网络技术知识大纲

---

1. 前言

写这篇文档主要是让网络技术方向的工程师对网络技术涉及的知识点有个大概的了解，便于前期的学习有一定的方向性，特别注意网络技术本身是一个非常宽泛的概念，所以这里主要以项目实践为出发点，阐述涉及到的技术点，对于基础知识的学习，仍应该以 路由基础知识培训 以及 《TCP/IP详解 卷1》 为主。

由于个人能力有限，诸多方面也并不精通，故希望这是一篇能够不断迭代的文档，能够得到各个领域高手们的改进和完善。更新需要由StackEdit辅助完成，并同时更新“CHANGELOG”一节。

---

2. 概述

职责：

1. 保证网络设备拥有正确的网络行为。例如开发路由器则需要考虑包的路由、实现一些代理功能
2. 数据包的hack。由于网关产品在网络中所处位置，通常需要在更底层对数据流进行处理，对网络协议栈进行定制开发
3. 综合性问题。例如吞吐率、系统性能、网络数据流异常等问题的分析

技术要求：

1. 对应“职责1”。要求对网络基础知识（网络协议、工作原理等）有很好的理解，熟悉常用的网络配置工具及调试手段
2. 对应“职责2”。要求了解内核的工作原理及相关知识，深入理解网络设备驱动和网络协议栈的工作原理
3. 对于“职责3”。结合上述两点要求，并具备丰富的经验和对网络的理解

---

3. 基础知识点

3.1 编辑器

虽然这看似偏离了知识大纲的范畴，但其实很重要，这是上战场带的武器之一。

不推荐集成IDE，例如VC、eclipse等。因为这个方向（或者直接理解为嵌入式开发）使用集成IDE不够灵活和轻量级，工作往往涉及到各类代码，如平台C程序、内核代码、各种脚本、makefile、页面文件等等，所以你不可能依

赖集成IDE提供的各种傻瓜式功能来完成所有事情，另外它们的索引功能实在太差。

推荐使用几种工具：
vim、emacs：上手难度稍大，但学会了很好用。

> - vim讲座资料
> - Linux命令行执行 `$ vimtutor` ，30分钟快速上手指南
> - vim的进阶设置可以参考 易水博客，主要是cscope的配置

SourceInsight：上手难度小，用于阅读代码也是不错的选择。

> - SourceInsight简介

Notepad++：最轻量级，适合编辑各类文件。

3.2 开发平台

产品线大多数产品基于嵌入式Linux平台（包括各种裁剪版Android平台），所以学习阶段至少应该基于Linux平台，了解Linux平台的网络配置/调试方法。网络技术方向需经常与Linux命令行工具打交道，故应该尽早习惯使用CLI。

> - 各种Linux入门书籍。略读就可以了，有个基本概念即可，具体要用到什么的时候再查资料。
> - shell脚本语法相关书籍，略读，初学阶段不要求掌握高级的脚本编写，以能够看懂为主

3.3 路由功能介绍

一定要读一遍 《路由器基础学习报告》 ，里面大部分的功能就是我们的网络开发需求，同时文章也介绍了这些功能的基本原理。

3.4 网络接口

首先来解答一个最初级的疑惑，路由器这么多接口到底是怎么关联的，数据包是怎么流过这些接口的。借助下面这幅网络接口图来展开（假设路由上的OS为Linux），一般的五口路由器的接口如下（网络接口的查看可以执行 `$ ifconfig -a` 或者 `$ ip link` ）：

图中组件的介绍：

- **switch**: 以太交换芯片。其下一般有5个port，图中由p0-p5表示，对于用户每个口就是一个网络接口，但对于路由来说，帧其实首先经过switch的处理，再会发往CPU。这里的CPU也可以理解为内核、网络接口驱动程序、协议栈。**路由器主要就是处理包的转发，而switch在当中起了至关重要的作用，因为它可以直接转发两个LAN以太口之间的帧，这样则不需要将每个包都发往CPU进行处理，大大降低的CPU的负载。**
- **eth0/eth1**: Linux实际看到的以太接口。**注意经过switch后，LAN域的4个以太口在Linux上就只能看**

到**1个接口了。**

- **wlan0**: 无线芯片对应的网络接口。
- **br0**: 通常的模式下，路由器希望有线和无线的网络接口都在同一个LAN，所以会使用到Linux kenrel的虚拟网桥。这时eth0和wlan0又被桥接到br0上。所以最终Linux系统上的WAN口是eth1，LAN口是br0。**对于稍上层的功能，例如防火墙，则可能只需要知道WAN口和LAN口对于的接口。**
- **ethernet driver**: 以太网驱动：
  - CPU和switch芯片之间的帧需要由以太驱动控制硬件进行收发处理，有些数据包hack也可能在这里实现，eth0和eth1则是由以太驱动程序创建
  - 一些swtich可以实现VLAN、HNAT[1]、QoS等等高级功能，需要通过驱动进行配置，当然后面的执行基本由swtich独立完成

