一种主动测量工具开发包①

王乐璇, 黎文伟

(湖南大学 信息科学与工程学院, 长沙 410082)

摘 要: 主动测量是进行网络测量常用的方法, 大部分主动测量工具的实现过程可分为数据包发送、接收与处理三个部分. 本文从这三个方面对现有主动测量方法进行研究和总结, 归纳了主动测量的通用过程, 以 libnet 和 winpcap 库为基础, 设计一种可快速实现多种主动测量工具的开发包, 将主动测量过程中所需各种操作进行封装, 提供统一的编程接口, 使主动测量工具的实现变得简单. 实验结果表明, 主动测量工具开发包能实现多种主动测量工具功能, 并且, 相比于现有工具, 大大减少了实现代码量, 提高了实现效率.

关键词: libnet; winpcap; 主动测量; 测量工具

Active Measurement Tool Development Kit

WANG Le-Xuan, LI Wen-Wei

(Department of Information Science and Engineering, Hunan University, Changsha 410082, China)

Abstract: Active measurement is a common method for the network measurement. The implementation process of most active measurement tool can be divided into three parts: sending, receiving and processing packets. In this paper, we study the current active detection method of network performance in those three aspects and summarize the common procedure of the active measurement. A kind of development kit is designed to quickly implement various active measurement tool based on libnet and winpcap. The implementation of the active measurement tool is simplified by encapsulating all kinds of the operation required in the active measurement procedure and providing unified programming interface in the new design. As the experimental results suggest, the active measurement development kit is able to realize various function of the active measurement tool and greatly reduce the code compared with existing tool, and the efficiency is highly improved.

Key words: libnet; winpcap; active measurement; measurement tool

随着互联网的飞速发展, 网络测量[1,2]作为了解互联网性能的基本手段成为人们关注的热点. 它根据测量方式的不同可分为主动测量与被动测量. 由于主动测量的易实现性, 大部分网络测量项目都会用到主动测量技术[3-6]. 现有主动测量工具有很多, 其实现一般采用了 socket 编程. 但 socket 编程对其实现过程的低封装性使得数据包发送与接收过程效率较低; 并且, 在进行多任务操作时, 对数据封装的低实时性难以满足高速率下对发送速率的要求. 这使得主动测量方法的研究者不仅要考虑为测量指标设计准确、低开销的算法, 还需对测量工具实现中的数据包发送、接收及处理等进行研究, 增大了研究者的工作量.

此外, 网络指标测量的方法有很多, 由于实现机制的不同, 同种方法得到的结果也会有差别. 在实际测量过程中, 由于实现机制的不同, 甚至可能出现理论上较好方法比较差方法得到的结果还差的现象. 这就需要一个统一的实现机制来作为测量方法评估的基础, 减小由于实现机制不同对测量方法评估的影响.

因此, 本文基于 libnet 和 winpcap 库, 设计一种能实现多种主动测量工具的开发包, 改善实现过程低封

① 基金项目:国家自然科学基金(61173168);中央高校基本科研业务费项目;湖南省普通高校青年骨干教师培养计划
收稿时间:2014-04-13;收到修改稿时间:2014-05-26

装性的问题，使测量过程更为简单，并为指标测量方法的评估提供统一的实现机制. 本文以实验为主，在第 1 节对主动测量技术进行研究；第 2 节对主动测量工具开发包的设计过程进行介绍；第 3 节则详细介绍了主动测量工具开发包的实现过程；最后在第 4 节中对测量工具开发包的功能进行验证，并对指标测量的准确率和实现效率进行分析.

1　主动测量方法的研究

主动测量技术在网络测量的很多方面都有涉及，本文主要针对它在网络性能方面的运用进行研究. 主动测量技术在性能方面的测量主要针对连通性、延迟[7,8]、带宽[9-12]以及丢包[13-15]等指标. 连通性的测量较为简单，并且端到端路径已经确定连通是进行其它指标测量的前提，故本文对连通性不做研究，将主要针对延迟、带宽和丢包这三种性能指标进行研究.

