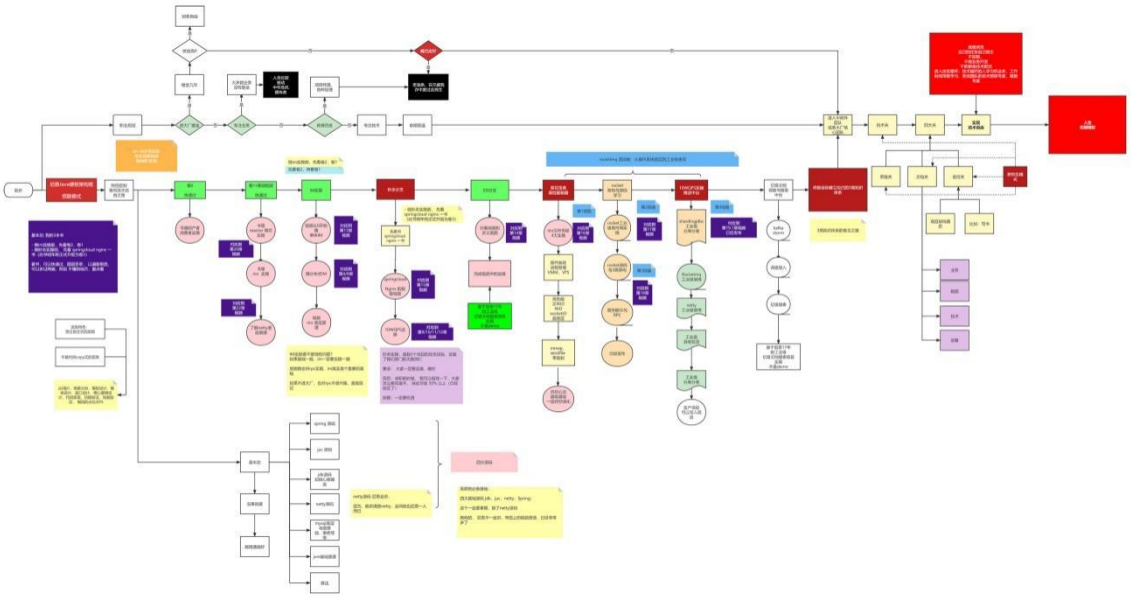
牛逼的职业发展之路

40 岁老架构尼恩用一张图揭秘：Java 工程师的高端职业发展路径，走向食物链顶端的之路

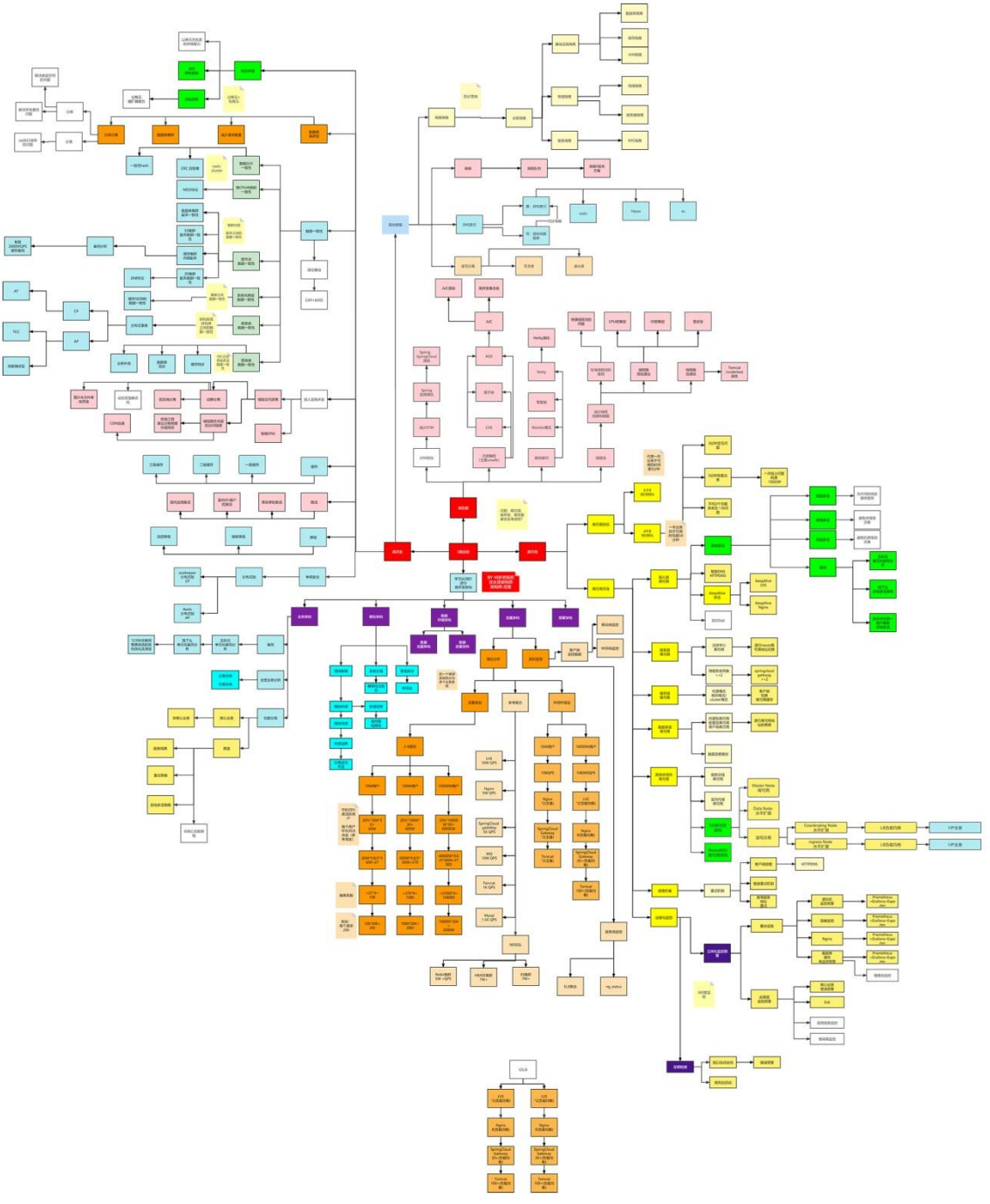
链接：https://www.processon.com/view/link/618a2b62e0b34d73f7eb3cd7



史上最全：价值10W的架构师知识图谱

此图梳理于尼恩的多个 3 高生产项目：多个亿级人民币的大型 SAAS 平台和智慧城市项目

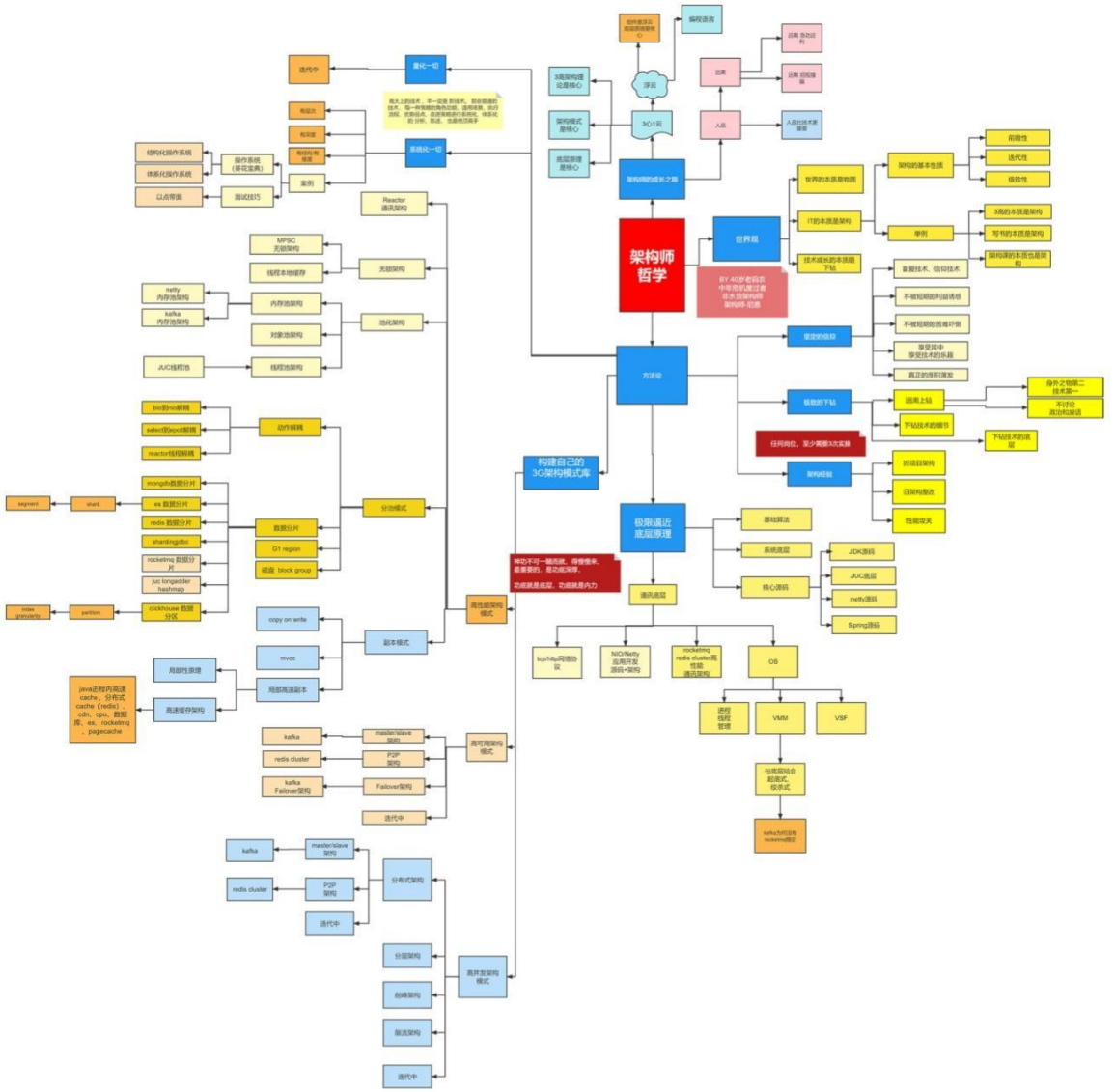
链接：https://www.processon.com/view/link/60fb94216376897l9d246739



牛逼的架构师哲学

40 岁老架构师尼恩对自己的 20 年的开发、架构经验总结

链接：https://www.processon.com/view/link/616f801963768961e9d9aec8



牛逼的3高架构知识宇宙

尼恩 3 高架构知识宇宙，帮助大家穿透 3 高架构，走向技术自由，远离中年危机

链接：https://www.processon.com/view/link/635097d2e0b34d40be778ab4

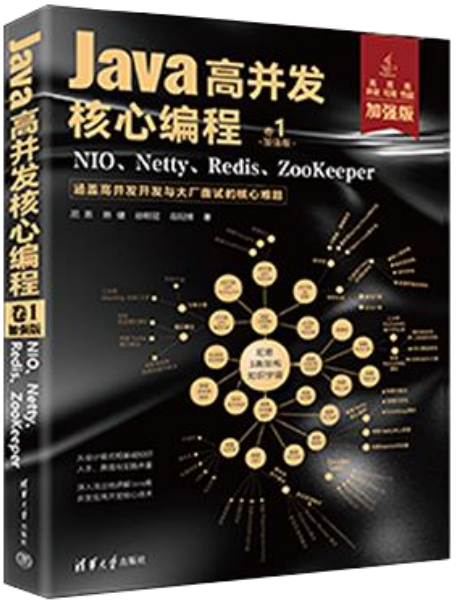
尼恩Java高并发三部曲（卷1加强版）

老版本：《Java 高并发核心编程 卷 1：NIO、Netty、Redis、ZooKeeper》（已经过时，不建议购买）

新版本：《Java 高并发核心编程 卷 1 **加强版**：NIO、Netty、Redis、ZooKeeper》

- 由浅入深地剖析了高并发 IO 的底层原理。
- 图文并茂的介绍了 TCP、HTTP、WebSocket 协议的核心原理。
- 细致深入地揭秘了 Reactor 高性能模式。
- 全面介绍了 Netty 框架，并完成单体 IM、分布式 IM 的实战设计。
- 详尽地介绍了 ZooKeeper、Redis 的使用，以帮助提升高并发、可扩展能力

详情：https://www.cnblogs.com/crazymakercircle/p/16868827.html



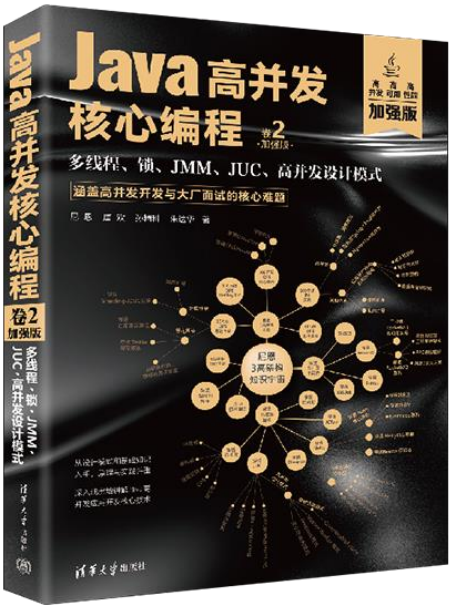
尼恩Java高并发三部曲（卷2加强版）

老版本：《Java 高并发核心编程 卷 2：多线程、锁、JMM、JUC、高并发设计模式》（已经过时，不建议购买）

新版本：《Java 高并发核心编程 卷 2 **加强版**：多线程、锁、JMM、JUC、高并发设计模式》

- 由浅入深地剖析了 Java 多线程、线程池的底层原理。
- 总结了 IO 密集型、CPU 密集型线程池的线程数预估算法。
- 图文并茂的介绍了 Java 内置锁、JUC 显式锁的核心原理。
- 细致深入地揭秘了 JMM 内存模型。
- 全面介绍了 JUC 框架的设计模式与核心原理，并完成其高核心组件的实战介绍。
- 详尽地介绍了高并发设计模式的使用，以帮助提升高并发、可扩展能力

详情参阅：https://www.cnblogs.com/crazymakercircle/p/16868827.html



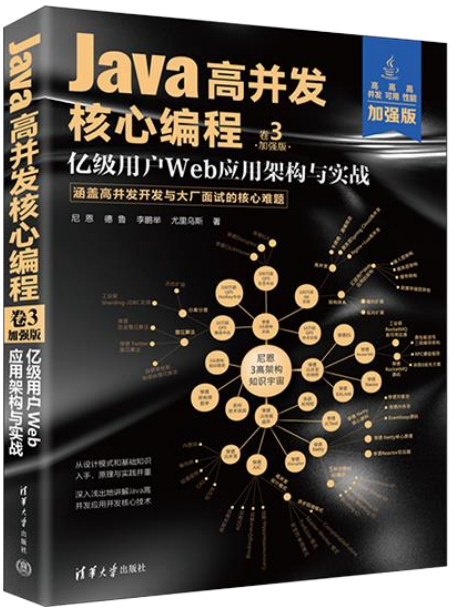
尼恩Java高并发三部曲（卷3加强版）

老版本：《SpringCloud Nginx 高并发核心编程》（已经过时，不建议购买）

新版本：《Java 高并发核心编程 卷 3 **加强版**：亿级用户 Web 应用架构与实战》

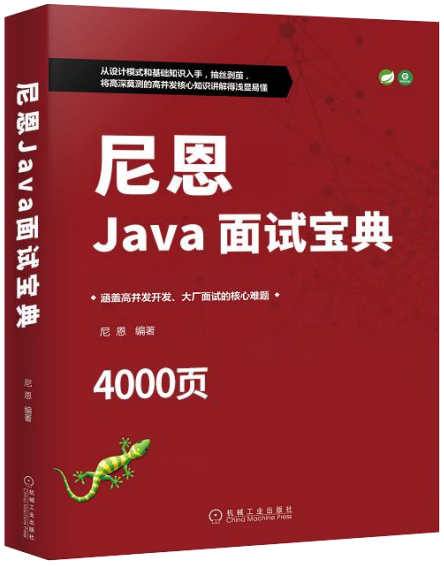
- 在当今的面试场景中，3 高知识是大家面试必备的核心知识，本书基于亿级用户 3 高 Web 应用的架构分析理论，为大家对 3 高架构系统做一个系统化和清晰化的介绍。
- 从 Java 静态代理、动态代理模式入手，抽丝剥茧地解读了 Spring Cloud 全家桶中 RPC 核心原理和执行过程，这是高级 Java 工程师面试必备的基础知识。
- 从Reactor 反应器模式入手，抽丝剥茧地解读了Nginx核心思想和各配置项的底层知识和原理，这是高级 Java 工程师、架构师面试必备的基础知识。
- 从观察者模式入手，抽丝剥茧地解读了 RxJava、Hystrix 的核心思想和使用方法，这也是高级 Java 工程师、架构师面试必备的基础知识。

详情：https://www.cnblogs.com/crazymakercircle/p/16868827.html



尼恩Java面试宝典

40 个专题（卷王专供+ 史上最全 + 2023 面试必备）
详情：https://www.cnblogs.com/crazymakercircle/p/13917138.html



专题01：JVM面试题（卷王专供 + 史上最全 + 2023面试必备）-V20-from-尼恩Java面试宝典.pdf
专题02：Java算法面试题（卷王专供 + 史上最全 + 2023面试必备）-V2 -from-Java面试红宝书-release.pdf
专题03：Java基础面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书 -release.pdf
专题04：架构设计面试题（卷王专供+ 史上最全 + 2023面试必备）-V24-from-Java面试红宝书-release.pdf
专题05：Spring面试题_专题06：SpringMVC_专题07：Tomcat面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-尼恩面试宝典-release.pdf
专题08：SpringBoot面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题09：网络协议面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题10：TCPIP协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题11：JUC并发包与容器类（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题12：设计模式面试题 （卷王专供+ 史上最全 + 2023面试必备）-V14-from-Java面试红宝书.pdf
专题13：死锁面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题14：Redis 面试题 （卷王专供+ 史上最全 + 2023面试必备）-V22-from-Java面试红宝书-release.pdf
专题15：分布式锁 面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题16：Zookeeper 面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题17：分布式事务面试题（卷王专供 + 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题18：一致性协议 （卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题19：Zab协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题20：Paxos 协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题21：raft 协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题22：Linux面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题23：Mysql 面试题（卷王专供+ 史上最全 + 2023面试必备）-V29-from-尼恩Java面试宝典-release.pdf
专题24：SpringCloud 面试题（卷王专供+ 史上最全 + 2023面试必备）-V12-from-Java面试红宝书-release.pdf
专题25：Netty 面试题 （卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题26：消息队列面试题：RabbitMQ、Kafka、RocketMQ（卷王专供+ 史上最全 + 2023面试必备）-V10-from-Java面试红宝书-release.pdf
专题27：内存泄漏 内存溢出（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题28：JVM 内存溢出 实战（卷王专供+ 史上最全 + 2023面试必备）-V17-from-Java面试红宝书-release.pdf
专题29：多线程面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题30：HR面试题：过五关斩六将后，小心阴沟翻船！（史上最全、避坑宝典）-V2-from-Java面试红宝书-release.pdf
专题31：Hash连环炮面试题（卷王专供+ 史上最全 + 2023面试必备）-V16-from-Java面试红宝书-release.pdf
专题32：大厂面试的基本流程和面试准备（阿里、腾讯、网易、京东、头条.....）-V2-from-Java面试红宝书-release.pdf
专题33：BST、AVL、RBT红黑树、三大核心数据结构（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题34：Elasticsearch面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-Java面试红宝书-release.pdf
专题35：Mybatis面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-尼恩Java面试宝典-release.pdf
专题36：Dubbo面试题（卷王专供+ 史上最全 + 2023面试必备）-V21-from-尼恩Java面试宝典-release.pdf
专题37：Docker面试题（卷王专供+ 史上最全 + 2023面试必备）-V26-from-尼恩Java面试宝典-release.pdf
专题38：K8S面试题（卷王专供+ 史上最全 + 2023面试必备）-V26-from-尼恩Java面试宝典-release.pdf
专题39：Nginx面试题（卷王专供+ 史上最全 + 2023面试必备）-V27-from-尼恩Java面试宝典-release.pdf
专题40：操作系统面试题（卷王专供+ 史上最全 + 2023面试必备）-V28-from-尼恩Java面试宝典-release.pdf

专题10：TCPIP协议（史上最全、定期更新）

本文版本说明：V57

> 此文的格式，由markdown 通过程序转成而来，由于很多表格，没有来的及调整，出现一个格式问题，尼恩在此给大家道歉啦。
>
> 由于社群很多小伙伴，在面试，不断的交流最新的面试难题，所以，《尼恩Java面试宝典》， 后面会不断升级，迭代。
>
> 本专题，作为 《尼恩Java面试宝典》专题之一， 《尼恩Java面试宝典》一共**40个面试专题，后续还会增加**

《Java面试红宝书》升级的规划为：

后续基本上，**每一个月，都会发布一次**，最新版本，可以扫描扫架构师尼恩微信， 发送 “领取电子书” 获取。

尼恩的微信二维码在哪里呢？ 请参见文末

面试问题交流说明：

如果遇到面试难题，或者职业发展问题， 或者中年危机问题，都可以来 疯狂创客圈社群交流，

加入交流群，加尼恩微信即可，

TCP/IP协议的分层模型

在展开介绍TCP/IP协议之前，首先介绍一下七层ISO模型。国际标准化组织ISO为了使网络应用更为普及，推出了OSI参考模型，即开放式系统互联（Open System Interconnect）模型，

一般都叫OSI参考模型。

OSI参考模型是ISO组织在1985年发布的网络互连模型，其含义就是为所有公司使用一个统一的规范来控制网络，这样所有公司遵循相同的通信规范，网络就能互联互通了。

OSI模型的七层框架

OSI模型定义了网络互连的七层框架（物理层、数据链路层、网络层、传输层、会话层、表示层、应用层），每一层实现各自的功能和协议，并完成与相邻层的接口通信。OSI模型各层的通信协议，大致举例如下表所示：

表：OSI模型各层的通信协议举例

应用层	**HTTP、SMTP、SNMP、FTP、Telnet、SIP、SSH、NFS、RTSP、XMPP、Whois、ENRP、等等**
表示层	XDR、ASN.1、SMB、AFP、NCP、等等
会话层	ASAP、SSH、RPC、NetBIOS、ASP、Winsock、BSD Sockets、等等
传输层	TCP、UDP、TLS、RTP、SCTP、SPX、ATP、IL、等等
网络层	IP、ICMP、IGMP、IPX、BGP、OSPF、RIP、IGRP、EIGRP、ARP、RARP、X.25、等等
数据链路层	以太网、令牌环、HDLC、帧中继、ISDN、ATM、IEEE 802.11、FDDI、PPP、等等
物理层	例如铜缆、网线、光缆、无线电等等

