如何自学计算机网络？

车小胖

09/22/2017

前言

如果参加此live的同学，希望三个月精通网络，请选择退款，此live不能达到速成的效果。

但，参加此live的同学，如果真的如这个live所要求的做，三个月可以基础入门，三年可以成为合格的工程师。

Live 内容大纲

* 为何要学计算机网络
* 学好计算机网络的基本素质
* 掌握正确的学习方法
* 如何突破计算机网络的学习瓶颈

为何要学计算机网络

生活离不开计算机网络

计算机网络是信息社会的中流砥柱

- 网络向上对软件提供服务，向下需要指挥硬件工作，起到一个承上启下的中枢作用。
- 计算机网络对终端用户透明，让不懂网络的用户可以毫无障碍上网。
- 一旦用户感受到网络存在时，用户就会知道网络是如此地重要，这时也是网络工程师最繁忙的时刻！

为何要学好计算机网络

业余爱好者

- 单位、家庭搭建小型网络不求人，即使遇到问题，也是小菜一碟
- 给亲朋好友搞定网络问题，有水平、有面子

软件工程师

- 深入研究TCP/IP协议的内部代码，才能更深入的理解Socket
- 对网络理解的越深入，越能知道如何提升软件的性能

测试工程师

- 广泛学习真实网络部署场景，将有助于QA工程师搭建接近真实的测试平台
- 对网络技术的理解的越深入，越能发现隐藏很深的bug

网络工程师

- 可以到知乎回答这样的问题：月入三万却存不下钱是一种什么样的体验？

计算机网络的复杂性

- 小明在家上网的协议

PPPoE 、PPP、PAP、CHAP、IPCP、802.11、Ethernet、WEP、WPA2、TKIP、AES、SHA、MD5、DHCP、ARP、IP、ICMP、NAT、UDP、 DNS、TCP、TLS、HTTP、FTP、SMTP

- 小明远程拨公司VPN

SSH、L2TP、PPTP、GRE、IKEv1/2、AH、ESP、LEAP、FAST 、PEAP、Radius、TCACAS+、Kerberos

- 小明拨打公司VoIP电话

Skinny、SIP、SDP、H.323、H.245、H.225、RAS、Q.931、SIP-T、MGCP、RTP、RTCP、RTSP、G.729、G.726、G.723、G.711

- 小明公司网络

STP、RSTP、MST、UDLD、VRRP、LLDP、LACP、VTP、802.1x、801.1Q、LWAPP、LISP、ALG、Sock5、Proxy、DHCP Relay、IGMP、IGMP-snooping、ACL 、OSPF、IS-IS、BGP、PIM dense 、 PIM sparse、PIM-SSM、PIM-bdir

- 小明公司数据中心

VxLAN 、TRILL 、 QINQ、Fabric Path、OTV、vPC、EVPN、CIFS、SMB、NFS、FC、FCoE、FCIP、iSCSI

- 小明公司连接的运营商

MP-BGP、LDP、RSVP-TE、MVPN、VPLS、MPLS-TP、MPLS-OAM、SDH、WDM、PON、ATM、FR、HDLC、STMx

计算机网络复杂性的客观因素

不是每家厂商都能严格遵守RFC协议规范，厂商对于某项技术有自己的独树一格的实现，比如SNAT

- source NAT　　一般厂家
- stateful NAT　　Cisco System
- static NAT　　Watch Guard
- secure NAT　　F5 / Microsoft

除了熟悉RFC协议标准，还需要通过工作经历来熟悉厂家的私有实现

学好计算机网络的基础素质

英语阅读能力

- 养成阅读英文文献的习惯， 可以流畅阅读计算机网络的协议标准、文献、白皮书，不能因为语言本身而造成阅读障碍。

逻辑推理能力

- 计算机网络是一门集数学、逻辑、实践相结合的工程应用学科。当处理网络故障时，需要一个清晰的逻辑推理能力，排除不可能，缩小排查的范围，减少工作量并提高效率

计算机网络的三个学习阶段

基础入门

- 熟练掌握TCP/IP协议，协议的字段，协议完成的功能。

中级修炼

- 多协议融合，清晰协议在场景里的发挥的角色。

高级进阶

- 具有宏观、前瞻性意识，能敏锐发现网络隐藏的缺陷、不足。
- 逆向工程，能够根据黑盒子的外在表现推测内部实现。

基础入门之学习内容

- •OSI参考模型
- •TCP/IP协议
- •IP地址、网络掩码、缺省网关
- •同网段通信、不同网段通信
- •Ping、Traceroute
- •桥接、路由、NAT
- •二层交换机、二层交换
- •三层交换机、三层路由
- •静态路由、路由协议
- •VLAN、Trunk、子接口、SVI
- •Spanning Tree
- •Access List

基础入门之OSI参考模型

一层设备

- 物理层设备，只对物理信号（光、电、电磁波）放大、修复、中继的设备，对信号里的内容不观察、不讨论、不关心！ 如光中继设备、HUB等

二层设备

- 链路层设备，读取二层协议头，做出转发动作。二层设备只会做二层转发，如果做三层转发，那就是三层设备了。二层设备包括交换机、AP。

三层设备

- 网络层设备，读取三层协议头，并依据地址信息做出转发动作。只要愿意，可以读取三层到七层任何数据，但只是基于三层地址转发，包括路由器、三层交换机。

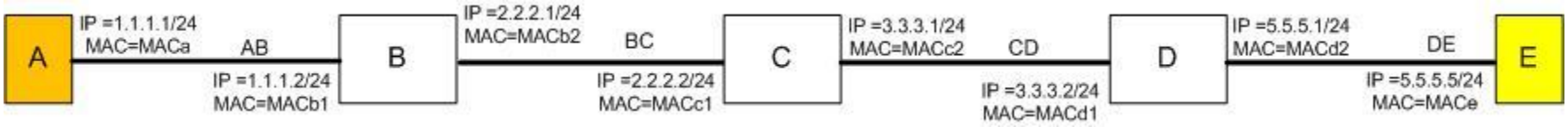
四到七层设备

- 读取三、四、七层的协议信息，基于session的负载均衡、基于session的防火墙、应用层网关。
- 精通各种不同应用层协议的格式，精确化区分各种应用流量，从而做出恰当的流量转发动作。

一个误区：
二层设备只认识二层协议头？三层或三层以上的头不认识？
不是的！ 二层负载均衡的时候，二层设备也会读取三层的IP、四层的端口号来做基于session 的负载均衡，通俗地说，就是用户的一个会话连接用同一条物理链路。
既然三、四层的协议格式都是公开的，二层设备完全可以读取它们来为自己服务。

基础入门之示例

1. A、E 为主机， B、C、D为路由器，现在来看看A的IP报文如何到达E?
2. 在每一跳链路层协议头如何变化？



- 主机A接到上层协议的请求，请把数据包Payload发给5.5.5.5

为何不是5.5.5.5/24?

主机A无法知道5.5.5.5的掩码长度，通常获得E的IP地址是通过DNS解析而来，而DNS只提供IP地址服务，并不提供掩码服务！退一步讲，知道对方的掩码又如何？掩码只是本地有效，A的掩码只是本地有意义，E的掩码也只是在E上有意义。

- A首先查找路由表，按照最长匹配原则，匹配的长度越长越优先使用，反之越后使用。

匹配到一条路由：0.0.0.0/0　1.1.1.2 (下一跳)　E0 (本地网卡接口)

意思是目的IP = 5.5.5.5的IP报文，需要发给下一跳1.1.1.2中转，与下一跳直连的接口是E0，使用E0接口的IP=1.1.1.1作为源IP，IP封装头部可以完成封装了,见下图：

基础入门之示例

目的IP、源IP想必大家都已经明白了，Payload是上层协议请求传递过来的，那这个Protocol = 6 IP层是如何确定的？

也是上层协议请求传递过来的，其代表TCP协议。

目的IP=5.5.5.5 也是上层协议传递下来的。

IP协议头的关键信息其实都是上层协议传递下来的，只要想想生活中的场景就很好理解。

我们得出一个结论：IP层的关键信息是上层协议传递下来的！

为何要用这个协议号？

因为IP层的客户很多，不仅有TCP，还有UDP、ICMP、IGMP、GRE、OSPF、ESP、AH、IP、IPv6，需要至少有一个标识符来区分这些协议

IP Header里的其他信息，ToS，IP头长度、IP报文长度、TTL、Checksum、与IP分片有关的ID、DF/MF、Offset，都由IP层自己来填写。

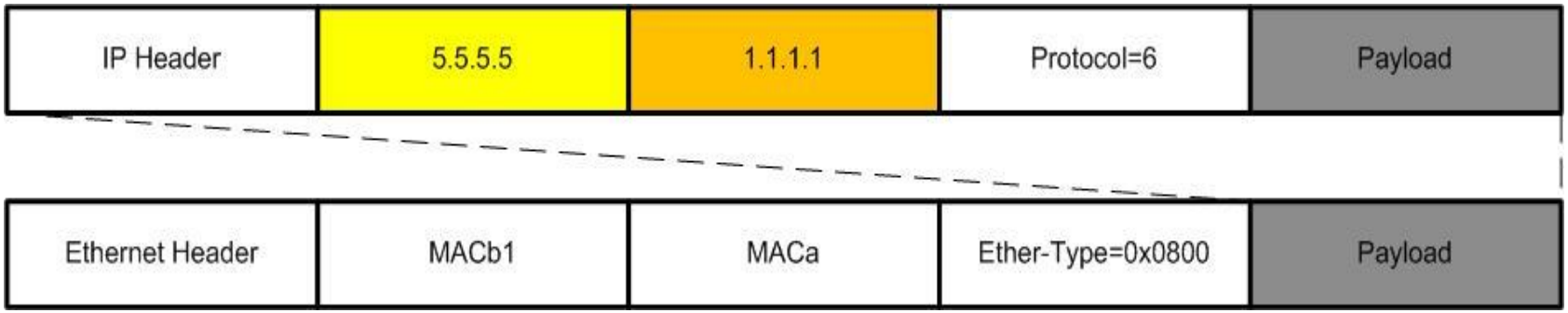
协议号里竟然有IP、IPv6，这是什么意思？

难道IP层头里可以直接封装IP报文、或者IPv6报文？是的，其实对于IP层来说，根本不在乎上文灰色的Payload里装的是什么！如果协议的制定者，再为PPP、Ethernet、FC定义各自的协议号，也是可以封装它们的！

PPP、Ethernet、FC不都是二层头吗？是的，没错！这并不矛盾，在IP眼里，这些协议头仅仅是自己的Payload，至于是TCP、或IP、或PPP，IP层一点也不care。

基础入门之示例

- IP层需要借助ARP协议，动态发现对端B的MAC地址，这里为MACb1，有了这些信息，就可以将此信息、IP报文一起传递给网卡，网卡用自己的MACa作为源MAC地址，完成自己的二层封装，如下图：



IP层的关键地址信息是由上层传递下来的，此法则同样适用于链路层。潜台词是IP层要负责查找到下一跳的MAC地址信息，然后再传递给网卡。

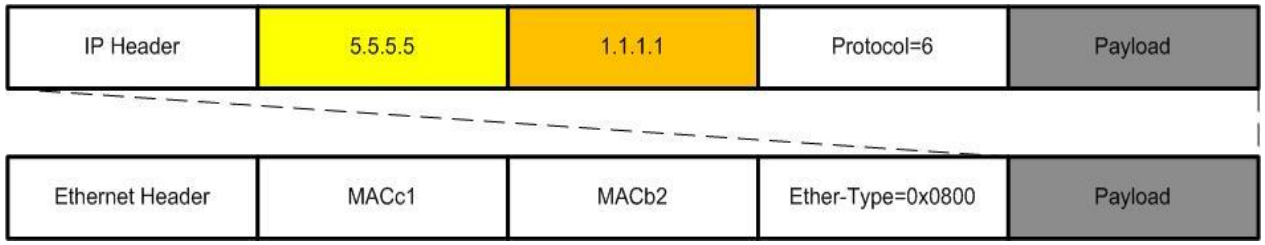
在二层封装的眼里，上部的IP报文已经成为自己的Payload。这里又出现了一个Ether-Type= 0x0800，代表着Payload是IP报文，同样也是由IP层提供，这个字段的目的是为了区分上层协议，因为二层不仅仅可以封装IP，还可以封装ARP、IPv6、PPPoE、FC等等。封装好，就可以将以太帧发送出去了。