- **wlan driver**: 无线驱动：
  - CPU和WLAN芯片之间的帧需要由无线驱动控制硬件进行收发处理，并且会将**802.11的帧转化成802.3的帧**，同样很多功能会在这里hack数据包，wlan0则由无线驱动创建
  - WLAN功能非常复杂，涉及到各类桥接模式，如client模式、ap模式等。根据这些模式，**路由器接口间的连接方式会发生改变，而并非永远如图这样**

- **Linux virtual bridge**: Linux内核实现的虚拟网桥，甚至可以认为完成的功能与switch芯片相同。内核对应实现在 `linux_kernel_source/net/bridge` ，项目中应用较为频繁，但对实现并没有太多关注。建议阅读 《深入理解LINUX网络技术内幕》 第十四章，了解基本原理。
- **TCP/IP stack**: 内核网络协议栈。一个非常复杂的内核子系统，其中又包含一些重要的内容，如netfilter（重要的数据包hack手段）、conntrack（连接跟踪机制）、TC（流量控制）。

网络数据包的一般流向：

- ①<->②: 帧流经switch后上CPU，进入Linux内核协议栈，经过路由决策后发往特定的网络接口
- ②<->③: 帧流经swtich后直接被swtich转发，不会上CPU，此类数据包你也无法通过抓包程序抓取
- ②<->④: 无线802.11的帧被wlan driver转化成有线802.3的帧，此后类似于①<->②的流程

通过这幅图，就大致概括了网络技术方向底层的知识结构。但**这幅图放在这一节的主要目的并不是为了介绍底层的知识结构，这节的名称已经很明确地说明我其实是想介绍“网络接口”**，所以这节可能会给读者产生一些疑惑，但先不必纠结于此，随着对网络技术的深入理解，这些内容自然便会有清晰的认识。

3.5 基本的技术方向

借助前面的介绍，我把“概述”一节中的”技术要求“进一步细分到技术点，便于你对自己的技术发展方向有一个大致的定位：

- 上层网络功能的配置和开发（如拨号方式、linux网桥配置、防火墙、DHCP功能等）
- 协议栈开发（netfilter/iptables模块开发、conntrack开发、TC开发）
- 以太驱动开发
- switch芯片配置（主要是高端机型需要offload cpu或者一些特殊应用场景需求）
- wlan驱动开发及桥接模式的开发（与WiFi技术方向有交叉）
- 网络相关debug（在工作中占很重要的比例，需要对具体的技术方向有深入了解）

3.6 常用工具

iptables： 用于配置防火墙，hack数据包。

> - 先阅读 《iptables个人总结》

- 再阅读 《iptables指南1.2.2》 ，这篇文档可能今后你会反复查阅，另外命令行执行 `man iptables` 查看手册也是不错的选择

总之第一步要理解iptables是什么、有什么用。这个工具必须熟练掌握，后面的很多知识点也会与此有关联。

ifconfig： 查看接口状态、配置接口参数。

ip： 新一代的网络配置工具。功能十分强大，可以配置/查看很多内容，例如接口参数、路由表、策略路由、隧道等等。一些高级功能需要用到，比较依赖内核态的支持，目前基本功能用得不多，需要时多查资料和手册 `$ man ip` 。

tcpdump： 抓包工具。DUT[2]上抓包分析需要用到。

- 《tcpdump中文手册》

wireshark： 抓包工具。tcpdump抓下来的数据可以用wireshark来分析，毕竟图像化界面更易于分析。

- wireshark基本使用指南

route： 查看/配置路由表。

netstat： 查看socket连接状态及路由表。

Tips： 很多网络工具，如tcpdump、route、netstat等执行时会解析域名，但大多数情况下调试的时候DUT并没有连接internet，导致返回结果缓慢，所以记得加 `-n` 参数关闭域名解析

brctl： Linux虚拟网桥配置工具。路由往往会把LAN域的以太口和WLAN桥接在一起形成同一个LAN域（参考"网络接口"一节）。

/proc/net/nf_conntrack： 记录了连接跟踪表，学习NAT时，往往需要用到。（较老的内核是 `/proc/net/ip_conntrack` ）

