Linux 下 Ping 命令在内核中动态运行过程的跟踪与分析

董志超 王柄璇 倪光玉 鲁 曈 荆 琦*
(北京大学软件与微电子学院 北京 100871)

摘 要 自由而开源的 Linux 日渐被人们所熟悉,而当今互联网产业飞速发展,使得 Linux 内核中网络部分源码被越来越多的人所关注。详细分析 Linux 新版本 3.5.4 内核的网络子系统的组织结构、功能和运行机制,通过建立 ddd + qemu + busybox 实验环境和 Ping 指令应用场景来追踪 Ping 指令在 Linux 内核中的动态运行过程,然后对 ping 指令的运行特点进行介绍和原因分析,揭示网络子系统在特定应用时的动态运行过程和静态分析调用流程的区别与联系。阐明网络子系统因应用而发展的特点,加深和拓宽对 Linux 网络子系统机制和发展的理解与思考。

关键词 Linux 内核 Linux 网络 Ping 指令 qemu

中图分类号 TP393 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2015.07.024

TRACKING AND ANALYSING PING'S DYNAMIC WORKING PROCESS UNDER LINUX

Dong Zhichao Wang Bingxuan Ni Guangyu Lu Tong Jing Qi*
(*School of Software and Microelectronics*, *Peking University*, *Beijing 100871*, *China*)

Abstract Free and open-source Linux is familiar to us gradually; with the rapid development of the Internet industry today, more and more people are interested in the source code in network part of Linux kernel. We thoroughly analyse the organisational structure, functions and working principle of network subsystem in new Linux version 3.5.4 kernel, and track the dynamic working process of Ping in Linux kernel by establishing experimental environment of "ddd + qemu + busybox" and setting application scenario of Ping command, then we introduce the features of Ping command in running state and analyse the causes, reveal the differences and connections between dynamic working process and static analysis calling process when the network subsystem is running a specific application, clarify the network subsystem's development characteristics due to application requirements, and deepen and broaden the understanding and consideration of Linux network subsystem's mechanism and development.

Keywords Linux kernel Linux network Ping command qemu

0 引 言

Linux 系统由于其自由而且源码开放的特点,日渐受到人们的认知和青睐,而当下互联网产业蓬勃发展,使得越来越多的人开始关注和学习 Linux 内核网络部分的源码,并对其进行深入研究和优化。Linux 初学者可以通过阅读经典书籍来接触和学习经典内核版本中(如 2.6 版本)Linux 网络子系统的整体结构和各个组成部分,同时也需要一些在内核中实际运行的案例和简单实用的实验环境以加深对内核网络子系统的理解。

以下首先对新版本 Linux-3.5.4 内核网络子系统的整体架构、功能和运行机制进行详细介绍,然后搭建实验环境构造应用场景,跟踪和调试了 Ping 指令在 Linux 内核中的动态运行过程。通过和网络子系统静态的函数调用流程图进行对比,阐述了 Ping 指令自身运行特点,解释说明了 Ping 指令在 Linux 内核动态运行过程的特点,加深对 Linux 网络子系统的机制和应用的理解。

1 Linux 网络子系统简介

Linux 网络子系统的总体架构成功借鉴了 OSI 七层网络协议栈架构和 TCP/IP 协议四层结构[1-3],通过众多学者坚持不懈的研究和开发,适应了互联网的起步和发展过程中的应用需求,经历了一个从无到有,从简单到复杂的漫长过程。Linux 网络子系统的各种功能的实现分别在不同的时间,由不同的代码块完成。Linux-3.5.4 版本内核网络子系统的组织架构[4-8]可以自上而下表示为:系统调用接口层、协议无关接口层、网络协议层、网络接口层、设备驱动层,如图 1 所示。

网络子系统被分成五层结构即五大模块,每个模块又可以细分为几个小模块,每个小模块中标注了一些实现功能的核心函数或数据结构。

收稿日期:2013 - 10 - 16。国家科技重大专项核高基专项(2012ZX01039-004)。**董志超**,硕士生,主研领域:嵌入式软件工程。**王柄璇**,硕士生。**倪光玉**,硕士生。**鲁曈**,硕士生。**荆琦**,副教授。

