基于SCTP多宿特点的多路径同时传输研究

朱桂勇，吴庆波

（国防科学技术大学 计算机学院，湖南 长沙 410073）

摘　要：多宿是SCTP协议的一个重要特点，利用这个特点，发送端可以将数据通过不同的路径发送给接收端。虽然SCTP提供了对多宿的支持，但仅仅是为了提高关联的可靠性，当前SCTP协议只允许同时使用一条路径传输数据；如果能够同时利用多条路径传输数据，必能极大提升关联的吞吐量。分析了基于SCTP多宿特点的多路径同时传输关键技术，这些关键技术包括快重传触发技术、拥塞控制窗口增长技术、延迟应答技术、接收端缓冲区阻塞减轻技术以及重传时的路径选择技术。

关键词：流控制传输协议；多宿；负载均衡；多路径同时传输；CMT

中图分类号：TP393　　**文献标识码**：A　　**文章编号**：1673－629X(2007)03－0005－05

Research on Multipath Transfer Using SCTP Multihoming

ZHU Gui-yong, WU Qing-bo

(Department of Computer Sciences, National University of Defense Technology, Changsha 410073, China)

Abstract: Multihoming is one of the most important features in SCTP, which can be used by the data sender to send data to a receiver through different paths. Although SCTP provides support for multihoming, the basic reason for such a provision was to improve reliability of associations, simultaneous transfer of new data to multiple paths is currently not allowed in SCTP. We can gain significant throughput improvement if simultaneously transfer new data across multiple paths to the receiver. In this article, the multipath transfer using SCTP multihoming is discussed and the techniques are analyzed. These techniques include fast retransmission starting technique, CWND growth technique, delayed ACK technique, receive buffer block reducing technique and retransmission policies.

Key words: SCTP; multihoming; load balancing; multipath transfer; CMT

0 引　言

多宿是指一个主机拥有多个网络接口，这些网络接口可以连接到一个或多个网络服务提供商。如果一个主机拥有多个IP地址，通常就称这个主机是多宿的。目前广泛使用的传输层协议TCP和UDP都不支持多宿，TCP只允许连接的一端关联一个IP地址。流控制传输协议(SCTP)是IETF推荐的传输层协议，它的一个重要特点就是支持多宿。SCTP的多宿允许关联(SCTP中表示连接的术语)的一端绑定多个IP地址，这种绑定允许发送者将数据通过不同路径发送到接收端，从而提高了关联的可靠性。但当前SCTP协议只允许同时使用其中一条路径进行数据传输。

随着多宿的大量出现以及SCTP的广泛应用，研究如何利用SCTP关联中的多条路径同时进行数据传输具有非常现实的意义。如果关联中的多条路径能够同时进行数据传输，将会大大提高系统的吞吐量，从而极大提高网络资源利用率。

1 SCTP及其多宿特点

1.1 SCTP简介

SCTP[1,2]是由SIGTRAN工作组制定的，最初的目的是用来在IP网络上传输SS7信令。但SCTP的许多特点也使它非常适合于其它的应用。与TCP类似，SCTP也是面向连接的，能够提供可靠和有序的数据传输。在SCTP中，TCP的连接被引申为关联(association)，其概念要比TCP中的连接概念含义更广。SCTP在建立关联的时候会向对方提供一个端口号和一个IP地址列表，这样每个关联都由两个端口号和两个IP地址列表来组成。

如图1所示，一个SCTP报文由一个公共报文头和多个块(chunk)组成。SCTP的块分成控制块和数据块两种，控制块用于关联的建立和关闭、路径可达性测

收稿日期：2006－05－22

基金项目：国家863软件重大专项(2004AA1Z2240)

作者简介：朱桂勇(1981－)，男，江苏大丰人，硕士研究生，研究方向为操作系统、网络；吴庆波，硕士生导师，研究员，研究方向为操作系统和嵌入式系统软件。

试以及报文确认;数据块用于传输上层协议数据,在一个 SCTP 报文中可以绑定多个数据块,即 SCTP 报文中可以存在多个数据块。下面将重点介绍数据块,选择性应答(SACK,Selective ACK)控制块将在的拥塞控制机制一节中介绍。

