doi:10.3772/j.issn.1002-0470.2013.09.003

面向高速网络的多核并行 TCP 拥塞控制算法①

查奇文②[*,**] 张 武[*] 曾学文[*] 郭秀岩[*]

([*]中国科学院声学研究所 国家网络新媒体工程技术研究中心 北京 100190)

([**]中国科学院大学 北京 100190)

摘 要 为了解决在高速网络下,多核处理器系统中由于传统 TCP 的串行处理方式带来的性能瓶颈,以 TCP 协议的滑动窗口机制和拥塞控制算法为基础,提出了一种多滑动窗口并行 TCP 拥塞控制算法:MulWinTCP。MulWinTCP 将全局滑动窗口划分为多个子滑动窗口,利用多核处理器对多个子滑动窗口进行并行处理,消除了由于 TCP 串行处理方式带来的性能瓶颈。同时,MulWinTCP 保证了其对应用层的透明,并且兼容传统 TCP 协议。基于 TCP 吞吐率的 Mathis 数学模型,推导出了 MulWinTCP 吞吐率的数学模型,并通过实验仿真论证了算法的有效性和吞吐率数学模型的正确性。

关键词 多核处理器,传输控制协议,拥塞控制算法,滑动窗口,高速网络

0 引 言

目前多核处理器是处理器应用的趋势,基于多核的并行 TCP 算法研究具有重要实际意义。传统 TCP 协议在面向高速网络时,由于窗口大小限制、窗口管理机制、抖动以及串行的处理方式等原因,造成 TCP 不能充分地利用带宽资源[1]。许多研究者对 TCP 的窗口管理机制和拥塞控制算法进行改进,提出了多种高速 TCP 协议改进方案,如 BIC-TCP[2]、H-TCP[3]、FAST-TCP[4]、CUBIC[5] 等。这些改进方案虽然可以提高 TCP 的处理性能,但仍然是基于串行 TCP 的思想,在应用于多核处理器时,串行的处理方式仍然会带来性能瓶颈。例如,1b/s 的 TCP 吞吐率要消耗 1Hz 左右的 CPU 资源[6],如果 TCP 串行处理,在 10Gb/s 以上甚至是 100Gb/s 的高速网络时,需要与之相当频率的处理器,这显然很难实现。而通过 TCP 并行处理,可提高 TCP 在高速网络下的性能,因而实现 TCP 并行处理是一个急需解决的问题。

文献[7,8]中提出了 MulTCP 算法,通过拥塞窗口的变化,在 TCP 层使用一条 TCP 数据流去模拟 N 条并行的 TCP 数据流。但是研究发现,MulTCP 算法无法达到单条 TCP 流吞吐量的 N 倍。文献[9]针对 MulTCP 进行改进,提出了 MulTCP2 算法。MulTCP2 算法对 MulTCP 算法中拥塞控制算法的加性增乘性减因子进行了调整,使得 MulTCP2 算法的吞吐量达到了单条 TCP 流的 N 倍。MulTCP 以及 MulTCP2 虽然提出了并行 TCP 的思想,但其仍是使用一条 TCP 流的串行处理的方式。并行 TCP 协议[10-14]通过采用真实的多条 TCP 流并行传输,可以很好地利用多核处理器通过并行处理来提升性能。但是实际应用中,多条 TCP 连接之间的数据同步问题,以及数据的拆分和重组带来的额外开销问题,导致其性能并不能随着 TCP 流数目的增长而线性增长。而且并行 TCP 协议属于应用层协议,并不能做到传输层以上透明,与传统的 TCP 协议也无法兼容。本文基于 TCP 滑动窗口机制和 TCP 拥塞控制算法,提出了一种基于多滑动窗口的多核并行 TCP 拥塞控制算法 MulWinTCP,并通过理论推导和实验证明了其有效性。

1 MulWinTCP 算法描述

1.1 TCP 拥塞控制算法

目前使用最广泛的 TCP 版本都使用基于滑动

① 863 计划(2011AA01A102)和中国科学院战略性先导科技专项(XDA06010302)资助项目。

② 男,1986 年生,博士生;研究方向:网络新媒体技术;联系人,E-mail:zhangw@dsp.ac.cn

(收稿日期:2012-12-11)

