服务器前端基于被动的可用带宽测量研究与实现

专　　业：无线电物理
作　　者：柯　礼
指导老师：余顺争教授

摘要

网络行为分析是网络安全的重要课题之一，而网络测量是网络行为分析的基础。网络测量是通过对网络状态的观察和使用特定的参数对其进行评估，从而判断网络的变化趋势并加以利用。IETF 的 TEWG 工作组对网络测量给出了一个指导性的理论框架。网络工程师们在实践中提出了许多巧妙的网络测量方法，其中被动测量是唯一对网络当前状态不产生影响的测量方法。它能在网络现有流量提供的信息中获得我们所需要的各种性能参数，并能利用业务流量本身的变化来进一步区分出正常流和异常流，达到安全防御的效果。

本文首先回顾了国际国内各种测量技术的发展状况。然后对可用带宽的被动测量技术进行深入的讨论。通过修正样本采集时丢失报文带来的误差，改进了利用 TCP 拥塞窗口测量可用带宽的方法。提出了相应的可用带宽的粗测量和精确测量两级测量方法。其中的精确测量方法，解决了已有算法使用 RTT 带来的误差问题。还提出了使用独立设备对网络进行监听并重构拥塞窗口的方法，避免了传统的、在服务器中直接采集 TCP 拥塞窗口数据所造成的对服务器性能的影响。

最后通过在已有工具 tcptrace 上扩展核心功能实现了本文提出的算法。并使用该算法在实验网络中进行了实际的测量实验。

关键词

可用带宽测量　被动测量　拥塞窗口　RTT　端到端测量

Passive Measurement of Available Bandwidth in Front of a Web Server

Major : Radio Physics

Author: Ke Li

Supervisor: Yu Shun Zheng

Abstract

Network behavior analysis, which based on the network measurement, is one of the most important tasks of the network security. Network measurement was defined as monitoring the state of the networks and evaluating them through some special parameters based on one's special motivations. After that, we can try to find the trend of the network changes and utilize it. The TEWG of IETF presents a comprehensive guiding framework of network measurement, while many network engineers invent their own way to measure Internet. As the only one measurement method without disturbing the network, passive measurement can obtain many performance parameters we need, just using the flows which have been in the network. And it can provide some security by distinguishing the normal flows from the abnormal flows through the state changes of the parameters.

A complete survey of measurement technologies is given in this paper. Then a passive method of available bandwidth measurement is discussed in detail. As an improvement of TCP congestion window-based method of available bandwidth measurement, our algorithm has improved the error rate caused by loss packets. Then, a measurement method that is implemented in two levels of sketchy measurement and precision measurement is put forward. We also solve the problem brought by using RTT in quondam algorithm in precision measurement. Besides, a technique of monitoring networks and reconstruct congestion windows by independent equipment is discussed, which is different from the traditional way that gets the TCP congestion window size directly from the Server.

Finally, some core functions are implemented based on tcptrace. And some tests in the experiment network are given.

Key words

Available Bandwidth Measurement Passive Measurement RTT

Congestion Window End-to-End Measurement

前言

我很荣幸能够在进入硕士研究生学习以后不久就找到了自己的研究方向。在我入学后，我的导师余顺争教授针对 2008 年奥运会网站的安全问题组织开展了一系列的研究工作。在余老师的指导下，我选择了服务器前端可用带宽的被动测量作为我的研究课题。

在两年的研究生学习过程中，我始终跟踪国际上相关研究团体的研究进展。在网络流量的研究领域中，我认为网络测量是分辨网络状态和认识网络行为的有效手段，具有很重要的作用。作为目前唯一一种不对网络产生任何附加影响的测量方法，被动式测量具有无干扰、隐蔽性强等不可替代的优势。在为安全检测服务的测量中，测量系统如何避免攻击也是必须研究的课题之一。

经过对国际上相关领域发展情况的研究和总结，我选定了拥塞窗口测量可用带宽的方法作为主要研究对象。它利用 TCP 本身的拥塞控制算法来实现对可用带宽的测量，虽然提出较早（1997 年），但延至今日，仍然存在一些不足。针对这些不足，我提出了一些自己的解决办法。并在 tcptrace 的基础上进行核心功能扩展，实现了修正后的算法，并在实验网络上进行了测试。

本论文主要分为下述几个部分：第一章综述了网络测量的发展以及现状，对网络测量的概念、测量参数和工具、基础设施以及测量组织进行介绍；第二章详细描述了网络监听技术与被动式的网络测量的技术原理；第三章详细介绍了 TCP 拥塞控制算法以及其不同的实现版本，它是拥塞窗口测量可用带宽方法的基础；第四章对现有的基于被动的可用带宽测量算法进行介绍，主要描述了拥塞窗口方法和吞吐量方法，并指出它们的不足；第五章是针对拥塞窗口方法的不足，提出了详细的改进方案，并给出了服务器前端基于被动的可用带宽测量算法；第六章主要介绍服务器前端基于被动的可用带宽测量算法的实现过程，它是对 tcptrace 进行功能扩展的过程；第七章给出了实验方案，并对实验结果进行分析，得到我们的结论。

由于时间紧张，以及本人能力有限，文中缺点错误在所难免。恳请各位老师和同学提出宝贵意见。

第一章 网络测量概述

1.1 网络测量概念及其历史

网络测量是指遵照一定的方法和技术,利用软件和硬件工具来测试或验证表征网络性能和状态的指标的一系列活动的总和[1]。它监视网络运行状态，对网络的性能和状态进行分析，既是网络优化的基础，也成为检测某些特定攻击的有力工具。

网络测量是随着计算机网络的发展而逐渐被人们认识的。1969 年劳动节，第一台接口报文处理器（Interface Message Processor, IMP）被送到了美国加州大学洛山矶分校的网络测量中心（Network Measurement Center），这标志这现代计算机网络发展的开始。网络测量中心当时是美国高级研究项目局（Advanced Research Projects Agency, ARPA）的一个下属机构，随后它成了 ARPANET 的第一个节点。当时连在网络上的设备不多，1970 年的网络节点数也只有 10 个，但是以网络测量为中心的网络就已经受到了足够的重视。1972 年，Gerald D. Cole 等人对 APRANET 的性能进行了测量[2]，这标志着网络测量研究的开始。当时网络的规模还很小，网络测量的内容主要集中在性能方面，而对网络结构的探测之类的课题并没有提出来。

20 世纪 70 年代，新的机构和团体不断加入计算机网络的阵营[3]。美国能源部建立了 MFENet(Magnetic Fusion Energy Net)以及 HEPNet(High Energy Physics Net)；美国航空航天局建立了 SPAN 网络；美国国家科学基金会(National Science Foundation）资助建立了 CSNET（Computer Science Net)。另外还有基于 AT&T 的 UNIX 系统的 USENET 和用于连接大型计算机的 BITNET。

1986 年开始，现代因特网（Internet）的主干部分在 NSF 的资助下完成，网络进一步复杂化。虽然如此，但网络仍然只是在少数几个机构控制中，这些机构不仅仅关心各自的网络运行情况，而且对机构间的网络的全局性能也相当重视。80 年代后期，人们开始对实时的网络测量技术进行研究[4]，网络测量的研究逐渐走向成熟。

1995 年互联网私有化以后，呈现出两大特点：无政府状态的联合和异构性。无政府状态的联合默许了不同的网络服务提供商（Internet Service Provider ISP）在竞争中只关注自己的网络构建，使得 ISP 间的全局性能问题、网络的发展预测与评估一度处于停顿状态。TCP/IP 把不同性质的网络连接在一起，但其在本质上并没有改变网络的异构性。这两个特点以及其庞大的规模所带来的复杂性，随着客户的快速膨胀、对性能和服务质量的要求日益提高的期望，更雪上加霜的加剧了单纯的增加容量、内容和服务的扩张模式与网络效率之间的矛盾。于是，人们开始从盲目的竞争中回到理性的合作中来。网络测量作为网络性能优化和安全状态分析的利器才又一次得到人们的重视。90 年代里涌现的多个测量项目组织，如 NLANR（The National Laboratory for Applied Network Research）的 MOAT（Measurement and Operations Analysis Team），CIADA（the Cooperative Association for Internet Data Analysis）、欧洲的 PPNGG（Particle Physics Network Coordinating Group）等，它们给出了一系列的测量基本框架模型。网络私有化后的测量合作过程，也是网络测量逐渐形成体系的过程。

我国网络的发展起步较晚，直到 90 年代初才引入 Internet。因此我国在网络测量方面的研究也较晚。中国科学院计算机技术研究所、国防科技大学、西南交通大学、东南大学、哈尔滨工业大学、复旦大学以及中山大学等研究机构和大学都在网络测量方面开展了研究工作。但由于起步晚，跟国际相比仍然存在着相当大的差距。

爆炸式的网络发展速度及其越来越复杂的结构，使得目前的各种测量技术的研究和发展远不能跟上网络本身的发展。对网络测量的研究，必将随着网络的变化而不断地提出新的要求。

1.2 网络测量和分析的研究内容和应用

网络测量所牵涉到的内容很多，不仅仅在测量基础上的建模，甚至测量后的控制，都进入了网络测量的研究范畴之中。其在研究领域上主要包括三个方面内容[5][6]：

1. 网络测量系统和方法的研究。网络测量系统和方法是对网络进行测量的技术手段，包括获取各种参数，对获取的数据进行保存和管理等方面的内容。

可以使用主动方式实现，也可以使用被动方式。它监视网络当前的运行状态，从而找出可能导致发生错误的设备或操作；监视网络服务质量及其连续性；检测流量工程策略的有效性并提供反馈信息以及在不同的ISP之间提供边界流量信息等。为进一步的建模分析和控制提供原始数据。

2. 建模分析。实现对网络未来状态的预测及对流量的变化发展进行评估是网络测量的目标之一。建模分析是要在参数测量的基础上，建立正式的网络传输与流量模型和模拟方案，对网络的状态变化进行预测和评估的过程。通过模型化，识别流量的模式及其变化，分析流量的分布及其发展，预测不同的路由器和网络上不同服务类型的未来表现。并在预测未来流量的变化趋势的基础上，对未来的协议设计和网络优化等工作提供指导。

3. 流量控制。根据参数测量和建模分析的结果可以对网络的流量进行控制，实现网络资源的合理配置和利用。可以针对特定的网络事件对网络的性能进行优化，为 MPLS 等协议提供对流量信息进行反馈的机制，支持基于测量的接入控制等。

测量网络的拓扑结构，测量各个参数对大规模网络的流量进行动态描述，并根据网络的变化分析网络的性能，对网络效率和行为进行评价，在网络的性能优化和安全保障方面有着重要的意义。从应用上来讲，至少包含以下几个方面：

1)、网络监视。包括对网络运行情况的监视、网络资源的监视和网络性能（如业务吞吐量、时延、丢包率、RTT、带宽等）的监视等，并在网络监视过程中报告故障以及异常情况，做出相应的评价；

2)、网络质量控制和辅助性网络管理。如发现并改正病态路由、根据长期观察的路由数据对网络选路指定测量、网络被破坏后的网络资源自组织等。

3)、防范大规模网络攻击。通过一定的流量模型对测量信息进行分析可以发现攻击，为防范大规模网络攻击提供预警手段，使得对网络的管理更具有宏观控制能力。

此外，网络测量对于对不同的 ISP 的服务质量（quality of service， QoS）的比较、移动 IP 的位置发现、代理服务器的选择[7]均能提供重要的参考，并为协议设计和评价分析提供研究基础和检验方法，以及为 Internet 流量工程和网络行为学的研究提供辅助依据及验证平台。网络测量作为了解实际网络的有效手

段，已经成为对网络的设计、性能优化、协议分析和设计以及安全检测等方面的依据和最有效的验证方法之一。

1.3 网络测量技术的分类及其特点

网络的复杂性、异构性以及服务类型的多样性对网络测量最直接的影响是带来了测量方法的多样性，人们对网络测量的多方面思考，也使得网络测量具有多种分类方法。

根据测量信息获取的方式，可以分为主动测量和被动测量。主动测量是根据测量需要，向网络中发送特定的测量分组以引起网络设备或者终端主机的特殊响应，实现网络状态信息的获取，如用 ping 的方式测量连通性和 RTT 等。被动测量的信息提取是建立在对现有流量的监控基础上的，如 tcpdump 可以根据需要把各种格式任意长度的报文采集下来，对吞吐量、丢包率等进行统计。主动测量具有更好的针对性和可控性，而被动测量则具有不产生附加流量的优势。如何把两种不同的测量方式整合，使其达到互补的效果，是现代网络测量的发展趋势之一。

根据测量点的数目多少，可分为单点测量和多点测量。单点测量的测量点只有一个，可以实现测量点的状态信息或者经过该监测点的某些链路信息。多点测量则在网络中分布多个测量点，除具备单点测量的特性外，通过对多个监测点的信息进行综合分析，还可以实现对整个网络或其某个局部的某些全局性参数的测量。但是当多点测量的测量点数目很大的时候，将出现对信息的综合及存贮问题、对测量设备及其软件的协同更新等问题。在能够满足需要的情况下，人们总倾向于单点测量。

根据测量者与被测量者之间的关系可以分为协作式测量与非协作式测量。协作式测量是一种在被测网络或设备知情的情况下，给予配合实现的测量方式，协作式测量往往是主动式的测量（但主动测量却不一定式协作测量），如 ping 程序是在被测量的设备协作返回 ICMP 报文的情况下实现的。而非协作式测量则是不需要网络或设备配合，可以在网络及其设备完全不知情的情况下实现测量，被动测量是非协作测量的典型实现，它对网络的现有流量进行监测，这对被测量的网络和设备却是透明的。现在的 Internet 是个规模庞大，结构复杂多样的网络，大范围的协作式测量往往难以实现，因此，协作式测量常常只能在小范围的网络内

实现，而当需要对牵涉网络设备较多的大规模网络进行测量的时候，常常采用非协作式的测量。

目前有多种可用于网络测量的协议，如 SNMP、ICMP、ROMN、CMIP 等，因此可以根据测量所使用的协议方式可以将测量分成多种基于某种协议的测量。当前网络中的流量绝大多数都是 TCP 报文[8]，因此，最多的仍然是基于 TCP/IP 协议的测量，针对 IP 报文测量的 IPMP 协议也正在设计讨论中。如果在将来不出现新的协议对现有的流量结构进行改变，网络测量的这一倾向也将继续保持。

依据测量对象的不同可以分成对流，对接口、链接和节点，对节点对以及对路径的测量。文献[5]对这几种测量作了详细的描述。依据测量对象不同，也有人将网络测量分为点测量与端到端测量。

各种方式的测量均有其优缺点，对网络的测量，采用什么样的实现方式、单点测量还是多点测量、协作与否、采用什么协议以及测量什么对象和内容，总是视测量的需要而定。网络是多样的，测量的目的也是多样的，不存在绝对优势的测量方式。

1.4 网络测量的参数

网络测量根据不同的目的，总是对某些特定的参数进行测量。到目前为止，在网络测量方面比较关注的参数如下：

1)、流量大小。流量大小的测量不仅包括流量的绝对大小，还包括给定时间内流量的均值和变化等。测量单位可以是比特、字节和分组。测量的基准可以是 IP 子网、网络接口、链接、节点、节点对、路径、网络边界以及自治系统的边界。在网络最忙碌的时候进行测量，可以获得网络的容量。流量大小的变化，可以用于区分网络是否异常，并可判断异常的程度等。

2)、平均持续时间。平均持续时间指的是流的平均持续时间或者 MPLS 路径的持续时间。它和传统电信网络的呼叫持续时间参数相类似。对 MPLS 路径而言，静态路径的持续时间反映了网络设备的可靠性和失效规律，从而可以得到一些服务相关的信息。

3)、连接或路径的容量（带宽）和可用带宽。容量指的是在某条路径或某个连接上没有其他网络流量时，该连接能够提供的最大吞吐量。可用带宽指的是在

有交叉流的情况下，该路径或该连接能够提供的最大吞吐量。可用带宽与网络的目前使用状况密切相关，可为负载均衡以及基于测量的接入控制提供参考。

4)、吞吐量。这里的吞吐量指的是有效的吞吐量，即除去丢失的、错发的以及出错数据之外的在两个端节点之间传输的数据数量。对该参数进行测量时，应提供网络的环境参数，如当时网络的负载情况、测量的协议层次等。可以从比特速率和分组速率来获得平均分组大小，由于路由器对报文的处理是以报文为单位的，因此这一数据对测量路由器的性能很有意义。

5)、延迟。主要包括在路由器内部队列中等待的延迟和节点间端到端的单向和双向延迟以及延迟的变化（即延迟抖动）。延迟可以反映网络的拥塞情况，延迟的变化体现网络状态的变化，因此也可以通过延迟对网络的状态进行判断。

6)、分组丢失率。传输或协议的错误和拥塞都有可能导致报文丢失。比较高的报文丢失率将意味某个地方发生错误或者拥塞。它是衡量网络性能的一个重要参数。

7）资源利用率。资源利用率包括连接、路由器的利用率以及缓冲区的利用率等。根据路由器的体系结构的不同，资源利用率的测量还可以包括每个业务线卡或者中央控制单元的处理器、内存的利用率等。

此外还有一些与连接或路径建立有关的实体也可能成为测量的参数，如路径建立的延迟、路径释放延迟以及路径恢复时间等。

1.5 典型的测量工具介绍

作为网络测量研究的成果，同时作为进一步的分析基础，测量工具在网络测量中都扮演着相当重要的角色。网络测量的研究自 70 年代开始至今，人们根据各种目的，各种方法设计出数以百计的测量工具来，在文献[9]中，提供了超过二百种测量或者测量相关的软硬件信息。本节将采用文献[5]中的分类方法，对其中典型的测量工具进行介绍。

1．点对点测量工具

点对点测量工具以 ping 为代表，并包括其衍生版本 Nikhef ping、Fping[10]、Pingplotter[11]、Imeter 等。ping 是最早的网络测试与诊断工具，现在已经成为了 TCP/IP 协议族的一个标准工具，它以主动方式向某主机或路由设备发出探测包

来确认该主机或路由设备是否处于活动状态，并可以使用参数指定发送探测包个数、探测时间、分析输出形式等，视不同平台的实现而略有不同。一般来讲，ping能够报告测量出来的最大、最小和平均的往返时间（Round Trip Time，RTT），RTT 和丢包率是非常重要的网络状态参数，因此通过 ping 工具可以判断本机到目的主机的网络状态。

Fping 增加了用参数指定对多个主机进行 ping 操作的功能，而 Pingplotter 则和下面提到的 Mtr 一样，是整合了 traceroute、ping 和 whois 三个功能的程序。

2．路由信息测量工具

这类工具以 traceroute 及其衍生程序 Nikhef traceroute、Mtr、GeoBoy、GTrace、Xtraceroute、Skitter 等为代表，主要用于测量端到端的路由信息，包括沿途经过的路由器顺序、路由器地址和往返时延。traceroute，通过设定 TTL 值，当 TTL 为零时，路由器返回相应的 ICMP 报文，从而获得该路由器的相关信息。它也是基于主动方式的测量工具，产生的附加流量比 ping 要大。

Skitter 与 traceroute 相比增加了图形界面，Geoboy 和 GTrace 添加了地理信息，可以产生部分路径与地理位置的映射，并在图中显示。对路由信息进行测量的其他工具还有 EdgeScape、RouteExplorer 和 VisualRoute 等，其功能都类似于 traceroute，但都提供了图形界面。

3．吞吐量与带宽、可用带宽测量工具

这类工具主要用于测量网络带宽，以 Treno[12]、bing[13]、Bprobe 等为代表。主要是利用探测分组、分组对、分组链（packet train）等方式对网络进行探测和分析，以测量网络带宽和吞吐量。

