多路径传输协议缓存研究分析

周峰[1] 周星[1] 王琨[1] 廖宇力[1] 符发[1]

(1.海南大学信息科学技术学院，海南 海口，570228)

摘　要：在多路径传输协议 MPTCP 中，由于多条子流的存在，需要维持一个较大的缓存用于数据的整理和重组，缓存不足会导致程序的挂起。该论文首先介绍了 MPTCP 协议的技术原理，接着对 TCP 协议和 MPTCP 协议的缓存大小和吞吐量的关系进行研究分析，根据研究结果提出一个针对低延时、高带宽场景的缓存计算模型，用于计算 MPTCP 缓存大小。接下来设计一组本地实验和 NorNet Core 测试床实验进行分析，结果表明，根据模型所设计的缓存取值仅为当前 Linux 内核缓存默认值的一半，大大的减少了资源浪费的同时传输也达到了吞吐量的最大值。

关键词：　多路径传输协议；缓存分析；缓存模型

中图分类号：TP 3-05　　文献标志码：A

1 概述

随着互联网通信的快速发展，对支持多宿主设备的需求不断增长，如云计算、大数据就都需要大吞吐量的网络作为其基础支持。同时越来越多的终端设备拥有多种接入方式（如：IP 网络、4G 网络、WLAN，卫星通信）与 Internet 连接。如何满足人们对吞吐量日益增长的需求，是当今通信领域研究的重要课题之一。当前互联网中使用 TCP 单路径传输协议，尽管终端设备拥有多种接入方式，但仍然仅存在一条传输数据的路径，不能充分利用端设备的多接口及冗余 ISP 资源，其他路径、网络接口，无法同时使用，仅处于备份状态，这造成了大量的资源浪费，故通过多个 IP 接口实现冗余的 ISP 连接是当前国际 IETF 组织讨论的热点。

MPTCP（Multi-Path TCP）协议是 TCP 协议的扩展，很好的解决了 TCP 协议无法建立冗余 ISP 连接的问题[1]。MPTCP 使用多个 TCP 作为子流进行传输，充分利用资源，提高了吞吐量，并增强了链路的鲁棒性。随着 MPTCP 协议的逐渐完善[1]，协议中的大部分参数都有了较为合理的配置，但仍有部分参数的配置需要研究。例如，关于缓存的取值设置，传统的 TCP 协议的缓存设置的是一个固定值，这对于单路径传输来说足够使用[4]，然而 MPTCP 相对于 TCP 而言，子流数量和网络吞吐量有着很大的提升，数据包的排序需要更大的缓存空间，如果直接设置一个很大的值会造成资源浪费，如果设置太小则达不到最大吞吐量，这是一个动态的问题，无法使用静态的方法来解决[5]。

本文首先介绍了 MPTCP 协议技术原理和 TCP 缓存工作原理；提出一个缓存计算模型，可根据网络的实时情况，动态的设置缓存的大小，减少资源浪费；并且通过搭建实验环境，设计实验对缓存的变化进行分析，对所提出的模型进行分析验证；最后总结并提出下一步计划。

2 MPTCP 协议技术原理

2.1 MPTCP 简介

多路径传输的概念于 1995 年提出，2009 年 IETF 组织建立了 MPTCP 工作组，正式开展 MPTCP 草案审核的工作[6]。MPTCP 的结构如图 1 所示[7]，MPTCP 使用多条子流（IPv4 子流或者 IPv6 子流）传输数据，子流和正常的 TCP 子流类似。MPTCP 协议包括两个模块分别负责包调度（PS）和路径管理

收稿日期：2016-03-13

基金名称：下一代互联网多宿主系统国际测试床构建与性能分析，基金号：61363008。

基金名称：基于异构移动互联接入标准的多路径传输性能改进技术研究，基金号：61662020

作者简介：周峰(1990-)，男，江苏淮安人，海南大学信息科学技术学院 2014 级硕士研究生。研究方向为 MPTCP 分析，手机：15248942133，Email：zofon@qq.com

通信作者：周星(1958-)，女，重庆人，教授，主要研究方向为下一代互联网协议分析，手机：18078978177，E-mail：xingzhou50@126.com

