Linux 内核移动性支持机制与实现

谢婉君,贾 濡
(北京交通大学 电子信息工程学院,北京 100044)

摘 要:随着信息技术的高速发展及移动终端的普及,用户期望网络能够在任何时间、任何地点,以任何方式提供信息服务。为了更好地支持移动性,业界提出移动IP、身份位置分离机制、SIP等多种方案。然而,它们都不能同时支持主机移动性和服务移动性。文中提出的移动性支持机制是在Linux内核中,利用一体化网络中的连接标识管理服务,在协议栈中同时支持终端和服务的移动性。首先,提出了两种内核中快速捕获IP地址更新的机制;然后,修改Linux内核中的套接字,实现IP地址改变但连接保持的目标,从而实现移动性;最后通过比较两种机制在实现移动性上的时延,选择了内核通知链机制作为移动性支持机制。

关键词:Linux内核;IP地址;定时器;通知链;移动性
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2015)03-0103-05
doi:10.3969/j.issn.1673-629X.2015.03.024

Mechanism of Mobility Support in Linux Kernel and Its Implementation

XIE Wan-jun, JIA Ru
(School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China)

Abstract: With the rapid development of communication technology and the widespread of portable terminal devices, the users require that the Internet can provide access service at any time, in any location, and by all means. In order to support mobility, the community has proposed a few schemes such as Mobile IP, SIP and Identifier/Locator separation. However, none of them can support host mobility and service mobility at the same time. It has proposed a solution to support both host mobility and service mobility in this paper, which is realized by using the CID (Connection IDentifier) in the universal network to manage connections. Firstly, come up with two mechanisms to realize quick access to updated IP addresses in Linux kernel. Secondly, modify the API socket created by the system when a connection is established between a client and a server, so that the connection keeps unchanged, regardless of the change of IP addresses. At last, after comparing and analyzing, choose the latter as the support proposal in mobility.

Key words: Linux kernel; IP address; timer; notification chain; mobility

0 引 言

科技是推动当今社会不断进步的巨大动力,而信息网络也在其中扮演举足轻重的作用。随着手机、PDA、笔记本电脑等移动终端的发展,用户对于目前网络的移动性支持要求越来越高。通常来讲,有两类移动性:节点移动性和服务移动性[1]。节点移动性是指网络节点的位置发生改变;而服务移动性是指服务的存储位置发生了改变。

近年来,业界提出了许多解决节点移动性的方案。比较典型的是:一、移动IP技术(MIPv4[2]和MIPv6[3]),它通过允许移动节点同时拥有两个IP地址(家乡地址和转交地址)来实现移动性;二、一体化网络中提出的身份和位置分离技术[4-6],它通过分离节点的身份标识与位置标识,及相应的分离映射机制来实现移动性[7]。

在服务移动性支持方面,业界提出了SIP(Session Initiation Protocol,会话初始协议)[8]。它通过代理和重定向请求到用户当前位置来支持用户移动性。

收稿日期:2014-03-25 **修回日期**:2014-06-26 **网络出版时间**:2015-01-20
基金项目:国家自然科学基金重点项目(61232017);中央高校基本科研业务费专项资金(2014YJS025);北京市自然科学基金项目(4122060)
作者简介:谢婉君(1990-),女,硕士研究生,研究方向为下一代互联网和路由技术。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20150120.2155.003.html

以上所讨论的三种解决终端和服务移动性的方案都有各自的缺点和不足:

移动 IP 存在三角路由、隧道封装和管理开销等问题,此外移动 IP 对服务质量,安全性,移动性类型的多样化支持等还有待提高[9];一体化网络利用身份位置分离机制虽然可以有效解决移动 IP 的部分缺陷,但是它要求移动终端要与接入路由器直接相连,这就不适用于广大使用 NAT 进行网络接入的设备。此外,一体化网络中更新映射信息对映射服务器压力也比较大;SIP 移动性方案中会话双方直接通信,不需要 IP 隧道封装,因此具有时延小、带宽效率高的优点。但是该方案具有很大的弊端,即在子网间切换时,用户终端的 IP 地址发生改变,因此无法保持 TCP 连接。

