LINUX网络实用知识

田志鹏 2018.12.06

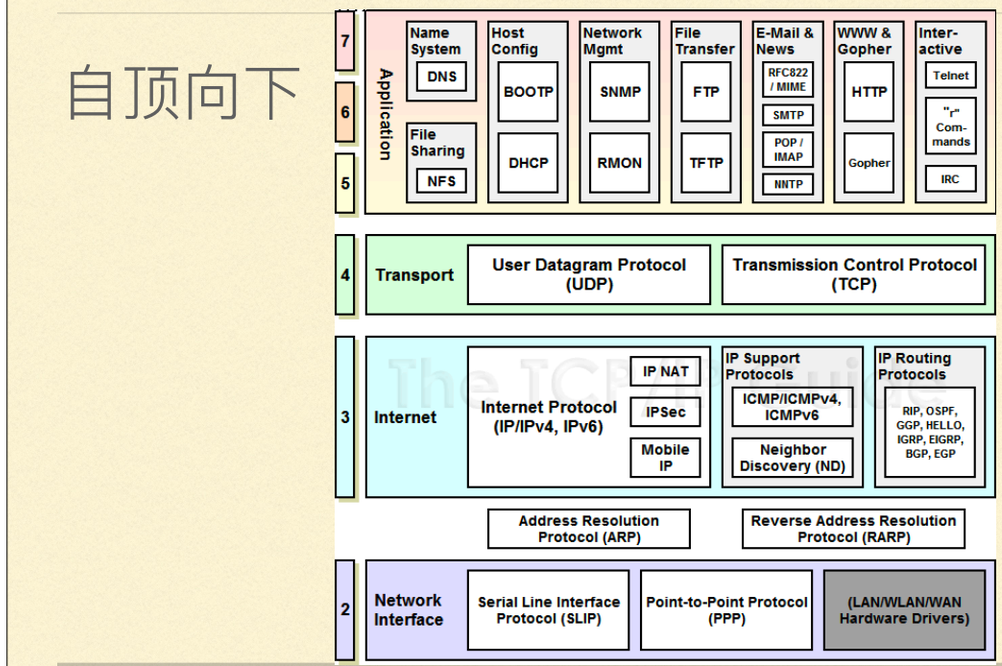
目录

- 应用层问题分析
- 传输层状态查探
- 网络层IP和路由
- 使用场景实例

今天主要讲的就是linux中和网络有关的命令, 将他分为三个部分 ... 然后也会讲一些示例和使用场景, 会穿插着讲, 不会完全从前到后的顺序.

每个命令man一下
test11启动8666
h31启动16884
2297

自顶向下



本来是不想讲这个分层的事, 但是想不到更好的方式把内容组织起来, 所以还是先过一下这个

应用层问题分析 DNS

- libresolv库 : /etc/nsswitch.conf (~~/etc/host.conf~~)
- /etc/hosts : 本地固定存储的域名映射信息 (/etc/networks)
- /etc/resolv.conf : 配置可用的nameservers
- dig (nslookup) : 向nameserver发起dns请求
  - 实例: dig @8.8.8.8 google.com

Domain name service 域名解析服务
linux系统默认安装一个libresolv库用于dns解析功能, 像java里面用到dns也是使用这个库
nsswitch.conf就是这个库用到的一个配置文件
dns请求默认是基于UDP的

应用层问题分析 HTTP

- curl 远超想象的神器
  - DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP
  - url参数, body参数, cookie, header, 保存文件(替代wget) (文档)
  - 实例: curl -vv http://127.0.0.1:16884/hello --header 'Host:example.com'

curl支持的协议很多, 但我们主要用于调试http问题
支持http里面各种能想到的需求

应用层问题分析

- DNS和HTTP以外的不再详述
- SSH SFTP SNMP NFS POP3

传输层状态查探 NETSTAT

- netstat -npa
- netstat -nptl
- 连接的概念, TIME_OUT过多/CLOSE_WAIT过多

这里可以看一下连接的概念, 回忆一下状态转化图. 有可能遇到的问题从这个命令可以看出UDP是无连接的

传输层状态查探 NCAT

nc/ncat 网工的瑞士军刀：操作TCP/UDP socket

- 监听 ncat -k -l 16884 < index.html
- 发请求(替代telnet测端口)(chat一下)
- 配合标注输入输出和管道拼接命令
- 代理 ncat -k -l 16884 --proxy-type http
- 转发 ncat -k -l 16884 --sh-exec "ncat localhost 16883”
- 实例: nginx转发tomcat的spring应用, 收到的url有问题.

