ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.2, February 2006, pp.275−284
DOI: 10.1360/jos170275

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于通用 PC 架构的高精度网络时延测量方法*

黎文伟[1]，张大方[2+]，谢高岗[3]，杨金民[2]

[1](湖南大学 计算机与通信学院,湖南 长沙 410082)

[2](湖南大学 软件学院,湖南 长沙 410082)

[3](中国科学院 计算技术研究所 信息网络室,北京 100080)

A High Precision Approach of Network Delay Measurement Based on General PC

LI Wen-Wei[1], ZHANG Da-Fang[2+], XIE Gao-Gang[3], YANG Jin-Min[2]

[1](College of Computer and Communication, Hu'nan University, Changsha 410082, China)

[2](School of Software, Hu'nan University, Changsha 410082, China)

[3](Network Research Division, Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100080, China)

+ Corresponding author: Phn: +86-731-8821980, Fax: +86-731-8821977, E-mail: dfzhang@hnu.cn, http://ss.hnu.cn/rjxy/

Li WW, Zhang DF, Xie GG, Yang JM. A high precision approach of network delay measurement based on general PC. *Journal of Software*, 2006,17(2):275−284. http://www.jos.org.cn/1000-9825/17/275.htm

Abstract: Delay is the foundation for accurately measuring network performance metrics such as delay jitter, bandwidth, etc. While the delay measurement methods currently used have poor precision, for there exist clock errors and location errors. In this paper, an improved method for delay measurement is proposed. It replaces system clock with TSC (time stamp counter) register as time-stamping to eliminate clock errors, and it removes the time-stamping place from application to network driver to eliminate location errors. The precision has been elevated largely. Experiments show that when comparing with traditional methods under different packet lengths, the improved method can reduce measurement errors 21%~150%, and the measured delays are more stable. Furthermore, it basically has no effect on system throughput. The improved measurement method is based on general PC, so it has lower measurement cost and can be applied widely.

Key words: delay measurement; network measurement; active measurement; TSC (time stamp counter)

摘 要: 时延是准确测量时延抖动、带宽等网络性能指标的基础.目前的时延测量方法由于存在时钟误差和位置误差因而精度较差.提出一种改进的时延测量方法,以 TSC(time stamp counter)寄存器取代系统时钟计时来消除测量的时钟误差,将时间戳记录位置由应用程序转移到网卡驱动来消除位置误差,极大地提高了时延测量精度.实验结果表明,与传统方法相比,不同包长度下,所提出的方法可降低测量误差 21%~150%,且测量结果稳定,对系统吞吐量基本无影响.该方法基于通用 PC 架构,测量成本低,适于普遍采用.

关键词: 时延测量;网络测量;主动测量;TSC(time stamp counter)

中图法分类号: TP393 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant Nos.60273070, 60403031, 60473031 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2005AA121560 (国家高技术研究发展计划(863)), the Natural Science Foundation of Hu'nan Province of China under Grant No.05JJ30116 (湖南省自然科学基金)

Received 2005-07-04; Accepted 2005-08-15

时延是网络的固有属性之一,也是评价网络性能的基本指标.时延测量在网络性能监测[1]、网络行为分析[2]、网络应用设计[3]等领域有着广泛的应用,同时,也是测量时延抖动[4]、网络带宽[5]等性能指标的基础.因其重要性,IETF 组织的 IP 网络性能指标工作组(IP Performance Metrics Working Group,简称 IPPM)对时延定义进行了标准化,将其分为往返时延[6]和单向时延[7].文献[8]提出了基于组播的时延测量方法,文献[9,10]提出了基于 TCP 的往返时延测量方法,文献[11]对消除单向时延测量中的时钟偏差进行了研究,文献[12−14]对网络路径、协议及设备的时延特征分别进行了测量和分析,文献[15]对以时延测量为主的网络测量技术进行了总结.这些研究主要涉及时延的定义、测量方法及网络路径或设备的时延特征,没有分析时延测量的精度问题.网络时延通常在数十到数百毫秒范围内,测量存在几毫秒的误差完全可以接受.通用的 PC 软/硬件架构无须任何改动,已可以达到毫秒级测量精度.故单就时延而言,测量精度不是主要问题.但对于时延抖动、带宽等性能指标,其测量准确性完全依赖于精确的时延差值或精确的包发送、接受时间间隔.研究高精度的时延测量方法,对于这些指标的测量至关重要.

时延测量主要有两个误差来源[16]:一是测量主机系统时钟的分辨率、偏移及抖动等影响测量计时精度而产生的误差,这里称为时钟误差;二是由于测量记录的收/发包时间戳不是真实的收/发包时刻而产生的测量误差,这里称为位置误差.这两类误差都是在测量主机的过程中产生的,而测量时延、时延抖动、带宽等指标的目的在于了解网络状况和性能,测量结果若包含了由主机所带来的测量误差,将使网络性能评价不准确.故必须尽量消除测量主机所产生的时钟误差和位置误差,提高时延测量精度.

