协议栈内部非阻塞通信的一种实现策略

A Implementation Scheme of Non – blocked Communication in Protocol Stack

何友鸣　郭凯红　（中南财经政法大学信息学院 武汉 430064）

摘要：以 TCP/IP 为例介绍了网络协议栈内部通信的传统实现方式，由此提出了栈内各协议层间通信的一种新的实现方式，即非阻塞 Callback 方式；介绍并讨论了它的基本思想、实现方案、服务原语和其他相关问题，如计时器、内存管理、死锁等；最后说明了它的局限性和应用领域，以及未来的研究目标。

关键词：协议栈　阻塞　函数注册　Callback

1　引言

计算机网络最基本的功能是资源共享和信息交换。为了实现这些功能，网络中各实体间经常需要进行各种通信和对话，这就要求各实体必须遵循一些事先制定好的规则与标准，这就是协议。为了降低设计的复杂性和便于维护，一般的网络设计都采用了层次结构，即协议分层。协议是系统中关于同一层次的对等实体之间的概念，而协议栈则是特定系统中所有层次的协议的列表。

协议栈内部相邻两层间的通信是通过服务访问点 SAP 实现的，通信的内容一般为控制包、数据包和相关的事件消息，实现方式一般采用滑动窗口机制，其优点是有效地利用了网络带宽，提高了数据流传输过程的效率。本文提出了一种新的栈内通信的实现方式，即非阻塞 Callback 方式，它也同样具有上述优点，除此之外，还具有较高的实时响应能力。本文介绍了它的基本思想、实现方案和服务原语；对于其他相关问题也给予了必要的讨论。

2　协议栈内部通信的传统实现方式

协议栈内部各层协议可为其分组投递提供两种服务，即不可靠的数据投递服务和可靠的数据流投递服务。从技术角度来讲，前者被定义成不可靠的、尽最大努力投递的、无连接的分组投递系统。这种服务通常由底层协议实现，它将每个分组都独立对待，并且不能保证其正确投递，分组可能会丢失、重复、延迟或不按序投递，但服务不检测这些情况，也不提醒发送方和接收方。一个典型的提供不可靠投递服务的协议是 IP 协议。

与其相比，可靠的数据流投递服务确保在协议的对等端之间进行没有重复和丢失的数据流的投递，通常由高层协议实现。这种服务使用一个名为“带重传的肯定确认（Positive Acknowledgement with Retransmission）”的技术作为提供可靠性的基础。这项技术要求接收方收到数据之后向发送方回送确认信息 ACK。发送方对发送的每个分组都保存一份记录，在发送下一个分组之前等待确认信息；同时发送方还在送出分组的时候启动一个计时器，并在计时器的计时期满而确认信息还没到来的情况下重发刚才的分组。一个典型的提供可靠数据流投递服务的协议是 TCP 协议。本文所要讨论的栈内非阻塞通信的实现策略也即是针对上述可靠的数据流投递服务的。

对提供可靠服务的协议层，其栈内通信通常是采用线程阻塞的方式来实现的。为了获得可靠性，发送方启动数据发送线程，发送一个数据分组，然后将相关的事件同步对象置为无信号状态，等待相应的确认信息到来而不发送下一个数据分组。此时发送线程处于阻塞状态。当确认信息到来后，由其他相关的读取线程将上述事件同步对象置为有信号状态，唤醒被阻塞的发送线程，使该线程继续发送下一个数据分组。这是最简单的使用带重传的肯定确认技术传输数据的情况。

使用上述机制传输数据，尽管网络具有同时进行双向通信的能力，但在一段时间内，数据只在机器之间

单向传输。在机器响应的时延期间,如计算路由或检测校验和时,网络完全处于空闲状态。设想在一个时延很大的网络上,这个问题就更加突出了:由于在接到前一个分组的确认信息之前必须推迟下一个分组的发送,这样,简单的肯定确认协议浪费了大量宝贵的网络带宽。

滑动窗口技术是简单的带重传的肯定确认机制的一个更复杂的变形。滑动窗口协议允许发送方在等待一个确认信息到来之前可以发送多个分组,这就有效地利用了网络带宽。对滑动窗口操作的最简单的设想就是一个传输分组序列,滑动窗口协议在分组序列中放置了一个固定长度的小窗口后将窗口内的所有分组都发送出去,如图1所示。