窗口的拥塞控制算法。TCP 的拥塞控制算法包括四个部分,分别是慢启动、拥塞避免、快速恢复和快速重传[15,16]。

图 1 描述了 TCP 拥塞控制各阶段拥塞窗口的变化曲线图,MSS(maximum segment size)为最大报文段,RTT(round-trip time)为往返时延。

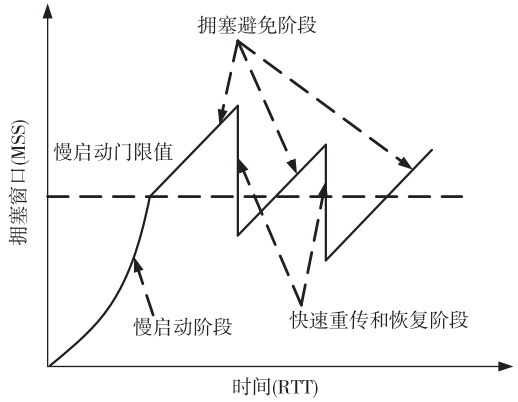

图 1　TCP 拥塞控制算法拥塞窗口变化曲线

1.2　MulWinTCP 算法的提出与算法描述

TCP 之所以只能进行串行处理,是因为 TCP 的发送数据是面向字节流的,每一个字节都有着连续的编号,TCP 通过唯一的滑动窗口来管理数据的发送、确认、重传以及拥塞控制。MulWinTCP 算法将传统 TCP 中的单一滑动窗口划分为多个子滑动窗口,对应多核处理器的多个处理核心,每个处理核心单独维护与之对应的子滑动窗口,从而达到多核并行处理的目的。

1.2.1　MulWinTCP 多滑动窗口

对于实际 TCP 算法,数据的发送都以 MSS 为基准进行处理,每次发送的数据大小为 MSS。而 MulWinTCP,将应用层提交的数据按 MSS 大小划分为数据块,然后将数据块对应到多个滑动窗口,对应关系如图 2 所示(由于在实际处理过程中,都是以 MSS 为基准大小,所以滑动窗口也是以 MSS 为大小进行滑动,为了方便叙述,本文用 MSS 的编号代替实际的 TCP 字节序号)。

如图 2 所示,MulWinTCP 维护着全局窗口,*send_una* 为已发送但未确认的编号,*send_next* 为可以发送的编号,*send_win* 为发送窗口大小,$send\_win = min\{cwnd, recv\_win\}$,cwnd 为拥塞窗口,*recv_win* 为接收方通告的接收窗口。

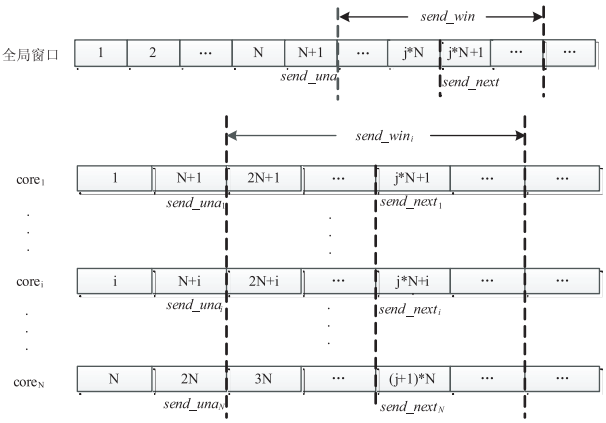

图 2　MulWinTCP 多滑动窗口

每个处理核心维护着与之对应的子滑动窗口,由于将一条 TCP 连接的数据划分到多个处理核心并行处理,所以为了减小系统同步的复杂度,本算法将所有处理核心维护的子滑动窗口对齐,也就是按照图 2 中所示的三条虚线对齐。所有子滑动窗口必须满足以下约束条件:

$$send\_wini = send\_win/N, 1 \leqslant i \leqslant N \tag{1}$$

$$send\_unai = send\_unai - 1 + N, 1 \leqslant i \leqslant N \tag{2}$$

$$send\_nexti = send\_nexti - 1 + N, 1 \leqslant i \leqslant N \tag{3}$$

1.2.2　MulWinTCP 算法的拥塞控制算法

MulWinTCP 的拥塞控制算法和传统 TCP 拥塞控制算法类似,也分为慢启动、拥塞避免、快速恢复和快速重传四个阶段。

(1)MulWinTCP 慢启动算法

当 TCP 连接建立时,MulWinTCP 发送方将拥塞窗口 *cwnd* 初始化为 N 个 MSS,N 为系统的滑动窗口数,相当于所有的子滑动窗口的拥塞窗口 $cwnd_i$ 为一个 MSS。

