1. **网络层解析**

**在系统初始化时会进行网络协议栈的初始化，**



<1>数据接收和发送

<2>Linux内核路由实现

传统路由和策略路由

路由表在内核中的组织：

路由表在内核中是以hash链的形式存在的

**<3>Linux防火墙组件**

防火墙是指设置在不同网络或网络安全域之间的一些列的部件的组合，他能增强系统内部网络的安全性。

Linux内部的防火墙组件是netfilter，它嵌套在网络层，通过hook的形式对各个节点的数据进行过滤。

**——Netfilter功能的基础是conntrack**

conntrack功能是netfilter中其他功能的基础（后面会给出解释），Linux内核只要启动netfilter功能就会启动conntrack功能，ipv4协议栈中会在PRE\_ROUTING（conntrack\_in）、LOCAL\_OUT(conntrack\_local)、POST\_ROUTING(ipv4\_confirm)、LOCAL\_IN(ipv4\_confirm)这几个节点执行相应的hook函数。在内核源码中有这么一段注释: Connection tracking may drop packets, but never alters them, so make it the first hook (nf\_conntrack\_l3proro\_ipv4.c中ipv4\_conntrack\_ops的注释)。所以conntrack是优先级的最高的hook。

就是conntrack，内核中基于tcp/ip的连接都会有一个通过src-ip/src-port dst-ip/dst-port为基础数据生成一个tuple，下面是Linux内核中定义的tuple数据结构：

/\* This contains the information to distinguish a connection. \*/

struct nf\_conntrack\_tuple

{

struct nf\_conntrack\_man src;

/\* These are the parts of the tuple which are fixed. \*/

struct {

union nf\_inet\_addr u3;

union {

/\* Add other protocols here. \*/

\_\_be16 all;

struct {

\_\_be16 port;

} tcp;

struct {

\_\_be16 port;

} udp;

struct {

u\_int8\_t type, code;

} icmp;

struct {

\_\_be16 port;

} dccp;

struct {

\_\_be16 port;

} sctp;

struct {

\_\_be16 key;

} gre;

} u;

/\* The protocol. \*/

u\_int8\_t protonum;

/\* The direction (for tuplehash) \*/

u\_int8\_t dir;

} dst;

};

网名词络篇

1. 路由

Linux内核如何实现路由的

1. Linux内核从2.1开始支持策略路由
2. 虚拟网卡

配置：

Interface 静态配置

1. 网关 route ifconfig
2. 子网 如何将本机作为网关
3. Interface
4. 防火墙

Linux防火墙的演变

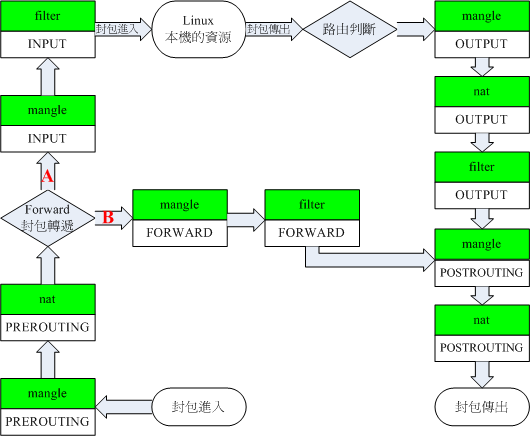
1>简单包过滤

2>第四代应用程序netfilter/iptables

TLs

1. NAT的起源和发展 rfc1631

一个常见的变种是负载均衡，在一组机器上做映射。如果你要进行严格的比例限制， 可能需要参考Linux Virtual Server。<http://linuxvirtualserver.org>

[](http://d.hiphotos.baidu.com/zhidao/pic/item/a08b87d6277f9e2f5c08900e1b30e924b899f3f3.jpg)

1. 代理
2. Vmware 虚拟交换机
3. 网桥
4. 无线网络标准（802.11b工作在2.4GHz频带、802.11a工作于%GHz频带[和b/g标准不兼容]、802.11g工作在2.4GHz、802.11n双频模式可以工作于2.4GHz和5GHz）
5. Netstat/arp 命令的操作层次
6. 区分策略路由和FW

策略路由是制定路由表，默认的路由表选择顺序是

fib\_validate\_source 检查是否从期望的网卡接收的数据包

1. Sk\_buf

Old linux sourcelist set in sourcelist.txt

Skbuff.h

Sk\_buf和是否面向连接没有关系

Sk\_buf如何实现数据分包 ip头部提供分组和重组功能

Linux内核空间和用户空间;