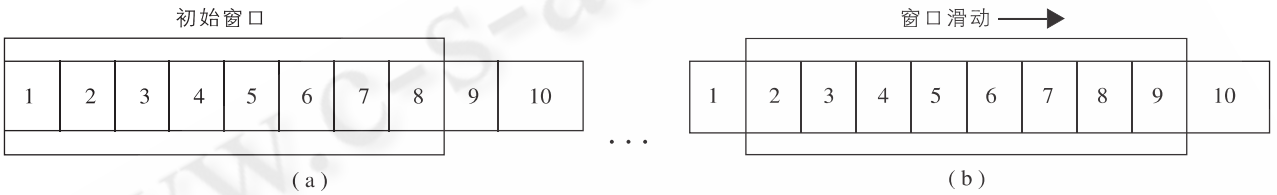

图 1　滑动窗口示例

如果一个分组被发送出去而没有收到确认信息,我们称这个分组为未被确认的。从技术上讲,任意时刻的未被确认的分组数取决于窗口大小,并被限定在一个固定大小的数目内。例如,当滑动窗口协议规定窗口大小等于 8 时,发送方可在收到确认信息之前送出 8 个分组,如图1(a)所示。此时发送线程处于被阻塞状态,等待确认信息的到来。

当发送方收到了窗口内第一个未被确认分组的确认信息后,发送线程被唤醒,它向后滑动窗口并发送下一个分组,如图1(b)所示。随着确认信息的不断到达,窗口也在不断地向后滑动。

当滑动窗口大小等于1时,滑动窗口协议就等同于简单的肯定确认协议。通过增加窗口大小,可以完全消除网络的空闲状态,即在稳定的情况下,发送方能以网络传送分组的最快能力来发送分组。

从概念上讲,滑动窗口协议要记录哪些分组被确认,并为每个未确认的分组设定计时器。如果某个分组丢失,对应的计时器超时之后发送方就会重新传送这个分组。当发送方滑动其窗口时要把所有已经确认的分组移到窗口之外;在接收端,协议软件要建立一个类似的窗口,在分组到达之后进行接收并回送确认信息。这样,滑动窗口就把所有的分组划分成三个部分:窗口左边的分组是已经成功地传输、接收和确认了的分组,而窗口右边的分组是还没有发送的分组,窗口内的分组是已经发送了的、但还没有收到确认信息的分组。窗口内序号最小的分组是还没有收到确认的分组序列中的第一个分组。

TCP 协议即是通过滑动窗口机制来实现可靠的数据流投递服务。可靠的数据流投递服务是如此的重要,以至于整个协议栈被命名为 TCP/CP。我们应理解滑动窗口机制的关键点,即发送方可以发送窗口内所有的分组而不必等待确认信息。

3　栈内非阻塞通信的实现策略

使用滑动窗口机制发送分组时,在等待窗口内未确认分组的确认信息到来之前,发送线程将会处于阻塞状态。此时上层协议如果有新的数据要发送,则必须等待。为了提高协议栈内部的实时响应能力,我们提出了一种新的栈内通信策略,这种策略使本层协议的发送线程可以尽力地发送由上层协议传来的分组,不会因为等待确认信息的到来而被阻塞。事实上,发送线程不必考虑确认信息何时到来的问题。当接收方的确认信息到来之后,由下层协议回调本层注册的 Callback 函数,进行相应的处理工作。我们称这种实现策略为非阻塞 Callback 方式。

3.1　非阻塞 Callback 方式的设计思想

按照软件工程的指导思想,系统中功能函数的调用通常是自顶向下、逐层调用。为了实现协议栈内部非阻塞通信的功能,我们需要下层协议反向调用上层协议的某些函数以方便地执行特定的操作。我们称这种调用方式为 Callback 方式,即回调。

协议栈内部各协议层在启动时需要进行初始化操作。初始化操作的一个重要功能是向下层协议注册用来响应相应事件的 Callback 函数,这些 Callback 函数的功能在本层协议实现。这个注册操作非常重要,当有分组或确认信息到来时,下层协议就要使用这些注册信息来回调本层协议的 Callback 函数以执行相应的功能。如果协议层提供的是可靠服务,则初始化操作还要建立几个空的队列,如事件同步对象队列、数据缓冲队列、计时器队列等。事件同步对象队列用来实现提供给上层协议(或用户)的 API 是阻塞方式的函数调用接口;数据缓冲队列和计时器队列用来记录分组或确认信息的状态和内容,实现超时重发功能。

当上层协议(或用户)发送一个网络请求或分组后,本层协议将启动发送线程。如果协议层提供的是可靠服务,则先将这些请求或分组复制至数据缓冲队列,同时增加相应的计时器节点到计时器队列中,以备超时重发之用;如果提供给上层协议(或用户)的 API 是阻塞方式的函数调用接口,还要将相应的事件同步对象置为无信号状态,并将其增加到事件同步对象队列中。然后发送线程将尽力地发送由上层协议传来的分组,不必等待确认信息的到来,也就不会因此而被阻塞。

当接收方收到请求或分组而做出响应后,下层协议将调用本层注册的 Callback 函数以做出相应的动作。通常是删除本层数据缓冲队列和计时器队列中相应的节点,并将分组或确认信息向上层传递;如果提供给上层协议(或用户)的 API 是阻塞方式的函数调用接口,还要将事件同步对象队列中相应的节点置为有信号状态。