TCP/IP协议是Internet互联网最基本的协议，其在一定程度上参考了七层ISO模型。

OSI模型共有七层，从下到上分别是物理层、数据链路层、网络层、运输层、会话层、表示层和应用层。但是这显然是有些复杂的，所以在TCP/IP协议中，七层被简化为了四个层次。

TCP/IP模型中的各种协议，依其功能不同，被分别归属到这四层之中，常被视为是简化过后的七层OSI模型。

TCP/IP协议与七层ISO模型的对应关系

TCP/IP协议与七层ISO模型的对应关系，大致如下图所示：

图：TCP/IP协议与七层ISO模型的对应关系

TCP/IP协议的应用层的主要协议有HTTP、Telnet、FTP、SMTP等，是用来读取来自传输层的数据或者将数据传输写入传输层；传输层的主要协议有UDP、TCP，实现端对端的数据传输；网络层的主要协议有ICMP、IP、IGMP，主要负责网络中数据包的传送等；链路层有时也称作数据链路层或网络接口层，主要协议有ARP、RARP，
通常包括操作系统中的设备驱动程序和计算机中对应的网络接口卡，它们一起处理与传输媒介（如电缆或其他物理设备）的物理接口细节。

（一）TCP/IP协议的应用层

应用层包括所有和应用程序协同工作，并利用基础网络交换应用程序的业务数据的协议。一些特定的程序被认为运行在这个层上，该层协议所提供的服务能直接支持用户应用。应用层协议包括HTTP（万维网服务）、FTP（文件传输）、SMTP（电子邮件）、SSH（安全远程登陆）、DNS（域名解析）以及许多其他协议。

（二）TCP/IP协议的传输层

传输层的协议，解决了诸如端到端可靠性问题，能确保数据可靠的到达目的地，甚至能保证数据按照正确的顺序到达目的地。传输层的主要功能大致如下：

（1）为端到端连接提供传输服务；

（2）这种传输服务分为可靠和不可靠的，其中TCP是典型的可靠传输，而UDP则是不可靠传输；

（3）为端到端连接提供流量控制、差错控制、QoS(Quality of Service)服务质量等管理服务。

传输层主要有两个性质不同的协议：TCP传输控制协议和UDP用户数据报协议。

TCP协议是一个面向连接的、可靠的传输协议，它提供一种可靠的字节流，能保证数据完整、无损并且按顺序到达。TCP尽量连续不断地测试网络的负载并且控制发送数据的速度以避免网络过载。另外，TCP试图将数据按照规定的顺序发送。

UDP协议是一个无连接的数据报协议，是一个“尽力传递”和“不可靠”协议，不会对数据包是否已经到达目的地进行检查，并且不保证数据包按顺序到达。

总体来说，TCP协议传输效率低，但可靠性强；UDP协议传输效率高，但可靠性略低，适用于传输可靠性要求不高、体量小的数据（比如QQ聊天数据）。

（三）TCP/IP协议的网络层

TCP/IP协议网络层的作用是在复杂的网络环境中为要发送的数据报找到一个合适的路径进行传输。简单来说，网络层负责将数据传输到目标地址，目标地址可以是多个网络通过路由器连接而成的某一个地址。另外，网络层负责寻找合适的路径到达对方计算机，并把数据帧传送给对方，网络层还可以实现拥塞控制、网际互连等功能。网络层协议的代表包括：ICMP、IP、IGMP等。

（四）TCP/IP协议的链路层

链路层有时也称作数据链路层或网络接口层，用来处理连接网络的硬件部分。该层既包括操作系统硬件的设备驱动、NIC（网卡）、光纤等物理可见部分，还包括连接器等一切传输媒介。

在这一层，数据的传输单位为比特。

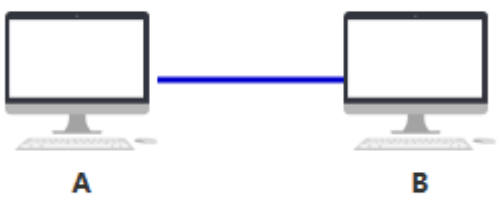
其主要协议有ARP、RARP等。

图解 物理层：使用MAC解决设备的身份证问题

通信的原始时代

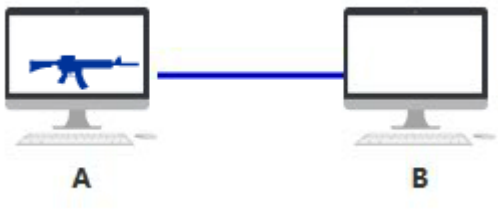
很久很久之前，你不与任何其他电脑相连接，孤苦伶仃。

直到有一天，你希望与另一台电脑 B 建立通信，于是你们各开了一个网口，用一根**网线**连接了起来。



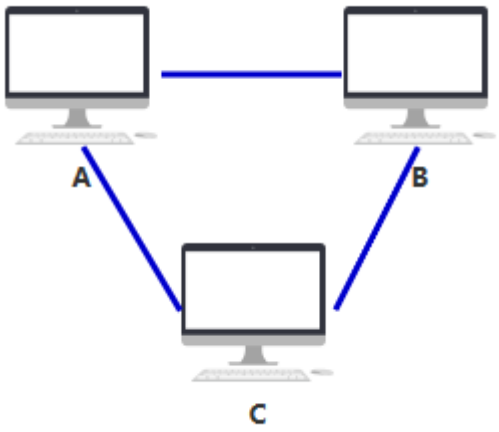
用一根网线连接起来怎么就能"通信"了呢？我可以给你讲 IO、讲中断、讲缓冲区，但这不是研究网络时该关心的问题。

如果你纠结，要么去研究一下操作系统是如何处理网络 IO 的，要么去研究一下包是如何被网卡转换成电信号发送出去的，要么就仅仅把它当做电脑里有个小人在**开枪**吧~

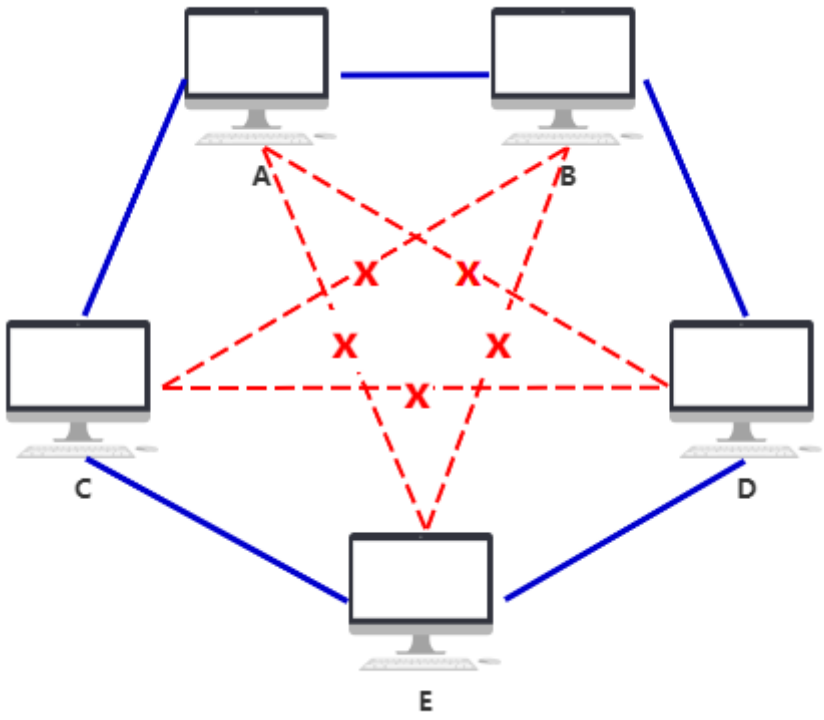


反正，你们就是连起来了，并且可以通信。

有一天，一个新伙伴 C 加入了，但聪明的你们很快发现，可以每个人开**两个网口**，用一共**三根网线**，彼此相连。

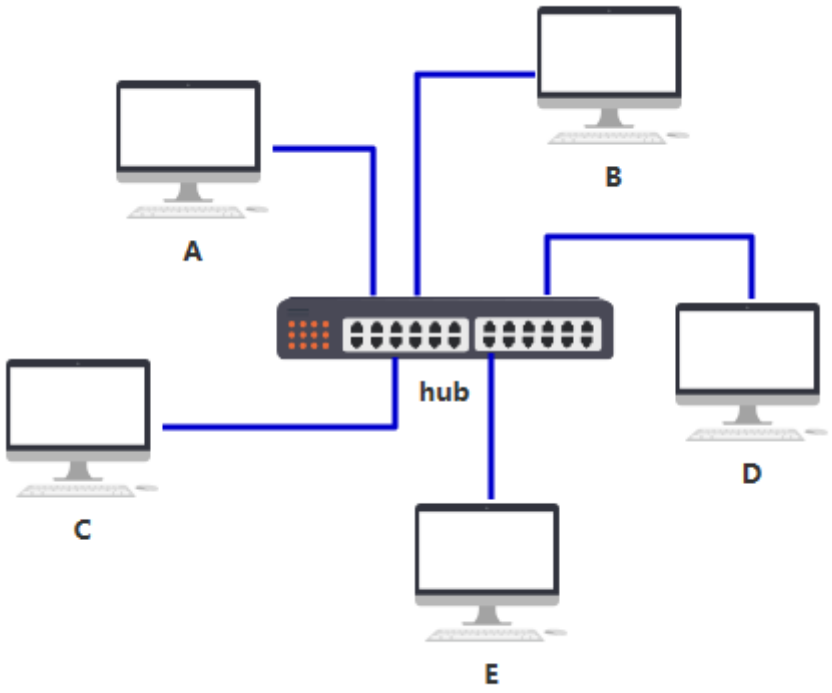


随着越来越多的人加入，你发现身上开的网口实在太多了，而且网线密密麻麻，混乱不堪。（而实际上一台电脑根本开不了这么多网口，所以这种连线只在理论上可行，所以连不上的我就用红色虚线表示了，就是这么严谨哈哈~）

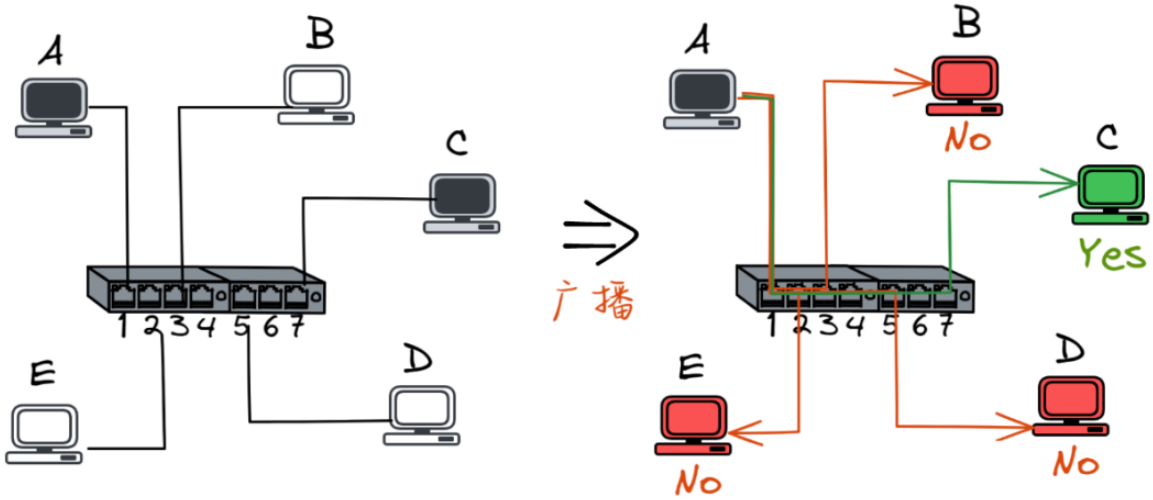


集线器的诞生

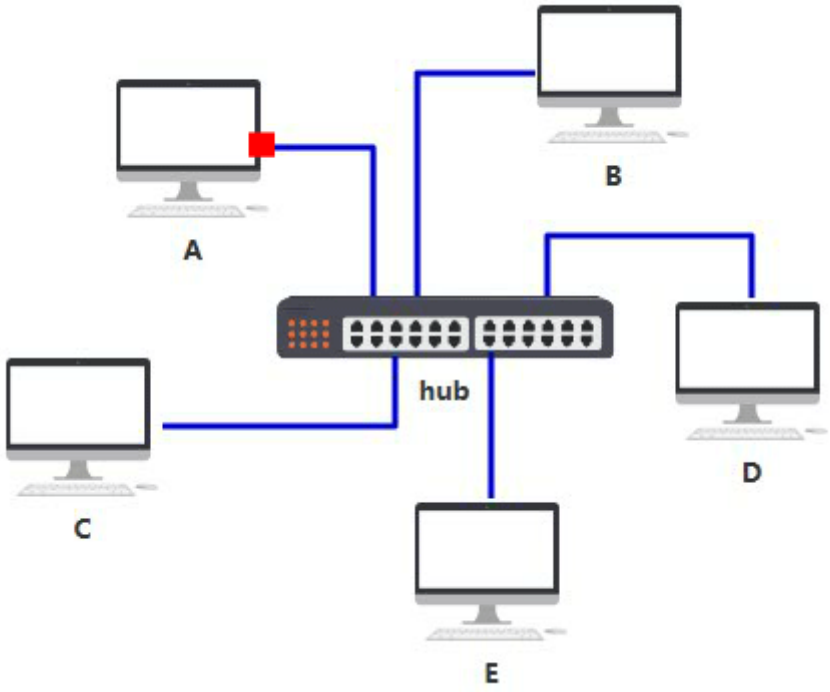
于是你们发明了一个中间设备，你们将网线都插到这个设备上，由这个设备做转发，就可以彼此之间通信了，本质上和原来一样，只不过网口的数量和网线的数量减少了，不再那么混乱。



你给它取名叫**集线器**，集线器的通讯手段很简单，就是两个字：广播



它仅仅是无脑将电信号**转发到所有出口（广播）**，不做任何处理，你觉得它是没有智商的，因此把人家定性在了**物理层**。

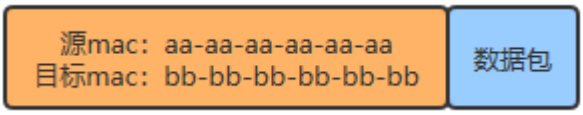


由于转发到了所有出口，那 BCDE 四台机器怎么知道数据包是不是发给自己的呢?

首先，你要给所有的连接到交换机的设备，都起个名字。原来你们叫 ABCD，但现在需要一个更专业的，**全局唯一**的名字作为标识，你把这个更高端的名字称为 **MAC 地址。**

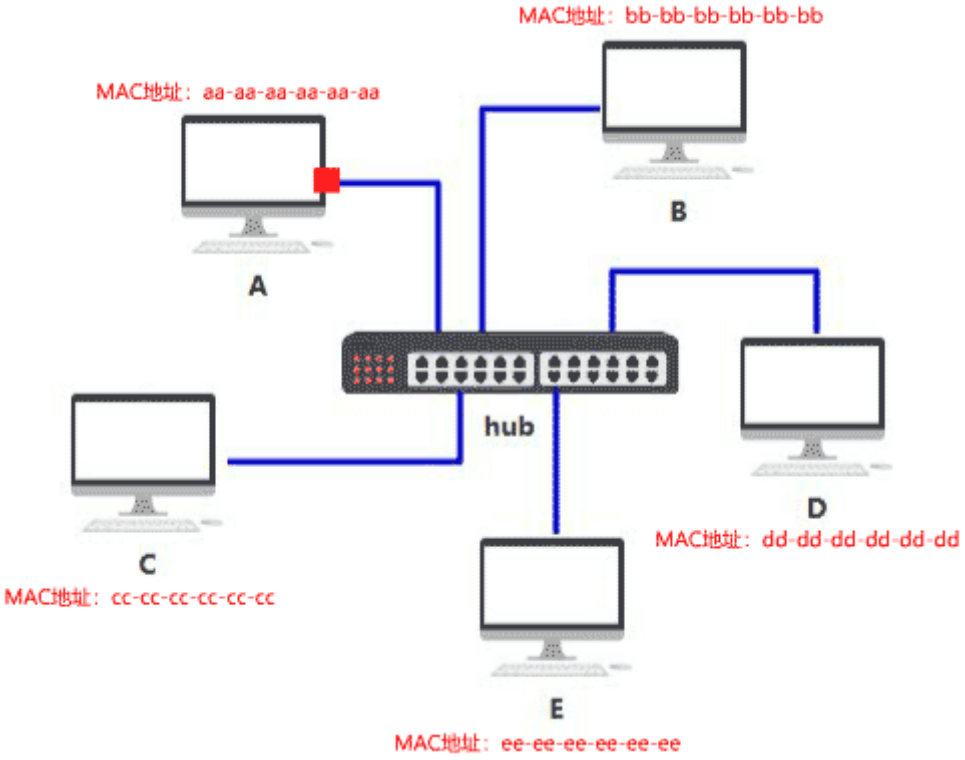
你的 MAC 地址是 aa-aa-aa-aa-aa-aa，你的伙伴 b 的 MAC 地址是 bb-bb-bb-bb-bb-bb，以此类推，不重复就好。

这样，A 在发送数据包给 B 时，只要在头部拼接一个这样结构的数据，就可以了。



B 在收到数据包后，根据头部的目标 MAC 地址信息，判断这个数据包的确是发给自己的，于是便**收下。**

其他的 CDE 收到数据包后，根据头部的目标 MAC 地址信息，判断这个数据包并不是发给自己的，于是便**丢弃。**

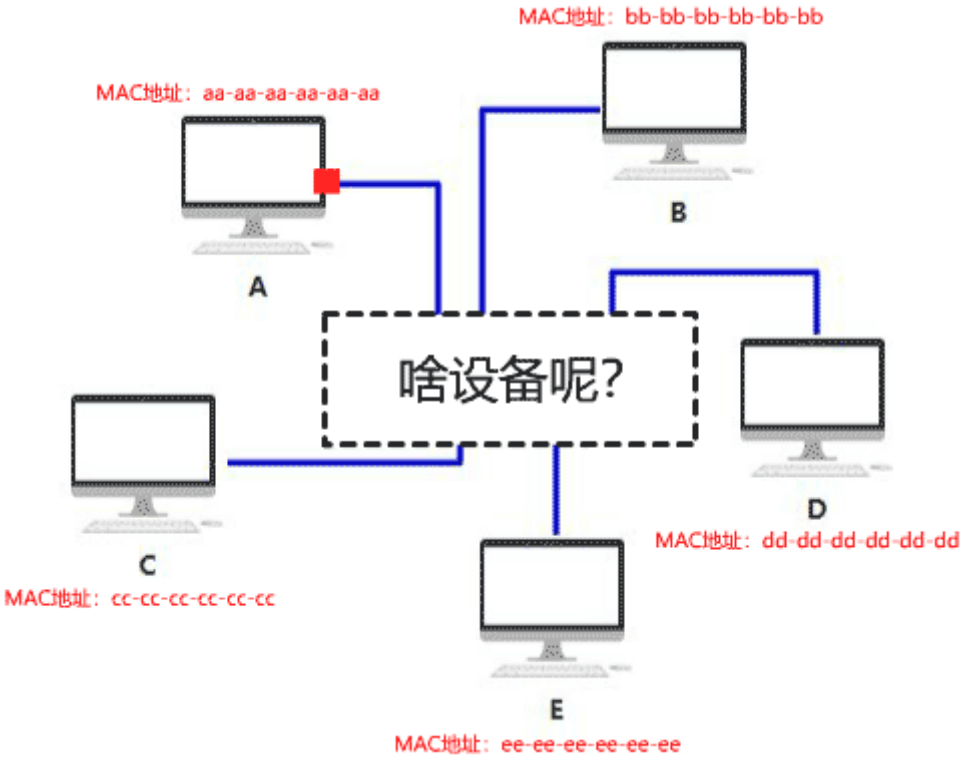


虽然集线器使整个布局干净不少，但原来我只要发给电脑 B 的消息，现在却要发给连接到集线器中的所有电脑，这样既不安全，又不节省网络资源。

图解 数据链路：使用交换机解决MAC 地址映射问题

集线器的问题

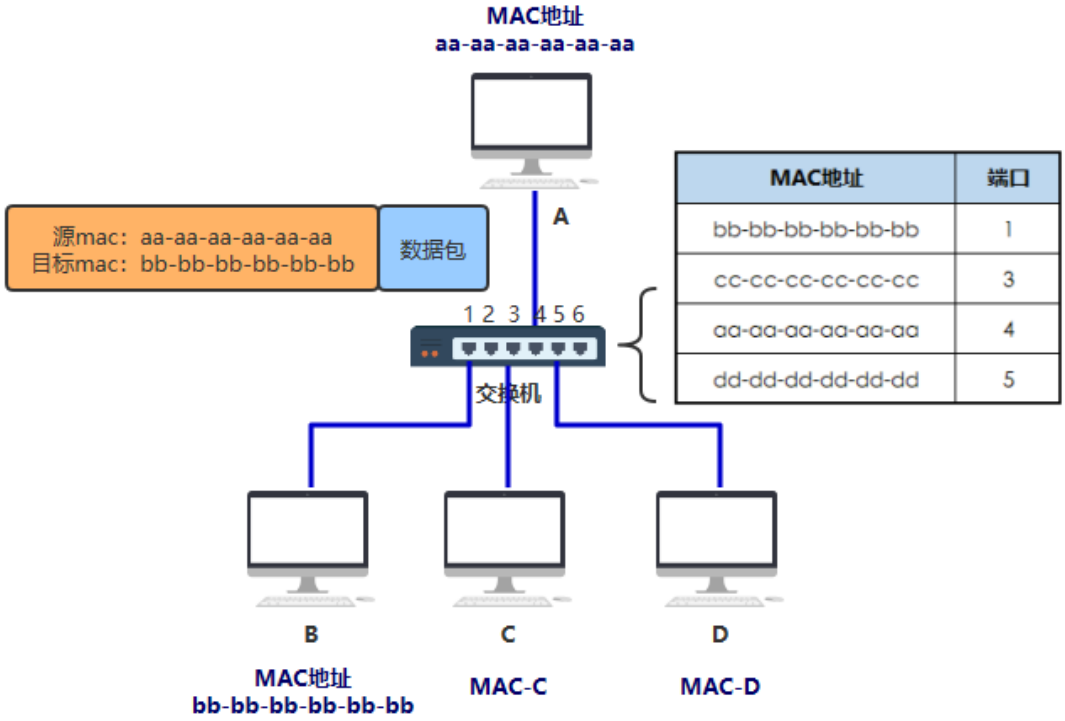
如果把这个集线器弄得更智能一些，**只发给目标 MAC 地址指向的那台电脑**，就好了。



交换机的诞生

虽然只比集线器多了这一点点区别，但看起来似乎有智能了，你把这东西叫做**交换机**。

也正因为这一点点智能，你把它放在了另一个层级，**数据链路层**。

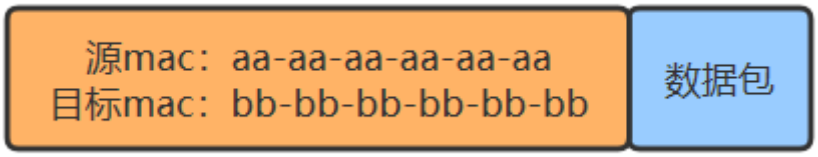


如上图所示，你是这样设计的。

交换机内部维护一张 **MAC 地址表**，记录着每一个 MAC 地址的设备，连接在其哪一个端口上。

MAC 地址	端口
bb-bb-bb-bb-bb-bb	1
cc-cc-cc-cc-cc-cc	3
aa-aa-aa-aa-aa-aa	4
dd-dd-dd-dd-dd-dd	5

假如你仍然要发给 B 一个数据包，构造了如下的数据结构从网口出去。

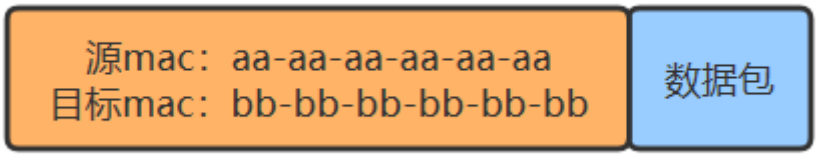


到达交换机时，交换机内部通过自己维护的 MAC 地址表，发现**目标机器 B 的 MAC 地址 bb-bb-bb-bb-bb-bb 映射到了端口 1 上**，于是把数据从 1 号端口发给了 B，完事~

你给这个通过这样传输方式而组成的小范围的网络，叫做**以太网。**

当然最开始的时候，MAC 地址表是空的，是怎么逐步建立起来的呢?