源端口		目标端口
验证标签		
校验和		
类型	标识	长度
一个或多个块		

图 1 SCTP 的通用报文格式

如图 2 所示,在 SCTP 的数据块中存在三个序列号:传输序列号、流标识符、流序列号。SCTP 除了多宿特点外,还有一个很重要的特点就是多流。流是两个 SCTP 端点之间建立的一个单向逻辑通道。SCTP 使用流标识符(Stream Identifier)来区别这些不同的逻辑通道。在 SCTP 中,一个流中的数据有序递交不影响其它流中的数据有序递交,这有效避免了队列头阻塞问题(HOL)。流中的有序递交是通过流序列号(Stream Sequence Num, SSN)来保证的,而可靠性传输是通过传输序列号(Transmission Sequence Number, TSN)保证的。SCTP 将数据可靠性传输和数据有序传输分开,就是依靠 SSN 和 TSN 两种不同的序列号。

类型=0×00	标识	数据块长度
传输序列号		
流标识符		流序列号
协议标识符		
用户数据		

图 2 数据 chunk 格式

1.2 SCTP 多宿特点

SCTP 的一个重要特点就是支持多宿,如果一个关联的两端都是多宿的,那么这个关联就存在多条路径可供传输数据。当前 SCTP 只允许使用其中一条路径来传输数据,这条路径称为主路径,另一条路径称为备选路径。缺省情况下,数据发送端总是使用主路径发送数据;只有在主路径失效的情况下,才会使用备选路径传输数据。SCTP 的多宿特点极大地提高了 SCTP 会话的潜在存活能力。

在图 3 中,主机 A 有本地接口 A1 和 A2,主机 B 也有本地接口 B1 和 B2,从而 A 和 B 就会有两条路径:路径 1(A1,B1)以及路径 2(A2,B2)。

2 SCTP 的拥塞控制机制

SCTP 的拥塞控制算法是基于 RFC2581[3]的。与 TCP 类似,SCTP 使用接收端的通知窗口(Receiver advertised window,rwnd)、发送端的拥塞控制窗口(Congestion control window,cwnd)、发送端的慢启动阈值(Slow-start threshold,ssthresh)三个控制变量来控制传输速率。因为每一个目的地址都意味着不同的传输路径,而不同的路径具有不同的状态,SCTP 必须对每一个目的地址保留一组拥塞控制变量。

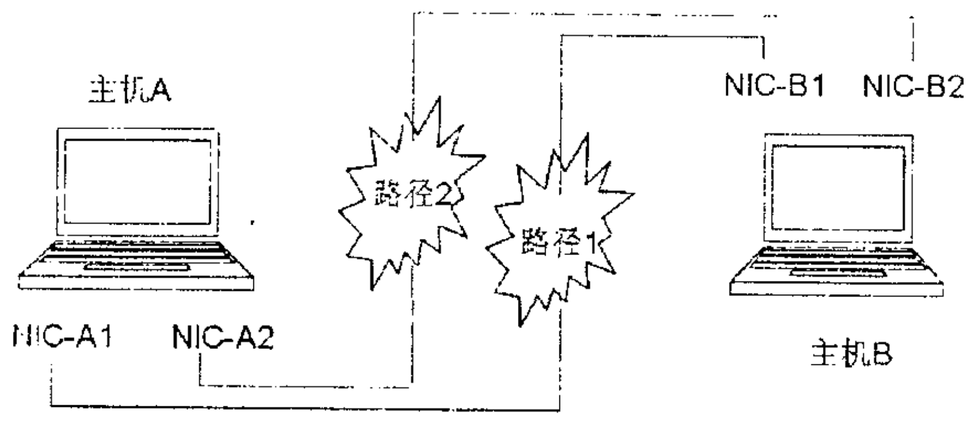

图 3 多宿

接收端使用 SACK 控制块(如图 4 所示)来对接收到的数据进行应答,同 TCP 一样,SCTP 也使用延迟应答机制。SACK 控制块中有一个称之为“累积 TSN”(Cumulative TSN ACK)值,这个值表示接收端已成功接收到的最大连续 TSN 值,只有小于累积 TSN 值的 TSN 才可认为已经成功发送。SACK 控制块中还有一个称为“间隔确认”(Gap ACK)的字段,它表示接收到了不连续的 TSN 值。如果发送端第四次接收到暗示某个数据块丢失的 SACK 控制块,那么发送端就会认为这个数据块已经丢失,并使用快速重传机制来重传这个数据块。