/proc/net/arp： 本机arp表，查看MAC与IP地址的对应关系。

---

4. 进阶知识点

iptables基础应用

很多功能实际都依赖iptables规则来实现的，所以iptables对网络技术方向的工程师来说是必须掌握的工具之一。

防火墙

防火墙是路由器的重要功能。防火墙iptables规则一般放于filter表中，一个简单的防火墙规则举例（对规则有任

何疑问时应即时查看iptables的使用手册）：

```
    # $lan_if，$wan_if： LAN/WAN接口名
    # $lan_ip，$wan_ip： LAN/WAN接口IP
    # $lan_mask，$wan_mask： LAN/WAN接口IP掩码

    # 默认策略。一般的没有被路由确认的包都应该被DROP
    iptables -t filter -P INPUT DROP
    iptables -t filter -P OUTPUT ACCEPT
    iptables -t filter -P FORWARD DROP

    # LAN口数据一般是可信的，所以接受LAN口的一些数据包，而WAN口数据包仍然按照默认策略被丢弃
    iptables -t filter -I INPUT 1  -i $lan_if ! -p icmp -s $lan_ip/$lan_mask -j ACCEPT
    iptables -t filter -I INPUT 2  -i $lo_if -p ALL -j ACCEPT -m comment --comment "loop back"
    iptables -t filter -I INPUT 3  -i $lan_if -p icmp --icmp-type echo-request -j ACCEPT
    iptables -t filter -I INPUT 4  -i $lan_if -p udp --dport 67 --sport 68 -j ACCEPT
    iptables -t filter -I FORWARD 1  -i $lan_if -j ACCEPT

    # 所有被确认的包（连接跟踪表中有记录），则都可以通行。所以一般的情况是LAN首先对外发出请求，而WAN回复的数据包则被连接跟踪表记为RELATED，此时则不会被防火墙过滤掉
    iptables -t filter -I INPUT 5  -m state --state RELATED,ESTABLISHED -j ACCEPT
    iptables -t filter -I FORWARD 2  -m state --state ESTABLISHED,RELATED -j ACCEPT

    # 适当限制一些TCP状态包发包频率，防止路由器遭到flood攻击
    iptables -t filter -A syn_flood -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -m limit --limit 25/sec --limit-burst 50 -j RETURN
    iptables -t filter -A syn_flood -j DROP
    iptables -t filter -A INPUT -p tcp -m tcp --tcp-flags FIN,SYN,RST,ACK SYN -j syn_flood
```

NAT

NAT（Network Address Translation）是一个很重要的概念，对于家用路由来说，与其说它是个路由，不如说它其实是个NAT。NAT有不同的类型，会对一些应用产生影响，将在“NAT类型”一节中介绍。本节介绍iptables如何完成SNAT的配置。NAT的iptables规则需要放到nat表中（对照上节，思考为什么防火墙规则要放在filter表，而NAT规则要放在nat表中）。

```
    iptables -t nat -A POSTROUTING -o $wan_if -j MASQUERADE
```

MASQUERADE是iptables的一个target，用于实现SNAT，另外iptables也有SNAT这个target，主要的不同在于SNAT需要指定IP地址，而一般情况下路由在规则建立后WAN口才获得IP，所以用MASQUERADE更便捷。

TCP MSS clamping

中文姑且翻译成”TCP MSS钳制”，这是一个不能被忽视的问题，否则我们的路由产品可能出现在某地完全无法使用的严重问题。

为了理解这个问题，首先需要明白一些基本原理。在IP数据包传输时，有时是不希望进行IP分片的，因为这会引起一些效率问题，而且接收端如果是一些轻量级的协议栈，那它不一定能处理IP重组，所以TCP协议往往都设置了DF标志位，你可以通过wireshark抓取一些TCP syn包分析，几乎都设置了DF标记。在这种情况下，TCP协议报文会根据发送接口的MTU（Maximum transmission unit）来确定所使用的MSS（maximum segment size）。

接下来通过实例来分析这个问题，假如客户端通过HTTP访问互联网中的web服务器：

路由器LAN口MTU一般为1500，而客户机一般也没有限制同样设置为1500，所以这条链路的PMTU（Path Maximum Transmit Unit）为1500；web服务器所在链路的PMTU假设为1480；当路由器连上ISP后，WAN口MTU设置为它所在链路的PMTU，接口驱动通过跟运营商协商，设置为一个较小的值1300。

HTTP访问使用TCP协议的连接建立过程如下：

1. PC发送TCP SYN，协议中MSS值根据本地接口MTU计算得到，即 $1500 - 40 = 1460$ 。
2. web服务器回应TCP SYN ACK，携带自身MSS。
3. 协商后这条TCP连接使用使用两者中较小值作为最终的MSS值，即1440。那么链路层的帧的MTU则为1480。
4. 此后1480长度的帧将无法通过ISP与路由器之间的这段链路。