文中提出的方案不同于这三种,解决思路:利用一体化网络中提出的“连接标识”(CID, Connection IDentifier),通过连接标识管理服务,能够在协议栈中同时实现服务和主机的移动性。具体而言,当通信一方因为移动而改变了 IP 地址之后,Linux 内核能够迅速感知此变化,并且获取更新之后的 IP 地址信息。随后,通过修改通信双方在通信之初建立的套接字信息来更改 CID 的属性,实现连接的保持,最终实现移动性。整个工作都将在 Linux 内核协议栈中实现以支持不同的应用。

1 Linux 内核快速获取 IP 地址机制

如何在 Linux 内核[10]中快速获取 IP 地址,文中探索了两种方法:一是通过设定定时器的定时周期,频繁去探测保存有网口信息的关键结构体;二是利用内核通知链机制,使得当 IP 地址有所改变时,通知链能够自动感知并主动通知系统,进而将更新后的 IP 地址提交到上层应用。

1.1 定时检测网口信息

定时检测网口信息的含义是,利用 Linux 内核定时器[11],每隔一定时间去主动探测包含本地 IP 地址信息的内核中的结构体,将返回的 IP 地址与链表中已存在的 IP 地址进行比较,来确定 IP 地址是否改变。若改变,则将更新后的 IP 地址传送给上层,实现终端和服务的移动性。

Linux 内核中包含很多存储 IP 地址信息的结构体[12],它们之间相互调用,利用指针传递参数的现象非常频繁。最常用和最基本的三个结构体分别是网络设备结构 net_device,设备参数块 in_device,和 IP 地址结构 in_ifaddr。

内核定时器到期后,通过访问 net_device 结构体从而访问网络设备,对某一确定的网络设备,通过 in_device 查看参数块中的 IP 地址信息。获取 IP 地址之后,与链表中已储存的信息进行比对,判断地址是否发生改变,若改变,在链表中存储新的 IP 地址;若没有改变,则不作为。之后判断此网络设备是否是该主机的最后一个网络设备,若不是,则返回 net_device 继续查看下一个设备;若是,则重新设定定时器。依次循环。

该功能的伪代码和注释如下:

```
time_handler(data)    //定时器到期时的执行函数
{
mod_timer(&stimer,jiffies+HZ);    //修改定时器下一次到期时间
for(dev=first_net_device(&init_net);    //循环查询网络设备
{
list_for_each(ipaddr_chain);
if(p->ip==new_ipaddr)    //若出现新 IP,则保存
list_add(&listnode->list,&new_ipaddr);
list_for_each_safe(pos,t,&numhead.list);    //释放链表的指针
}
}
```

1.2 内核通知链主动推送

Linux 系统存在内核通知链机制,可以让某个子系统在发生某个指定事件的时候去通知对此感兴趣的其他子系统,并做出一定动作。考虑到 IP 地址改变带来的网口设备属性的改变是一种可以被探测的事件,因此可以作为通知者注册在内核通知链当中。之后当 IP 地址改变时,通知链发出信号,通知对此感兴趣的其他系统(即获取更新之后 IP 地址并将其传送到上层应用),以此实现 IP 地址更改消息主动推送。

通知链表是一个函数链表,链表上的每一个节点都注册了一个函数。当某个事情发生时,链表上所有节点对应的函数就会被执行。通知链的运作机制包括两个角色:

被通知者:对某一事件感兴趣一方。定义了当事件发生时,相应的处理函数,即回调函数。但需要事先将其注册到通知链中(被通知者注册的动作就是在通知链中增加一项)。

通知者:事件的通知者。当检测到某事件,或者本身产生事件时,通知所有对该事件感兴趣的一方事件发生。他定义了一个通知链,其中保存了每一个被通知者对事件的处理函数(回调函数)。通知这个过程实际上就是遍历通知链中的每一项,然后调用相应的事件处理函数。

通知链技术可以概括为:事件的被通知者将事件发生时应该执行的操作通过函数指针方式保存在链表(通知链)中,然后当事件发生时,通知者依次执行链表中每一个元素的回调函数完成通知。