网络时间协议(network time protocol,简称 NTP)[17]可用作测量主机的时钟源,并使测量主机时钟同步.但其时钟准确性部分依赖于同步主机间的路径时延,而这正是时延测量的测量对象.故 NTP 协议不宜作为时延测量的时钟源.文献[18]引入如全球定位系统(global position system,简称 GPS)的外部时钟源,可以消除测量的时钟误差,其时钟精度可达 1μs.但 GPS 系统价格昂贵,且测量主机与 GPS 天线连接距离不能太长,测量主机部署位置受限;此外,位置误差仍然没有消除.文献[19]设计了专用于包捕获的 DAG 卡.它可以在系统收到数据包的同时记录时间戳,从而消除测量的位置误差.DAG 卡与 GPS 系统集成,可以同时消除测量的时钟误差和位置误差,实现高精度的时延测量[20].但是,这种方式所需硬件价格昂贵,难以在诸如 PingER[1],AMP[21]等大规模网络测量系统中普遍部署.这些系统包含数百个测量主机,均采用通用 PC 和 Linux 或 FreeBSD 等源码开放操作系统.在当前通用 PC 硬件性能已得到极大提升的背景下,研究基于通用 PC 架构的高精度时延测量方法对实现成本低廉、实施方便、测量精确的大规模网络性能监测和网络行为分析具有重要意义.

针对时延测量的两个误差来源,本文提出一种基于通用 PC 架构的高精度时延测量方法.这种方法以 CPU 内部的时间戳计数器(time stamp counter,简称 TSC)取代操作系统时钟计时,具有分辨率高、稳定、额外开销小的特点,消除了测量的时钟误差;通过修改操作系统内核,将时间戳记录位置从测量程序转移到网卡驱动,基本消除了位置误差.由于是基于通用 PC 架构实现的,因此测量成本低,适于普遍采用.与传统时延测量方法相比,包长为 64 字节时,该方法降低测量误差近 150%,且测量结果稳定,对系统吞吐量基本无影响.具体实现时,使用 Intel 公司的奔腾Ⅳ CPU 和 Linux 操作系统.

1 基于 TSC 的时钟误差消除

1.1 系统时钟分析

通用 PC 自带两个时钟源:硬件时钟和软件时钟(或称为系统时钟).访问硬件时钟开销较大,系统运行时通常不使用.软件时钟是调度系统任务和用户任务的计时依据,Linux 的系统调用 gettimeofday()可以读取到,它依据定时器中断计数来完成计时.定时器中断通过将标准石英晶体振荡器(quartz oscillator)频率为 1.19318MHz 的输出信号除以 100Hz 产生.故软件时钟分辨率较差,仅为 10ms.Linux 内核以 CPU 的 TSC 寄存器在定时器中断间内插计时来提高软件时钟分辨率.TSC 寄存器计数自系统启动至今的 CPU 时钟周期数,将其用于计时,需要知道准确的 CPU 频率或时钟周期大小.Linux 在系统启动时,有 50ms 的校准时间来计算 CPU 频率,并用于后续的 TSC 计数转换.受温度变化、芯片老化等因素影响,CPU 频率可能会有小幅度的变化,TSC 的计数速率不是固定

的;此外,50ms 的校准时间来自系统时钟.故这种方式仍有较大误差.因为有系统调用 gettimeofday()的开销限制,插值后软件时钟分辨率最多提高到 1ms,不能满足时延测量的时钟要求.

1.2 基于TSC的时钟误差消除

TSC 寄存器是 Intel 奔腾系列 CPU 独有的 64 位计数器.在系统启动时计数器清零.之后,每过一个时钟周期自动增加 1,其数值可以通过 rdtsc 汇编指令读出.作为时钟使用,TSC 寄存器的时钟分辨率取决于 CPU 频率.若 CPU 频率为 f_{CPU},则 TSC 时钟分辨率为 $1/f_{CPU}$.目前,主流 CPU 频率已超过 1GHz,以 TSC 寄存器为时钟,可达纳秒级的时钟分辨率,计时精度高.此外,读取 TSC 寄存器计数只需 1 条汇编指令,节省开销,进一步减小了时钟误差.故 TSC 寄存器是高精度时延测量的理想时钟源.

用 TSC 寄存器计时,最关键的问题是如何精确地估计 CPU 频率或时钟周期.因为 CPU 频率可能受温度变化、芯片老化等因素的影响而小幅抖动,必须对其进行周期性估计.精确估计 CPU 频率需要有高精度的时钟源作参考,简单的方法是外接如 GPS 等高精度的时钟源作为参考时钟来估计.但本文的目标是仅依靠廉价的 PC 软/硬件架构来实现高精度的时延测量,故不考虑 GPS 等昂贵的外部时钟源.