- B的网卡接到A发出的以太帧，CRC校验通过，网卡发现此帧的MACb1= myself，接收并剥掉二层协议头放入缓冲区，依据ether-type=0x0800通知IP层（事先注册回调函数）来取走数据。
- 二层以太帧头结束了其平淡而短暂的一生！
- 但IP协议头却依然健在，因为B、C、D要依靠目的IP地址来寻找最终的目的地。

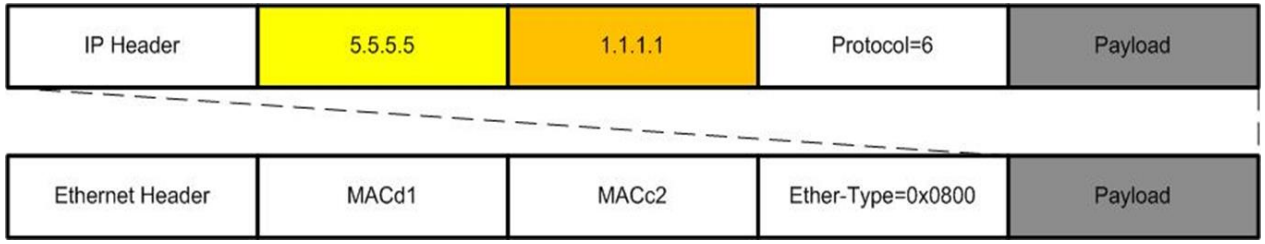
基础入门之示例

接下来就是一个迭代过程，只是需要不断变换二层头部信息

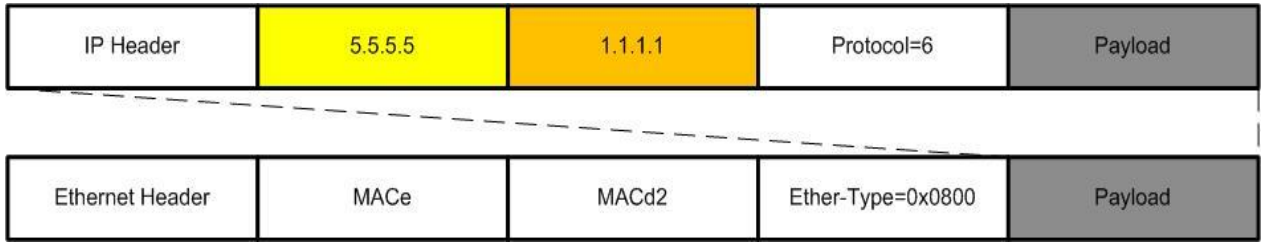
- B-C之间的链路为：



- C-D之间的链路为：



- D-E之间的链路为：



基础入门之示例

- IP报文终于抵达了目的地，**E发现目的IP竟然是自己的**，IP层剥离掉IP协议头，IP协议头也结束了其漫长而神奇的一生！

归纳 ：

IP协议头地址信息一直保持不变。

以太帧协议头每经过一条物理链路，就会发生变化。

IP报文从源出发，到达目的地的整个过程，是一个不断重复迭代的路由查找过程，有三种结果：

a. 自己的　　依据protocol值通知上层协议处理数据

b. 别人的　　查出口对端的链路层地址，完成二层封装，发送出去

c. 查无此人　　丢

以上这段是对初学者最大的困扰，这是最最基础的东西，也是最最重要的东西，把以上文字理解透彻了，将有助于计算机网络的学习。

基础入门之学习心得

看了很多书，但一直没有硬件练手，一直有一个问号，这样是否可以学出来？

曾经把Jeff Doyle的卷一、卷二读过N遍时，一直没有做过实验，当真有机会摸路由器的时候，如饥似渴一直在敲击键盘做好多实验，来验证自己对路由协议的理解，我可以几乎不看书，凭自己的理解与记忆，基本上配好了也就好了，没有遇到难题。

只是在配置BGP时遇到一些挫折，配完之后BGP没有丝毫动静，BGP两端我使用的用loopback 接口做router ID，同时peer neighbor 也是对方的router ID，知道问题出在哪里吗？因为没有配置update-source-interface loopback，BGP一直使用默认的与对方直连的物理接口建立BGP session，而对方期望的却是本地的loopback接口地址，所以拒绝连接。

对理论理解的深入，做实验也有针对性，可以验证深入思考的地方，这样可以在有限的实验环节做到高效率。

另外，当看书看到一定程度时，会有一种冲动，会抓住一切机会练手，在公司会打开抓包软件，会去研究各种流量，ARP、BPDU、DHCP、DNS、ICMP、HTTP。

在家还可以抓包研究PPP、PPPoE。PPPoE协议是我隧道协议的启蒙老师，同时我也理解了，什么是控制层面、数据层面、管理层面。

基础入门之测试题

第一个问题：10.1.207.2/20 ，10.1.208.3/20，在不在一个网段？请于30秒之内算出

第二个问题：一个主机192.168.1.199/26 能否和直连主机192.168.1.200/24 通信？

第三个问题：Ping 的Request、Reply 分别对应的ICMP消息类型，源根据什么字段来把Request/Reply一一对应起来？

第四个问题：Traceroute 的工作原理。

中级修炼之学习内容

深度学习路由协议

- 精通，防环机制、协议本身的失效检测机制、BFD失效检测机制、快速收敛机制、
- 影响OSPF邻居建立的参数，Network Type、MTU、Authentication、Key 、ACL、FLAG
- 影响BGP邻居建立的因素，IGP、Update-Source-Interface、Authentication、TTL、Capabilities

深度学习二层交换技术

- CST/ PVST+ / RSTP/ MST Spanning Tree 的关系及进化原因
- Trill / Fabric Path / VPC / MLAG 如何高效实用冗余物理链路
- VxLAN / EVPN 大二层技术的产生、演化、成熟的历史过程

深度学习QoS技术

- Coloring / Re-marking / Policing / Shaping / Rate Limit / WFQ / CB-WFQ / PQ / LLQ / WRED / Tail-Drop / DSCP / Commit rate / Burst Rate
- 精通QoS如何更好地为实时语音、TCP流量提供端对端的支持

✓深度学习网络安全

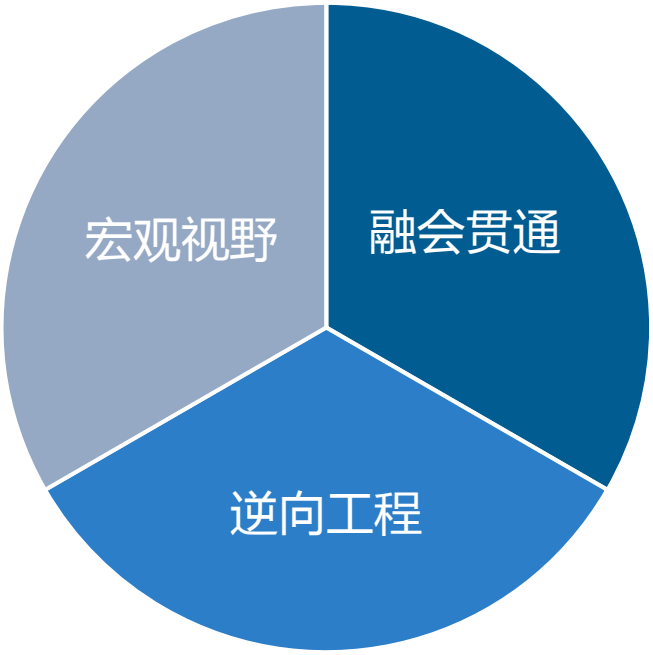
- 网络安全的六大基础组成元素：对称加密 / 非对称加密 / Hash函数 / DH算法 /RSA算法 /Nonce(盐)
- 用户认证、数字证书、数据加密、数据鉴权离不开前三大基本元素。
  无论IPsec / TLS / SSH 等安全技术基本上是这这些技术的组合。

中级修炼之学习内容

深度研究TCP协议

- [ ] •Three-Way Handshake
- [ ] •Four-Way Disconnect
- [ ] •Half-Open
- [ ] •Half-Close
- [ ] •Delayed ACK
- [ ] •Nagle 算法
- [ ] •Retransmit Timer
- [ ] •Persist Timer
- [ ] •Round Trip Time
- [ ] •Duplicate ACK
- [ ] •Fast Retransmit
- [ ] •Fast Recovery
- [ ] •Selective ACK
- [ ] •Scaling Window
- [ ] •Authentication Option
- [ ] •Maximum Segment Size
- [ ] •Maximum Segment Life

高级进阶



融会贯通

- 能用网络安全知识解释令牌认证、指纹认证、瞳孔认证、身份证、随机码、短信认证、盐。

逆向工程

- 根据一个黑盒子的外在表现，能够大体推测出其内部实现。

宏观视野

- 能够洞悉网络存在的缺陷，提前予以应对。

一个人的精力、经历是有限的，而网络场景是无限的，一个人不可能熟悉所有的设备、协议，但是凭借清晰而正确的概念，可以快速地学习、掌握、并精通它！

高级进阶之融会贯通示例

一个防火墙有三个feature

- ACL
- NAT
- Encryption

在出方向上，谁最先执行、谁最后执行？

Encryption 肯定是最后执行的，否则，一旦加密别的feature 看不到payload 的明文，无法看到端口，如何NAT？

如果NAT在最前，那么ACL处理的就是替换后的IP、端口号，这不是ACL的本意，ACL的本意是针对客户的原始IP 的！

顺序如下：

ACL → NAT → Encryption

高级进阶之逆向工程示例

知乎有这样一个问题，三层交换机路由口和svi的区别？
拿思科三层交换机来说，有switchport 和no switchport 区别。但是在华为早期的交换机要配置ip，只能用interfere vlan 。
交换机和路由器三层互联思科普遍用no switchport 配置ip，华为undo portswitch很多交换机不支持，和路由器互联普遍用interfere vlan ，所以想问这两种接口区别。

No switch-port 是路由模式。Switch-Port 是桥接模式。

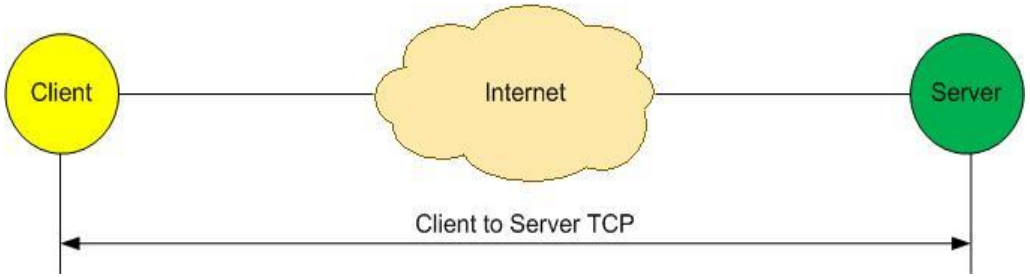
桥接模式
如果桥接模式下，没有配置 interface VLAN (SVI)，那么这个交换机在此VLAN下，完全是一个数据的旁观者，因为反正目的MAC都不是自己的，只是查询MAC Address Table帮别人做二层转发服务。
如果配置了interface VLAN 并配置了IP，那么凡是从属于此VLAN的端口上接收到的帧，首先要检查目的MAC是否自己的，如果是，则路由处理，如果否，做二层交换处理。