在定义和初始化通知链之后,通知者定义通知链,即将 IP 地址改变或网络设备状态发生变化这一事件作为通知者;之后,被通知者调用 notifier_chain_register 函数注册回调函数,该函数按照优先级将回调函数加入到通知链中。此回调函数为通知者事件发生时,被通知者需要执行的函数或功能,即将新的 IP 地址传送到上层。此后,系统等待注册事件是否发生,若发生,则执行已注册的回调函数,否则继续等待注册事件的发生。

在这里,IP 地址改变之后,通知链需返回一个信号,主动告知地址改变,同时将新的地址传送到上层应用。

该功能实现的伪代码注释如下:

```
tcp_inetaddr_event( )
{switch (event) {
case NETDEV_UP:    //查看网络设备状态是否改变
sk_state_change(ptr);//若改变,执行 sk_state_change 回调函数
default:break;
}
return NOTIFY_DONE;
}
static struct notifier_block tcp_inetaddr_notifier={    //通知链进行通知
. notifier_call=tcp_inetaddr_event,
};
void __init tcp_v4_init(void){
register_inetaddr_notifier(&tcp_inetaddr_notifier); } //将事件注册进通知链
```

将 IP 地址改变这一事件作为通知者,注册到通知链中。每当 IP 地址改变或网络设备状态发生变化时,这一事件被触发,通知链随即返回一个信号,执行触发函数:sk_state_change( ),将新得到的 IP 地址发给通信对端,以维持通信的继续。

2　内核中的移动性实现

在服务器和客户端建立连接进行通信的过程中,双方都有可能发生移动问题。服务器可能因为某种原因改变了地理位置,从而改变了 IP 地址;客户端发生移动的可能性则更大。随着越来越多移动设备的出现,人们对移动性的支持性要求也越来越高。人们更希望在移动的过程中依然能够获得完整的服务。例如:在线观看视频时,无论自己如何移动,视频都能保证流畅连续的播放,而不是因为本身空间位置的改变带来连接的中断。

不同于移动 IP 借助于 CoA 来实现移动切换的技术,文中的做法是:在服务器和客户端建立点对点通信连接之后,若一方发生移动,造成 IP 地址改变,分别使用两种机制及时获取更新之后的 IP 地址信息,将原 IP 与现有 IP 一起封装到 socket[13] 包中,发送给通信对端。对端收到带有特殊标志 CID 的包后,修改自己的 socket 属性,将原 IP 换成更新之后的 IP 地址,重新发送。因只修改了 socket 五元组中的一个 IP 地址,套接字的其他属性不变,因此依然可以表示与刚才相同的链接,通信过程依然不会中断,且包的发送序号依次顺延,连接不会从头开始。

实验拓扑如图 1 所示。

图 1　实验环境网络拓扑结构

2.1　服务器端修改部分

系统移动性涉及到服务器和客户端两个部分,所以对内核源代码的修改也分为了服务器端和客户端。以服务器端移动为例,移动性实现具体有四个步骤:

(1)判断移动,获取更新后的 IP 地址;

(2)修改 TCP[14] 报头信息;

(3)封装原 IP 与更新后 IP;

(4)发送更新之后的 TCP 包。

为保证客户端可以区分服务器端修改 IP 地址之后发送的特殊包与普通 TCP 通信包,将 TCP 报头的 URG、ACK 和 PSH 位置 1,其他位置 0。即增加 TCPHDR_CID 值为 0X38。

为了使客户端能够识别哪一个通信对端的服务器地址发生改变,所以必须将原 IP 一同封装入 TCP 包中发送给客户端,同时,为了建立新的连接,也需要将更新后的 IP 地址一同发送[15]。

2.2　客户端修改部分

客户端代码的修改部分不多,主要功能是:

(1)通过 TCP 报文中的 CID 特殊报文来识别服务器发来的申请改变 IP 地址的包。

(2)获得原 IP 地址与服务器更新之后的 IP 地址之后,修改本地套接字属性[16] 重新发送 ACK 确认,保持连接不断。

此外,若想实现客户端的移动,只需将服务器端代码中的 sk_state_change 做少量修改即可。

因为服务器以特定的 80 端口来绑定客户端发出的套接字请求(HTTP 连接),所以在移动之后只需修改 80 端口的套接字属性。但是对于客户端来说,建立连接的端口号是不确定的,因此首先要寻找建立该连接的端口号。