（PM）[8]。应用首先通过 TCP socket API 接口将数据发送到包调度模块，然后由路径管理模块决定发送到哪条子流，子流即为 IPv4 或者 IPv6 的 TCP 子流。

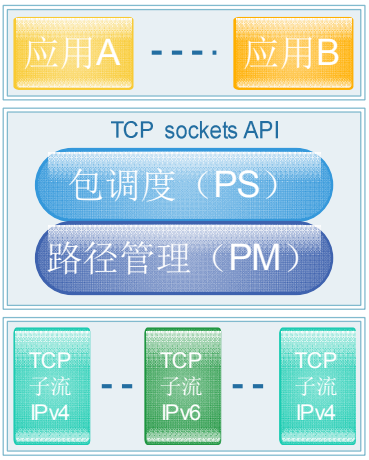


图 1 MPTCP 结构图

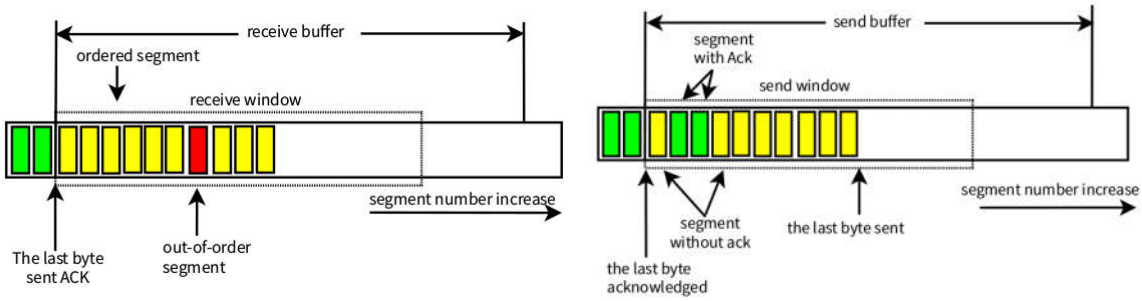
MPTCP 协议最重要的优点是向后兼容性，它能兼容现有的 TCP 的所有应用，是国际 IETF 组织关于下一代互联网协议的研究热点[9]。

当进行数据传输时，如果发送方传送的数据量超过了接收方的处理范围，那么接收方会出现丢包现象。为了避免出现此类问题，在进行数据传输时需要进行流量控制，滑动窗口是其中一个技术, TCP 数据包包头里有一个字段称通告窗口，该字段包含接收端剩余缓存的信息，发送端根据接收端的处理能力来发送数据从而保证接收端顺利处理数据。具体实现为数据传输双方在每次进行数据交互前声明各自的接收窗口 rwnd 大小，用来表示自己能保存最大的数据量。

2.2 缓存工作机制

2.2.1 TCP 缓冲机制

缓存是内核中接收数据的队列，拥塞窗口是缓存的一部分，在当前的 TCP 协议中采用滑动窗口的方式进行拥塞控制，发送端和接收端的滑动窗口如图 2 的(a)(b)所示。缓存的取值决定了拥塞窗口可以变化的上限。



（a）接收端缓存示意图　　（b）发送端缓存示意图

图 2 TCP 缓存结构示意图

Linux 内核中根据 net.ipv4.tcp_adv_win_scale 的值决定开销。其取值可以为 1 或 2，值为 1 表示缓存的 1/2 被用来做额外开销；值为 2 表示缓存的 1/4 被用来做额外开销，据此，缓存的合理值具体计算方法如公式 1 所示：

$$\text{Buffer\_size} = \frac{\text{BDP}}{1-\frac{1}{2^{\text{tcp\_adv\_win\_scale}}}} \quad (1)$$