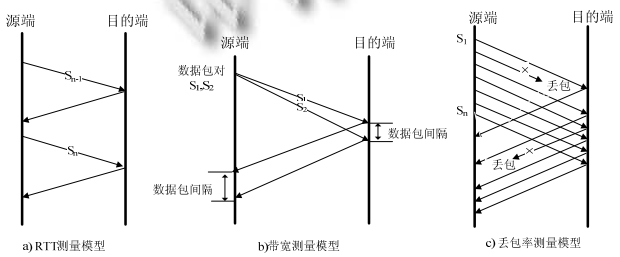

图 1　测量模型

1.1 往返延迟测量方法

往返延迟是网络状态的一个重要指标，通过对常用测量往返延迟的方法[7-10]以及往返延迟测量工具(如 ping)的实现原理进行研究，总结出利用主动测量技术对往返延迟进行测量的一般模型，如图 1 a)所示，测量者在源主机上发送单个探测包，利用源主机接收到反馈数据包与发送探测包的时间差进行估算.

1.2 带宽测量方法

带宽测量的经典方法是包对和包列两种方法. 应用最为广泛的由 Jacobson 等最早提出的包对模型，如图 1 b)所示，向网络中注入两个等长探测包 S1,S2，且探测包间距足够小，在测量点捕获数据包，利用公式(1)对带宽进行估算，式中 s2 表示第二个数据包长度，t 表示到达测量点数据包 S1,S2 的时间间隔.

$$b_{bnl}=s_2/t \tag{1}$$

包列方法是对包对方法的一种扩展，该方法由包对方法中只发送一个包对改为发送多个包对，且发送的包对间隔相等，通过分析相邻探测包的往返延迟(RTT)的变化得到路径带宽.

1.3 丢包测量方法

丢包率的测量方法较为简单，通过对常用的测量丢包率的方法[13-15]以及模型的分析，总结出利用主动测量技术进行丢包测量的一般模型如图 1 c)所示. 源主机向目的主机发送 n 个数据包，在测量点接收到数据包数量为 r，则公式(2)可对丢包率进行估算:

$$L=(n-r)/n \tag{2}$$

1.4 发包模式总结

基于以上指标主动测量技术的研究，从探测包类型、发送方式、长度、间隔、测量点以及计算所需参数六个方面对测量模式进行总结，如表 1 所示. 测量指标不同，方法不同，则对应的测量模式也会不同.

表 1　主动测量模式总结

测量指标	往返延迟	带宽	丢包率
探测包类型	tcp/icmp/udp	icmp/udp/tcp	icmp/udp/tcp
发送方式	单包	包对/包列	包列
探测包长度	---	大小一致	---
探测包间隔	---	足够小	---
测量点	发送端	发送/接收端	发送/接收端
计算所需参数	时间戳	时间戳/数据包长度	数据包个数

2　主动测量工具开发包的设计

2.1 功能分析

第 1 节中研究了主动测量方法在网络性能方面的测量模式，本节根据上文总结的测量模式对主动测量工具开发包的功能进行设计，主要从以下三点进行:

1)数据包构造与发送: 能对常用探测包进行构造，将探测包按不同的发包模式发送到网络，并且对其实现过程进行封装，提供统一的编程接口;

2)数据包接收: 能对网络中数据包进行有选择的捕获;

3)数据包处理: 对指标测量所需的计算信息进行提取. 由于指标测量的方法不同，对应的计算方法也会不同. 则现阶段我们仅对常见算法所需的计算参数进行提取，为计算做准备，而不对具体的算法进行实现.

2.2 功能模块分析

根据对主动测量工具开发包的功能分析，将其分

为具体的功能模块, 如图 2 所示, 分为三个主要部分: 数据包构造与发送、数据包接收以及数据包处理.