最后,本层的 Callback 函数返回。如果提供给上层协议(或用户)的 API 是阻塞方式的函数调用接口,则上层协议(或用户)调用的阻塞被释放并返回相应的结果。

3.2 实现方案

限于篇幅,这里仅给出函数注册、数据发送、数据接收等较为关键过程的实现方案。

(1) 函数注册。所谓的上层协议向下层注册自己的 Callback 函数,实际上就是使下层协议通过函数指针实现对上层协议相关函数的调用。假设协议层 A(上层)定义了连接请求应答函数 A_ConnectCfm(PARAM papam1, . . .),在协议层 A 的初始化操作中,要将函数 A_ConnectCfm 注册到协议层 B(下层)中。这需要调用协议层 B 向上提供的注册函数,该函数接口声明的伪码如下:

```
LRESULT B_RegisterFun( FUN_CONNCFM A_CnCfm,
PARAM papam1, . . . );
```

其中 FUN_CONNCFM 是用户自定义函数指针类型,其声明为

```
typedef LRESULT( * FUN_CONNCFM) ( PARAM pa-
pam1, . . . );
```

协议层 A 的注册过程很简单:它向下调用协议层 B 的注册函数,并将被注册函数名作为其调用函数的参数

```
B_RegisterFun( A_ConnectCfm, papam1, . . . );
```

(2) 数据发送。提供可靠服务的协议层在发送数据时需要使用数据缓冲区。上层协议如果有数据,就会启动发送数据线程,或直接调用下层协议发送数据的函数,直到所有的数据都被发送出去后才返回。这并不会阻塞协议层的其他操作。这是因为协议层在发送数据时通常都会有流量控制,当协议层不能再发送数据时,它的流量控制会退出发送操作转而进行其他的操作。功能调用如图 2 所示:

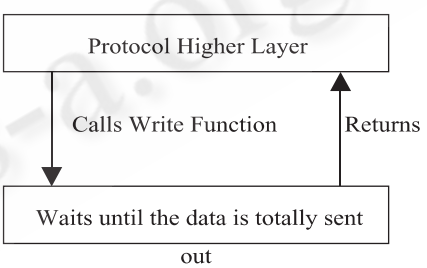

图 2　发送数据功能调用

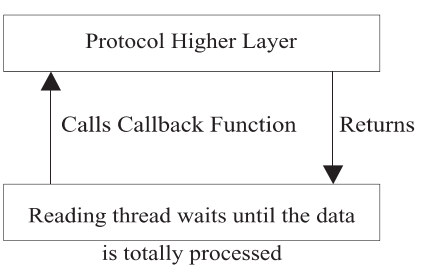

图 3　接收数据功能调用

数据接收。接收数据时需要使用回调函数。上层协议可以使用接收缓冲区,也可以不使用,这要视需要而定。当有数据到达时,协议层就调用上层提供的回调函数,由回调函数负责对接收到的数据进行解析和处理。如果需要,可将接收的数据继续向上层传递。