3 程序测试与结果分析

文中以服务器端移动为例,通过使用网络视频播放器 MPlayer 观看视频及 wireshark 抓包程序验证了连接不断的结果,同时,也测试了相关移动切换所需的时延。

3.1 wireshark 抓包测试结果

服务器因移动而造成 IP 地址改变时,与客户端的通信不断,视频经过不到 1 s 的停顿后继续播放,下层 IP 地址改变对上层应用屏蔽,用户感觉不到连接中断。

使用 wireshark 进行抓包测试时,设置客户端的 IP 地址为 192.168.78.46,服务器端 IP 地址首先是 192.168.78.29,当移动之后,IP 地址改变为 192.168.78.30。

抓包结果显示,当改变 IP 地址的瞬间,客户端可以收到一个 TCP 包,报头中 PSH、ACK、URG=1。这是所发出的特殊标志包,在这个包里含有原来连接状态的新旧 IP 地址。客户端根据此包中的内容相应地修改自己套接字中的目的地址属性,重新发送 ACK 回应包,继续 HTTP 的通信。

3.2 时延分析

(1)两种机制获取更新后 IP 地址的时延分析。

通过手动修改 IP 地址,编写脚本记录下 IP 地址实际修改的时间,使用定时器机制在定时器分别设定为 200 ms、500 ms 和 1 000 ms 时获取更新后 IP 地址的时间,以及使用通知链机制获取更新后 IP 地址的时间。重复实验 20 次,绘制时延曲线如图 2 所示。

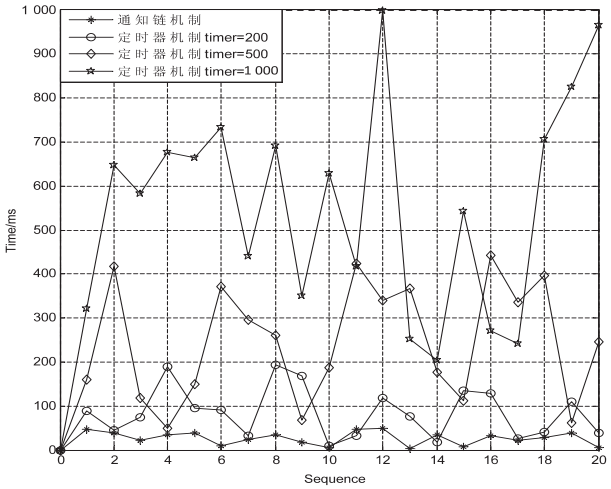

图 2 定时器机制与通知链机制获取 IP 地址时延

(2)两种机制恢复连接时延分析。

在测试了两种机制获取更新后 IP 地址的时延基础上,以收到含有 CID 的第一个包为标准,分别测试了两种机制恢复连接所需要的时延。由于定时器时间间隔设置过小时可能因指针释放不完全而引起内存泄露,所以定时器测试的条件是定时器设定为 500 ms。两种机制各自的时延情况如图 3 所示,对比图如图 4 所示。