Treno（Traceroute RENO）是匹兹堡超级计算中心开发的用于测量运行 TCP 协议的网络吞吐量的工具，它是 traceroute 和 tcp 拥塞控制算法（Reno 算法）的结合，前者实现探测分组发送，后者根据 TCP 拥塞窗口的回退算法（back off）实现测量。

bing 和 Bprobe 是基于 ping 的点到点的带宽测量工具，前者通过测量不同大小分组本身与其 ICMP echo 响应报文的来回时间来测量真实的吞吐量[13]，后者测量的是瓶颈带宽和拥塞带宽。

Pathload 通过测量单向时延来获得某条链路的可用带宽。其基本思想是发送

端慢慢增加探测流的发送速率，接收方对接收到的探测流的时延进行观察，当探测流速率小于可用带宽时，其时延特征与大于可用带宽的情况有明显区别。Pathload 就是通过观察时延特征的转折点来获得可用带宽的值[14]。

对带宽测量的工具还有很多，如 IPerf、I2perf、Tcpspray、ttcp、netperf、ABwE、Pathchar、Nettimer、Btest 等。它们都使用分组、分组对或分组链的时延特性完成对吞吐量、带宽、可用带宽等参数的测量。

4．流量监视工具

这类工具主要是以被动方式实现的，包括以 TcpDump[15]为代表的流量采集工具和以 Tcptrace[16]为代表的流量分析工具。

TcpDump（其 windows 版本为 WinDump）是著名的流量截取工具，它可以截取任何网络接口卡所发送或者接收的所有数据链路层的数据帧，通过将网卡置为混杂模式，还可以对所有经过该网卡的数据进行截取。它有着灵活的截取规则，可以根据主机、子网、协议、端口、传输方向等条件对截取流量进行过滤。还可以根据需要，指定采集数据帧的长度。

Tcptrace 是由 Shawn Ostermann 写的一个流量分析工具。输入的流量可以是 TcpDump、snoop、etherpeek 等多种流量截取格式。其输出信息相当丰富，包括连接持续时间，发送和接收的字节和报文数量、重发报文数、RTT、通告窗口信息、吞吐量等。它在 xplot 的帮助下还可以把部分分析结果（如吞吐量、RTT 等）以图形的形式表示出来。

类似的工具还有 ethereal、ethedetect、Netflow、SNMP MIB、Mmdump 等，它们也都以被动的方式对网络中现有流量进行监视、提取以及分析。

5．统计分析工具

这类工具与流量监视工具不同，它们往往能给出网络状态数据，并进行一定的统计分析。以 MRTG、NeTraMet 等为代表。

MRTG 是由 Tobias Oetiker 和 Dave Rand 设计的一个基于 Perl 的图形化界面流量统计分析工具。它通过调用外部的实用程序完成 SNMP 查询、生成 GIF 图形并创建 HTML 页面。MRTG 主要由四个模块组成：SNMP 模块用于网络状态变量数据的收集；日志文件记录收到的流量数据；RateUp 模块用于快速生成统计图形；配置模块用于辅助用户生成 MRTG 的配置文件。

NeTraMet 可以根据流的地址信息来实现其包的报文或者字节数量统计。程序分成“manager”和“collector”两个部分，前者负责定义下载的规则，并根据这些规则对流进行分类，后者则通过 SNMP 协议来收集流量数据。

除以上软件外，还有 WAND（Waikato Applied Network Dynamics）开发的 DAG 硬件采集卡，以及 NLANR 在此基础上配合 CAIDA 的 CoralReef 软件套件搭建的 OCxMON 监控器等硬件设备等。

视测量的目的不同，采用不同的工具，也可以采用多种工具的组合使用实现多方面的测量，下一节中描述的网络测量基础设施的主要思想之一就是为多种测量工具提供整合的平台。

1.6 网络测量系统及基础设施

上一节中所述的工具能够完成某些特定的测量任务，但它们还不能完成系统性的测量任务。为适应网络的变化以及网络测量发展的需求，人们构建出多个测量平台和基础设施，以适应测量功能的扩充和发展。

国际上许多科研组织和大学都成立了与网络测量有关的组织，建立了多个网络测量系统或基础设施，借助广泛分布的测量点，在全球范围内对 Internet 的性能状况进行监控和分析[17]。文献[17]对部分具有代表性的网络测量系统或基础设施作了详细的介绍，本节在其基础上进行补充。

NIMI（National Internet Measurement Infrastructure）[18]早期由美国科学基金会（The National Science Foundation, NSF）资助，现在由美国国防部高级研究计划局（Defense Advanced Research Projects Agency，DARPA）资助，用于测量全球的 Internet。美国 Berkeley 大学 Vern Paxson 编写的 NPD（Network Probe Daemon）软件是第一个执行大规模端到端 Internet 行为测量的软件。NIMI 是由 NPD 的工作继承和扩展而来，它是构造网络测量设施的软件系统。NIMI 强调构造一个网络测量的架构，而不仅仅为了某种特殊目的执行一系列特定的测量。它并不针对特定的测量工具，而是将测量工具视为独立的第三方软件模块。

NLANR 测量和网络分析工作组（MOAT）[19]的目标是监测高性能连接的网络行为。它们开发并维护了一个网络分析设施（Network Analysis Infrastructure，NAI），用来更好的理解系统服务模型和网络的测量参数。基于 NAI，MOAT 包

含了被动测量和分析项目（Passive Measurement and Analysis project，PMA）和主动测量项目（Active Measurement Project，AMP）两个项目。其中前者测量Internet并保存IP分组报文首部的轨迹文件、SNMP RMON数据和基于BGP数据的路由信息，而后者则通过主动测量保存大量测量点（2004年7月时达到150个[20]）之间的RTT、丢包率、拓扑和吞吐量来监测网络的性能状况。

Surveyor（Advanced Network & Services/Common Solutions Group R&P Network Measurements）[21]是一个世界范围的网络测量基础设施。它依据IETF IPPM工作组制定的标准来测量Internet路径的性能，如单向时延、损耗、路由测量等，同时对测量方法和测量工具加以改进。

依托UCSD/SDSC（University of California，San Diego Supercomputer Center）的研究部门CAIDA（Cooperative Association for Internet Data Analysis）[22]，开展网络测量、分析、可视化工具的研发，维护全球Internet平台的健壮性和可扩充性。研究对象包括Internet拓扑结构、网络负载、网络性能、网络路由、监测异常活动，关注带宽估计，以进行流量工程设计、安全迹象监测等。

此外，国外还有其他的一些网络测量基础设施，如IEPM（Internet End-to-End Monitoring）、Nws（Network Weather Service）、PPNGG（Particle Physics Network Coordinating Group）、RIPE-NCC（Reseaux IP Europeans Network Coordination Center）等。

国内的哈尔滨工业大学计算机科学与工程系实现了一个大规模网络拓扑测量的原型平台，能够针对大规模网络进行路由IP拓扑结构的自动发现，并进行可视化显示。该课题对全国范围内的近17万个IP进行了探测，获得了1612个路由IP节点间的2940个连接关系，并结合各IP的地理信息，生成了分层次的地理拓扑图，与实际相比，准确度达到90%以上。国防科技大学、西南交通大学等单位在基于ICMP协议的IP拓扑探测方面的技术比较成熟，但未曾有面向Internet的大规模网络测量和分析[6]。

复旦大学计算机科学与工程系于1999年成立了因特网流量工程研究组（FUITEG）[23]，主要研究包括：网络性能测量方法，因特网流量分析与建模，网络行为模拟仿真等。并试图建立复旦大学网络分析平台（FUNAP）。其基本框图如图1-1所示。目前该系统仍然在开发中。

中山大学电子与通信工程系网络研究室从安全和负载均衡的角度也开始了对网络测量方面的研究工作。通过对网络可用带宽的测量及建模分析，试图判别网络是否处于正常状态，并尝试对异常状态进行分级描述。并根据可用带宽以及用户识别等方式实现负载均衡等。

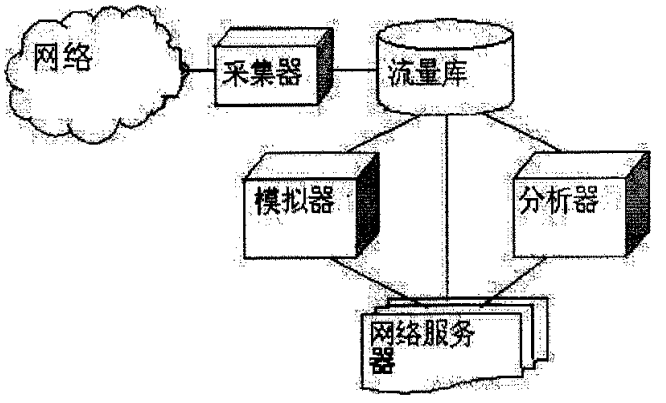

图 1-1 FUNAP 基本框架

1.7 小结

现有的网络测量技术分类繁多，根据其实现可以将其归为主动和被动两类。

在主动测量方面，由于其技术本身的局限，必须向网络中注入只用于测量的附加流量。因此，在测量过程中必将对网络造成影响，既影响网络的性能，也影响测量结果的准确性。目前网络测量的困难在于大规模的全局性测量难于实现，使用主动方式进行全局性的大规模测量，其附加流量的影响将不可忽略。

被动测量的主要问题则主要在于对数据的存贮和处理的能力上。被动测量不需要向网络中注入流量，但其分析更依赖于实际网络所能够产生的有效样本。现在的网络速度的高速增长，被动测量在对数据的采集、存贮、以及处理方面都难以应付。大规模的被动测量实施，将意味着大规模的存贮设备和处理设备的投入，这显然也不太可能。主动和被动测量两种方式的测量技术各有特点，测量应视具体的目的，选择适当的测量方法。

综观网络测量的发展以及网络本身的发展过程，网络测量的发展远远落后于网络本身的发展。网络的规模快速膨胀、异构性与复杂性的增加、网络服务类型

的多样性发展、网络的私有化带来的ISP之间的合作问题以及速率的不断增长，成为网络测量进一步发展的主要障碍。目前已有多个测量组织、测量项目正在逐步的推进网络测量技术。但它仍然摆脱不了这些问题的困扰。人们已经意识到全局性测量的重要性以及测量成本之间的矛盾，开始考虑使用一种被称为IPMP的标准协议，使各个运行该协议的联网主机本身具备测量条件。目前该协议仍在研究中。

第二章 网络监听与被动测量技术

面对快速膨胀的网络，寻求快速高效的测量技术对网络的各种参数进行测定成为了解网络各种状态及其变化趋势的主要途径之一。被动测量是测量技术中重要的方法之一，与主动方法相比，它具有不对网络造成附加影响的优势。网络监听与被动测量是极其密切的两个概念，网络监听是对网络流量进行观察和采集的过程，而被动测量则是对现有流量的统计分析，它总是建立在网络监听的基础上，它是用于测量目的的网络监听。

2.1 被动测量原理与分类

被动测量是在对网络现存流量的采集和分析的基础上完成的测量过程。其基本原理是在链路上的通信实体之间放置监测器（Monitor），通过探针、路由器的监测端口或以太网的共享特性等方法监听两实体间的所有或部分流量，如图2-1。只要能在形成实际通信链路的两个实体之间进行监测，就能够以被动方式对其进行测量。

对流量做什么样的分析是根据目的以及其特定位置而定的（如对接入链路及主干链路的测量其分析就可能有所不同），其分析和处理可以分为实时与非实时的处理。前者是采集与处理同时进行，边采集边处理；后者把文件先保存下来等采集工作完成后再进行处理。对于完全采集所有协议层次数据的情况，由于数据量极大，通常都只能实时处理。

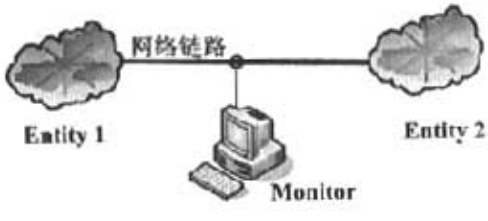

图2-1：被动测量原理

被动测量与主动测量相比具有不可替代的技术优势：

(1)、不产生和不改变网络的任何流量。如果采用独立的设备进行测量可以

达到不对网络、网络设备终端主机等造成任何影响的效果；

(2)、不需要占用合法 IP，对外部透明。利用探针技术、监测端口或者以太网的共享特性实现的被动测量可以在不占用合法 IP 的情况下实现测量。针对网络安全而进行的测量，测量系统本身也可能成为被攻击对象，不占用合法 IP 的被动测量具有更好的隐蔽性，可以减少受到外界攻击的可能；

(3)、被动测量获得的数据比主动测量携带更丰富的信息。它可以提供网络测量点上的一系列详细信息[24]，如流量、协议、精确的位速率和包速率等。

当然，也存在一定的缺点：首先，与主动测量有针对性地向网络发包并采集数据不同，被动测量对于网络中将要出现的分组无法预知，测量的样本数量难以控制，因此其可控性较差，实现起来比较困难；也因此，被动测量中要采集足够的数据才能进行较为准确的测量，这给测量带来了巨大的数据存贮和处理方面的压力，虽然用于存贮和处理的硬件有了很大发展和抽样测量技术的出现，但以更快速率膨胀的网络流量使得这种压力有增无减；此外，被动测量还必须对许多无用或是无关紧要的信息进行处理，测量设备的利用效率比主动测量低。

被动测量根据其测量的范围可分为点测量和路径测量，前者只对测量点本身的数据流量进行分析（如某个路由器的netflow采集分析），而后者则对测量点所在的链路特性进行测量；根据测量点的数目可分为单点测量和多点测量，被动测量的单点测量只能实现测量点的点测量或路径测量，如果要实现某个网络的测量必须依靠多点测量的协作来实现。从实现方案上可以分为软件实现和硬件实现两种。

2.2 软件实现的被动测量

基于软件的测量是软件系统在没有任何特殊硬件的特别帮助下提供测量功能。软件实现的监控器（Monitor）往往需要系统内核的帮助。大多数内核包含了在应用程序和网络接口卡之间提供接口的网络堆栈。在正常的操作过程中，数据从网卡到达用户程序将出现两次过滤。首先是网络接口卡的过滤，它只传输需要本机处理的数据包到网络堆栈；接着是内核的过滤，它把到达网络堆栈的数据

包按照处理对象进行分类，需要内核本身处理的一部分直接由内核处理，另外的部分则转发给适当的应用程序来处理。第一层过滤可以通过把网络接口卡设置成混杂模式去掉；而对于第二层过滤，由于软件监控器是以应用程序来运行的，它要捕捉到以别的应用程序为目的的包就必须要有特殊的接口到网络堆栈。这个接口通常被称为 Pcap 接口（但不限于它，也可以重新写一个驱动程序来给测量软件提供接口），它是通过一个称为 libpcap(Unix/Linux，Windows 下为 winpcap)的 C 函数库来对网络堆栈进行存取，并传递给测量软件（如图 2-2）。

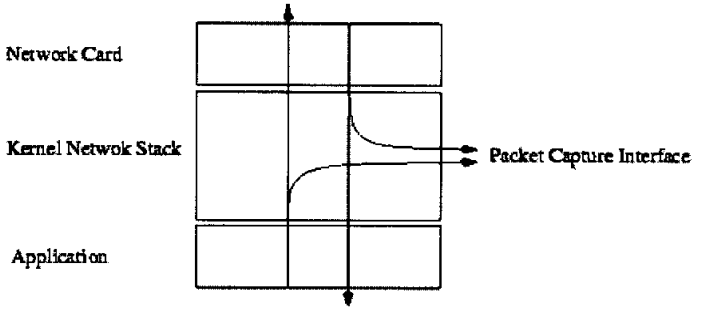

图 2-2: 软件监控器的包采集原理

libpcap是一个与系统无关，采用分组捕获机制的分组捕获函数库[25]。由于包捕获和过滤一般在是在内核层（网络堆栈）实现，其具体实现就必须依赖操作系统，如BSD使用BPF（Berkeley Packet Filter）机制；Linux使用SOCKET_PACKET类型套接口；而SVR4一般使用DLPI（Data Link Provider Interface）。这与软件的可移植性期望是背离的，libpcap解决了这个矛盾，它建立在包捕获和过滤模块的基础上，依赖于操作系统，但提供一套系统无关的调用接口供用户程序使用。在安装了libpcap的平台上，以libpcap为接口编写的应用程序能够在不同的操作系统上运行，具有良好的可移植性。

文献[25]对libpcap作了详尽的描述，这里援引其部分内容，以BSD为例对其结构作简单介绍。图2-3描述了libpcap在BSD中构建监听程序的原理，其结构一般可分成三个部分：Network Tap、Berkeley Packet Filter和libpcap。Network Tap是个探针，可以监听共享网络中的所有包；BPF用过滤条件匹配所有Network Tap监听到的包，若匹配成功则将其从网卡的缓冲区中复制到核心缓冲区中。libpcap隐藏了用户程序与操作系统内核交互的细节，主要完成以下工作：

1. 向用户程序提供了一套功能强大的抽象接口；

2. 根据用户要求生成过滤指令；

3. 管理用户缓冲区（User Buffer，对用户程序不可见）；

4. 负责用户程序与系统内核的交互。

软件监控器有着廉价和快速搭建等优点的同时，也存在着大量的延迟、时间戳不准确以及较高的丢包率等缺点。

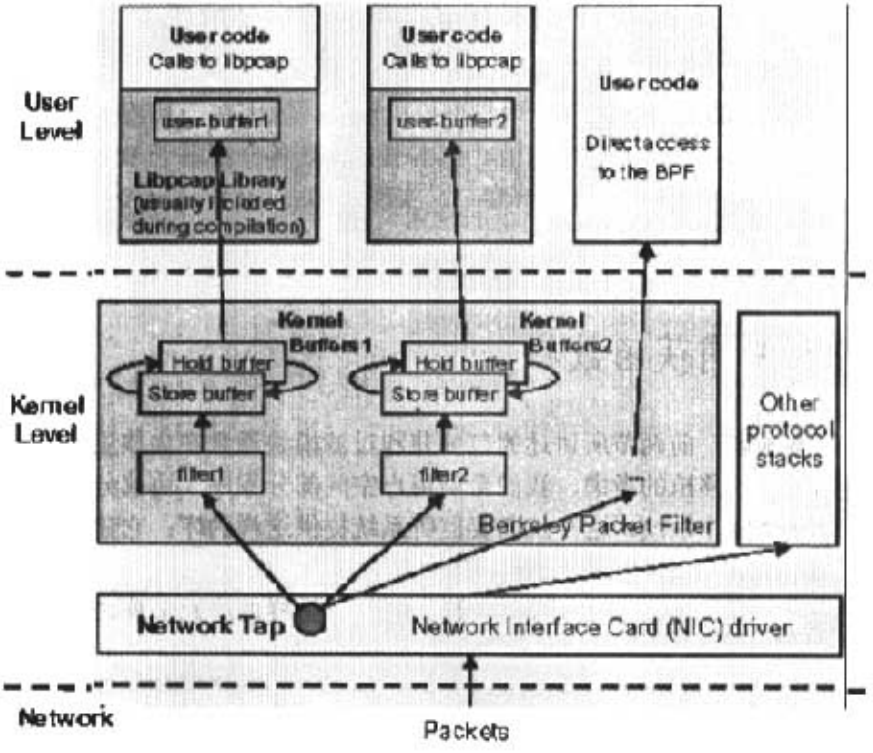

图2-3 基于BSD系统的监听程序结构[25]

基于硬件的测量系统试图去掉软件监控器的以上限制。在完全的硬件方案中，大部分的监控系统在定制的硬件上实施，而系统则用来保存和格式化数据。相对软件监控器，硬件方案允许监控器处理比软件监控器更快的数据速率。但存在成本较高，搭建周期太长等缺点。

2.3 tcpdump 简介

tcpdump是一个基于系统无关库libpcap的监听程序，它打印出监听的网络接口卡上与过滤表达式匹配的数据包头信息。tcpdump具有精确的时间戳、能够根

据用户要求，采集特定的数据包，如特定的主机、端口或协议等以及数据报文长度等。它是大多数Linux系统中默认安装的软件之一。

其使用格式如下：

tcpdump [-adeflnNOpqRStuvxX] [-c count] [-C file_size] [-F file]
[-i interface] [-m module] [-r file] [-s snaplen] [-T type]
[-U user] [-w file] [-E algo:secret] [expression]