假如在 MAC 地址表为空是，你给 B 发送了如下数据



由于这个包从端口 4 进入的交换机，所以此时交换机就可以在 MAC地址表记录第一条数据：

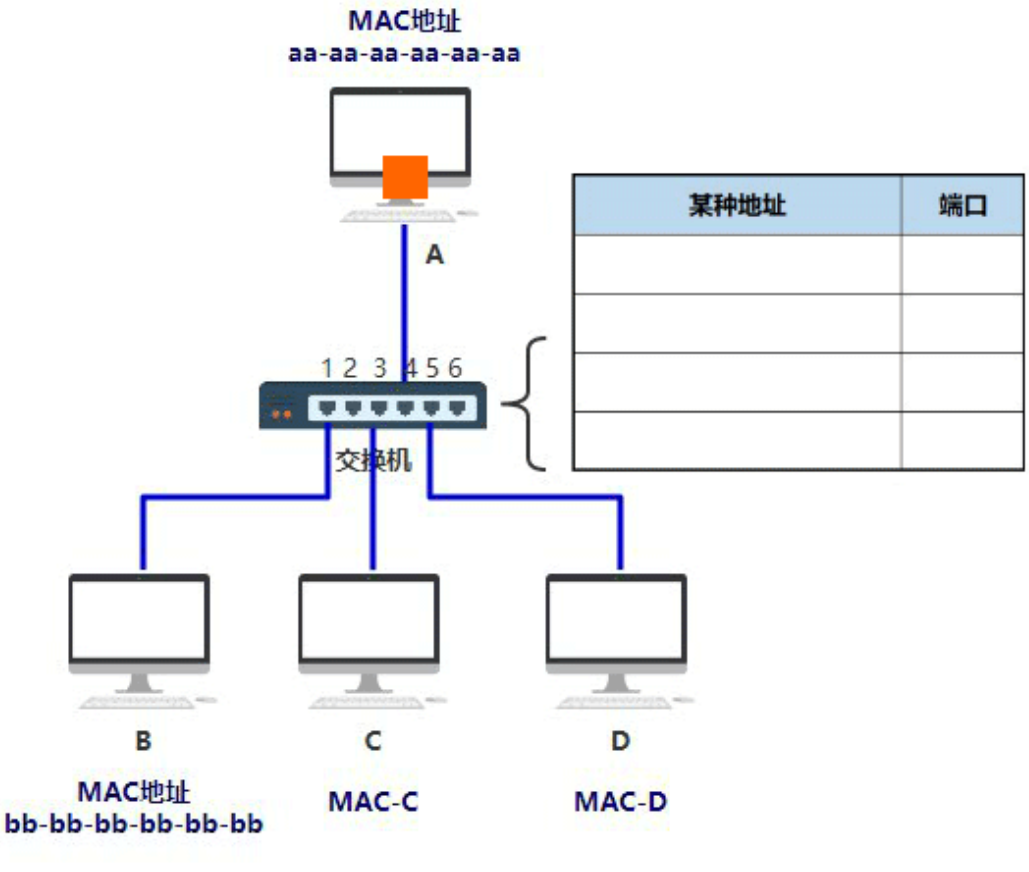
MAC：aa-aa-aa-aa-aa-aa-aa
端口：4

交换机看目标 MAC 地址（bb-bb-bb-bb-bb-bb）在地址表中并没有映射关系，于是将此包发给了**所有端口**，也即发给了所有机器。

之后，只有机器 B 收到了确实是发给自己的包，于是做出了**响应**，响应数据从端口 1 进入交换机，于是交换机此时在地址表中更新了第二条数据：

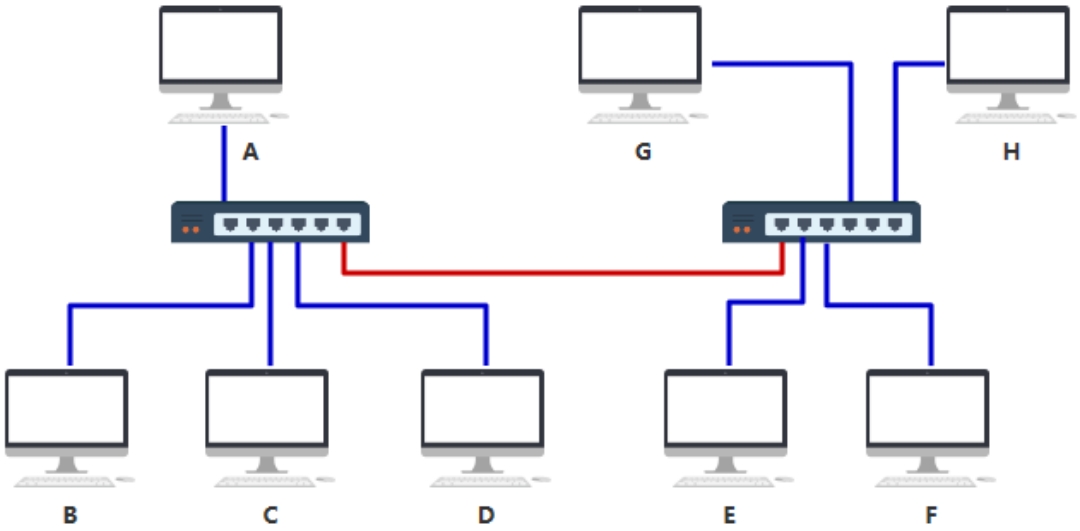
MAC：bb-bb-bb-bb-bb-bb
端口：1

过程如下



经过该网络中的机器不断地通信，交换机最终将 MAC 地址表建立完毕~

随着机器数量越多，交换机的端口也不够了，但聪明的你发现，只要将多个交换机连接起来，这个问题就轻而易举搞定~



你完全不需要设计额外的东西，只需要按照之前的设计和规矩来，按照上述的接线方式即可完成所有电脑的互联，所以交换机设计的这种规则，真的很巧妙。你想想看为什么（比如 A 要发数据给 F）。

但是你要注意，上面那根红色的线，最终在 MAC 地址表中可不是一条记录呀，而是要把 EFGH 这四台机器与该端口（端口6）的映射全部记录在表中。

MAC 地址和端口的映射记录

最终，**两个交换机将分别记录 A ~ H 所有机器的映射记录。**

左边的交换机

MAC 地址	端口
bb-bb-bb-bb-bb-bb	1
cc-cc-cc-cc-cc-cc	3
aa-aa-aa-aa-aa-aa	4
dd-dd-dd-dd-dd-dd	5
ee-ee-ee-ee-ee-ee	6
ff-ff-ff-ff-ff-ff	6
gg-gg-gg-gg-gg-gg	6
hh-hh-hh-hh-hh-hh	6

右边的交换机

MAC 地址	端口
bb-bb-bb-bb-bb-bb	1
cc-cc-cc-cc-cc-cc	1
aa-aa-aa-aa-aa-aa	1
dd-dd-dd-dd-dd-dd	1
ee-ee-ee-ee-ee-ee	2
ff-ff-ff-ff-ff-ff	3
gg-gg-gg-gg-gg-gg	4
hh-hh-hh-hh-hh-hh	6

这在只有 8 台电脑的时候还好，甚至在只有几百台电脑的时候，都还好，所以这种交换机的设计方式，已经足足支撑一阵子了。

但很遗憾，人是贪婪的动物，很快，电脑的数量就发展到几千、几万、几十万。

图解 传输层：IP地址和路由器

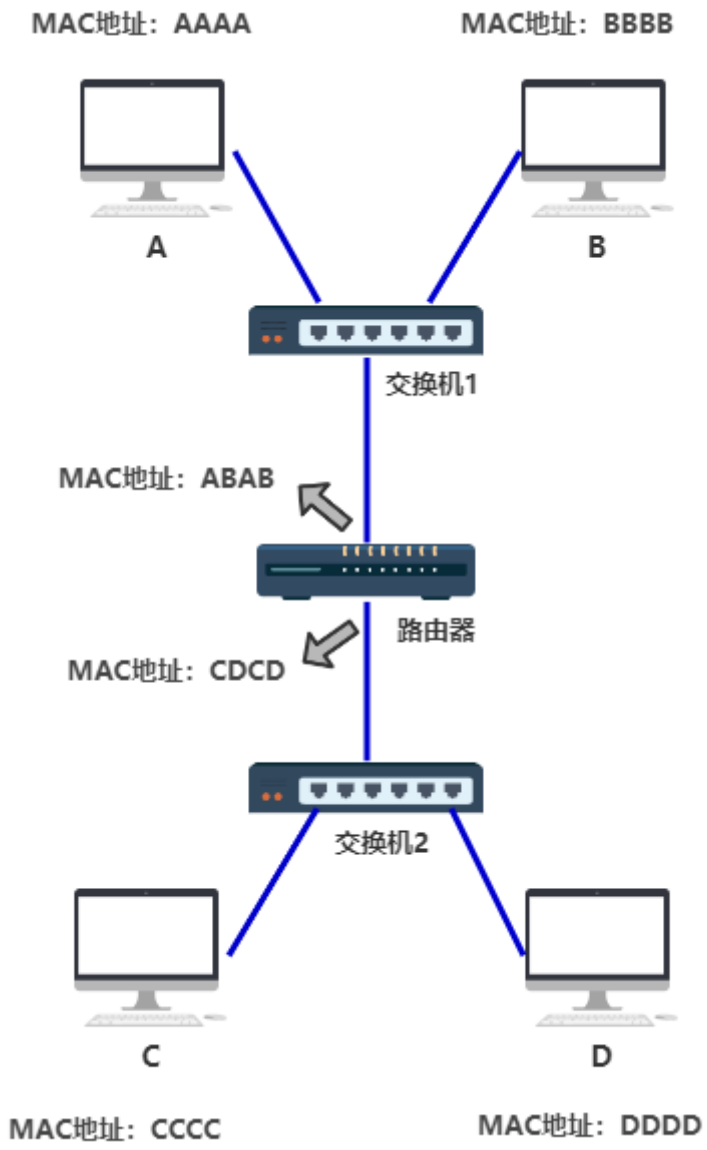
二层交换机的问题

交换机已经无法记录如此庞大的映射关系了。

此时你动了歪脑筋，你发现了问题的根本在于，连出去的那根红色的网线，后面不知道有多少个设备不断地连接进来，从而使得地址表越来越大。

那我可不可以让那根红色的网线，接入一个**新的设备**，这个设备就跟电脑一样有自己独立的 MAC 地址，而且同时还能帮我把数据包做一次**转发**呢？

这个设备就是**路由器，**它的功能就是，作为一台独立的拥有 MAC 地址的设备，并且可以帮我把数据包做一次转发，你把它定在了**网络层。**



注意，路由器的每一个端口，都有独立的 MAC 地址

好了，现在交换机的 MAC 地址表中，只需要多出一条 MAC 地址 ABAB 与其端口的映射关系，就可以成功把数据包转交给路由器了，这条搞定。

那如何做到，把发送给 C 和 D，甚至是把发送给 DEFGH.... 的数据包，统统先发送给路由器呢?

不难想到这样一个点子，假如电脑 C 和 D 的 MAC 地址拥有共同的前缀，比如分别是

C 的 MAC 地址：FFFF-FFFF-CCCC D 的 MAC 地址：FFFF-FFFF-DDDD

那我们就可以说，将目标 MAC 地址为 **FFFF-FFFF-? 开头的**，统统先发送给路由器。

这样是否可行呢？答案是否定的。

IP地址的诞生

我们先从现实中 MAC 地址的结构入手，MAC地址也叫物理地址、硬件地址，长度为 48 位，一般这样来表示

00-16-EA-AE-3C-40

它是由网络设备制造商生产时烧录在网卡的EPROM（一种闪存芯片，通常可以通过程序擦写）。

其中**前 24 位（00-16-EA）代表网络硬件制造商的编号**，后 24 位（AE-3C-40）是该厂家自己分配的，一般表示系列号。

只要不更改自己的 MAC 地址，MAC 地址在世界是唯一的。形象地说，MAC地址就如同身份证上的身份证号码，具有唯一性。

那如果你希望向上面那样表示将目标 MAC 地址为 **FFFF-FFFF-? 开头的**，统一从路由器出去发给某一群设备（后面会提到这其实是子网的概念），那你就需要要求某一子网下统统买一个厂商制造的设备，要么你就需要要求厂商在生产网络设备烧录 MAC 地址时，提前按照你规划好的子网结构来定 MAC 地址，并且日后这个网络的结构都不能轻易改变。

这显然是不现实的。

于是你发明了一个新的地址，给每一台机器一个 32 位的编号，如：

11000000101010000000000000000001

你觉得有些不清晰，于是把它分成四个部分，中间用点相连。

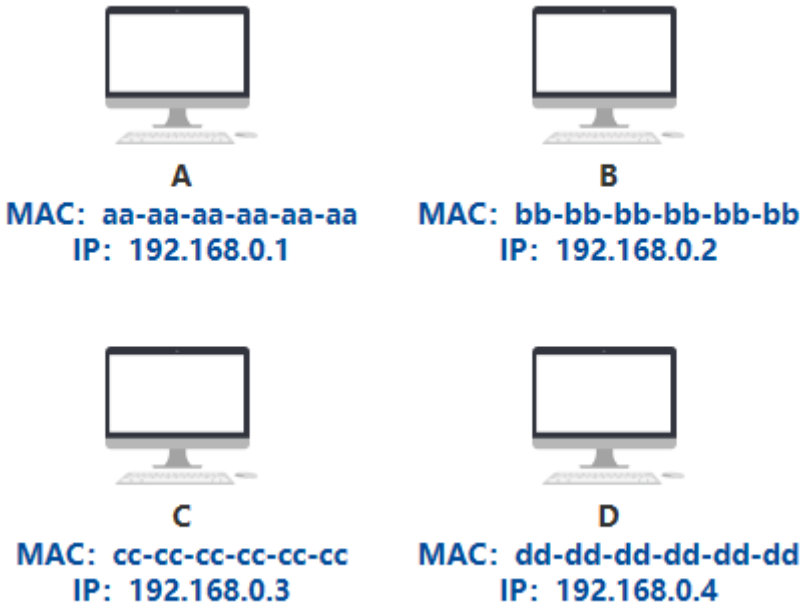
11000000.10101000.00000000.00000001

你还觉得不清晰，于是把它转换成 10 进制。

192.168.0.1

最后你给了这个地址一个响亮的名字，**IP 地址**。现在每一台电脑，同时有自己的 MAC 地址，又有自己的 IP 地址，只不过 IP 地址是**软件层面**上的，可以随时修改，MAC 地址一般是无法修改的。

这样一个可以随时修改的 IP 地址，就可以根据你规划的网络拓扑结构，来调整了。



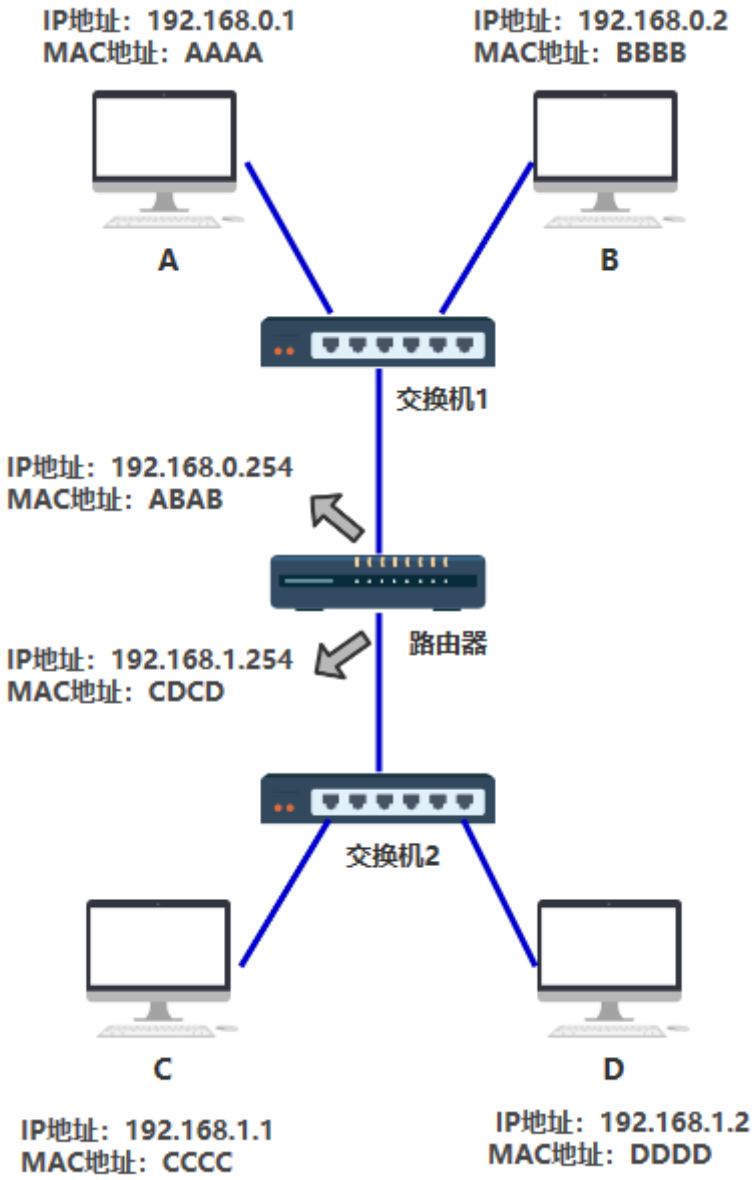
如上图所示，假如我想要发送数据包给 ABCD 其中一台设备，不论哪一台，我都可以这样描述，**"将 IP 地址为 192.168.0 开头的全部发送给到路由器，之后再怎么转发，交给它！"**，巧妙吧。

路由器的诞生

路由器诞生了，专门负责IP地址的寻找。那报文交给路由器之后，路由器又是怎么把数据包准确转发给指定设备的呢？

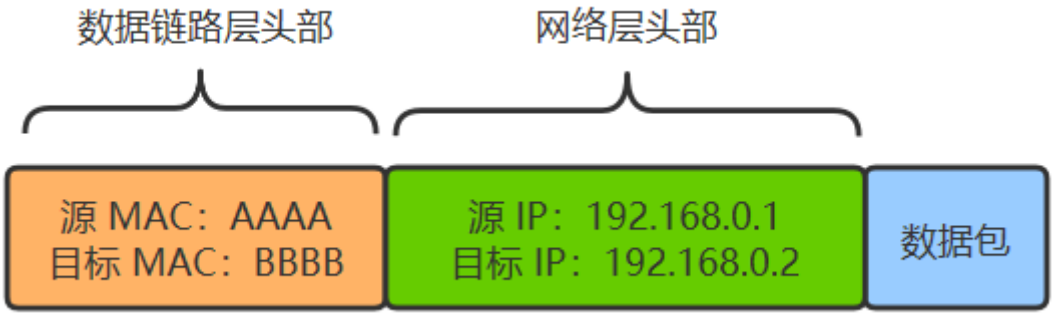
别急我们慢慢来。

我们先给上面的组网方式中的每一台设备，加上自己的 IP 地址



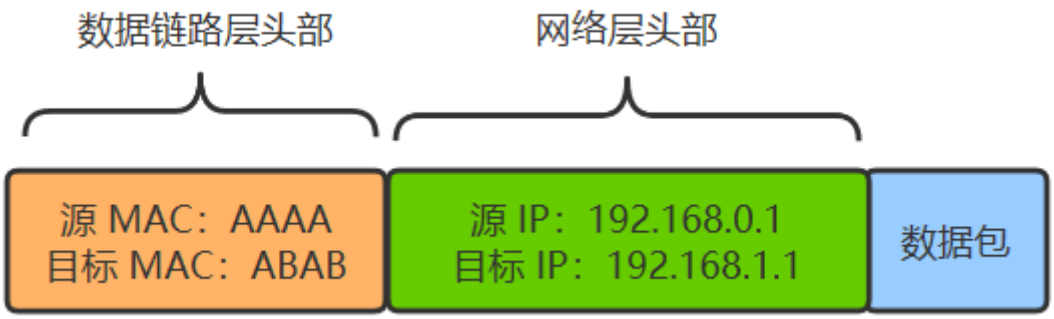
现在两个设备之间传输，除了加上数据链路层的头部之外，还要再增加一个网络层的头部。

假如 A 给 B 发送数据，由于它们直接连着交换机，所以 A 直接发出如下数据包即可，其实网络层没有体现出作用。

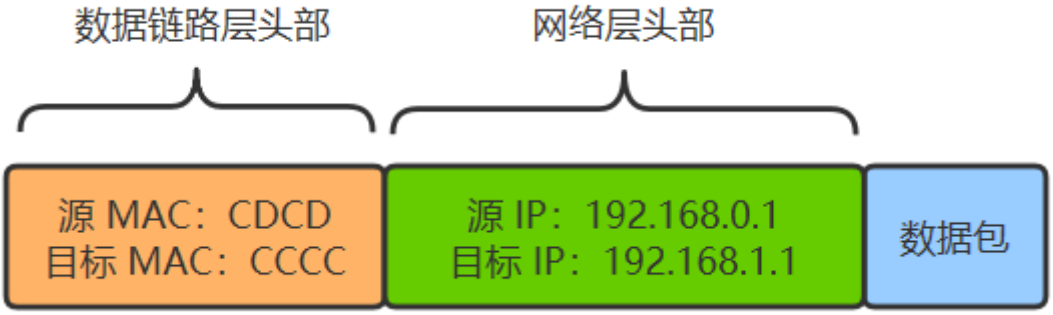


但假如 A 给 C 发送数据，A 就需要先转交给路由器，然后再由路由器转交给 C。由于最底层的传输仍然需要依赖以太网，所以数据包是分成两段的。

A ~ 路由器这段的包如下：



路由器到 C 这段的包如下：



好了，上面说的两种情况（A->B，A->C），相信细心的读者应该会有不少疑问，下面我们一个个来展开。

子网的由来

A 给 C 发数据包，怎么知道是否要通过路由器转发呢?