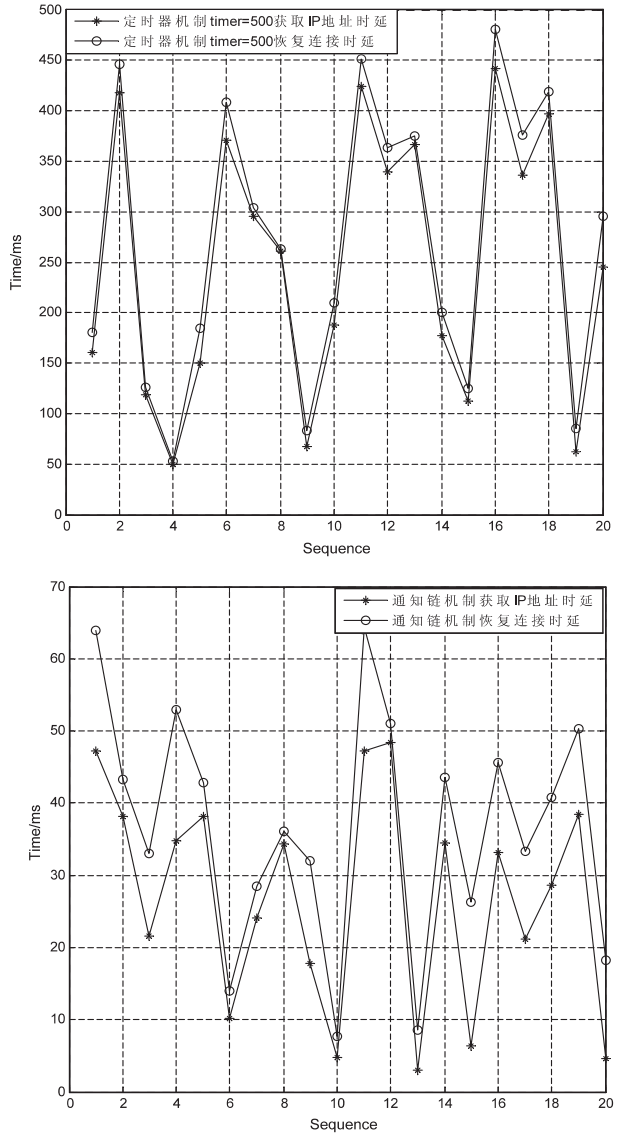

图 3 定时器机制与通知链机制恢复连接时延

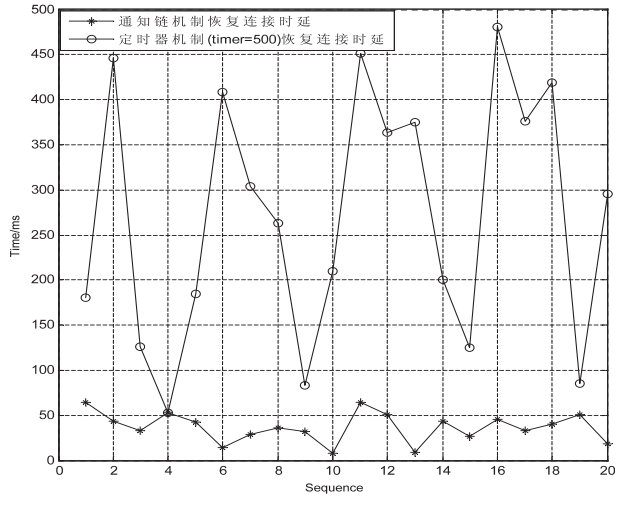

图 4 定时器机制和通知链机制恢复连接所需时延对比

由以上各图可以看出,利用 Linux 内核定时器去

读取包含本地 IP 地址的关键结构,是在 Linux 内核中实现对 IP 地址探测的最简单的方法。这种方法的优点是简单易行,并且用链表存储之前的地址可以记录 IP 地址的变化过程;缺点是依靠定时器到期后的执行函数,这是一种被动探测的行为。获取 IP 地址的时延与定时器时间的设定有线性关系,即使定时间隔可以缩短到很小,依然无法保证 IP 地址改变时能够实时地得到改变的信息。同时,实验证明,当定时器间隔很小时,由于链表指针可能释放不完全,有可能导致内存泄露,进而导致系统崩溃。

而利用通知链来获得 IP 地址的优点在于,这是一个主动推送的过程。每当 IP 地址发生改变时,通过通知链就可以非常及时地获取新的 IP 及恢复连接,时延保证在 100 ms 以下;而地址不变时,也不用反复对网络设备状态进行探测,避免给系统增加过多负担。因此更适合在 Linux 内核中支持移动性。

4　结束语

文中所完成的工作是在 Linux 内核中,使用内核定时器和内核通知链两种机制快速获取 IP 地址,并且在此基础上实现了服务器移动,IP 地址改变,而连接不中断的移动性测试。通过对两种机制在获取更新后 IP 地址及恢复连接的时延测试可以看出,通知链机制因其主动推送、及时有效、所占用资源较少而更符合在 Linux 内核中支持移动性的需求。

现代社会中,随着手机、笔记本电脑等移动终端的出现和使用,用户对于移动性支持的需求越来越高,传统网络由于 IP 地址的二重性已注定无法很好地实现移动性支持,这就要求在未来的新型网络架构中更好地去考虑移动性方案,使用户使用网络时,更加方便快捷。

参考文献:

[1] 陈山枝,时　岩,胡　博. 移动性管理理论与技术的研究[J]. 通信学报,2007,28(10):123-133.