其中,系统调用接口层给用户空间提供了正常访问内核的合法途径,用户可以提出网络应用需求,如接收或者发送数据等;协议无关接口层是由 socket 来实现并提供一组通用函数来支持各种不同的协议,支持 BSD socket 和 INET socket 两种 socket; 网络协议层包含了 Linux 所支持的多种网络协议,如 TCP 协议、UDP 协议和 IP 协议等;网络接口层(网络设备接口层)将网络协议与具有很多各种不同功能的硬件设备连接在一起;网络设备驱动层是负责管理物理网络设备的设备驱动程序[5,6]。

Linux-3.5.4 版本内核的网络子系统中功能实现的核心函数及其静态函数调用流程如图 2 所示。

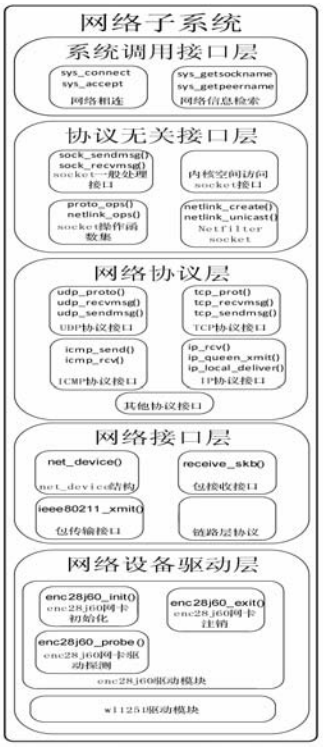

图 1　Linux-3.5.4 版本网络子系统分层结构图

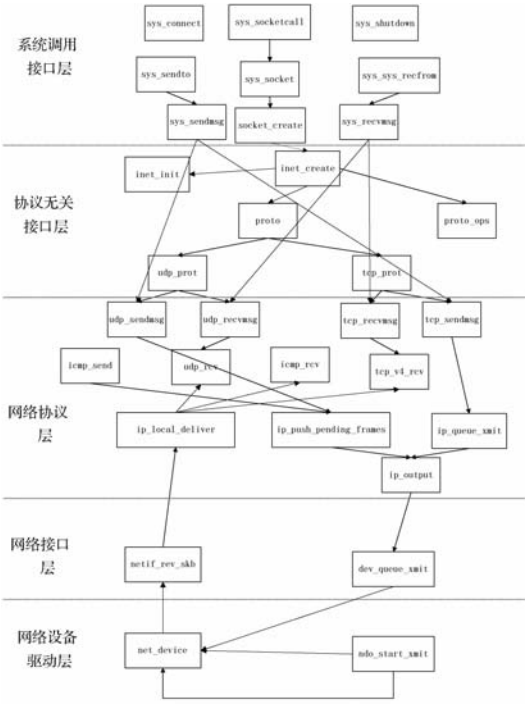

图 2　Linux-3.5.4 网络子系统核心函数调用流程图

网络子系统最基本、最重要的功能是发送和接收数据。由图 2 中核心函数以及箭头表示的调用方向,以 TCP 协议为例,网络子系统典型的发送和接收数据的工作过程可以描述如下[5-7]:

当网络子系统有 TCP 操作响应,应用程序需要通过网络传输数据时,首先调用系统调用层 sys_socketcall,开始创建一个 socket,通过调用 socket_create 等函数完成 socket 创建,然后通过 sys_connect,sys_bind,sys_listen 等实现 socket 连接、绑定、监听等工作;进入到协议无关接口层,通过 proto --> tcp_prot 创建和选定支持 TCP 协议的通用函数接口;调用网络协议层的 tcp_sendmsg --> ip_queue_xmit --> ip_output 进行数据包的 TCP 封装、IP 封装;进入网络接口层,将数据包封装成帧类型,由 dev_queue_xmit 交给下一层;网络设备驱动层通过数据结构 net_device 和 ndo_start_xmit 之间的指针操作管理网卡将数据帧发送出去。