然后每个处理核心发送一个最大的数据段。如果在定时器过期之前收到数据段的确认 ACK,更新全局窗口,并将所有 $send\_una_i$ 对齐向前滑动。然后将全局拥塞窗口 *cwnd* 增加 N 个 MSS,即所有 $cwnd_i$ 增加一个 MSS。然后每个处理核心发送两个最大报文段,依此类推,使全局拥塞窗口成 N 倍的指数增长。算法伪代码如下:

```
MulWinTCP_slow_start( )
{
    cwnd = N;
    cwnd_i = 1;
    if( ACK received) {
```

```
        if(seq_ack > send_una){
          send_una = seq_ack;
          cwnd_i + = seq_ack/N - send_una_1;/* 满足约束条件式(1) */
          cwnd = ∑cwndi;
          send_una_i = (seq_ack/N) * i;/* 满足约束条件式(2) */
        }
        else if(seq_ack < send_una){
          not need to update slide windows;/* 收到已经确认过的确认号 */
        }
        else if(seq_ack = = send_una){
          seq_cnt + +;
          if(seq_cnt = = 3){     /*出现丢包*/
          retransmit packet;
          cwnd = N;
          cwnd_i = 1;
        }
      }
    }
    else {
      Send packets and update send_next_i;/* 满足约束条件式(3) */
      send_next = send_next_1;
    }
  }
```

(2)MulWinTCP 拥塞避免阶段

当拥塞窗口增长到慢启动门限值(ssthresh)时,MulWinTCP 进入拥塞避免阶段,算法伪代码如下:

```
MulWinTCP_ congestion_ avoid( )
{
  if(ACK received){
    if(seq_ack > send_una){
      send_una = seq_ack;
      cwnd_i + = (seq_ack/N - send_una_1)/cwnd_i;
      cwnd = ∑cwndi;
      send_una_i = (seq_ack/N) * i;
    }
    else if(seq_ack < send_una){
      not need to update slide windows;
    }
    else if(seq_ack = = send_una){
      seq_cnt + +;
      if(seq_cnt = = 3){
        进入快速重传阶段;
      }
    }
  }
  else{
    send packets and update send_next_i;
    send_next = send_next_1;
  }
}
```

在 MulWinTCP 拥塞避免阶段,收到 ACK 确认之后,将 $cwnd_i$ 按下式更新:

$$cwndi \leftarrow cwndi + \frac{1}{cwndi} \tag{4}$$

将式(4)两边叠加,可以得到式

$$\sum cwndi \leftarrow \sum cwndi + \sum \frac{1}{cwndi} \tag{5}$$

变化即可得到式

$$cwnd \leftarrow cwnd + \frac{N^2}{cwnd} \tag{6}$$

由式(6)可知,MulWinTCP 和传统 TCP 算法在拥塞避免阶段,拥塞窗口 $cwnd$ 都是加性增长,但 MulWinTCP 增长更快。

(3)MulWinTCP 快速重传阶段

MulWinTCP 发送端在收到 3 个或以上重复的 ACK 时,即认为数据包已经丢失,并马上重传数据包。本文暂不考虑 SACK 的情况,算法伪代码如下:

```
MulWinTCP_quick_ retransmit( )
{
  /* 假设序号 a*N+b 丢失,则 core_id = b */
  send_una_i = a*N+i(0<i<b);
  send_una_i = (a-1)*N+i(b-1<i<N+1);
  重传序号 a*N+b 以后的数据包,并使 send_una_i 满足公式(3);
  cwnd = cwnd/2;
  cwnd_i = cwnd/N;
  send packets and update send_next_i;
  send_next = send_next_1;
}
```