答案：子网

如果源 IP 与目的 IP 处于一个子网，直接将包通过交换机发出去。

如果源 IP 与目的 IP 不处于一个子网，就交给路由器去处理。

好，那现在只需要解决，什么叫处于一个子网就好了。

- 192.168.0.1 和 192.168.0.2 处于同一个子网
- 192.168.0.1 和 192.168.1.1 处于不同子网

这两个是我们人为规定的，即我们想表示，对于 192.168.0.1 来说：

192.168.0.xxx 开头的，就算是在一个子网，否则就是在不同的子网。

那对于计算机来说，怎么表达这个意思呢？于是人们发明了**子网掩码**的概念

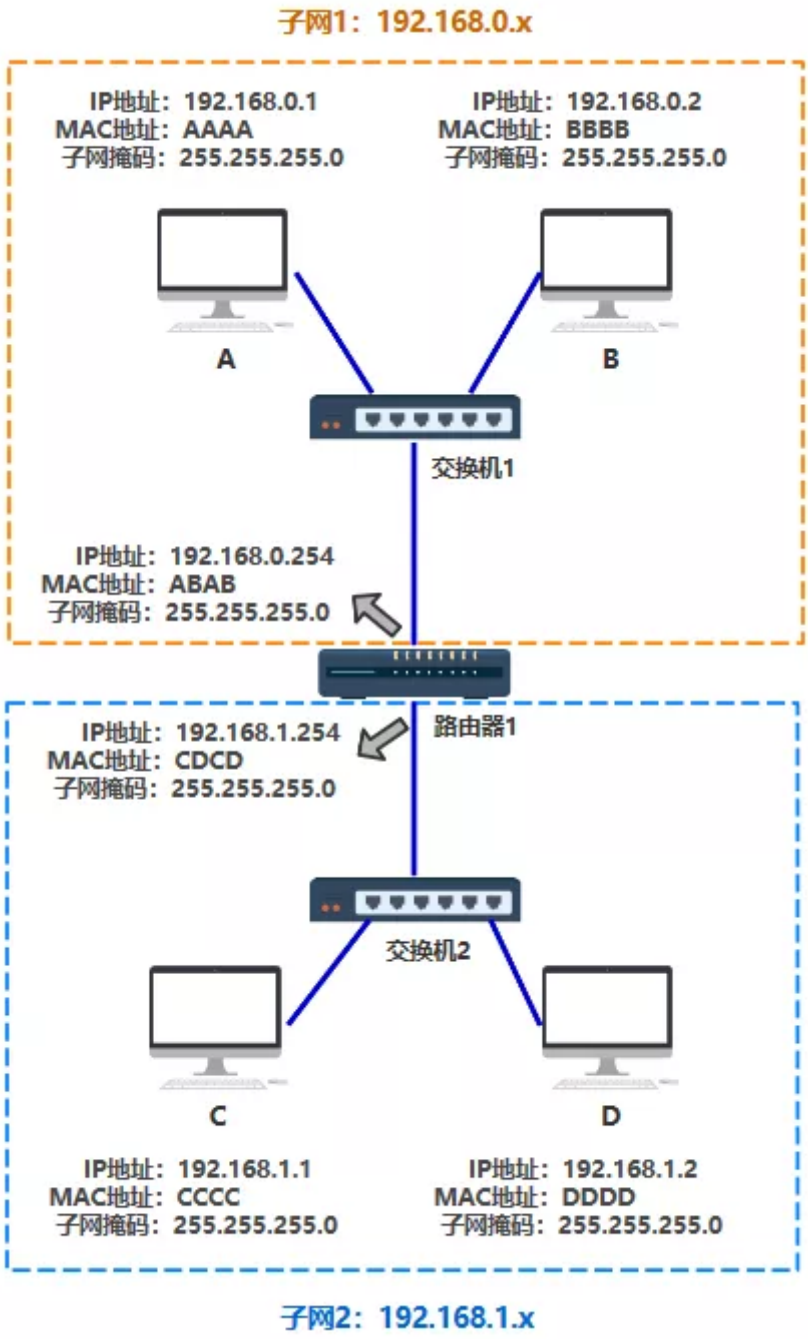
假如某台机器的子网掩码定为 255.255.255.0

这表示，将源 IP 与目的 IP 分别同这个子网掩码进行**与运算**，相等则是在一个子网，不相等就是在不同子网，就这么简单。

比如

- **A电脑**：192.168.0.1 & 255.255.255.0 = 192.168.0.0
- **B电脑**：192.168.0.2 & 255.255.255.0 = 192.168.0.0
- **C电脑**：192.168.1.1 & 255.255.255.0 = 192.168.1.0
- **D电脑**：192.168.1.2 & 255.255.255.0 = 192.168.1.0

那么 A 与 B 在同一个子网，C 与 D 在同一个子网，但是 A 与 C 就不在同一个子网，与 D 也不在同一个子网，以此类推。



所以如果 A 给 C 发消息，A 和 C 的 IP 地址分别 & A 机器配置的子网掩码，发现不相等，则 A 认为 C 和自己不在同一个子网，于是把包发给路由器，就不管了，**之后怎么转发，A 不关心。**

A 如何知道，哪个设备是路由器?

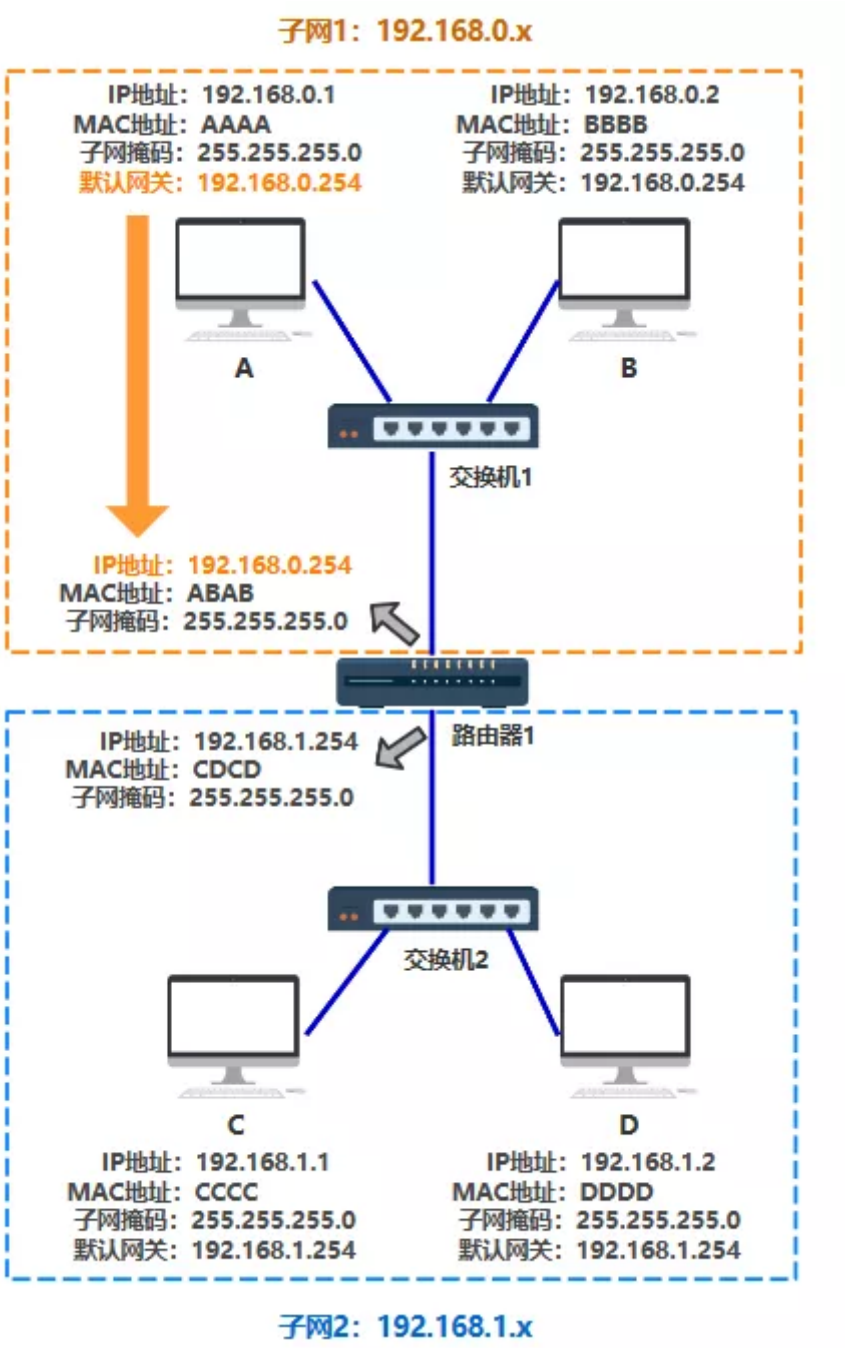
答案：在 A 上要设置默认网关

上一步 A 通过是否与 C 在同一个子网内，判断出自己应该把包发给路由器，那路由器的 IP 是多少呢?

其实说发给路由器不准确，应该说 A 会把包发给**默认网关。**

对 A 来说，A 只能**直接**把包发给同处于一个子网下的某个 IP 上，所以发给路由器还是发给某个电脑，对 A 来说也不关心，只要这个设备有个 IP 地址就行。

所以**默认网关，就是 A 在自己电脑里配置的一个 IP 地址**，以便在发给不同子网的机器时，发给这个 IP 地址。



仅此而已！

路由表的由来（和Mac表的由来好像，都是逼出来的）

路由器如何知道C在哪里?

答案：路由表

现在 A 要给 C 发数据包，已经可以成功发到路由器这里了，最后一个问题就是，**路由器怎么知道，收到的这个数据包，该从自己的哪个端口出去**，才能直接（或间接）地最终到达目的地 C 呢。

路由器收到的数据包有目的 IP 也就是 C 的 IP 地址，需要转化成从自己的哪个端口出去，很容易想到，应该有个表，就像 MAC 地址表一样。

这个表就叫**路由表。**

至于这个路由表是怎么出来的，有很多路由算法，本文不展开，因为我也不会哈哈~

不同于 MAC 地址表的是，路由表并不是一对一这种明确关系，我们下面看一个路由表的结构。

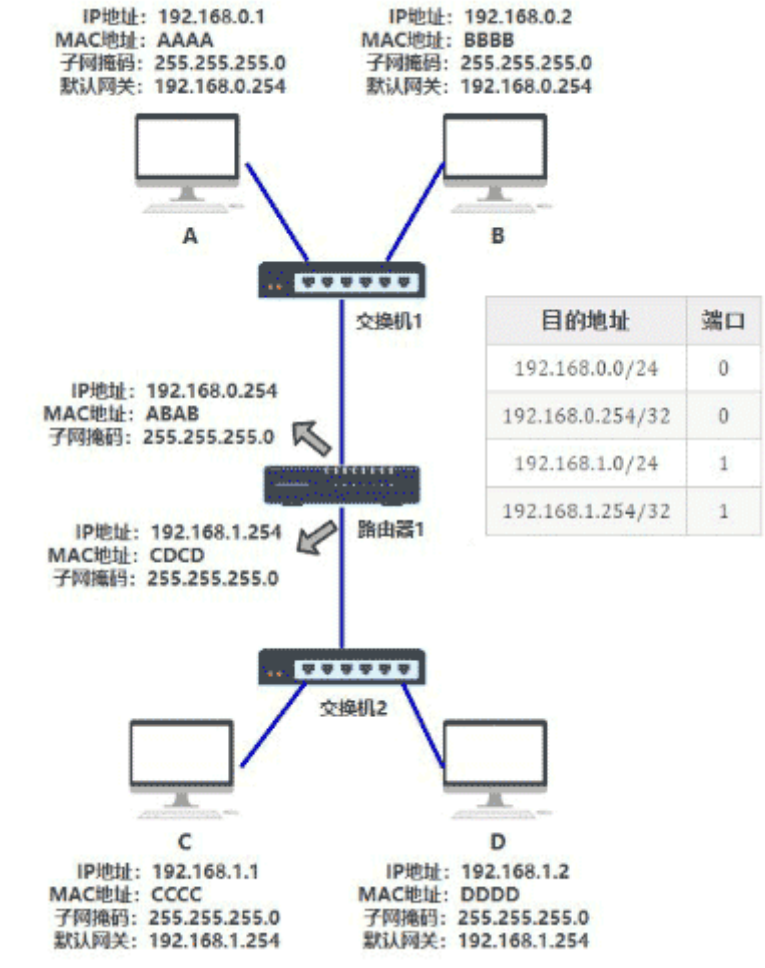
目的地址	子网掩码	下一跳	端口
192.168.0.0	255.255.255.0		0
192.168.0.254	255.255.255.255		0
192.168.1.0	255.255.255.0		1
192.168.1.254	255.255.255.255		1

我们学习一种新的表示方法，由于子网掩码其实就表示前多少位表示子网的网段，所以如 192.168.0.0（255.255.255.0） 也可以简写为 192.168.0.0/24

目的地址	下一跳	端口
192.168.0.0/24		0
192.168.0.254/32		0
192.168.1.0/24		1
192.168.1.254/32		1

这就很好理解了，路由表就表示，**192.168.0.xxx 这个子网下的，都转发到 0 号端口，192.168.1.xxx 这个子网下的，都转发到 1 号端口。**下一跳列还没有值，我们先不管

配合着结构图来看（这里把子网掩码和默认网关都补齐了）



刚才说的都是 IP 层，但发送数据包的数据链路层需要知道 MAC 地址，可是我只知道 IP 地址该怎么办呢?

答案：arp

假如你（A）此时**不知道**你同伴 B 的 MAC 地址（现实中就是不知道的，刚刚我们只是假设已知），你只知道它的 IP 地址，你该怎么把数据包准确传给 B 呢?

答案很简单，在网络层，**我需要把 IP 地址对应的 MAC 地址找到**，也就是通过某种方式，找到 **192.168.0.2** 对应的 MAC 地址 **BBBB。**

这种方式就是 **arp 协议**，同时电脑 A 和 B 里面也会有一张 **arp 缓存表**，表中记录着 **IP 与 MAC 地址**的对应关系。

IP 地址	MAC 地址
192.168.0.2	BBBB

一开始的时候这个表是**空的**，电脑 A 为了知道电脑 B（192.168.0.2）的 MAC 地址，将会**广播**一条 arp 请求，B 收到请求后，带上自己的 MAC 地址给 A 一个**响应。**此时 A 便更新了自己的 arp 表。

这样通过大家不断广播 arp 请求，最终所有电脑里面都将 arp 缓存表更新完整。

图解：整个传输过程

从各个节点的视角来看

电脑视角：

- 首先我要知道我的 IP 以及对方的 IP
- 通过子网掩码判断我们是否在同一个子网
- 在同一个子网就通过 arp 获取对方 mac 地址直接扔出去
- 不在同一个子网就通过 arp 获取默认网关的 mac 地址直接扔出去

交换机视角：

- 我收到的数据包必须有目标 MAC 地址
- 通过 MAC 地址表查映射关系
- 查到了就按照映射关系从我的指定端口发出去
- 查不到就所有端口都发出去

路由器视角：

- 我收到的数据包必须有目标 IP 地址
- 通过路由表查映射关系
- 查到了就按照映射关系从我的指定端口发出去（不在任何一个子网范围，走其路由器的默认网关也是查到了）
- 查不到则返回一个路由不可达的数据包

如果你嗅觉足够敏锐，你应该可以感受到下面这句话：

网络层（IP协议）本身没有传输包的功能，包的实际传输是委托给数据链路层（以太网中的交换机）来实现的。

涉及到的三张表分别是

- 交换机中有 **MAC 地址**表用于映射 MAC 地址和它的端口

- 路由器中有**路由表**用于映射 IP 地址(段)和它的端口
- 电脑和路由器中都有** arp 缓存表**用于缓存 IP 和 MAC 地址的映射关系

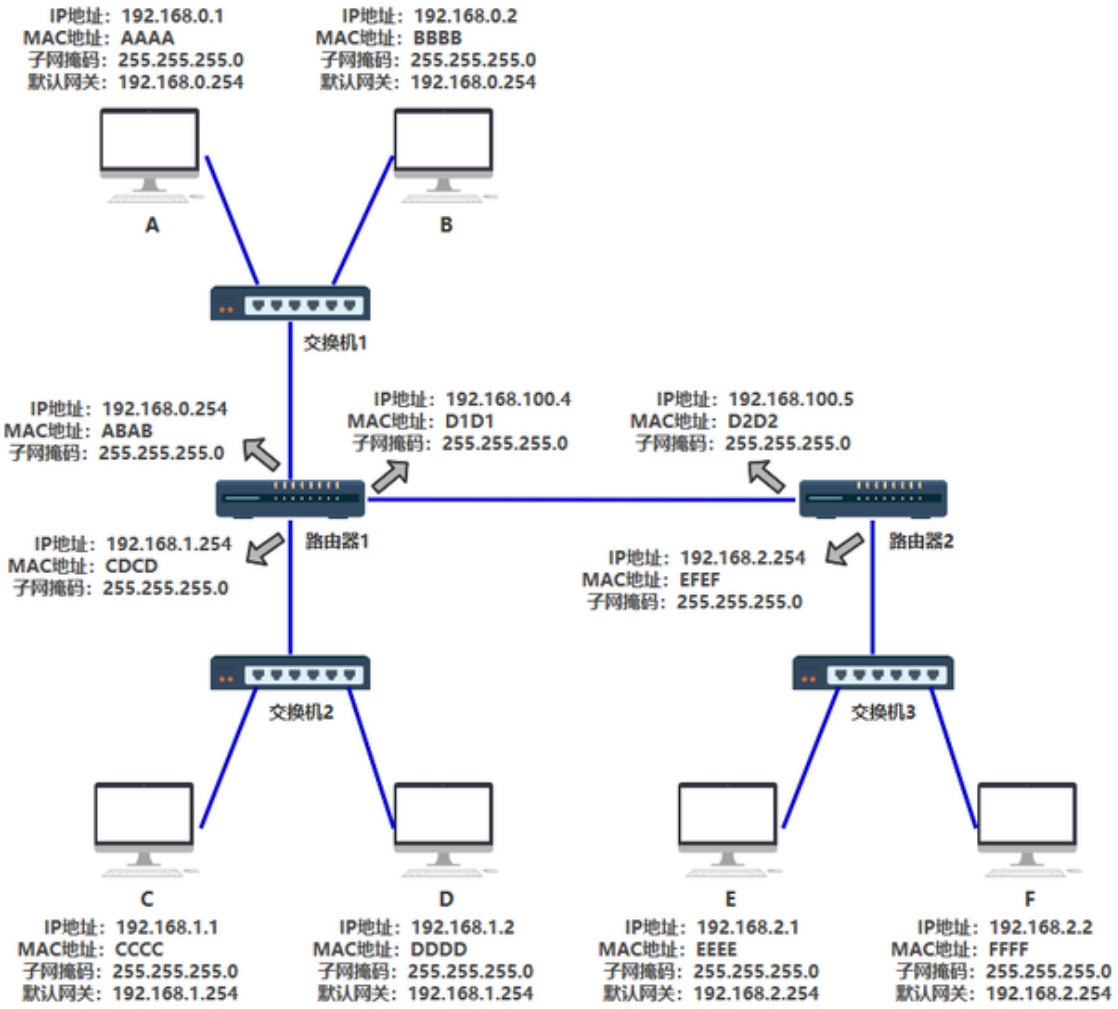
这三张表是怎么来的

- MAC 地址表是通过以太网内各节点之间不断通过交换机通信，不断完善起来的。
- 路由表是各种路由算法 + 人工配置逐步完善起来的。
- arp 缓存表是不断通过 arp 协议的请求逐步完善起来的。

知道了以上这些，目前网络上两个节点是如何发送数据包的这个过程，就完全可以解释通了！

参考的网络拓扑图

那接下来我们就放上参考的 **最后一个**网络拓扑图吧，请做好 **战斗** 准备！

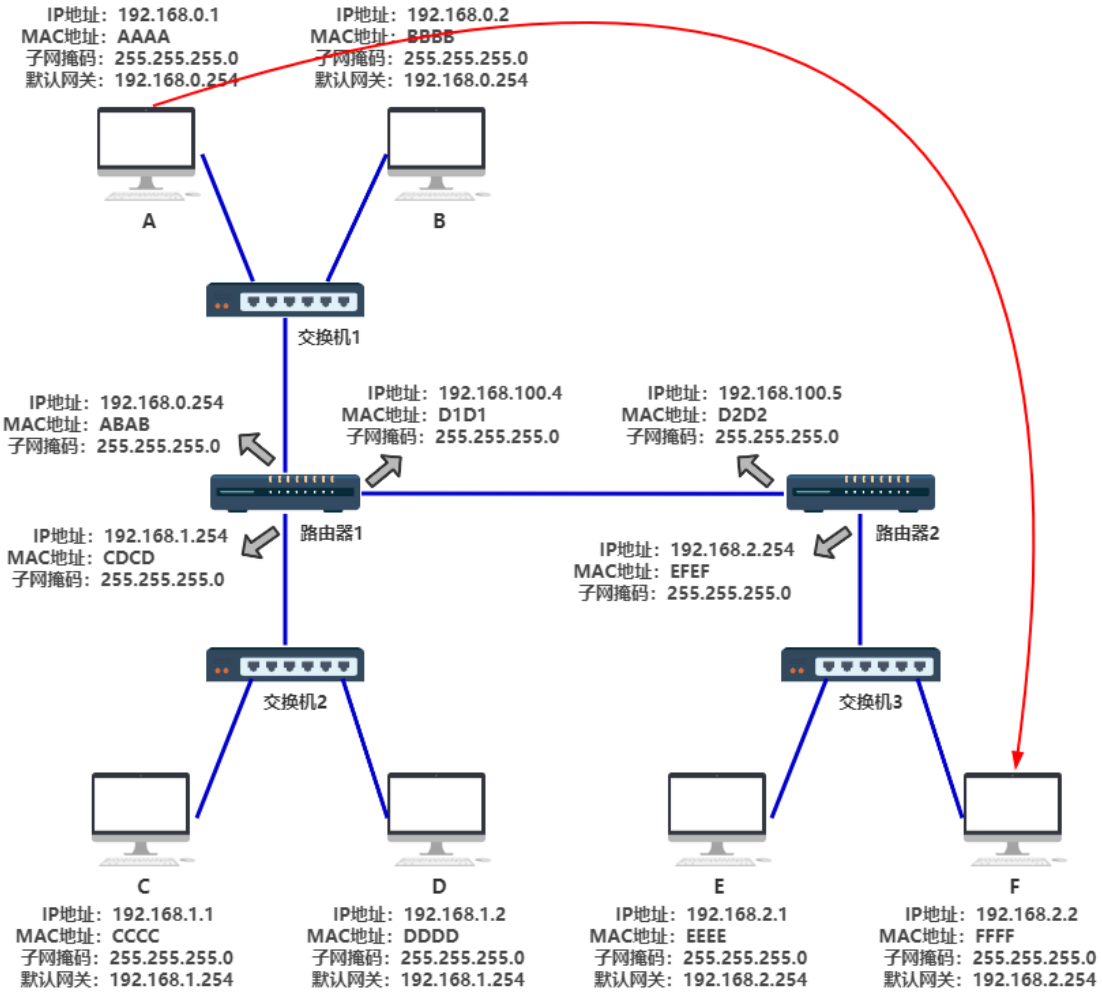


这时路由器 1 连接了路由器 2，所以其路由表有了下一条地址这一个概念，所以它的路由表就变成了这个样子。如果匹配到了有下一跳地址的一项，则需要再次匹配，找到其端口，并找到下一跳 IP 的 MAC 地址。