1.2.1 CPU 频率估计方法

网络中已存在很多以 GPS 进行时钟同步的高精度 NTP 服务器[22],这些 NTP 服务器计时精度往往为次纳秒级[23].但因网络时延的存在和变化,直接使测量主机时钟与 NTP 服务器时钟同步存在较大误差,难以准确估计 CPU 频率.事实上,准确估计 CPU 频率的关键是获得较精确的一个时间区间,并记录这个时间区间起止时刻的 TSC 计数.基于这一考虑,并结合网络路径最小时延特性,本文提出了一种以高精度 NTP 服务器为间接时钟源的 CPU 频率估计方法.方法过程如图 1 所示.

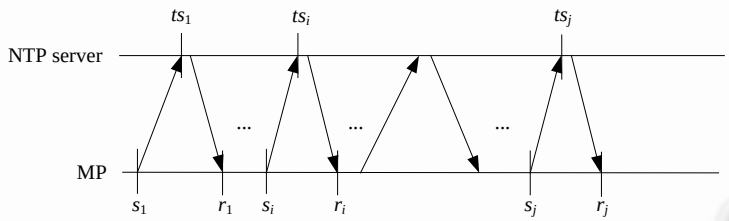

Fig.1 The CPU frequency estimation process

图 1 CPU 频率估计过程

MP 为进行时延测量的 PC.当估计 CPU 频率时,MP 向 NTP 服务器发送 NTP 请求包,并记录 TSC 寄存器计数作为包发送时间 *s*.NTP 服务器收到请求后,将记录请求收到时间戳 *ts*,并将其写入 NTP 包返回给 MP;MP 收到 NTP 响应包后记录 TSC 寄存器计数,作为包收到时间 *r*,此时 MP 可以获得一个到 NTP 服务器的往返时延测量值 *rtt*,其单位是 CPU 时钟周期,以及一个 NTP 服务器收到 NTP 请求包的时间戳 *ts*.MP 以较小周期(如每隔 1s)重复这一过程,经过一段时间后,MP 可以测量到数个 *rtt* 的最小值.最小时延是数据包在网络的传输时延及信号在物理链路的传播时延之和,是由网络拓扑及传输技术引起的固有时延.它不受网络负载影响,其大小是固定不变的.因而对于两个连续测量到的最小往返时延测量包,它们从 MP 到 NTP 服务器的单向时延也相等.如图 1 所示,设第 *i*, *j* 个请求包测量到最小 *rtt*,且它们之间的请求包均没有测量到最小 *rtt*,有

$$ts_i - s_i / f_{CPU} = ts_j - s_j / f_{CPU},$$

$$(s_j - s_i) / f_{CPU} = ts_j - ts_i.$$

即两个请求包在源端的发送时间间隔等于在目的端的接收时间间隔.故 CPU 频率 f_{CPU} 可通过下式进行估计:

$$f_{CPU} = (s_j - s_i)/(ts_j - ts_i) \quad (1)$$

文献[14,24]等研究发现,无论是网络路径还是路由器,短时间内测量到其最小时延的概率非常大.故 MP 小时间间隔地重复上述最小时延测量过程,即可持续地估计 CPU 频率.估计时可选择往返时延较小的 NTP 服务器,以进一步减少网络的影响.

1.2.2 性能比较

实验比较了 TSC 寄存器计时与 gettimeofday()调用计时的性能.实验时关闭了系统的后台任务和进程,以避免对结果的影响.实验 PC 配置为:CPU Intel 奔腾Ⅳ 1.0G,256M 内存,操作系统为 REDHAT Linux7.2.两种计时方式的时间开销见表 1.表中数据为分别重复两种计时方式 100 次后取平均值得出来的.显然,gettimeofday()调用的时间开销比 TSC 寄存器计时高一个数量级,导致测量时钟误差较大.

Table 1 The time cost of TSC and gettimeofday methods

表 **1** TSC 计时与 gettimeofday 计时的时间开销

Timing method	Cost (ns)
TSC	37
gettimeofday	430

由于缺乏精确的外部时钟源,假设关闭系统的后台任务和进程后,系统执行从 1~1 000 的累加计算所需时间是不变的.长时间重复这一累加计算,并分别以两种计时方式记录每次累加计算所需时间,以比较两种计时方式的稳定性和准确性.图 2 为 24h 内记录的累加计算所需时间,重复时间间隔为 1min.TSC 寄存器计时得到的时间开销低于 gettimeofday()计时所得.因无法知道累加计算需要的确切时间,不能就此确定 TSC 寄存器计时比 gettimeofday()精确.系统轻载稳定条件下,可以认为累加计算的时间开销不变.随着时间推移,用 gettimeofday()计时所得到的时间开销出现增长趋势,表明系统时钟出现时钟偏移,计时不准确.gettimeofday()计时除了具有定向的偏移趋势以外,还有较大的随机抖动.因此,对于高精度的时延测量而言,gettimeofday()计时会带来较大的系统误差和随机误差.TSC 寄存器计时得到的时间开销则基本保持平稳,表明 TSC 寄存器计时稳定,没有时钟偏移;随机误差也较小,约为数十纳秒.故 TSC 寄存器计时可基本消除测量中的时钟误差.

