第 35 卷 第 6 期 计 算 机 工 程 2009 年 3 月
Vol.35 No.6 Computer Engineering March 2009
·网络与通信· 文章编号：1000—3428(2009)06—0113—03 文献标识码：A 中图分类号：TP393.08

区分服务网络中 SCTP 协议的改进

刘 辉，唐 扬
(国家数字交换系统工程技术研究中心，郑州 450002)

摘 要：针对双窗口算法难以判断包丢弃类型的问题，提出一种新的包丢弃类型判断方法，对 SCTP 协议的确认块进行扩展，从确认块中获取包丢弃类型信息。在 NS2 上进行仿真实验，实验结果表明，改进后的 SCTP 协议在吞吐率方面有显著提高。
关键词：流控制传输协议；确保服务；双窗口算法；吞吐率

Improvement of SCTP in Differentiated Services Networks

LIU Hui, TANG Yang
(National Digital Switching System Engineering & Technological R&D Center, Zhengzhou 450002)

【Abstract】 Aiming at the problem that two-window algorithm can not judge the dropped packets marked type, a novel judgement method for dropped packets marked type is proposed, which extends the SACK chunk of SCTP, and the dropped packets' marked type information is obtained from the extended SACK chunks. Simulation experiments are made in NS2, and the results indicate the improved SCTP achieves more throughput.
【Key words】 Stream Control Transmission Protocol(SCTP); assured services; two-window algorithm; throughput

1 概述

SCTP 协议是种新型传输协议，可以完成所有 TCP 所能完成的业务传输，并提供一些新功能。从承载的业务类型来看，SCTP 协议传输的业务大多需要一定的服务质量保证[1]。目前，互联网最可行的服务质量解决方案是 DiffServ 体系架构，SCTP 能否与区分服务网络有效地结合是个值得研究的问题。以往研究表明，TCP 在应用于区分服务的 AF 服务时其性能并不理想。由于 SCTP 流的行为特性与 TCP 非常相似，因此从理论上说，SCTP 协议也面临同样问题。

本文结合 SCTP 的特点，将双窗口算法应用于 SCTP，并通过 NS2 进行仿真实验，实验结果表明，使用该方法对 SCTP 进行改进，可以有效提高 SCTP 在使用 AS 服务时的吞吐率。

2 TCP 双窗口算法分析

2.1 双窗口算法原理

区分服务网络中的 AS 服务提供机制根据用户的订购带宽要求对包进行测量并在包头的 DSCP 字段写入区分 OUT 包和 IN 包的标记，OUT 包表示超过定购带宽到达的包，IN 包表示在定购带宽范围以内到达的包。中间路由节点采用 RIO 队列管理算法进行调度，OUT 包的丢弃概率远大于 IN 包的丢弃概率，当带宽资源不足时，OUT 包首先被丢弃以保护 IN 包，进而保护用户的订购带宽[2]。

文献[3-4]对 TCP 在使用 AS 服务时的性能进行研究，研究表明，TCP 流的实际吞吐率在多数情况达不到其定购带宽要求，为解决这一问题，本文给出一种双窗口的 TCP 拥塞控制算法，将用户的预定带宽参数引入到 TCP 的拥塞控制机制中。算法如下：

```
当检测到有包丢失时：
if (OUT packet loss) {
rwnd = rtt×contract_bandwidth;
if (rwnd < cwnd) {
ewnd = cwnd – rwnd;
cwnd = rwnd + ewnd / 2;
}}
else{                                    // IN packet loss
cwnd = cwnd / 2;
}
```

算法 1(双窗口算法)

算法中，rwnd 表示定购带宽的门限，rwnd 由 RTT 乘以用户的定购带宽 contract_bandwidth 获得。通过 rwnd，发送端可以大致估计出协议当前的发送速率。当 rwnd<cwnd 时意味着发送速率超过用户的订购带宽，否则，意味着发送速率还没有达到订购带宽。当拥塞窗口大于 rwnd 时，拥塞窗口可看作 2 个部分，一部分为定购带宽窗口 rwnd，另一部分为额外带宽窗口，用 ewnd 表示。当发现 OUT 包丢失时，双窗口 TCP 仅使 ewnd 减半。当 IN 包丢失时，才使整个窗口 cwnd 减半。因为 OUT 包的丢失意味着额外带宽减少，所以用户的订购带宽没有受到影响。IN 包的丢失，意味着网络拥塞较为严重，没有足够的定购带宽满足用户的定购需求。