路由模式
此模式和路由器接口没有任何差别，主要检查目的MAC = myself ，如果是，路由处理，如果否，忽略处理。

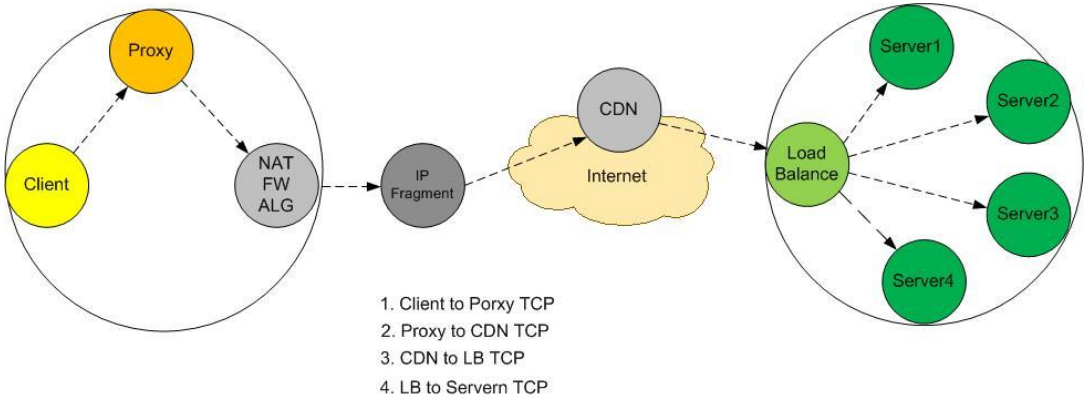
这就是一个逆向工程，根据交换机的行为，推测其内部代码逻辑。

高级进阶之宏观视野示例

理想中的Client与Server的TCP通信如下图所示：



现实中的Client与Server的TCP通信却如下图所示：



在client与 server中间也许存在有很多个中继路由器，但这些路由器只基于IP包目的IP的转发，这里只有一个client 到 server 的end-to-end TCP连接。

这里一共产生了4个TCP连接，涉及到以下元素：

- *NAT、IP Fragment 、Stateful FW、ALG、Proxy 、CDN、Load Balance*

造成了本来的一个end-to-end TCP数据传输，变成了4个hop-to-hop TCP连接串联起来的数据传输，*由于牵扯到很多元素，一个IP包要经历过多次的改写、覆盖、替换、分片、缓冲、重组，然后再不断重复这个过程，一旦出现连接不上、或性能下降，排查的节点多，需要一段段排查才能找到root cause。这需要前瞻性的视野，能够提前预见到这种复杂的局面，在网络部署时能够充分了解应用程序的特定需求，调节网络参数，避免出问题时再补救。*

计算机网络之学习方法

类比法

- 类比法，就是将相似的东西，放在一起研究，一方面当想到其中之一时，会自然联想到其它的，另一方面，通过比较可以找出不同点，不同点才会让你真正地记忆住某件事物。

顺藤摸瓜

- 顺藤摸瓜其实就是打破沙锅问到底，凡是在看书的过程中，遇到不理解的，追根溯源，直到问题得到答案为止。

类比法示例 – 校验

每当我看IP Checksum时，都会自然联想到TCP/UDP/ICMP的checksum，甚至Ethernet CRC，我会问自己一些问题:

第一个问题：CRC与checksum的算法一样吗？

第二个问题：为什么二、三、四层都需要校验？

第三个问题：checksum与CRC各自所覆盖的范围是什么？

第四个问题：AH与ESP HMAC覆盖的范围是什么？

特别是第三、第四个问题，如果你想计算机网络学习上一个台阶，从今天开始，把最后两个问题弄得透彻!

类比法示例 – 校验

问题1：CRC与checksum的算法一样吗？

Checksum名字已经说明了一切，check summary，中文意思是校验和，将一串0、1二进制流，按照16位为一个word，将这些 word简单相加，即得到checksum，这是一种**线性计算**。

Checksum = $X_1 + X_2 + X_3 + ... + X_{n-2} + X_{n-1} + X_n$

如果把 X_1与X_n互换位置，checksum值不变，换句话说，checksum检测不出数据已经变化了，所以这是一种**弱校验**。

CRC是一种**非线性计算**，比如 $Y = X_1 + X_2^2 + X_3^3 + ... + X_n^n$

如果把 X_1与X_n互换位置，Y很容易检测出，**CRC是强校验**。

但是CRC校验也不是100%能检测出数据错误,其可靠性只有 $(1 - 2^{-n})$

以32位CRC为例，其可靠性无限接近100%，即使出现小概率误判，也无关紧要，必经三层、四层还有校验，甚至应用层还有MD5校验，可以将数据误差控制在极其渺小的概率。

类比法示例 – 校验

问题2：为什么二、三、四层都需要校验？

一串0、1数据从源到达终点的过程，会从源的应用层（用户进程）出发，被copy到TCP/IP协议栈（内核系统进程），再被copy到硬件接口（驱动程序），最后被发送到链路上。

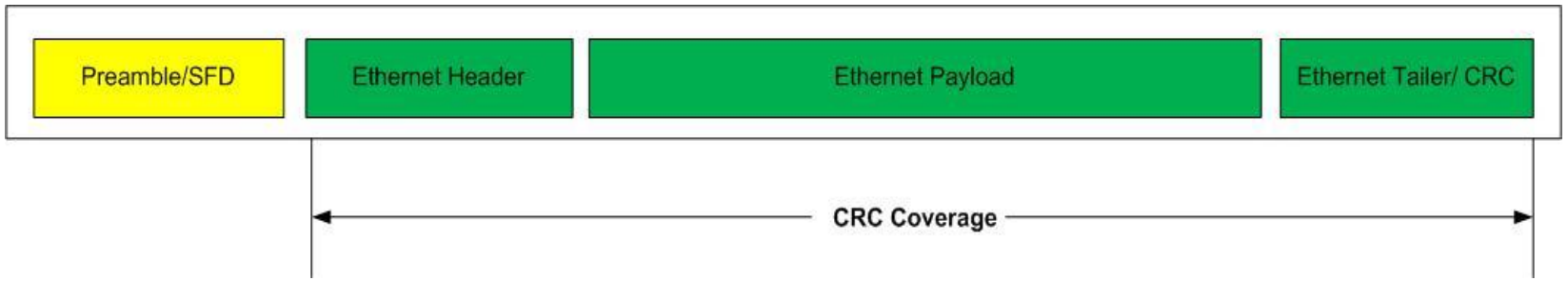
沿途经过N多的交换机、路由器、NAT、防火墙，这些设备会对数据做copy动作，同时还会改写二、三、四层的某些字段，如TTL、Source IP 、Source Port、Destination IP、Destination Port， 甚至七层的数据被改写，从而引起IP长度的变化，进而引起TCP/UDP长度的变化，如何能保证这一系列的操作不出差错？

最好的方法就是每一个独立的进程，实现自己的校验方法，主动校验错误，而不能用别人的错误来惩罚自己。

类比法示例 – 校验

问题3： checksum与CRC各自所覆盖的范围是什么？

如图所示，绿色部分为CRC所覆盖的范围，而Preamble/SFD并没有被包含在内，如何可以检测出？



Preamble/SFD是一种硬件的同步信号，一共8个字节，二进制的表示为：10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011，既然这是一个已知值，如果出错了，接收端很容易知道是否出错了，一旦同步信号出错，放弃接收随后的二进制流。

在计算CRC时，需将接收到的帧CRC先copy出来，再清0处理，然后计算自己的CRC，与copy值做比较，一致则校验通过。

类比法示例 – 校验

问题3： checksum与CRC各自所覆盖的范围是什么？

IP Checksum覆盖范围

Ethernet为了保证数据的最大限度地可靠传输，对自己的Payload长度也有一个上限、下限的限制，即 46 ≤ Ethernet Payload ≤ 1500

刚刚谈到那个长度为44字节的IP报文，显然没有满足下限的要求。

在尾部再添加2字节不就可以了吗？可以，但是46又无法满足4字节的整数倍，所以最好的方法是添加4个字节到48字节，双赢的节奏。

添加得很欢乐，但是IP报文的最终的接收者能否知道哪些是IP Payload，哪些是人为添加的IP Tailer?

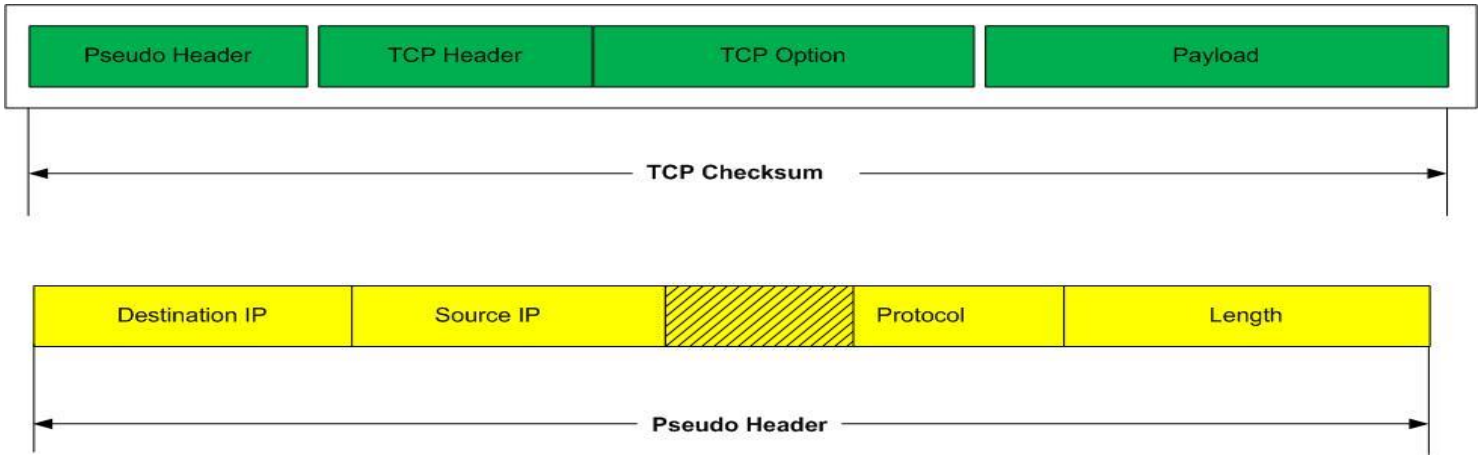
这个不是什么难题，因为IP header里有两个长度字段，一个是 Head Length，另一个就是Total Length，根据前者可以确定IP header的长度，根据后者可以确定Payload的长度，剩下的全是尾部添加，可以丢弃处理。

类比法示例 – 校验

问题3： checksum与CRC各自所覆盖的范围是什么？

TCP Checksum覆盖范围

绿色部分为TCP覆盖范围，所以TCP不仅仅覆盖自己的势力范围，还覆盖本该属于IP的字段Pseudo Header，那这个Pseudo Header是如何得到的？



黄色填充的部分为Pseudo Header，这些信息都是从IP Header得到的，一共12字节，由于Protocol只有一个字节，用1个字节填充，填充值为0，另外length = IP Total Length – Head Length。