其中较为常用的选项如下：

-i interface：指定监听的网络接口卡，如eth0指以太网编号0的设备，默认对编号最低的网络接口卡进行监听；

-s snaplen：指定tcpdump采集的数据长度，以字节为单位，默认为68字节；

-c count：采集count个数据包后结束；

-w file：采集下来的数据将写进file文件中，用于后续分析；

-r file：tcpdump的采集数据源来自于file。即读出file的首部信息并打印出来；

expression：过滤表达式。如host domain，表示只对跟主机domian的数据进行跟踪采集。

默认情况下，tcpdump将只采集数据包的前68个字节。这68个字节包含数据链路层信息、IP首部和TCP首部。如果对更高层的信息不感兴趣，只对数据的报文首部信息进行采集可用大大减少处理的数据量。

tcpdump将采集的数据的报文首部分析并输出到标准终端，如图2-4，其输出的通用格式如下：

ts src > dst: flags seqno ack win urg options

其中ts、src、dst 和 flags 总是被打印出来，而其它字段则取决于信息包的 TCP 首部内容，并仅在适当的时候输出。其各个字段意义如下：

ts　包到达时间戳，微秒单位。

src　指示源（主机）地址和端口。总是指定 src 字段。

dst　指示目的地址和端口。dst字段总是被指定。

flags　指定标志 S（SYN）、F（FIN）、P（PUSH）或 R（RST）的组合或单个.（句点）来指示没有标记。总是指定 flags 字段。

seqno　描述该信息包中数据占据的序列空间部分。（起始/结束（总长度））

ack　　指定（通过应答）在该连接的其它方向上预期的下一数据的序列号。

win　　指定该连接的其它方向上可用的接收缓冲空间的字节数。

urg　　指示该信息包中有紧急数据。

options　　指定在尖括号中的 TCP 选项（例如：< mss 1024>）。

```
16:33:04.091944 192.168.17.149.1520 > 192.168.17.252.2222: P 49:441(392) ack 24 win 16361 (DF)
16:33:04.091997 192.168.17.252.2222 > 192.168.17.149.1520: . ack 441 win 6432 (DF)
16:33:04.093840 192.168.17.252.2222 > 192.168.17.149.1520: P 24:568(544) ack 441 win 6432 (DF)
16:33:04.094386 192.168.17.149.1520 > 192.168.17.252.2222: P 441:465(24) ack 568 win 15817 (DF)
16:33:04.101236 192.168.17.252.2222 > 192.168.17.149.1520: P 568:784(216) ack 465 win 6432 (DF)
```

图2-4 tcpdump输出

2.4 TCP 报文分析工具 tcptrace 简介

tcptrace是由美国Ohio大学的Shawn Ostermann设计的用于对采集下来的TCP报文进行分析的软件，支持包括tcpdump在内的多种采集数据文件格式。tcptrace最新版本（6.6.7）能够实现以下几种功能：

1. 针对连接的统计信息：各种报文数量（如报文总数，重传报文数、响应报文数、同步和结束报文数量等），各种报文包含的数据字节数量，通告窗口的统计量；
2. 部分测量功能：包括对平均吞吐量的测量统计、对RTT的测量统计、利用某个RTT周期里发出去而明显未得到响应的数据量（owin）估算拥塞窗口值等，其拥塞窗口的估计并不是总是有效的。
3. 集成部分统计量的绘图功能。如吞吐量、RTT、owin、报文段的大小等等。tcptrace可以形成以连接为名称的绘图文件，绘图的实现需要借助xplot程序来完成。
4. 在分析方法上，实现离线和在线两种分析。离线分析是对某个已经保存下来的采集数据文件的分析，其用法如下：

   tcptrace [options] filename

在线分析则通过直接使用tcpdump采集，tcptrace现场分析的方法实现。其使用如下：

```
tcpdump -w - | tcptrace [options] stdin
```

在实际的统计过程中的各种统计和测量功能均开启，选项options只是决定tcptrace的输出方式以及是否生成绘图文件等。

在tcptrace中实现了对RTT的估计，其基本算法与TCP中的RTT相类似，但在两个方面做了改进：一)、TCP设计了一个定时器，只有在该定时器空闲的时候才对RTT进行测定，即其测量只是针对部分的RTT值， tcptrace则对每个对新数据的ACK报文均做了RTT的统计；二)、TCP对RTT的估计建立在定时器的基础上，其精度是500ms，而tcptrace直接采用tcpdump的时间戳，其时间精度在10^{-6}秒，即使考虑网络接口卡的缓存因素带来的影响，其时间精度仍然比TCP本身的估算要高得多。在后面的实现方案中，我们将直接利用tcptrace对RTT的测量结果。

tcptrace对拥塞窗口的估计也做了努力。但其估算拥塞窗口的方法是针对一个RTT周期，由发送端发送出去的但明显未得到响应的数据量（owin）作为拥塞窗口的估计值。这种方法测得的拥塞窗口值并不是真正的拥塞窗口值。本文的实现将根据TCP拥塞控制算法以重构方式来获取拥塞窗口值。

另一个与可用带宽有关的测量值为吞吐量，根据该连接传送的全部数据量与持续时间之间的商来估算。由于TCP的发送速率受拥塞窗口的控制，当重传出现时可能出现拥塞窗口的过度减小，从而使得实际的发送速率并不能完全利用可用带宽。它总是一个小于可用带宽的值。

第三章 TCP 拥塞控制及其实现

由于本文是利用对TCP拥塞窗口的测量来估计网络的可用带宽，所以我们必须对TCP的拥塞控制机制及其实现有深入的了解。传输控制协议TCP（Transfer Control Protocol）是整个TCP/IP协议的传输层，是面向连接的协议，其设计目的是在源端主机与目的端主机之间提供可靠的、按序传送的数据传输服务。其功能包括确保连接的可靠性、保证处于网络中的各个连接在带宽上享有的公平性、在动态的发现当前可用带宽的基础上避免网络流量过大而造成的网络拥塞或者网络崩溃等。作为本文算法实现的基础，本章描述了TCP的可靠传输机制、拥塞控制机制及其实现。

3.1 TCP 可靠传输机制

为了保证可靠性，TCP 实现了两点：一是使用面向连接的方式传输数据。源端和目的端在传送数据之前，先要发送三次握手的同步信号来建立连接，连接建立后才开始数传输数据，在数据传输完成以后拆除连接。二是包含确认的传输，即每个数据包的传输都要求对方以某种方式给予确认。此外，TCP 还对那些丢失的包和发生错误的报文制定了一套完整的重传机制。

TCP 报文要实现确认，必须对报文进行标识，该标识在每个连接里面均是唯一的，这就是 TCP 的编号，编号的另一个重要目的是要实现按序传送。TCP 将该连接所要传送的所有报文看成是一个字节组成的数据流，然后对其中的每一个字节进行编号。在建立连接的时候，两端主机使用同步握手报文协商初始序号，然后在这个初始序号的基础上增加每一个字节的偏移量作为自己的编号。连接一旦建立，序号的编排规则将一直坚持到该连接的终止。发送方将每次所传送的报文段的第一个数据字节的序号，放在 TCP 首部的序号字段中。而接收方也是根据这个序号来判断这个报文的相对位置。

TCP 在接收到一定量的数据后，总要对所接收到的数据向另一方发送确认信息。确认是通过放在 TCP 首部的确认（ACK）字段的一个序号来完成。它告诉

数据发送方其所期望接收的下一个字节序号来实现。由于 TCP 能够提供全双工的通信，即通信双方可以同时在两个不同的方向上发送数据，因此通信中的每一方都可以在传送数据时顺便把确认信息捎上。使用协议分析器 Ethereal 对 TCP 连接报文进行观察可以发现，在双方传送数据时，总是会把 ACK 的标志位置位，也就是说，双方在传输数据时总会带上最新的一个确认信息。

发送方把某个数据包发送出去以后并不马上把该数据包从缓存中删除，而是一直保持到接收到对方的确认信号，才清空该部分缓存。如果数据包出现错误，接收方也通过确认序号来通知发送方，从哪里开始有数据包出错。发送方可以直接从缓存中将该部分数据重新发送。若发送方在规定的时间（超时时间）内没有收到确认，会认为数据包可能已经丢失，自动将该部分数据重发。

由于IP协议中对每个独立的数据包都采用独立的路径寻址，因此同一个TCP连接的多个数据包可能因为所走的路径不同而产生到达时间的差异，从而造成乱序。对于数据包本身没有错误，而只是不按序号传送的数据包的处理，在 TCP 的协议中并未明确规定[26]。不同的实现者可以有不同的处理方法，丢弃，或者暂存在缓存中，等待补全。后一种策略对保证网络的性能更有好处，可以提高网络的传输效率和利用率。

3.2 TCP 的拥塞控制策略

TCP 采用可变发送窗口的方式进行流量控制，窗口的大小的单位是字节。TCP 的拥塞控制来自两个方面，源端的拥塞控制窗口以及接收端的通告窗口。前者根据网络状况估计当前网络能正常传送的数据量，通常以报文为单位；而后者则是接收方根据自己的资源（如缓存等）状况来通知对方自己能够接收的最大数据量。两者共同控制 TCP 的发送速率，实际的发送速率将由拥塞窗口和通告窗口的较小者决定。

TCP 拥塞控制四个主要过程简要介绍如下：

“慢启动”过程：早期开发的 TCP 实现在启动一个连接时会一下子向网络中发送大量的数据包，当很多主机都这样做的时候，将很容易导致路由器缓存空间耗尽，从而造成网络拥塞，使得整个网络的吞吐量急剧下降。“慢启动”算法建议，由于 TCP 发送端主机不知道网络资源当前的利用状况，因此新建立的 TCP

连接不能一开始就发送大量数据，而只能逐步增加同时发送的数据量，以避免拥塞现象的发生。具体地说，发送端主机在建立 TCP 连接时会设置两个参数，一个是拥塞窗口（congestion window，cwnd），另一个是“慢启动阈值”（slow-start threshold，ssthresh）。连接建立时，cwnd 初始化为一个数据包大小。发送方按 cwnd 大小与接收方通告窗口大小的最小值发送数据，若收到了对所有发出的报文段的确认，就在下一次发送时将拥塞窗口加倍[26]，这里所说的一次是假设接收方不进行流量控制的情况下，发送方总是整块发送拥塞窗口容许的报文数量，发送完成一个 cwnd 值的报文并都收到响应，成为一次。这样 cwnd 就将随着来回时间（Round Trip Time，RTT，该 RTT 也是上面所说的一次为测量周期。）呈指数增长，发送端向网络发送的数据量将急剧增加，如图 3-1，其中的时间是以次数为单位的。可见，“慢启动”一点也不慢，要达到每个 RTT 时间内发送 W 个数据包所需时间仅为 $RTT*\log_2 W$[27]。由于在发生拥塞时，拥塞窗口会减半或降到 1，因此“慢启动”确保了发送方的发送速率最多是目前链路可用带宽的两倍。

拥塞避免阶段：在连接建立的时候，会设置 ssthresh，该参数是用来划分 TCP 连接的“慢启动”阶段和拥塞避免阶段的。如果 cwnd 的值比 ssthresh 大，则进入拥塞避免阶段，否则为“慢启动”阶段。如果 TCP 发送方探测到超时或 3 个相同 ACK 副本时，即认为网络发生了拥塞（因为由传输引起的数据包损坏和丢失的概率很小）。此时，ssthresh 被设置为当前拥塞窗口大小的一半，但最小不能小于两个数据包单元；同时 cwnd 也被置为与 ssthresh 相等的值，如果 ssthresh 为两个报文大小，则 cwnd 被置为一个报文。然后，再收到下一个对新的数据的 ACK 确认时， cwnd 将大于 ssthresh，TCP 进入了拥塞避免阶段，执行拥塞避免算法。此时，cwnd 在每次收到一个新的 ACK 确认时只增加 1/cwnd 个报文大小。这样，在完成一次的传送的时间内（一个 RTT），cwnd 将只增加 1，如图 3-2。可见，在拥塞避免阶段，cwnd 不再是指数增长，而是线性增长。

快速重传和快速恢复：在收到一个失序的报文段时，TCP 需要立即产生一个 ACK（一个重复的 ACK），告诉发送方报文失序，并指出希望得到的报文序号。发送方并不知道重复的 ACK 是因为报文失序还是报文丢失而导致，因此它会等待少量的 ACK。假如这只是一些报文段的重新排序，则在重新排序的报文段被处理并产生一个新的 ACK 之前，只可能产生 1～2 个重复的 ACK。如果连续收到 3 个或 3

个以上的 ACK 副本，就非常可能是一个报文段丢失了[28]，即认为网络发生拥塞，发送方不等待超时定时器溢出，马上重传其认为可能已经丢失的数据包，这种算法能够比超时重发所需要等待的时间要小，被称为快速重传。同时，发送方认为已经发生拥塞而执行回退算法：将 ssthresh 设置为当前 cwnd 值的一半，并且将 cwnd 减半。而快速恢复则是基于“管道”模型（pipe model）的“数据包守恒”原则(conservation of packets principle)，即同一时刻在网络中传输的数据包数量是恒定的，只有当“旧”数据包离开网络后，才能发送“新”数据包进入网络。由于重复的 ACK 是在接收方接收到期望外的报文所触发，这表明有期望之外的报文离开网络而进入接收方的缓存中。因此发送方接收到一个重复的 ACK，则认为已经有一个数据包离开了网络，于是将拥塞窗口加 1。如果“数据包守恒”原则能够得到严格遵守，那么网络中将很少会发生拥塞；本质上，拥塞控制的目的就是找到违反该原则的地方并进行修正。假如发送方接收到 3 个相同的 ACK，则会认为有报文丢失， TCP 将 ssthresh 和 cwnd 减半，此外，接收方还认为有三个报文已经离开网络，然后马上用 cwnd+3 替代 cwnd。这样的算法过程就是快速恢复。

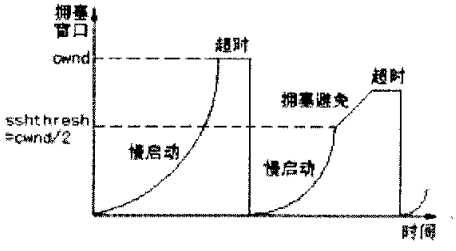

图 3-1：“慢启动”和拥塞避免

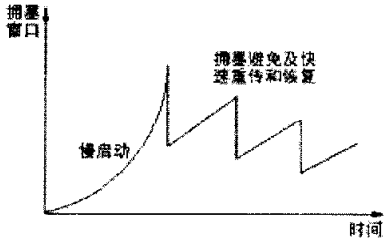

图 3-2：快速重传和快速恢复

3.3 TCP 对 RTT 的测量

在 TCP 中，RTT（Round—Trip Time）是一个用来对超时定时器进行设置的重要变量。RTT 指的是在发送方发出数据开始，到接收到该数据的 ACK 信号之间的这个时间间隔。文献[28]对 TCP 的 RTT 测量做了详细描述。大多数源于 Berkeley 的 TCP 实现在任何时候对每个连接均仅测量一次 RTT 值。在发送一个报文段时，如果给定连接的定时器已经被使用，则该报文段不被计时，如图 3-3（文献[28]图 21-2），由于定时器被序号 3 的报文占用，对序号 4 的报文不再进

行计时。如果定时器没有被占用，则对此时出现的报文进行计时，如报文 1、3、6。在每次调用 500 ms 的 TCP 的定时器例程时，就增加一个计数器来完成计时。这意味着，如果一个报文段的确认在它发送 550 ms 后到达，则该报文段的往返时间 RTT 将是 1 个滴答（即 500ms）或是 2 个滴答(即 1000ms)。此时除了记录该报文发出的时间外，报文段中的起始序号也被记录下来。当收到一个包含对该序号报文的确认后，该定时器就被关闭。如果 ACK 到达时数据没有被重传，则被平滑的 RTT 和被平滑的均值偏差将基于这个新的测量值进行更新。

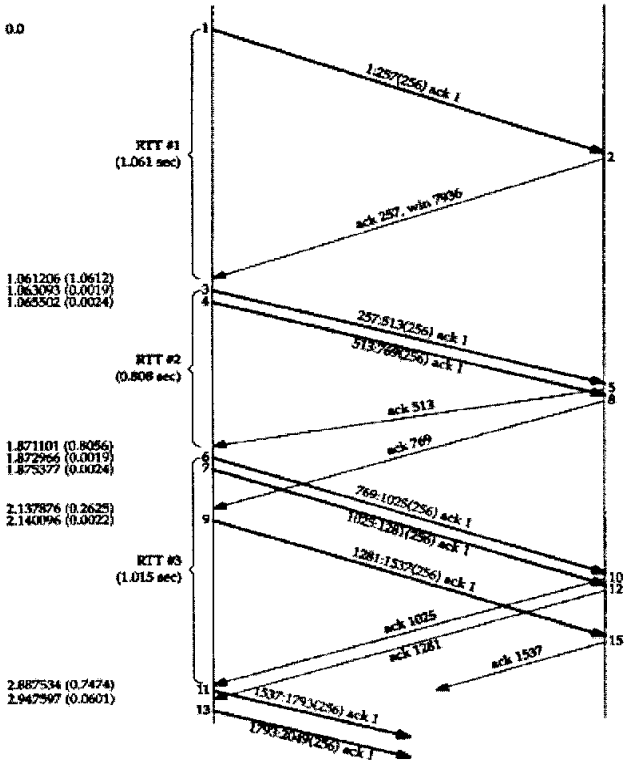

图 3-3：TCP 中对 RTT 的测量[28]

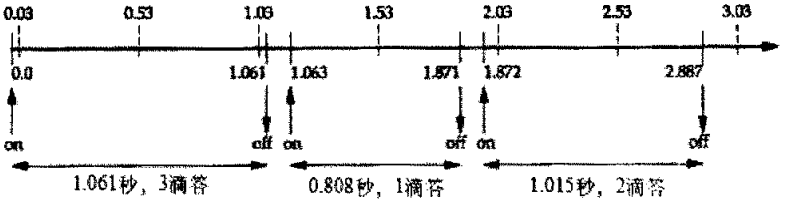

图 3-4：RTT 测量与时钟滴答[28]

图 3-4（文献[28]图 21-2）中连接上的定时器在发送报文段 1 时启动，并在

确认（报文段 2）到达时终止。尽管它的 RTT 是 1.061 秒（tcpdump 的输出），但插口排错的信息显示该过程经历了 3 个 TCP 时钟滴答，即 RTT 为 1500ms。可见，在 TCP 本身的实现中对于 RTT 的测量比较粗略的,其精度是以 500ms 为单位的。

3.4 TCP 的实现：Reno & Vegas

TCP 的 Reno 版本和 Vegas 版本实现的主要差异体现在它们所使用的拥塞控制方法的差别。Vegas 增强了 Reno 的拥塞避免算法，它通过监测 RTT 的值，动态地增/减 TCP 的发送窗口；而 Reno 则是在检测到报文丢失之前连续增加包的大小。

3.4.1 TCP Tahoe/Reno/New Reno 的拥塞控制

TCP Tahoe、Reno 和 New Reno 三个版本的拥塞控制有很多共同的地方，其中的后者是前者的改进版本。它们的拥塞控制窗口在检测到报文丢失前连续的增加。它里面包含两个过程:“慢启动”过程和拥塞避免过程,“慢启动”阀值 ssthresh 是为了区分两个过程的变量。当发送方在 $t+t_1$ 时刻（假设该时刻为继 t 时刻之后的第一个响应报文到达时刻。）接收到对方发来的 ACK 确认报文，TCP 将对其拥塞控制窗口值进行更新，如果没有检测到报文丢失，则有：

$$\mathbf{cwnd(t+t_1)}=\begin{cases}\mathbf{cwnd(t)+1} & \mathbf{if\ cwnd(t)<ssth(t)；“慢启动”}\\ \mathbf{cwnd(t)+1/cwnt(t)} & \mathbf{if\ cwnd(t)\geq ssth(t)；拥塞避免}\end{cases} \tag{3-1}$$