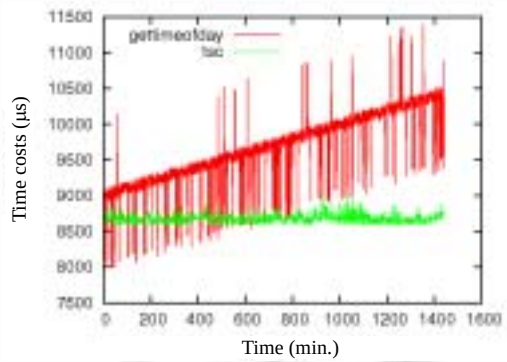

Fig.2 The time series of the cost of accumulative adding computation

图 2 累加计算的时间开销时序

2 基于时间戳位置转移的位置误差消除

2.1 位置误差分析

如图 3 所示为 Linux 系统的 UDP 数据包处理流程,TCP 数据包处理流程与之类似.数据包在应用程序和物理链路之间的来回传递,经历了系统内核的多层处理.在应用程序记录包发送时间戳,不是测量包被传送到物理链路的真实发送时刻,而是比它提前了一段时间;应用程序记录的包接收时间戳也不是测量包从物理链路接收到网卡的真实时刻,而是比它滞后了一段时间.目前的时延测量程序通常在应用程序记录测量包的发送时间戳和接收时间戳,计算网络时延,即在图 3 中 ts1 位置记录包发送时间戳,在 ts5 位置记录包接收时间戳,以二者之差作为时延测量值.由此产生了位置误差.它实际上是系统内核完成数据包发送和接收所需要的处理时间.

位置误差受机器配置、内核实现、系统负载、进程调度等因素的影响.如文献[25]测量其为数百微秒,文献[26]测量其为数十微秒.在大流量导致主机出现“中断活锁”的极端情况下[27],位置误差可达数百毫秒甚至无穷

大.此时,主机网卡收到了数据包,但因系统没有足够的资源处理,网卡只能推迟数据包向上层的提交,甚至无法提交.

2.2 位置误差消除

消除位置误差可行的做法是,使测量的发/收包时间戳尽量靠近测量主机真实发/收包时刻.新版本的 tcpdump 等流量监测工具实现了在 Linux 系统的设备独立层,即图 3 中 ts4 位置记录收包时间戳.这减少了收包的位置误差,但发包位置误差仍未消除,位置误差仍然存在.本文对 Linux 系统内核网络处理模块进行修改,提出在网卡驱动中记录发/收包时间戳的时延测量方法,即在图 3 中的 ts2 位置记录发包时间戳,在 ts3 位置记录收包时间戳.这是系统内核软件中最接近真实发/收包时刻的位置,基本消除了测量中的位置误差.

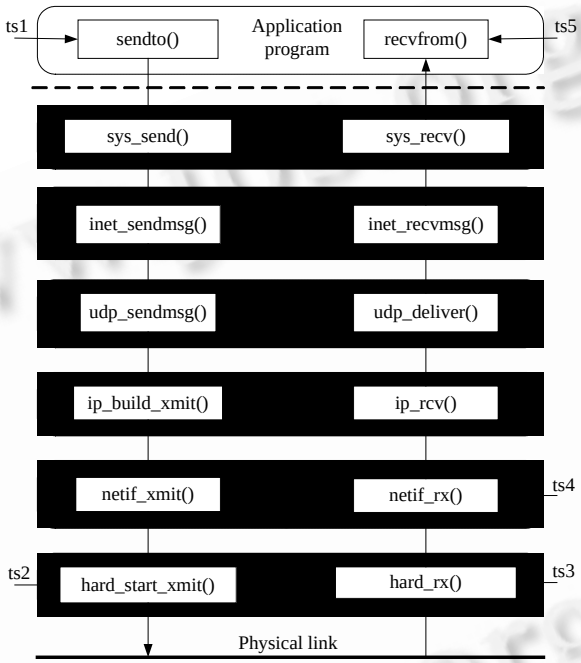

Fig.3 The UDP packet treatment process of Linux

图 3 Linux 系统的 UDP 数据包处理流程

2.2.1 收包位置误差消除

消除收包的位置误差主要分两个步骤:第 1 步是在网卡驱动中记录收包时间戳;第 2 步是将记录的收包时间戳传递给测量程序.

网卡驱动中记录收包时间戳较为简单,在网卡驱动的数据包接收函数 hard_rx()的开始部分加入计时语句即可.这样,网卡在处理接收到的数据包时将首先记录收包时间戳.如何将记录的收包时间戳传递给测量程序需要更多分析.网卡收到数据包时,将分配一个 struct sk_buff 类型(include\linux\skbuff.h)的缓存 skb 保存该数据包及一些相关信息,其中包括一个 struct timeval 类型的成员 stamp 用于保存数据包接收时间戳.在未修改的数据包接收函数中,stamp 并未赋值,因此,在数据包接收函数的末尾,可将之前记录的收包时间戳复制到缓存 skb 的 stamp 成员中.在设备独立层的 netif_rx()函数(net\core\dev.c)中,系统会对接收到的数据包打时间戳并保存在 skb 结构的 stamp 成员,这一动作会覆盖之前网卡驱动记录的收包时间戳,需要调整.方法是:在 netif_rx()给数据包打时间戳之前,先判断 skb 结构的 stamp 成员是否为空,若不为空,表明网卡驱动已记录了包接收时间戳,无须再记录;否则,对数据包打时间戳并保存在 skb 结构的 stamp 成员中.之后,skb 结构将逐层向上传递,并连接到 struct sock 结构 sk(include\net\sock.h)中.sk 结构在应用程序调用 socket()创建套接字时被创建,在套接字关闭前一直存