需要强调的是，如果这个IP报文途径一个NAT设备时，做了IP地址、端口号的替换，则需要重新计算TCP、IP的checksum，否则会校验失败。

类比法示例 – 校验

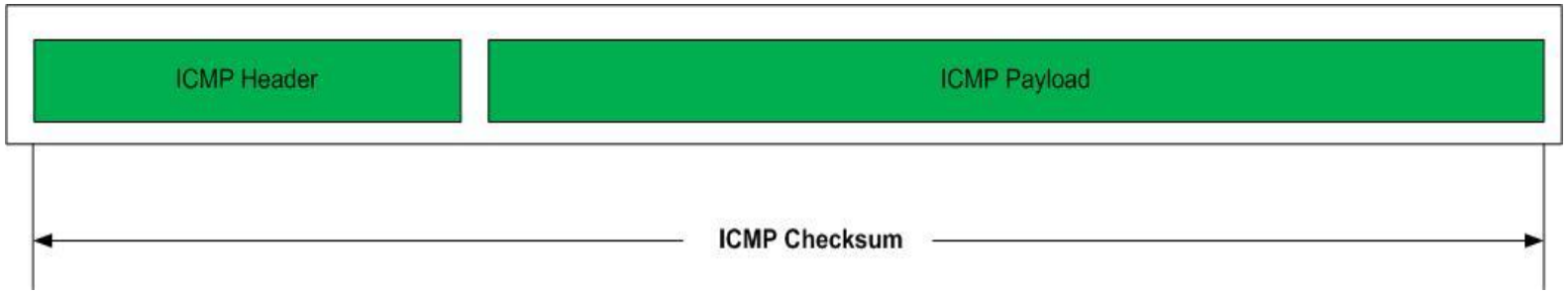
问题3： checksum与CRC各自所覆盖的范围是什么？

UDP Checksum覆盖范围

UDP与TCP的计算方法相同，唯一的不同的是，TCP checksum是强制的，而UDP checksum却是可选的，如果接收到的UDP checksum则认为发送方没有做UDP的校验。

ICMP checksum覆盖覆盖范围

ICMP的checksum计算没有什么特别指出，这里罗列出来仅仅是为了比较，只有比较才会记住各自的特征。



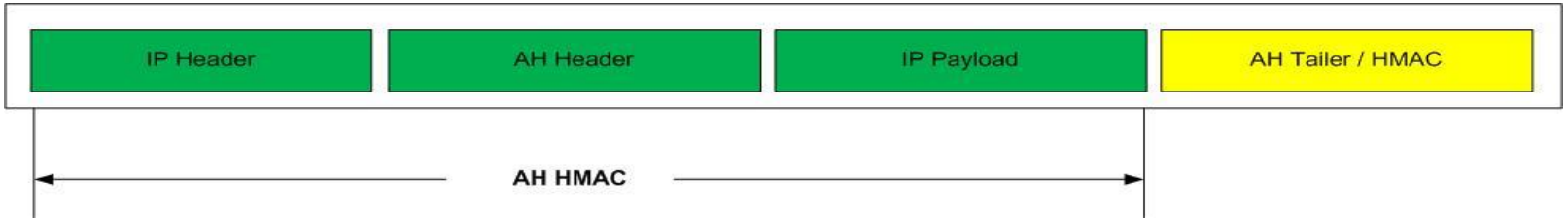
类比法示例 – 校验

问题4： AH与ESP HMAC覆盖的范围是什么？

AH(Authentication Header)

一种安全协议，用于通信双方对传输的数据DATA进行认证，发送方将图片中的绿色部分做鉴权处理，

HMAC = HASH (Green DATA , KEY)，将此HMAC附在报文尾部，见黄色部分。



如果此报文途径NAT设备时，NAT修改了IP地址，并重新计算IP checksum，那修改后的IP报文到达终点时，如何处理？ 丢弃处理！

虽然终点通过IP校验，但是却无法通过AH 的校验，因为终点结算的AH HMAC与接收到的不一致。

那NAT 设备可以重新计算AH HMAC吗？不可以，因为NAT设备不知道KEY是多少，无从修改。

AH就是为了避免end-to-end的数据被任何中间设备改写，所以提供了KEY进行了保护。如果被修改了，也失去了该协议的初衷。

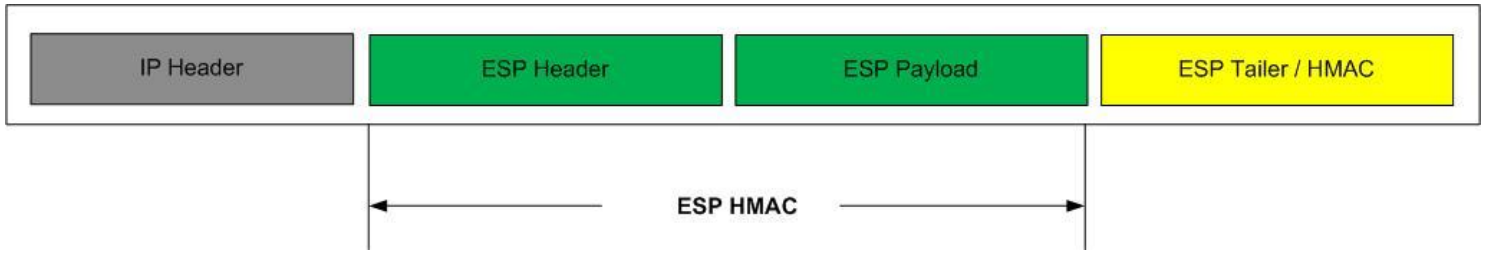
传输模式下的AH与NAT设备不能兼容工作！

类比法示例 – 校验

问题4： AH与ESP HMAC覆盖的范围是什么？

ESP (Encryption Security Payload)

如果此报文途径NAT设备时，NAT修改了IP地址，但还需要修改被ESP加密的TCP端口号，由于NAT不知道加密/解密的key，无法修改，即使强行修改，到达终点通过IP校验，但是却无法通过ESP 的校验，因为终点计算的ESP HMAC与接收到的不一致而丢弃。



传输模式下的ESP与NAT也是不能兼容工作的。

类比法示例 – 树

学习计算机网络的过程中，经常能遇到树这个概念。现在来回忆一下有哪些树？

- 二层交换机的spanning tree
- 路由协议OSPF/IS-IS使用的SPF算法树
- 组播里的基于RP点的RPT树、或基于组播源的SPT树

观察这棵树，闭上眼睛，冥想这棵树的特征。

我观察到一个特征，树呈发散状，没有环路，根部从大地吸收的水分，会顺着跟流到树干、树枝、树杈、叶子，不会出现水在某一处转圈圈，否则树的上部因为缺水而枯死。

这些协议恰好也是利用了树不环路这一点特征，只要保证自己的逻辑拓扑，像一颗树，那么就可以避免二层、三层流量的环路。

类比法示例 – 树 Spanning tree

二层交换网络，交换机之间的连接，杂乱无章。

二层交换机对于广播帧、组播帧、未知单播帧的处理，是泛洪处理，所谓泛洪，就是洪水泛滥的意思，哪里有路，洪水就往那里流，很显然，如果二层交换网络物理拓扑上有环路的存在，泛洪的水会一直在这个封闭的物理环路里流动。

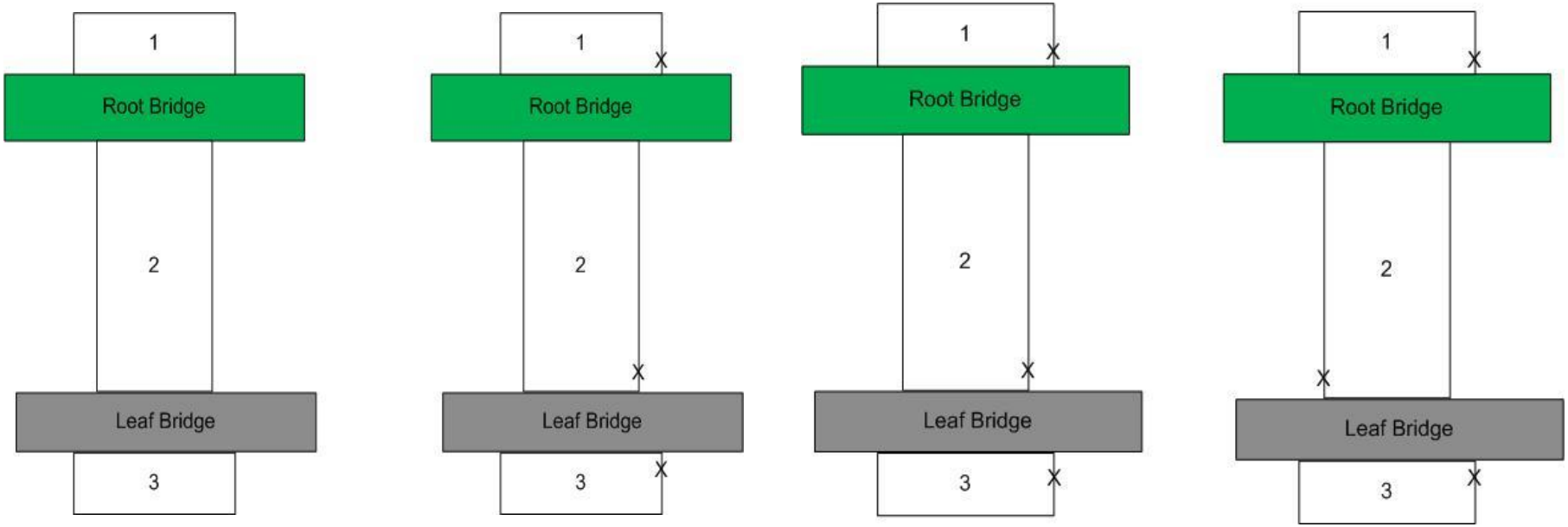
Spanning tree 的核心思想是：把存在物理环路的网络，通过spanning tree协议协调成一个逻辑上没有环路的网络，所谓逻辑无环路，就是将次优的冗余链路逻辑断，逻辑断的意思是：虽然物理依然连接，但其实和无连接没有本质区别。

类比法示例 – 树 Spanning tree

用4根网线将root bridge, leaf bridge自环、互环，形成了编号为1、2、3的3个环路，需要spanning tree 将三个物理环路逻辑断开成右侧的拓扑，这样就没有封闭的环路

相比物理不连接的优势是：当最优物理链路断时，次优物理链路会立马满血复活，让网络重新收敛成一个全新的树。

当区域2的左侧链路断，右侧的链路则满血复活，形成一个新的树。



类比法示例 – 树 Spanning tree

如果大家完全明白spanning tree的核心思想与意义，那么接下来就只剩技术细节问题。

无非就是如何选择Primary root bridge, second root bridge, designated port, root port, non-designed port。

对应成树的概念就是谁是树跟，谁是备份树根。其它的都是小树叶，当树根的自然要能力强的，位于网络的核心，流量的中枢，所以只能选择核心交换机做树根，另外一个备份核心交换机做备份树根，树根的所有端口都是designated。

对于叶子来说，选择一个最靠近树根的树杈做root port 也是顺理成章的事。如果洪水从树根流向叶子，那么树根处于最上游，叶子的树杈（root port）则属于叶子的上游接口，那么如果两片叶子中间有交织（物理链路），则比较两片叶子谁距离树根近，谁近谁做designated, 谁远谁做 non-designated，non-designated端口需要自裁，将自己的端口逻辑切断。