为了和传统 TCP 算法保持一致,在 MulWinTCP 算法中,出现丢包时,全局拥塞控制窗口变为当前拥塞窗口大小的一般。由于式(1)的限制,为了使所有的窗口相等,我们将全局窗口减半的值平均到所有子滑动窗口。具体的算法如下:

$$cwnd \leftarrow cwnd - \frac{1}{2}cwnd \tag{7}$$

$$cwndi \leftarrow cwndi - \frac{1}{2}cwndi \tag{8}$$

由式(7)可知,在遇到丢包时,MulWinTCP 与传统 TCP 算法的是一致的。

(4)MulWinTCP 快速恢复算法

在快速重传之后,不是执行发送窗口从一个最大报文段开始的慢启动算法,而是执行 MulWinTCP 拥塞避免算法,这就是 MulWinTCP 快速恢复算法。

2 算法性能数学模型分析

TCP 吞吐量的建模方法主要是分析 TCP 拥塞控制的过程,对拥塞控制的各个阶段进行建模,推导出 TCP 吞吐量的表达式。目前已经提出了多种 TCP 吞吐量的理论分析模型,其中最基本的模型是 Mathis 模型[17]和 Padhye 模型[18],其它几个模型基本上都是对这两个模型的改进。所以本文基于 Mathis 模型,对 MulWinTCP 在高速网络下的性能进行数学分析。

Mathis 模型并没有考虑在高速网络中 CPU 被耗尽的因素。高速网络中,会出现下面两种情况:

(1)丢包率低,MulWinTCP 的拥塞窗口可以增大到 CPU 耗尽,约束 MulWinTCP 的吞吐率;

(2)丢包率高,MulWinTCP 的拥塞窗口无法充分的增长,从而不会导致 CPU 耗尽的情况。

下面分别推导这两种情况下 MulWinTCP 的吞吐率模型。

2.1 CPU 为瓶颈时 MulWinTCP 的吞吐量模型

在较低丢包率的高速网络环境下,MulWinTCP 的拥塞窗口可以增加到使 CPU 的处理能力成为系统瓶颈,此时系统的全局拥塞窗口随时间的变化曲线如图 3 所示。

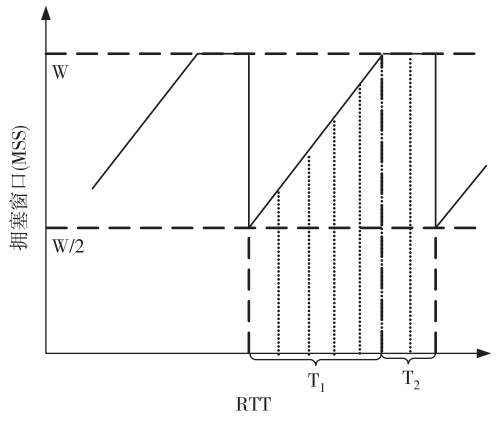

图 3 CPU 为瓶颈时 MulWinTCP 拥塞窗口变化曲线

根据 Mathis 模型可知,此时 MulWinTCP 的吞吐量为图 3 中虚线部分面积和时间 T 的商:

$$Throughput = ((T1 * \frac{W}{2} + \frac{1}{2} * T1 * \frac{W}{2} + T2 * W) * MSS)/((T1 + T2) * RTT) \quad (9)$$

由于每个 RTT 内,MulWinTCP 的拥塞窗口增加 N^2,所以可以得到

$$T1 = \frac{W}{2N^2} \quad (10)$$

同时,根据 Mathis 模型,每发送 $1/p$ 个包,出现一次丢包,所以有

$$\frac{1}{p} = T1 * \frac{W}{2} + \frac{1}{2} * T1 * \frac{W}{2} + T2 * W = (\frac{3}{4}T1 + T2) * W \quad (11)$$

将式(10)与(11)代入式(9)可以得到

$$Throughput = \frac{8N^2W}{8N^2 + pW^2} \frac{MSS}{RTT} \quad (12)$$

假设每个处理器核心的最大处理能力为 w,则 $W = Nw$,代入式(12)得

$$Throughput = \frac{8Nw}{8 + pw^2} \frac{MSS}{RTT} \quad (13)$$

由于丢包率 p 很小,由式(13)就可以看出,MulWinTCP 的吞吐率随着 w 的增加而增加,即随着单核心的处理能力增加而增加。同时,如果不考虑多核同步带来的系统性能损耗,MulWinTCP 的吞吐率是随着多核处理器的处理核心增加而线性增加的。