其中 ssth(t)指 t 时刻的“慢启动”阀值，cwnd(t)是 t 时刻的拥塞窗口值。如果 TCP 检测到超时重传，则其拥塞窗口值将按下面算法进行更新：

$$\mathbf{cwnd(t+t_1)=1；\ ssth(t+t_1)=cwnd(t)/2} \tag{3-2}$$

上面所述的这些算法在 Tahoe 版本中就已经存在，TCP Reno 对 Tahoe 版本的主要改进是增加了快速重传和快速恢复算法，当 TCP 以快速重传算法检测到报文丢失，即连续收到 3 个或 3 个以上相同的 ACK 副本的时候，其拥塞窗口更新如下：

$$\mathbf{ssth(t+t_1)=cwnd(t)/2\ ；\quad cwnd(t+t_1)=ssth(t+t_1)} \tag{3-3}$$

在这一阶段，TCP 发送方将紧接着运行快速恢复算法，发送方接收到一个

ACK 副本，则拥塞窗口值加 1，如果收到的 ACK 是对新的数据的确认，则又将拥塞窗口值恢复到 ssth，开始一个新的拥塞避免过程。TCP New Reno 是对 Reno 的一个更新，主要解决了对同时丢失多个报文在 Reno 中对 TCP 性能的损坏问题的处理。

3.4.2 TCP Vegas 的拥塞控制

在TCP Reno中，如果发现有报文丢失，拥塞窗口值恢复到1或者减小到原来的一半，以这样的方式减少TCP的吞吐量来缓解网络拥塞。TCP Reno存在一个严重的缺点：它只能通过报文丢失（超时或者快速重传）的方法来检测网络的拥塞，因此对网络拥塞的程度不了解。其本身设计的回退算法对缓解严重的拥塞情况是卓有成效的，但对于由于其自身的窗口过大造成的拥塞，或者因为报文错误引起的重传（虽然这样的几率很小），这样的回退算法显然过分地减少了其窗口值，从而降低其有效的吞吐量。TCP Vegas就是针对这些方面对TCP Reno进行了改进。Vegas是基于对RTT的监测来对拥塞窗口进行更新的，如果监测到RTT增大了，发送方认为其可能已经发生拥塞或者即将发生，减少其发送窗口；另一方面，如果RTT减小了，TCP认为网络状况正在改善，拥塞情况正在或者已经减轻，则增加其发送窗口。在拥塞避免阶段，其算法如下：

$$\mathbf{cwnd(t+t_1)}=\begin{cases}\mathbf{cwnd(t)+1}, & \mathbf{if}\ diff < \dfrac{\alpha}{base\_rtt}\\ \mathbf{cwnd(t)}, & \mathbf{if}\ \dfrac{\alpha}{base\_rtt} \le diff \le \dfrac{\beta}{base\_rtt}\\ \mathbf{cwnd(t)-1}, & \mathbf{if}\ \dfrac{\beta}{base\_rtt} < diff\end{cases} \tag{3-4}$$

其中，$diff = \dfrac{cwnd(t)}{base\_rtt} - \dfrac{cwnd}{rtt}$，rtt为当前监测到的RTT值，而base_rtt是RTT的最小值。α 和 β 是两个常数。

而在“慢启动”过程里，其拥塞窗口的更新速度比Reno的慢一半，即每接收到两个对不同数据的ACK响应报文才对拥塞窗口进行加1。研究表明，TCP Vegas比TCP Reno的吞吐量将有40%的增量[29]，但其在与TCP Reno的并存的情况

下不能获得平等共享瓶颈带宽的待遇，在与多个Reno共存的瓶颈中，Vegas会总是认为自己在跟单一的Reno进行带宽竞争[30]。

第四章 基于 TCP 的可用带宽被动测量方法

基于TCP的被动测量方法是利用TCP层或者TCP层以下的信息来实现对网络状况（带宽与可用带宽）的测量方法。TCP层是TCP/IP协议体系中的运输层，它的优秀在于其使用了一套完整的机制来实现数据报文的可靠传输，而尤其重要的是，它设计了一套行之有效的针对网络状况的探测方案，并在此基础上实现对自身流量的控制。因此，TCP本身便给我们提供了许多可以利用的参数，而我们的网络工程师们也在这方面做了相当多的努力和工作。本章将对已有的利用TCP层信息进行带宽和可用带宽测量的算法进行介绍。

4.1 基于 TCP 拥塞窗口方法估计带宽

TCP拥塞窗口方法是利用TCP的拥塞窗口值以及测量得到的RTT值来估计带宽的一种算法，文献[31] [32] 对这种算法做了详细的描述，它们在推导的时候有些细微的差别。文献[31]给出了TCP拥塞窗口方法的完整模型，其基本描述如下：

假设：1)、TCP运行于一个有丢包的通道，该通道具有常数的RTT，它具有充足的带宽和足够低的总体载荷，不需要维持任何排队；2)、近似的认为它以常数概率p随机丢失包。根据假设，链路在丢失一个包后，将大约运送1/p个连续包。

令窗口的最大值为W个包，根据拥塞避免的定义，我们知道达到平衡时，最小的窗口必然是W/2个包，此时的拥塞窗口的增加保持为拥塞避免阶段。如果接收者对每个报文段都进行响应的话，则窗口的增加在每个来回中将被每一个包激活，所以每个周期必然是W/2个来回，或者RTT*W/2秒。在前面所述的假设情况下，拥塞窗口将形成完美的周期性锯齿，如图4－1。则每个周期锯齿下的面积就是该周期传送的总数据量，即$(\frac{W}{2})^2+\frac{1}{2}(\frac{W}{2})^2=\frac{3}{8}W^2$。通过假设，每个周期也运送1/p个包，则有$W=\sqrt{\frac{8}{3p}}$。

在文献[32]中，假设条件完全一样，丢包率也为p。每个周期只有一个丢包

的机会，因此有：$p=\frac{1}{\frac{w}{2}+(\frac{w}{2}+1)....+(w-1)+w}=\frac{1}{\frac{1}{2}*\frac{w}{2}(\frac{w}{2}+w)}=\frac{8}{3w^2}$，由此可得：$w=\sqrt{\frac{8}{3p}}$。

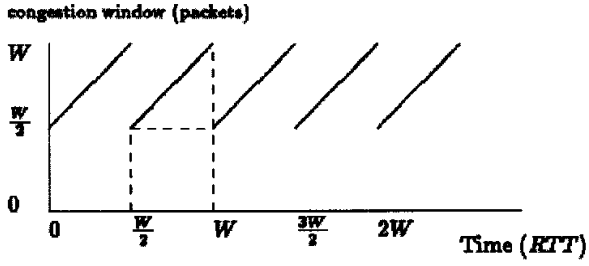

图 4-1：TCP 窗口随 RTT 的变化[31]

替换 W 到下面的带宽方程中得：

$$BW=\frac{data-per-cycle}{time-per-cycle}=\frac{MSS*\frac{3}{8}W^2}{RTT*\frac{W}{2}}=\frac{MSS/p}{RTT\sqrt{\frac{2}{3p}}} \quad (4\text{-}1)$$

BW 为带宽，把所有的常数集合成为 C，即 C=$\sqrt{\frac{3}{2}}$，可以得到：

$$BW=\frac{MSS}{RTT}*\frac{C}{\sqrt{p}} \quad (4\text{-}2)$$

该模型描述了一种测量端到端链路带宽的方法。它假设网络处于一种轻负载的情况底下，并要求对TCP的流量控制完全由发送方的拥塞窗口控制的情况下，TCP自身增长的发送窗口值而造成拥塞，然后根据丢包的概率来估计这一链路的带宽。

4.2 基于 TCP 拥塞窗口方法估计可用带宽

基于TCP拥塞窗口方法估计可用带宽的算法其实是从利用拥塞窗口估计带宽[31]引申出来的。在公式4-1中，取其前面部分，可得：

$$BW=\frac{data-per-cycle}{time-per-cycle} \quad (4\text{-}3)$$

这在网络几乎没有负载的情况下，这代表了该链路的带宽，实质上在这样轻载荷的假设下，带宽跟可用带宽是可以互换的。因此，考虑负载正常，或者负载较重的时候，可用带宽（被测的这个连接可以使用的带宽）就可以表述为在特定的时间周期里能够正确传送的数据量，化为标准单位时则为bytes/sec。

拥塞窗口在TCP当中实现的功能是对发送速率的控制，不管是Tahoe、Reno、New Reno还是Vegas、Westwood，其拥塞窗口值要么直接来自对可用带宽的估计，要么来自于与可用带宽相关的值，如Reno中的报文丢失表明拥塞发生，Vegas的RTT增大反映被测链路有拥塞的趋势等。为了更好的利用网络，TCP鼓励每个连接尽可能多的占用网络带宽以达到网络的最大利用，在数据报文出现丢失以前或者其RTT变得足够长之前，其拥塞窗口值将一直增加或者保持某个常数，直到出现报文丢失或者RTT过大为止。换句话说，TCP尽可能的向网络注入流量，在网络可用带宽能够承受的情况下，注入的流量一直处于增长或者保持某个常数的状态，增长到一定程度以后，网络可用带宽被占用完毕，网络不能及时传送所有进入网络的流量，就出现报文丢失或者RTT的急剧增大（报文丢失可以认为RTT为无限大）。那么这个报文丢失和RTT急剧增大的临界点，就代表了该连接所能够使用的最大可用带宽值。这就是出现报文丢失或RTT急剧增大前一刻的那个拥塞窗口值就代表了一个时间周期里能够传送的最大数据量（其实正确传送的数据量应当减去该时刻丢失的报文数据量）。由于拥塞窗口值表示的是发送方可以同时发送的数据量，而当发送方一下子发完这些数据，它在收到新的数据确认以前不能发送新的数据。因此，时间周期就是一个RTT。因此，由4-3式我们可以得到利用拥塞窗口实现可用带宽估计的算法：

$$BW = \frac{data - per - cycle}{time - per - cycle} = \frac{cwnd_{\max} * MSS}{RTT} \tag{4-4}$$

其中$cwnd_{max}$指的是拥塞窗口的峰值。而RTT则是该窗口值的保持时间，之所以要乘以MSS（最大数据单元），是因为拥塞窗口值总是以报文为单位的。如果不产生报文丢失，即cwnd一直保持增大，则cwnd取最大值[36]。

文献[36]对这种算法进行讨论的目的仍然是希望能够获得一种有效估计可用带宽的算法来设置适当的“慢启动”阀值（ssthresh）。在实践当中将拥塞窗口算法（3-7）用于其他目的的可用带宽测量目前只有威廉玛丽学院的Bruce B.

Lowekamp和Marcia Zangrilli所设计的Wren带宽监测工具[37]。文献[38][39]中也给出了一些测量结果，但这两篇文献中的主要思想在于将被动测量和主动测量结合起来，在有业务流量发生的时候使用被动方法对端到端的可用带宽测量，而在没有业务流的时候采用主动测量的方式。对于拥塞窗口方法测量可用带宽的算法，并没有进行改进。Wren的发布网站也没有给出具体的程序，使得本文无法与该算法的已有实现进行对比实验。

4.3 基于 TCP 吞吐量估计可用带宽

基于TCP的可用带宽估计研究的焦点仍然在于优化TCP性能本身。当TCP的某个连接找到一种方法来估计它所能够使用的带宽，即可用带宽，它可以根据可用带宽来控制其数据发送速率，如果所有TCP都能实现这样的可用带宽探测并很好的控制各自的发送速率，则整个网络将能够得到最大的利用率和发生最少的拥塞状况。TCP Reno版本中对于网络的拥塞探测并不是基于带宽值，而是基于报文丢失的探测方法，这样的方法在拥塞发生后总是大幅度的减小发送窗口，事实证明乘性减少的方法限制了TCP的性能，降低了它的吞吐量。而TCP Vegas使用RTT来对主机的发送窗口进行控制。研究者们对如何使用可用带宽来直接控制TCP的发送速率进行了思考。针对无线环境的TCP Westwood是其中的典型。

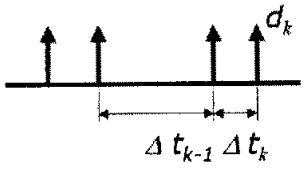

图4-2：相邻ACK及其时间区间[35]

在TCP Westwood中，人们尝试了使用吞吐量来获得可用带宽的近似方法。其基本思想如下：

假设一个由发送方单独控制发送速率的网络环境中，发送方在时刻t_k接收到对方的ACK报文信号，这意味着有一定的数据量d_k已经成功的传送到接收方，如图4-2，此时被该TCP连接使用的带宽值b_k可以按下面的公式[33][35]计算：

$$b_k = \frac{d_k}{t_k - t_{k-1}} = \frac{d_k}{\Delta k} \tag{4-5}$$

其中t_{k-1}是收到前一个ACK的时间，Δk是两个相邻的ACK之间的时间区间。由公式4-5获得的b_k是一个ACK到达时刻t_k获得的采样值，令BE_k为t_k时刻通过一个一阶低通滤波器得到的平滑平均值，令α_k为t_k时刻时变的指数滤波器系数，TCP Westwood的平滑滤波器为：

$$BE_k = \alpha_k BE_{k-1} + (1-\alpha_k)(\frac{b_k + b_{k-1}}{2}) \tag{4-6}$$

其中$\alpha_k = \frac{2\tau - \Delta t_k}{2\tau + \Delta t_k}$，$1/\tau$为滤波器的截止频率。

然后根据得到的可用带宽估计值BE_k按以下方式设置ssthresh的值[34]：

如果超时或超时前接收到三个相同的ACK副本，则判断为拥塞发生，TCP重置cwnd和ssthresh，其中cwnd的处理与TCP Reno中的一样。而ssthresh的设置则按如下算法进行：

ssthresh＝max（（BE_k*RTT_{min}）/ seg_size，2） (4-7)

其中RTT_{min}是RTT的最小值，BE_k为最新测得的可用带宽估计值，BE_k *RTT_{min}是一个RTT中能够安全传送的最小数据量的估计值，而seg_size是报文的长度。就这样，TCP Westwood利用动态测得可以安全使用的带宽作为“慢启动”阀值，用于消除随机错误引起的报文丢失对TCP发送窗口带来的过度减小。

4.4 拥塞窗口方法和吞吐量方法存在的问题

拥塞窗口方法的估算可用带宽的原理可用图4-3a表示，图中的拥塞窗口为3个报文时，没有丢包，当拥塞窗口增加到4个报文大小的时候，产生一个丢包，按照拥塞窗口方法的算法将在此时获得一个拥塞窗口的采样值和对应的RTT值。并按照cwnd/RTT的公式参与到最终的可用带宽的计算中。这里可以发现拥塞窗口方法中存在的一些问题：

1)、此时记录下来的cwnd包含了丢失报文本身，但是丢包证明可用带宽不能在RTO溢出前把cwnd个报文全部传送到客户端，实际完成传输的报文数量应为：cwnd－k，其中k指丢失报文数量；

2)、测量的算法思想是基于一个时间周期能够完成的数据传输量，而实际上报文从服务器到达客户端的时间（单向延迟）完成的由服务器端传输给客户端的

数据才真正体现可用带宽。在t时间内，报文已经到达接收端，也就是说，RTT是个来回时间，但返回的时候很可能没有数据，或者有数据捎带回服务器却并没有被计算在内，即多计算了一段由客户端返回到服务器端的ACK报文的传输时间。

3)、拥塞窗口算法要求在测量中发送端的发送速率完全由拥塞窗口来控制，这并不总是与实际相符，如果测量样本获得时的报文发送速率是由接收方的通告窗口控制（即此时通告窗口值小于拥塞窗口值）的，将导致错误的测量。

4)、算法中需要对拥塞窗口值进行提取，原有的算法从服务器主机上通过接口方式访问提取拥塞窗口值，这将对服务器的性能造成附加影响，不符合无声测量和独立测量的要求。

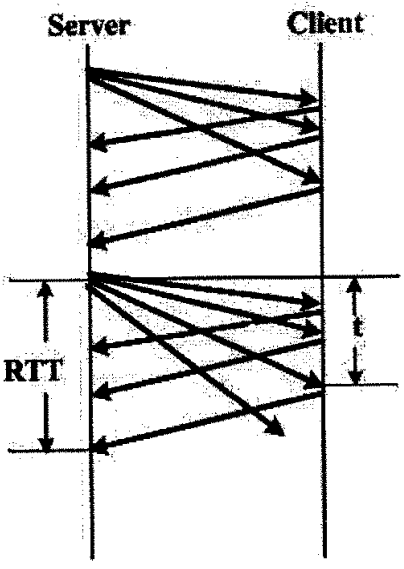

图4-3a

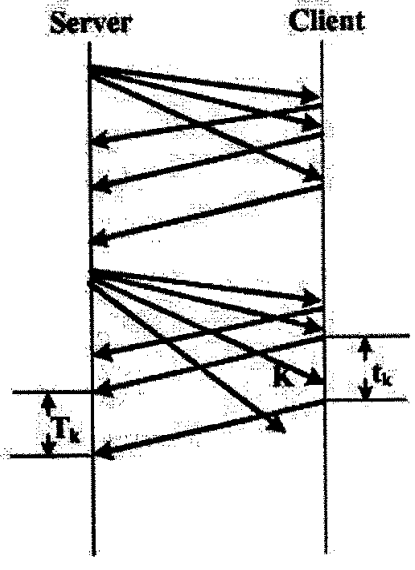

图4-3b

利用吞吐量的可用带宽测量方法可以使用图4-3b来示意，它使用新响应的数据量与相邻两个对新数据进行确认的ACK报文间的间隔的商来获得。图中的T_k为服务器收到的两个ACK的时间间隔，设相邻的ACK均为不含负载的空报文，相邻的两个ACK在网络中传输的时间应该非常接近，可以认为其传输时间线是平行的。此时，便有$T_k=t_k$，不考虑延迟响应的影响，则t_k可以认为是第k个报文与其前一个报文到达接收方的时间间隔。在没有交叉流的情况下，该算法也可以用图4-4表示。两个报文由发送端背靠背发出，经过瓶颈的时候被拉长，产生时延，其值为$\Delta t = t_k - t_0$，其中t_0为发出的时候两报文的时间间隔，等于包长除以

发送方的网卡速率。按100M以太网卡，1500字节的报文算，t_0=0.12ms，可以忽略。因此有$\Delta t = t_k$。可见，吞吐量方法的实质是测量除瓶颈处的可用带宽来，从而得到整条链路的可用带宽。

图4-5显示了在有交叉流的时候可能出现的一种情况，同样是背靠背发出的报文，在瓶颈处也被拉长，造成时延t_b，而在经过瓶颈以后，如果在两个报文到达下一跳（或后面的其中一跳）的时候，前面已经有报文在等待处理，即两个报文在下一跳路由器中（或后面的其中一跳）需要排队。此时，两个报文之间的时间间隔将会缩小，甚至可能重新变成背靠背的情形。此时将出现t_k不等于t_b的现象，甚至视路由器的情况可能出现$t_k<t_0$的情况。这表明，测量的两个响应报文的延迟并不总是瓶颈处的时延。也就是说，利用吞吐量方法测量出来的值并不总是可用带宽值，尤其是在网络负载较重的时候。

