并行多路传输中数据调度算法的研究*

陶 洋[1], 张传欣[1], 代 堑[2], 李 攀[1]
(1. 重庆邮电大学, 重庆 400065; 2. 广州杰赛科技股份有限公司, 广州 510310)

摘 要: 针对传统并行多路传输中数据调度算法存在的问题,基于 MPTCP 协议,提出了带宽预测和前向时延的数据调度算法(data-scheduling algorithm using bandwidth estimation and forward trip-time, DA-BEFT)。该算法充分考虑子流间传输时延差较大的影响,结合性能好的重传选路策略,减轻接收端因数据乱序导致的缓存阻塞,提高整个连接吞吐量。通过仿真实验验证了 DA-BEFT 在子流时延差变化时能够提高带宽利用率,提高网络的吞吐量。

关键词: MPTCP; 并行多路传输; 缓存阻塞; 数据调度
中图分类号: TP393 **文献标志码**: A **文章编号**: 1001-3695(2013)04-1155-03
doi:10.3969/j.issn.1001-3695.2013.04.053

Research of data-scheduling algorithm in concurrent multipath transfer

TAO Yang[1], ZHANG Chuan-xin[1], DAI Qian[2], LI Pan[1]
(1. *Chongqing University of Posts & Telecommunications*, *Chongqing* 400065, *China*; 2. *GCI Science & Technology Co., Ltd*, *Guangzhou* 510310, *China*)

Abstract: This paper proposed a data-scheduling algorithm using bandwidth estimation and forward trip-time (DA-BEFT) combined with the existing works. It considered fully the negative impact caused by the significant difference of time delay along subflows, and reduced the receive buffer blocking caused by out-of-order data combined with optimal retransmission policy, and improved the entity connect-level throughput. Simulation results show that, DA-BEFT can effectively improve the utilization of available bandwidth resources and the throughput of the entire connection.

Key words: MPTCP; concurrent multi-path transmission; buffer blocking; data-scheduling

0 引言

MPTCP[1]是 IETF 组织提出的 Multipath TCP,实现多路并行传输,并向后兼容常规 TCP 和中间件。MPTCP 协议机制部分与 CMT-SCTP 相近,如数据调度都是采用 Round Robin 机制,但 MPTCP 又有自己的独特机制,如发生丢包超时需要在原子流和其他不同的子流上同时重传丢包。因此,对 MPTCP 的研究可以借鉴 CMT-SCTP 的研究成果,但又要结合 MPTCP 的特性,针对相关问题提出可行、有效的方案。

MPTCP 协议中问题的分析如下:a)并行多路传输要求主机上分配较大的缓存资源[2],但是实际上不可能按理论要求足额分配,因此 MPTCP 中缓存资源相对紧张;b)并行多路传输中,由于子流间性能的差异(带宽、时延),接收端数据乱序是不可避免的,MPTCP 数据调度采用 Round Robin(轮询)机制,优点是调度机制简洁,缺点是没有考虑 RTT 差异对数据乱序的影响,当子流间的 RTT 差异较大,特别是当 RTT 较大的子流发生丢包超时或连续的丢包超时,会造成严重的接收缓存阻塞;c)MPTCP 标准中指定,数据需要在连接级和发送该数据的子流上都被确认,MPTCP 推荐采用保守的重传算法,即先在原子流上重传,当重传超时发生后,在其他子流和原子流上同时重传丢包,当子流意外中断,要经过较长的时间才能侦测到,此时会发生严重的发送缓存阻塞,频繁的丢包重传也会导致一定程度的缓存阻塞;d)缓存阻塞限制了子流完全利用可用的带宽资源,MPTCP 的高吞吐量得不到实现,性能差的子流还会影响性能好的子流,降低其吞吐量。