2.2 丢包率为瓶颈时 MulWinTCP 的吞吐量模型

在较高丢包率的环境下,MulWinTCP 的全局拥塞窗口无法充分增长,丢包率成为了系统性能的瓶颈。此时 MulWinTCP 进入拥塞控制算法的稳定阶段后,全局拥塞窗口的变化曲线如图 4 所示。

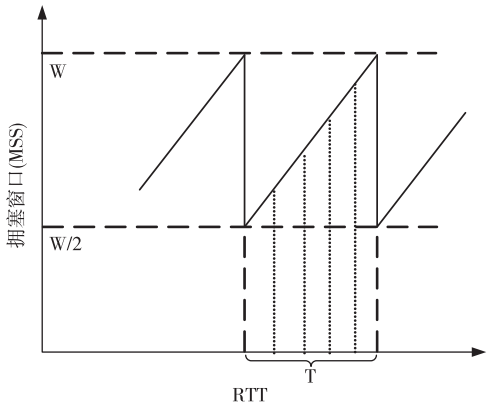

图 4 丢包率为瓶颈时 MulWinTCP 拥塞窗口变化曲线

假设 MulWinTCP 的最大全局发送窗口为 W 个最大报文段 MSS,根据式(7)可知,稳定状态下的最小窗口为 $W/2$ 个最大报文段 MSS。同时可以知道,T 为发送 $1/p$ 个数据包所需要的往返时间。

根据 Mathis 模型可知,MulWinTCP 的吞吐量为

图 4 中虚线所示的梯形面积和所需时间的商：

$$Throughput = ((T * \frac{W}{2} + \frac{1}{2} * T * \frac{W}{2}) * MSS) / (T * RTT) \quad (14)$$

根据式(6)可知，在拥塞避免阶段，每个 RTT 内，MulWinTCP 的拥塞窗口加性增加 N^2 个最大报文段 MSS，所以有

$$T = (W - \frac{W}{2}) / N^2 = \frac{W}{2N^2} \quad (15)$$

而根据假设，MulWinTCP 每发送 $1/p$ 个数据包出现一次丢包，所以有

$$\frac{1}{p} = T * \frac{W}{2} + \frac{1}{2} * T * \frac{W}{2} = \frac{3}{4}TW \quad (16)$$

将式(15)代入式(16)，变化之后可得

$$W = N \sqrt{\frac{8}{3p}} \quad (17)$$

再将式(17)代入式(14)，可得到 MulWinTCP 的吞吐量表达式为

$$Throughput = N \frac{MSS}{RTT} \sqrt{\frac{3}{2p}} \quad (18)$$

由式(18)可知，在高丢包率的高速网络中，如果不考虑多核同步带来的损耗，MulWinTCP 的吞吐率可以达到 TCP 吞吐率性能的线性增长。这是由于在拥塞避免阶段，如式(6)所示，MulWinTCP 窗口增长的速度更快，导致最大拥塞窗口更大。

3 实验与分析

本文在嵌入式多核网络服务器 ATCA-8000R 上编程实现了 MulWinTCP 的多滑动窗口和拥塞控制机制的仿真。ATCA-8000R 的配置参数如表 1 所示。

表 1 ATCA-8000R 配置参数表

处理器名称	Cavium OCTEON 5860
处理器核数	16
处理器架构	MIPS64
处理器频率	750MHz
内存	4GB
网络接口	10GE

在仿真中，假设对端的处理能力足够强，且接收窗口足够大。通过编程控制丢包率 p 和往返时延 RTT，采用多核处理器 spinlock(自旋锁)来保证全局滑动窗口和各个子滑动窗口之间的同步。仿真使用的网络接口为 10Gb/s 的以太网接口。分别仿真在 RTT = 2ms，$p = 10^{-5}$(高速低丢包率局域网)与 RTT = 20ms，$p = 10^{-3}$(高速高丢包率广域网)环境下 MulWinTCP 吞吐率。结果如图 5 和图 6 所示。