在.它保存了数据包的传输协议、本地 IP 地址、目的地 IP 地址等协议信息,还包括一个 struct timeval 结构的成员 stamp,用于保存数据包的接收时间戳.skb 缓存的 struct timeval 结构成员 stamp 最终被复制到 sk 结构的成员 stamp 中.应用程序创建套接字后,系统提供了一个 ioctl()调用,可对创建的套接字进行 I/O 控制.该调用有一个 SIOCGSTAMP 的调用参数,用于读取 sk 结构中的 stamp 成员,即数据包接收时间戳.对系统内核进行上述改动后,时延测量程序在接收到数据包后,以 SIOCGSTAMP 为参数调用 ioctl(),即获得之前在网卡驱动中记录的收包时间戳.这样,时延测量程序记录的数据包收到时间,将是系统在图 3 中 ts3 位置打的时间戳,基本消除了收包的位置误差.

2.2.2 发包位置误差消除

消除发包时的位置误差同样分两个步骤:第 1 步是在网卡驱动中记录发包时间戳;第 2 步是将记录的发包时间戳传递给测量程序.

网卡驱动中记录发包时间戳比较简单,在网卡驱动的数据包发送函数 hard_start_xmit()的开始部分加入计时语句即可.这样,网卡发送数据包时将首先记录发包时间戳.如何将记录的发包时间戳传递给测量程序则更为困难.首先,系统内核用于存储数据包的 struct sock 结构缓存 sk 和 struct sk_buff 结构缓存 skb 中,没有定义数据成员保存数据包发送时间戳;其次,在网卡成功发送数据包后,struct sk_buff 结构缓存 skb 将在 ip_build_xmit()函数中被释放,因而在网卡驱动记录的发包时间戳也被销毁;最后,系统也没有提供与读取收包时间戳类似的 ioctl()调用来获得发包时间戳.

一种可能的解决方案是:测量程序创建一个内存区域与网卡驱动共享,程序对发送的测量包做特殊标记.网卡驱动识别到有该标记的数据包就记录包发送时间戳,并将该标记与包发送时间戳存入共享内存中.测量程序通过访问共享内存中的记录来获取包发送时间戳.这种方式要保证网卡和应用程序的读写同步,需要网卡进行额外的判断运算,降低了网卡性能.寻找合适的测量包标记也存在困难.用户空间和内核空间的内存共享实现复杂且影响系统效率.因此,受系统收包时间戳传递机制的启示,对系统内核进行少许改动,提出了一个简单、高效率的发包时间戳从网卡驱动向测量程序传递的机制.

首先,修改 struct sock 和 struct sk_buff 结构类型定义,分别在两个结构中增加 struct timeval 结构类型的成员 sndstamp,用于保存包发送时间戳.然后,可以在网卡驱动的数据包发送函数 hard_start_xmit()记录包发送时间戳,并保存到缓存 skb 的成员 sndstamp 中.包发送成功,系统返回到 ip_build_xmit()函数时,skb 缓存将被释放,所以,在 ip_build_xmit()函数中,将 skb 的 sndstamp 成员复制到缓存 sk 的 sndstamp 成员中.最后,修改 ioctl()系统调用(net/ipv4/af_inet.c),增加一个调用参数 SIOCGSNDSTAMP,带该参数的 ioctl()调用,将缓存 sk 的 sndstamp 成员即网卡驱动记录的数据包发送时间戳复制给应用程序.这样,时延测量程序记录的数据包发送时间是系统在图 3 中 ts2 位置打的时间戳,基本消除了发包的位置误差.

3 完整实现方案

综合上面的分析,对时延测量方法主要进行了两方面的改进:一方面是以 tsc 计数器取代 gettimeofday()系统调用计时;另一方面是对系统内核进行修改,将收/发包时间戳记录位置转移到网卡驱动中.这种方案基于通用 PC 架构,测量成本低;消除了时延测量中的时钟误差和位置误差,极大地提高了时延测量的精度.与传统时延测量过程(如图 4(a)所示)相比,测量程序的流程发生了改变.如图 4(b)所示,时间戳记录由网卡驱动完成,应用程序以 ioctl()调用替代 gettimeofday()读取记录的时间戳;发送时间戳在测量包发送后读取;最后,根据当前所测 CPU 频率来计算网络时延.定期估计 CPU 频率的过程会增加程序实现复杂度.为此,可将 CPU 频率估计过程单独实现,作为系统后台进程运行;在计算网络时延时,测量程序从后台进程获取当前 CPU 频率估计值.估计 CPU 频率时,同样以网卡驱动记录的时间戳计算最小时延,以减少误差.