当网络子系统接收数据包时,首先由硬件中断触发开始,网络设备驱动层通过 net_device 和 ndo_start_xmit 管理网卡将接收到的帧传递至网络接口层的 netif_rev_skb,完成对帧数据的解析后,进入到网络协议层,经过 IP 分组解析完毕数据,由 ip_local_deliver 指向 TCP 协议的接收队列 tcp_v4_rcv,应用程序可以通过系统调用 sys_sys_recfrom --> sys_recvmsg --> tcp_recvmsg 流程从 TCP 的接收队列里面接收数据。

2　实验环境和应用场景

Ping 指令是同时存在于 Window 环境和 Linux 环境,通常用于网络连接成功与否的测试,同时也可以帮助我们分析和研究网络故障。最简单的应用格式:ping IP 地址,该命令还可以加许多参数使用。

通过 ddd + qemu + busybox 这个工具套件搭建实验环境,来跟踪和调试 Ping 指令在 Linux 内核中的运行过程:利用 qemu 模拟 Linux 内核的真实工作状态,使用 busybox 集成的常见 Linux 命令,用 ddd 进行程序调试。

在 ddd + qemu + busybox 的实验环境中,在 qemu 虚拟出来的内核环境中 Ping 目标主机 127.0.0.1,目的是首先通过 Ping 指令,贯穿了 Linux 网络系统的整体架构,将信息发送到目标主机;同时 qemu 又会收到目标主机的回复信息,接收回复信息的时候,又会贯穿整个 Linux 整体架构,这样从发送数据和接收数据两条主线,可以将 Linux 网络系统很好的贯穿起来,能够很好地体现验证环境的真实性和内核工作的动态性。

3　实验过程和结果

在 ddd 调试窗口插入数个断点如下:

sys_socketcall()	ping_rcv()
sys_socket()	tcp_recvmsg()
sys_bind()	tcp_sendmsg()
sys_connect()	tcp_v4_rcv()
sys_listen()	tcp_transmit_skb()
sys_accept()	icmp_send()
sys_send()	icmp_rcv()
sys_recv()	udp_recvmsg()
sys_sendto()	udp_rcv()
sys_recvfrom()	ip_local_deliver()
sys_shutdown()	ip_rcv()
sys_sendmsg()	ip_queue_xmit()
sys_recvmsg()	ip_push_pending_frames()
ping_init_sock()	ip_output()
ping_bind()	dev_queue_xmit()
ping_queue_rcv_skb()	netif_receive_skb()
ping_sendmsg()	e1000_xmit_frame()
ping_recvmsg()	e1000_intr()

插入断点是为了更有效率地跟踪 Ping 指令的运行过程,此

外上述断点的设置目的也可以分为两种：一种断点是为了验证 Ping 指令运行过程会经过此类断点——说明运行过程经历了该断点所处的某段分支，如 sys_socketcall( )，ip_push_pending _frames( )等；一种断点是为了验证 Ping 指令不会通过此类断点或者暂时不确定是否通过此类断点——说明运行过程未经历该段分支或者解答 Ping 指令运行过程中的疑问，如 tcp_transmit _skb( )，udp_rcv( )，e1000_xmit_frame( )，e1000_intr( )等。通过 ddd 提供的 continue、step、next、step instruction 和 next instruction 等几种不同的单步调试方式来实时跟踪和调试 Ping 指令在内核的实时运行。

实验环境搭建完毕后，首先连接 ddd 和 qemu：qemu 切换至控制台状态后输入“gdbserver tcp::1234”，gdb 的命令行中输入“target remote localhost:1234”。在 qemu 中输入指令“ping -c 1 127.0.0.1 ”。-c 1 参数表示只给目标主机发送一个数据包。然后在 ddd 窗口对 Ping 指令的运行过程进行跟踪和调试，各个断点的通过情况和通过顺序如表 1 所示（表中上下顺序代表断点通过的顺序）。

