | 6题 (动态规划) | - |

C++ 重点：定时器实现 (timerfd 或自定义)、std::chrono

第6周：功能完善 + 错误处理

| 任务 | 具体内容 | 产出 |
|----------|--------------------------|--------------|
| 日志系统 | 集成 spdlog，关键路径打印日志 | logger.h |
| 配置文件 | 支持 JSON 配置 (服务器地址、超时参数等) | config.cpp/h |
| 错误处理 | 完善 socket 错误、超时、异常情况处理 | - |
| 简单应用 | 基于 P2P 通道的简易聊天/文件传输 demo | demo/ |
| LeetCode | 6题 (回溯/贪心) | - |

C++ 重点：异常处理、RAII 资源管理、C++17 optional/variant

第7周：打磨 + 文档 + 架构图

| 任务 | 具体内容 | 产出 |
|----------|---------------------|-----------------------|
| 代码审查 | 检查代码风格、命名规范、注释完善 | - |
| README | 项目介绍、架构说明、使用方法、构建指南 | README.md |
| 架构图 | 绘制系统架构图、连接流程图 | docs/architecture.png |
| 面试准备 | 整理项目亮点、可能被问到的技术点 | - |
| LeetCode | 6题（图论） | - |

第7周末检查点：

- 项目能在本地两台机器/进程间成功建立 P2P 连接
- 能讲清楚：epoll 事件循环、NAT 类型探测、UDP 打洞原理、状态机设计
- 代码展示现代 C++ 特性：智能指针、RAII、lambda、多线程
- LeetCode 完成 42 题

阶段二：eBPF + Kubernetes 云原生网络观测（第8-19周）

阶段目标：在 Kubernetes 环境下实现基于 eBPF 的网络观测系统，完成云原生网络方向的项目积累

第8-9周：eBPF 基础 + 开发环境

| 周数 | 学习内容 | 实践任务 | LeetCode | 产出 |
|----|---|--|----------|-------------|
| 8 | eBPF 原理：BPF 字节码、verifier、map 类型、程序类型 (kprobe/tracepoint/tc/xdp) | 搭建 eBPF 开发环境 (clang/llvm/libbpf)、跑通 hello world | 6题 | eBPF 环境就绪 |
| 9 | libbpf 使用：CO-RE、BPF skeleton、map 操作 | 实现简单的 syscall 追踪程序 (如追踪 open/read) | 6题 | 第一个 eBPF 程序 |

第10-11周：网络相关 eBPF hook 点

| 周数 | 学习内容 | 实践任务 | LeetCode | 产出 |
|----|--|--|----------|-------------------|
| 10 | 网络 eBPF hook: tc (Traffic Control) 、 XDP、 socket 相关 tracepoint | 实现 TCP 连接追踪: hook tcp_v4_connect/tcp_close | 6题 | 能打印 TCP 连接建立/关闭事件 |
| 11 | eBPF map 进阶: per-cpu map、 LRU map、 ring buffer | 用 map 统计每个 IP 的连接数、发送/接收字节数 | 6题 | 网络统计 eBPF 程序 |

第12-13周：用户态 Agent 开发 (C++)

| 周数 | 开发内容 | 具体任务 | LeetCode | 产出 |
|----|-----------------|---|----------|--------------------|
| 12 | Agent 骨架 | 加载 eBPF 程序、读取 map 数据、定时采集 | 6题 | agent 基础框架 |
| 13 | Prometheus 指标暴露 | 实现 HTTP server、暴露 /metrics 端点 (Prometheus 格式) | 6题 | 指标可被 Prometheus 抓取 |

第14-15周：Kubernetes 部署

| 周数 | 学习/开发内容 | 具体任务 | LeetCode | 产出 |
|----|--|---|----------|----------------------|
| 14 | K8s 基础: Pod/Service/DaemonSet、网络模型 (CNI) | 搭建单节点 k8s (microk8s/k3s) 、部署简单应用 | 6题 | k8s 环境就绪 |
| 15 | 容器化 Agent | Dockerfile、DaemonSet yaml、privileged 权限配置 | 6题 | Agent 以 DaemonSet 运行 |

第16-17周：可观测性完善

| 周数 | 开发内容 | 具体任务 | LeetCode | 产出 |
|----|------------|--------------------------------|----------|---------|
| 16 | 指标丰富化 | 添加更多指标: TCP RTT、重传次数、连接延迟分布 | 6题 | 指标覆盖完整 |
| 17 | Grafana 大盘 | 部署 Prometheus + Grafana、创建监控大盘 | 6题 | 可视化大盘完成 |