2.2 双窗口算法面临的问题

实现双窗口算法时的关键问题是如何使发送端知道丢失的包是 IN 包还是 OUT 包。针对此问题，文献[3]给出 2 种方法：(1)让丢弃包的路由器向发送端发回丢包原因；(2)在传输协议中加入标记器，标记工作由发送端来完成。但这 2 种方法都存在问题。

第(1)种方法面临以下困难：(1)会使 AS 服务实现机制复杂化。AS 服务的一个特色就是实现简单，实现时边界路由器使用双色标记策略，中间路由器使用 RIO 队列调度策略。如果加入向发送端发回丢包信息的功能，需要引入新的机制，

基金项目：国家“863”计划基金资助项目(2005AA121210)
作者简介：刘 辉(1981－)，男，硕士研究生，主研方向：网络通信协议；唐 扬，硕士研究生
收稿日期：2008-09-25 **E-mail：**yihui_mail@hotmail.com

使 AS 服务实现复杂化。(2)加重路由器的负担。路由器作为中间节点，要对来自许多不同源地址的数据进行转发，本身负担已经很重，不应再加重其开销。(3)要对传输协议进行较多的修改。因为即使路由器能发回丢包类型信息，所以也需要协议的发送端有相应的机制对其进行接收处理。

第(2)种方法实现起来相对简单，但存在以下问题：(1)与区分服务网络的结构不相符。区分服务网络通常由边界路由器对进入的流做标记，标记器一般位于边界路由器中，而不应该在 TCP 的发送端。(2)位于协议发送端的标识算法与位于路由器中的标识算法不一定能同步工作[5]。发送端的标记器是根据自己的发送速率对包进行测量标记。在路由器上的标记器，则是根据当时整个到来的网络流量进行测量标记。在协议发送端加标记器的方法只适合仿真环境，不适合现实网络。

3 SCTP 上双窗口算法的实现

本文提出一种在 SCTP 中实现双窗口算法的方法，主要利用 SCTP 的选择确认机制获取丢包类型信息。

3.1 包丢弃类型的判断原理

在 AS 服务中，当业务流的速度没有达到定购速率时，通过标记器的 IP 包都被标记为 IN 包，超过定购速率后，被标记为 OUT 包。通过在 NS2 上跟踪包的标记情况，发现经历区分服务网络的 IP 包，被标记的情况为连续的 IN 包或连续的 OUT 包。因此，根据时间空间局部性原理，相邻比较近的 IP 包在网络中会经历相似的网络状态和标记状态，即当一个包被标记为 IN 包或 OUT 包时，在很大概率上，它前后相邻的包也都标记为 IN 包或 OUT 包。根据这一特征，当某些包被 RIO 队列管理机制丢弃时，可以根据它邻近的没有被丢弃的包中的标记信息来判断丢失包的丢弃类型，即当一个包被丢弃时，如果它邻近的包是 IN 包，就可以认为该丢弃的包是 IN 包，如果是 OUT 包可以认为该丢弃的包是 OUT 包。所以，只要能获得丢弃包邻近包的标记信息，就可以判断出丢弃包的标记信息。

3.2 对 SCTP 确认块的修改

本文对 SCTP 确认块进行扩展，使其能捎回丢弃包的邻近包的标记信息。SCTP 使用选择确认机制，并有独立的确认块来返回确认信息，扩展后的 SACK 块如图 1 所示。

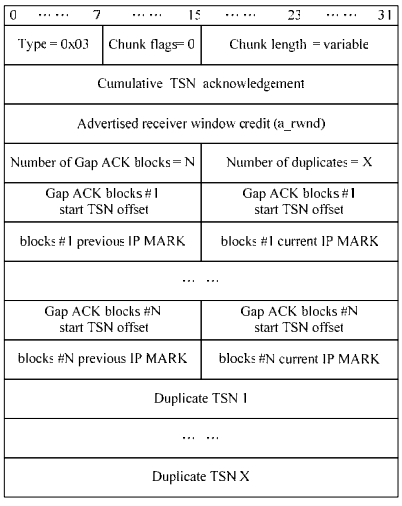


图 1 修改后的确认块

在每个 GAP 后加入 2 个域 previous IP MARK 和 current IP MARK。previous IP MARK 表示该 GAP 指示的数据块所在 IP 包的前一个到达接收端的 IP 包的标记值，current IP MARK 表示当前 GAP 中第 1 个数据块所在 IP 包的标记值。这样，发送端在收到一个带有 GAP 的 SACK 后，会知道有包丢失，并根据 GAP 中当前分组的的 IP 标记和它前一个分组的 IP 标记，可以判断出丢失包的标记情况。接收端从 IP 包中提取 SCTP 分组时，需要记录 IP 包的 DSCP 标记值，在构造 SACK 块的 GAP 时，写入 current IP MARK, previous IP MARK 域。