依此方式，每个叶子都找准自己的定位与选举，如果整个过程完成，则二层交换网络收敛，所谓收敛，就是网络的所有设备完成同步，对网络逻辑拓扑的状态达成一致。

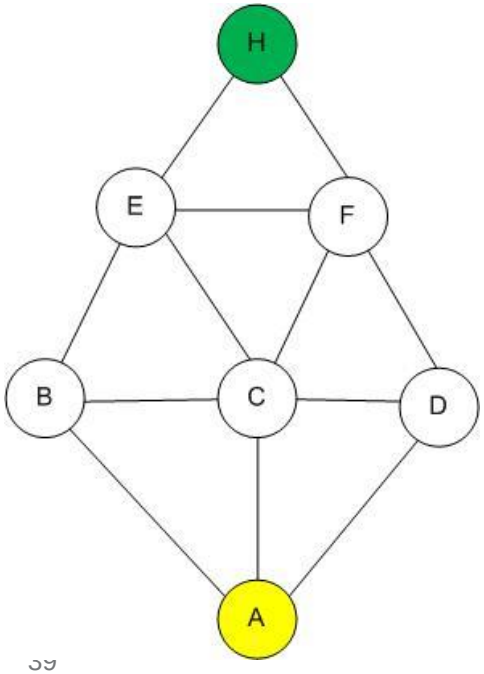
额外说一句，如果选择一个能力弱的叶子做树根，流量能撑死它，所以万万不可！

类比法示例 – 最短路径树 OSPF/ISIS

7个圆圈代表的是路由器，假设每条链路的cost =1，现在路由器A计算到H的路由表。A ----→ H

以A为树根，H为叶子，罗列所有的链路组合，剔除出所有环路的组合，然后在剩下的无环路组合里选择整体cost 值最小的，视为最优路由，进入路由表，如果有多条cost值相同的路径，则共同进入路由表。

还有N多组合，但整体cost 一定大于等于4，所以A-H之间最优的路径的cost 值=3，一共有四条路径，所以最终进入路由表的条目为4。

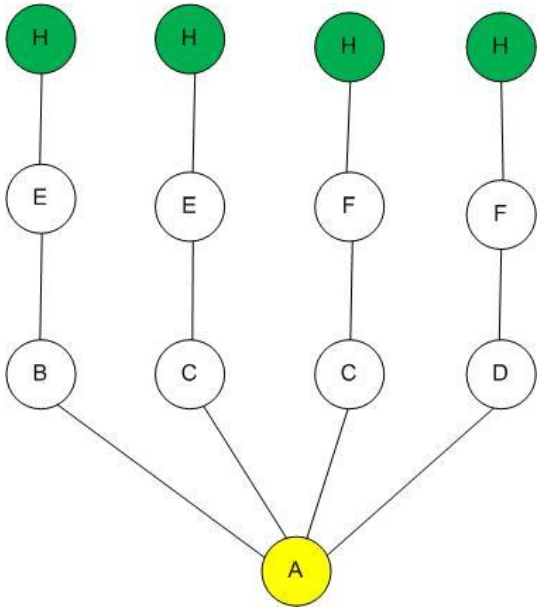


路径组合:

A-B-E-H (3)
A-C-E-H (3)
A-C-F-H (3)
A-D-F-H (3)
A-B-C-A (环路)
A-C-B-A (环路)
A-C-D-A (环路)
A-D-C-A (环路)
A-B-A (环路)
A-B-E-F-H (4)
A-B-E-C-F-H (5)
A-D-C-E-H (4)
A-D-C-B-E-H (5)
……

路径组合:

A-B-E-H (3)
A-C-E-H (3)
A-C-F-H (3)
A-D-F-H (3)



类比法示例 – 最短路径树 OSPF/ISIS

三层路由转发不使用二层泛洪机制，而是查询路由表，按照目的IP与路由表条目的最长匹配原则，获得下一跳，然后发给下一跳，直到到达目的地。

所以三层流量转发能否形成环路，全依赖于路由表的神勇，如果路由表没有环路，那么一切ok，你好我好大家好。

如果万一路由表环路了，IP包要在封闭环路的绕圈圈了，直到TTL减到0为止。

路由表是从哪里来的呢？路由协议！所以**路由协议的防环机制**至关重要。

根据人类的常识，当一个人从A地出发，目的地是H，可以任意选择 B、C、D做下一站，但是到达B、C、D时，肯定会向上走，而不会后退向下走，也不会朝着左右的方向走，因为向上走才更靠近H，而如果向左右走，则意味着绕路，而向下走则意味着南辕北辙，永远到达不了目的地。

既然有4条路由进入路由表，如何将A-H流量分担到4条路径上呢？这又牵扯到负载均衡算法了，而负载均衡一般支持

- packet-based

严格地将所有流量按照round-robin 的方式平均分摊到每条路径上。

- session-based

将一个session 的所有流量映射到相同的路径上，这样可以保证session 报文顺序到达，顺序到达有什么好处？ 或者换句话说，乱序到达有什么坏处？ **这将牵扯到另一个关于TCP的话题。**