Linux简化了分段机制，是的虚拟地址和线性地址总是一致，因此Linux的虚拟地址空间0~4G。Linux内核将这4G字节的空间氛围

消息队列 管道 共享内存 ==》 内存拷贝次数

**Linux RCU机制**

**http://blog.csdn.net/lickylin/article/details/38326719**

Rpfilter

Netdevice.h struct net\_device{}

1. 内核路由表实现

路由表 fib\_table 🡺路由表id、以及相关路由表的操作函数 路由表查找、添加、删除等

Fn\_hash 用掩码长度未hash值 管理路由

Fn\_zone 将所有掩码相同的路由都存在同一个fn\_zone中 结构是fz\_hash

/\*fib\_node结构变量的个数，fn\_node并不能说是对一个路由项的抽象，可以看成对一个

目的网络地址的抽象，而对于相同的目的网络地址，可以根据tos、priority、mark值创建

不同的路由，所以一个fn\_node结构中fn\_alias才能算作是对一个路由项的抽象\*/

Fib\_node为一个目的的网络地址抽象出相同的基础结构

Fn\_alias 是已存在的路由项的链表

Fib\_alias 这个路由项的路由部分参数不同

Fn\_key是fib\_node对应的目的地址

1. Socket

(1)socket和文件节点

Linux下一切皆文件，那么网络设备是如何被虚拟问文件，如何实现网络数据的收发和文件的读写对应？

通过一个数字来表示socket==》系统调用通过sockfd\_lookup将数字映射成一个socket结构体，这个结构体保存了该socket的所有属性和数据==》所有的socket都是基于BSD socket，socket是在虚拟文件系统上创建的，可以看成文件，可以被安全的扩展==》socket中有state变量用来表示socket状态（类似于状态机管理，socket未分配、未连向任何socket、正在连接过程中、已连向一个socket、正在断开连接过程）

(2)socket的调用都是通过io\_ctl实现的，在系统中的调用流程

Io\_ctl会触发软中断，调用内核中的对应的匹配函数 socket所有的调用都是经过这个流程的

(3)系统是如何记录socket应该调用哪个函数发送，socket的状态

Socket中AF\_INET或UNIX等等参数会传递进来

Netfilter的基础功能是connection track

Netfilter用“来源地址/来源端口+目的地址/目的端口”（ 一个tuple）来唯一表示一个连接，用一张连接跟踪表来描述所有的连接状态，该表用了Hash算法。

基于connection track会生成一个tuple，可以标记一个数据包是否需要做 nat转换 所有不需要nat的连接会有一个公共的 tuple

建立规则时没有 local\_in这个规则链，但是在内核中会在local\_in做reply的转换

Nf\_nat\_in/nf\_nat\_out

Nf\_contract.h

Struct nf\_ct连接跟踪项

1. **Linux网络四层框架**



* 1. 划分

为什么网络已在系统中的实现被称为网络协议栈？

网络被设计成分层的，所以网络的操作就可以称作一个“栈”，这就是网络协议栈的名称的由来。

从交换机/路由器/网关 看协议栈框架划分

* 1. 传输层

数据包的封装、解封装、转发

端到端通信实现

Qos

* 1. 网络层

数据包的封装、解封装、转发

Fw (netfileter/iptables)基于contrack的跟踪

Route 路由表的组织 路由的实现机制

路由缓存

* 1. 链路层硬件层（驱动相关）

数据包的封装、解封装、转发

组网

数据校验和链路保持

1. **Linux内核实现**



* 1. socket协议栈实现
  2. socket和文件描述符的关系
  3. socket控制块
  4. socket消息buff

网络框架演变

1. 系统结构划分netfilter/iptables
2. Windows 和 Linux

网络质量分析

1. 吞吐量 丢包率 延时 带宽
2. MTU 传输速率 网络限速
3. 连接/传输的稳定性分析
4. 流量监控
5. 分析工具 wireshark tcpdump speedtest
6. 硬件层性能测试