通过上面的分析可以看出,缓存阻塞导致数据得不到及时提交[3],其中重传超时影响最大,而接收缓存阻塞主要是由乱序和丢包导致的。因此,本文在 MPTCP 协议上提出基于带宽预测和前向时延的数据调度算法,来减轻缓存阻塞问题,充分地利用可用的带宽资源。

1 带宽预测和前向时延的数据调度算法(DA-BEFT)的设计

1.1 基本描述

1)可用带宽 B_i

B_i 为子流上 i 的实际可用带宽。常用的估算算法为 SBPP[4] 和 Westwood[5]。

为了更准确地估算动态带宽,增加平滑性,需要采集多个

收稿日期: 2012-09-04; **修回日期**: 2012-10-28 **基金项目**: 重庆市科技攻关计划资助项目(CSTC,2009AB2245);重庆市教委科技资助项目(KJ090516)

作者简介:陶洋(1964-),男,重庆人,教授,博士(后),主要研究方向为自组织网络、网络管理技术研究及应用(chuanxin3000@126.com);张传欣(1987-),男,重庆人,硕士研究生,主要研究方向为自组织网络、网络管理及应用;代堑(1979-),男,重庆人,工程师,本科,主要研究方向为传送网、数据网的咨询、规划、设计;李攀(1986-),男,重庆人,硕士研究生,主要研究方向为自组织网络、通信与信息系统.

样本的值,如式(1)所示(α 是一个比率因子,本文中取经验值 0.875):

$$B_i \leftarrow \alpha B_i + (1-\alpha) B_i \tag{1}$$

2) RTT_i[6] 和 FT_i

$$FT_i = \mathrm{RTT}_i / 2 \tag{2}$$

当子流 i 接收到 ACK 确认,就会估算当前的实际的 RTT_i 值。为了平滑性,结合前面的取值样本,计算 RTT_i 的公式为(β 是一个比率因子,本文中取经验值 0.75)

$$\mathrm{RTT}_i \leftarrow \beta \mathrm{RTT}_i + (1-\beta)\mathrm{RTT}_i \tag{3}$$

3)相关性因子 γ

MPTCP 建立的连接中,发送节点与接收节点之间建立多条子流,利用 γ 判断子流之间的相关性。

假设连接中有 n 条子流,相对应的前向传输时延组成集合 $\{FT_1, FT_2, \cdots, FT_n\}$,取 $FT_{\min} = \min\{FT_1, FT_2, \cdots, FT_n\}$ 为集合中的最小值。满足 $FT_i \leqslant \gamma FT_{\min}$ 的子流就认为具有相关性,这些子流统称为第一类子流;集合 $[FT_1, FT_2, \cdots, FT_n]$ 中去掉第一类子流后,按相同的方法选择的子流为第二类子流;后面可以接着选择第三类、第四类子流,直到所有子流都被归类。

4)发送起始 DSN

并行多路传输中,用多条路径同时传输数据到接收端[7]。任意路径 i 包括参数 RTT_i、FT_i 和 C_i,C_i 为路径 i 上的拥塞窗口。假设包具有相同的大小 MSS(maximum segment size),所以每个包的队列延迟是固定的,则在路径 i 上传输每个数据包的时间 t_i 为

$$t_i = \frac{\mathrm{size}_{\mathrm{pack}}}{B_i} \tag{4}$$

假设在路径 i 上从时间 t 发送 C_i 个数据包,则到达接收端的时间分别为 $t + FT_i + t_i$、$t + FT_i + 2t_i$、$\cdots$、$t + FT_i + C_i t_i$,相应的确认时间为 $t + \mathrm{RTT}_i + t_i$、$t + \mathrm{RTT}_i + 2t_i$、$\cdots$、$t + \mathrm{RTT}_i + C_i t_i$。在成功接收到确认后,拥塞窗口 C_i 将增加,一个新的传输将从 $t + \mathrm{RTT}_i + C_i t_i$ 开始,则子流 i 发送 C_i 个数据包到达接收端的时间为 $FT_i + C_i t_i$。