也就是说找来找去，最终必须能映射到一个端口号，然后从这个端口号把数据包发出去。

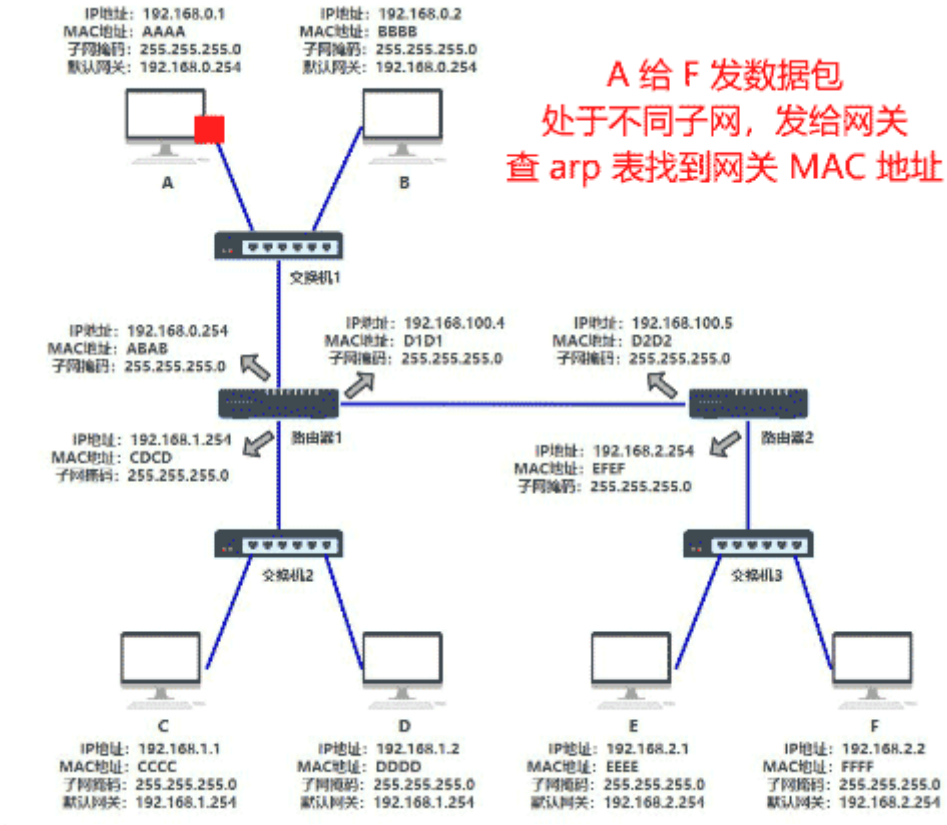
目的地址	下一跳	端口
192.168.0.0/24		0
192.168.0.254/32		0
192.168.1.0/24		1
192.168.1.254/32		1
192.168.2.0/24	192.168.100.5	
192.168.100.0/24		2
192.168.100.4/32		2

这时如果 A 给 F 发送一个数据包，能不能通呢？如果通的话整个过程是怎样的呢?



思考一分钟...

详细过程动画描述：



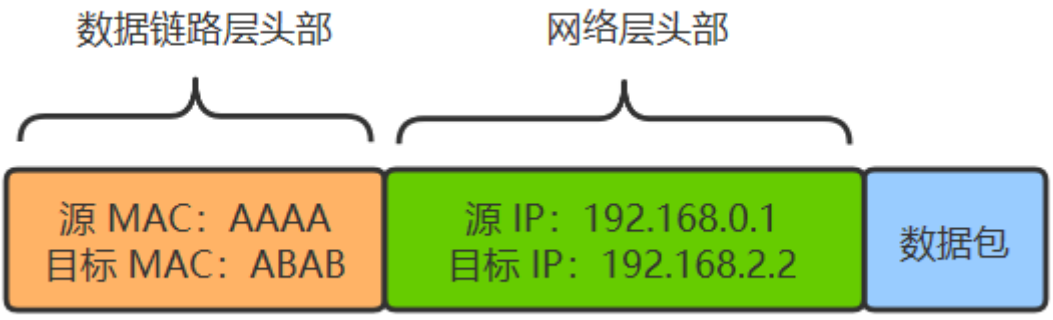
详细过程文字描述：

1. 首先 A（192.168.0.1）通过子网掩码（255.255.255.0）计算出自己与 F（192.168.2.2）并不在同一个子网内，于是决定发送给默认网关（192.168.0.254）

2. A 通过 ARP 找到 默认网关 192.168.0.254 的 MAC 地址。

3. A 将源 MAC 地址（AAAA）与网关 MAC 地址（ABAB）封装在数据链路层头部，又将源 IP 地址

（192.168.0.1）和目的 IP 地址（192.168.2.2）（注意这里千万不要以为填写的是默认网关的 IP 地址，从始至终这个数据包的两个 IP 地址都是不变的，只有 MAC 地址在不断变化）封装在网络层头部，然后发包



4. 交换机 1 收到数据包后，发现目标 MAC 地址是 ABAB，转发给路由器1

5. 数据包来到了路由器 1，发现其目标 IP 地址是 192.168.2.2，查看其路由表，发现了下一跳的地址是 192.168.100.5*

6. 所以此时路由器 1 需要做两件事，第一件是再次匹配路由表，发现匹配到了端口为 2，于是将其封装到数据链路层，最后把包从 2 号口发出去。

7. 此时路由器 2 收到了数据包，看到其目的地址是 192.168.2.2，查询其路由表，匹配到端口号为 1，准备从 1 号口把数据包送出去。

8. 但此时路由器 2 需要知道 192.168.2.2 的 MAC 地址了，于是查看其 arp 缓存，找到其 MAC 地址为 FFFF，将其封装在数据链路层头部，并从 1 号端口把包发出去。

9. 交换机 3 收到了数据包，发现目的 MAC 地址为 FFFF，查询其 MAC 地址表，发现应该从其 6 号端口出去，于是从 6 号端口把数据包发出去。

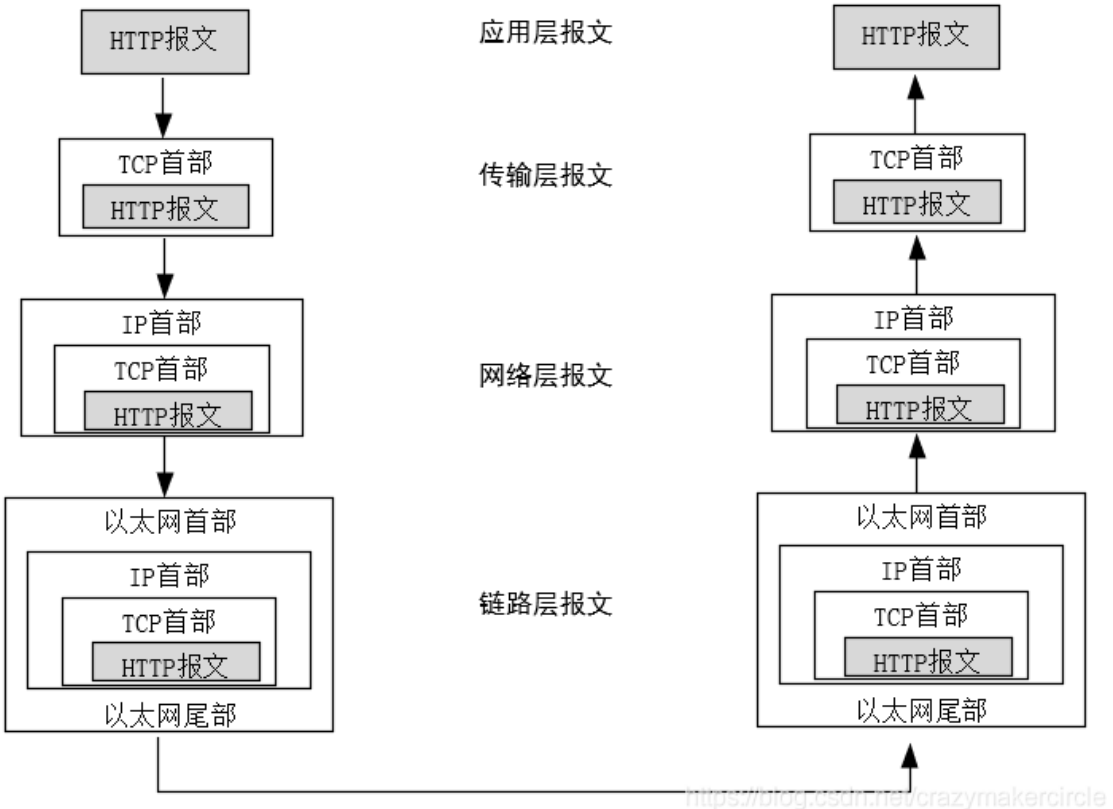
10.F 最终收到了数据包！**并且发现目的 MAC 地址就是自己，于是收下了这个包

HTTP报文传输原理

利用TCP/IP进行网络通信时，数据包会按照分层顺序与对方进行通信。发送端从应用层往下走，接收端从链路层往上走。从客户端到服务器的数据，每一帧数据的传输的顺序都为：应用层->运输层->网络层->链路层->链路层->网络层->运输层->应用层。

HTTP报文传输过程

以一个HTTP请求的传输为例，请求从HTTP客户端（如浏览器）和HTTP服务端应用的传输过程，大致如下图所示：

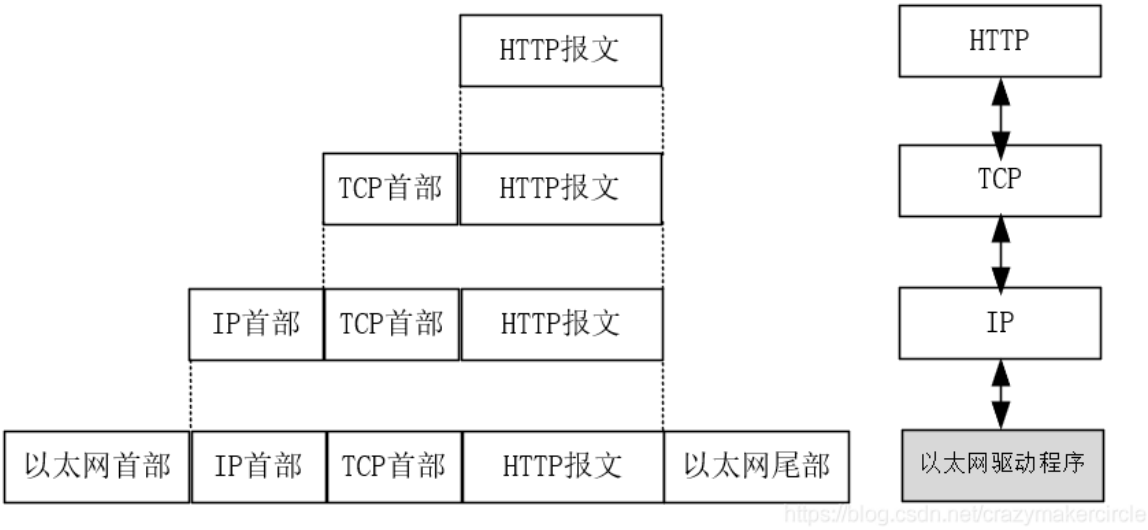


图：HTTP请求报文的分层传输过程

数据封装和分用

接下来，为大家介绍一下数据封装和分用。

数据通过互联网传输的时候不可能是光秃秃的不加标识，如果这样数据就会乱。所以数据在发送的时候，需要加上特定标识，加上特定标识的过程叫做数据的封装，在数据使用的时候再去掉特定标识，去掉特定标识的过程就叫做分用。TCP/IP协议的数据封装和分用过程，大致如下图所示：



图：TCP/IP协议的数据封装和分用过程

在数据封装时，数据经过每个层都会打上该层特定标识，添加上头部。

在传输层封装时，添加的报文首部时要存入一个应用程序的标识符，无论TCP和UDP都用一个16位的端口号来表示不同的应用程序，并且都会将源端口和目的端口存入报文首部中。

在网络层封装时，IP首部会标识处理数据的协议类型，或者说标识出网络层数据帧所携带的上层数据类型，如TCP、UDP、ICMP、IP、IGMP等等。
具体来说，会在IP首部中存入一个长度为8位的数值，称作协议域：
1表示为ICMP协议、2表示为IGMP协议、6表示为TCP协议、17表示为UDP协议、等等。IP首部还会标识发送方地址（源IP）和接收方地址（目标IP）。

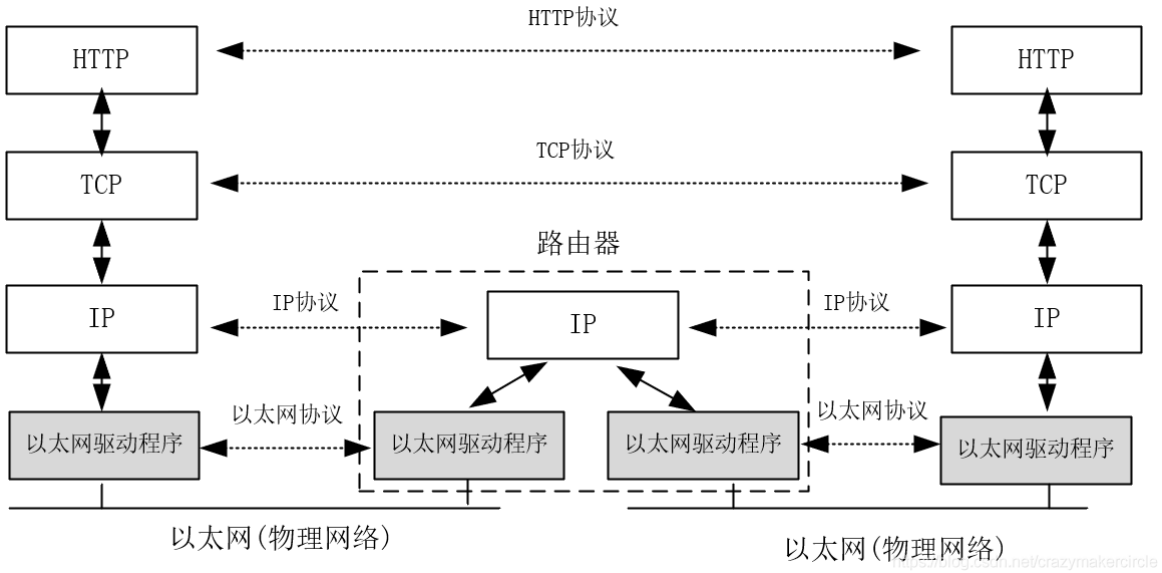
在链路层封装时，网络接口分别要发送和接收IP、ARP和RARP等多种不同协议的报文，因此也必须在以太网的帧首部中加入某种形式的标识，以指明所处理的协议类型，为此，以太网的报文帧的首部也有一个16位的类型域，标识出以太网数据帧所携带的上层数据类型，如IPv4、ARP、IPV6、PPPoE等等。

数据封装和分用的过程大致为：发送端每通过一层会增加该层的首部，接收端每通过一层则删除该层的首部。

总体来说，TCP/IP分层管理、数据封装和分用的好处：分层之后若需改变相关设计，只需替换变动的层。各层之间的接口部分规划好之后，每个层次内部的设计就可以自由改动。层次化之后，设计也变得相对简单：各个层只需考虑分派给自己的传输任务。

TCP/IP与OSI的区别主要有哪些呢？除了TCP/IP与OSI在分层模块上稍有区别，更重要的区别为：OSI参考模型注重“通信协议必要的功能是什么”，而TCP/IP则更强调“在计算机上实现协议应该开发哪种程序”。

实际上，在传输过程中，数据报文会在不同的物理网络之间传递，还是以一个HTTP请求的传输为例，请求在不同物理网络之间的传输过程，大致如下图所示：



图：HTTP请求在不同物理网络之间的传输过程

数据包在不同物理网络之间的传输过程中，网络层会通过路由器去对不同的网络之间的数据包进行存储、分组转发处理。构造互连网最简单的方法是把两个或多个网络通过路由器进行连接。路由器可以简单理解为一种特殊的用于网络互连的硬件盒，其作用是为不同类型的物理网络提供连接：以太网、令牌环网、点对点的链接和FDDI（光纤分布式数据接口）等等。

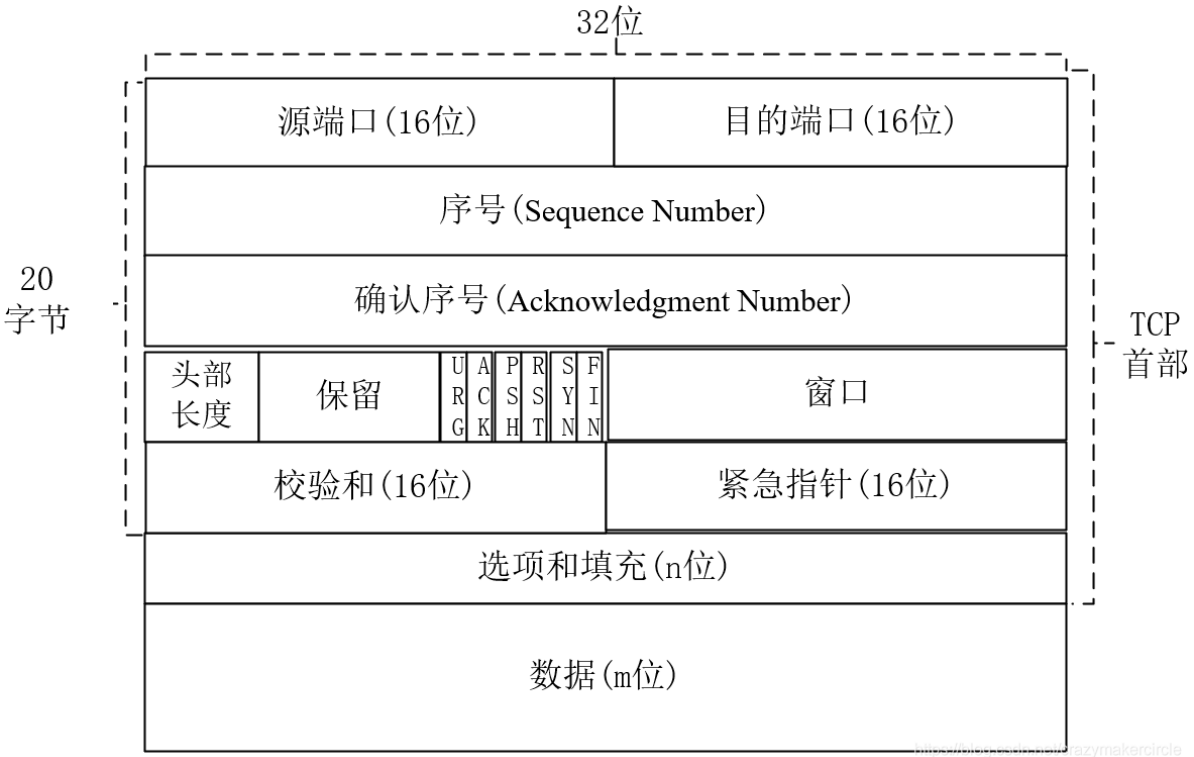
物理网络之间通过路由器进行互连，随着增加不同类型的物理网络，可能会有很多个路由器，但是对于应用层来说仍然是一样的，TCP协议栈为大家屏蔽了物理层的复杂性。总之，物理细节和差异性的隐藏，使得互联网TCP/IP传输的功能变得非常强大。

接下来，开始为大家介绍与传输性能有密切关系的内容：TCP传输层的三次握手建立连接，四次挥手释放连接。不过在此之前，还得先介绍一下TCP报文协议。

TCP协议的报文格式

在TCP/IP协议栈中，IP协议层只关心如何使数据能够跨越本地网络边界的问题，而不关心数据如何传输。整体TCP/IP协议栈，共同配合一起解决数据如何通过许许多多个点对点通路，顺利传输到达目的地。一个点对点通路被称为一“跳”（hop），通过TCP/IP协议栈，网络成员能够在许多“跳”的基础上建立相互的数据通路。

传输层TCP协议提供了一种面向连接的、可靠的字节流服务，其数据帧格式，大致如下图所示：



图：传输层TCP协议的数据帧格式

一个传输层TCP协议的数据帧，大致包含以下字段：

（一）源端口号

源端口号表示报文的发送端口，占16位。源端口和源IP地址组合起来，可以标识报文的发送地址。

（二）目的端口号

目的端口号表示报文的接收端口，占16位。目的端口和目的IP地址相结合，可以标识报文的接收地址。

TCP协议是基于IP协议的基础上传输的，TCP报文中的源端口号+源IP，与TCP报文中的目的端口号+目的IP一起，组合起来唯一性的确定一条TCP连接。

（三）序号（Sequence Number）

TCP传输过程中，在发送端出的字节流中，传输报文中的数据部分的每一个字节都有它的编号。序号（Sequence
Number）占32位，发起方发送数据时，都需要标记序号。

序号（Sequence Number）的语义与SYN控制标志（Control
Bits）的值有关。根据控制标志（Control Bits）中的SYN是否为1，序号（Sequence
Number）表达不同的含义：