开一个http server需要几步 使用netstat看连接

chat一下

开一个http 代理需要几步

场景curl -vv http://h31.mzhen.cn:8004/rest/v1/segments/list

网络层IP和路由 PING

- 测试目标的可达性
- ICMP(Internet Control Message Protocol)
- traceroute: 包的路由路径
  - 不断改变ip包中的ttl值, 使得路径上的主机放回发“Request timed out”

接着来到了网络层, 主要负责ip选址和路由转发. 第一个大家都知道的命令就是ping, 测试可达性, 很简单
一个我面试问过的问题, ping使用哪个端口
icmp协议, 回头看在图里的位置.
除了ping的时候: 尝试与主机的一个端口建立tcp连接, 然而没人在监听这个端口, 怎么回复?

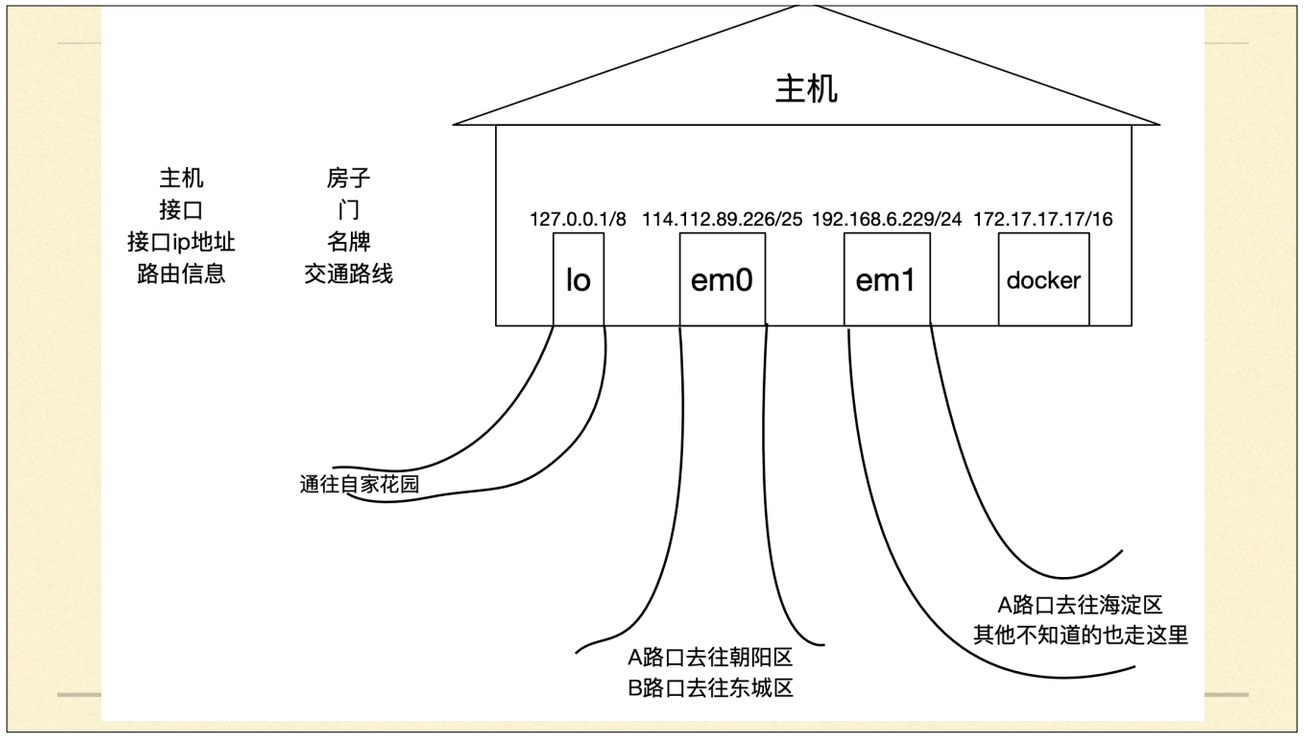
traceroute命令, 也是通过icmp协议发送ip包, 通过不断改变ttl值, 使得路径上的主机放回发“Request timed out”, 就知道路径了