路由器如何决定哪些IP包是属于一个session 的? **这又将牵扯到另外一个关于HASH的话题。**

类比法示例 – 组播树Multicast

无论是二层交换网络、还是三层路由网络，最终都要收敛成一个树。

组播使用树的概念，其核心思想是想，让组播源当树根，组播接收者当树叶子，流量只能从树根流向叶子，而不是相反！

组播源是上游，接收者是下游，对于每个路由器来说，明确知道哪个接口更靠近上游(根据单播路由表)，定义为上游接口，其它的一律为下游接口。

上游接口接收到的流量是合法流量，我们称这种通过RPF(Reverse Path Forwarding)检查的流量为合法流量。

下游接口接收到的流量则没有通过RPF检查，则丢弃。

依此准则，避免组播流量的环路，这就是树的精髓。

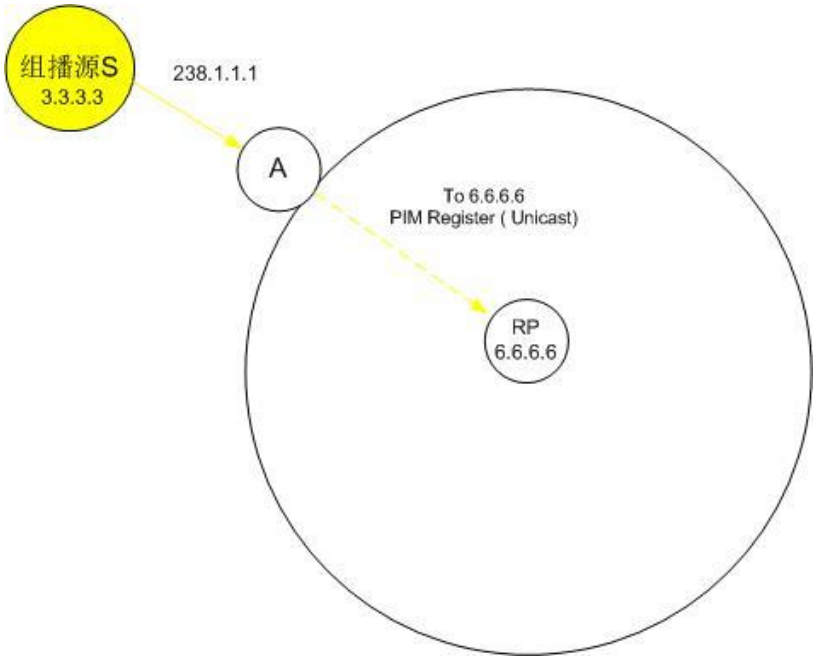
类比法示例 – 组播树Multicast

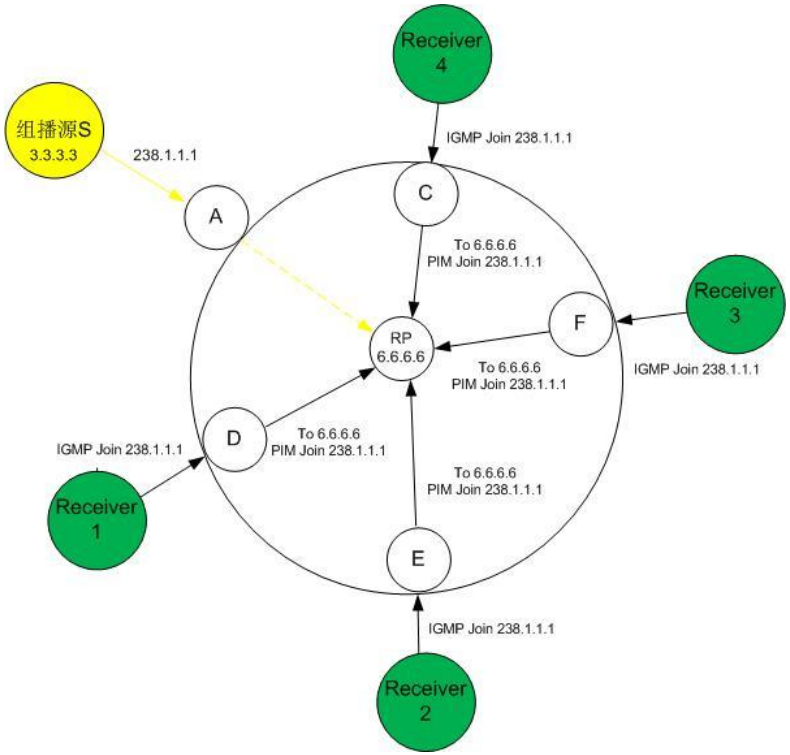
我们先来看看这个组播源(238.1.1.1)的数据是如何到达组播接收者的？

1. 组播源S 向自己的硬件接口发送238.1.1.1,源IP =3.3.3.3

2. A接收到该组播，深深地知道，需要将该组播通过单播的方式**注册(Register)**到RP处，RP相当于房产中介，A相当于准备卖房子的客户，A想让中介RP物色感兴趣的买家（Receiver）

3. RP接收到该组播，查询自己的 (*，238.1.1.1)表，看看有没有下游接口OIL（Outgoing Interface List），假设为暂时为空，那么RP就默默地收到的组播丢了。

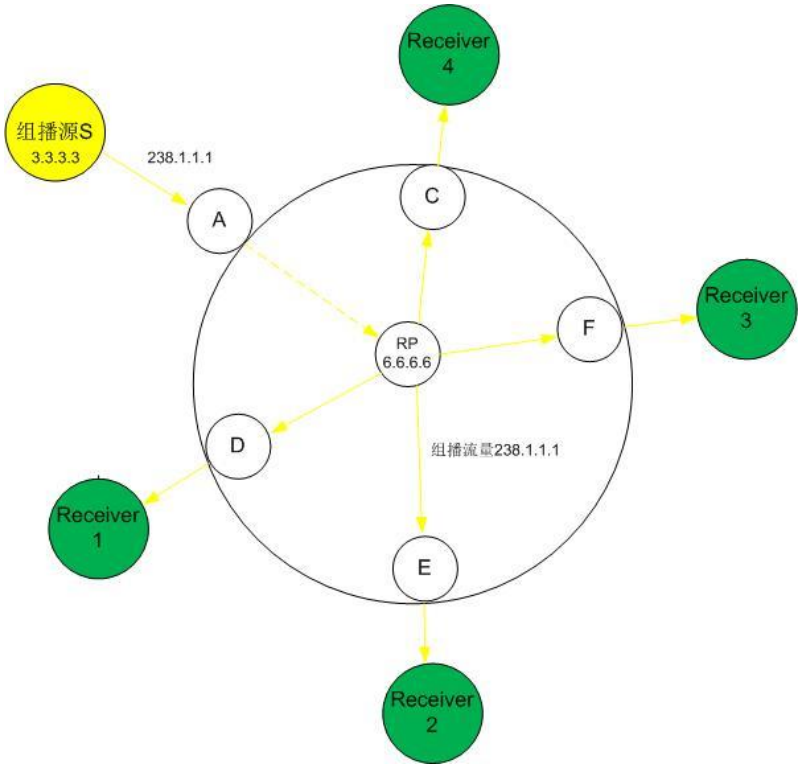


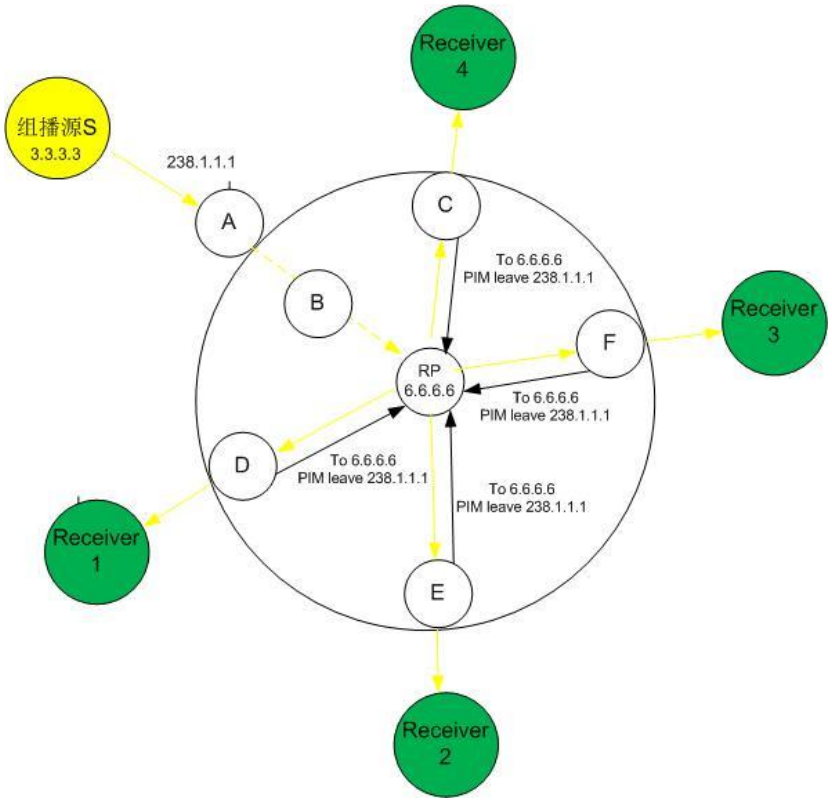


类比法示例 – 组播树Multicast

4. 假设4个接收者提交了加入238.1.1.1组的请求，然后其直连的路由器通过PIM消息，联系RP， 希望自己加入238.1.1.1，然后RP 将这个四个接口放入OIL，并将流量向4个OIL接口进行复制转发。

5 组播流量顺利到达4位接收者，这时路由器C、D、E、F不安分，因为接收到组播数据了，知道源在哪里了（3.3.3.3），希望自己直接加入以源3.3.3.3 为树根的树，我们称之为SPT, 并决定从以RP为树根的RPT树退出来。



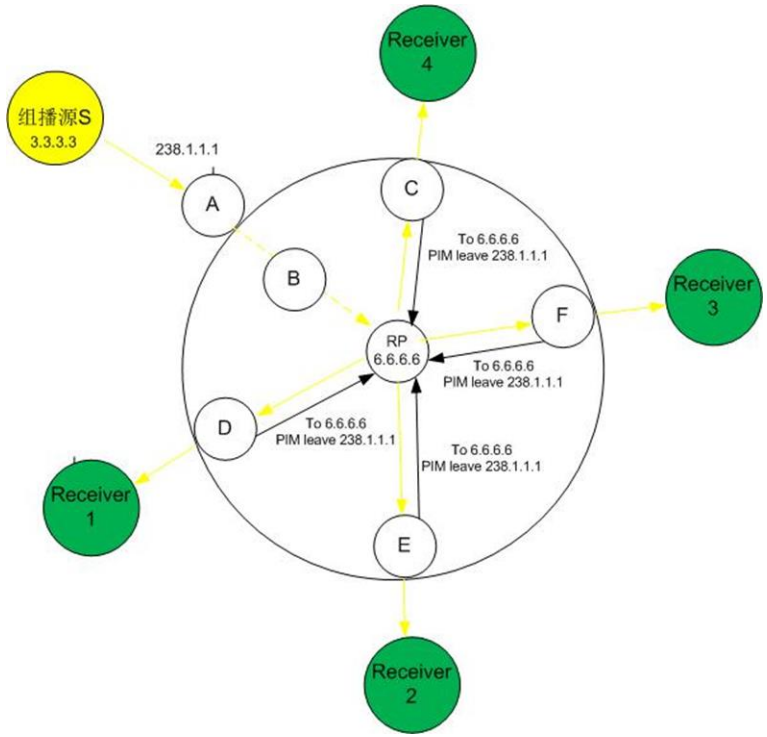


类比法示例 – 组播树Multicast

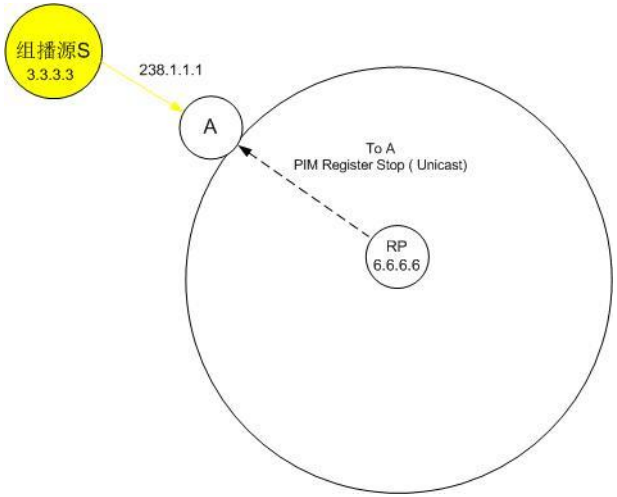
以D为例，查询单播路由表，发现到源3.3.3.3 的上游是A, 于是给A 发PIM 消息洽谈自己加入的意向，A欣然同意，并将直连D的接口放入自己的下游接口OIL。

其他的路由器按照D 一样的思路纷纷加入以A为树根的SPT树。

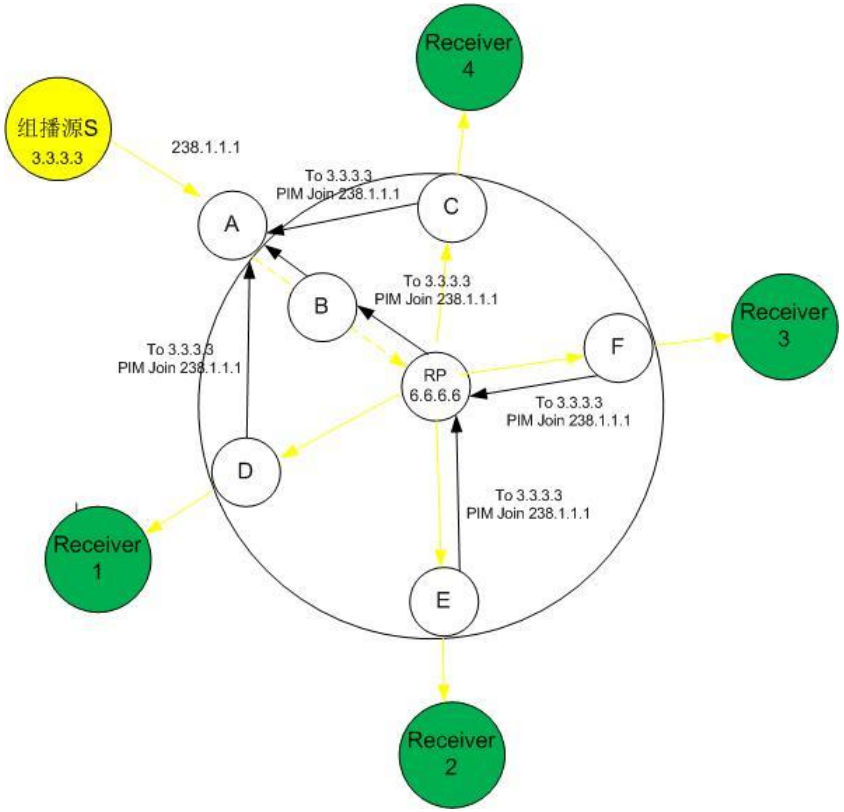
同时发消息给RP, 纷纷要求散伙，RP丝毫没有觉得脸挂不住，欣然应允。



其实RP早早地、偷偷地也加入了SPT，并发register-stop 消息给A，意思是不再劳烦您老人家，我自己已经知道你在哪里了，并已经通过SPT树接收流量了，所以请您老休息吧，A怅然所失地点头同意了。



类比法示例 – 组播树Multicast



如果把组播源3.3.3.3比喻成一棵树的树根，其他Receiver比喻成叶子，把组播流量比喻成水分，水分顺着树根，单向流向叶子，按照这个树状的发散结构，不会发生环路。

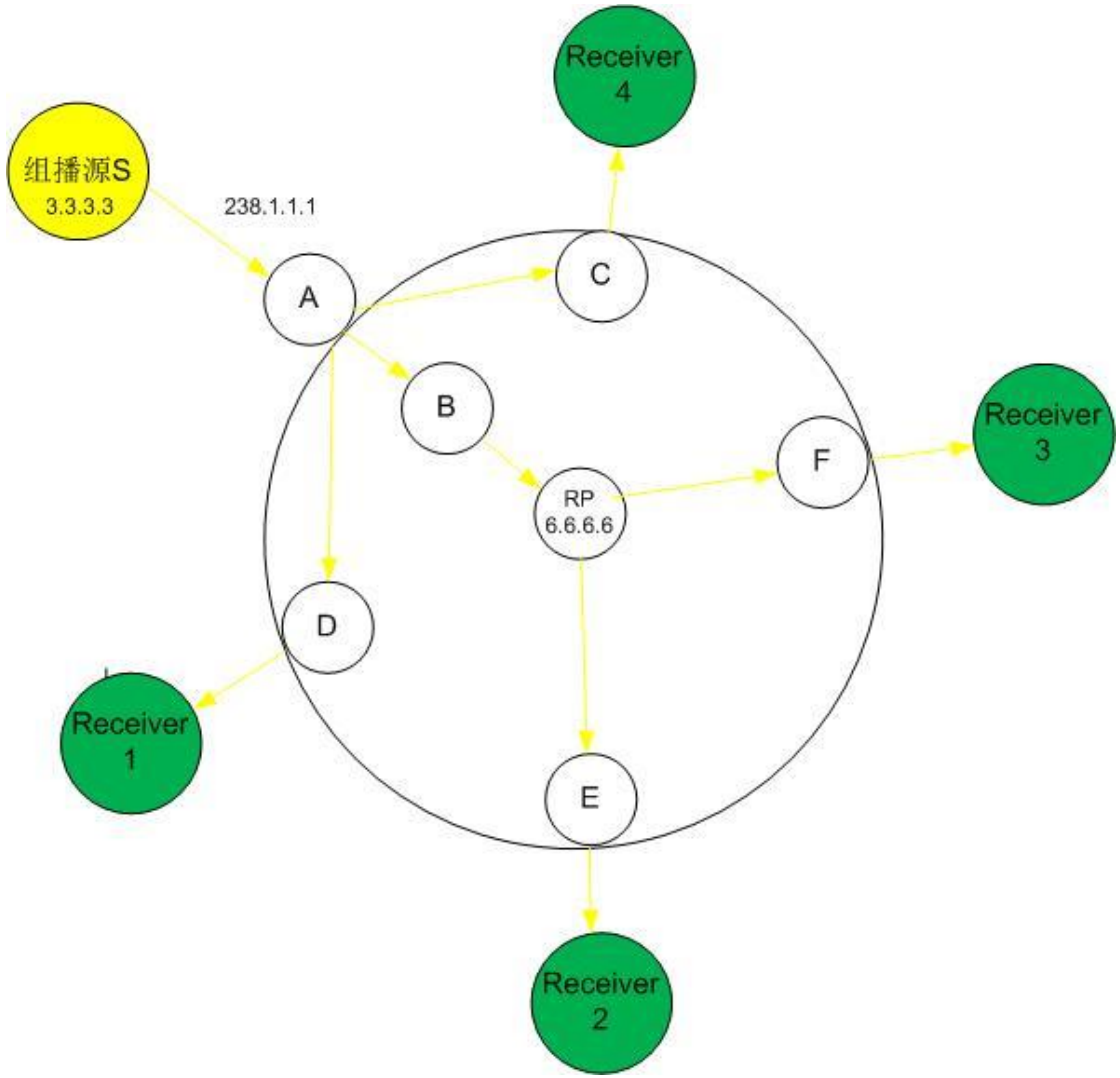
其实对于图中每一个节点来说，最重要的就是明确谁是自己的上游邻居，与上游邻居直连的接口就是上游接口，上游接口是水管的入口；同理，还需要明确谁是自己的下游邻居，与下游邻居直连的接口就是下游接口，下游接口只能是水管的出口。

如果从下游接口接到水，那么可能发生环路，需要将水立马丢弃处理。

对于每一个节点，有且仅有一个入口，可能有0个、1个、或多个出口。

类比法示例 – 组播树Multicast

新的组播SPT树生成了



类比法示例 – 组播树Multicast

什么是RPT? 为什么一个组播需要RP这个角色?