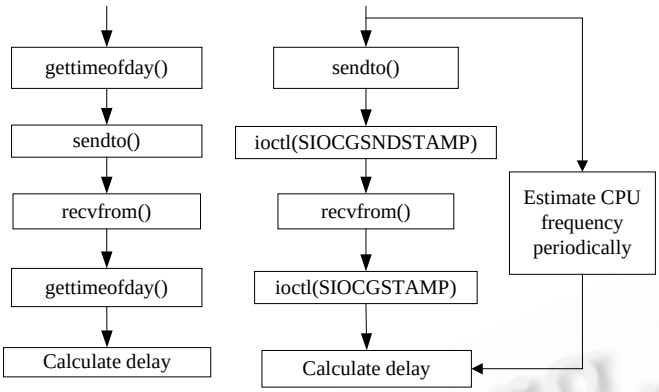

Fig.4 Comparison of delay measurement process

图 4 时延测量过程比较

4 实验评估

对 Linux 内核源码进行修改和重新编译,并编程实现了如图 4(b)所示的时延测量过程,以下称为 tsc-ping.比较其与传统时延测量方法如 ping 的性能差异.为避免网络负载引起的时延变化对实验评估有影响,将两台同配置 PC 机以双绞线直连,一台作为测量主机,另一台作为目的主机,并关闭两台机器的不必要的后台进程.网卡传输速率已知,如果忽略主机的包处理开销,实验环境下的理想往返时延则可以根据包长和传输速率计算.分别以 tsc-ping 和 ping 测量两台 PC 之间的往返时延并与理想值比较,以评价两种方法的测量精度.PC 配置为:CPU Intel 奔腾Ⅳ 1.0G,256M 内存,100Mbps 以太网卡;操作系统为 REDHAT Linux7.2.双绞线长度为 1m,由此产生的信号传播时延仅为 1ns,可忽略不计.

4.1 测量精度比较

实验测量的往返时延实际上包含了目的主机接收和响应测量包的处理时间.为准确计算往返时延测量值与理想值的误差,在目的主机做了同样的内核修改,使之记录测量包的接收、发送时间戳,计算目的主机对测量包的处理时间.以测量的往返时延减去目的主机的测量包处理时间为最终结果,计算测量误差.表 2 为在几种典型数据包长度下的往返时延理想值,以及 tsc-ping 和 ping 得到的测量值.在各种数据包长度下,ping 与理想值都有较大误差.包长为 64 字节时甚至达到 200%的误差,1500 字节的最大包长时仍有 25.6%的误差.tsc-ping 误差较低,包长为 1 500 字节时,误差仅为 4.6%,比 ping 的测量误差低了 21%.故 tsc-ping 测量精度远高于 ping.由于实验是在双机直连的小时延极端环境下进行的,可以预计,在时延较大的实际网络环境中,tsc-ping 的测量误差将更小.故 tsc-pin 极大地提高了时延测量精度,可以较为准确地测量网络时延.

Table 2 The measured values (μs) and errors of ping and tsc-ping

表 2 tsc-ping 和 ping 的测量值(μs)与测量误差

Packet length (Byte)	Ideal value	ping		tsc-ping	
		Average	Error (%)	Average	Error (%)
64	20.5	61.5	200	31.7	54.6
512	163.8	243.4	48.6	185.3	13.1
1 500	480	603.7	25.6	502.1	4.6

4.2 测量稳定性比较

如图 5 所示为 24 小时的往返时延测量时序,包长为 1 500 字节.其他包长度的结果与之类似.测量在直连的两台轻载 PC 上完成,往返时延实际值不会有太大抖动,故图中的理想往返时延曲线代表实际往返时延.tsc-ping

所测量的往返时延比理想值稍高,但整体较平稳,标准差约为 5μs.这表明,tsc-ping 有较小的系统误差和随机误差,测量结果误差小且稳定.而 ping 所测量的往返时延则比理想值高 100 多 μs,且抖动较大,标准差达 50μs.这表明 ping 除了系统误差较大之外,还有较大的随机误差,测量结果误差大且不稳定.过大的测量系统误差和随机误差导致难以准确地分析网络性能和网络状况.

图 6 为不同 CPU 负载下,tsc-ping 和 ping 测量的 rtt 平均值及标准差.由图 6 也可知,ping 有较大的系统误差和随机误差.随着 CPU 负载的增长,ping 的系统误差和随机误差亦同步增长.在进行大规模网络测量时,通常一台主机同时要对数十台目的主机进行测量,系统负载较大,显然,这种场合下 ping 很难准确地测量网络时延.随着 CPU 负载的增长,tsc-ping 则一直保持较低的系统误差和随机误差不变;其测量误差与 CPU 负载无关.与 ping 等传统方法相比,tsc-ping 测量结果稳定,不受系统负载影响,在主机测量负载较重时,仍能保持较高的测量精度.