其中，BDP = BandWidth * RTT。

2.2.2 MPTCP 缓存机制

在 TCP 中，接收窗口 cwnd 的合理值取决于带宽延迟积 BDP（BDP 等于带宽（BandWidth）和延迟（RTT）

的乘积）的大小，然而由于 MPTCP 中多条子流存在的原因，简单的参照 BDP 来设置缓存大小不再能满足传输需求[10]。

MPTCP 协议中增加了一个 MPTCP 连接级缓存用于重组各条子路径的数据[10]，这样可以保证每条子路径独立完成数据的接收之后将数据重组递交到应用层。

在多路径传输中，设有 n 条路径，各条路径的带宽和往返时间分别为 $Band_i$ 和 rtt_i，则 Buffer_size 至少应设置为公式 2：

$$\text{Buffer\_size} = 2 \times [\max_{1 \le i \le n}\{rtt_i\} \times \sum_{i=1}^{n} Band_i] \quad (2)$$

当出现拥塞或重传的时，最差的情况下要求最大 RTO（超时重传）的缓存时间，因此缓存所需要的空间应为公式 3：

$$\text{Buffer\_size} = \left(3 \times \max_{1 \le i \le n}\{RTT_i\} + \max_{1 \le i \le n}\{RTO_i\}\right) \times \sum_{i=1}^{n} Band_i \quad (3)$$

对于多路径传输本身而言，缓存的大小越大越好。然而系统的资源有限，传输数据占用太多内存就会影响到其他应用的运行，如果某些应用也有传输数据的请求，但是因为内存不足，这些应用就要进行内存对换，反过来会影响传输速度。

2.3 MPTCP 缓存优化模型

以传统的客户端服务器 C/S 模型为例，如图 3 所示，客户端有两个 ISP，服务器有三个 ISP，服务器服务 1000 个用户。使用传统的 TCP 协议时只需要建立 1000 个子流，而当服务器使用 MPTCP 协议之后，服务的用户数量还是 1000 个，但是建立的 TCP 子流数量变成了 6000 个，是原来的 6 倍，此时如果不能合理的设置缓存的大小，造成的资源浪费是很惊人的，且因为极高的并发量，内存很容易耗尽。

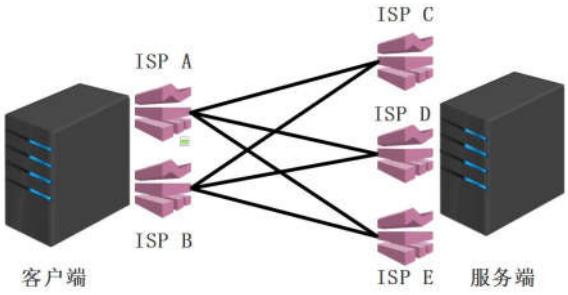


图 3 MPTCP 子流数量

通过对上文 TCP 和 MPTCP 工作机制的分析，为了提高 MPTCP 协议的工作效率，减少资源消耗，本文中提出一个计算高速低延时场景（如数据中心、局域网等）的 MPTCP 缓存的模型：

$$\text{Buffer\_size} = e^{\frac{TP}{3n \times 10^n} + n}$$

Buffer_size 是缓存设置值的大小，其取值和子流的数量有着密切的关系，TP 是本次传输能达到的最大传输速度，n 表示子流的数量，如图 3 中，发送端有两个 ISP，接收端有三个 ISP，则 n 为 6, 为了更好的体现出窗口和缓存的关系，在本模型中所使用的单位均为基础单位，模型函数如图 8，X 轴代表的是缓存的设置值，其单位为 Byte。Y 轴代表的是吞吐量，其的单位为 Bit。

通过该模型可以在传输开始之前计算出 MPTCP 连接所需要的缓存的大小，预先设置好相关参数，在不浪费系统资源的情况下达到最大传输速度。该模型在数据中心、局域网等场景下可以节省服务器大量的内存资源。

3 测试环境配置

测试环境分别对比了本地测试床和 NorNet Core 测试床，二者均基于真实网络，本地测试床排除了网

络波动的因素，可以在没有干扰的情况下得出 MPTCP 缓存增加的规律。NorNet Core 测试床规模更大，节点更多，可以反映真实网络环境中的状况，其得出的结果可以直接适用于当下网络的参数设置。