拿生活中的场景来举例：

司机老王想卖房，但是不知道买家在哪里，于是需要一个房产中介机构做为一个信息交换的平台，老王去中介处挂牌（register），隔壁张木匠需要买房，也不知道谁要售房，于是跑到中介处搜集房产信息，恰好看到老王的房产信息，于是中介从中撮合，安排老王与老张见面，老王与老张趁中介不注意，偷偷交换了手机号码，至此之后，老王一脚踹开中介，老王与老张单线联系，最后达成交易。

来谈正题。

组播场景下，凡是需要RP存在的，都是组播接收者无从知道组播源在哪里！

RP作为一个临时的信息交换平台，让组播接收者能够获得组播源的IP地址信息，在这个临时的特殊时期，一样需要保证组播没有环路，那么就让RP临时担任树根，凡是对组播有兴趣的接收者则充当叶子的角色，我们称这种树根是RP的树为RPT（RP-based Tree)。

在这个临时的特殊时期，还有一件特殊的事情一直在发生着，那就是组播源直连的路由器A一直用单播隧道的方式，将组播数据包裹在单播隧道里，传输给RP，RP解封装，得到原本的组播源，并将组播源向自己的RPT复制扩散。

类比法示例 – 组播树Multicast

什么是SPT?

一旦叶子接到组播数据，组播的Source IP清清楚楚地写明 3.3.3.3，与叶子直连的节点希望加入组播源3.3.3.3 为树根的树上，因为这些节点认为可能距离源3.3.3.3比RP(6.6.6.6)更近，延迟会更小，我们称这种以组播源IP为树根的树为**SPT (Shortest Path Tree)。**

同时还需要从RPT树上退出，否则一个节点既在RPT上、又在SPT上，会接收到组播的两个copy，这显然没有必要，更是网络资源的浪费！

一句话结束这个主题，树是用来干嘛的？ 防环（Loop-Free）！

顺藤摸瓜法

在读RFC或白皮书的时候，经常会遇到一些疑问，而这些疑问作者限于篇幅无法深入展开，只是告诉你结果，没有告诉你过程，这时如果不去细究背后的前因后果，可能学习只停留在表面，而要做到深入学习，必须去带着这个疑问去阅读相关的文献，目前我有两个疑问：

疑问一：

乱序对TCP有什么影响？

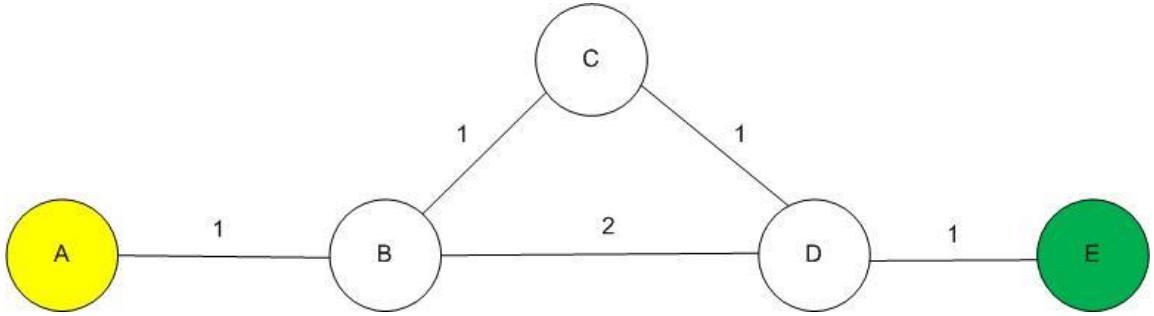
疑问二：

路由器如果判定哪些IP包属于一个session？

顺藤摸瓜其实就是打破沙锅问到底，凡是在看书的过程中，遇到不理解的，可以查询维基百科，维基百科的英文版写作质量很高，遇到不理解的，可以在线查询关键词，然后看文章的时候，还会遇到不理解的，直接点击关键词的链接，可以一直看到没有任何障碍为止。

遇到一时无法理解的，也不要灰心气馁。客观理性面对这种认知规律，对于知识的探究是螺旋式的上升过程，一时理解不了，就先放一放，过一段时间也许因为理解了相关的技术细节，那些不理解的知识点会豁然开朗。

顺藤摸瓜法示例 - 乱序对TCP有什么影响？



A 到E有两条等价路由：A-B-C-D-E (cost = 4) ， 路径1; A-B-D-E (cost = 4)， 路径2

从A发出的IP报文应该是这个样子的：1、2、3、4、...9、10

按照严格的负载分担的原则: 路径1 分担IP报文1、3、5、7、9...; 路径2 分担IP报文 2、4 、6、8、10...

由于路径1多一跳，正常情况下路径2会比路径1早到达节点E，E接收到的IP报文顺序应该是：2、1、4、3、6、5、8、7、10、9。然后提交给TCP，也是这个顺序。

当TCP接收到2时，知道乱序了，其期望接收1的，那把2丢了吗？ NO！缓存一下，结束了吗？ No! 还需要发送一个ACK给对方，告诉对方，我要收1，同时还要把 advertised window 减少一个报文的长度，A接收到了E的ACK消息，会重传1吗？ NO！假装什么都没有发生，只是把自己的发送window 减小一个报文的长度。

少顷，报文1到了，TCP需要排序好，顺序为1、2，通知应用程序前来取走，假如取走了，这时需要ACK对方，准备接收3，同时更新自己的advertised window到最大

顺藤摸瓜法示例 - 乱序对TCP有什么影响？

按照这种类似的节奏完成数据的缓存、警告对方、排序、确认的动作。

而如果没有乱序，当E接收到报文1时，会发ACK吗？一般不会，因为TCP/IP协议栈现在是收到两个报文才确认一次，紧接着就是通知应用程序取走数据，当2到达时，通知应用程序取走数据，同时ACK对方，由于缓存的数据都被取走，所以advertised window 一直是最大的，这样A就可以按照最快的速率来发送数据。

所以当乱序的发生时，会让通信的双方有更多的动作，更多的消息交互，会降低TCP的数据传输效率。

所以作为IT基础架构的网络层，需要意识到乱序对TCP、应用层的影响，最大程度避免乱序，让TCP字节流按序发送并提交给对方。

至此，各位对什么因素影响TCP的性能有强烈好奇心了？是延迟、丢包率、还是乱序?

到底哪个因素对TCP性能影响最大？

延迟是如何造成的，是路由器缓冲队列太深了、还是网络发生了拥塞？

丢包率增大，是无线的丢包还是有线的丢包？无线丢包是由于信号干扰还是AP无法处理过多的用户请求？ 有线丢包是由于网络拥塞还是线路质量还是硬件接口出故障、或者QOS人为丢包？

乱序是如何造成的？除了这里提到的场景，比如路由器在缓冲队列时，小包总是优先于大包处理，那么意味着小包可能会后发先至，这样也会造成乱序。

等把这些问题都研究透彻，就可以从容地回答一个经典的面试问题：用户抱怨网络慢，你觉得是什么原因造成的？

顺藤摸瓜法示例 -路由器如果判定哪些IP包属于一个session？

很多人都知道，对于TCP/UDP，使用五元组，

Destination IP + Source IP + Destination Port + Source Port + TCP/UDP

Destination IP	Source IP	Destination Port	Source Port	Protocol	Interface_ID
6.6.6.6	3.3.3.3	80	65233	6	0
5.5.5.5	2.2.2.2	25	1056	6	1

表中是两个不同的session ，被负载分担到两个不同的物理接口上。

但是这个方法比较原始、笨拙，对于一个IP包需要一个一个字段来比对，需要五次比对，然后还需要查表来决定用哪个物理接口。如果条目不在表中，还需要建表。

最有挑战的还需要维护一张这样的映射表，对于一个大型的路由器来说，每秒几十万、甚至几百万次IP包的转发来说，这张表是非常庞大的，耗费内存资源不说，查表还需要大量的计算资源，所以这不是一个好办法。

有没有更好的办法？

顺藤摸瓜法示例 -路由器如果判定哪些IP包属于一个session？

哈希（HASH）表!

HASH是一个函数，将TCP/UDP五元组作为输入，将会产生一个输出，这个输出称为HASH值。

HASH ID = HASH (Destination IP , Source IP , Destination Port , Source Port , TCP/UDP)

那么以HASH ID的不同来映射不同的接口ID。

HASH ID	Interface ID
DX43422346432566	0
TEF34XWFEGV37T81	1

假设以上的两个session 经过hash变换，得到的HASH ID，然后做一个映射表，分别映射到不同的物理接口上。

对于接收到的IP包，先提取出五元组，然后做HASH变换，得到HASH ID，进行查表动作，进而选择对应的接口ID。

但这个方案依然需要维护一张表，有没有不需要维护表的方法？

顺藤摸瓜法示例 -路由器如果判定哪些IP包属于一个session？

能否根据HASH ID的个位数，假设为M，让M去除等价路由的数目，得到的余数为Q

Q = M mod N

如果余数Q为0，则对应接口ID =0; 如果余数Q为1，则对应接口ID =1

HASH ID “DX43422346432566”的个位数是6，等价路由为2

则M=6, N=2 ➔ Q = 6 mod 2 =0 ➔ 所以映射接口ID =0

HASH ID “TEF34XWFEGV37T81”的个位数是1，等价路由为2

则M=1, N=2 ➔ Q = 1 mod 2 =1 ➔ 所以映射接口ID =1

假如现在有16个等价路由，则 N=16。

通过此种算法，可以避免维护一张表，但是唯一的不足是，可能由于HASH ID个位数值分布不均，很难做到完全的负载均衡。

顺藤摸瓜法示例 -HASH函数

在计算机领域，经常听到Hash这个词，Hash 是一个函数，无论输入值有几项，或输入值有多长，其输出长度总是固定的、输出值是恒定的、而且根据输出量很难计算出输入量。

Y = Hash （ X1, X2, ... Xn)

