计算机网络实验

-PROMISE-

Monday 28th March, 2022

1 概述

使用 pthon 实现一个设备功能，在 Mininet 构建的网络中运行自己的设备，用 Wirsen_packeteshark 进行抓包分析，借助 SwitchYard 实现

• Mininet: 用来创建一个包含主机交换机控制器和链路的虚拟网络

• Wireshark：截取网络封包，并尽可能多的显示数据包内容

• Switchyard: 用于创建、测试和测试网络系统的软件实现，威斯康辛大学的教学框架，包装好了一些包的结构和在网络中收发包的函数，我们需要设置合适的参数去构建合适的包，利用收发包函数实现计算机网络的工作

• 总体工作，用 python 实现交换机、路由器、可靠传输和防火墙的实现逻辑，借助 Switchyard 提供的函数和 API 进行构建，并且用 wireshark 工具抓包检查实现结果

2 Learning Switch

• basic

提取以太包头的源地址和目的地址，对于源地址记录在自己的 mac 和接口的对照表当中，比如改包源地址为 mac1，进入的端口为 interface1，则在记录表中记录这一项。对于目的地址，和自己的 mac 地址比较，相同说明收到了这个包，不同在列表中寻找，找到了在对应端口发出，否则进行 flooding，除了进入的端口，向其他所有端口泛洪 (发送 ARP 包询问)。

• timeout

用 time 记录当前时间，每次收到一个新包的时候，存储接口信息时记录当时的时间，之后每次收发包都对当前字典中的内容进行一次超时判

断，并且清除超时的内容

• LRU

这时候我们需要注意 Recently used 的情况，所以不能直接用字典实现存储了，而换成有序的 OrderedDict，如果 table 已满，删去最早的信息，如果信息已存在，删去原有信息并且重新插入，这样可以保证 table 中的顺序是按照使用时间排序的

• traffic

将替换标准替换为频率，设计两个字典，一个记录频率，一个记录接口 mac 对应的数据，当需要替换的时候，先在频率字典当中找到频率最低的项，然后在 table 中删除

• ping

进行 ping 操作，利用 wireshark 抓包可以查看 switch 的工作情况

3 IPv4 Router

3.1 Respond ARP

• 只需要判断包是否有 ARP 头就可以判断是否是 ARP 包，对于 ARP 包，针对与自己相关的包进行回复，构造一个 ARP 回复包，提取请求包中的 mac 地址和 ip，添加自己的 ip 地址，设置 ARP 属性为 reply，并且发送

• 构建路由器的 ARP 缓存，记录 IP 和 Mac 的对应关系，以 ip 作为索引，timestamp 和 mac 作为 value 值存储，table 用字典实现，timestamp 和 mac 构成一个类，当目的地不是全 0 的时候，说明包含了 IP-Mac 对应信息，进行存储，其中已经存在的也要存储，用于更新 timestamp。每次收到包的时候都对 table 进行一次超时删除处理

3.2 Forwarding Packets

• 实现路由器的转发功能

首先从 txt 文件中读取初始的 table，包括前缀、子网掩码、下一跳地址等信息。

对于收到的包，首先根据转发表找到对应的下一跳地址，如果下一跳地址为 None，说明可以直接到达该 ip，否则我们会得到下一跳的 ip（注意这里要找到最长匹配的子网），根据 ip 地址在 cache 中找到对应的 mac 地

址，将包发送出去，如果 ip 地址对应的 mac 在 cache 中找不到还要涉及到收发 ARP 包获取 mac 地址的处理

实际上通过转发表找到的下一跳信息就是该包的管辖范围，找到最长匹配就是找到最小的管辖范围，可以提高效率

我们处理找不到 mac 地址的情况的时候，用一个队列保存当前找不到 mac 的包，把他们存储下来，并且在收到 ARP 回复包的时候对队列中的包进行处理。队列中存储的信息包括数据包本身的信息，它对应的 ARP 请求包，时间戳，请求次数等信息，我们实现的是在五次发送 ARP 包没有收到 reply 的时候丢弃这个数据包。不管有没有收到包，路由器持续对队列中的数据包进行处理，对于 ARP 请求次数小于 5 且等待上次 ARP 请求的回复信息已经超时的情况进行重新发送 ARP 请求的处理。

路由器收到 ARP 请求包的情况和 switch 类似

3.3 Respond to ICMP

•ICMP request

对于 request 进行回复，构建 reply 包（ping 指令项网络主机发送 ICMP 回显请求的分组，判断网络是否正常）

对于几种错误情况，依据参考文件设置参数构建 ICMP 的错误信息包，并且发送

利用 traceroute 进行抓包，可以查看 ICMP 包的正确性

4 Reliable Communication

• 对 Switchyard 中更高层次的部分进行处理，Switchyard 中涉及的是 blaster 经过 middlebox 发送一个包 blastee。我们在 middlebox.py 的文件中用取随机数的方式模拟丢包的情况

从 start_mininet.py 文件中获取 blastee 和 blaster 的接口、ip 等信息，用来之后构建包的时候使用

• blastee

文件中只需要对收到的包构建 ack 包并且发送就可以

• blaster

文件中，设置发送包的窗口，当没有收到包的时候，对于窗口信息进行更新并查看数据包是否发送完毕。否则对收到的包进行处理

对于发送包的存储，需要记录几个参数，send 表示包是否发送，ack 代表是否确认，发送时记录 timestamp 一边判断是否超时

对于窗口信息，用 lhs 和 rhs 模拟窗口的实现，lhs 是当前第一个以发送还没有收到 ack 的包序号，rhs 是最后发出去的包序号，当 lhs 到序列最后的时候，发送完毕，lhs 和 rhs 之间是当前窗口中的包。

收到包的时候，对对应的包进行 ack 设置，并且更新 lhs 和 rhs，如果包已经 ack 则 lhs 右移一位，代表该包已经成功被接受

没收到包的时候，路由器持续对窗口进行处理，获取当前可发送包的序号，如果有新包可以发送，就返回该包的序号，如果窗口已满，返回-1，说明有包丢失，lhs 指向的包需要重发。

除此之外，比对 lhs 指向的包和当前时间判断是否超时，如有超时现象也需要重发（即使窗口未满）

5 DNS

• 读取 dns 文件，并且对输入的域名进行解析

CNAME 项表示别名，替换为另一个域名，A 类对应到 ip 地址，对应多个 ip 地址的时候，采用负载均衡的算法，选择一个 ip 地址，这里我们采用的方法是用 IP_Utils 的的工具得到 IP 的地址，返回最短路

6 Http 请求

• 实现 cache，当收到请求的时候，先在 cache 中查找（key 是对应的路径 path），如果不存在，则向 mainserver 请求，用提供的函数获取 HTTPResponse，获得 body headers 信息存入 cache 并返回。

•Get 和 Head

利用上面的 touchitem 函数获得对应的 headers 和 body，返回给 client，否则调用错误处理函数，返回错误包给 client（404）

Head 只要返回 headers 就行