物理层一致性测试

~~——WIFI接收性能：射频接受指标（802.11b）、误码率测试仪~~

~~——Ethernet~~

~~http://www.mwrf.net/tech/tm/2014/13678\_3.html~~

~~误帧率~~

应用层吞吐量测试

丢包率测试

延时测试

带宽测试

**软件测试工具——Iperf：**

Iperf 是一个网络性能测试工具。Iperf可以测试最大TCP和UDP带宽性能。Iperf具有多种参数和UDP特性，可以根据需要调整。Iperf可以报告带宽，延迟抖动和数据包丢失。

带宽

延时

抖动

数据包丢失

QOS应用

**硬件测试工具（只是了解）**

1. [**以太网接口物理层一致性测试**](http://www.baidu.com/link?url=Tj7Bj0ltiFDjR_XSOUZrO-PJhR3j5bEkNVQfLHDJx72O1rr98VyqWxQejkmKLxEttG5H_N_LnyqMyV4Jfvgd3yFqJkCN3FSwR80yUV_PYKG)**——力科的QualiPHY**
2. **Wlan物理层测试——WLAN Meter**

**这个月想办法完成ngi-tbox 基本通信的环境搭建 == netfilter --**

**Iptables libpcap**

**//netfilter移植 config network netfilter/2.4---2.6**

**明天任务：发送一个简单的测试工具总结邮件，包括测试参数和测试工具**

**下个礼拜：给出基本的tcp/udp—ip协议栈和netfilter/iptables fw内核分析**

**Eht0 eth1**

**Wlan**

**Ap**

**Ethernet:virtual interface**

**下下个礼拜：给出NAT和fw的发展**

**CIDR**

Nat是为了解决网络地址耗尽的问题，既然nat可以解决这个问题，那么为什么还需要继续开发ipv6来替换现有的解决方案，具体的原因在paper RFC1631

<http://xueshu.baidu.com/s?wd=paperuri%3A%28ee7784398725f354b9efc37b52419428%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fwww.doc88.com%2Fp-8061930132613.html&ie=utf-8>

Tcp传输

——滑动窗口、拥塞控制、慢启动

——超时、重传

——饱和链路

——狙击网络高延时

常规分析：

通过 初始的syn、ack包确认延时 因为这个时候的数据包只在内核处理，不会到四层以上，如果这个过程存在严重的延时那就是设备问题

这个过程没有延时，那么延时会出现在

为什么产生大量Previous segment not captured

丢包原因大讨论

Arp表更新问题

广播故障

路由表更新故障

病毒攻击

数据包间隔超过200ms会有延时

基本应用场景协议

1. 流媒体
2. 可靠传输
3. 控制协议
4. ftp应用

ftp主动模式和被动模式

是否要在ipv4和ipv6两种环境下同时使用，如果需要 启动两份vsftpd，配置两个vsftpd.conf（ipv6模式下不需要启动listen）

开源库：

Libpcap

Iptables-devel

互联网云平台网络技术

1. 基本网络框架Openflow、ovs、vxlan
2. Docker容器
3. **创建source源 管理特定的安装程序**

<http://code.google.com/p/iperf/downloads/list>

网络服务层次

首先是 有网络服务存在 产生流量

网络服务的稳定性

低网络延时

**<1>ip层接收数据流程源码分析：http://blog.csdn.net/shanshanpt/article/details/20699543**

<2>网络数据包粘包、分包<http://blog.csdn.net/cherish_2012/article/details/41681853>

<3>window size value/ tcp dup ack

<4>网络表http://blog.csdn.net/qy532846454/article/details/6806197

<5>提高linux上socket的性能http://www.ibm.com/developerworks/cn/linux/l-hisock.html

<6>多播流量

<7>慢速下载 快速下载

NF\_IP\_PRI\_CONNTRACK\_DEFRAG = -400,

NF\_IP\_PRI\_RAW = -300,

NF\_IP\_PRI\_SELINUX\_FIRST = -225,

NF\_IP\_PRI\_CONNTRACK = -200,

NF\_IP\_PRI\_MANGLE = -150,

NF\_IP\_PRI\_NAT\_DST = -100,

NF\_IP\_PRI\_FILTER = 0,

NF\_IP\_PRI\_SECURITY = 50,

NF\_IP\_PRI\_NAT\_SRC = 100,

NF\_IP\_PRI\_SELINUX\_LAST = 225,

NF\_IP\_PRI\_CONNTRACK\_CONFIRM = INT\_MAX,

NF\_IP\_PRI\_LAST = INT\_MAX,