[2] Perkins C. IP mobility support for IPv4[S]. RFC 3344,2002.

[3] Johnson D B, Perkins C, Arkko J. Mobility support in IPv6[S]. RFC 3775,2004.

[4] 张宏科,苏　伟. 新网络体系基础研究——一体化网络与普适服务[J]. 电子学报,2007,35(4):593-598.

[5] 董　平,秦雅娟,张宏科. 支持普适服务的一体化网络研究[J]. 电子学报,2007,35(4):599-606.

[6] 杨　冬,周华春,张宏科. 基于一体化网络的普适服务研究[J]. 电子学报,2007,35(4):607-613.

[7] 王　义. 一体化网络的一种移动切换机制的设计与实现[D]. 北京:北京交通大学,2008.

[8] Handley M, Schulzrinne A H. Session initiation protocol[S]. RFC 3261,2002.

[9] Perkins C E, Johnson D B. Route optimization for mobile IP[J]. Cluster Computing,1998,1(2):161-176.

[10] 肖宇峰,李　昕,时　岩. Linux 网络内核分析与开发[M]. 北京:电子工业出版社,2010.

[11] Love R. Linux kernel development[M]. 2nd ed. [s. l.]:Addison-Wesley Professional,2010.

[12] 严蔚敏,吴伟民. 数据结构[M]. 北京:清华人民出版社,1996.

[13] 李卓桓,瞿　华. Linux 网络编程[M]. 北京:机械工业出版社,2000.

[14] 罗　钰. 深入浅出 Linux TCP/IP 协议栈[M]. 北京:人民邮电出版社,2010.

[15] Bovet D P. 深入理解 Linux 内核[M]. 北京:中国电力出版社,2008.

[16] 陈莉君. 深入分析 Linux 内核源代码[M]. 北京:人民邮电出版社,2002.

(上接第 102 页)

[5] 袁超伟,张金波,姚建波. 三网融合的现状与发展[J]. 北京邮电大学学报,2010,33(6):1-8.

[6] 隋宗见,程春玲,崔国亮. 面向三网融合的综合网管系统的设计与实现[J]. 计算机技术与发展,2011,21(11):129-132.

[7] Wood D. Model behaviour for 3D. HDTV[J]. Electronics Letters,2010,46(15):1045-1047.

[8] Woo K S, Lee K I, Paik J H, et al. ADSFBC-OFDM for a next generation broadcasting system with multiple antennas[J]. IEEE Trans on Broadcasting,2007,53(2):539-546.

[9] Hoffmann H, Itagaki T, Wood D, et al. A novel method for subjective picture quality assessment and further studies of HDTV formats[J]. IEEE Trans on Broadcasting, 2008, 54(1):1-13.

[10] Lee G M, Lee C S, Rhee W S, et al. Functional architecture for NGN-based personalized IPTVservices[J]. IEEE Trans on Broadcasting,2009,55(2):329-342.

[11] 苏　伟,刘　琪,张宏科. 一体化标识网络体系及关键技术[J]. 中兴通讯技术,2011,17(2):1-4.

[12] 杨　冬,周华春,张宏科. 基于一体化网络的普适服务研究[J]. 电子学报,2007,35(4):607-613.

[13] 董　平,秦雅娟,张宏科. 支持普适服务的一体化网络研究[J]. 电子学报,2007,35(4):599-606.

[14] 张宏科,苏　伟. 新网络体系基础研究——一体化网络与普适服务[J]. 电子学报,2007,35(4):593-598.

Linux内核移动性支持机制与实现

作者： 谢婉君， 贾濡， XIE Wan-jun， JIA Ru
作者单位： 北京交通大学 电子信息工程学院,北京,100044
刊名： 计算机技术与发展 ISTIC
英文刊名： Computer Technology and Development
年，卷(期)： 2015(3)