如果有两条子流 i 和 j,$\mathrm{RTT}_i < \mathrm{RTT}_j$,两条子流上同时发送数据包,$i$ 子流将先发送到对方,希望数据尽量按序到达接收端,因此子流 i 发送 DSN 小的数据包,子流 j 发送 DSN 大的数据包。如果子流 j 允许发送 C_i 个数据包,估算花费的时间为 P_{cj},需要计算在时间 P_{cj} 内,子流 i 能够发送多少个数据包,才能计算出子流发送的起始包。接下来估算子流 j 需要发送的起始包 N_j,即算出子流 i 在 P_{cj} 时间内能传输的总的数据包数。假设允许发送 n_s 个包,则第 n_s 个包到达接收端花费的时间为 $FT_i + n_s t_i$。定义 n_s 为

$$n_s = \begin{cases} n_s, n_s < C_i, \sum FT_i + n_s t_i < P_{cj} \\ 0, n_s \geqslant C_i, \text{或} \sum FT_i + n_s t_i \geqslant P_{cj} \end{cases} \tag{5}$$

$$N_j = \sum\nolimits_{\sum} FT_i + n_s t_i < P_{cj} n_s \tag{6}$$

这里利用相关性定义把子流归为各自的集合,集合内的子流传输时延差相对较小,不采用上述策略计算发送起始包,每个集合在逻辑上看做一个子流,集合间时延可能相差较大,需利用式(5)和(6)计算集合间的起始发包 DSN 值 N_m。规定如下:

如果 i 为第一类子流,则

$$N_1 = 0 \tag{7}$$

发送整个连接需要发送的下一个包的数据序列号,即 DATA-ACK;

否则

$$N_m = \sum\nolimits_{0<i<m-1} \sum\nolimits_{\sum} FT_i + n_s t_i < P n_s \quad i = 1, 2, \cdots, m \tag{8}$$

其中:N_m 为第 m 类子流的发送起始包,P 为 m 类子流发送允许大小的包的预测时间。

1.2 DA-BEFT 设计

利用 MPTCP 协议实现并行多路传输,提高源主机与目的主机之间的吞吐量。但是 MPTCP 中使用的数据调度策略为:采用 Round Robin(轮询)的数据调度算法[8],任何子流有 ACK 确认信息,则以轮询的方式发送该路径上拥塞窗口允许个数的数据包到网络上。这种策略简化了包调度策略的复杂度,但是并行传输中多子流的性能(带宽、延时)不同,会导致后发的包抢先到达接收端,引起数据乱序,先到的乱序包占用了本来紧张的缓存资源,使并行多路传输的高吞吐量得不到好的实现。文献[9]指出:考虑了丢包率的重传策略表现更好,SSTHRESH 能够更好地反映路径的丢包状况;选择最大的 SSTHRESH 的路径重传丢包,能够减少丢包重传的发生,提高并行多路传输的吞吐量。

基于以上的分析,本文提出基于带宽预测和前向时延的数据调度算法。该算法使用的策略包括:

a)估算每条路径的当前可用带宽。

b)利用相关性因子给所有子流分类。

c)采集每条路径的 RTT,估算前向传输时延 FT_i。结合可用带宽,采用一种基于 FT_i 的数据调度策略,促使多子流上的数据尽可能按原顺序到达接收端;同一类别的子流集合内,时延相对较小,利用最快子流优先算法(fastest subflow first, FSF)。

d)当发生丢包时,利用最大 SSTHRESH 优先算法选择丢包重传的路径[10]。

最快子流优先算法 FSF 是为了尽量避免接收端乱序。发送端收到 ACK 确认,同时多条子流上有新的累积确认值,按要求更新各子流的拥塞窗口。算法思想如下:

(a)对于 $\forall i$ 子流,具有新的累积确认值,假设其可用带宽为 B_i,发送但未确认的数据量大小为 O_i。

(b)发送端在发送一个大小为 D 的分组前,计算其处理时间 $R_i = (O_i + D)/B_i$。

(c)在允许发包的子流中,选择具有最小的 R_i 传输分组。

R_i 的大小取决于两个因素:O_i/B_i 表示当前正在发送的数据量需要花费的时间;D/R_i 是当前路径上传输大小为 D 的数据快花费的时间。这种算法子流的选取取决于上次发送的数据量,与整个连接的生命周期无关。但这种方法没有考虑传输时延的影响,适合时延差异较小的并行传输。

DA-BEFT 算法的目的是尽量保证数据按序到达接收端,并利用最大 SSTHRESH 优先策略选择路径传输丢失的数据包,减少重传超时的发生。算法描述如下:

a)已经建立的多条子流,获得各条子流的 FT_i 和可用带宽 B_i。发包前利用 SBPP 算法估算 B_i,发送过程中利用 Westwood 算法估算。

b)采用相关性因子和子流分类原则给所有子流分类;刚加入的子流直接归为最后一类子流。

c)发送端准备传输数据,利用式(7)(8)计算各类子流的发送起始 DSN,各类子流内利用最快子流优先算法;子流集合间每次从最优子流集合开始。

d)子流 i 上发生丢包,采用保守的重传算法重传丢包,即先在原子流上重传丢包,如果发生丢包超时,则利用最大 SSTHRESH 优先算法选择不同子流 j,在 i 和 j 上同时重传丢包。

e)发送数据过程中,利用式(1)(3)统计任意正在工作的子流 i 上的 RTT_i 和可用带宽 B_i,每隔 60 s 更新子流的 FT_i 值,重新归类。

2　仿真与分析

并行多路传输中,带宽和时延对整个连接的性能影响较大,主要是接收端乱序问题,从而导致的缓存阻塞。本文设计的 DA-BEFT 数据调度算法的目的是提高带宽利用率,因此设计如下场景与原 MPTCP 协议进行仿真对比,如表 1 所示。

表 1　仿真环境参数设置

仿真参数	数值
传输层协议	MPTCP/USM_BEFT_MPTCP
节点数	2
拥塞控制算法	RTT_Compensator
包重排序算法	D_SACK
发送数据量	10 000 000 Byte

场景一通过下列指标反映对网络性能的影响:

平均带宽利用率(average bandwidth utilization efficiency),即在传输一定量数据中的带宽利用率的平均值。

$$\text{平均带宽利用率} = \frac{1}{N}\frac{\sum \text{Pac}_{\text{rec}} \times P_{\text{size}} \times 8}{\text{RTT}_T \times \sum B\omega t}$$

其中:N 为采集的总的带宽利用率的样本数。

同时给出了在某个特定时延下的指标,即带宽利用率(bandwidth utilization efficiency)和总的拥塞窗口(total cwnd)。

本场景的设置主要反映四条子流时,路径间的时延差异对 MPTCP 协议的影响,比较了它们的平均带宽利用率,同时给出了当 $n=0$ 时的带宽利用率。仿真数据选取如表 2 所示。图 1 为仿真模拟网络拓扑。

表 2　仿真数据选取

路径 ID 参数	子流 1	子流 2	子流 3	子流 4
带宽/Mbps	5	5	2	2
时延/ms	20	(20 + n × 5)	(20 + n × 10)	(20 + n × 20)
发送缓存	128 000 Byte			
接收缓存	64 000 Byte			
其中, n = 0,1,2,3,4,5,6,7,8,9,10				

图 2 为平均带宽利用率。从图 2 中能够得出 DA-BEFT 的平均带宽利用率比 MPTCP 要高;但随着时延差异变大,它们的平均带宽利用率总体上呈下降趋势,但变化相对平稳。同样,DA-BEFT 的性能优于 MPTCP,主要是由于 DA-BEFT 采用了 DA-BEFT 数据调度算法,尽量保证了数据按序到达接收端,减少了缓存阻塞,提高了带宽利用率。