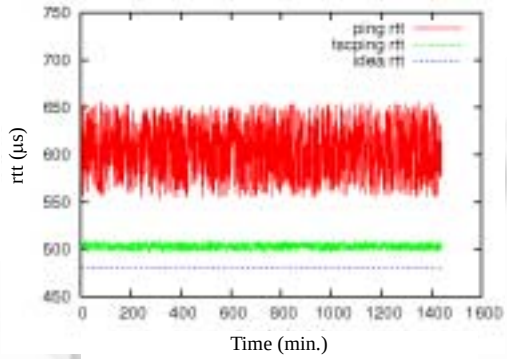

Fig.5 The time series of round trip delay (packet length: 1 500 Byte)

图 5 往返时延时序(包长为 1 500 字节)

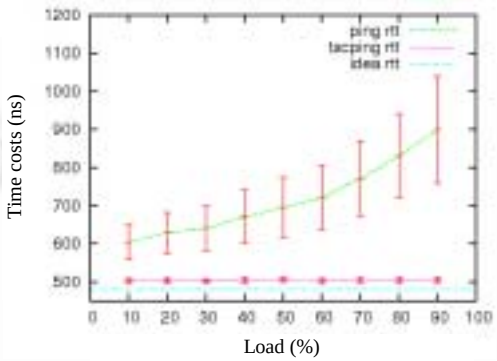

Fig.6 The relation between CPU load and measured RTT

图 6 CPU 负载与 RTT 测量值关系

4.3 对系统网络吞吐量影响分析

本文提出的时延测量方法 tsc-ping 对系统内核进行修改,以提高测量精度.需要评估这些修改是否影响了系统的网络吞吐量,结果如图 7 所示.图 7 为不同包长度下,系统内核修改前后的系统网络吞吐量对比.当包长较小时,内核修改影响系统吞吐量,但影响不大.64 字节包长时,系统吞吐量仅降低约 3Mbps.而在包长度超过 1 000 字节后,吞吐量降低不到 0.5%,内核修改已基本不影响系统吞吐量.考虑到实际测量时,测量包的长度是可以指定的,而且时延抖动、带宽测量等通常选择较大的测量包长度,故在实际使用时,本文提出的时延测量方法 tsc-ping 不会显著降低系统网络吞吐量.

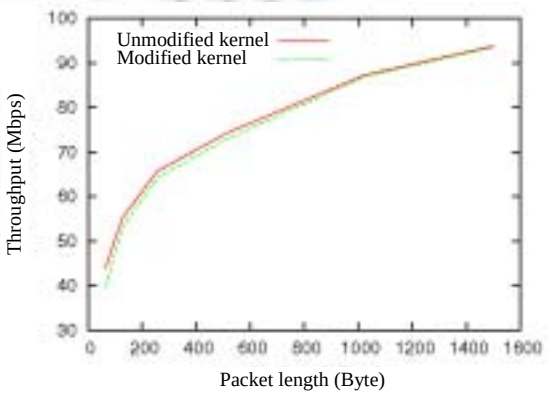

Fig.7 The throughputs of system before and after kernel modification

图 7 系统内核修改前后的吞吐量

5 结 论

时钟误差和位置误差是目前时延测量精度不高的主要原因.基于通用 PC 软硬件架构,本文讨论了如何消除这两种误差,提出了一种成本低廉、实施方便的高精度时延测量方法,适于普遍应用.实验表明,该方法极大地提高了测量精度,保证了测量结果的稳定性,同时对系统吞吐量影响很小.

目前,许多带宽测量工具在模拟实验中精度较高,但在实际互联网环境中则出现较大误差.下一步的工作主要是将本文提出的时延测量方法集成到目前常用的带宽测量工具中,提高其测量精度,以准确测量网络带宽.

References:

[1] Matthews W, Cottrell L. The PingER project: Active Internet performance monitoring for the HENP community. IEEE Communication Magazine, 2000,38(5):130−136.

[2] Paxson V. End-to-End internet packet dynamics. IEEE/ACM Trans. on Networking, 1999,7(3):277−292.

[3] Francis P, Jamin S, Cheng J. IDMaps: A global internet host distance estimation service. IEEE/ACM Trans. on Networking, 2001,9(5):525−540.

[4] Li WW, Wang JF, Xie GG, Zhang DF. An IPDV measurement method based-on packet-pair sampling. Journal of Computer Research and Development, 2004,41(8):1354−1360 (in Chinese with English abstract).

[5] Lai K, Baker M. Measuring bandwidth. In: Proc. of the IEEE INFOCOM'99. New York: IEEE Communication Society Press, 1999. 235−245.

[6] Almes G, Kalidindi S, Zekauskas M. A round-trip delay metric for IPPM. RFC2681, 1999.

[7] Almes G, Kalidindi S, Zekauskas M. A one-way delay metric for IPPM. RFC2679, 1999.

[8] Lu GH, Sun SX, Sao ZL, Zhang Y. Multicast-Based measurement of network delay. Journal of Software, 2001,12(11):1704−1709 (in Chinese with English abstract).