表 1 断点通过顺序和解释说明

通过顺序	断点序号	断点名称	断点功能及备注
1	1	sys_sockecall	print call 命令查看 call 的值，输出为 2，进入 sys_socket
2	2	sys_socket	socket 的创建过程
3	1	sys_socketcall	call = 14，进入 sys_setsockopt，进行 socket 设置
4	1	sys_socketcall	call = 14，进入 sys_setsockopt，进行 socket 设置
5	1	sys_socketcall	call = 11，进入 sys_sendto，准备发送数据
6	9	sys_sendto	数据发送开始
7	33	ip_push_pending _frames	进入到网络协议层
8	34	ip_output	继续通过 ip 协议发送数据
9	36	dev_queue_xmit	进入到网络接口层
10	31	ip_rcv	发送到目标主机的消息得回应，收到回应数据，开始通过 ip 协议进行接收。
11	30	ip_local_deliver	向上层传递
12	22	icmp_rcv	收到的数据传递至 ICMP 协议接收缓冲队列。
13	33	ip_push_pending _frames	再次向目标主机发送一次数据
14	34	ip_output	进入到网络接口层
15	36	dev_queue_xmit	进入网络接口层，继续发送数据
16	31	ip_rcv	收到目标主机的第二次回复数据
17	30	ip_local_deliver	将收到的数据向上层传递
18	22	icmp_rcv	ICMP 协议接收数据至接收队列
19	20	ping_rcv	ping 指令的接收队列，ping_rcv 也是源码 ping. c 中极少的能被外部文件调用的函数之一，多数有关 ping. c 中功能函数被定义为静态函数
20	1	sys_socketcall	此处 call 值为 12，进入 sys_recvfrom
21	1	sys_recvfrom	应用程序获得一次传递数据
22	1	sys_socketcall	call 值为 12
23	10	sys_recvform	应用程序获得第二次传递数据
24	-	-	结束

通过以上步骤，针对我们所感兴趣的范围，可以简略做出 Ping 指令在 Linux-3.5.4 版本内核中的动态运行过程简图，如图 3 所示。

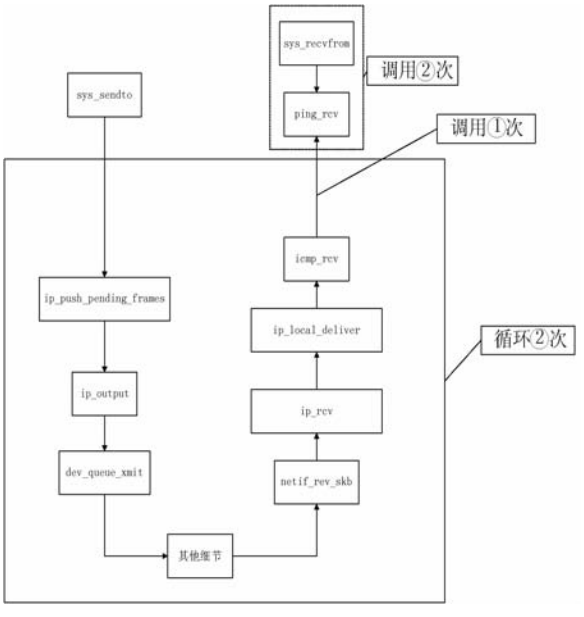

图 3 Ping 指令运行过程简图

图 3 是 Ping 指令运行过程中比较重要的部分，其描述如下：

（1）创建 socket 过程，创建成功后，socket 选项设置、socket 绑定等过程；

（2）准备工作完成后，系统调用 sys_sendto，开始发送数据包；

（3）从 ip_push_pending_frames --> ip_output --> dev_queue_xmit --> ……是数据包在网络子系统自顶向下的传递发送过程；从 netif_rev_skb --> ip_rcv --> ip_local_deliver --> icmp_rcv 是数据包在网络子系统自下向上的传递接收过程；在本次实验中，这个过程循环了 2 次，即：从 ip_push_pending _frames --> …… --> icmp_rcv 一次循环结束后马上进行第二次 ip_push_pending_frames --> …… --> icmp_rcv 循环；