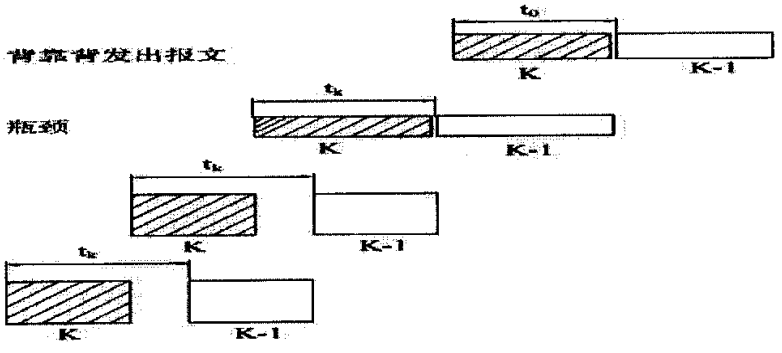

图4-4 吞吐量方法测瓶颈可用带宽原理（无交叉流）

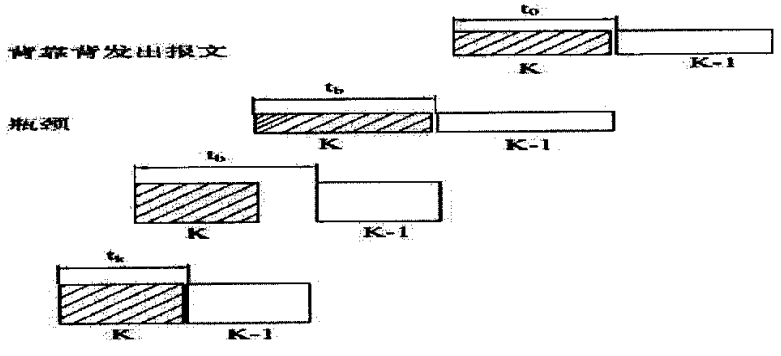

图4-5 吞吐量方法测瓶颈可用带宽（有交叉流时可能出现的情况）

由此可以对吞吐量的算法做出如下评价：在没有交叉流的时候，用它来测量瓶颈处的可用带宽（此时应为带宽）总是有效的，但在有交叉流的情况下，它并不总是适用的。而且，如果返回的ACK信号是由业务报文附带捎回（实际的应

用当中经常出现这样的情况），则并不总是可以认为T_k与t_k近似相等。同时，由于该算法的设计目的是为了TCP更好的设定“慢启动阈值”，同样是在服务器端的实现。

比较两种算法，显然拥塞窗口方法更贴切的体现了可用带宽的概念。因此服务器前端基于被动的可用带宽测量算法将在其基础上进行改进，使其符合课题的测量需要。

第五章 服务器前端基于被动的可用带宽测量

本章在前述章节的基础上提出服务器前端基于被动的可用带宽测量的方案，并在对已有的测量算法（拥塞窗口算法与吞吐量方法）评价的基础上，提出对拥塞窗口方法的改进建议，对其中的粗测量算法和精确测量算法进行探讨，并结合课题提出服务器前端基于被动的可用带宽测量的算法。

5.1 服务器前端基于被动的可用带宽测量

服务器前端基于被动的可用带宽测量原理如图5-1所示，在服务器前端架设测量系统（模块），利用探针技术、监听端口或者以太网的广播特性对进出服务器的流量进行采集，针对业务量大的客户（一般是某些代理服务器），测量其端到端的可用带宽。

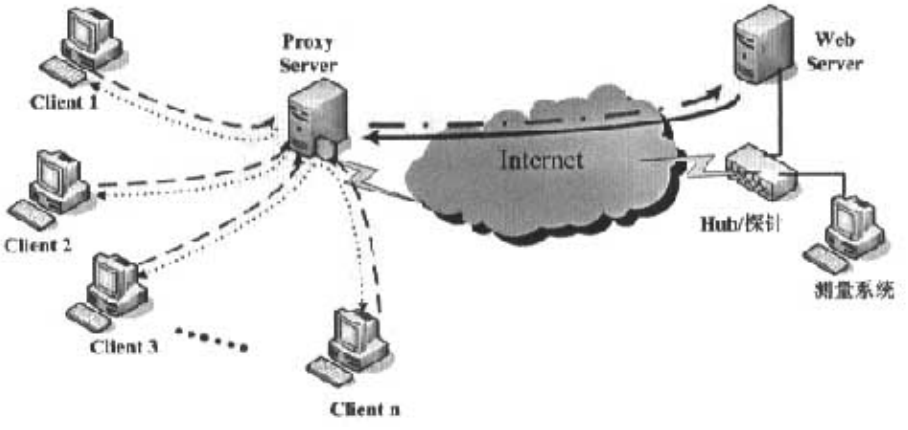

图5-1 服务器前端被动测量结构

服务器前端测量指出了测量的位置是在服务器附近，测量系统与服务器利用探针、监听端口或者共享型集线器相连，中间不再有路由器或其他转发设备。此时，数据报文到达测量系统的时间可以近似地认为是到达服务器的时间，这相当于在利用独立设备在服务器端的测量。

在测量方法上采用被动测量，只对网络现有的数据进行采集和分析，而不人

为注入任何附加的信息。测量过程中，数据的采集和分析工作都由独立的模块或机器实现，在服务器遭受攻击的时候，测量系统因其良好的隐蔽性而仍然能够正常工作。

测量的对象是业务关系密切的客户或代理与服务器之间的端到端的可用带宽。端到端的可用带宽反映了某条特定链路的当前通信状态，因此它可以作为该链路是否正常的判断参考。对可用带宽的测量，正是为了能够向上级系统提供可靠的用于服务器外围网络状态分析的原始数据。

本文课题是所在课题组承担的国家自然科学基金项目“大型活动网站的时空特性”的子课题，该题目配合整体课题期望实现所需目的，其主要的目标如下：

1. 独立测量。实现相对于网络和服务器独立的测量，除服务器前端需要添加探针或者共享性集线器外，不需要对服务器、客户端或者其他中间设备进行硬件或者软件的改变来进行配合，测量环境搭建起来以后，服务器、客户端以及中间设备的任何更改都不对测量系统造成影响；
2. 无声（透明）测量。测量的整个过程中仅进行监听，而不向网络添加任何流量，不对网络以及服务器的性能产生任何附加影响；测量系统对不管是网络、服务器还是客户端，都是透明的；
3. 快速测量。实现对网络可用带宽的测量和相关计算，要求能够在需要的时候尽可能快的提供所需的测量数据，最终目标是希望能够实现实时测量，至少也要能够在线测量；
4. 在以上几点的基础上，尽可能提高测量的准确性；

在第四章中介绍了利用TCP拥塞窗口或者吞吐量对可用带宽进行估计的算法，本文针对拥塞窗口方法的不足提出以下修改建议：

1)、当测量样本产生时，如果发送端的拥塞窗口值大于接收端的通告窗口值，则对发送速率起实际作用的是接收端通告窗口值，本文将使用通告窗口值代替拥塞窗口值进行测量。即实际测量的样本值是对发送速率起控制作用的值，即拥塞窗口和通告窗口的较小值；

2)、报文发送的数据量对应的时延应为单向时延，我们把服务器前端可用带宽的被动测量算法分为粗测量和精确测量，在精确测量的算法中将给出根据RTT对单向时延进行估计的办法，并使用单向时延实现精确的可用带宽测量；

3)、当测量样本产生时（即报文丢失时），如果服务器的发送速率受拥塞窗口控制（拥塞窗口小于通告窗口值），则测量中使用的发送报文数量将考虑丢失报文的影响。这一点在通告窗口值小于拥塞窗口的时候也应当考虑，但此时常常存在多个报文同时丢失的情形，鉴于对多个报文同时丢失的情况检测起来有一定的困难，可以先实现每次的报文丢失数量均为1的情况；

4)、在实现上，根据拥塞控制算法在中间设备上对发送方的拥塞窗口值进行模拟重现。由于初始值的影响，拥塞窗口值的重构将受到必须获得同步信号报文的限制。

下一节将给出服务器前端基于被动的可用带宽测量的粗测量算法和精确测量算法的详细介绍。粗测量算法将在拥塞窗口算法的基础上，参考修改建议1、3、4来进行改进，并使用活跃区间对可用带宽的范围进行描述，具有实现容易以及更快速的实现测量的特点。精确测量则把第2点修改意见也引进算法中来，由于涉及到K值测量、报文及其响应报文的长度提取等，实现起来相对较难，并且其测量所需的时间也较长，但它能够对可用带宽的值进行较精确的测量。

5.2 利用拥塞窗口算法测可用带宽的活跃区间

我们把可用带宽的可能变化范围定义为其活跃区间，对活跃区间测量的过程是对可用带宽进行粗测量的过程，并不给出可用带宽的确定值。进行粗测量的目的有两个：一是在不需要对可用带宽进行精确测量的时候实现快捷测量；二是用于估计被测链路当前的排队状况，为可用带宽的精确测量提供参考（如果精确测量需要的话）。

TCP拥塞控制机制是一个通过对可用带宽的探测，给TCP服务提供反馈信息，对新的发送速率进行调节的闭环控制系统。其反馈信息是报文的丢失（通过超时和重传进行控制），当出现报文丢失的时候，表明发送速率已经超过可用带宽，TCP将通过减小拥塞窗口值来对其进行调节，此时拥塞窗口将出现由缓慢增加到大幅减小的跳变，从而导致极大值的出现。

TCP 拥塞窗口的变化过程如图 5-2 所示。连接建立以后，TCP 将根据网络的拥塞情况对 cwnd 进行更新。开始的时候处于“慢启动”阶段，拥塞窗口值按指数方式增加，进入拥塞避免阶段以后，将按线性方式增加。设 BW_m 为在第 m 个

拥塞窗口峰值所测得的可用带宽，用 cw_m 表示第 m 个拥塞窗口峰值，如果该峰值大于当时的接收方通告窗口，则为通告窗口值，即 cw_m 表示出现拥塞窗口峰值时（也即报文丢失前一时刻的值）的拥塞窗口与通告窗口的较小值。在该峰值拥塞窗口值出现的时候，其对应的往返时间为 RTT_m。cw_m 代表了在它持续的时间内 TCP 实际发送的报文数量，而拥塞窗口出现峰值时则代表了当时的网络所可能同时发送的最大报文数量。这与耗尽式的带宽测量方法相似（如 pathload），耗尽式的带宽测量是采用主动方式向网络注入流量，逐渐增加注入流量的大小，使得网络可用带宽减少，当注入的流量超过网络能够处理的能力的时候，就会出现报文丢失或者 RTT 的明显延长，则注入流量的速率表现特征（RTT、丢包等）的转折点就代表了网络的带宽值。这里不同的是，注入的流量是正常的业务流，它也将随着拥塞窗口的增加而逐渐增加，当报文出现丢失时，拥塞窗口达到某个峰值。

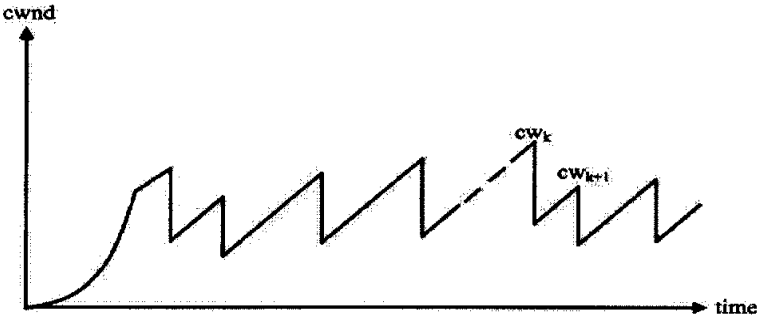

图 5-2 拥塞窗口随时间变化

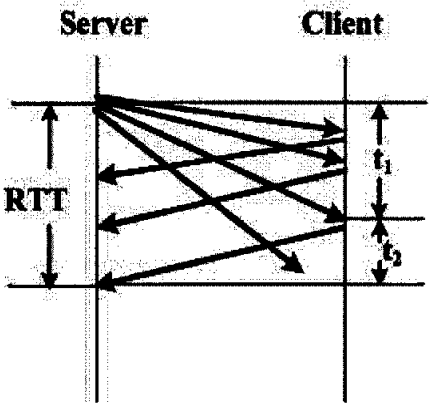

图5-3 拥塞窗口报文延迟示意

报文的时延如图5-3所示，网络在时间t_1里顺利地把cw_k-k个报文的数据传送

到客户端，其中k为丢失报文数。准确的可用带宽估算可由下式表达：

$$BW_m = \frac{(CW_m - k) * MSS}{t_{1m}} \tag{5-1}$$

没有客户端的配合，一起发送出去的几个报文中的最后一个在何时到达客户端是无法知道的。即t_1的准确值仅在服务器端测量是难于获取的。为方便实现，粗测量不考虑t_1值的估计问题。

假设服务器发出的报文与客户端发回来的响应走相同的路径（该假设是必须的，虽然这常常与实际不符），由于服务器发送给客户端的通常是满载荷的信息报文，而响应报文则很多时候是较小的报文，因此有：

$$t_1 \geq \frac{RTT}{2} \geq t_2 \tag{5-2}$$

又因为RTT=t_1＋t_2，可得：

$$RTT > t_1 \geq \frac{RTT}{2} \tag{5-3}$$

结合5-1和5-3可得：

$$\frac{(CW_m - k) * MSS}{RTT_m} < BW_m = \frac{(CW_m - k) * MSS}{t_{1m}} \leq \frac{2 * (CW_m - k) * MSS}{RTT_m} \tag{5-4}$$

利用公式5-4可以确定可用带宽的上限和下限，即可用带宽的范围。

下面我们来确定k的值。TCP连接的拥塞控制过程分成“慢启动”和拥塞避免阶段。当第一次出现报文丢失的时候，将设置“慢启动阈值”ssthresh为报文丢失前的拥塞窗口的一半，此后，报文将在此基础上进入拥塞避免阶段。因此，第一次报文丢失之后，到其再次发生报文丢失期间，通常都处于拥塞避免阶段。此时，一个RTT的时间里，拥塞窗口的大小只会增加1（线性增长）。而在前一个RTT中，即cwnd的时候并不出现丢包报文丢失。而当拥塞窗口增加1以后，即cwnd＋1的时候出现报文丢失。由于我们的测量对象是具有大量数据交流的连接，因此认为其总是有数据传输而不会出现长时间的空闲状态。在这样的情况下，从概率上来讲，k最可能的取值应该为1，但当cw_k表示的接收方的通过窗口时，并不总是丢失一个报文。为了处理方便，我们近似地认为k＝1（也可以这样认为：当丢包发生的时候，至少丢失了一个报文，在无法获取准确的丢失报文数量的时候，认为其只丢失了一个）。于是有：

$$\frac{(CW_m-1)*MSS}{RTT_m}<BW_m\le\frac{2*(CW_m-1)*MSS}{RTT_m} \tag{5-5}$$

令$D_m=(cw_m-1)*MSS$，利用平均值表示的可用带宽的活跃区间可用下式表示：

$$\frac{\overline{D}}{\overline{RTT}}<\overline{BW}\le\frac{2*\overline{D}}{\overline{RTT}} \tag{5-6}$$

不考虑来回路径不同的问题，t_1和t_2都在一定程度反映出该条链路的可用带宽状况，并且它们的大小也决定了它们对应的测量值在整个测量过程中的权重。由于RTT由t_1和t_2构成，可以用RTT值来决定其对应的测量值对某段时间内可用带宽的平均值的权重。因此D的加权平均值为：

$$\overline{D}=\frac{\sum_c D_m*RTT_m}{\sum_c RTT_m} \tag{5-7}$$

其中c为样本数量，RTT_m为第m个测得的来回时间的样本值。对RTT按次数求其平均值：

$$\overline{RTT}=\frac{\sum_c RTT_m}{c} \tag{5-8}$$

于是5-6可以表示为：

$$\frac{c*(\sum_c D_m*RTT_m)}{(\sum_c RTT_m)^2}<\overline{BW}\le\frac{2c*(\sum_c D_m*RTT_m)}{(\sum_c RTT_m)^2} \tag{5-9}$$

因此，最终测得的可用带宽的活跃区间为：

$$(\frac{c*(\sum_c D_m*RTT_m)}{(\sum_c RTT_m)^2},\ \frac{2c*(\sum_c D_m*RTT_m)}{(\sum_c RTT_m)^2}]。$$

服务器前端可用带宽粗测量算法对拥塞窗口算法主要作了两个方面的修改：

1)、修正了在直接使用拥塞窗口极大值进行测量时，丢失报文产生的影响；

2)、采用了可用带宽的活跃区间作为被测链路的端到端可用带宽的状态参考。利用该算法可以得到可用带宽的某个确定范围。

5.3 利用拥塞窗口算法精确测量可用带宽的算法讨论

有时候，可用带宽的粗测量所提供的可用带宽的活跃区间不足于判断网络的状态，这时候就需要对可用带宽进行精确测量以获得更准确的可用带宽估计值。本节讨论了在粗测量的基础上，实现可用带宽精确测量的算法。它与粗测量的最大区别就是采用了一种估计方法对单向延迟进行估计。

图5-3描述了RTT的组成，要实现对可用带宽的精确测量，关键在于精确地获得t_1的值。从另一个角度来看，时延包括三个组成部分[1][40][41]：排队时延——报文到达路由器时，由于路由器正忙于处理其他报文而需要排队等待的时间；传输时延——包含路由器对IP报文的处理时延和路由器将报文由输入缓存全部转发到输出端口的转发时延，前者为常数，后者可通过报文长度与路由器容量的商来计算；传播时延——报文在介质上的传送时间，等于实际距离与光速之比。

为方便讨论，我们做了一些假设：1)、不考虑来回路径不同的问题，即认为来回所经过的路径是同一路径；2)、由于传播时延通常很小，且实际距离不可测，不考虑传播时延。3)、各个路由器的资源使用状态在较短的时间内不发生突变，即认为某特定连接相邻的两个报文经过该路由器时的排队时延是相同的。4)、不考虑RTT中包含的接收方响应造成的时延，即认为接收方对数据的确认响应不被延迟。

根据数据报文在该链路上各个路由器的排队状况，可分为三种情况：无排队，轻微排队，严重排队。下面分别讨论各种情况下的可用带宽测量。

1．路由器上没有排队的情况

在没有交叉流，或者有交叉流但负载较轻的时候，即在各个途经的路由器上均不需要排队，没有排队时延。RTT可以用下式表示：

$$RTT_j = \sum_n \frac{L_{pj}}{\mu_i} + \sum_n \frac{L_{ackj}}{\mu_i} + 2T_{pro} \tag{5-10}$$

其中L_p是服务器发送出去的报文长度，L_{ack}是对应的响应报文长度，μ_i表示第i跳路由器的容量，n为总跳数，则$\sum_n \frac{L_{pj}}{\mu_i}$表示服务器发送报文的转发时延的总和，$\sum_n \frac{L_{ackj}}{\mu_i}$则为对应的响应报文的转发时延总和，$T_{pro}$为单向处理时延的总和。

取两个相邻的RTT做减法运算，由于处理时延为常数，可得：

$$\Delta RTT = RTT_i - RTT_{j-1} = \sum_n \frac{1}{\mu_i}(L_{pj} + L_{ackj} - L_{pj-1} - L_{ackj-1}) \qquad (5\text{-}11)$$

把5-11变换一下，并令 $K = \sum_n \frac{1}{\mu_i}$，可得：

$$K = \sum_n \frac{1}{\mu_i} = \frac{\Delta RTT}{(L_{pj} + L_{ackj} - L_{pj-1} - L_{ackj-1})} \qquad (5\text{-}12)$$

由于 $\sum_n \frac{1}{\mu_i}$ 是各个路由器容量的倒数之和，它只与网络的链路组成有关，在拥塞窗口算法当中，对被测连接的所有报文，均认为其在同一条路径中，理想情况下K应为常数。因此，所取的两个相邻的RTT值可以在连接中任意时刻选取，但必须注意：

1)、分母不能为零，即必须满足 $L_{pj} + L_{ackj} - L_{pj-1} - L_{ackj-1} \neq 0$。

2)、K应为正数，如果测得的K值为负值，认为该测量出现意外，将其放弃重新选取数据进行测量。

由式5-10可得：

$$T_{pro} = \frac{RTT_j}{2} - \sum_n \frac{1}{\mu_i}(L_{pj} + L_{ackj}) = \frac{RTT_j}{2} - K*(L_{pj} + L_{ackj}) \qquad (5\text{-}13)$$