和 TCP 类似,SCTP 有慢启动和拥塞避免两种模式,SCTP 处于哪种模式是由上面介绍的三个控制变量决定的。因为关联的一端到另一端可能存在多个路径,不同的路径将会有不同的控制变量值;当主路径处于拥塞避免模式时,备选路径可能还处于慢启动模式。当 cwnd 超过 ssthresh 时,路径就会从慢启动模式进入拥塞避免模式。在慢启动模式中,cwnd 的增长将会比较快,当接收到含有新的累积 TSN 值的 SACK 控制块时,cwnd 通常会增加一个 MTU(最大传输路径值,以字节为单位)的值。当处于拥塞避免模式中时,cwnd 的增长比较慢,大约每个 RTT(Round Trip Time)时间里增加一个 MTU 值。当重传发生时,ssthresh 值将会急剧减少,而 cwnd 将会被重置。

SCTP 将路径分为主路径和备份路径,且任何时刻都使用主路径传输数据,只有在主路径失效或重传发生时,才使用备份路径传输数据。规范中的拥塞控制算法都是在这种数据传输方式下制定的,并没有考虑多路径同时传输的情况。下面将会看到,在多路径同时传输时,使用这些算法会产生一些问题。

类型=3	标识=0	SACK 控制块长度
累积 TSN		
接收端通知窗口		
间隔确认块总数=N	重复的 TSN 总数=M	
间隔确认块 1 起始 TSN	间隔确认块 1 结束 TSN	
……		
间隔确认块 N 起始 TSN	间隔确认块 N 结束 TSN	
第 1 个重复的 TSN		
……		
第 M 个重复的 TSN		

图 4　SACK 控制块

3　基于 SCTP 的多路径同时传输及其关键技术

3.1　基于 SCTP 的多路径同时传输

利用 SCTP 的多宿特点,可以在传输层进行多路径同时传输和负载分配。虽然可以通过多个 TCP 连接在应用层实现多路径同时传输,可是在应用层实现多路径同时传输具有如下缺点:

1) 应用层在将数据通过不同 socket 连接传输的时候,并不知道底层链路的差别。这样接收端在重新组合数据的时候,很容易出现类似于队列头(head of line)阻塞的问题,也就是说一个慢速 socket 连接会对整个虚拟连接的吞吐量造成严重影响。

2) 在应用层进行多路径同时传输还增加了应用程序编程的复杂性,因为此时应用层必须负责将数据传输分配到不同的 socket 上,并且还要负责数据的重排序。

3) 在应用层进行多路径同时传输还会增加传输层与应用层之间接口的复杂性,因为要不停地在传输层与应用层之间交换各种信息,并且还影响了程序的可移植性。

传输层是第一个端到端的层,它拥有详细的端到端路径信息,因此在传输层实现多路径同时传输可以做到更好的负载均衡,可以更好地利用网络资源;而且传输层的实现对上层应用透明,有利于程序移植。SCTP 有着更广泛的应用场合,它既适合静态主机也适合移动主机,且可用于任何类型的网络,包括无线网络。作为 IETF 的标准传输协议,SCTP 正得到越来越多的应用。

根据数据发送时的路径选择策略,可以将基于 SCTP 的多路径同时传输技术分成如下几种:

(1) 随机选择策略:随机选择其中一条路径。

(2) 最低报文丢失率选择策略:选择具有最低报文丢失率的路径。

(3) 最大带宽选择策略:选择具有最大带宽的路径。

随机选择策略不需要考虑底层路径的差别,是最简单的策略。最低报文丢失率选择策略倾向于使用报文丢失率低的路径,这种选择策略的目的是为了减少报文丢失,从而减少重传次数。因为路径的状态在不断变化,它的丢失率是很难准确预测的,实际上一般使用 cwnd,ssthresh 的值作为路径选择的依据。最大带宽选择策略倾向于使用高带宽的路径,它能够有效减轻接收端的报文乱序问题。在实际路径选择时,也可以综合考虑报文丢失率和带宽。