(1)数据包构造与发送功能主要是对探测包进行构造与发送, 从类型、长度、发送方式、发包间距四个方面实现探测包不同的发包模式. 根据探测包类型的不同, 现提供 ip、arp、tcp、udp、icmp 探测包的构造; 根据探测包发送方式的不同, 现提供单个数据包、数据包包对以及等间距数据包队列发送方式. 至于数据包的长度与间隔可根据指标测量方法设定.

(2)数据包接收功能是指捕获参数计算所需的数据包. 则此功能需要对常见的 ip、arp、tcp、udp、icmp 数据包进行捕获. 网络中存在许多与测量无关的数据包, 这就需要一个匹配机制来对数据包捕获进行匹配.

(3)数据包处理功能是对数据包进行信息的提取, 为参数计算做准备. 提取的数据包信息有数据包的长度、到达测量点的时间戳信息, 并且需要对数据包数量进行统计. 由于主动测量工具开发包包含数据包发送与接收功能, 则数据包的信息需分开统计.

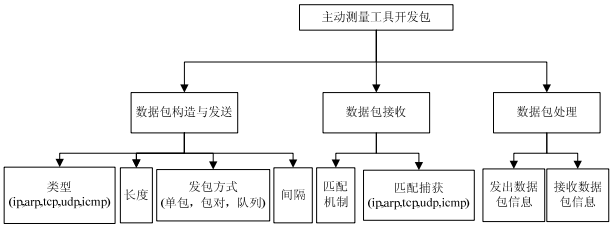

图 2 主动测量工具开发包功能图

3 主动测量工具开发包的实现

第 2 节中对测量工具开发包需要实现的功能进行了分析, 本节则在实现方面对开发包进行介绍.

3.1 开发包整体实现框架

主动测量工具开发包的整体实现框架如图 3 所示, 列出数据包发送、捕获和处理模块包含实现机制.

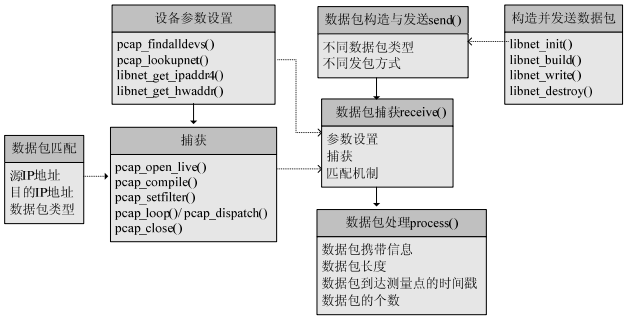

图 3 主动测量工具开发包实现框架图

1)数据包构造与发送模块 send(): 现将 libnet 库移植到 Windows, 再以它为实现基础从不同的数据包类型和发包方式对探测包进行构造, 再定义 send()函数, 将具体的实现函数进行封装, 留出必要的接口函数. void send(char*fd,int len,int num,int s,char*daddr,int dp) 其中 fd, len, num, s 分别表示数据包类型, 长度(bytes), 数量, 数据包间距(ms). addr, dp 表示目的主机 IP 和端口号. 数据包类型、长度、数量等都由此输入设定.

2)数据包接收模块 receive(): 此模块从 3 个方面实现数据包的捕获: 设备参数的输入、捕获数据包和匹配数据包. 并且定义 receive()函数, 对此三个方面实现的过程进行封装. void receive(char*daddr), 其中 daddr 表示目的主机 IP 地址.

3)数据包处理模块 process(): 定义了 process()函数来实现对捕获到的数据包的处理. 并且, 还可以写入指标计算方法, 实现对确定指标同种方法下的测量.

本节余下部分对各个功能具体实现过程以及主动测量工具开发包的程序实现流程进行详细的介绍.

3.2 数据包构造与发送模块

数据包构造与发送模块是利用 libnet 库实现数据包构造与发送功能. 对数据包的构造主要是从不同数据包类型和不同的发送方式两方面实现.

实现不同类型数据包的构造可通过 send()中输入的 fd 值对不同类型数据包进行构造. switch(fd)-case 可满足选择功能的实现, 其实现就为简单, 故我们将着重介绍不同发送方式数据包的构造过程.