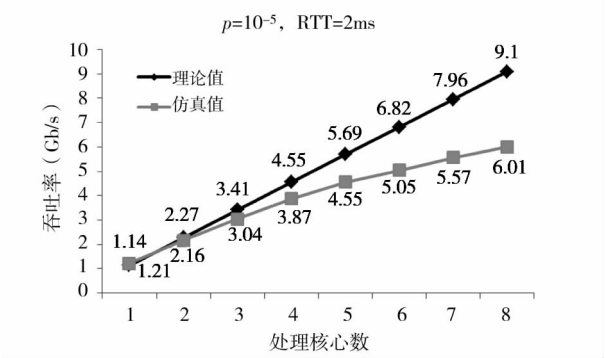

图 5 高速低丢包率局域网仿真结果

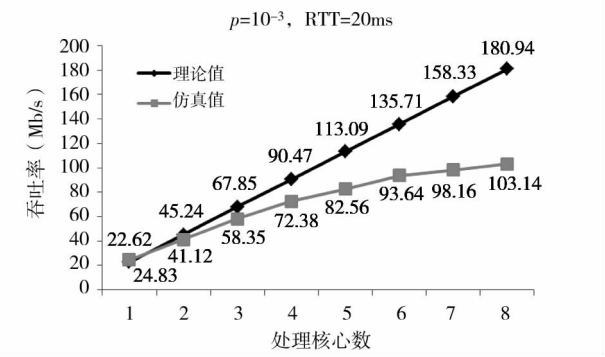

图 6 高速高丢包率广域网仿真结果

由图 5 可知，在高速低丢包率的网络环境下，由于单核心的处理能力已经达到了极限，所以通过多核并行处理可以极大地增加系统的性能。MulWinTCP 的吞吐率的理论值和仿真值在 $N \leqslant 4$ 时是基本吻合的。在 $N > 4$ 时逐步偏离，这是由于在多核处理器中自旋锁的竞争带来的性能损耗随着处理核心的增加而增加。

如图 6 所示，在高丢包率的高速网络中，MulWinTCP 的吞吐率急剧下降，丢包率成了性能的瓶颈。此时，MulWinTCP 的吞吐率的理论值和仿真值在 $N \leqslant 3$ 时是基本吻合的。在 $N > 3$ 时逐步偏离，这是由于在多核处理器中自旋锁的竞争带来的性能损耗随着处理核心的增加而增加。而且由于丢包重传是对全局窗口的访问更加频繁，加锁带来的性能影响更加明显。

实验证明,通过数学模型推导的 MulWinTCP 吞吐率模型基本符合仿真结果,由于多核的同步加锁带来的性能损耗可以通过添加乘积因子 $f(N)$ 实现,$f(N)$ 是处理核心数 N 的减函数,且 $0 < f(N) < 1$。则 MulWinTCP 在 CPU 处理能力成为瓶颈和丢包率成为瓶颈的情况下,吞吐率分别修正为

$$Throughput = f_1(N)\frac{8N^2W}{8N^2 + pW^2}\frac{MSS}{RTT} \quad (19)$$

$$Throughput = f_2(N)N\frac{MSS}{RTT}\sqrt{\frac{3}{2p}} \quad (20)$$

其中,$f_1(N)$ 和 $f_2(N)$ 为多核损耗因子。本文通过 MATLAB 工具进行曲线拟合,给出建议函数

$$f_1(N) = -0.07208 * N^{0.8448} + 1.074 \quad (21)$$

$$f_2(N) = -0.09604 * N^{0.8116} + 1.091 \quad (22)$$

4 结 论

在网络带宽急速增长的环境下,传统 TCP 以及其已有的改进算法的串行处理方式不能充分利用多核处理器系统的并行处理来提高系统的性能。虽然并行 TCP 协议等方法的提出可以通过多条 TCP 连接的方式达到并行处理的目的,但是由于其属于应用层协议,对 TCP 协议不兼容,使其应用受到约束。本文提出的多核并行 TCP 拥塞控制算法 MulWinTCP 将 TCP 的滑动窗口划分为多个子滑动窗口,利用多核处理器对子滑动窗口并行处理,消除了由于 TCP 串行处理方式带来的处理器性能瓶颈。同时 MulWinTCP 保持传输层之上透明,并且兼容 TCP,使其可以有更好的兼容性和扩展性。本文基于经典的 TCP 吞吐率 Mathis 模型,对 MulWinTCP 在高速网络下的吞吐率模型进行了数学分析。分析发现,在高速网络下,特别是在高速低丢包率的网络环境下,MulWinTCP 有着较好的性能。实验仿真验证了在高速网络下 MulWinTCP 的吞吐率性能,实验结果证明了其吞吐率数学模型的正确性。