（4）从 icmp_rcv --> ping_rcv 只调用了一次；

（5）系统调用 sys_recvfrom 从 ping_rcv 中提取数据包，共调用了 2 次。

4 实验结果分析

（1）在 TCP 协议和 UDP 协议的发送和接收的重要函数位置设置了断点，这些断点都没有被捕捉到，说明 Ping 指令没有使用 TCP 和 UDP 协议，这点符合预测，因为 Ping 使用的是 ICMP 协议[9,10]。

（下转第 154 页）

[4] 李鹏飞,陈朝武,李晓峰. 智能视频算法评估综述[J]. 计算机辅助设计与图形学报,2010,22(2):352-360.
[5] Liu L M,Li Z,Edward J. Efficient and low-complexity surveillance video compression using backward-channel aware Wyner-Ziv video coding [J]. IEEE Transactions on Circuits and Systems for Video Technology,2009,19(4):453-465.
[6] Wang Z,Liang L,Alan C B. Video quality assessment using structural distortion measurement[C]//International Conference on Image Processing,Rochester,NY,USA,2002,3:65-68.
[7] Yu Z H,Wu H R,Winkler S,et al. Vision-model -based impairment metric to evaluate blocking artifacts in digital video[J]. Proceeding of the IEEE,2002,90(1):154-169.
[8] Nill N B,Bouzas B H. Objective image quality measure derived from digital image power spectra[J]. IEEE Signal Processing Letter,2002,9(3):388-392.
[9] Wang Z,Alan C B,Hamid R S. Image quality assessment:from error visibility to structural similarity[J]. IEEE Transactions on Image Processing,2004,13(4):600-612.
[10] 高杨. 视频质量诊断算法研究与实现[D]. 沈阳,东北大学,2011.
[11] 欧阳伟. 基于图像分析的监控视频图像异常诊断系统的研究与实现[D]. 武汉,华中师范大学,2012.
[12] Zol. [EB/OL]. http://security. zol. com. cn/278/2783706. Html/.
[13] 柴云峰,黄显林,介鸣,等. 一种快速灰度投影算法的实现与仿真[C]//Proceeding of the 25th Chinese Control Conference,2006,08-07,Harbin,Heilongjiang,882-885.
[14] Baidu. [EB/OL]. http://baike. baidu. com/link? url = YfxA81 eAkTkMFjIpfHAYx4G1ruvjXXs8U0Al-A21sVuNF7jlb-QRTNE _ hYfogi _ J6XTea3DzeXpDkjdAnxWaWq.

(上接第 104 页)

(2) Linux 内核版本中网络部分的更新是比较快的,新版本中持续不断进行各种算法优化和创新。对于 Linux-3.5.4 内核版本,Ping 指令使用 ICMP 协议的个别细节如下:Ping 指令使用了 ICMP 的查询报文,但在发送过程中,没有通过 ICMP 协议自身的发送方式,而是跨越了 ICMP 协议的发送功能的核心函数(断点 icmp_send 没有捕捉到),直接通过 IP 协议进行发送;数据接收的时候,还是通过了 ICMP 协议的接收功能函数(icmp_rcv 捕捉到)进行数据接收。在发送过程中直接透过 ICMP 协议这一层而进入 IP 协议开始数据包的发送,提高了网络子系统的工作效率,避免了不必要的时间和资源浪费。

(3) Linux-3.5.4 版本中 Ping 指令不需要建立稳定的 socket 连接:验证过程中,在 sys_bind 和 sys_connect 的断点没有被 Ping 指令通过,这是由于 Ping 指令的应用特性,主要是用于探测主机 A 到主机 B 之间是否可通信,不需要传递大量数据,所以根据需要进行了相应的简化。