（1）当SYN = 1时，当前为连接建立阶段，此时的序号为初始序号ISN((Initial Sequence
Number)，通过算法来随机生成序号；

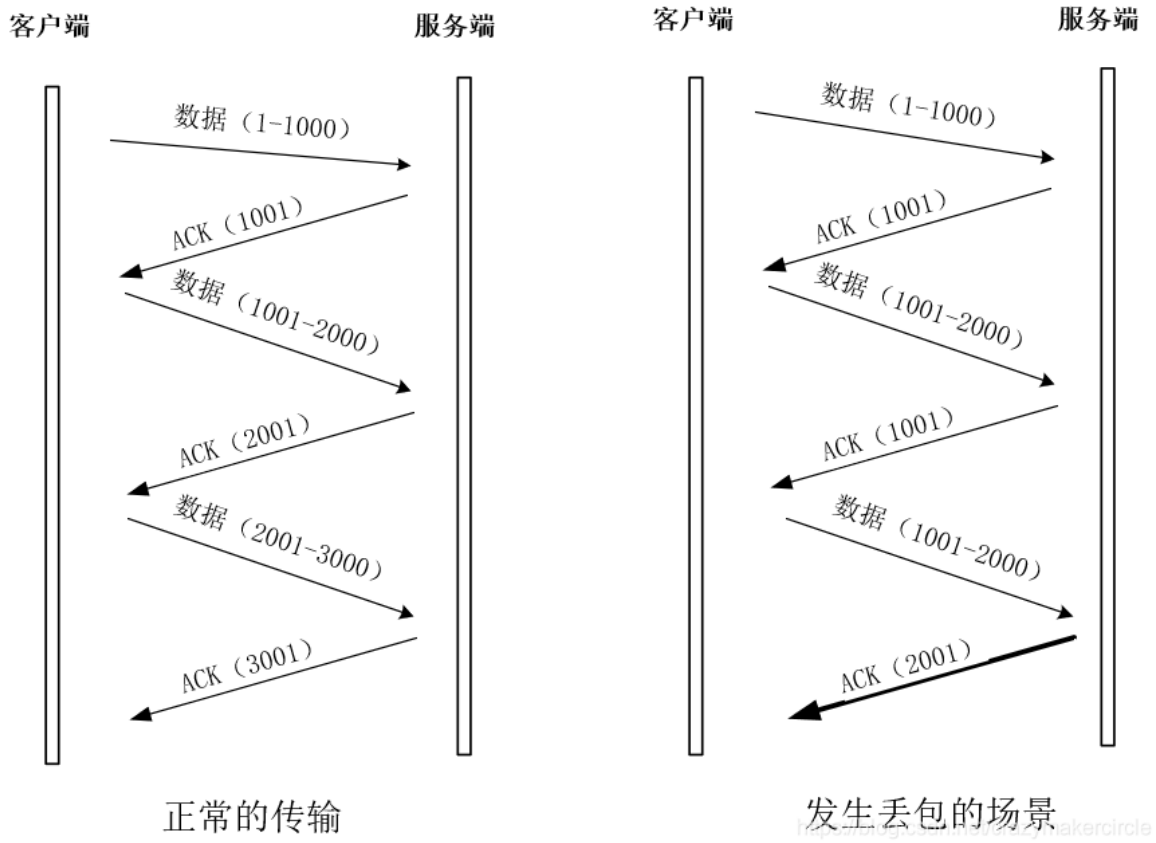
（2）当SYN = 0时在数据传输正式开始时，第一个报文的序号为 ISN +
1，后面的报文的序号，为前一个报文的SN值+TCP报文的净荷字节数(不包含TCP头)。比如，如果发送端发送的一个TCP帧的净荷为12byte，序号为5，则发送端接着发送的下一个数据包的时候，序号的值应该设置为5+12=17。

在数据传输过程中，TCP协议通过序号（Sequence Number）对上层提供有序的数据流。发送端可以用序号来跟踪发送的数据量；接收端可以用序号识别出重复接收到的TCP包，从而丢弃重复包；对于乱序的数据包，接收端也可以依靠序号对其进行排序。

（四）确认序号（Acknowledgment Number）

确认序号（Acknowledgment Number）标识了报文接收端期望接收的字节序列。如果设置了ACK控制位，确认序号的值表示一个准备接收的包的序列码，注意，它所指向的是准备接收的包，也就是下一个期望接收的包的序列码。

举个例子，假设发送端（如Client）发送3个净荷为1000byte、起始SN序号为1的数据包给Server服务端，Server每收到一个包之后，需要回复一个ACK响应确认数据包给Client。ACK响应数据包的ACK Number值，为每个Client包的为SN+包净荷，既表示Server已经确认收到的字节数，还表示期望接收到的下一个Client发送包的SN序号，具体的ACK值如下图左边的正常传输部分所示。



图：传输过程的确认序号（Acknowledgment Number）值示例图

在上图的左边部分，Server第1个ACK包的ACK Number值为1001，是通过Client第1个包的SN+包净荷=1+1000计算得到，表示期望第2个Client包的SN序号为1001；Server第2个ACK包的ACK Number值为2001，为Client第2个包的SN+包净荷=2001，表示期望第3个Server包的SN为2001，以此类推。

如果发生错误，假设Server在处理Client的第二个发送包异常，Server仍然回复一个ACK Number值为1001的确认包，则Client的第二个数据包需要重复发送，具体的ACK值如上图右边的正常传输部分所示。

只有控制标志的ACK标志为1时，数据帧中的确认序号ACK Number才有效。TCP协议规定，连接建立后，所有发送的报文的ACK必须为1，也就是建立连接后，所有报文的确认序号有效。如果是SYN类型的报文，其ACK标志为0，故没有确认序号。

（五）头部长度

该字段占用4位，用来表示TCP报文首部的长度，单位是4bit位。其值所表示的并不是字节数，而是头部的所含有的32bit的数目（或者倍数），或者4个字节的倍数，所以TCP头部最多可以有60字节（4*15=60）。没有任何选项字段的TCP头部长度为20字节，所以其头部长度为5，可以通过20/4=5计算得到。

（六）预留6位

头部长度后面预留的字段长度为6位，作为保留字段，暂时没有什么用处。

（七）控制标志

控制标志（Control
Bits）共6个bit位，具体的标志位为：URG、ACK、PSH、RST、SYN、FIN。6个标志位的说明，如下表所示。

表：TCP报文控制标志（Control Bits）说明

标志位	说明
URG	占1位，表示紧急指针字段有效。URG位指示报文段里的上层实体（数据）标记为“紧急”数据。当URG=1时，其后的紧急指针指示紧急数据在当前数据段中的位置(相对于当前序列号的字节偏移量)，TCP接收方必须通知上层实体。
ACK	占1位，置位ACK=1表示确认号字段有效；TCP协议规定，接建立后所有发送的报文的ACK必须为1；当ACK=0时，表示该数据段不包含确认信息。当ACK=1时，表示该报文段包括一个对已被成功接收报文段的确认序号Acknowledgment Number，该序号同时也是下一个报文的预期序号。
PSH	占1位，表示当前报文需要请求推（push）操作；当PSH=1时，接收方在收到数据后立即将数据交给上层，而不是直到整个缓冲区满。
RST	占1位，置位RST=1表示复位TCP连接；用于重置一个已经混乱的连接，也可用于拒绝一个无效的数据段或者拒绝一个连接请求。如果数据段被设置了RST位，说明报文发送方有问题发生。
SYN	占1位，在连接建立时用来同步序号。当SYN=1而ACK=0时，表明这是一个连接请求报文。对方若同意建立连接，则应在响应报文中使SYN=1和ACK=1。 综合一下，SYN置1就表示这是一个连接请求或连接接受报文。
FIN	占1位，用于在释放TCP连接时，标识发送方比特流结束，用来释放一个连接。当 FIN = 1 时，表明此报文的发送方的数据已经发送完毕，并要求释放连接。

在连接建立的三次握手过程中，若只是单个SYN置位，表示的只是建立连接请求。如果SYN和ACK同时置位为1，表示的建立连接之后的响应。

（八）窗口大小：

长度为16位，共2个字节。此字段用来进行流量控制。流量控制的单位为字节数，这个值是本端期望一次接收的字节数。

（九）校验和：

长度为16位，共2个字节。对整个TCP报文段，即TCP头部和TCP数据进行校验和计算，接收端用于对收到的数据包进行验证。

（十）紧急指针：

长度为16位，2个字节。它是一个偏移量，和SN序号值相加表示紧急数据最后一个字节的序号。

以上十项内容是TCP报文首部必须的字段，也称固有字段，长度为20个字节。接下来是TCP报文的可选项和填充部分。

（十一）可选项和填充部分

可选项和填充部分的长度为4n字节（n是整数），该部分是根据需要而增加的选项。如果不足4n字节，要加填充位，使得选项长度为32位（4字节）的整数倍，具体的做法是在这个字段中加入额外的零，以确保TCP头是32位（4字节）的整数倍。

最常见的选项字段是MSS（Maximum Segment
Size最长报文大小），每个连接方通常都在通信的第一个报文段（SYN标志为1的那个段）中指明这个选项字段，表示当前连接方所能接受的最大报文段的长度。

由于可选项和填充部分不是必须的，所以TCP报文首部最小长度为20个字节。

至此，TCP报文首部的字段，就全部介绍完了。TCP报文首部的后面，接着的是数据部分，不过数据部分是可选的。在一个连接建立和一个连接终止时，双方交换的报文段仅有TCP首部。如果一方没有数据要发送，也使用没有任何数据的首部来确认收到的数据，比如在处理超时的过程中，也会发送不带任何数据的报文段。

总体来说，TCP协议的可靠性，主要通过以下几点来保障：

（1）应用数据分割成TCP认为最适合发送的数据块。这部分是通过MSS（最大数据包长度）选项来控制的，通常这种机制也被称为一种协商机制，MSS规定了TCP传往另一端的最大数据块的长度。值得注意的是，MSS只能出现在SYN报文段中，若一方不接收来自另一方的MSS值，则MSS就定为536字节。一般来讲，MSS值还是越大越好，这样可以提高网络的利用率。

（2）重传机制。设置定时器，等待确认包，如果定时器超时还没有收到确认包，则报文重传。

（3）对首部和数据进行校验。

（4）接收端对收到的数据进行排序，然后交给应用层。

（5）接收端丢弃重复的数据。

（6）TCP还提供流量控制，主要是通过滑动窗口来实现流量控制。

至此TCP协议的数据帧格式介绍完了。接下来开始为大家重点介绍：TCP传输层的三次握手建立连接，四次挥手释放连接。

TCP的三次握手

TCP连接的建立时，双方需要经过三次握手，而断开连接时，双方需要经过四次分手，那么，其三次握手和四次分手分别做了什么呢？又是如何进行的呢？

通常情况下，建立连接的双方，由一端打开一个监听套接字（ServerSocket）来监听来自请求方的TCP（Socket）连接，当服务器端监听开始时，必须做好准备接受外来的连接，在Java中该操作通过创建一个ServerSocket服务监听套接字实例来完成，此操作会调用底层操作系统（如Linux）的C代码中三个函数socket()、bind()、listen()
来完成。开始监听之后，服务器端就做好接受外来连接的准备，如果监听到建立新连接的请求，会开启

一个传输套接字，称之为被动打开（Passive
Open）。

一段简单的服务端监听新连接请求，并且被动打开（Passive
Open）传输套接字的Java示例代码，具体如下：

```
public class SocketServer {

public static void main(String[] args) {

try {

// 创建服务端socket

ServerSocket serverSocket = new ServerSocket(8080);

//循环监听等待客户端的连接

while(true){

//监听到客户端连接，传输套接字被动开启

Socket socket = serverSocket.accept();

//开启线程进行连接的IO处理

ServerThread thread = new ServerThread(socket);

thread.start();

......

}

} catch (Exception e) {

// 处理异常

e.printStackTrace();

}

}

}
```

客户端在发起连接建立时，Java代码通过创建Socket实例，调用底层的connect(...)方法，主动打开
(Active
Open)Socket连接。套接字监听方在收到请求之后，监听方和发起方（客户端）之间就会建立一条的连接通道，该通道由双方IP和双方端口所唯一确定。

一段简单的客户端连接主动打开(Active Open)的Java示例代码，具体如下：

```
public class SocketClient {

public static void main(String[] args) throws InterruptedException {

```

```
try {

// 和服务器创建连接

Socket socket = new Socket("localhost",8080);

// 写入给监听方的输出流

OutputStream os = socket.getOutputStream();

…..

// 读取监听方的输入流

InputStream is = socket.getInputStream();

…..

} catch (Exception e) {

e.printStackTrace();

}

}

}
```

三次握手过程

TCP连接的建立时，双方需要经过三次握手，具体过程如下：

（1）第一次握手：Client进入SYN_SENT状态，发送一个SYN帧来主动打开传输通道，该帧的SYN标志位被设置为1，同时会带上Client分配好的SN序列号，该SN是根据时间产生的一个随机值，通常情况下每间隔4ms会加1。除此之外，SYN帧还会带一个MSS（最大报文段长度）可选项的值，表示客户端发送出去的最大数据块的长度。

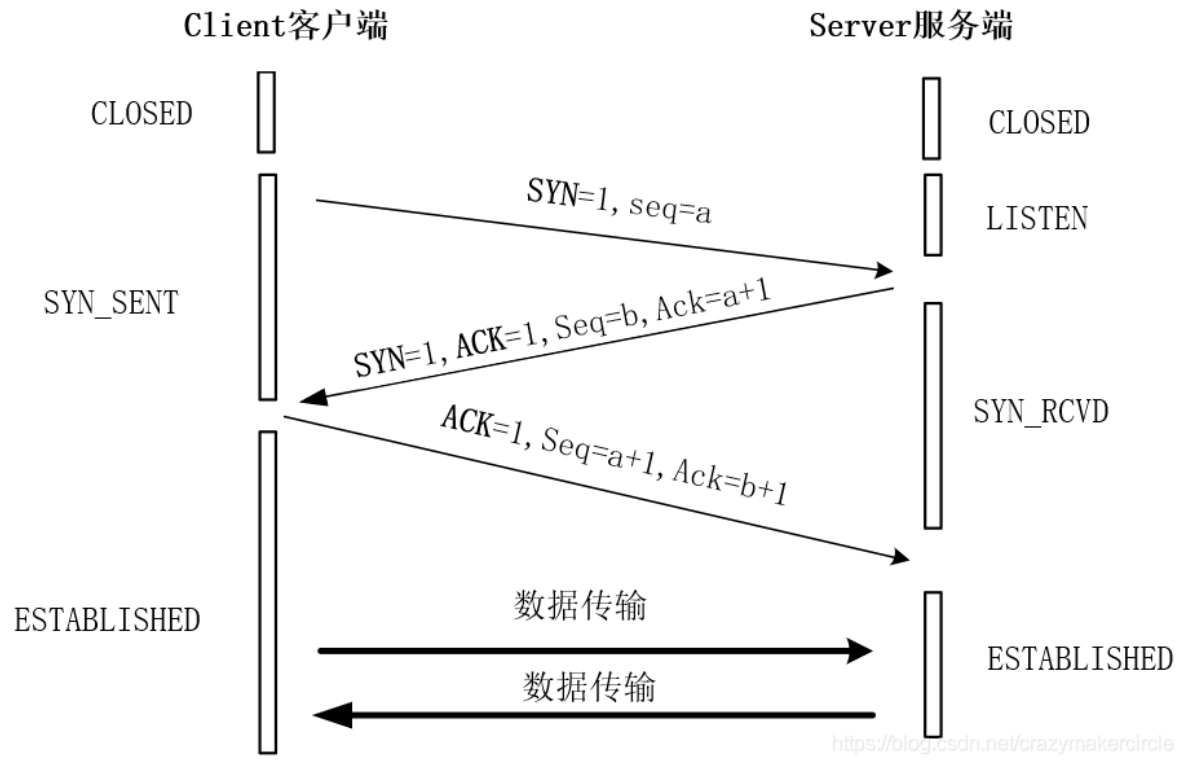
（2）第二次握手：Server端在收到SYN帧之后，会进入SYN_RCVD状态，同时返回SYN+ACK帧给Client，主要目的在于通知Client，Server端已经收到SYN消息，现在需要进行确认。Server端发出的SYN+ACK帧的ACK标志位被设置为1，其确认序号AN（Acknowledgment Number）值被设置为Client的SN+1；SYN+ACK帧的SYN标志位被设置为1，SN值为Server端生成的SN序号；SYN+ACK帧的MSS（最大报文段长度）表示的是Server端的最大数据块长度。

（3）第三次握手：Client在收到Server的第二次握手SYN+ACK确认帧之后，首先将自己的状态会从SYN_SENT变成ESTABLISHED，表示自己方向的连接通道已经建立成功，Client可以发送数据给Server端了。然后，Client发ACK帧给Server端，该ACK帧的ACK标志位被设置为1，其确认序号AN（Acknowledgment Number）值被设置为Server端的SN序列号+1。还有一种情况，Client可能会将ACK帧和第一帧要发送的数据，合并到一起发送给Server端。

（4）Server端在收到Client的ACK帧之后，会从SYN_RCVD状态会进入ESTABLISHED状态，至此，Server方向的通道连接建立成功，Server可以发送数据给Client，TCP的全双工连接建立完成。

三次握手的图解

三次握手的交互过程，具体如下图所示：



图：TCP建立的连接时三次握手示意图

Client和Server完成了三次握手后，双方就进入了数据传输的阶段。数据传输完成后，连接将断开，连接断开的过程需要经历四次挥手。

TCP的四次挥手

业务数据通信完成之后，TCP连接开始断开（或者拆接）的过程，在这个过程中连接的每个端的都能独立地、主动的发起，断开的过程TCP协议使用了四路挥手操作。

四次挥手具体过程

四次挥手具体过程，具体如下：

（1）第一次挥手：主动断开方（可以是客户端，也可以是服务器端），向对方发送一个FIN结束请求报文，此报文的FIN位被设置为1，并且正确设置Sequence
Number（序列号）和Acknowledgment
Number（确认号）。发送完成后，主动断开方进入FIN_WAIT_1状态，这表示主动断开方没有业务数据要发送给对方，准备关闭SOCKET连接了。

（2）第二次挥手：正常情况下，在收到了主动断开方发送的FIN断开请求报文后，被动断开方会发送一个ACK响应报文，报文的Acknowledgment
Number（确认号）值为断开请求报文的Sequence Number
（序列号）加1，该ACK确认报文的含义是：“我同意你的连接断开请求”。之后，被动断开方就进入了CLOSE-WAIT（关闭等待）状态，TCP协议服务会通知高层的应用进程，对方向本地方向的连接已经关闭，对方已经没有数据要发送了，若本地还要发送数据给对方，对方依然会接受。被动断开方的CLOSE-WAIT（关闭等待）还要持续一段时间，也就是整个CLOSE-WAIT状态持续的时间。

主动断开方在收到了ACK报文后，由FIN_WAIT_1转换成FIN_WAIT_2状态。

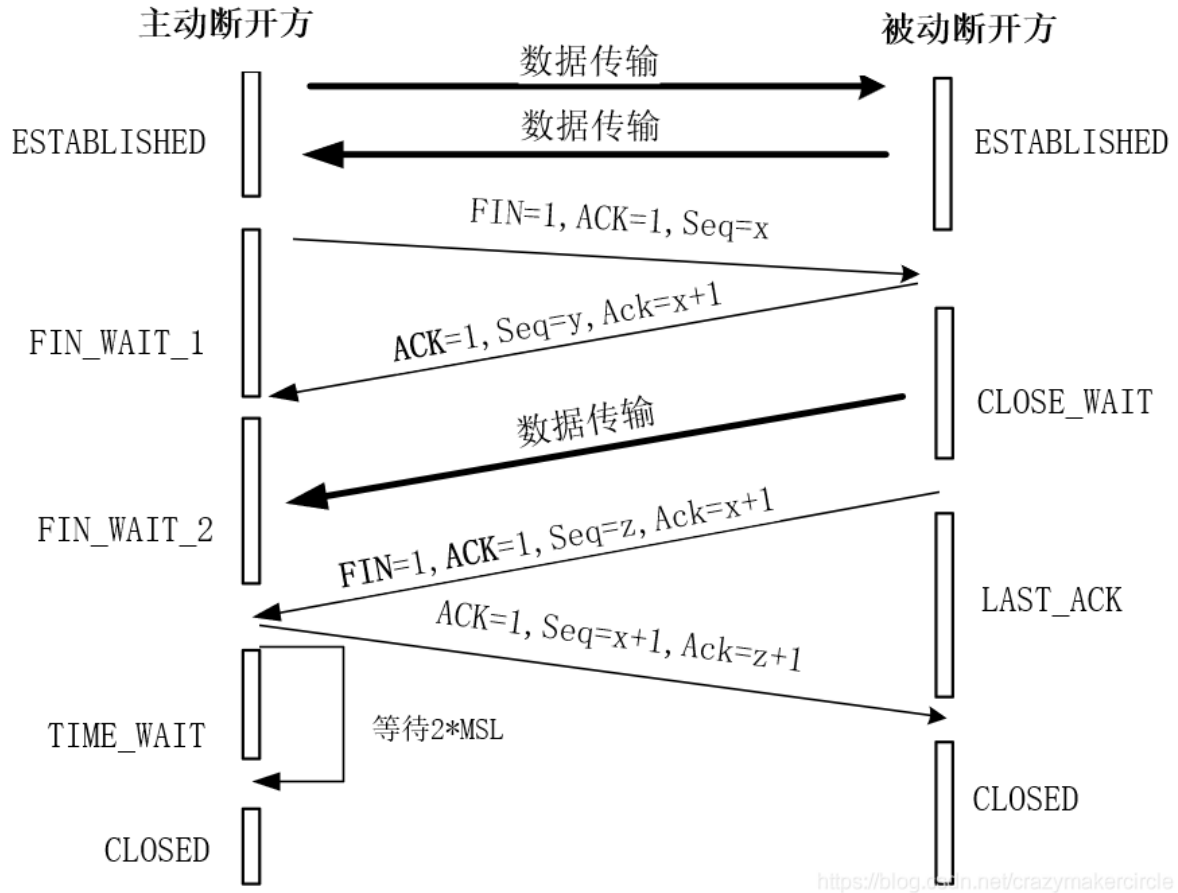
（3）第三次挥手：在发送完成ACK报文后，被动断开方还可以继续完成业务数据的发送，待剩余数据发送完成后，或者CLOSE-WAIT（关闭等待）截止后，被动断开方会向主动断开方发送一个FIN+ACK结束响应报文，表示被动断开方的数据都发送完了，然后，被动断开方进入LAST_ACK状态。

（4）第四次挥手：主动断开方收在到FIN+ACK断开响应报文后，还需要进行最后的确认，向被动断开方发送一个ACK确认报文，然后，自己就进入TIME_WAIT状态，等待超时后最终关闭连接。处于TIME_WAIT状态的主动断开方，在等待完成2MSL的时间后，如果期间没有收到其他报文，则证明对方已正常关闭，主动断开方的连接最终关闭。

被动断开方在收到主动断开方的最后的ACK报文以后，最终关闭了连接，自己啥也不管了。

四次挥手图解

四次挥手的全部交互过程，具体如下图所示：



图：TCP建立的连接时四次挥手的示意图

处于TIME_WAIT状态的主动断开方，在等待完成2MSL的时间后，才真正关闭连接通道，其等待的时间为什么是2MSL呢?