此时，t_1可以表示为：

$$t_{1m} = \sum_n \frac{L_{pj}}{\mu_i} + T_{pro} = \frac{RTT_j}{2} - K*L_{ackj} \qquad (5\text{-}14)$$

结合式5-1，并令k=1，可得可用带宽为：

$$BW_m = \frac{(cw_m - 1)*MSS}{t_{1m}} = \frac{2*(cw_m - 1)*MSS}{RTT_j - 2K*L_{ackm}} \qquad (5\text{-}15)$$

其中 $K = \frac{RTT_j - RTT_{j-1}}{(L_{pj} + L_{ackj} - L_{pj-1} - L_{ackj-1})}$，MSS为最大数据单元，$cw_m$为第m个测量样本的拥塞窗口值与通告窗口的较小值。

2. 路由器繁忙，其排队队列较长，排队时延远大于转发和处理时延的情况。

在有交叉流且负载较重的情况下，传输的报文及其响应报文在路由器上都可

能出现严重排队。其RTT可以用下式表示：

$$RTT_j = \sum_n \frac{L_{pj}}{\mu_i} + \sum_n \frac{L_{ackj}}{\mu_i} + T_{Pj} + T_{ackj} + 2T_{pro} \tag{5-16}$$

其中T_{pj}和T_{ackj}分别表示业务报文和相应的响应报文的排队时延总和，T_{pro}表示单向的处理时延的总和。排队时延主要与网络的拥塞情况相关，而与报文的长度无关，可以近似地认为服务器发送出去的业务报文与其对应的响应报文的排队时延相等，即：$T_{Pj} = T_{ackj}$。

由于此时处理和转发时延与排队时延相比要小得多，对RTT的主要贡献来自于排队时延。可近似的认为$\sum_n \frac{L_{pj}}{\mu_i} + T_{Pj} + T_{pro} = \sum_n \frac{L_{ackj}}{\mu_i} + T_{ackj} + T_{pro}$。

近似处理后的t_1为：

$$t_{1j} = \sum_n \frac{L_{pj}}{\mu_i} + T_{Pj} + T_{pro} \approx \frac{RTT_j}{2} \tag{5-17}$$

结合式5-1，并令k=1，可得可用带宽的估计值为：

$$BW_m = \frac{(CW_m - 1) * MSS}{t_{1m}} = \frac{2*(CW_m - 1) * MSS}{RTT_m} \tag{5-18}$$

3．路由器上有轻微排队，排队时延与转发传播时延相当的情况

除了上面两种情况外，还存在一种中间情况，即路由器上有轻微排队，排队时延与转发、处理时延对RTT的贡献相差不大的情况。我们仍然可以使用式5-16来表示RTT，此时不能忽略转发和排队时延，但仍然可以认为业务报文的排队时延跟它对应的响应报文的排队时延近似相等。取相邻两个RTT的值做减法运算可得：

$$RTT_j - RTT_{j-1} = \sum_n \frac{1}{\mu_i}(L_{pj} + L_{ackj} - L_{pj-1} - L_{ackj-1}) + \Delta T_{pj} + \Delta T_{ackj} \tag{5-19}$$

其中：

$$\Delta T_{pj} = T_{pj} - T_{pj-1}, \qquad \Delta T_{ackj} = T_{ackj} - T_{ackj-1}$$

由于同一连接相邻的两个报文在各个路由器上的排队时延非常接近，近似地认为它们相等，即$\Delta T_{pj} = \Delta T_{ackj} = 0$，并令$M = \sum_n \frac{1}{\mu_i}$，则有：

$$M=\sum_{n}\frac{1}{\mu_i}=\frac{RTT_j-RTT_{j-1}}{(L_{pj}+L_{ackj}-L_{pj-1}-L_{ackj-1})} \tag{5-20}$$

同K一样，M也表示了各个路由器容量的倒数之和，只与组成链路的各个路由器，在连接一直使用同一组链路假设下是个常量，可以在连接的任意时刻选取一对相邻的RTT进行测量，也需注意两点：分母不能为0；M应为正数。

报文的排队时延与它对应的响应报文的排队时延近似相等，于是有：

$$T_{Pj}=T_{ackj}=\frac{RTT_j}{2}-\sum_{n}\frac{1}{2\mu_i}(L_{pj}+L_{ackj})-T_{pro}$$

$$=\frac{RTT_j}{2}-M*\frac{(L_{pj}+L_{ackj})}{2}-T_{pro} \tag{5-21}$$

而t_1是由服务器向客户发送的报文的转发、处理时延与排队时延的总和，因此有：

$$t_{1j}=T_{pj}+\sum_{n}\frac{L_{pj}}{\mu_i}+T_{pro}=\frac{RTT_j}{2}-M*\frac{(L_{pj}+L_{ackj})}{2}+M*L_{pj} \tag{5-22}$$

合并后面两项得：

$$t_{1j}=\frac{RTT_j}{2}+M*\frac{(L_{pj}-L_{ackj})}{2} \tag{5-23}$$

结合式5-1，并令k=1，可得可用带宽为：

$$BW_j=\frac{(CW_j-1)*MSS}{t_{1j}}=\frac{2*(CW_j-1)*MSS}{RTT_j+M*(L_{pj}-L_{ackj})} \tag{5-24}$$

综合式5-15、5-18、5-24可得，注意到K和M的表达式相同，均采用K来表示。拥塞窗口方法对可用带宽进行精确测量的算法可用下式表示：

$$BW_j=\begin{cases}\dfrac{2*(CW_m-1)*MSS}{RTT_j-2K*L_{ackm}} & \text{无排队}\\ \dfrac{2*(CW_j-k)*MSS}{RTT_j+K*(L_{pj}-L_{ackj})} & \text{轻微排队}\\ \dfrac{2*(CW_j-k)*MSS}{RTT_j} & \text{严重排队}\end{cases} \tag{5-25}$$

其中：

L_{pj} —— 第j个测量样本的报文长度，以字节为单位；

$$K=\frac{RTT_j-RTT_{j-1}}{(L_{pj}+L_{ackj}-L_{pj-1}-L_{ackj-1})};$$

L_{ackj} —— 第j个测量样本的响应报文长度；

cw_j —— 第j个测量样本的发送报文数（拥塞窗口值与通告窗口的较小者），以报文为单位；

MSS—— 最大数据单元，以太网为1500；

RTT_j—— 第j个测量样本的来回时间；

k —— 测量样本采集时丢失的报文数量，除第一次外通常为1；

对多个测量样本进行统计，以获得测量的平均效果。在精确的可用带宽测量中，仍然采用加权平均的方法对可用带宽进行统计。各个样本的权重仍然用RTT来表示，如下式：

$$\overline{BW}=\frac{\sum_c(BW_k*RTT_k)}{\sum_c RTT_k} \tag{5-26}$$

利用式5-25和5-26进行可用带宽的精确测量需要注意以下问题：

1. 无排队、轻微排队以及严重排队的状态划分标准问题

报文在路由器上是否需要排队，排队状况如何，只有路由器本身知道，在服务器前端难以测量。一种可行的估计方法是早期测量获得可用带宽的变化范围，并依据其变化范围的最小值或者最大值设定一个或两个阈值。当可用带宽的范围的最大值小于某个阈值的时候，认为网络负载较重，采用严重排队的算法测定可用带宽的准确值；而当可用带宽范围的最小值大于某个阈值（两个阈值不相同）的时候，认为网络负载较轻，报文在路由器上基本上不需要排队，采用无排队的算法对可用带宽的准确值进行测量。两个阈值之间的部分，则认为是轻微排队，采用轻微排队的算法进行测量。这种方法的主要不足在于被测量的链路多种多样，针对不同的链路，应该如何设定阈值才能保证其测量的准确性。如果能够知道链路的容量，按照一定的比例来确定阈值不失为一种可行的操作办法。

其实无排队和严重排队在算法讨论中是两个特殊情况，前者把排队时延忽略掉，后者把处理和转发时延忽略掉。如果无法获得合理的区分排队状况的方法，也可以直接使用式5-24（即5-25的中间式子）计算，即认为排队时延和处理、转发时延总是并存的。无排队的时候表现为计算的排队时延接近零，而严重排队的

情况表现为计算的转发和处理时延接近零。本文的初步实现就是采用这种处理方法。

2. 常数K的非负性

从定义上看，常数K为各个路由器容量的倒数之和。由于路由器容量一定是正数，因此，K也必然为正数。因此当测得的K出现负值的时候，说明出现了一些意料之外的影响因素，如网络拥塞状况的突变等，则此次测得的K值无效，必须重新对其进行测量，直到测得的值为正数才作为有效测量值。为保证K值的有效性，也可以对K值进行多次测量来获得一个平均值。为简单起见，本文的初步实现对每个连接只进行一次K值测量，在获得第一个K的有效测量值后就停止K值测量。

3. 不适用情况

本算法建立在一些假设的基础上，因此当假设条件不成立的时候，该算法失效，下列情况下，算法不适用。

1)、来回路径的差异不能忽略的情况；

2)、传播时延很大的卫星链路组成的网络路径，传播时延不能忽略的情况；

3)、网络的拥塞状况的变化总是以突变形式出现，且频繁出现的情况；

5.4 小结

本章描述了服务器前端基于被动的可用带宽测量的粗测量和精确测量算法。粗测量部分对拥塞窗口方法的改进主要是修正了丢失报文以及报文丢失时通告窗口小于拥塞窗口所带来的误差，以达到在精度要求不高的时候快速测量的目的。精确测量在此基础上进一步提出了单向时延的估计办法，以提高其精度。由于精确测量牵涉K值的测量，并在对实际的样本处理的时候也牵涉到形成RTT的一对报文的长度的提取，其计算过程也较为复杂。因此，在实际的实现过程中，我们先实现粗测量，在粗测量实现以后，再实现精确测量。

第六章 服务器前端基于被动的可用带宽测量实现

本章使用c/c++语言对服务器前端基于被动的可用带宽测量算法进行软件实现。在tcptrace的基础上进行核心功能扩展，使tcptrace能够根据所采集的流量轨迹测定各个连接可用带宽的活跃区间，并在此基础上初步实现了可用带宽的精确测量算法。

6.1 总体设计

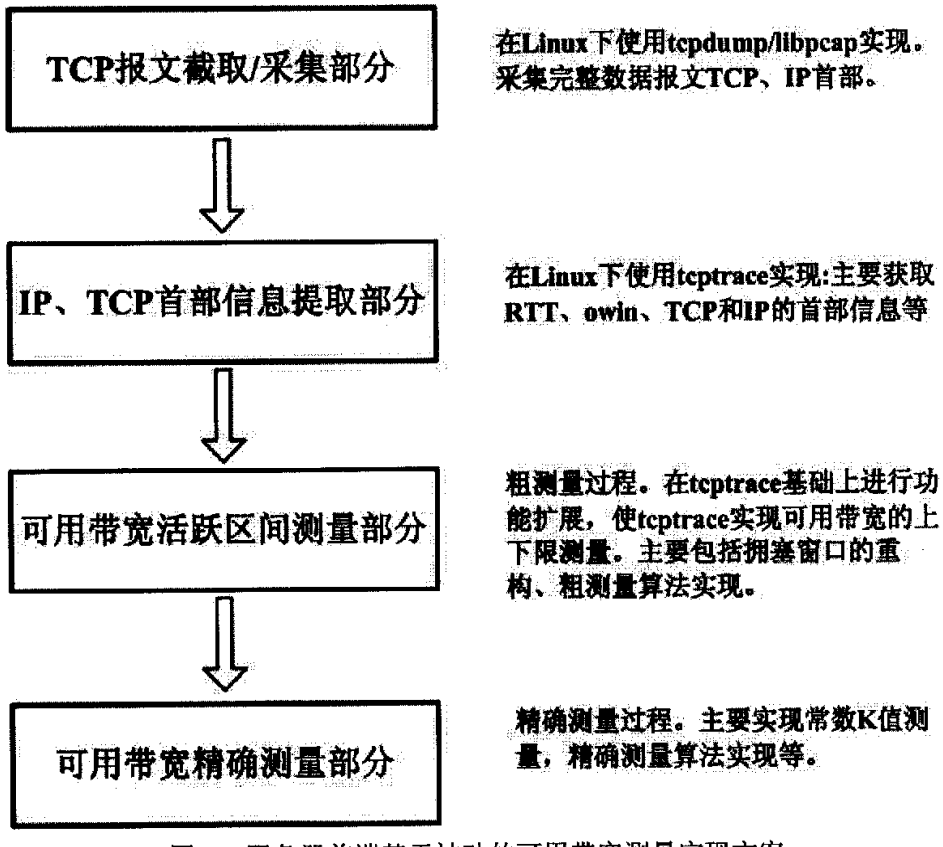

图6-1 服务器前端基于被动的可用带宽测量实现方案

从功能上可以将实现方案分成四个组成，如图6-1：TCP报文的采集部分、TCP/IP首部信息提取部分、可用带宽活跃区间测量（粗测量）和可用带宽精确测量。前两个部分是实现第三部分的前提，而且也由一些已有的工具可以直接或者

间接获得。为保证测量算法实现的可靠，这里采用tcpdump/libpcap来实现数据采集，使用tcptrace来实现对报文TCP、IP首部信息的提取。

可用带宽粗测量算法实现，主要是在tcptrace的基础上添加代码实现拥塞窗口的重构、活跃区间上下限测量算法的实现等。

精确测量在粗测量实现的基础上实现常数K值的测量、并按照轻微排队的算法实现精确测量，它也是基于tcptrace的功能扩展。

6.2 基于 tcptrace 的功能扩展——可用带宽粗测量实现

粗测量从功能上实现上限和下限的测量，从实现的模块上可分为拥塞窗口算法模拟重构、RTT提取、测量算法实现三个部分。其中RTT的提取是tcptrace已经实现的功能,程序将直接使用其结果，本文将不做更多的描述。

6.2.1 拥塞窗口算法的模拟重现

由于RTT可以直接从tcptrace获得，拥塞窗口算法的模拟重构成为可用带宽粗测量实现的关键部分。

不同的TCP实现有不同的拥塞窗口算法，如TCP Reno和TCP Vegas就不一样。这里首先实现了目前使用最广泛的TCP Reno版本（其他版本在后续的工作中补足）。TCP Reno的拥塞窗口更新算法主要包含初始化、非重传窗口更新、快速重传和快速恢复三个部分。

在tcptrace中，设计了一个数据结构**tcb**来存贮从采集下来的数据中提取的信息和必要的中间结果。利用其原有结构，为实现TCP拥塞窗口算法的模拟重构，添加变量如下（其原有变量设计部分省略）：

```
typedef struct tcb {
    u_long      scwnd;              /*当前包的拥塞窗口*/
    u_long      s_ssthresh;         /* 慢启动阈值*/
    seqnum      s_lastack;          /*前一个ACK序号*/
    Bool        s_dup_ack;          /*是否为响应副本*/
    Bool        s_last_retrans;     /*前一个包是否为快速重传包。*/
    ……
```

} tcb;

报文丢失可能是因为超时，也可能是快速重传（收到三个相同的ACK副本），因为对这两种原因引起的拥塞窗口的更新方法不同，因此设计对响应副本的判断变量和对快速重传的判断变量。

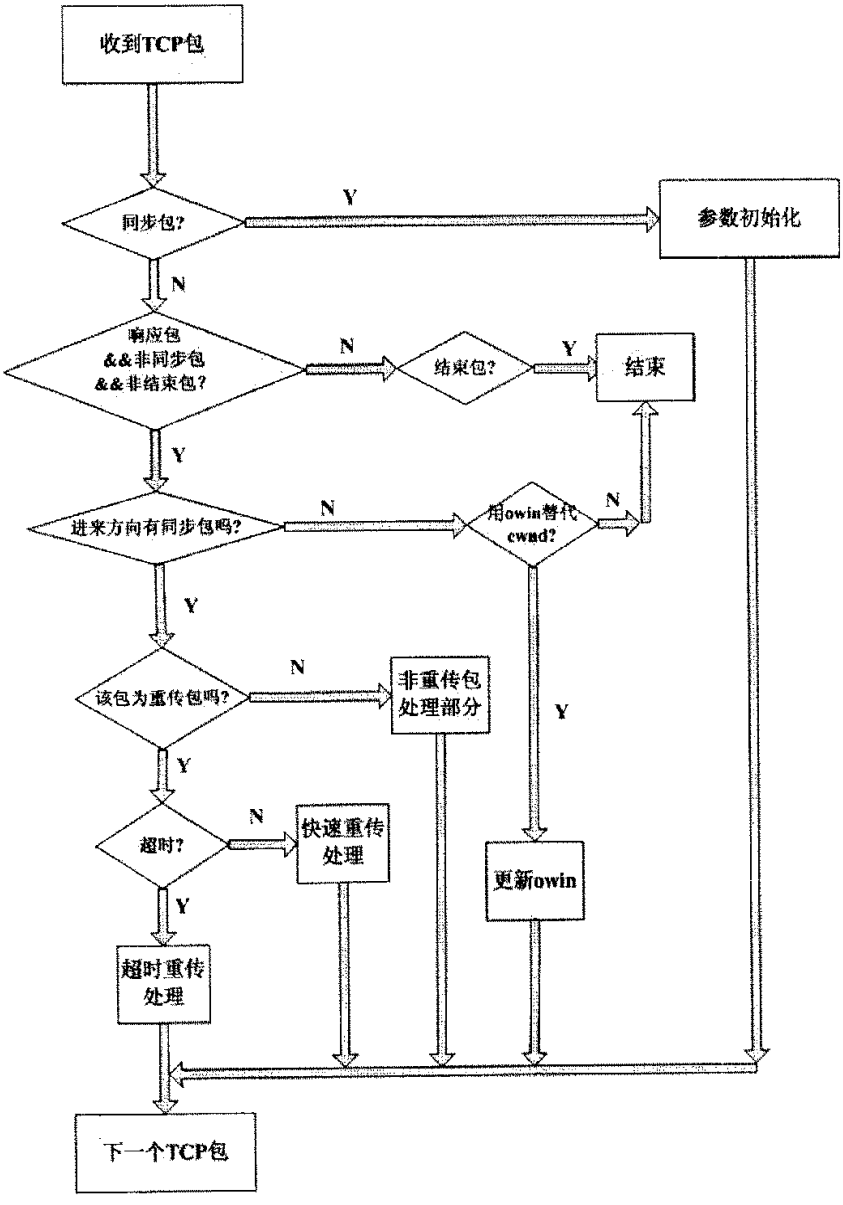

图6-2 TCP拥塞窗口模拟重现流程

实现的过程中需要注意的问题：

1. 首先对报文的类型进行判断。这一步工作主要根据TCP报文首部的各个

标志位来确定是同步报文还是响应报文等。不同的报文有不同的后续处理，如SYN同步报文的时候，对窗口值和“慢启动阈值”以及其他的变量进行初始化，只有接到的是响应报文才会对窗口值进行更新等；

2. 对重传的原因进行判断。超时（RTO定时器溢出）和快速重传（收到3个或者以上的响应报文副本）引起的重传对窗口的更新是不同的，根据不同的情况对拥塞窗口进行更新；

3. 特殊情况的处理。TCP连接在进行数据采集之前就已经完成了三次握手过程，从而造成对该连接进行采集的数据中不包含同步报文，这就是说对拥塞窗口的初始化不能正确完成，这将导致整个拥塞窗口估计算法失效，对此必须做特殊处理。 如使用服务器已经发出去且未得到响应的报文数量（owin）来代替cwnd或者对该连接的测量作放弃处理等。