3.1 本地测试平台

为了排除其他的干扰因素对传输的影响，首先构建一个如图 4 所示的本地测试环境。

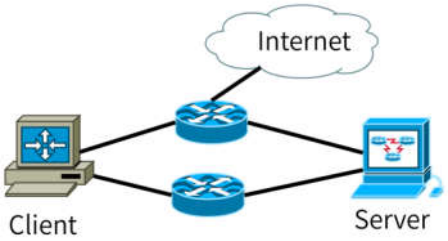


图 4 本地测试床结构示意图

本地测试平台的构建使用了两个路由器和两台安装 Ubuntu16.04 的多网卡主机。在操作系统中安装了 MPTCP 0.91.2 版本和最新版本的 Netperfmeter，其主要用于建立连接和发送数据。数据的处理由自己编写脚本完成。

3.2 NorNet Core 测试床

NorNet Core 多宿主系统测试床是由挪威 Simula 实验室主导，世界各高校参加构建的大规模测试床[12]，该测试床用于运行传输协议相关的测试实验，其结构图 5 所示：

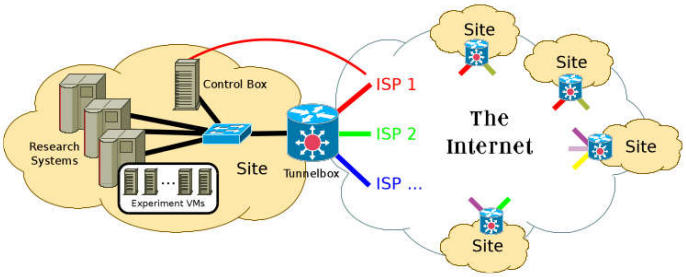


图 5 NorNet Core 结构图

左边是测试床中每个站点的结构图，其中包括 control box、tunnelbox、二至多个供应商（ISP）。Tunnelbox 右边是测试床的其他站点[13]。Linux 内核设置如下：

(a) Linux 内核版本 4. 1. 27。

(b) MPTCP 版本 0. 91. 3 使用“fullmesh”路径管理算法。

(c) TCP 和 MPTCP 的缓存默认取值为 4MiB，最大值限制在 16MiB（包括 SO_SNDBUF 和 SO_RCVBUF 的设置）。

在本次实验选取了 NorNet 国际测试床的 UiB 与 UiS 两个大学的站点中的 floeibanen 和 sokn 两个的节点作为实测对象，两个均为高带宽低延时节点，其节点信息如表 1 所示：

表 1 节点信息

	Site	Node	ISP1	ISP2	ISP3
FromNode	UiB(Universitetet i Bergen)	floeibanen	Uninett(fibre)	BKK(fibre)	
ToNode	UiS(Universitetet i Stavanger)	sokn	Uninett(fibre)	Altibox(ADSL)	PowerTech(ADSL)

NorNet Core 测试床上进行的实验均通过在 Python 脚本中使用 NetPerfMeter 进行测量，运行的次数为 10 次取平均值以满足统计的需求。脚本由三个核心部分组成：

(a) 登录到测试床和 PLC，由调用测试床提供的 API 实现：

```
1    loginToPLC() #登录到测试床
```

```
2    fullSiteList = fetchNorNetSiteList() #获取实验节点的网络参数
3    … #获取其他信息
```

(b) 根据运行的实验配置参数，运行脚本，在本文的实验中通过多重循环缓冲区的取值实现。

```
1   for localProviderIndex in localProviderList: #根据供应商设置测试顺序。
2     for remoteProviderIndex in remoteProviderList:
3       for buffer_size in range(…) #设置缓冲区的取值。
4         … #配置测试参数
5         allRuns.append(…) #将参数添加到一次测试中
```