3.3 包丢弃类型的判断算法

发送端根据收到的 SACK 块判断哪些数据块丢失，并根据下面的算法来判断丢失的数据块所在的 IP 包在网络中的标记情况。

(1)lowtsn = SACK.CTSN; tempmark = BE;

(2)判断 SACK 中是否有 GAP，有转(1)；否则退出。

(3)从 SACK 中取出一个 GAP, hightsn = SACK.CTSN + GAP.startTSNoffset。

(4)如果 current IP MARK=IN 且 previous IP MARK=IN，则 tempmark = IN。

(5) 如果 current IP MARK=OUT 且 previous IP MARK=OUT，则 tempmark = OUT。

(6) 如果 current IP MARK=OUT 且 previous IP MARK=IN，则 tempmark = OUT。

(7) 如果 current IP MARK=IN 且 previous IP MARK=OUT，tempmark = OUT。

(8)设置发送队列中(lowtsn, hightsn)范围内的数据块的丢弃类型标记为 tempmark，并放入到重传队列。

(9)判断 SACK 中是否还有 GAP，如果有令 lowtsn = hightsn; hightsn = SACK.CTSN + GAP.startTSNoffset 转 2;否则结束。

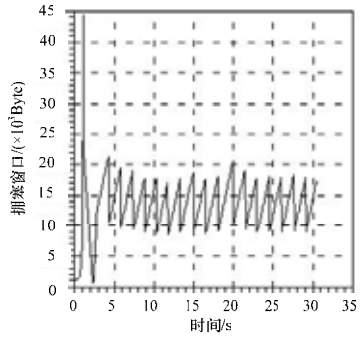
算法 2(包丢弃类型判断算法)

算法中的 SACK.CTSN 为 SACK 块中的累积确认值 cumulative TSN acknowledgement。tempmark 用于临时存放标记值。SCTP 以数据块为单位设定传输序号，确认重传都以数据块为单位。多个数据块在传输时被封装到一个 SCTP 分组，然后再封装到 IP 包中，因此，需要将 IP 包的标记情况映射到其中包含的每个数据块。算法中的(temptsn, hightsn)用来表示 TSN 号在此范围内的数据块是丢失 IP 包中所含的数据块。

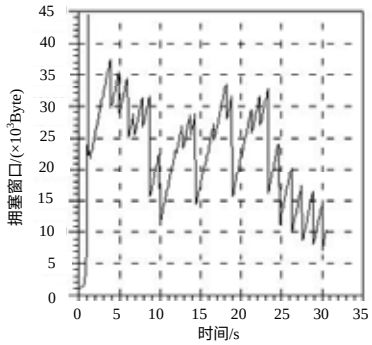
3.4 NS2 中双窗口算法的实现

在 NS2.30 中 SCTP 协议的基础上，实现一个改进的 SCTP 协议。对协议中的 SctpGapAckBlock_S{}结构进行扩展，加入 current IP MARK, previous IP MARK 域。对构造块的 GenChunk()函数进行修改，使其在构造确认块时，能写入 current IP MARK, previous IP MARK 信息。在发送端加入一个新的函数 void SctpAgent::ChunkIPMark(u_char *ucpSack Chunk)用来实现算法 2。

对于拥塞控制部分仍采用算法 1，在实现时对 NS2 中 SCTP 协议的 void SctpAgent::FastRtx()函数进行修改。通过多次 gdb 跟踪调试可以发现，采用本文方法可以准确地对丢包类型做出判断，并执行相应的双窗口控制算法。图 2 显示了协议修改前后 cwnd 的变化情况。从图 2(a)中可以看出，在每次发生丢包时，修改前协议的窗口都减小为原来的一半；从图 2(b)可以看出，改进后的协议可以区分出是 OUT 包丢失还是 IN 包丢失，当 OUT 包丢失时，窗口仅减小为额外带宽窗口的一半，当 IN 包丢失时，窗口才减小为原来的一半。