2MSL翻译过来就是两倍的MSL。MSL全称为Maximum Segment Lifetime，指的是一个TCP报文片段在网络中最大的存活时间，具体来说，2MSL对应于一次消息的来回（一个发送和一个回复）所需的最大时间。如果直到2MSL，主动断开方都没有再一次收到对方的报文（如FIN报文），则可以推断ACK已经被对方成功接收，此时，主动断开方将最终结束自己的TCP连接。所以，TCP的TIME_WAIT状态也称为2MSL等待状态。

有关MSL的具体的时间长度，在RFC1122协议中推荐为2分钟。在SICS（瑞典计算机科学院）开发的一个小型开源的TCP/IP协议栈——LwIP开源协议栈中MSL默认为1分钟。在源自Berkeley的TCP协议栈实现中MSL默认长度为30秒。总体来说，TIME_WAIT（2MSL）等待状态的时间长度，一般维持在1-4分钟之间。

通过三次握手建立连接和四次挥手拆除连接，一次TCP的连接建立及拆除，至少进行7次通信，可见其成本是很高的。

三次握手、四次挥手的常见面试题

有关TCP的连接建立的三次握手及拆除过程的四次挥手的面试问题，是技术面试过程中的出现频率很高的重点和难点问题，常见问题大致如下：

问题（1）：为什么关闭连接的需要四次挥手，而建立连接却只要三次握手呢?

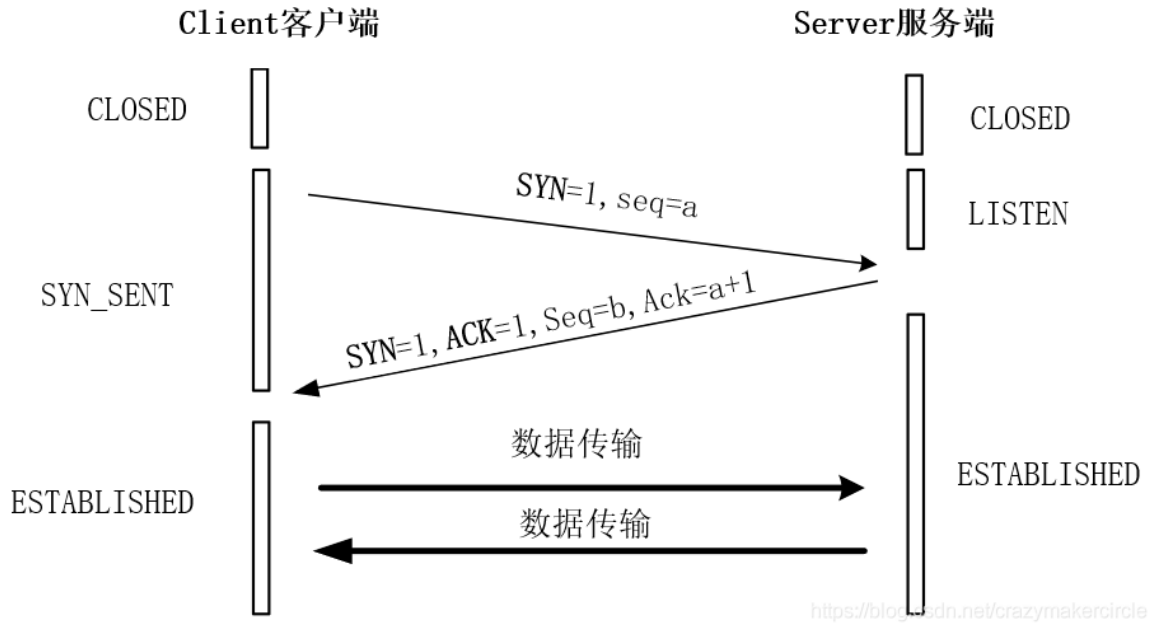
关闭连接时，被动断开方在收到对方的FIN结束请求报文时，很可能业务数据没有发送完成，并不能立即关闭连接，被动方只能先回复一个ACK响应报文，告诉主动断开方："你发的FIN报文我收到了，只有等到我所有的业务报文都发送完了，我才能真正的结束，在结束之前，我会发你FIN+ACK报文的，你先等着"。所以，被动断开方的确认报文，需要拆开成为两步，故总体就需要四步挥手。

而在建立连接场景中，Server端的应答可以稍微简单一些。当Server端收到Client端的SYN连接请求报文后，其中ACK报文表示对请求报文的应答，SYN报文用来表示服务端的连接也已经同步开启了，而ACK报文和SYN报文之间，不会有其他报文需要发送，故而可以合二为一，可以直接发送一个SYN+ACK报文。所以，在建立连接时，只需要三次握手即可。

问题（2）：为什么连接建立的时候是三次握手，可以改成两次握手吗?

三次握手完成两个重要的功能：一是双方都做好发送数据的准备工作，而且双方都知道对方已准备好；二是双方完成初始SN序列号的协商，双方的SN序列号在握手过程中被发送和确认。

如果把三次握手改成两次握手，可能发生死锁。两次握手的话，缺失了Client的二次确认ACK帧，假想的TCP建立的连接时二次挥手，可以如下图所示：



图：假想的TCP建立的连接时二次握手的示意图

在假想的TCP建立的连接时二次握手过程中，Client发送Server发送一个SYN请求帧，Server收到后发送了确认应答SYN+ACK帧。按照两次握手的协定，Server认为连接已经成功地建立了，可以开始发送数据帧。这个过程中，如果确认应答SYN+ACK帧在传输中被丢失，Client没有收到，Client将不知道Server是否已准备好，也不知道Server的SN序列号，Client认为连接还未建立成功，将忽略Server发来的任何

数据分组，会一直等待Server的SYN+ACK确认应答帧。而Server在发出的数据帧后，一直没有收到对应的ACK确认后就会产生超时，重复发送同样的数据帧。这样就形成了死锁。

问题（3）：为什么主动断开方在TIME-WAIT状态必须等待2MSL的时间?

原因之一：主动断开方等待2MSL的时间，是为了确保两端都能最终关闭。假设网络是不可靠的，被动断开方发送FIN+ACK报文后，其主动方的ACK响应报文有可能丢失，这时候的被动断开方处于LAST-ACK状态的，由于收不到ACK确认被动方一直不能正常的进入CLOSED状态。在这种场景下，被动断开方会超时重传FIN+ACK断开响应报文，如果主动断开方在2MSL时间内，收到这个重传的FIN+ACK报文，会重传一次ACK报文，后再一次重新启动2MSL计时等待，这样，就能确保被动断开方能收到ACK报文，从而能确保被动方顺利进入到CLOSED状态。只有这样，双方都能够确保关闭。反过来说，如果主动断开方在发送完ACK响应报文后，不是进入TIME_WAIT状态去等待2MSL时间，而是立即释放连接，则将无法收到被动方重传的FIN+ACK报文，所以不会再发送一次ACK确认报文，此时处于LAST-ACK状态的被动断开方，无法正常进入到CLOSED状态。

原因之二：防止“旧连接的已失效的数据报文”出现在新连接中。主动断开方在发送完最后一个ACK报文后，再经过2MSL，才能最终关闭和释放端口，这就意味着，相同端口的新TCP新连接，需要在2MSL的时间之后，才能够正常的建立。2MSL这段时间内，旧连接所产生的所有数据报文，都已经从网络中消失了，从而，确保了下一个新的连接中不会出现这种旧连接请求报文。

问题（4）：如果已经建立了连接，但是Client端突然出现故障了怎么办?

TCP还设有一个保活计时器，Client端如果出现故障，Server端不能一直等下去，这样会浪费系统资源。每收到一次Client客户端的数据帧后，Server端都的保活计时器会复位。计时器的超时时间通常是设置为2小时，若2小时还没有收到Client端的任何数据帧，Server端就会发送一个探测报文段，以后每隔75秒钟发送一次。若一连发送10个探测报文仍然没反应，Server端就认为Client端出了故障，接着就关闭连接。如果觉得保活计时器的两个多小时的间隔太长，可以自行调整TCP连接的保活参数。

免费领取11个技术圣经PDF

技术自由圈 7个 学习圣经 PDF

1. 《SpringCloud Alibaba 学习圣经》 377页 PDF
   v1 版，发布日期：2023年03月15日
2. 《Docker 学习圣经》 145页 PDF
   v2 版，发布日期：2023年03月12日
3. 《Kubernets 学习圣经》 530页 PDF
   v1 版，发布日期：2023年04月11日
4. 《响应式圣经》 114页 PDF
   v2 版，发布日期：2023年02月14日
5. 《缓存之王 Caffeine 红宝书》 175页 PDF
   v2 版，发布日期：2022年11月30日
6. 《队列之王 Disruptor 红宝书》 75页 PDF
   v1 版，发布日期：2022年10月05日
7. 《Prometheus+grafna 红宝书》 100页 PDF
   v1 版，发布日期：2022年10月08日

技术自由圈 3个 高并发圣经 PDF

8. 《Java高并发核心编程 卷1 加强版：NIO、Netty、Redis、ZooKeeper》 609页 PDF
   加强版，发布日期：2023年01月01日
9. 《Java高并发核心编程 卷2 加强版：多线程、锁、JMM、JUC、高并发设计模式》 448页 PDF
   加强版，发布日期：2022年11月14日
10. 《Java高并发核心编程 卷3 加强版：亿级用户Web应用架构与实战》 481页 PDF
    加强版，发布日期：2022年11月14日

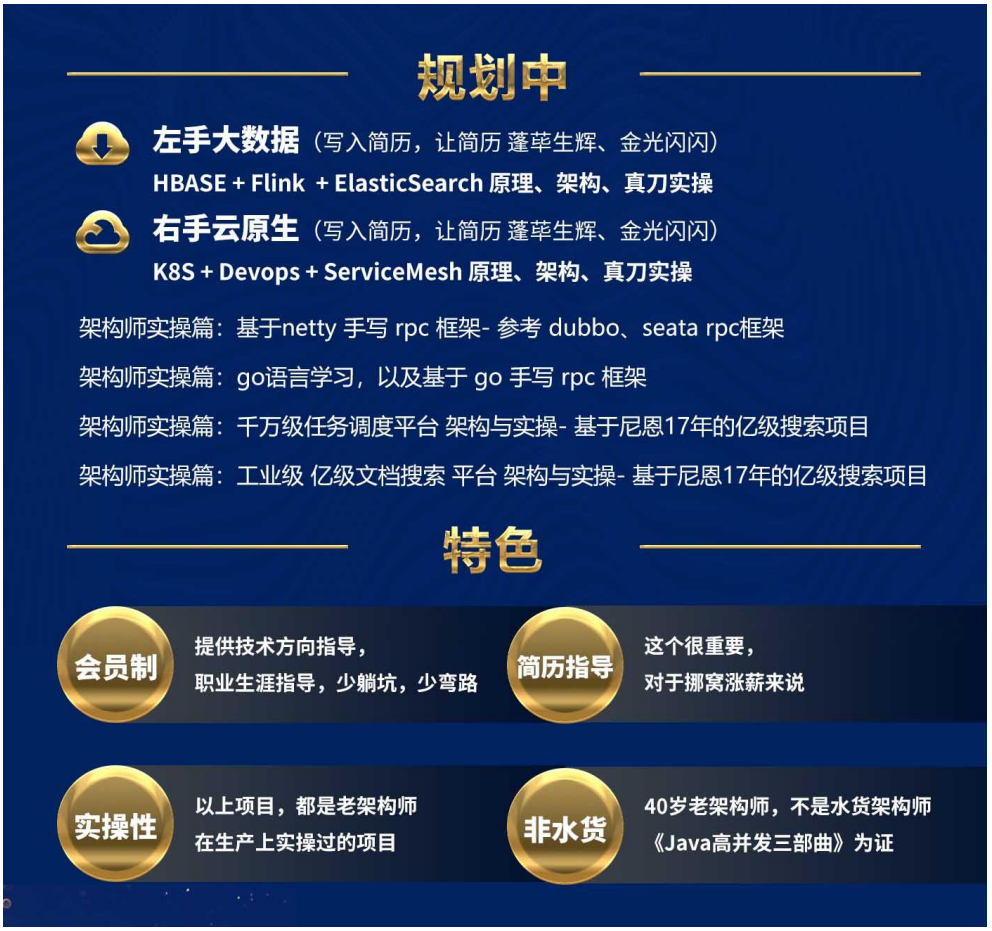
技术自由圈 面试圣经 40个 面试题 PDF

11. 《尼恩 Java 面试宝典》40个专题 4000页 PDF
    v60 版，发布日期：2023年03月11日

领取方式： 技术自由圈 公众号

硬核推荐：尼恩Java硬核架构班

详情：https://www.cnblogs.com/crazymakercircle/p/9904544.html


规划中
左手大数据（写入简历，让简历 蓬荜生辉、金光闪闪）
HBASE + Flink + ElasticSearch 原理、架构、真刀实操
右手云原生（写入简历，让简历 蓬荜生辉、金光闪闪）
K8S + Devops + ServiceMesh 原理、架构、真刀实操
架构师实操篇：基于netty 手写 rpc 框架- 参考 dubbo、seata rpc框架
架构师实操篇：go语言学习，以及基于 go 手写 rpc 框架
架构师实操篇：千万级任务调度平台 架构与实操- 基于尼恩17年的亿级搜索项目
架构师实操篇：工业级 亿级文档搜索 平台 架构与实操- 基于尼恩17年的亿级搜索项目
特色
会员制
提供技术方向指导，
职业生涯指导，少躺坑，少弯路
简历指导
这个很重要，
对于挪窝涨薪来说
实操性
以上项目，都是老架构师
在生产上实操过的项目
非水货
40岁老架构师，不是水货架构师
《Java高并发三部曲》为证

架构班（社群 VIP）的起源：

最初的视频，主要是给读者加餐。很多的读者，需要一些高质量的实操、理论视频，所以，我就围绕书，和底层，做了几个实操、理论视频，然后效果还不错，后面就做成迭代模式了。

架构班（社群 VIP）的功能：

提供高质量实操项目整刀真枪的架构指导、快速提升大家的：

- 开发水平
- 设计水平
- 架构水平

弥补业务中 CRUD 开发短板，帮助大家尽早脱离具备 3 高能力，掌握：

- 高性能
- 高并发
- 高可用

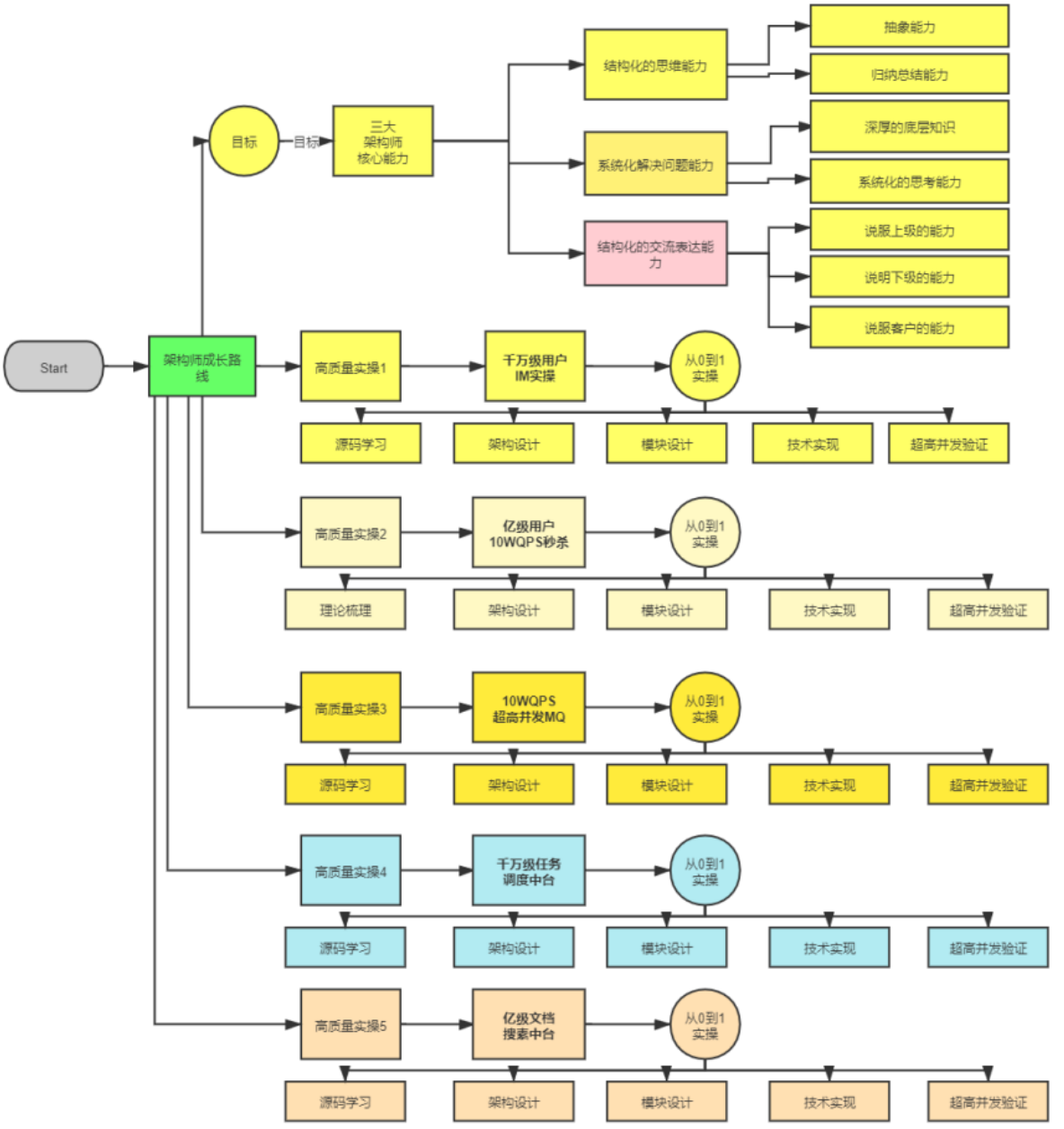
作为一个高质量的架构师成长、人脉社群，把所有的卷王聚焦起来，一起卷：

- 卷高并发实操
- 卷底层原理
- 卷架构理论、架构哲学
- 最终成为顶级架构师，实现人生理想，走向人生巅峰

架构班（社群 VIP）的目的：

- 高质量的实操，大大提升简历的含金量，吸引力，增强面试的召唤率
- 为大家提供九阳真经、葵花宝典，快速提升水平
- 进大厂、拿高薪
- 一路陪伴，提供助学视频和指导，辅导大家成为架构师
- 自学为主，和其他卷王一起，卷高并发实操，卷底层原理、卷大厂面试题，争取狠卷 3 月成高手，狠卷 3 年成为顶级架构师

N 个超高并发实操项目：简历压轴、个顶个精彩



【样章】第 17 章：横扫全网 Rocketmq 视频第 2 部曲：工业级 rocketmq 高可用（HA）底层原理和实操

工业级 rocketmq 高可用底层原理，包含：消息消费、同步消息、异步消息、单向消息等不同消息的底层原理和源码实现；消息队列非常底层的主从复制、高可用、同步刷盘、异步刷盘等底层原理。