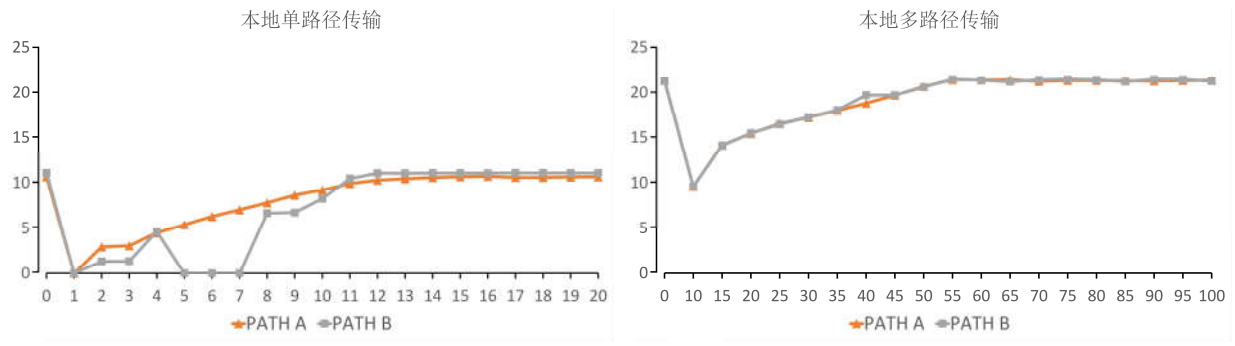
(c)数据采集，可视化，通过 R 脚本编写的 copyFromNodeOverRSync()模块实现。

4 缓存测试分析

本文实验顺序如下：首先，利用本地测试床进行实验测试，根据得到的结果，分析缓存用量增加的规律；然后在 NorNet Core 测试床上进行实验验证，利用结果建立模型。

4.1 本地测试床结果分析

本地测试床的数据结果如图 6 所示。



(a)TCP 单路径传输　　(b)MPTCP 多路径传输

图 6 本地测试床数据展示

图 6 中，X 轴是缓存用量的变化值，其单位 KiByte，缓存设置的最大取值为 100KiByte。Y 轴是吞吐量，其单位为 MiByte。当脚本的缓存用量设置为 0 时，操作系统使用内核中固定值的设置方法，该值作为实验的参照值。

在本地测试床的结果中，在达到上限之前吞吐量和缓存大小的取值呈线性关系，我们可以根据公式准确的计算出所需要的缓存的大小。同时，因为数据调度增加了 MPTCP 的复杂性，增加了系统的开销，MPTCP 的吞吐量小于两条单路径吞吐量之和。

在单路径情况下，将 PATH B 缓存大小设置值为 5、6、7 的时候，出现了吞吐量为 0 的现象，这种现象极少出现，可能和特定路径路由有关，这组结果很有意思，值得继续研究，目前暂不清楚是什么原因导致这样的情况，有待进一步分析。

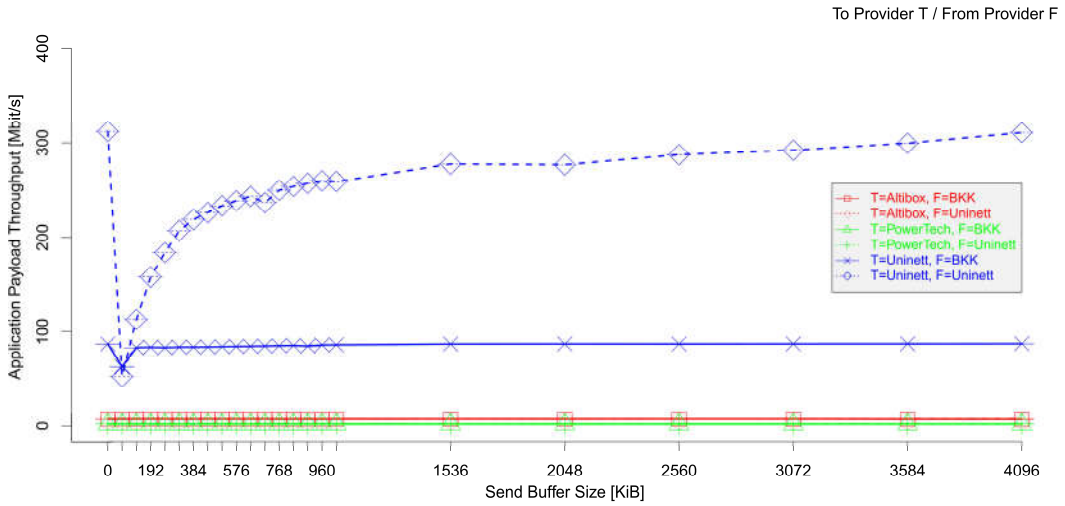
从本地测试床展示的结果中，可以推断出，在一定范围内缓存和吞吐量呈线性关系，可以用对数函数或者分段线性函数来模拟。分段函数的模拟结果如公式 4：

$$\mathrm{TP}=\begin{cases}\text{Buffer\_size} * 0.942 & (0 < Buffer\_size < 15)\\ \text{Buffer\_size} * 0.183 + 11.38 & (15 \le Buffer\_size < 55)\\ 21.37 & (55 \le Buffer\_size)\end{cases} \quad (4)$$