第18-19周：项目打磨 + 文档

| 周数 | 任务 | 具体内容 | LeetCode | 产出 |
|----|----------|------------------------|----------|---------|
| 18 | 代码完善 | 错误处理、日志、配置化、代码审查 | 6题 | 代码质量提升 |
| 19 | 文档 + 架构图 | README、架构图、部署指南、面试要点整理 | 6题 | 项目完整可展示 |

第19周末检查点：

- eBPF 程序能采集 Pod 间 TCP 连接指标
- Agent 以 DaemonSet 运行，指标可被 Prometheus 抓取
- Grafana 大盘能展示网络观测数据
- 能讲清楚：eBPF 程序类型、hook 点选择、map 使用、k8s 网络模型
- LeetCode 完成 114 题

阶段三：面试冲刺（第20-26周）

阶段目标：八股文熟练、两个项目能深挖、算法手写无压力

| 周数 | 八股文重点 | 项目复盘 | LeetCode | 产出 |
|----|---|---|----------|-----------|
| 20 | C++ 高频：虚函数表、智能指针实现、移动语义、内存模型、RAII | 整理 P2P/NAT 项目亮点：epoll 封装、状态机设计、NAT 穿透方案 | 8题+错题 | C++ 八股文笔记 |
| 21 | 网络高频：TCP 状态机、拥塞控制、NAT 原理、epoll 原理（LT/ET）、UDP 打洞 | 整理 P2P/NAT 项目深挖问题：为什么用 epoll、如何处理半包、打洞失败率 | 8题+错题 | 网络八股文笔记 |
| 22 | 云原生高频：K8s 网络模型、CNI、Service Mesh 原理、eBPF 原理 | 整理 eBPF 项目亮点：为什么选 eBPF、hook 点选择、性能影响 | 8题+错题 | 云原生八股文笔记 |
| 23 | 系统高频：进程/线程、内存管理、Linux 网络栈、socket 编程 | 准备系统设计题：设计 NAT 网关、设计网络监控系统 | 8题+错题 | 系统八股文笔记 |
| 24 | 算法高频专项 | 两个项目代码精读，确保每行代码都能解释 | 10题+错题 | - |
| 25 | 模拟面试 | 找朋友/AI 模拟面试，查漏补缺 | 10题+错题 | - |
| 26 | 最后冲刺 | 复习所有笔记、错题重做、心态调整 | 错题重做 | 面试准备完成 |

第26周末最终目标：

- LeetCode 180+ 题（120+ Medium, 30+ Hard）
- 项目一：C++ P2P/NAT 网络库（GitHub + 架构图 + 技术博客）

- 项目二：Kubernetes eBPF 网络观测系统（GitHub + Grafana 大盘截图）
 - 八股文：C++/网络/云原生/系统核心问题能流畅作答
 - 能手写：智能指针、LRU、epoll echo server、简单状态机
-

算法题分类规划

总目标：180+ 题，聚焦高频 + 网络/系统相关

| 类型 | 题数 | 重点题目 |
|---------|----|---------------------------------------|
| 链表 | 20 | 反转链表、合并K个链表、LRU Cache、复制带随机指针的链表、环形链表 |
| 哈希表 | 20 | 两数之和、三数之和、LRU/LFU、设计哈希表、字母异位词分组 |
| 二叉树 | 20 | 层序遍历、最近公共祖先、二叉树序列化、BST相关、路径总和 |
| 动态规划 | 25 | 最长子序列系列、背包问题、编辑距离、股票问题、零钱兑换 |
| 滑动窗口 | 15 | 最小覆盖子串、无重复最长子串、滑动窗口最大值、字符串排列 |
| 双指针 | 15 | 接雨水、盛水最多的容器、三数之和、移除元素 |
| BFS/DFS | 20 | 岛屿数量、课程表、单词搜索、图的遍历、全排列、子集 |
| 二分查找 | 15 | 搜索旋转数组、寻找峰值、分割数组最大值、搜索二维矩阵 |
| 贪心/栈/堆 | 20 | 跳跃游戏、区间调度、合并区间、有效括号、前K个高频元素 |
| 系统设计题 | 10 | LRU/LFU Cache、设计哈希Map、最小堆实现、Trie树 |

学习资源

C++ 现代特性

- Effective Modern C++ (Scott Meyers)
- C++ Concurrency in Action
- Linux高性能服务器编程 (游双) - epoll/socket 部分

eBPF / 云原生网络

- Learning eBPF (Liz Rice)
- BPF Performance Tools (Brendan Gregg)
- libbpf-bootstrap 官方示例
- Kubernetes 官方文档 (网络部分)
- Cilium 文档 (了解 eBPF 在 k8s 中的应用)