3.2　快重传触发技术

由于不同的路径具有不同的传输延时和带宽,接收端将会出现严重的乱序现象。在图 5 中,如果两条路径的传输延时差别比较大时,主机 B 接收到 TSN 序列号是 1,2,3,4,5,6…顺序的可能性是非常小的。当在接收端出现 TSN 不连续时,接收端会使用 SACK 控制块向发送端进行间隔报告(gap reports),当一个 TSN 号在间隔报告中第 4 次出现时,发送端将使用快速重传机制来发送这个 TSN,但是此时数据并没有丢失。

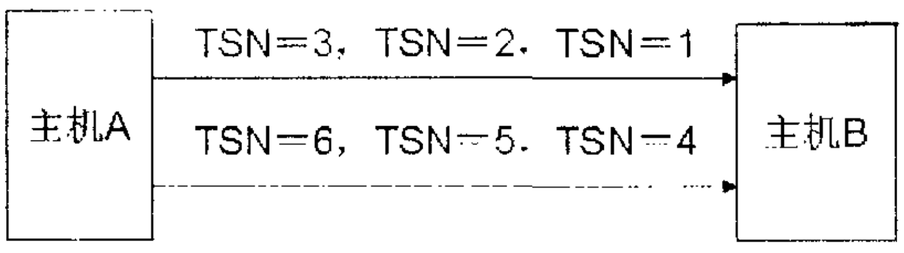

图 5　多路径同时传输

由以上分析可知,在使用多路径同时传输时,会触发不必要的快速重传机制。而快速重传会导致如下两个结果:

1)发送者会减少对应路径上的拥塞控制窗口值(一般会变成原来的一半)。

2)由于发送端无法区分接收到的 SACK 控制块中的累积 TSN 针对的是原来的 TSN 还是重传的 TSN,原 TSN 的确认报文会使发送端认为是对重传报文的确认,这将导致重传路径上对应的 cwnd 出现过快增长。

由上面的分析知,不必要的快速重传会导致不必要的拥塞控制窗口变化,从而严重影响了数据传输效率。这种不必要的变化主要是因为当前 SCTP 对 SACK 控制块的理解。SCTP 认为 SACK 控制块中的间隔报告暗含着数据丢失,这在只有一条路径用于传输数据的情况下是正确的。在多路径同时传输的情况

下,由于接收端会出现大量的乱序现象,这将导致大量的间隔报告,但是这些间隔报告并不意味着报文丢失。因此需要不同的快重传触发技术,它应将 SACK 控制块中的消息看成是接收端目前接收到的 TSN 的精确描述。

分裂快速重传[4](Split Fast Retransmit,SFR)在发送端为每条路径维持一个"虚拟队列",算法然后将这些消息与接收到的 SACK 控制块结合起来判断一个 TSN 是否丢失。SFR 算法能够有效减少不必要的快重传触发,而且 SFR 算法的一个优点是无需对数据接收端做任何修改。

3.3 拥塞控制窗口增长技术

在第 2 节中的 SCTP 拥塞控制机制中可知,发送端只有在接收到一个新的累积 ACK 时,才会增加拥塞控制窗口值。换句话说,当发送端接收到一个没有变化的累积 ACK 时,发送端是不会修改对应路径上的拥塞控制窗口值的。这现象的产生,同样是因为 SCTP 的当前拥塞控制算法认为累积 ACK 没有增加就意味着报文丢失,进而意味着网络出现拥塞了。

因为多路径同时传输时,接收端会观察到大量的乱序现象,大量的 SACK 控制块只包含间隔报告而没有新的累积 ACK。发送端只有在接收到新的累积 ACK 时才会改变 cwnd 值,但是这种改变仅仅反映了接收端刚刚接收到的 TSN,以前通过间隔报告确认的 TSN 没有对 cwnd 的改变做出贡献。即使接收端在同一个路径上接收到的 TSN 号是有序的,对应路径上的 cwnd 也有可能得不到及时更新。

这种 cwnd 偏低的增长速度是因为当前累积 ACK 是基于整个关联的,而不是基于路径的。TCP 和当前的 SCTP 在任何时候都只使用一条路径来传输数据,因而当前的 cwnd 管理方式可以很有效的工作。而由于现在是多路径同时传输,这种基于关联的 cwnd 增长方式必须改变,必须跟踪每条路径上接收到的最大有序 TSN 值。