图 3 显示的是当 $n=0$ 时,MPTCP 和 DA-BEFT 总的带宽利用率随时间的变化。从图中可以看出,DA-BEFT 的带宽利用率高于 MPTCP,其变化相对平稳。此时,四条子流都属于一个集合,DA-BEFT 在集合内部利用最快子流优先算法,实现了负载均衡,同时也在一定程度上保证了数据的按序到达,减少了缓存阻塞,提高了带宽利用率。

以上仿真结果表明,在不同的子流数和时延差的情况下,MPTCP 采用 DA-BEFT 数据调度算法,比采用 Round Robin 能提供更高的带宽利用率,提高了整个系统的吞吐量。

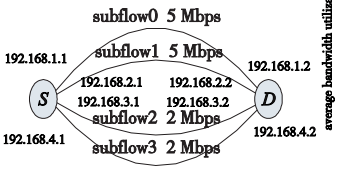

图1　仿真模拟网络拓扑

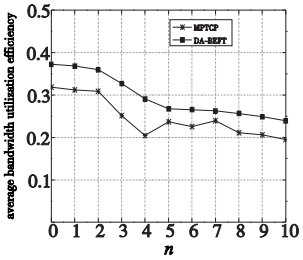

图2　平均带宽利用率

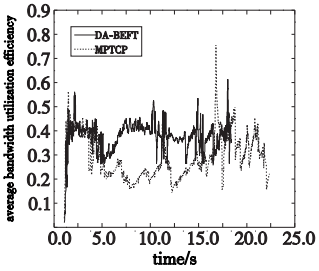

图3　带宽利用率(n=0)

3　结束语

DA-BEFT 采用新的数据调度算法,即在子流类别集合内采用最快子流优先算法,集合间采用预算发送起始 DSN,尽量保证数据按序到达接收端,减少因乱序导致的缓存阻塞。仿真实验验证了 DA-BEFT 在子流时延差变化(四条子流)时能够提高带宽利用率和网络的吞吐量。仿真结果表明,DA-BEFT 在一定程度上提高了其带宽利用率和并行多路传输的吞吐量。

参考文献:

[1] IYENGAR J, FORD B. An architectural perspective on MPTCP[S]. 2009.

[2] RAXSON V, ALLMAN M. RFC 2988, Computing TCP's retransmission timer[S]. 2000.

[3] 谢希仁. 计算机网络[M]. 5 版. 北京:电子工业出版社,2007.

[4] OU C S, ZHANG Jing, ZANG Hui, *et al*. Near-optimal approaches for shared-path protection in WDM mesh networks[C]//Proc of IEEE International Conference on Communications. 2003:1320-1324.

[5] CASETTI C, GERLA M, MASCOLO S, *et al*. TCP westwood: bandwidth estimation for enhanced transport over wireless links[J]. ACM Wireless Networks, 2002, 8(5):467-479.

[6] KARN P, PARTRIDGE C. Improving round-trip time estimates in reliable transport protocols[J]. ACM SIGCOMM Computer Communication Review, 1995, 25(1):66-74.

[7] FORD A, RAICIU C, HANDLEY M, *et al*. RFC 6182, Architectural guidelines for multipath TCP development[S]. 2011.

[8] Mptcp-ns3-implement multipath TCP on ns3 [EB/OL]. http://code.google.com/p/mptcp-ns3/wiki/MakeIt.

[9] IYENGAR J R, AMER P D, STEWART R. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths [J]. IEEE/ACM Trans on Networking, 2006, 14(5):951-964.

[10] IYENGAR J, AMER P, STEWART R. Retransmission policies for concurrent multipath transfer using SCTP multihoming[C]//Proc of IEEE International Networks Conference. 2004.