工业级 rocketmq 高可用底层原理和搭建实操，包含：高可用集群的搭建。

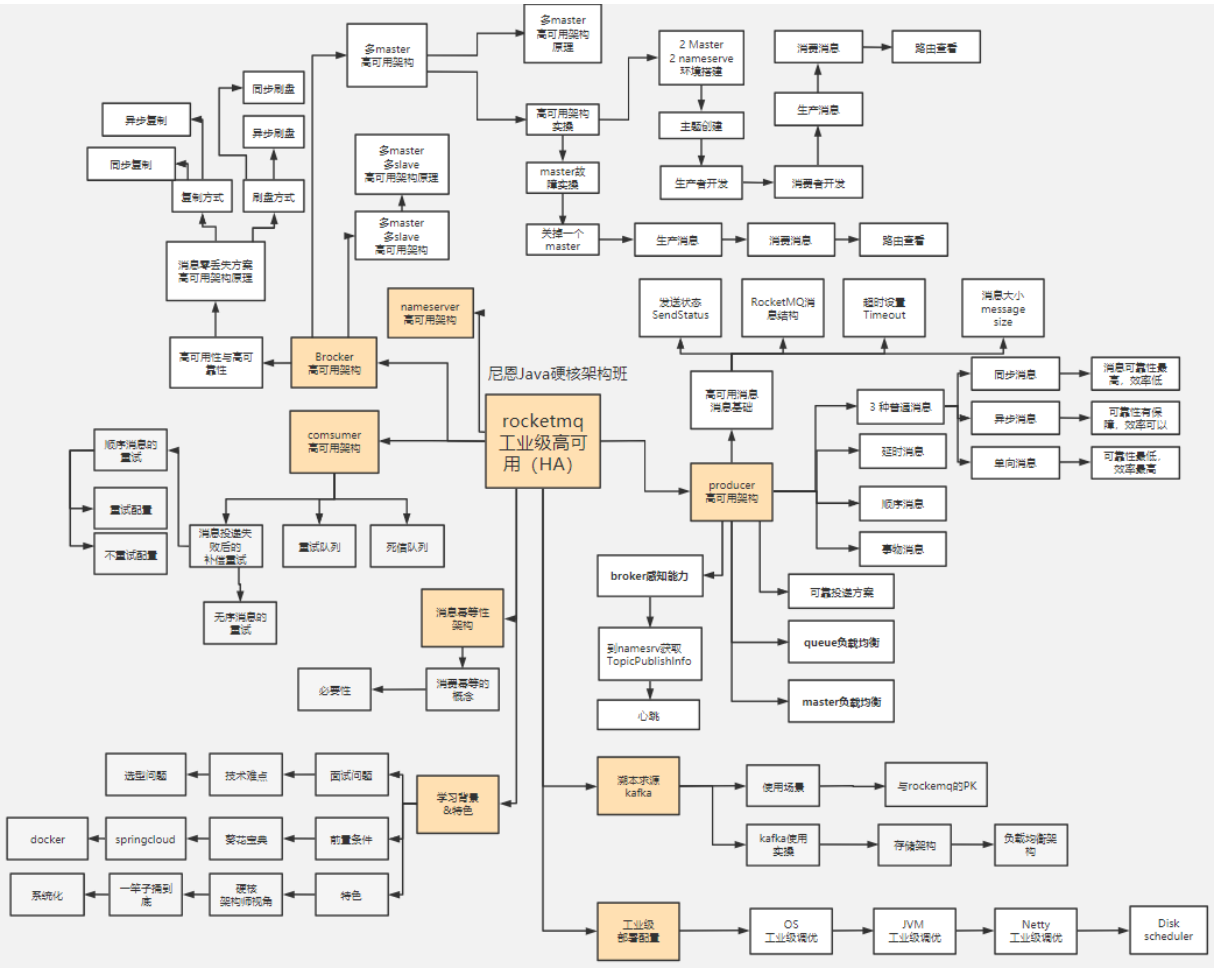
解决以下难题：

1、技术难题：RocketMQ 如何最大限度的保证消息不丢失的呢？RocketMQ 消息如何做到高可靠投递？

2、技术难题：基于消息的分布式事务，核心原理不理解

3、选型难题： kafka or rocketmq ，该娶谁？

下图链接：https://www.processon.com/view/6178e8ae0e3e7416bde9da19



成功案例：2 年翻 3 倍，35 岁卷王成功转型为架构师

详情：http://topcoder.cloud/forum.php?mod=forumdisplay&fid=43&page=1

最新　最后发表　热门　精华

成功案例：[1057号卷王] 3年小伙拿到外企offer，薪酬涨了200%
卷王1号　超级版主　前天 17:41

成功案例：[645号卷王] 4年经验卷王逆袭，被毕业后，反涨24W
卷王1号　超级版主　2022-9-21

成功案例：[878号卷王] 小伙8年经验，年薪60W
卷王1号　超级版主　2022-8-13

年薪70W案例：通过尼恩的指导，小伙伴年薪从40W涨到70W
卷王1号　超级版主　2022-2-11

成功案例：[493号卷王] 5年小伙拿满意offer，就业寒冬季逆涨30%
卷王1号　超级版主　前天 17:43

成功案例：[250号卷王] 就业极寒时代，收offer 涨25%
卷王1号　超级版主　前天 17:38

成功案例：[612号卷王] 就业极寒时代，从外包到自研
卷王1号　超级版主　前天 17:15

成功案例：[913号卷王] 热烈祝贺6年经验卷王，年薪40W
卷王1号　超级版主　2022-9-21

成功案例：[959号卷王] 4年经验卷王，喜获百度、Boss直聘等N个优质offer，最高涨100%
卷王1号　超级版主　2022-9-21

成功案例：[529号卷王] 5年经验卷王喜收2大offer，最高涨5K
卷王1号　超级版主　2022-9-21

成功案例：[811号卷王] 热烈祝贺7年经验卷王，薪酬涨30%
卷王1号　超级版主　2022-9-21

成功案例：[287号卷王] 不惧大寒潮，卷王逆市收4 offer，涨30%，可喜可贺
卷王1号　超级版主　2022-5-30

成功案例：[1002号卷王] 5月份"被毕业"，改简历后，斩获顶级央企Offer，涨薪7000+
卷王1号　超级版主　2022-7-5

成功案例：[7号卷王] 热烈祝贺小伙伴涨薪120%

卷王1号 超级版主 2022-8-13

成功案例：[134号卷王] 大三小伙卷1年，斩获顶级央企Offer，成功逆袭

卷王1号 超级版主 2022-7-6

成功案例：[1008号卷王] 5年经验卷王收42W offer，月涨8000，可喜可贺

卷王1号 超级版主 2022-5-30

成功案例：[453号卷王] 非全日制 6年卷王喜提3 offer，年薪30W，可喜可贺

卷王1号 超级版主 2022-5-21

成功案例：[924号卷王] 6年卷王喜提4 offer，最高涨薪9000，可喜可贺

卷王1号 超级版主 2022-5-21

成功案例：[15号卷王] 4年卷王入职 微软，涨薪50%，可喜可贺

卷王1号 超级版主 2022-5-12

成功案例：[527号卷王] 4年卷王喜提2 offer，涨薪50%，可喜可贺

卷王1号 超级版主 2022-5-13

成功案例：[788号卷王] 3年卷王喜提优质Offer，涨薪60%

卷王1号 超级版主 2022-5-11

成功案例：热烈祝贺：非全日制卷王，喜提2个心仪offer，面3家过2家

卷王1号 超级版主 2022-4-21

成功案例：[693号卷王] 二线城市6年卷王喜提4大优质Offer，含央企offer，最高薪酬35W

卷王1号 超级版主 2022-4-16

成功案例：[85号卷王] 双非2本小伙，春招大捷，喜提9个offer，最高薪酬近30万

卷王1号 超级版主 2022-4-14

成功案例：[741号卷王] 卷王逆袭！6年小伙从很少面试机会到搞定35K*14薪Offer

卷王1号 超级版主 2022-4-12

成功案例：[642号卷王] 热烈祝贺，6年卷王喜提优质国企offer

卷王1号 超级版主 2022-4-7

成功案例：[796号卷王] 热烈祝贺，36岁卷王喜提52万优质offer

卷王1号 超级版主 2022-3-25

成功案例：[15号卷王] 小伙卷1年，涨薪9K+，喜收ebay等多个优质offer

卷王1号 超级版主 2022-3-24

成功案例：[821号卷王] 小伙狠卷3个月，喜提10多个offer

卷王1号 超级版主 2022-3-21

成功案例：[736号卷王] 3年半经验收22k offer，但是小伙志存高远，冲击25k+

卷王1号 超级版主 2022-3-20

成功案例：热烈祝贺1群小卷王offer拿到手软，甚至拒了阿里offer

卷王1号 超级版主 2022-3-16

简历案例：简历一改，腾讯的邀请就来了！热烈祝贺，小伙收到一大堆面试邀请

卷王1号 超级版主 2022-3-10

成功案例：祝贺我圈两大超级卷王，一个过了阿里HR面，一个过了阿里2面

卷王1号 超级版主 2022-3-10

成功案例：小伙伴php转Java，卷1.5年Java，涨薪50%，喜收多个优质offer

卷王1号 超级版主 2022-3-10

成功案例：4年小伙狠卷半年，拿到 移动、京东 两大顶级offer

尼恩 超级版主 2022-3-5

成功案例：[267号卷王] 助力3年经验卷王，拿到蜂巢的17k x 14薪的offer

卷王1号 超级版主 2022-2-27

成功案例：[143号卷王] 二本院校00后卷神，毕业没到一年跳到字节，年薪45W

卷王1号 超级版主 2022-2-27

成功案例：[494号卷王] 尼恩分布式事务助力卷王拿到 中信银行offer

卷王1号 超级版主 2022-2-27

成功案例：[76号卷王] 2线城市卷王，狠卷1.5年，喜收22K offer

卷王1号 超级版主 2022-2-27

成功案例：[429号卷王] 小伙伴在社群卷5个月，涨8k+

卷王1号 超级版主 2022-2-27

成功案例：[154号卷王] 双非学校毕业卷王，连拿 京东到家&滴滴 两个大厂Offer

卷王1号 超级版主 2022-2-27

成功案例：[232号卷王] 涨薪10K，继续卷向食物链顶端

卷王1号 超级版主 2022-2-27

成功案例：狠卷1年技术，喜收 腾讯、阿里、微软三大Offer，最高年薪56W

卷王1号 超级版主 2022-2-27

成功案例：[449号卷王] 应届毕业卷王喜收 滴滴offer, 年薪33W

卷王1号 超级版主 2022-2-27

成功案例：[551号卷王] 小伙伴学完后，成功进入大厂，并且推荐自己的朋友加VIP学习

卷王1号 超级版主 2022-2-10

成功案例：[214号卷王] 助力2年经验卷王，成功拿到17K月薪

卷王1号 超级版主 2022-2-10

成功案例：[92号卷王] 课程实操助力社群小伙伴喜收 喜马拉雅Offer

卷王1号 超级版主 2022-2-10

成功案例：社群卷王小伙伴成功过了滴滴三面 获滴滴Offer

卷王1号 超级版主 2022-2-10

[612号卷王]滴滴小伙伴，蹲点考察半年，觉得靠谱后加入 疯狂创客圈

卷王1号 超级版主 2022-2-10

成功案例：[732号卷王] 尼恩助力3年经验卷王收获 京东offer, 年薪35W

卷王1号 超级版主 2022-2-27

成功案例：[558号卷王] 2年经验卷王，喜收 网易和阿里子公司两个优质offer

卷王1号 超级版主 2022-2-27

成功案例：[569号卷王] 双非应届生卷王, 喜收字节跳动实习offer

卷王1号 超级版主 2022-2-25

成功案例：[420号卷王] 狠卷1年，卷王涨薪80%，涨薪12000元!

卷王1号 超级版主 2022-2-25

成功案例：[76号卷王] 通过尼恩1年半的指导，专科学历小伙伴从0.8K涨到22K

卷王1号 超级版主 2022-2-10

简历优化后的成功涨薪案例（VIP含免费简历优化）


9年 小伙伴拿到
年薪90W offer
9月11日改简历
11月29日晒offer
秘诀:
简历指导+ 狠狠卷
这个是给你微调的
省略号的地方，需要你再补充一点
上面你留着这个就行
Java 开发 9年-修改.docx
34.1 KB
薪资高 稳
主要的工作是啥?
提升了自己的实力，就不用怕
易所 关于数字货 的 易
po 也有 打盹的时候，该裁员，照样一个不少
年包比 po 多 19w
这么多
po 估计有 70 万
那你不有 90 个 W?
po 给我 68
就是吗


小伙8年经验
年薪60w
7月12日改简历
8月10日晒offer
秘诀:
改简历 + 狠狠卷
明天晚上哈
好哒
恩哥，今晚还改简历吗?
今晚还在外边应酬，估计回去比较晚
要不，咱们延迟到明天，如何
明天白天也行
好的，白天吧，答应别人明天给他们简历了。
OK
那就上午 11 点左右哈
好的
涨幅多少呀?
或者，额度多少呀
之前 36*15，现在这个 39*15
今年行情不太好，还有一些 offer 基本都是平薪，或降薪的。
OK
这个马上来
刚在指导简历
哈哈
稍等哈
辛苦啊恩哥
这次找工作，您的 起到效果了。
我这次复习基本看的都是咱们课程的 。


6年小伙伴
年薪40w
9月6日改简历
9月21日晒offer
秘诀:
简历指导+ 狠狠卷
Java - 6年.docx
24.7 KB
恩哥，简历我改好了，您再帮我看一下
Java - 6年(1).docx
24.5 KB
恩哥，看第二个吧
给你前面调整了一下
涨这多
今年这行情，也算可以了
总包多少呀，让我也陶醉一下
大概 40w 吧
谢谢恩哥的指导和鼓励
是在深圳
深圳的行情尤其难
能有面试电话就不错啦
是的
太牛啦
哈哈哈哈，恩哥的鼓励指导也很重要


5年小伙喜提3个offer
年薪 35个W
5月22日改简历
11月29日晒offer
我简历还没写出来 不然可以给老师您看看
算法方面，可以找点 刷刷，当作复习
项目的话，再好好改造一下
好的 谢谢老师
java 开发-本科统招-2022-5 年经验.docx
36.0 KB
OK
我排好号，回头找你哈
秘诀:
简历指导+ 狠狠卷
尼恩老师晚上好，汇报下最近的 offer 情况。最近面试收到了三个 offer，两个是平薪，一个是跨境电商公司的 offer，涨幅暂时不到 20%。通过这次面试也让我知道自己距离高级开发还有一点点距离，还要再多看才能突破。
最后结合自己的情况，先选择去跨境电商的公司再提升下
之前是 20k*13.5，跨境电商这个是说 23k*（14-15）
恭喜恭喜
就业寒冬，能提到 3 个 offer，已经很 ，已经是高手了
很多小伙伴，面试电话一个都接不到，简历海投 7000 份，只收到 3 次面试机会，没有一个机会拿到最终 offer
您这个年薪，算下来也有 35W 了吧
年终部分还要确认下，大部分人听说只有两个月
这个时间点，拿到这个水平，挺不错的啦
持续加油卷哈
嗯嗯！看看明年自己有没有能力冲击高开
把你我视频都吃透
辛苦老师再帮忙指导下哈


1.5年小伙搞定15K offer
就业寒冬涨100%
5月7日改简历
11月21日晒offer
秘诀：
简历指导+狠狠卷
多谢啦
你好，好多公司都觉得我的简历太简单了，我感觉是不是没法优化了
简历.pdf
397.4 KB
好的
晚上指导你改
推送中台需要加不
我还没做啊，准备八股文暂时没时间搞这些，入职了会开始搞
OK
还有其他的项目吗
一个项目，太少啦
那种练习项目也行
他那个不止两倍吧，我才是两倍，原先7k，现在15k
那也很牛
都翻倍了，都是牛人
正常就30%
慢慢来
下一步可以瞄准25k
其实我工作一年
那你就更牛逼拉
一年经验，搞定15k offer，舍你其谁


卷王逆袭成功案例
6年小伙从很少面试机会到
搞定35K*14薪
3月5日改简历
4月11日拿offer
面试邀请少
小伙很苦恼
理想很丰满
目标35K
我要努力争取35+
我指导简历，非vip收费3000，值这个价吧？
一个月拿到了
理想的offer
面试法宝
rocketmq四部曲

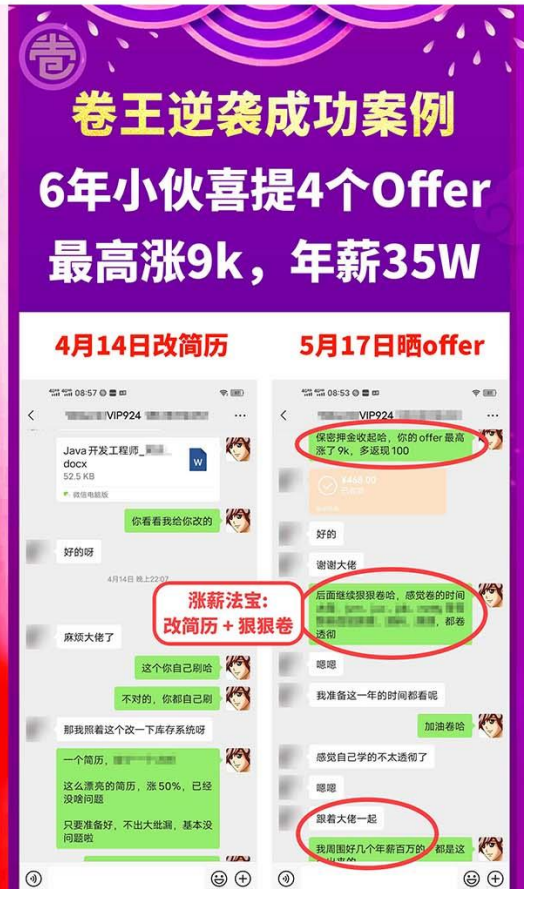

6年 经验小伙伴
喜收25K offer
3月12日改简历
12月1日晒offer
秘诀：
简历指导+狠狠卷
老师
咱们以这个项目为模板改哈
嗯好
项目有多少人，你带领多少人？
你工作几年了？
6年了
项目有多少人，你带领多少人？
语音交流一下
14:48
好
你方便吗
OK
不知道往哪个方向发展
改简历挪窝，20 涨到 25 k，高级开发，P7 带的后端团队
任务重，道路远，并不是没有方向


7年经验卷王
薪酬涨30%
7月11日改简历
9月1日晒offer
秘诀：
改简历 + 狠狠卷
咱们开始修改哈
要语音还是打字也行，还没到家
解决过最复杂的难题是啥？说3个哈
看看，有没有问题？
没问题
看看，有没有问题？
只能挪窝
下次你喊足一点哦
只要本事好，拿 offer 比较容易的
喊了，但是我流水太差了。。。我这家是补贴，银行不认这玩意，只能在上家的基础上加到 30% 封顶（特别优秀除外，我只是不错。。。）
入职了，最终并不能大幅涨薪，还是旧的太保，只涨了 30%
恭喜一下
现在不比往年
能涨 30% 是大牛
拿到 offer 都算大牛
现在很多人投几百发，连个电话都没有
你已经很


4年经验卷王逆袭
被毕业后，反涨24W
7月改简历
8月30日晒offer
秘诀:
改简历 + 狠狠卷
这就是你的简历
差得太多啦
ok
总共写了四个项目，最近的一年的还没补充上
你是在职，还是离职呢?
离职
原因大概是啥?
这个地方要这样写
项目被终止
方便语音沟通不
ok
是的 经你这么指导，非常重要
老哥 我八月十号开始找工作，今天已经入职了
现金基本持平，股票 +24W
总计涨了多少呢
能涨24W
股票这个吧只能到手了才算
也不错啦
很多小伙伴，面试机会都没有
感谢老哥的指导，继续跟你卷技术
继续狠狠卷哈，马上就技术自由拉

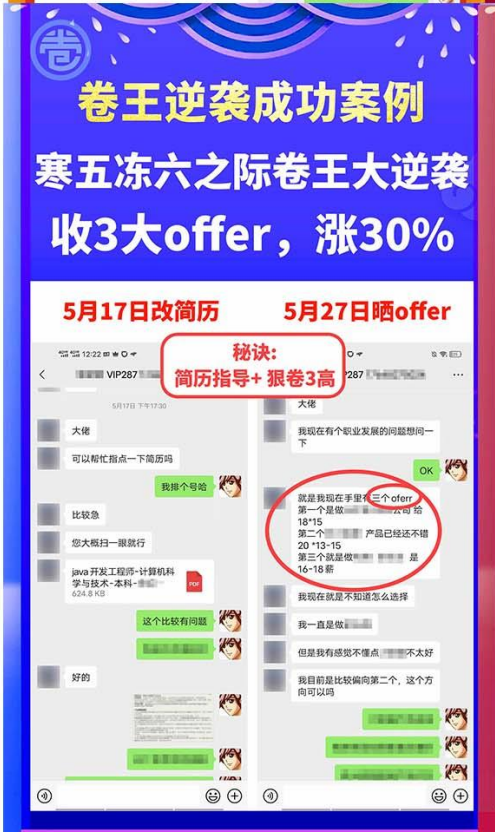

小伙5月份"被毕业"，改简历后
斩获顶级央企Offer
涨薪7000+
5月29日改简历
7月5日晒offer
秘诀:
简历指导+ 狠卷3高
快速看书，就要不求甚解，把目录和 场景大概一下，然后重点的地方，用到的地方，再去回炉
尼恩
我被 " 毕业 " 了
这周末或下周我你改一下简历
毕业了没有关系
ok，发我吧
it行业，跳来跳去，太频繁啦
嗯，其实有点心理准备
简历
不太会写简历
尼恩
我拿到 的offer 了
涨20%，
不还价的
看起来 不差钱呀
超过了8000没
平均算下来
7000多
好的
有啥面试的心得吗
可以分享给其他小伙伴的


卷王逆袭成功案例
武汉6年喜收4个优质offer
最高的年薪35W
2月9日改简历
4月15日晒offer
面试法宝:
改简历 + 实操
尼恩老师，新年好!
能帮忙修改下简历吗?
金三银四准备跳了
可以的
java 开发-6 年-简历-
.pdf
拜托了，恩师。希望能拿 25k 回来给你报喜
好，我加一下
还有吗?
恩哥，决赛圈 offer 了
截图是以我目前认知能写出来的评分了，麻烦帮我参考下
选择大于努力，恩师助我上岸
这么多 offer，我看着哈
都是恩哥指点有方，本来还有个新能源汽车的，35w 给拒了，主要太远了
跟着恩哥卷的时间太短了，目前实力也只能到这儿了
这里边有个大数据的，感觉也不错


卷王逆袭成功案例
6年小伙喜提4个Offer
最高涨9k，年薪35W
4月14日改简历
5月17日晒offer
涨薪法宝:
改简历 + 狠狠卷
Java 开发工程师_
.docx
你看看我给你改的
好的呀
麻烦大佬了
这个你自己刷哈
不对的，你都自己刷
那我照着这个改一下库存系统呀
一个简历，
这么漂亮的简历，涨 50%，已经没啥问题
只要准备好，不出大纰漏，基本没问题啦
保密拼金收起哈，你的 offer 最高涨了 9k，多返现 100
好的
谢谢大佬
后面继续狠狠卷哈，感觉卷的时间 都卷透彻
嗯嗯
我准备这一年的时间都看呢
加油卷哈
感觉自己学的不太透彻了
嗯嗯
跟着大佬一起
我周围好几个年薪百万的，都是这


卷王逆袭成功案例
5年经验小伙收2个offer
最高涨薪8k，年薪42W
5月9日改简历
5月30日晒offer
秘诀:
简历指导+ 狠卷3高
以此为样
大家狠狠卷
打造最卷IT社群


卷王逆袭成功案例
非全日制 6年经验卷王
喜提3个Offer，年包30W
5月9日改简历
5月18日晒offer
面试法宝:
改简历 + 狠狠卷


卷王逆袭成功案例
寒五冻六之际卷王大逆袭
收3大offer，涨30%
5月17日改简历
5月27日晒offer
秘诀:
简历指导+ 狠卷3高


卷王逆袭成功案例
4年卷王入职微软，涨50%
3月7日改简历
5月12日晒offer
涨薪法宝:
改简历 + 狠狠卷


4年小伙喜收百度、Boss直聘
等N个顶级Offer
最高涨幅100%
6月27日改简历
9月19日晒offer
秘诀:
改简历＋狠狠卷


卷王逆袭成功案例
4年卷王入收2个offer，涨50%
3月23日改简历
5月12日晒offer
涨薪法宝:
改简历＋狠狠卷


小伙大三暑期很焦虑
跟着尼恩卷一年
校招斩获顶级央企Offer
去年5月19日加入VIP群
今年7月5日晒offer
秘诀:
狠狠卷 书+视频


小伙高中学历
薪酬涨120%
5月6日改简历
7月22日晒offer
秘诀:
改简历＋狠狠卷


卷王逆袭成功案例
非全日制卷王 面试3家
收2个offer 涨薪30%
4月13日改简历
4月21日晒offer
面试法宝：
改简历 + 面试题


5年卷王喜收2大Offer
最高涨5K
5月19日改简历
9月13日晒offer
秘诀：
改简历 + 狠狠卷


卷王逆袭成功案例
3年经验卷王，涨60%
4月16日改简历
5月11日晒offer
涨薪法宝：
改简历 + 狠狠卷


卷王逆袭成功案例
双非二本小伙春招大翻身
喜提9大offer
2月22日改简历
4月13日晒offer
面试法宝：
改简历 + IM实操
9大offer 最高年薪30万

修改简历找尼恩（资深简历优化专家）

- 如果面试表达不好，尼恩会提供 简历优化指导
- 如果项目没有亮点，尼恩会提供 项目亮点指导
- 如果面试表达不好，尼恩会提供 面试表达指导

作为 40 岁老架构师，尼恩长期承担技术面试官的角色：

- **从业以来，“阅历”无数，对简历有着点石成金、改头换面、脱胎换骨**的指导能力。
- **尼恩指导过刚刚就业的小白，也指导过 P8 级的老专家，都指导他们上岸。**

如何联系尼恩。尼恩微信，请参考下面的地址：

语雀：https://www.yuque.com/crazymakercircle/gkkw8s/khigna
码云：https://gitee.com/crazymaker/SimpleCrayIM/blob/master/疯狂创客圈总目录.md