[9] Hao J, Dovrolis C. Passive estimation of TCP round-trip times. ACM SIGCOMM Computer Communication Review, 2002,32(3): 75−88.

[10] Horneffer M. Assessing Internet performance metrics using large-scale TCP-syn based measurements. In: Proc. of the Passive and Active Measurement Workshop 2000 (PAM 2000). Hamilton: University of Waikato, 2000.

[11] Wang JF, Yang JH, Zhou HX, Xie GG, Zhou MT. Detecting clock dynamics in one-way delay measurement. Journal of Software, 2004,15(4):584−593 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/584.htm

[12] Bolot J. Characterizing end-to-end packet delay and loss in the Internet. In: Proc. of the ACM SIGCOMM'93. New York: ACM Press, 1993. 289−298.

[13] Lin Y, Wang CG, Wang WD, Cheng SD. A delay analysis of RMTP. Journal of Software, 2002,13(8):1710−1717 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1710.pdf

[14] Papagiannaki K, Moon S, Fraleigh C. Measurement and analysis of single-hop delay on an IP backbone network. IEEE Journal on Selected Areas in Communications, 2003,21(6):908−921.

[15] Zhang HL, Fang BX, Hu MZ, Jiang Y, Zhan CY, Zhang SF. A survey on Internet measurement and analysis. Journal of Software, 2003,14(1):110−116 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/110.htm

[16] Paxson V, Almes G, Mahdavi J. Framework for IP performance metrics. RFC2330, 1998.

[17] Mills DL. Network time protocol (version 3) specification, implementation and analysis. RFC1305, 1992.

[18] Georgatos F, Gruber F, Karrenberg D. Providing active measurements as a regular service for ISP'S. In: Proc. of the Passive and Active Measurement Workshop 2001 (PAM 2001). Amsterdam: RIPE NCC, 2001.

[19] Micheel J, Donnelly S, Graham I. Precision timestamping of network packets. In: Paxson V, ed. Proc. of the 1st ACM SIGCOMM Workshop on Internet Measurement. New York: ACM Press, 2001. 273−277.

[20] Alves M, Corsello L, Karrenberg D. New measurements with the RIPE NCC test traffic measurement setup. In: Proc. of the Passive and Active Measurement Workshop 2002 (PAM 2002). Fort Collins: Agilent, 2002.

[21] McGregor T, Braun HW, Brown J. The NLANR network analysis infrastructure. IEEE Communications Magazine, 2000,38(5): 122−128.

[22] Stratum one time servers. 2006. http://ntp.isc.org/bin/view/Servers/StratumOneTimeServers

[23] Levine J, Mills D. Using the network time protocol (NTP) to transmit Int'l atomic time (TAI). In: Proc. of the 32nd Annual Precise Time and Time Interval (PTTI) Meeting. Reston: US Naval Observatory, 2000. 431−439.

[24] Downey AB. Using pathchar to estimate internet link characteristics. In: Proc. of the ACM SIGCOMM'99. New York: ACM Press, 1999. 241−250.

[25] Papadopoulos C, Parulkar GM. Experimental evaluation of SUNOS IPC and TCP/IP protocol implementation. IEEE/ACM Trans. on Networking, 1993,1(2):199−216.

[26] Guo CX, Zheng SR. Analysis and evaluation of the TCP/IP protocol stack of LINUX. In: Proc. of the IEEE ICCT 2000. New York: IEEE Computer Society Press, 2000. 21−25.

[27] Mogul J, Ramakrishnan K. Eliminating receive livelock in an interrupt-driven kernel. ACM Trans. on Computer System, 1997, 15(3):217−252.

附中文参考文献:

[4] 黎文伟,王俊锋,谢高岗,张大方.基于包对采样的 IP 网络时延变化测量方法.计算机研究与发展,2004,41(8):1354−1360.

[8] 卢光辉,孙世新,邵子立,张艳.基于组播的网络延迟测试.软件学报,2001,12(11):1704−1709.

[11] 王俊锋,杨建华,周虹霞,谢高岗,周明天.单向延迟测量中的时钟动态性检测算法.软件学报,2004,15(4):584−593. http://www.jos.org.cn/1000-9825/14/28.htm

[13] 林宇,王重钢,王文东,程时端.RMTP 协议的时延分析.软件学报,2002,13(8):1710−1717. http://www.jos.org.cn/1000-9825/13/1710.pdf

[15] 张宏莉,方滨兴,胡铭曾,姜誉,詹春艳,张树峰.Internet 测量与分析综述.软件学报,2003,14(1):110−116. http://www.jos.org.cn/1000-9825/14/110.htm

黎文伟(1975－),男,博士生,主要研究领域为网络测试和性能评估.

谢高岗(1974－),男,博士,副研究员,CCF 高级会员,主要研究领域为下一代互联网.

张大方(1959－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为可信系统与网络,容错计算.

杨金民(1967－),男,博士,副教授,主要研究领域为软件容错,可信网络.