因此，在拥塞窗口模拟重现的实现程序段中至少应当实现4个处理：参数初始化、非重传拥塞窗口更新、重传拥塞窗口更新、不完整连接处理。

其流程框图如图6-2所示。

1. 参数初始化

参数初始化是指对拥塞窗口cwnd和“慢启动”阈值ssthresh的初始化， TCP Reno中，cwnd和ssthresh的初始值分别设置为1个报文和65535字节。其实，有关统计的变量初始化也是在此时进行。其伪码如下：

```
if(同步报文) {
        s_cwnd=init_cwnd;            /*init_cwnd=1*MSS(1500)*/
        s_ssthresh=init_ssthresh;    /*init_ssthresh=65535 */
}
```

2. 非重传拥塞窗口更新

非重传报文将按“慢启动”方式或拥塞避免方式更新窗口。只有在该报文为非重传报文，且非同步报文和非结束报文的时候才进行该部分处理。其处理也相当简单，只需判断其是处于“慢启动”阶段还是处于拥塞避免阶段，然后根据不同的情况实现即可。其伪码如下：

```
if（非重传报文&&非同步&&非结束包） {
```

```
        if(慢启动阶段)    s_cwnd+=MSS;
        else      s_cwnd+=(MSS*MSS)/s_cwnd;
}
```

3. 重传时拥塞窗口更新

重传包意味着之前的发送速率已经高于可用带宽，使得网络发生拥塞。重传有能是因为超时，也可能是快速重传。对这两种情况，拥塞窗口的更新算法也是不同的。

两种处理的伪码分别如下：

```
if(超时重传)  {
        s_ssthresh=max(s_cwnd/2, 2*MSS);
        s_cwnd=1*MSS;
}
if（快速重传）{
        s_ssthresh=max(s_cwnd/2, 2*MSS);
        if(s_ssthresh==2)     s_cwnd=1*MSS;
        else          s_cwnd=s_cwnd/2;
}
```

4. 不完整连接处理

不完整连接因为无法初始化拥塞窗口值而使得模拟重构失败，从而导致最终使用拥塞窗口估计可用带宽的算法失效。对此，有两种处理方案可供选择。一种是直接放弃测量该部分连接，该测量是在服务器前端进行的，可测的对象相当多，因此放弃其中的部分测量是可行的。另一种方案是替代方案。用拥塞前两个ACK之间的时间里发送方已经发出而未收到确认响应的数据字节数来代替拥塞窗口。虽然这样可以得到替代的数据，但是替代数据并不总是代表了可用带宽的值，尤其是网络有无线环境参与的时候。因此，我们对不是特别感兴趣的那些不完整的连接，放弃其测量，而对于特别感兴趣的某些链路的不完整连接，可以强制使用明显没有响应的数据量（tcptrace中用owin表示）代替拥塞窗口值来获得可用带宽参考值。而明显没得到响应的数据量提取也是tcptrace中实现的功能之一，可以直接利用。本文的实现上支持使用替代方案，但默认情况下作放弃处理。

6.2.2 拥塞窗口方法可用带宽粗测量算法实现

测量算法实现是指在拥塞窗口和RTT样本采集的基础上按照第五章所述的粗测量算法进行统计的过程。所需的变量也在结构tcb中添加代码实现。

```
typedef struct tcb {
......
    /*以下代码为统计粗测量结果而设计*/
    u_long      previous_scwnd_sample;    /*上一个ACK时拥塞窗口值*/
    double      previous_scwnd_time;      /*上一个ACK的RTT(10^-6s) */
    double      scwnd_sum;                /*拥塞窗口与RTT的乘积之和*/
    u_long      scwnd_avg;                /*采样窗口的按次数平均值*/
    double      scwnd_timesum;            /*测量样本的时间之和*/
    double      scwnd_wavg;               /*具有时间权重的加权平均值*/
                                          /*scwnd_sum/scwnd_timesum*/
    u_long      scwnd_tot;                /*采样窗口值的总和*/
    u_long      scwnd_count;              /*采样窗口数目计数*/
    double      rtt_scwnd_avg;            /*rtt样本平均值统计*/
    double      maxabw;                   /*可用带宽上限值*/
    double      minabw;                   /*可用带宽下限值*/
......
} tcb;
```

采用拥塞窗口方法对可用带宽进行测量的算法在重传发生时的前一个数据作为测量样本，由于不能预知丢包，测量样本的采样总是在重传发生以后。因此设计了前一个ACK的拥塞窗口值和来回时间来暂存丢包前一个ACK的拥塞窗口值和RTT。

可用带宽测量算法的实现需要注意以下几个问题：

1. 是否进行替代处理？添加命令行参数，让用户在运行的时候指定是否作替代处理。如果指定运行，将在没有SYN报文的时候使用owin来代替cwnd。需要owin替代cwnd的时候将直接取用tcptrace本身所获得的owin。由于我们的算法最根本的思想是在一定的时间里能够传输的数据量就是可用带宽，因

此，此时的RTT应为owin的保持时间，即两个对新数据进行响应的ACK报文之间的时间间隔。在拥塞窗口中，响应副本的影响已经体现在拥塞窗口中。

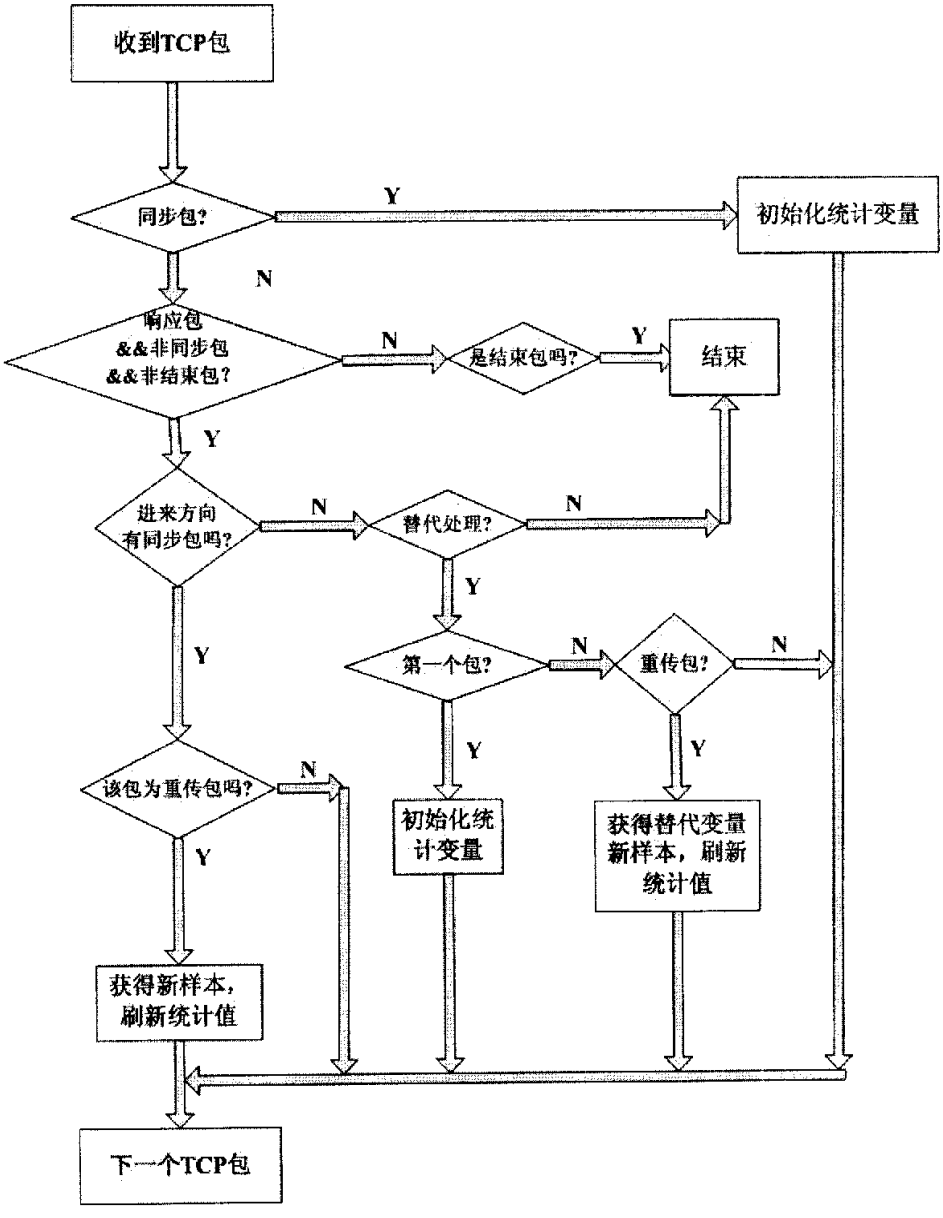

图6-3 粗测量算法实现流程框图

2. 参数的初始化问题。参数的初始化同样是在接到同步报文时进行，如果该连接不包含同步报文，且强制使用明显未响应的数据量（owin）来替代拥塞窗口继续进行测量，初始化的过程将在该连接采集到的第一个包时进行。

3. 重传报文处理。重传报文是测量样本采集的标志，但在没有SYN报文的

情况下使用owin代替cwnd的时候，第一个包可能会是个重传包，由于测量需要的是重传报文前一个报文的数据，在无法获得第一个包之前的数据的时候，对第一个包的重传视而不见是最好的办法。从第二个报文开始，在检测到重传的时候，对其前一个数据进行采样并计算到统计值中。

4. 样本值的保存问题。设置一个数组结构，用于保存采集到的样本值，该步骤是为精确测量准备的，将在精确测量中进行定义并给出相应的处理说明。

可用带宽粗测量算法实现流程框图如图6-3所示。

6.3 基于 tcptrace 的功能扩展——可用带宽精确测量实现

可用带宽的精确测量算法视网络的排队情况不同而不同。其主要内容包括两个方面：一是对网络排队状况的估计，初步设计不考虑该问题，直接采用轻微排队的算法，即认为排队时延和转发、处理时延均不被忽略的情况；二是K值的测定问题，K值是个只与链路各个路由器的容量相关的系统常数，其准确性将影响可用带宽测量的准确性。

可用带宽的精确测量需要K值，如果在获得有效的K值之前，已经出现报文丢失并完成部分样本的采集，就需要先把这些样本保存起来，在测得有效K值之后进行精确测量分析。设计数据结构cwnd_rtt对样本进行保存：

```
typedef    struct    cwnd_rtt    {
    timeval      sample_time;        /*该样本采集的时间*/
    u_long       scwnd;              /*样本的cw_k*/
    double       rtt;                /*样本的RTT*/
    u_long       packetlen;          /*样本业务报文长度*/
    u_long       acklen;             /*样本对应的响应报文长度*/
}cwnd_rtt;
```

对K值的测量只需要相邻的两对报文长度，不另外设计数据结构，直接在原有结构tcb里添加代码：

```
typedef struct tcb {
    /*以下代码为统计精确测量结果而设计*/
```

```
/*K值测量部分*/
short       packetlen_prev;      /*被前一报文响应的报文的长度*/
short       acklen_prev;         /*前一报文的长度*/
short       packetlen;           /*被当前报文响应的报文的长度*/
short       acklen;              /*当前报文的长度*/
double      cwnd_k;              /*当前测得的K值*/
double      cwnd_k_avg           /*K值的平均值*/
cwnd_rtt    sample[n];           /*采样结果暂存*/
/*可用带宽值测量统计部分*/
 double     abw;                    /*当前可用带宽测量值*/
 double     abw_sum;                /*拥塞窗口与RTT的乘积之和*/
 double     abw_timesum;            /*测量样本的时间之和*/
 double     abw_wavg;               /*具有时间权重的加权平均值*/
.....
} tcb;
```

由于采样结果只是在有效K值的获得之前才需要暂存，考虑资源问题，n值不宜设置过大，建议值为10。获得10个测量样本，即已经发生了同样次数的重传，正常情况下已经传输的报文数量将远超过10个，此时不能获得有效K值的可能性已经很小。也可以根据实际的情况设置得更大或更小，如果n值设置的足够大，也可以将每个样本的细节都存储出来。

本文实现的K值测量作了简化处理，对K值只进行一次测量。程序先对K值进行检查，如果K值为初始化时的零值，则进行K值测量，测得有效的K值后，保存直至该连接结束。如果之前已经获得K值，则直接进行当次测量分析。第一次获得有效的K值时，对每个已经测得的样本进行计算分析并统计，然后才对当次的测量样本进行分析统计。测量统计所需要的变量直接引用粗测量中的拥塞窗口值以及tcptrace本身测量的RTT值。如果不需要知道每个样本的细节，将不再对后面的样本进行保存。

可用带宽精确测量算法实现流程框图如图6-4所示。

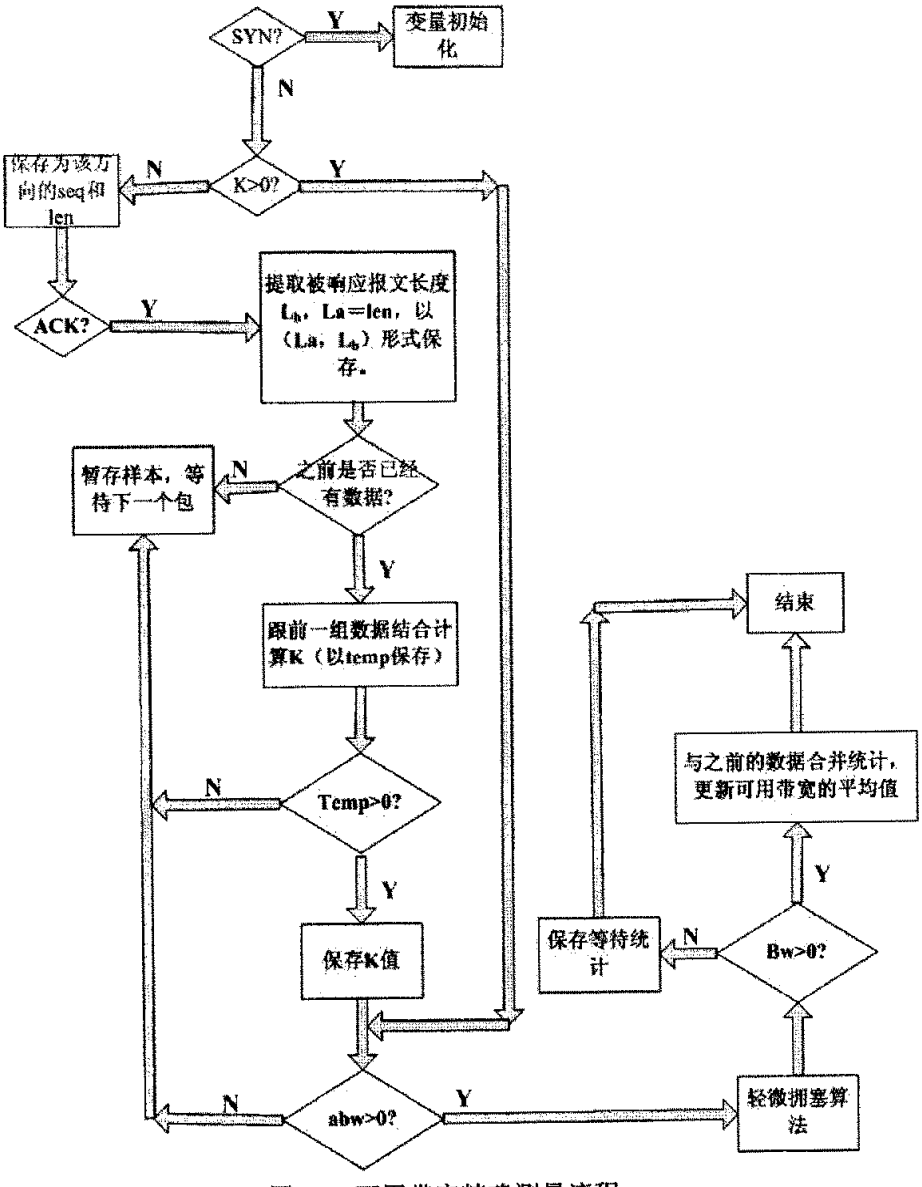

图6-4　可用带宽精确测量流程

6.4 测量结果的输出

tcptrace本身对各种功能的实现，除了文本格式的结果输出之外，还实现了部分统计量的绘图格式文件的生成，并可借助于xplot以图形形式输出结果。由于本文所讨论的内容是在软件实现的基础上对算法有效性的验证，作为进一步的硬件开发的基础，这里只实现了以文本的方式进行输出。

本文的软件实现方案设计了两种输出模式：详细输出模式和仅输出可用带宽测量值的模式。前者与tcptrace的详细输出选项联合(-l)使用，除了显示tcptrace原有的各种统计值外，添加输出拥塞窗口的平均值、加权平均值以及发送方发送周期的明显得到确认的数据量的平均值和加权平均值，可用带宽的下限、上限以及精确测量值等。

同时设计了仅输出可用带宽的选项（—J)。其输出的基本格式如下：

cip：cport　sip：sport　minABW　maxABW　ABW

其中cip、cport 、sip、sport分别指客户端IP、端口、服务器端IP和端口。通常情况下，在一个较短的测量时间内，不会出现不属于同一个连接的这四个数据的相同组合，因此具有唯一性。可见，这两个字段（以空格区分字段）指定了测量的某个特定连接。maxABW和minABW分别表示可用带宽的上限值和下限值，ABW表示精确测量值（如果需要的话)。

在程序调试的过程中，为调试方便，输出更详细的内容将有助与程序的调试，因此，设计了调试的输出模式：

cip:cport　sip:sport　count　cwnd　rtt　maxBW　minBW　K　ABW

添加的输出参考值count、cwnd、rtt、K依次对应样本数量、拥塞窗口样本的加权平均值、RTT样本的加权平均值和常数K。而当连接的SYN报文未被采集且需要继续测量的情况下，cwnd和rtt则表示相应的owin和保持时间。

第七章 实验与结果分析

本章将对服务器前端可用带宽粗测量和精确测量的实验测试结果进行分析和讨论。

7.1 服务器前端可用带宽测量实验

在一个三跳的实验网络中，将各个路由器速率设置为 10Mb/s（1Byte=8bit，1250KB/s），由于重放流量的存在，将消耗掉 R3 和 R4 的部分带宽，根据重放流速率以及路由器设置容量可以获得业务流量可用带宽的理论值。在服务器附近搭建监控器，采用 tcpdump 对业务流量进行采集，然后使用基于被动的测量方法对多种背景流速率环境下的网络进行粗测量和精确测量。对每个背景流速率均进行多次测量（本文为 10 次）以获得各项测量值的平均值。实验拓扑图如图 7-1 所示。

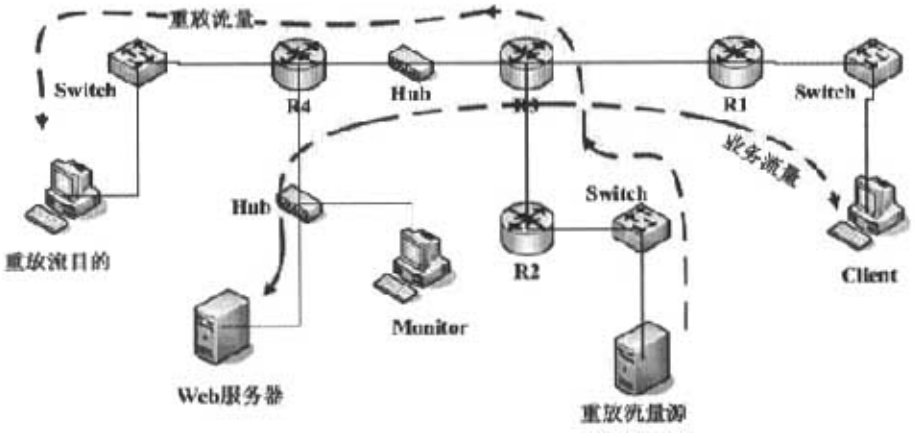

图 7-1 服务器前端可用带宽被动测量实验拓扑

7.2 实验结果

测量结果如表 7-1。由于重放流量在接近网络容量的时候，常常不能达到设

置的特定重放速率值，因此表中出现的 7.8Mb/s 和 8.4Mb/s 的情况，并使得实验无法达到更高的背景流速率。吞吐量在客户端获得，指的是业务流量在连接持续时间内的平均吞吐量。粗测量的上、下限是使用第五章中所述的粗测量算法获得的测量数据，而精确测量则使用引入了 K 值和报文长度的精确测量算法。由于算法不同，因此，精确测量值有可能在活跃区间之外。测量标准差是指各个精确测量值相对精确测量平均值的标准差，偏差按下式计算：

$$偏差=\frac{理论值-精确测量值}{理论值}*100\%$$

表 7-1 各种背景流速率下的测量结果

背景流 (Mb/s)	平均样本数	吞吐量 (KB/s)	粗测量下限(KB/s)	粗测量上限(KB/s)	精确测量 (KB/s)	理论值 (KB/s)	偏差 (%)	测量标准差
2	0	460.66	—	—	—	1000	—	—
4	6.75	248.55	510.84	1021.69	392.36	750	47.69	101.31
6	34.8	92.5	242.42	484.83	206.74	500	58.65	83.27
7	96.9	64.69	249.73	519.26	302.70	375	19.28	53.27
7.8	135.89	55.49	202.11	404.22	217.70	275	20.84	57.40
8	181.7	37.71	157.58	315.16	201.15	250	19.54	87.90
8.4	275.4	41.93	131.71	263.42	159.21	200	20.40	67.65