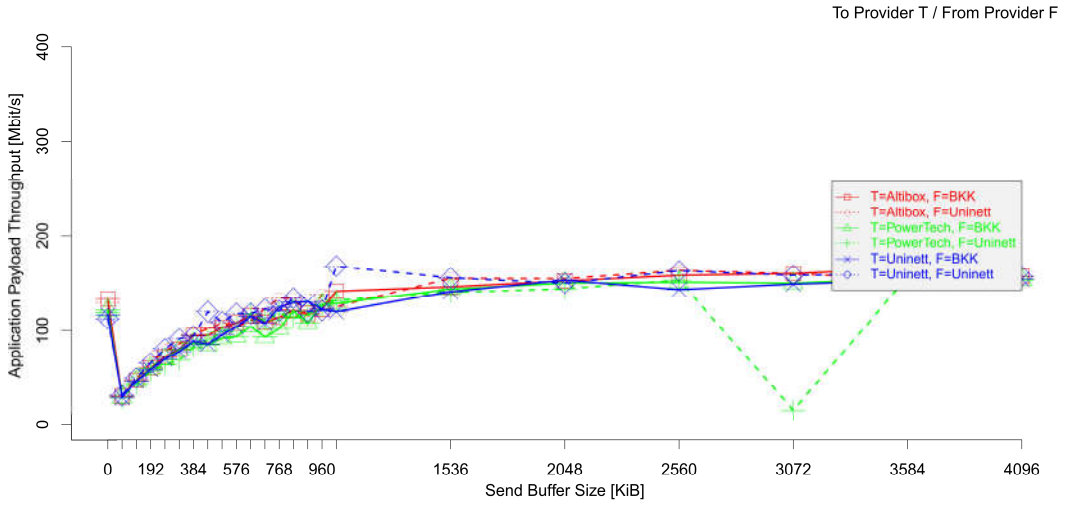
针对本地测试床场景，分段函数的拟合度非常高，甚至可以完全拟合。所以分段函数可以很好的描述缓存和吞吐量之间的关系，但是根据实验结果得到的描述函数很难用来预测缓存和吞吐量之间的关系。在函数公式 4 的基础上，可以使用对数函数来模拟。使用真实测试床的结果推导出对应的对数关系。

4.2 NorNet 测试床结果分析

因为测试床的网络状况极好，延时很低，带宽很高，其网络情况与数据中心内部传输很相似。实验结果如图 7 所示。



(a) TCP 单路径传输



(b) MPTCP 多路径传输

图 7 NorNet 国际测试床测试结果

图 7(a)为单路径传输，图 7(b)为单路径传输，缓存取值为 0 时表示采用内核默认的算法，此时的缓存设置为 4MiB。(a) (b)对比可以看出高带宽高速网络传输数据时所需的缓存明显大于低速网。在图 7(a)单路径传输中，子流 Uninett-Uninett 需要 1Mib 的缓存，而其他的子流只需要 200KiB 左右的缓存。在图 7(b)中，使用默认的缓存设置的方法造成了系统资源紧缺，反而无法达到吞吐量的上限，而合适的缓存取值则能达到吞吐量的上限，该值约为 1MiB。考虑到链路巨大的异构性，尽管图 7(b)中的吞吐量上限未达到单路径最佳子流(子流 Uninett-Uninett)的吞吐量，MPTCP 的表现仍然非常优秀，尤其在设置了合理的缓存的值之后。