3.2.1 构造单个数据包

根据 send()函数中 num 值对数据包的发送方式进行选择. 若 num=1, 构造单个数据包. 下面以 tcp 数据包为例, 对简化的构造过程进行介绍. 如图 4 所示, 简单分为了 4 个步骤.

1)初始化: 通过 libnet_init()函数完成对 libnet 的初始化, 其中包括内存分配、网络接口设置和类型设置等.

2)构造数据包: 调用 libnet 中提供的各协议块构造函数对不同类型的数据包进行构造.

```
tcp_protocol_tag = libnet_ build_tcp();
ipv4_protocol_tag = libnet_build_ipv4();
ethernet_protocol_tag = libnet_build_ethernet();
```

3)发送: 调用 libnet_write()函数发送数据包. 若发送失败, 则会显示“发送-1 字节数据包”.

4)关闭: 最后调用 libnet_destroy()函数关闭网络接口和释放程序创建的内存结构.

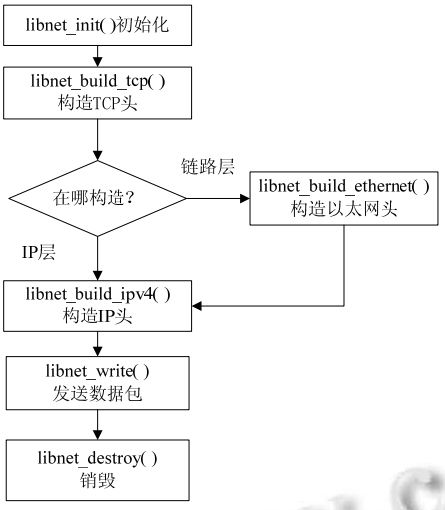

图 4　构造单个数据包流程

3.2.2 构造数据包队列

若 send()函数中参数 num>1, 则发送方式为数据包队列, 其构造过程如下:

1)根据单个数据包构造方法首先构造一个数据包.

2)利用 libnet_cq_add(l, label)向队列插入数据包.

3)重复执行步骤(1),(2)直到插入所有数据包. 完成后则由 libnet_write()实现队列的发送. 若是发送的数据包间隔 s 相等, 可在 libnet_write()后添加 Sleep(s).

3.3 数据包接收模块

数据包接收模块主要利用 winpcap 实现数据包的捕获. 对此模块主要是从设备参数设置、捕获数据包以及匹配模块三个方面实现的.

3.3.1 设备参数设置

winpcap 对数据包进行捕获时, 需要提供计算机的网络设备, 设备的网络号、掩码等设备参数. 通过 pcap_findalldevs()函数寻找本地计算机上所有的网卡设备, 并将其打印出来, 供用户选择网络设备; 通过 pcap_lookupnet()函数获取设备的网络号和掩码; 通过 libnet_get_ipaddr4()函数获取源主机 IP 地址; 通过 libnet_get_hwaddr()函数获取源主机硬件地址.

3.3.2 捕获数据包

数据包捕获是指对需要的数据包进行捕获. 实现过程如图 5 所示, 简单分为 5 个步骤.

1)打开指定的网卡设备: 利用 pcap_open_live()函数打开主机上可用的网卡设备.

2) 编译和设置过滤器过滤规则: 可由 pcap_compile()和 pcap_setfilter()实现对数据包进行过滤捕获.

3)捕获数据包: 数据包的捕获可以由不同的函数来完成, 主要有 pcap_dispatch()、pcap_loop()、pcap_next()等. 测量工具开发包实现时选用 pcap_dispatch(), 相比 pcap_loop()报错返回, 它还能超时返回; 相比 pcap_next()捕获一个数据包, 它能循环捕获数据包. pcap_dispatch()利用回调函数 packet_handler()对数据包信息进行收集.

4)关闭: 捕获数据包过程完成, 利用 pcap_close()函数关闭 winpcap 操作, 并销毁相应资源.