在文献[4]中,作者提出了一种新的 cwnd 增长算法,称为 CWND Update for CMT(CUC)。这个算法能够根据发送端和 SACK 控制块中的信息判断每条路径上的有序传输。使用 CUC 算法,即使没有接收到新的累积 ACK,发送端也能准确更新对应路径上的 cwnd 值。

3.4 延迟应答技术

SCTP 规定报文接收端应该使用延迟 ACK 算法[3]来应答数据,通常每隔两个报文发送一个 SACK 控制块,这有助于减少网络上的 ACK 流量。但是又规定只有在 TSN 连续的情况下,才可以使用这种延迟确认机制,那些不连续的 TSN 必须立即得到确认。由于多路径同时传输很容易造成接收端接收到大量不连续的 TSN,从而造成接收端经常不使用延时应答机制,这就产生了大量的 ACK 流量。

为了减少 ACK 流量,接收端可以简单地忽略上面提到的规则,也就是说,多路径同时传输时,接收端并不立即对那些不连续的 TSN 进行确认,而是统一使用延时确认机制。但是这又引入了一个新的问题。

当前 SCTP 的设计认为在正常情况下 TSN 是会有序到达的,如果接收端出现乱序,就意味着出现数据丢失,所以这时候接收端应该立即发送 SACK 控制块。如果发送端第 4 次接收到暗示某个 TSN 丢失的 SACK 控制块,发送端就会使用快速重传算法来发送这个 TSN,也就是说发送端有一个值为 4 的乱序阈值。由于在多路径同时传输时,接收端不能正确区分乱序或报文丢失引起的 TSN 不连续,上面的修改将造成接收端在出现报文丢失的情况下仍然使用延迟确认机制。也就是说当报文丢失时,发送端会在接收端收到 8 个报文后触发快速重传机制,这相当于发送端的乱序阈值是 8!

为解决这个问题,需要接收端在 SACK 控制块中包含更多的信息,这些信息可以帮助发送端推测每个不连续 TSN 的丢失报告次数。"多路径传输延时确认"[4]算法利用 SACK 控制块中的标识(flags)字段记录从最近一次发送出去的 SACK 控制块起到当前发送 SACK 控制块时所接收到的报文数目,接收端然后利用这个信息来判断不连续 TSN 的丢失次数。

3.5 接收端缓冲区阻塞减轻技术

每个 SCTP 关联的接收端都会维持一个接收缓冲区用于数据缓存,这个缓冲区由多条路径共享。当网络出现报文丢失时,接收端就需要等待发送端重传报文,而此时,由于流中的序列号不连续,接收端不能将这些数据传递给应用层。这些不连续的 TSN 必须留在接收端的缓存中,直到接收到重传的数据,因而会造成接收端的缓冲区阻塞。

前面说过,接收端的接收缓冲区是基于关联的,由关联中的多个路径共享。当这些路径的传输质量不同时(即不同的报文丢失率),由于质量差的路径会出现比较多的报文丢失,从而会占用更多的接收缓冲区,这样,接收端缓存中用于质量好的路径的空间就会变小,这会降低质量好的路径上的传输数据量,进而影响整个关联的吞吐量。

研究表明[5],当质量差的路径上的包丢失率为 10%,质量好的路径包丢失率为 1%时,如果使用 RTX-SAME 重传机制(下面将会介绍),多路径同时传输

方法传输 8M 大小的文件所花费的时间是只用质量好的路径的 4 倍!