路由器加入TCP MSS clamping之后的情况：
![tcpmss2][3]

路由器hack来往的TCP SYN/SYN ACK包，始终保证MSS被钳制于路由两端的最小PMTU，即 $1300 - 40 = 1260$ ，最后由于两端都认为对方的MSS为1260，所以TCP连接得以建立。

这个功能对应的iptables命令为

```
iptables -t mangle -I POSTROUTING 1  -p tcp --tcp-flags SYN,RST SYN -o rmnet0 -j TCPMSS --clamp-mss-to-pmtu
```

netfilter框架hook点

**********TODO**********

NAT进阶

除了基本的NAT配置，还有很多问题也离不开NAT，所以一定要对NAT进行深入学习。比较优秀的资料有 “NAT” from Wikipedia 以及《TCP/IP Illustrated, Volume 1: The Protocols (2nd Edition)》第7章的内容。

NAT类型

通过在 路由基础知识培训 中的学习，应该对NAT类型有了较深入的认识，如果还有疑问，可以参看Redmine上 不同类型的NAT 这条回复。

NAT Loopback

> NAT loopback, also known as NAT hairpinning or NAT reflection

主要出现在DMZ和Virtual Server功能中，分析和解决方法详见Redmine Nat Loopback的分析与解决。

Hairpin Mode

NAT Loopback的出现随即产生了另一个问题，这个问题需要发生在网桥下。但总体来说这个问题在实际应用中并不多见，只是理论上存在，但通过对这个问题的分析，可以进一步加深对NAT Loopback、桥接的理解。

![hairpinmode][4]

如图的红色虚线则为NAT Loopback的环回路径，但需要留意的是，回环实际发生在同一个物理接口上（即无线接口wlan0）。

回顾 Nat Loopback的分析与解决 ，当endpoint1通过路由器外网IP地址访问endpoint2映射到路由上的DMZ或者Virtual Server时，数据包在PREROUTING链进行了DNAT，到Linux虚拟网桥br0上，实践已是两个内网IP的通信。网桥工作在链路层，它根据数据包目的IP对应的MAC地址，查找fdb（Forwarding database）记录的目的MAC和网桥端口（这里包括eth0和wlan0）的对应关系获知应该发往哪个端口，所以最终确认这个数据包将发回wlan0。然而基于实际物理网桥的原理，它认为从某个端口进来的帧，必然是已经确认需要转发到其他端口才会进入网桥处理，所以它并不会将这个帧发回原始端口，故在无线接口这类支持客户端多连的条件下，NAT Loopback结合Linux虚拟网桥则会产生问题。

目前Linux虚拟网桥实现已经充分考虑到此问题，当执行 `echo 1 > /sys/class/net/br0/br_if/wlan0/hairpin_mode` 开启网桥hairpin mode后，问题得以解决。

更详细的分析可以查看Redmine【TR761】虚拟网桥下的NAT Loopback（hairpin模式）。

NAT Traversal

NAT穿越是一个比较宽泛的概念，你可以理解为**任何正常的网络应用需要通过一些特殊处理才能越过NAT网关进行工作**，则需要NAT穿越。比较优秀的资料有 “NAT traversal” from Wikipedia 和 《TCP/IP Illustrated, Volume 1: The Protocols (2nd Edition)》 7.4节。下面归纳下NAT穿越一般可以分为几类功能。

Hole Punching

http://en.wikipedia.org/wiki/Hole_punching
TODO

VPN over NAT

PPTP L2TP IPsec
TODO

转发规则

DMZ

port trigger

virtual server

upnp

ALG

ALG（Application Level Gateway），即应用层防火墙。在Redmine上已经总结有很好的文章，可以查看:

- ALG技术白皮书
- SIP ALG相关内容
- FTP ALG相关内容

总体来说，基于Linux平台的ALG已经比较完善，只要开启相应的配置项即可。

连接跟踪

ARP?

igmp/igmp proxy

dns/dns proxy

tcp window management/tcp congestion control

tcp state transitions

代理?

multi path routing

tc

LKM的编写

协议栈的基本数据结构skb

list & hlist

---

5. 高阶知识点

netfilter/iptables的实现

以太驱动的一般实现

数据包收发流程

linux网桥的实现

tc的实现

conntrack的实现

CHANGELOG

版本	负责人	备注
V1.0	欧阳雄弈	创建文档

> Written with StackEdit.

---

1. 硬件NAT，NAT一般在协议栈中完成。但为了进一步降低CPU负载，可以借助swtich来完成。 ↩
2. Device Under Test，即当前所开发或测试的设备，工作中会频繁用到。 ↩