(4) Linux 下应用 Ping 指令 ping 某个 ip 地址时,当参数-c 没有设置,如“ping 127.0.0.1”,在某些 PC 机上就会不停地 ping 目标地址,导致大量的计算机资源被占用,尤其是在虚拟机的环境下很容易导致系统过度繁忙而进入假死状态,所以建议在 Linux 环境下使用 Ping 命令时,要加上参数-c n(次数)。此外,如果-c 后面的参数不是大于等于 1 的自然数,而是 0 的话,例如实验中曾经尝试“ping - c 0 127.0.0.1”命令,会发现 ping 过程还是无限次数的持续下去。分析其中原因,可能性为两个:①0 不属于相应的参数设定范围而导致参数设定无效,而无限次的 ping;②-c 参数可能有如下特点:-c 参数是无符号整数类型,并且-c 参数设定完毕后,先进行一次 ping 过程,然后验证-c 参数的值减“1”是否为 0,来判断 ping 过程是否要结束,由于计算机本身特点,会导致-c 参数变成最大允许值而进行无限次的 ping 过程。

(5) 在 Linux 网卡处的断点 e1000_xmit_frame( ) 和 e1000_intr( ) 均没有被捕获,同时网卡流量数据也监测不到,这是由于本次实验的特点所导致。实验是用 qemu 模拟 Linux 系统,qemu 本身也建立在主机之上,即 qemu 和主机共用相同的硬件设备。当两者之间进行网络通信时,qemu 中 ping 到主机的 127.0.0.1,这个地址主要是用于网络软件测试以及本地机进程间的通信,所以在 qemu 中 ping 这个地址,主机不会进行任何物理的网络传输,即不会通过网卡。此外,如果主机自身 ping 自己的 ip 地址,或者 KVM 类型的虚拟机和主机之间 ping 过程,也不会进行物理意义上的网络传输,系统会在 IP 路由表中根据转发规则识别本地地址而不经过物理网卡;如果是发生在同一主机上 KVM 类型的虚拟机 A 和虚拟机 B 之间 ping 过程,是否会通过物理网卡进行数据传输,会因为环境设定的不同而变得复杂和不确定。

(6) 实验中数据包的发送过程循环了 2 次,原因可能有两点:①本次实验的 Linux 下网络环境参数中 MTU 值太小,导致 Ping 过程中,IP 协议对数据包进行的分片,产生 2 次循环;②由于本次试验中自身的特点,即是由 qemu 环境下 Ping 127.0.0.1 进行本地网络测试的特殊性而导致的,后者的可能性较大。

5 结 语

通过上述实验过程和问题分析,可以看出在 Linux 网络子系统的学习实践中,有着各种各样的未知环节和特殊性。它们从另一种角度诠释了 Linux 网络子系统,同时也加深了我们对 Linux 内核的理解。Linux 内核因自身的架构和内容特点与飞速发展的互联网络之间产生的“矛盾”而诞生出了多种应用需求,进而为 Linux 研发人员提供了研究方向,也为 Linux 自身创造了发展空间和机会,使得 Linux 拥有了越来越多的爱好者。

参 考 文 献

[1] Daniel P Bovet,Marco Cesati. Understanding the Linux Kernel[M]. 3rd ed. O'Reilly Media, Inc. ,2005.
[2] Wolfgang Mauerer. Professional Linux Kernel Architecture[M]. Wiley,2008-10.
[3] Robert Love. Linux Kernel Development[M]. Addison-Wesley,2010.
[4] Christian Benvenuti. Understanding Linux Network Internals [M]. O'Reilly Media, Inc. ,2006.
[5] Thomas F Herbert. The Linux TCP/IP Stack:Networking for Embedded systems(Networking Series)[M]. Charles River Media,2004.
[6] 樊东东,莫澜. LINUX 内核源码剖析——TCP/IP 实现[M]. 北京:机械工业出版社,2010.
[7] 胡希明,毛德操. Linux 内核源代码情景分析[M]. 浙江:浙江大学出版社,2001.
[8] 陈莉君. Linux 操作系统内核分析[M]. 北京:人民邮电出版社,2000.
[9] 吴国伟,李张,任广臣. Linux 内核分析及高级编程[M]. 北京:电子工业出版社,2008.
[10] 赵炯. Linux 内核完全注释[M]. 北京:机械工业出版社,2004.