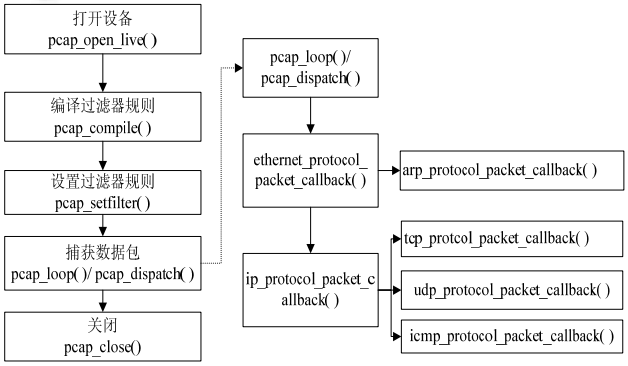

图 5　捕获数据包流程

3.3.3 匹配机制

数据包匹配机制的加入有两个目的: 一是为了减少网络中与测量无关的数据包对测量结果的影响; 二是找出目的主机返回的反馈数据包.

它的设计原理是利用以下代码将回调函数中代表数据包信息的 pkt_data 参数拆分, 将其内容存放在 pkt_data 数组中, 对其包含内容进行核对.

```
for(i=0;i<header->len;i++)
{   printf("%02x ",pkt_data[i]);   }
```

其中 header->len 表示数据包的长度. 在匹配机制设计时, 主要是对数据包的类型(数组第 23 位)、源 IP(第 26-29 位)、目的 IP 做匹配(第 30-33 位).

3.4 数据包处理模块

数据包的处理是根据测量所需计算信息对数据包信息进行提取. 我们定义了针对时间的全局数组 sec0、usec0、sec1、usec1; 数据包长度的数组 len0、len1 以及个数的 cnt0、cnt1. 定义 process()对过程进行封装.

3.4.1 发送数据包的处理

将 send()中 len、num 分别写入 len0、cnt0. 对于数据包发送时间戳信息, 则利用时间结构体 timeval 获取. 在 send()中定义 struct timeval tl, 记录每一个数据包发送的时间戳 tl.tv_sec, tl.tv_usec, 并将它们写入时间数组 sec0、usec0. 完成对主动测量工具开发包自身发出数据包信息的统计.

3.4.2 捕获数据包的处理

winpcap 的头文件已对数据包长度、时间戳的结构信息进行定义, 故将 packet_handler()中表示数据包长度以及接收时间戳的 header->len, header->ts.tv_sec, header->ts.tv_usec 写入 len1、sec1、usec1 即可. 数据包数量统计则在程序中设置全局变量 count, 每捕获一个数据包, count 值加 1, 捕获完成后将 count 写入 cnt1 中, 完成对接收数据包的信息处理.

3.5 开发包操作流程

上文对开发包各个模块的实现过程进行了详细的介绍, 本节则阐述如何利用开发包实现主动测量工具.

以 ping 工具为例, 首先分析 ping 的数据包构造发送, 接收以及处理的模式, 根据需求利用 send()构造探测包, 并发送到目的主机; 再调用 receive()等待捕获数据包; 最后调用 process()对捕获到的数据包进行处理, 代码如下, 实现过程较为简单, 仅为 3 行代码.

```
main(){ …
        send(icmp_echo,32,4,1000,dadd,dp);
        receive(dadd);
        process();
        …}
```

若需要得到延迟值, 则在 process()中加入以下代码, 完成对延迟的一次测量:

```
for(i=0;i<4;i++)
{ rtt=((sec1[i]- sec0[i])*106+(usec1[i]-usec0[i]));
  printf("测量得到的往返延迟为%d 微秒",rtt); }
```

其它主动测量工具的实现过程与上述过程类似, 用户可根据需求对各个函数中的参数进行设置.

4 实验

实验部分是主要是针对主动测量开发包在功能正确性, 测量准确率以及实现效率三个方面进行的.

实验选 windows XP 下 VC++ 6.0 作为开发环境, 湖南大学校园网作为网络环境, 计算机配置以及使用工具如表 2 所示.