参考文献

[1] Huston G. Gigabit TCP. *The Internet Protocol Journal*, 2006,9(2):2-26

[2] Xu L, Harfoush K, Rhee I. Binary increase congestion control(BIC) for fast long-distance networks. In: Proceedings of International Conference on Computer Communications. 23rd AnnualJoint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, 2004. 2514-2524

[3] Leith D, Shorten R. H-TCP: TCP for high-speed and long-distance networks. In: Proceedings of the 2nd International Workshop on Protocols for Fast Long-Distance Networks, Argonne, USA, 2004. 9-15

[4] Wei D X, Jin C, Low S H, et al. FAST TCP: motivation, architecture, algorithms, performance. *IEEE/ACM Transactions on Networking*, 2006, 14(6): 1246-1259

[5] Ha S, Rhee I, Xu L. CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS Operating Systems Review*, 2008, 42(5): 64-74

[6] Bullot H, Les Cottrel R, Hughes-Jones R. Evaluation of advanced TCP stacks on fast long-distance production networks. *Journal of Grid Computing*, 2003, 1(4): 345-359

[7] Crowcroft J, Oechslin P. Differentiated end-to-end Internet services using a weighted proportional fair sharing TCP. *ACM SIGCOMM Computer Communication Review*, 1998, 28(3): 53-69

[8] Mulroy P, Appleby S, Nilsson M, et al. The use of MulTCP for the delivery of equitable quality video. In: Proceedings of PV 2009: the 17th International Packet Video Workshop 2009, Seattle, USA, 2009. 1-9

[9] Nabeshima M. Performance evaluation of multcp in high-speed wide area networks. *IEICE transactions on communications*, 2005, 88(1): 392-396

[10] Baldini A, De Carli L, Risso F. Increasing performances of TCP data transfers through multiple parallel connections. In: Proceedings of the 2009 IEEE Symposium on Computers and Communications, Sousse, Tunisaia, 2009. 630-636

[11] Altman E, Barman D, Tuffin B, et al. Parallel TCP sockets: simple model, throughput and validation. In: Proceedings of the 25th IEEE International Conference on Computer Communications, Barcelona, Spain, 2006. 1-12

[12] 马永侠,叶进,宋晓燕等. 并行 TCP 在 NS2 TCP-Linux 中的实现与分析. 桂林电子科技大学学报,2012,32(2):145-149

[13] 彭娜. 并行 TCP 在广域网加速系统中的研究与实现:[硕士学位论文]. 长沙:中南大学,2009. 9-24

[14] Hacker T J, Athey B D, Noble B. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In: Proceedings of the International Parallel and Distributed Processing Symposium, Ft. Auderdale, USA, 2002. 434-443

[15] Stevens W R. TCP/IP Illustrated: the Protocols. Beijing: China Machine Press, 2000. 242-391

[16] Stevens W R. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms (RFC2001). IETF(Internet Engineering Task Force), 1997

[17] Mathis M, Semke J, Mahdavi J, et al. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 1997, 27(3): 67-82

[18] Padhye J, Firoiu V, Towsley D F, et al. Modeling TCP Reno performance: a simple model and its empirical validation. *IEEE/ACM Transactions on Networking*, 2000, 8(2): 133-145

A parallel TCP congestion control algorithm for multi-core systems in a high-speed network

Zha Qiwen[*,**], Zhang Wu[*], Zeng Xuewen[*], Guo Xiuyan[*]

([*] National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100190)

Abstract

To solve the TCP performance bottleneck caused by the serial processing mode of multi-core processor systems in a high speed network, a MulWinTCP, a parallel TCP congestion control algorithm based on the slide window mechanism and the congestion control of TCP, is proposed. The MulWinTCP divides the global slide window into multiple slide sub-windows, and parallely processes them by using multi-core processors to eliminate the performance bottleneck brought by the serial processing mode of TCP. The MulWinTCP is transparent to the network applications, and can be compatible completely with TCP. Through the analysis of the mathematical model, the throughput model of MulWinTCP is given based on the Mathis Model. The simulation experiment proves the validity of the algorithm and the correctness of the model.

Key words: multi-core processor, TCP, congestion control algorithm, slide window, high speed network