由于服务器前端基于被动的可用带宽测量的实现以业务报文的丢失为采样条件，在可用带宽速率较高的时候，业务报文可能不出现报文丢失，或者只出现极少数的报文丢失，致使测量的失效或者产生较大的偏差。表 7-1 中所示，背景流量为 2Mb/s 的时候，没有出现报文丢失；背景流为 4Mb/s 时，10 次测量中只有四次出现报文丢失（表中的各项数据只对 4 次有效测量进行统计）。在背景流量达到 6Mb/s 以上，每次测量均出现报文丢失，但其测量结果的偏差很大，我们认为这主要是由于算法在网络轻负载的情况下容易受其他因素影响而造成。在 7Mb/s－8.4 Mb/s 的背景流速率下，精确测量平均值的偏差保持在 20%附近，可以认为，在此背景流条件下该方法能够较好的实现对可用带宽的测量。

图 7-2 显示了各种背景流速率下的各个精确测量值的变化曲线。其中横轴 n 表示第 n 次测量的值，实线为各个测量值的连线，虚线为可用带宽的理论值而点线则为精确测量的平均值。图中 6 Mb/s 背景流下的可用带宽理论值与纵坐标的

上限重合，无法看到。由图 7-2 可以看到，虽然 7Mb/s－8.4 Mb/s 的背景流速率中，多次测量的平均值能够较好的体现可用带宽，但各个单次测量对可用带宽的理论值偏离较大。这在表 7-1 中表现为各次测量的标准方差较大。

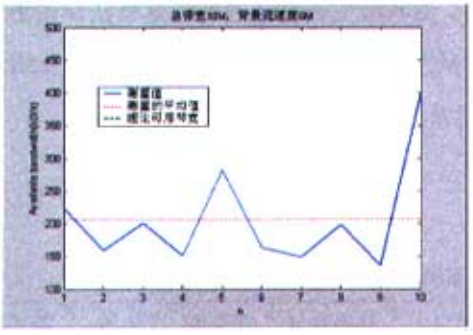

图 7-2a

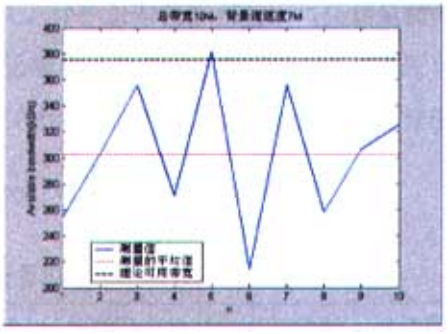

图 7-2b

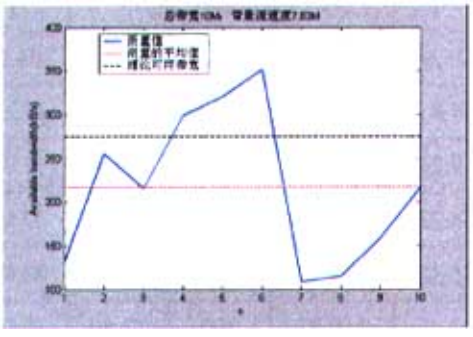

图 7-2c

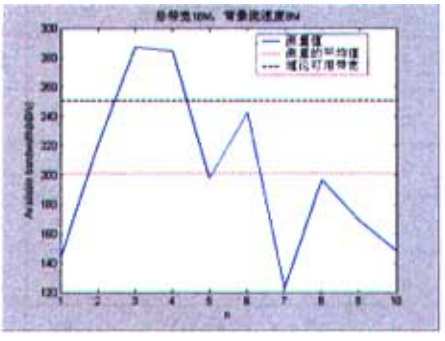

图 7-2d

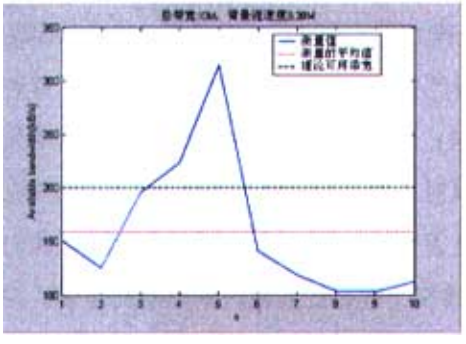

图 7-2f

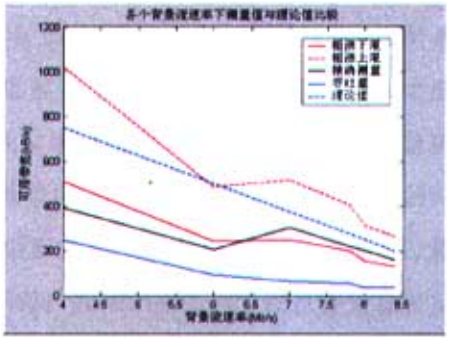

图 7-3

图 7-3 是可用带宽粗测量、精确测量与可用带宽的理论值、实际的吞吐量的比较。由图可以得到以下结论：

1、在 7M 以后的精确测量值具有较好的线性；

2、绝大多数时候，可用带宽的理论值在粗测量的活跃区间之内；

3、在背景流速率达到 7Mb/s 以上的时候，精确测量值也总是在可用带宽的活跃区间中，并更接近测量下限值。

可见，当对精度要求不高的时候，可利用活跃区间快速实现估计，并可将下限值作为参考值（由于粗测量下限算法与精确测量严重排队时的算法很相似，只是统计方法不同，由图中也可以看到在排队严重的时候，两者非常接近）。注意到实际的吞吐量总是比可用带宽的理论值和测量值小得多，这是由于通信双方在大多数时候是低于可用带宽的速率发送而造成的。

综合上面的实验结果，可以得出如下结论：

1. 服务器前端基于被动的可用带宽测量只适用于网络负载较重的情况下，在网络负载较轻的时候，由于测量中受其他因素影响（如实际发送的报文数与拥塞窗口和通告窗口的较小者不符等），会带来较大的误差，甚至无法实现测量；
2. 在负载较重的情况下，服务器前端基于被动的可用带宽测量基本能够实现对可用带宽的测量；且精确测量与活跃区间的测量结果在合理的范围内。鉴于目前对可用带宽测量的水平[42]，20%的偏差已经达到了相当不错的测量效果；
3. 精确测量中虽然多次测量的平均值较好地体现了可用带宽的理论值，但各个单次测量的结果与理论值比较存在较大的偏差。这表明测量中存在大量的随机误差，该误差在多次测量的统计效果中被消除。如何在单次测量中消除这些误差将是进一步提高其准确性需要解决的重点问题。

7.3 实验误差原因分析

从测量的结果来看，服务器前端基于被动的可用带宽测量仍然存在较大的误差。其影响因素主要有：

1. 丢包的数目问题。本文中认为每次发生丢包的丢包个数均为一个。而在实际的网络当中，通常会存在其他的情况：如发生丢包的时候同时丢失多个报文，或者同一个报文连续丢失多次。这些在本文的算法当中均未做出

相应的处理。当发生丢失报文的个数不止一个的时候，因为算法的处理与实际情况不符，造成误差；

2. 背景流量并不是一致保持设定值，而在设定值上下波动。背景流从宏观上看是稳定的，但从微观上，它也存在波动，因此，在两次测量的时候存在一些差异，即使在同一次测量中，也存在背景流与业务流量的竞争，业务流密集和稀疏的时候，背景流也可能出现相应的波动。测量实验中，不曾考虑背景流量的波动幅度，也引进部分误差；
3. 基于被动的可用带宽测量算法要求通信双方的发送速率只受拥塞窗口或者通告窗口的较小者控制。但某些时候可能会存在其他的因素使得其发送速率并不完全按照两者的较小值进行发送，如服务器的发送缓存不足的影响等；如果服务器的发送速率受其他因素的影响，则拥塞窗口和通告窗口的较小者并不完全反映出实际丢包的时候实际发送的报文数量，将造成异常的测量值；
4. RTT 的测量偏差也引进部分误差。本文算法测量是建立在 tcpdump 采集数据的基础上。Tcpdump 是个软件实现的监听和采集工具，其时间戳带有一定的误差，在此基础上实现的 RTT 估计也必然带有误差。此外，数据接收方的延迟响应将造成根据 RTT 估计单向延迟的偏差。
5. 精确测量中只对 K 值进行一次测量也必将引起误差。该误差可以通过使用 K 值的平均值来消除。

影响网络的因素很多，除上述几点外，还存在着其他可能带来误差的因素，如传播时延、路由器本身的不稳定等。

结束语

本论文对服务器前端基于被动的可用带宽测量算法进行了研究和实现，它是在对拥塞窗口的可用带宽测量算法进行改进的基础上完成的，获得了如下成果：

1. 算法上对拥塞窗口方法的改进。服务器前端基于被动的可用带宽测量算法对拥塞窗口测量算法做了多方面的改进，修正了由丢失报文、RTT 带来的误差，以及采用拥塞窗口和通告窗口的较小者作为测量对象，更符合网络的实际运行情况；
2. 实现上，对 tcptrace 进行功能扩展，使它能够利用服务器前端基于被动测量算法实现端到端的可用带宽估计。既实现了 Monitor 中对拥塞窗口的模拟重构，也实现了对可用带宽活跃区间的测量，并进一步实现了对可用带宽的精确测量；
3. 对服务器前端基于被动的可用带宽测量算法进行了实验测试研究。

由于时间和实验条件的限制，本课题尚存在以下不足和有待改进的地方：

1. 每次的丢包数目简单的估计为 1 个报文。如果实际的丢包数目并不是 1 的话，给测量带来误差。要修正该误差，应在测量过程中检测实际的报文丢失数目；
2. 对每个报文的大小均以最大报文对待，实际的业务报文并不一定总是满负荷报文。要实现更精确的测量，应该探测在发生报文丢失前背靠背发出的报文群的总字节数量；
3. 实际上背靠背发出的报文之间也存在一定的报文间延迟，该延迟可根据发送方的网络接口卡速率和报文的长度进行估计。本文的算法对此忽略处理；
4. 在实现上，目前只实现了对 Reno 版本的 TCP 拥塞窗口的估计算法，有待扩展。在精确测量上，简单的使用轻微排队的算法。实际上，由图 7-3 可以看到，排队严重的时候，轻微排队算法虽然也能测量出较准确的值，但其计算过程明显比排队严重的算法复杂。
5. 实验中，对 8.5Mb/s 以上的背景流速率的测量结果未进行测试，而且单

次测量的误差相当大。在实际测量的时候，需进行多次重复测量以获得平均效果。

进一步的研究工作集中在以下几个方面：

1. 进一步修正算法上的不足；
2. 在算法上实现对其他 TCP 版本的支持；
3. 进行更多的实验测试，以获得更准确的测量结果，并试图寻求有效的方法对测量中出现的随机误差进行修正；
4. 对程序进行代码优化，为下一步的基于 NP 的硬件开发做准备。

参考文献

[1]. 朱畅华，裴昌幸，李建东，金旗， 网络测量及其关键技术，西安电子科技大学学报(自然科学版)， Dec.2002 Vol. 29 No. 6 pp.813-818

[2]. Gerald D. Cole "Performance Measurements on the ARPA Computer Network" Communications, IEEE Transactions on [legacy, pre - 1988] Volume 20, Issue 3, Jun 1972 Page(s):630 - 636

[3]. Barry M. Leiner, Vinton G. Cerf, David D. Clark et al, A Brief History of the Internet. http://www.isoc.org/internet/history/cerf.shtml. 1999

[4]. Amer, P.D.; Cassel, L.N.. Management of sampled real-time network measurements. Local Computer Networks, 1989. Proceedings 14th Conference on 10-12 Oct. 1989 Page(s):62 - 68

[5]. 高传善，代春阳. 网络测量综述. 上海计量测试. 2004 年 03 期. pp.8-15.

[6]. 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 年 01 期. pp.110-116

[7]. S. Seshan, M. Stemm, R. H. Katz. SPAND: Shared Passive Network Performance Discovery. Symposium on Internet Technologies and Systems. 1997

[8]. 吕军，李星. 网络测量分析及研究综述. 计算机工程与应用. 2003 年 24 期. pp.19-22.

[9]. Network Monitoring Tools. http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html. March 3, 2005.

[10]. Fping, http://www.fping.com/

[11]. Pingplotter, http://www.pingplotter.com/

[12]. Treno, http://www.psc.edu/~pscnoc/treno_info.html .

[13]. bing. http://spengler.econ.duke.edu/~ferizs/bing.txt.

[14]. Manish Jain, Constantinos Dovrolis. End-to-End Available Bandwidth: Measurement methodology, Dynamics, and Relation with TCP Throughput. Proceedings of ACM SIGCOMM 2002 conference., August 2003. pp.295-308

[15]. TcpDump http://www.tcpdump.org 2005

[16]. Tcptrace http://www.tcptrace.org 2005

[17]. 蒋序平, 陈鸣, 赵金. 网络测量系统研究中亟待解决的若干问题. 电信科学. 2003 年 8 期. pp.62-65.

[18]. Paxson V, Mahdavi J, Adams A, et al. An architecture for large-scale internet measurement. IEEE Communications Magazine, 1998, 36(8). pp.48-54.

[19]. McGregor A J, Braun H W, Brown J A. The NLANR network analysis infrastructure. IEEE Communications Magazine, 2000, 38(5). pp.122-128.

[20]. AMP. http://watt.nlanr.net/. update by July 09,2004.

[21]. Kalidindi S. Surveyor an infrastructure for internet performance measurements. IEEE Communications Magazine, 1998, 36(9). pp.48-54.

[22]. CAIDA. http://www.caida.org/

[23]. FUITEG. http://www.cs.fudan.edu.cn/ch/third_web/FUITEG/chn/index.html.

[24]. passive measurement, http://wand.cs.waikato.ac.nz/old/wand/publications/jamie_420/final/node9.html

[25]. 谭思亮. 监听与隐藏——网络侦听揭密与数据保护技术. 人民邮电出版社. 2002.8, pp.131

[26]. 谢希仁；计算机网络（第 2 版）；电子工业出版社；1999 年 2 月；

[27]. Jacobson.V. Congestion avoidance and control. In Proceedings of SIGCOMM '88 (Stanford，CA，Aug. 1988), ACM. pp.314-329

[28]. W. Richard Stevens. TCP/IP 详解，卷 1：协议. 机械工业出版社. 2000.4

[29]. Lawrence S. Brakmo and Sean W. O'Malley and Larry L. Peterson. TCP Vegas: New techniques for congestion detection and avoidance. in Proceedings of ACM SIGCOMM'94, October 1994. pp.24-35

[30]. Kenji KURATA , Go HASEGAWA , Masayuki MURATA. Fairness Comparisons Between TCP Reno and TCP Vegas for Future Deployment of TCP Vegas. INET 2000 Proceedings. 2000. pp.126.

[31]. Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. Computer Communication Review, vol 27, No.3, July 1997 pp.67-82

[32]. Jamshid Mahdavi, Sally Floyd. TCP-Friendly Unicast Rate-Based Flow Control. Note sent to the end2end-interest mailing list. http://www.psc.edu/networking/papers/tcp_friendly.html

1997

[33]. Luigi Alfredo Grieco, Saverio Mascolo. End-to-End Bandwidth Estimation for Congestion Control in Packet Networks. Proceeding of the Second International Workshop, QoS-IP 2003, Milano, Italy, February 2003, Lectures Notes on Computer Science 2601, pp.645-658

[34]. M. Gerla, M. Y. Sanadidi, R. Wang, A. Zanella, C. Casetti, S. Mascolo. TCP Westwood: Congestion Window Control Using Bandwidth Estimation. In Proceedings of IEEE Globecom 2001, Volume: 3, pp.1698-1702, San Antonio, Texas, USA, November 25-29, 2001

[35]. Buzios, Brazil. Part 2: Bandwidth Estimation Techniques in TCP Westwood. Presentations in SBRC 2002;

[36]. Mark Allman, Vern Paxson. On Estimating End-to-End Network Path Properties. ACM SIGCOMM '99. pp.263-274

[37]. WREN Bandwidth Monitoring Tool. http://www.cs.wm.edu/~mazang/wren.html

[38]. Bruce B. Lowekamp. Combining Active and Passive Network Measurements to Build Scalable Monitoring Systems on the Grid. Performance Evaluation Review 30(4):19-26, March 2003.

[39]. Bruce B. Lowekamp, Marcia Zangrilli. Using Passive Traces of Application Traffic in a Network Monitoring System. Proceedings of the Thirteenth IEEE International Symposium on High Performance Distributed Computing (HPDC 13), June 2004. pp.77-86

[40]. Meng-Fu Shih, Alfred Hero. Unicast inference of network link delay distributions from edge measurements. Technical report, Comm. and Sig. Proc. Lab. (CSPL), Dept. EECS, University of Michigan, Ann Arbor, May 2001.

[41]. Lai K, Baker M. Measuring Link Bandwidths Using a Deterministic Model of Packet Delay [DB/ OL]. Proceedings of ACM SIGCOMM. 2000. pp.283-294

[42]. Jacob Strauss, Dina Katabi, Frans Kaashoek. A Measurement Study of Available Bandwidth Estimation Tools. Internet Measurement Conference(IMC). Florida. 2003. pp.39-44

致 谢

论文写完的时候，也是我即将离开中大校园的时候。有很多话要说，这里只能借这个方寸之地表达一下我的谢意。

首先感谢我的导师余顺争教授。两年以来，从论文的选题，到课题研究的每个细节以及硕士论文的完成，余老师事必躬亲，不曾忽视培养计划的每一个环节，他的指导使我在治学态度、知识结构、研发能力以及实践能力上都有了很大进步。严师如父，余老师对我学业以外的指导，也将使我终身受益。在此致以深深的感谢和敬意。

何海涛老师两年以来在实践方面给予我大量指导和帮助，他主持搭建的实验网络为后期的测试提供了良好的实验平台，并在东校区网络搭建的时候给我提供了与课题密切相关的实习机会，在此表示感谢。

感谢何利在 linux 操作系统方面、实验工作和其他各方面为我提供无私的帮助，与何利同学的讨论总是可以让我得到启发。

感谢这些年来与我一同渡过美好时光的邓凡、蔡伟杰、黎敏、卢炎石、张伟、王思萌、钟志毅、冯宏达、林海涛、谢志韬以及其他的师兄师姐、师弟师妹们。你们的鼓励和关怀让我在遇到困难的时候不曾气馁。

我要特别感谢我的女朋友黄梅喜，我攻读硕士期间她在各方面给予我周到的照顾和关爱。本文的完成，与她付出的汗水和心血不能分开。

同时，我要将这些年来的所有成绩献给我的父母和家人。他们的支持和关怀是我奋斗的最大动力。

原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：2005年 6月 6 日