Hash 函数有以下特点：
输出量Y长度固定
N为输入量个数，Xn长度任意
只要X1, X2, ... Xn 固定，输出量Y就是恒定的、唯一的。

B = Hash （A），如果两次计算得到B1 = B2
那是否意味着输入量 A1一定等于A2？

答案是：不一定，数学家无法保证A1就一定等于A2

但，这种不相等概率无限接近为0！换句话说，相等的概率无限接近于100%。

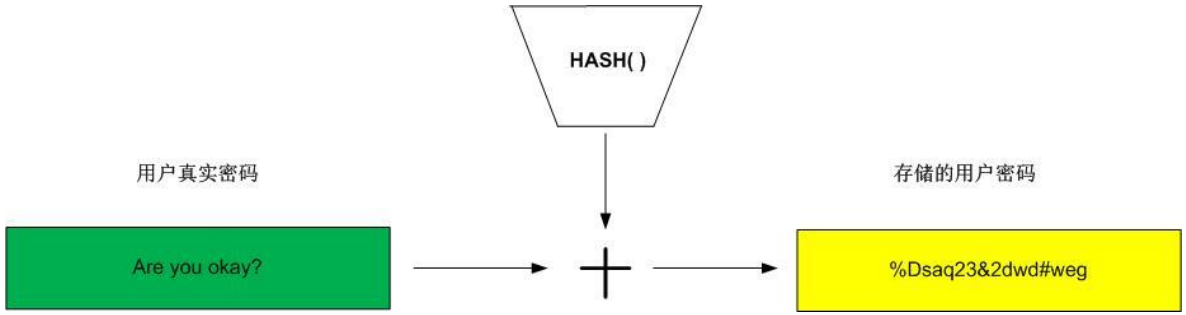
所以应用在工程上，就是如果两个Hash值相等，则意味着输入量相等。

顺藤摸瓜法示例 -HASH函数

密码加密

这种密码加密的方式通常用于OSPF / CHAP，以CHAP为例，使用MD5 Hash算法。

客户端将用户输入的明文密码P ，生成 Hc = MD5 （ Pc ）



然后将这个Hc发给服务器，服务器存储密码时，已经将用户密码P做了一次

H = MD5 （ P ）

于是服务器从自己数据库里取出这个H，如果Hc = H，则验证通过，否则认证失败。

这种方法避免
1）用户密码在网络上明文传输
2）服务器以明文方式存储密码

顺藤摸瓜法示例 -HASH函数

破解密码加密

上面没加盐的认证，很容易受到Hash库的比对攻击。

用户密码组合极限为 10个数字 + 26 小写字母 + 26大写字目 + 10多种特殊符号（算13个），一共有75种选择，那么长度为10的密码极限组合为：75^10 = 5.6^18次方，然后将这些组合的密码预先，分别计算出其MD5值，保存在彩虹表里。

图例中，在网上传输的用户密码是“%Dsaq23&2dwd#weg” ，被第三方捕获到，与自己的彩虹表一一比对，比对到了，同时就找到了其对应的密码。

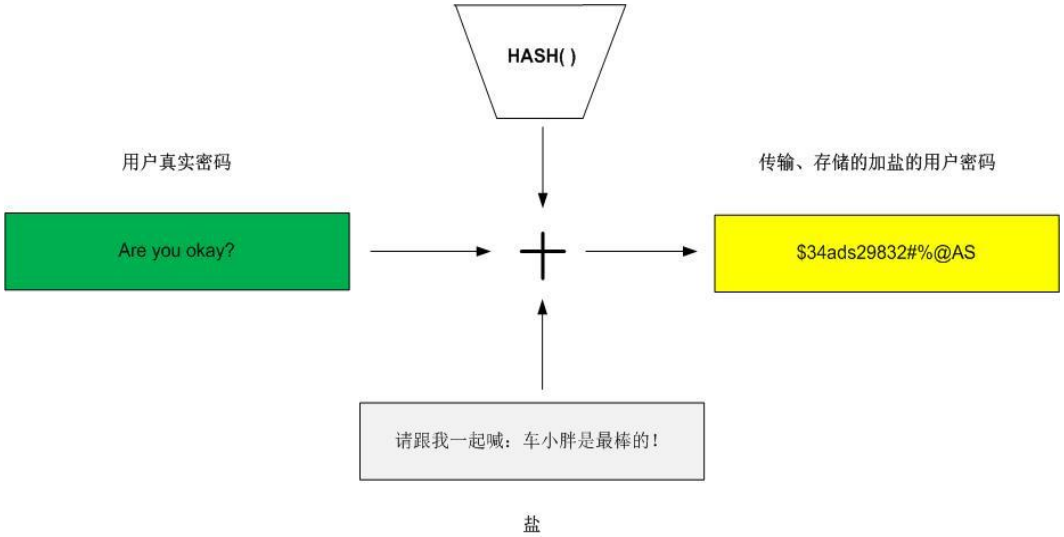
序列号	明文	密文
12121121	Are you okay?	%Dsaq23&2dwd#weg
12121122	I am no okay	?sdqasqwq#DHUy0%
…	…	…

所以，不加盐的密码加密有被破解的可能。

顺藤摸瓜法示例 -HASH函数

加盐的密码加密

当认证用户时，认证框上有服务器提供的随机码S(salt)，当用户做Hash时, Hc = MD5（Pc，S）



服务器也采用相同的计算方法，得到H，然后再比较两个值。

由于第三方预先并不知道随机码是多少，所以无法预先计算加盐的彩虹表，破解难度大大提高!

这个随机码就是盐！

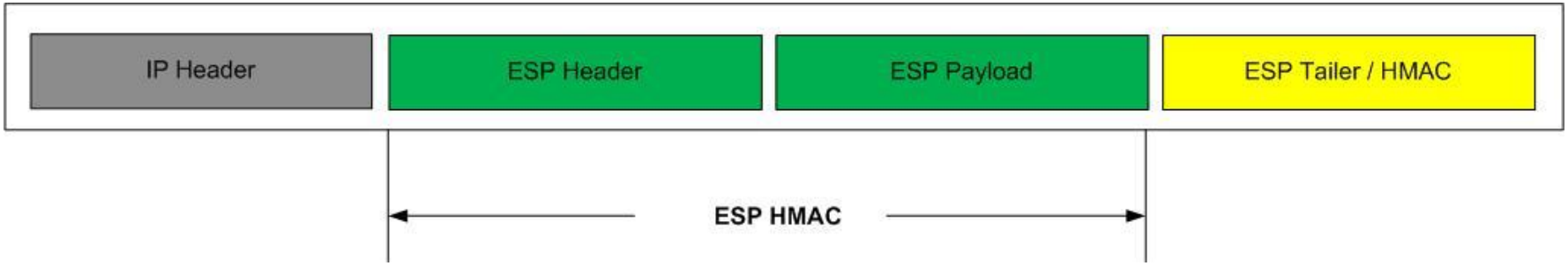
顺藤摸瓜法示例 -HASH函数

数据完整性保护

加密隧道传输的数据需要至少提供两种服务：

数据机密性（Confidential）

数据完整性 （Integrity）



图中打马赛克的部分为对明文数据加密的密文，但是如果没有完整性保护，第三方尽管无法读懂明文数据，但是可以任意修改这个密文，到达目的地不一定能够检测出来数据已经被修改。。

可以将绿色的部分用MD5/SHA等HASH算法，生成一个HASH，然后再用 HASH KEY 将这个HASH值进行加密，即得到黄色的部分ICV(Integrity Check Validation)，附在报文的最后方。

由于HASH key是通信双方动态协商出来的，只有他们双方知道，任何第三方都无法知道HASH key，也意味着无法知道HASH值是多少，即使修改了任何绿色覆盖的字段，都会在终点检测得出，并丢弃处理！

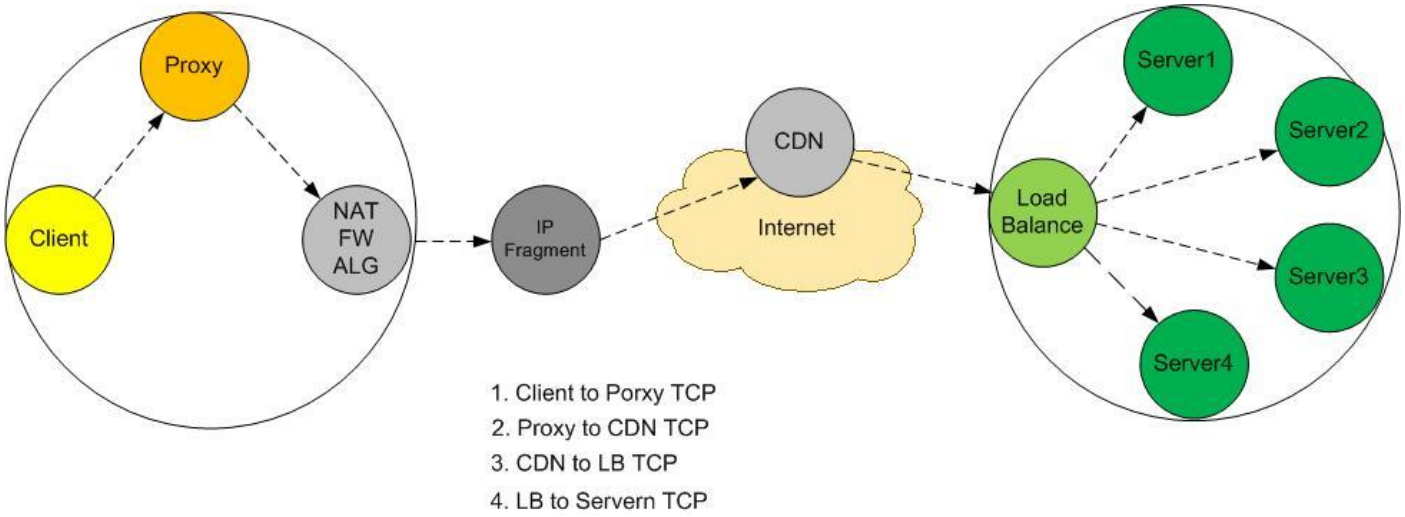
顺藤摸瓜法示例 -HASH函数

文件完整性保护

在cisco网站上下载一个IOS 文件，通常会在页面上有一个MD5值，需要用户将下载的文件，用md5 check 命令来计算出一个MD5值，如果和网页上的MD5值完全一样，你的文件是完整的！

如果不一致，很抱歉，你下载的文件发生了错误，需要重新下载。

Happy Ending 干货收场



Client可以与server 建立连接，但无法下载图片或上传大文件。

- 罪魁祸首 MTU、NAT次凶、防火墙是帮凶、ALG是小跟班

IP Fragment 发现IP包过大，而IP包却不允许分片，于是发ICMP type 3 给client，但FW不让过，丢了，TCP连接超时断

FW 让ICMP type 3 过了，client 发小包一路畅通，没有问题了。

来自服务器端的图片到达IP Fragment， 需要分片，IP包也允许分片，于是分成两片，一片有端口号，一片没有端口号，防火墙不让没有端口号的过，TCP连接断

防火墙允许没有端口号的IP分片过， 包顺利到达client, 没有问题了。

Happy Ending 干货收场

有些应用程序如SIP在应用层嵌入IP地址，如果IP包不分片，NAT做IP + 端口号的替换，调用ALG来做应用层嵌入的IP地址的替换。

但一旦分片了，ALG无法看到应用层的全部数据（也许数据分散在两个IP分片里），ALG替换失败，造成TCP连接断。

如果NAT可以做IP分片的重组，可以得到应用层的信息，然后调用ALG，完成替换，okay, 完美解决。

还没有完，如果应用层加密了，无论重组不重组，NAT都无法获得应用层的明文数据（密文），就无法做ALG，通信是一定会出问题的。

通过以上分析，会发现一个IP分片会造成多大的麻烦！

同时得到一个结论，如果应用层加密，最好不要嵌入IP地址、端口号信息！

如何学好计算机网络？

Thank You!

Live PPT 会共享给参与者