当使用多路径同时传输时,这种由于路径质量差异而引起的接收端缓冲区阻塞问题肯定会出现,除非缓冲区不再由多个路径共享。但是研究表明[5],重传时尽量选择质量好的路径可以有效降低接收端缓冲区阻塞问题。

3.6　重传时的路径选择技术

由于一个关联中可以存在多个路径,SCTP 的发送端在发送重传数据时就会面临多个选择。因为当前 SCTP 只允许数据从主路径发送,所以主路径上的拥塞控制信息(可用带宽、丢失率、RTT 等)比备选路径上的更加准确。正是基于这样的原因,通过原来的主路径发送重传数据将比使用备选路径更具性能优势[6]。这种重传机制称为 RTX - SAME,即重传的数据始终通过原先的路径来发送。

但是在多路径同时传输时,由于同时使用了所有可用路径来传送数据,所以每条路径上的拥塞控制信息都能够比较准确地反映当前路径状态,上面的结论[6]不再适用,研究表明[7]RTX - SAME 在多路径同时传输时,效果最差。

在文献[7]中,作者研究了 5 种重传策略:

1) RTX - SAME:使用报文原先发送的路径,除非路径失效。

2) RTX - ASAP:随机选择一条可用路径。

3) RTX - CWND:使用具有最大 cwnd 值的路径。

4) RTX - SSTHRESH:使用具有最大 ssthresh 值的路径。

5) RTX - LOSSTATE:使用具有最低报文丢失率的路径。

在这 5 种重传策略中,RTX - SAME 最简单;RTX - ASAP(as soon as possible)希望重传的数据能够尽快发送出去,这是一种随机策略;RTX - CWND, RTX - SSTHRESH, RTX - LOSSRATE 都倾向于将重传的数据发送到具有最低报文丢失率的路径上。研究表明[7],在多路径同时传输时,倾向于使用最低丢失率的三个重传策略的性能最好,而且这三种策略能够有效减轻上面提到的接收端缓冲区阻塞问题;而 RTX - SAME 的表现最糟糕;RTX - ASAP 的表现一般。

4　总　结

随着多宿主机在普通消费者中的流行,如何利用多宿进行负载分配正成为当前的研究热点。传输层是第一个端到端的协议层,它拥有详细的端到端的路径信息,所以传输层的多路径同时传输将比其它协议层在负载平衡上做的更好,同时在传输层还可以进行拥塞控制、丢失检测与错误恢复,而且还有利于程序移植。传统的 TCP, UDP 协议都不支持多宿;作为 IETF 推荐的传输层协议,SCTP 具有 UDP 和 TCP 的许多优点,更重要的是 SCTP 内在地支持多宿。

文中介绍了基于 SCTP 多宿特性的多路径同时传输;并且总结了多路径同时传输的关键技术,包括快重传触发技术、拥塞控制窗口增长技术、延迟应答技术、减轻接收端缓冲区阻塞问题的技术以及重传时的路径选择技术。虽然这个领域的研究刚刚开始,但是许多研究结果[4,6]表明,基于 SCTP 多宿特性的多路径同时传输能够获得极大的吞吐量提升。

参考文献:

[1] RFC2960 - 2000. Stream Control Transmission Protocol[S]. 2000.

[2] STEWART R, ONG L. Stream Control Transmission Protocol Implementer's Guide[R]. IETF Internet Draft, Work in Progress, 2005.

[3] RFC2581 - 1999. TCP Congestion Control[S]. 1999.

[4] LYENGAR J, SHAH K, AMER P, et al. Concurrent Multipath Transfer Using SCTP Multihoming[C]// In SPECTS 2004. San Jose, California:[s.n.],2004.

[5] IYENGAR J, AMER P, STEWART R. Receive Buffer Blocking In Concurrent Multipath Transport[R]. USA: CIS Dept, University of Delaware,2005.

[6] CARO A, AMER P, STEWART R. Transport Layer Multihoming for Fault Tolerance in FCS Networks[C]// In MILCOM 2003. Boston, MA:[s.n.],2003.

[7] LYENGAR J, AMER P, STEWART R. Retransmission Policies for Concurrent Multipath Transfer Using SCTP Multihoming[C]// In IEEE ICON. Singapore:[s.n.],2004.

(上接第 4 页)

[6] XU ZhiWei, BU GuanYing. A theorem on grid access Control [J]. J. Comput. Sci & Technol, 2003, 718(4): 515 - 522.

[7] 梁建民,李　伟,徐志伟.所有者为中心的网格文件共享研究[J].计算机研究与发展,2003,40(12):1781 - 1786.

[8] Iaminitchi A, Foster I. On Fully Decentralized Resource Discovery in Grid Environments[C]//Proceedings of the Second International Workshop on Grid Computing. Heidelberg: Springer - Verlag, 2001: 51 - 63.