3.3 整体通信机制描述

这里我们给出了某一抽象协议栈 Task 关联图和通信机制,如图 4 所示。

图 4 较好地描述了协议栈内部非阻塞 Callback 方式的通信机制。

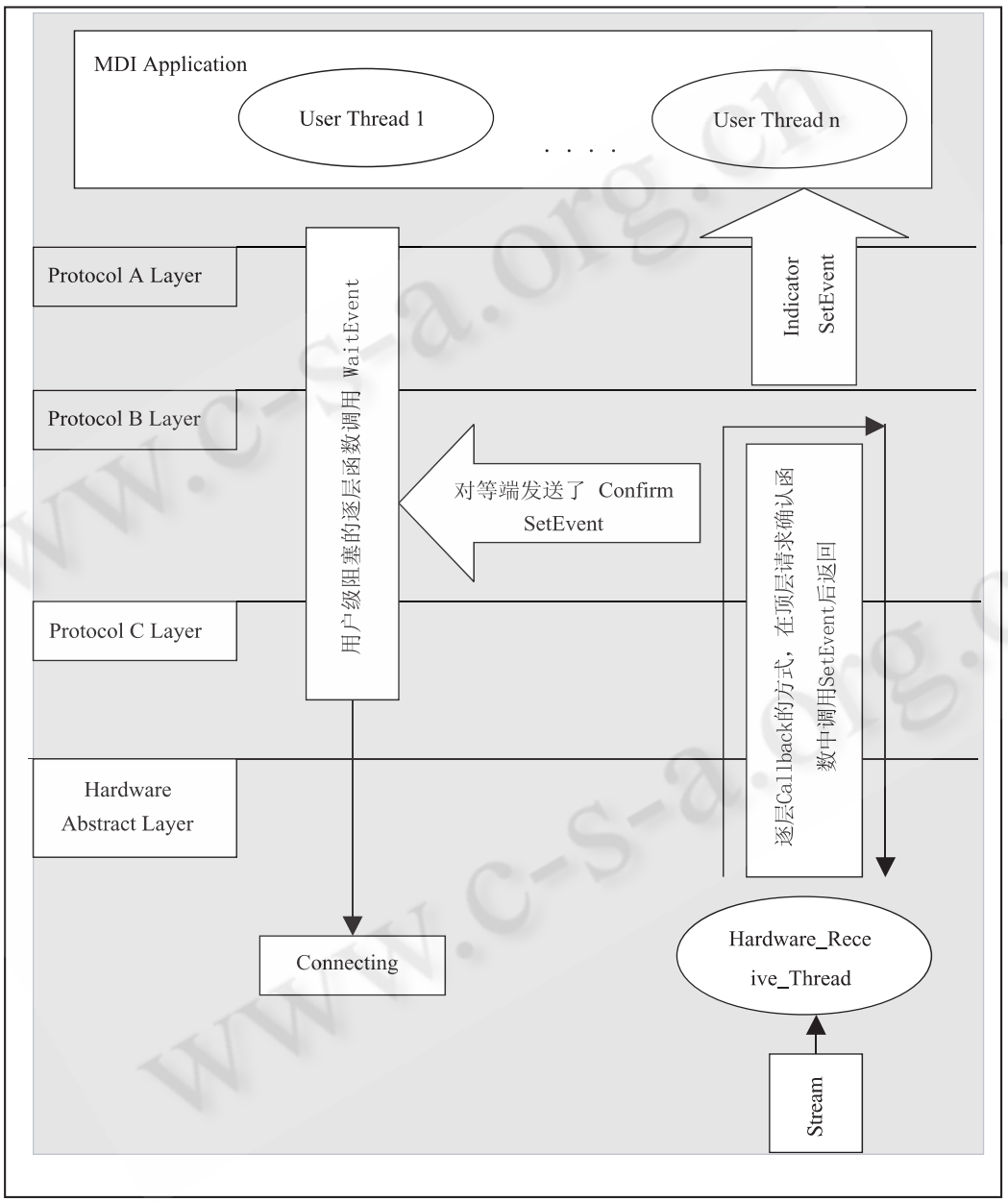

图 4 协议栈 Task 关联图与通信机制

这种方案实现起来方便快捷。它假定 CPU 对数据的处理速度远远大于协议栈从硬件上接收数据的速度,这样就不会引起硬件以及硬件驱动程序中接收缓冲区的溢出。功能调用如图 3 所示。

4 结论

协议栈内部通信机制的选择与建立,对于实现网络实体间信息交换的可靠性与高效性起着至关重要的作用。协议栈内部传统的通信方式是利用滑动窗口机制,以阻塞模式实现各协议层间的通信。本文提出了一种新的实现策略,即利用 Callback 机制,以非阻塞模式实现栈内各协议层间的通信;描述了它的核心思想及工作原理;对于方案中较关键过程的实现,给出了相应的措施;同时对协议栈相关问题也进行了讨论。这种非阻塞通信策略的优点是可以有效地利用网络带宽,增加系统的实时响应能力,在向对等方发送命令时,可以避免系统死锁;缺点是应用上有一定的局限性,适合于传输用户数据包较少而指示命令较多的应用系统。未来的研究目标是设计出更有效的非阻塞调度算法、流量控制算法和内存管理机制,以解决海量用户数据包的发送问题。

参考文献

1 Douglas E. Comer. Internetworking With TCP/IP Vol. Ⅰ: Principles, Protocols, and Architecture (Third Edition). New York: Prentice Hall, 1998.

2 Douglas E. Comer & David L. Stevens. Internetworking With TCP/IP Vol. Ⅱ: Design, Implementation, and Internals (Second Edition). New York: Prentice Hall, 1998.

3 Douglas E. Comer & David L. Stevens. Internetworking With TCP/IP Vol. Ⅲ: Client – Server Programming and Applications (Second Edition). New York: Prentice Hall, 1998.

4 W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols. New York: Addison Wesley, 1994.

5 Gray R. Wright & W. Richard Stevens. TCP/IP Illustrated, Volume 2: The Implementation. New York: Addison Wesley, 1995.

6 David J. Kruglinski. Programming Visual C + +. Seattle: Microsoft Press, 1998.

7 刘书生、赵海,蓝牙技术应用,沈阳 东北大学出版社,2001。

8 张立云等,计算机网络基础教程,北京 电子工业出版社,2000。

(编者按:原文篇幅较长,本刊对部分内容做了删略。)