根据测试结果，建立分析函数如图 8 所示，X 轴代表的是缓存的设置值，其单位为 Byte。Y 轴代表的是吞吐量，其的单位为 Bit。

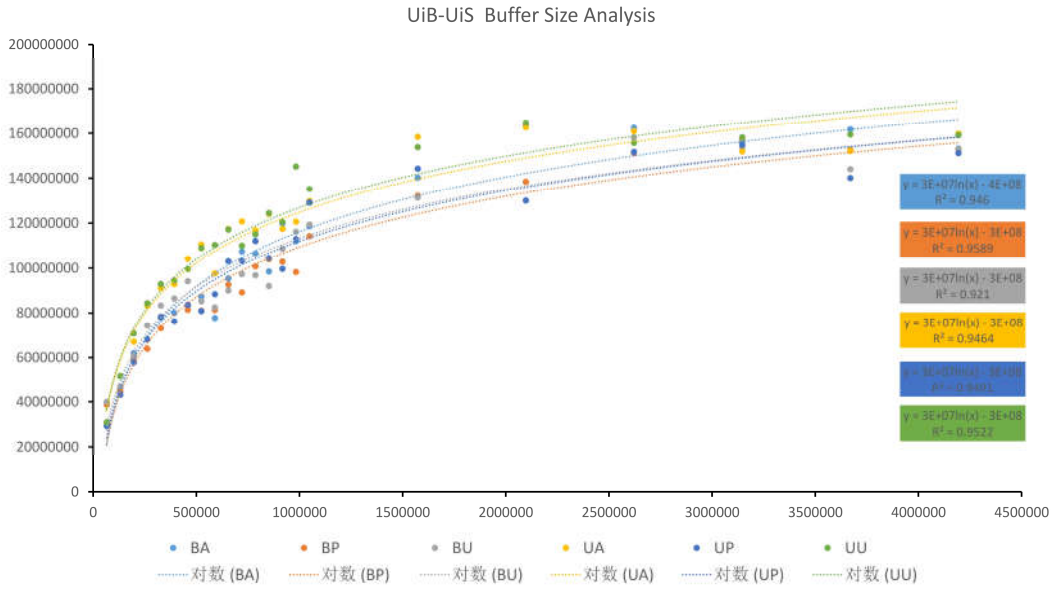


图 8 NorNet 测试床结果分析函数

当缓存的取值到达一定大小的时候，吞吐量达到上限。定义 TP_upper_limit 为达到吞吐量上限时分析函数的导数 Y’的取值，用于判定最佳的缓存设置值。模型中当 Y’〈TP_upper_limit 时，缓存已经达到一个合适的取值，无需增加。所得到的缓存和吞吐量关系如公式 5，根据实验结果，Y’的计算如公式 6。

$$\mathrm{TP} = 3 * 10^7 * \ln(\mathrm{Buffer\_size}) - 3 * 10^8 \quad (5)$$

$$\mathrm{TP\_upper\_limit} = \frac{3*10^7}{\mathrm{Buffer\_size}} \quad (6)$$

此时根据 2.3 节提出的模型计算缓存取值：$\mathrm{Buffer\_size} = e^{\frac{150000000}{18\times10^6}+6}$=1677810（1.67MiB），在图 8 中可以观察到最佳的缓存取值约为 1500000（1.5MiB），模型计算出的取值略大于最佳的缓存取值，然而远小于当前内核的默认的 4MiB，节省了近 60%的资源。

通过以上分析可以得出，对于当前的 MPTCP 协议子流数量增加带来的缓存消耗问题，使用内核默认设置会造成巨大的浪费，而本文提出的缓存计算模型在传输开始之前计算缓存的大小，所设置的缓存取值可以有效的提高吞吐量上限，减少资源浪费，是解决缓存资源分配的一个相对合理的方案。

5 结束语

本文主要研究 MPTCP 的缓存，并根据真实环境下的实验结果进行分析，对缓存与吞吐量的关系建立了对数函数模型。本文实验所选取的真实环境中的测试数据很好的表现出了缓存和吞吐量的关系，并以此为基础针对高速低延时网络环境建立了一个初步的模型，结果的拟合度高达 0.9，部分甚至达到了 0.95，测试的结果可以减少几乎一大半的缓存消耗。