表 2 实验环境

	主实验平台
CPU 主频	奔腾 2.8GHz 双核
内存容量	4GB
使用工具	VC++ 6.0
	libnet 1.1.2.1
	winpcap 4.1.2

4.1 功能性验证

功能的正确性主要从两个方面验证: 能否正确构造数据包; 能否捕获指定数据包. 下面从这两方面对实验过程进行介绍.

4.1.1 实验方案

构造功能的验证实验过程为: 利用工具 wireshark 捕获主动测量工具开发包发送的数据包, 从 wireshark 中验证数据包信息以及发送方式是否正确. 捕获功能的验证实验过程为: 利用主动测量工具开发包对其它主动测量工具发出数据包进行捕获, 以此检验其能否捕获到指定数据包. 以上两组实验多次进行.

4.1.2 结果分析

构造功能实验结果如图 6, 图 7 所示, 网络状况良好时, wireshark 捕获到的数据包类型、长度、个数与测量工具开发包构造发送的一致, 但是数据包间隔出现 110~210us 的偏差. 主动测量工具开发包捕获到的数据包个数、类型、长度都正确, 但是数据包间隔出现 160~230us 的偏差; 网络出现拥塞时, 数据包间隔会大于设定的间隔, 甚至出现丢包现象.

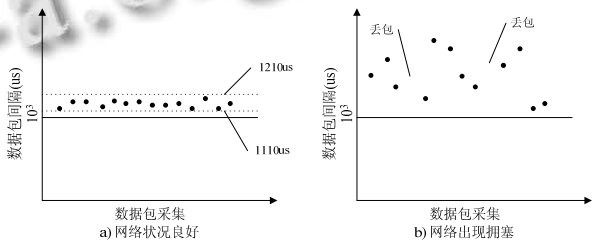

图 6 wireshark 抓包情况

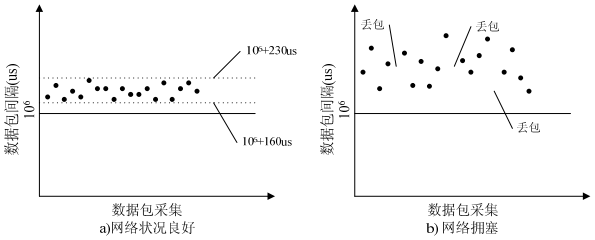

图 7 主动测量工具开发包抓包情况

实验结果表明, 网络状况良好时, 主动测量工具开发包能正确构造数据包, 并且对指定数据包进行捕获. 但数据包以队列方式发送或对数据包队列进行捕获时, 会出现数据包间隔偏差, 在网络出现拥塞时, 甚至会出现丢包现象. 此现象出现可从下面两个方面找原因: 1)程序运行消耗时间过多; 2)变化的网络状况. 第二点是无法避免的, 但可以作为网络状况评估的参考, 而第一点是应该改进的, 完善程序的编写, 提高程序运行效率, 减小由于程序而造成的测量偏差.

4.2 指标准确率分析

主动测量工具开发包的设计目的就是为了更方便的实现主动测量工具功能, 故指标测量的准确率也是需要考虑的. 指标测量准确率的验证实验设计如下: 利用开发包实现工具 Ping 和 Iperf 功能, 分别对往返延迟和带宽进行测量, 得到的测量结果与工具 Ping 和 Iperf 直接测量得到的结果相比较, 分析开发包在指标测量方面的准确率. 3.5 节中介绍了利用开发包实现 Ping 功能的过程, 且 Iperf 的实现过程与 Ping 相似.

实验结果如图 8 所示. a 图中, 由开发包测量的 RTT 值与 Ping 工具得到的 RTT 值相近, 但是也很明显的看出, 开发包得到的测量值数值较大, 在带宽方面的测量也是如此, b 图中开发包测量得到的结果与 Iperf 测量得到的相近, 但也有偏差, 但是偏差值较小, 多次测量得到指标的平均值可作为网络状况的参考. 偏差出现的原因是 4.1 节中介绍的两点, 故完善程序的编写, 减少程序消耗的时间是我们下一步任务.