网络 / 系统

- TCP/IP 详解卷一
- UNIX 网络编程 (W. Richard Stevens)
- 性能之颠 (Brendan Gregg)

每日阅读计划 (每天20分钟)

| 阶段 | 周数 | 书籍 | 阅读目标 |
|----|--------|---|----------------------------------|
| 一 | 1-7周 | Linux高性能服务器编程 + Effective Modern C++ | 重点: epoll、非阻塞 I/O、智能指针、移动语义 |
| 二 | 8-19周 | Learning eBPF + BPF Performance Tools | 重点: eBPF 程序类型、map、网络 hook 点、性能分析 |
| 三 | 20-26周 | TCP/IP 详解卷一 + C++ Concurrency in Action | 重点: TCP 状态机、拥塞控制、内存模型、原子操作 |

阅读方法:

- 工作日睡前20分钟，休息日可延长至30分钟
- 边读边做笔记，记录面试可能问到的点
- 遇到代码示例动手实践

项目目录结构参考

项目一：C++ P2P/NAT 网络库

```
p2p_nat_cpp/
├── CMakeLists.txt
└── src/
    └── core/
        └── event_loop.cpp/h      # epoll 事件循环
```

```

    |   └── tcp_connection.cpp/h # TCP 连接封装
    |   └── udp_socket.cpp/h   # UDP socket 封装
    |
    └── nat/
        ├── nat_detector.cpp/h # NAT 类型探测
        └── hole_puncher.cpp/h # UDP 打洞逻辑
    └── signaling/
        ├── signaling_server.cpp # 信令服务器
        └── signaling_client.cpp/h # 信令客户端
    └── connection/
        ├── connection_state.cpp/h # 连接状态机
        ├── heartbeat.cpp/h       # 心跳保活
        └── relay.cpp/h          # 中继回退
    └── util/
        ├── logger.h             # 日志 (spdlog)
        └── config.cpp/h         # 配置解析
    └── main.cpp
└── server/
    └── main.cpp              # 信令服务器入口
└── demo/
    └── chat_demo.cpp         # 简易聊天示例
└── tests/
└── docs/
    └── architecture.md      # 架构说明

```

项目二：Kubernetes eBPF 网络观测系统

```

ebpf_net_observer/
├── CMakeLists.txt
└── src/
    ├── bpf/
    │   ├── tcp_connect.bpf.c      # eBPF 程序: TCP 连接追踪
    │   ├── tcp_stats.bpf.c       # eBPF 程序: 流量统计
    │   └── common.h               # 共享数据结构
    ├── agent/
    │   ├── bpf_loader.cpp/h      # 加载 eBPF 程序
    │   ├── map_reader.cpp/h      # 读取 map 数据
    │   ├── metrics_server.cpp/h  # Prometheus HTTP server
    │   └── main.cpp
    └── util/
        ├── logger.h
        └── config.h
└── deploy/
    ├── Dockerfile
    ├── daemonset.yaml           # K8s DaemonSet 部署
    └── prometheus.yaml          # Prometheus 配置
└── grafana/
    └── dashboard.json           # Grafana 大盘配置
└── docs/
    └── architecture.md
```

每周检查清单

- ✓ 本周学习目标是否完成
 - ✓ 项目代码是否提交
 - ✓ LeetCode 是否完成 6 题
 - ✓ 是否有不理解的点需要下周补齐
 - ✓ 是否更新了八股文笔记
-

面试重点准备

C++ 高频

- 智能指针实现原理（引用计数、控制块）
- 移动语义（`std::move`、右值引用、完美转发）
- RAII 资源管理
- 内存模型（`memory_order`、原子操作）
- 虚函数表、多态实现

网络高频

- TCP 状态机、三次握手、四次挥手
- TCP 拥塞控制（慢启动、拥塞避免、快速重传、快速恢复）
- epoll 原理（LT/ET、红黑树、回调机制）
- NAT 类型、UDP 打洞原理
- socket 编程（非阻塞、`select/poll/epoll` 区别）

云原生 / eBPF 高频

- eBPF 程序类型（`kprobe/tracepoint/tc/xdp`）
- eBPF map 类型与使用场景
- eBPF verifier 限制
- Kubernetes 网络模型（Pod 网络、Service、CNI）
- Service Mesh 基本原理（sidecar、流量劫持）

系统高频

- 进程 vs 线程、协程
- 虚拟内存、页表、缺页中断
- Linux 网络栈流程（收包/发包路径）
- 文件描述符、IO 多路复用