本文下一步的工作是对更多的场景进行实验测试，以多场景测试数据为基础，对不同的模型进行比较分析。提出通用度更高的模型。

参考文献

[1] Bonaventure O, Handley M, Raiciu C. An overview of Multipath TCP[J]. USENIX login, October 2012, 37(5):17-

[2] Barr, bastien, Paasch C, et al. MultiPath TCP: from theory to practice[C]// NETWORKING 2011 -, International Ifip Tc 6 NETWORKING Conference, Valencia, Spain, May 9-13, 2011, Proceedings. DBLP, 2011:444-457.

[3] 刘骥, 谭毓银, 符发,等. MPTCP 与 CMT-SCTP 拥塞控制机制研究[J]. 计算机工程, 2015, 41(4):117-124.

[4] 韩涛, 朱耀庭, 朱光喜,等. 考虑慢启动影响的 TCP 吞吐量模型[J]. 电子学报, 2002, 30(10):1481-1484.

[5] Becke M, Adhari H, Rathgeb E P, et al. Comparison of Multipath TCP and CMT-SCTP based on Intercontinental Measurements[C]

Global Communications Conference (GLOBECOM), IEEE,2013: 1360-1366.

[6] Bagnulo M, Paasch C, Gont F, et al. Analysis of Residual Threats and Possible Fixes for Multipath TCP (MPTCP)[J]. Journal of Infectious Diseases, 2015, 200 suppl 1(4):S147-53.

[7] 符发, 周星, 杨雄,等. MPTCP 与 CMT-SCTP 多路径传输协议性能分析[J]. 计算机工程与应用, 2013, 49(21):79-82.

[8] M. Scharf, A. Ford, Multipath TCP (MPTCP) Application Interface Considerations[J]. Heise Zeitschriften Verlag, 2013. RFC 6897.

[9] Ford A, Raiciu C, Handley M, et al. TCP Extensions for Multipath Operation with Multiple Addresses : draft-ietf-mptcp-multiaddressed-03[J]. Internet draft - draft-ietf-mptcp-multiaddressed-07, 2011.

[10] 符发, 周星, 谭毓银,等. 多场景的 MPTCP 协议性能分析研究[J]. 计算机工程与应用, 2016, 52(5):89-93.

[11] Liu J, Rayamajhi A, Martin J. Using MPTCP subflow association control for heterogeneous wireless network optimization[C]// International Symposium on Modeling and Optimization in Mobile, Ad-Hoc and Wireless Networks. 2016:1-8.

[12] Dreibholz, T, Xing Zhou, Fu Fa. Multi-path TCP in Real-World Setups -- An Evaluation in the NORNET CORE Testbed[C]// IEEE, International Conference on Advanced Information NETWORKING and Applications Workshops. IEEE, 2015:617-622.

[13] Dreibholz T, Bjorgeengen J, Werme J. Maintaining and Monitoring the Infrastructure of the NORNET CORE Testbed for Multi-homed Systems[C] //IEEE, International Conference on Advanced Information NETWORKING and Applications Workshops. IEEE, 2015:611-616.

Research on Multi-path TCP Cache Size

ZHOU Feng[1], ZHOU Xing[1], WANG Kun[1],LIAO Yuli[1],Fu Fa[1]

(1. College of Information Science and Technology, Hainan University, Haikou Hainan, 570228)

Abstract: In the multi-path transport protocol MPTCP, it is necessary to maintain a large cache for data collation and reorganization due to the existence of multiple sub-streams. Insufficient cache will cause the program to hang. In this paper, the technical principle of MPTCP protocol is introduced firstly. Then, the relationship between cache size and throughput of TCP protocol and MPTCP protocol is studied. Based on the research results, a cache calculation model for low latency and high bandwidth scene is proposed to calculate MPTCP cache size. Next, this paper analyzed local experiments and NorNet Core test bed experiment. The results showes that the cache value designed according to the model was only half that of the current Linux kernel cache, which greatly reduced the waste of resources and reached maximum of throughput.

Keywords: Multipath Transport Protocol; Cache Analysis; Cache Model