(a)改进前 cwnd 的变化



(b)改进后 cwnd 的变化

图 2　协议修改前后拥塞窗口变化

4　仿真实验

4.1　实验方案

通过实验对改进前后协议的吞吐率性能进行对比。实验拓扑如图 3 所示。

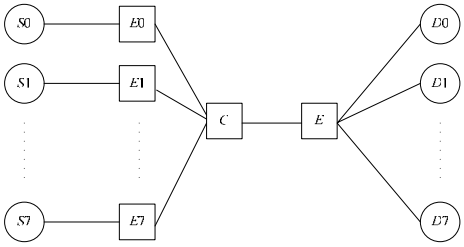
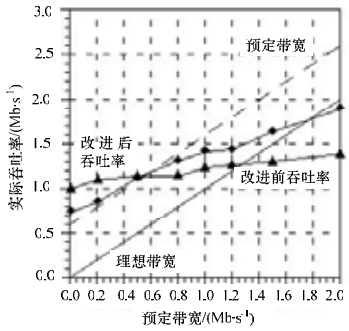


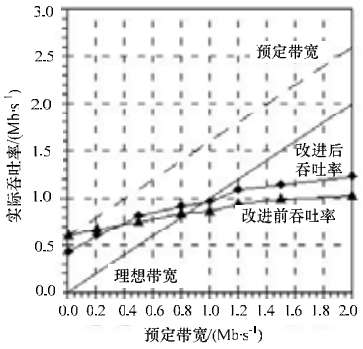
图 3　仿真网络拓扑

Si~*Di*(*i*=0,1,⋯,7)为 SCTP 协议的发送端和接收端，模拟 8 条 SCTP 流，应用层采用 FTP，每条链路的总时延均为 40 ms。节点 *Ei*(i=0,1,⋯,7), *C*, *E* 组成一个简单的区分服务网络。其中，*Ei*(i=0,1,⋯,7)；*E* 是边界路由器。使用 TSW2CM 标记策略，对进入区分服务网络的 IP 包进行测速、标记。节点 *C* 为区分服务网络内部的核心路由器，采用 RIO 队列调度机制来实现 AF 服务。参数配制为：maxth_in:100, minth_in:50，IN 包丢弃概率为 0.02，maxth_out:100, minth_out:50, OUT 包丢弃概率为 0.5。

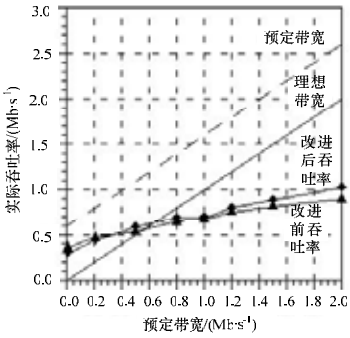
各 SCTP 流的定购带宽总和设为 7.2 Mb/s，共进行 3 组实验，在每组实验中，将图 3 中节点 *C*, *E* 之间的瓶颈链路带宽分别设置为：12 Mb/s, 8 Mb/s, 6 Mb/s，以表示预定带宽总和占整个瓶颈链路带宽的 60%, 90%, 120% 3 种情况，以此来观察预定带宽总量在占用瓶颈带宽 3 种比例情况下的 SCTP 性能。让基于 SCTP 的 8 条 FTP 流同时传输，每个 SCTP 分组的大小为 500 Byte，每次实验运行 30 s，每组实验运行 8 次。实验结果如图 4 所示，图中理想带宽是在假定总预定带宽之外的剩余带宽被各流平均分配的情况下计算得到的。其中，图 4(a)表示的是预定带宽占瓶颈带宽 60%时的情况，图 4(b)是 90%时的情况，图 4(c)是 120%时的情况。



(a)预定带宽占总带宽 60%时吞吐率



(b)预定带宽占总带宽 90%时吞吐率



(c)预定带宽占总带宽 120%时吞吐率

图 4　改进前后协议吞吐率对比

4.2　结果分析

从实验中可以得出以下结论：

(1)多数情况下标准 SCTP 流的实际吞吐率难以达到理想带宽。原因是使用 AIMD 的拥塞避免机制时，一旦检测到有数据丢失，协议就认为是网络拥塞，拥塞避免机制就通过加倍减小窗口来避免拥塞的发生。这样就会导致发送速率减少到预定带宽以下，从而降低实际吞吐率。

(2)同等条件下预定带宽小的 SCTP 流比预定带宽大的 SCTP 流，更接近其理想带宽和预定带宽。原因是预定带宽小的流，其窗口增大到预定带宽要求用时相对短，从而有更多机会去竞争剩余带宽。使用改进后的协议会使这种不公平性有所缓解。

(3)改进后的 SCTP 流的实际吞吐率比标准 SCTP 流的实际吞吐率有明显提高。改进后的协议可以有效地判断出是哪种类型的数据块丢失，进而可以针对丢弃原因做出正确的窗口减小决策。

(4)改进后各 SCTP 流的总吞吐率比改进前的总吞吐率增加，相应的对带宽的利用率也有提高。　(下转第 118 页)