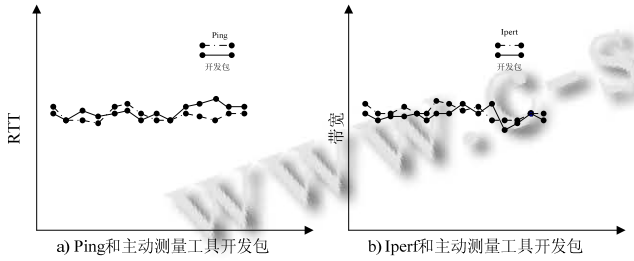

图 8　测量指标值比较

4.3 实现效率分析

在本节中, 我们将对主动测量工具开发包的实现效率进行分析, 方法是利用开发包实现其他主动测量工具功能, 利用实现代码量的比较来分析开发包的实现效率. 实验选用 Ping, Iperf, Pathload 作为对比工具. 我们对 ping 等工具进行代码量统计(除了空白行和注释行), 结果如表 3 所示. 表中数据表明利用主动测量工具开发包实现其他工具功能时, 由于封装性较好, 代码量远远小于其他工具实现代码量, 并且过程简单方便, 提高了测量工作的效率.

表 3 工具代码量

工具	主动测量工具开发包	ping	pathload	iperf
代码量	3+算法代码量	395	2687	869

5 结语

本文对主动测量技术进行研究, 设计一种能实现主动测量工具的开发包, 使测量过程变得简单, 也为网络指标测量方法的评估提供统一的实现机制. 当然, 主动测量工具开发包还有需要完善的方面, 下一步任务是在功能上实现非等间隔数据包队列的发送; 完善程序的编写, 提高程序运行效率.

参考文献

1 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.

2 谈杰,李星.网络测量综述.计算机应用研究,2006,23(2):5–8.

3 Shalunov S, Teitelbaum B, Karp A, Boote J, Zekauskas M. A One-way active measurement protocol. RFC 4656. Sep 2006.

4 Hedaya K, Krzanowski R, Morton A, Yum K, Babiarz J. A Two-Way Active Measurement Protocol. RFC 5357. Oct 2008.

5 Luckie MJ. Scamper: A scalable and extensible packet prober for active measurement of the internet. IMC'10. Melbourne, Australia, November 2010. 239–245.

6 Zhang Y, Oliverira R, Zhang HL, Zhang LX. Quantifying the pitfalls of traceroute in AS connectivity inference. PAM'10. Zurich, Switzerland, April 2010.

7 Choi JH, Yoo C. One-way delay estimation and its application. Computer Communications, 2005, 28(7): 819– 828.

8 Chen SQ, Qin YF, Wang JF, Zhou X. HPAP: High precision active probe for path roundtrip delay measurement. Proc. of 2012 IEEE 14th International Conference on Communication Technology. 2012. 82–87.

9 Jain M, Dovrolis C. Pathload: A measurement tool for end-to-end available bandwidth. Passive and Active Measurements, Fort Collins, CO. March 2002. 14–25.

10 Jain M, Dovrolis C. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with

TCP throughput. SIGCOMM Computer Communication Review, 2002, 32(4): 295–308.

11 Cheng Y. New exploration of packet-pair probing of available bandwidth estimation and traffic characterization. IEEE ICC. 2007. 588–594.

12 韦安明,王洪波,林宇,程时端.IP 网带宽测量技术研究与进展.电子学报,2006,34(7):1301–1310.

13 Wang YA, Huang C, Li J, Ross K. Queen estimating packet loss rate between arbitrary internet hosts. PAM 2009, LNCS 5448, 2009. 57–66.

14 Filho FS, De Souzae Silva E. Modeling the short term dynamics of packet losses. ACM Performance Evaluation Review, 2006, 34(3): 27–29.

15 Nguyen HX, Thiran P. Network loss inference with second order statistics of end-to-end flows. Proc. of IMC. 2007. 227–240.