网络层IP和路由 网络接口和路由

- hardware - driver - interface - (socket) 对底层网络的不断抽象
- 路由表 子网 网关

ip选址和路由转发

这块一般对应用开发的程序员不太需要, 已经配置好了


主机
主机
接口
接口ip地址
路由信息
房子
门
名牌
交通路线
127.0.0.1/8
114.112.89.226/25
192.168.6.229/24
172.17.17.17/16
lo
em0
em1
docker
通往自家花园
A路口去往朝阳区
B路口去往东城区
A路口去往海淀区
其他不知道的也走这里

网络层IP和路由 网络接口和路由

- ifconfig管理接口, route管理路由, 都可以用ip命令替代
- ip addr show up
- ip addr add IFADDR dev STRING
- ip route list scope host
- 实例: nc -lk 114.112.89.226 16884 bind外网ip, localhost访问不了(nc, netstat)
- 实例: java控制使用的网络接口

H31:
Em1 114.112.89.226/25
Em2 192.168.6.229/24 两块网卡, 分别绑定了内网ip和外网ip
jenkins:
一块网卡, 一个内网ip, 两个网关, 默认这个,
nc指定ip, 系统根据ip和路由表来定你这个socket是从哪个interface接收数据
程序也可以自己控制开个socket要用哪个接口, 但好像操作系统不推荐这么做
这就是为什么看到各种东西redis什么的, 配置里都会有bind 127.0.0.1的操作
0.0.0.0表示 '未指定' 相当于所有

网络层IP和路由 TCPDUMP

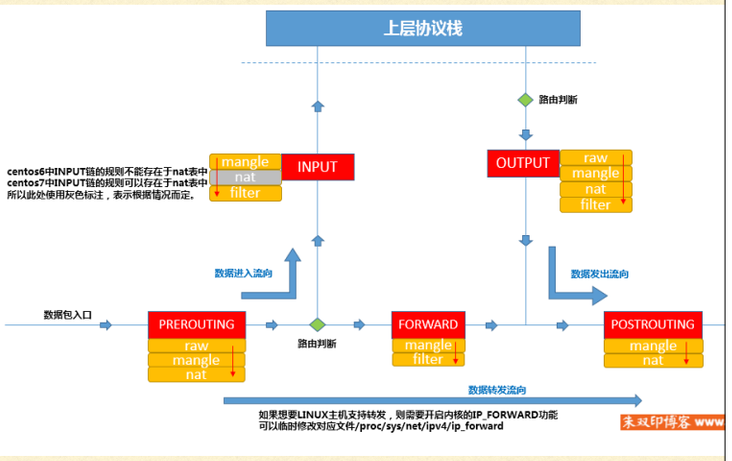
抓包分析工具: 所有到达网卡的数据包, tcp/udp/ip/icmp/甚至arp

- 参数: -i 接口 -w写文件 -n不要解析name -A ascii显示包内容 -xXX十六进制显示包内容
- 规则: 3种关键字 + 具体值 + 与或非
  - host， net， port　　src , dst　　ip,arp,rarp,tcp,udp
  - eg: tcpdump -nni em1 host 111.200.229.2 and port 8666
  - eg: tcpdump -nni eth0 icmp

tcpdump是linux下十分强大的抓包工具, 相当于通过pcap库将数据包从正常的traffic里拷贝出来一份
虽然名叫tcpdump, 实际上能抓到所有到达网卡的数据包. tcp/udp/ip/icmp/甚至arp都有
tcpdump命令分两部分
大家想看什么包. wireshark

网络层IP和路由 IPTABLES

- /etc/sysconfig/iptables
- service iptables start|stop|restart|save
- prerouting/input/forward/output/ postrouting
- raw->mangle->nat->filter
- -A INPUT -t filter -s 111.200.229.2/32 -p t -m state --state NEW -m tcp --dport 8025 ACCEPT



网络层IP和路由 IPTABLES实例

- iptables和tcpdump的顺序

入: wire -> NIC -> tcpdump -> iptables

出: iptables -> tcpdump -> NIC -> wire

- 实例: iptables关闭icmp, tcpdump能否收到

场景

在2297上无法访问114.118.13.6:8666的http服务, 望京机器可以.

- curl
- ping
- netstat
- nc telnet
- Iptables
- tcpdump

流量带宽情况

- dstat 查看cpu硬盘网络等件等统计信息, 每秒一次
- iftop 分端口分host显示2s 10s 40s内出入流量
- nethogs 可区分进程查看带宽使用
- 实例: nc发包网速, nethogs看网速
- iptraf 没试过
- iperf 没试过

划重点

- dig
- curl
- nc
- netstat
- tcpdump
- Iptables

REFERENCE

- 各个命令参考文档在前文中的超链接里
- man