数据交换平台数据传输器的设计与实现①

季士普[1,2]，廉东本[2]

[1](中国科学院 研究生院，北京 100049)
[2](中国科学院 沈阳计算技术研究所，沈阳 110171)

摘 要：为解决辽河流域水环境综合监管系统数据交换平台的数据传输问题，设计并实现一个简单实用的数据传输器组件，简单描述组件的消息结构和消息传输的具体实现，重点介绍消息传输的可靠性传输机制，最后提出基于日志的事务部分回滚策略。

关键词：数据交换；消息；可靠性；基于日志的事务部分回滚策略

Design and Implementation of Data Transmitter Based on Data Exchange Platform

JI Shi-Pu[1,2], LIAN Dong-Ben[2]

[1](Graduate School, Chinese Academy of Sciences, Beijing 100049, China)
[2](Shenyang Institute of Computering Technology, Chinese Academy of Sciences, Shenyang 110171, China)

Abstract: To solve the data transmission problems of data exchange platform in the Liao River Basin water environment monitoring system, this paper designs and implements a simple and practical data transmission component, which simplely descripts the component to the message structure and message transmission of the specific implementation and focuses on reliable message transmission of the transport mechanis. Finally, log-based partial rollback strategy is proposed.

Key words: data exchange; message; reliable; log-based partial rollback strategy

近年来，辽河流域开展了水体污染控制及水体污染治理的研究，研制了各种水环境管理信息化系统，包括辽河流域水环境污染源管理系统、水环境质量管理系统、水环境风险评估系统、水环境预警系统和应急响应系统等。如何解决数据资源的交换共享问题，发挥数据信息应有的效能，对辽河流域水环境综合监管系统建设影响重大。实践表明，建立统一的数据交换平台是实现水环境管理信息化系统间数据交换与信息共享的有效途径[1]。

数据传输器（Data Transmitter）是辽河流域水环境综合监管系统数据交换平台的重要组件。数据传输器是数据发送、接受、传输的通道，支持报文和文件两种模式，支持数据在各应用系统之间可靠、稳定、高效地双向传输。

辽河流域水环境综合监管系统数据交换平台数据传输具有如下特点：

① 传输只应用于结点到结点的直接通信，即中间不存在路由协议转换。

② 两个传输结点必定在同种类型的网络上，如以太网或 X.25 等。

本文在分析现有数据传输器技术，如 IBM WebSphere MQ 等技术的基础上，结合数据交换平台的实际应用情况，设计并实现了一个简单实用的数据传输器组件。

数据传输器传输的数据可能是报文，也可能是文件，可以将它们统一描述为消息。数据传输器通讯的过程是以消息的形式表现的[3-5]。在通信时可以对消息设置生命周期和阻塞级别，保证通讯的效率和资源利用。通讯时，不同的应用对消息的要求是不同的，因此数据传输器提供了多种机制来适应用户的实时性、

① 收稿时间:2011-06-24;收到修改稿时间:2011-07-22

可靠性和安全性的要求。

1　数据传输器消息结构

数据传输器的消息是由消息头和消息体组成的。在消息头中，包含了消息接受者的地址、消息长度、消息类型、消息传递的阻塞级别、以及其它一些消息控制信息，如：生命周期、超时时间等。消息体是消息的数据。

(1) 消息接收者的地址

一个进程要向另一个进程发送消息时，它必须知道消息接收进程的位置以及进程号。数据传输器采用“服务类+pid”的机制来实现对接收进程定位。

(2) 消息的阻塞级别

数据传输器提供了四种消息阻塞等待级别。非阻塞方式：应用将消息投递到本结点数据传输器核心后，不等待任何确认消息后立即返回；阻塞方式 1：应用将消息投递后，等待本地结点数据传输器核心的确认消息；阻塞方式 2：应用将消息投递后，等待本地结点数据传输器核心返回的确认消息，直到消息传输完成；阻塞方式 3：应用将消息投递后，等待目的结点返回的确认消息，直到消息递交给用户进程。

(3) 消息类型

数据传输器可以发送三种类型的消息：BUF 消息、文件消息和复合消息。BUF 消息是指消息存放在系统内存中；文件消息是消息传送的是一个磁盘文件；复合消息是由多个子消息构成，每个子消息既可以是 BUF 消息，也可以是一个文件。消息类型不同，数据传输器的传输方式也不同。复合消息在本质上是将各个子消息保存在一个临时文件中，然后传输的是这个临时文件。消息类型的表示在数据传输器中利用了消息长度来区分。BUF 消息：消息的长度就是该消息的实际长度；文件消息：消息长度设置为 0；复合消息：消息长度设置为-1。

(4) 消息内容

消息内容跟消息的类型相关。当发送的是一般的 BUF 消息，消息内容就是消息实际数据；当发送的是文件时，消息内容是一个形如“源文件名,目的文件名”的字符串；当发送的是复合消息时，消息内容为一个复合消息结构。

2　数据传输器消息传输具体实现

实质上，在数据传输器系统中，消息的传输包括两个层次上的数据交换：应用系统和数据传输器核心之间的数据交换，以及数据传输器核心之间的交换。

应用程序在调用数据传输器发送消息时，是将消息的信息写进系统消息队列，将消息数据写入缓冲区的过程；而接收消息时，是从消息队列取出消息的信息和存放位置，然后从缓冲区中取出消息数据的过程。数据传输器核心之间的通信也是先从消息队列和缓冲区中读消息，然后将消息通过网络传输到目的结点。目的结点的数据传输器核心再将消息写入系统的消息队列和缓冲区。如图 1 所示。

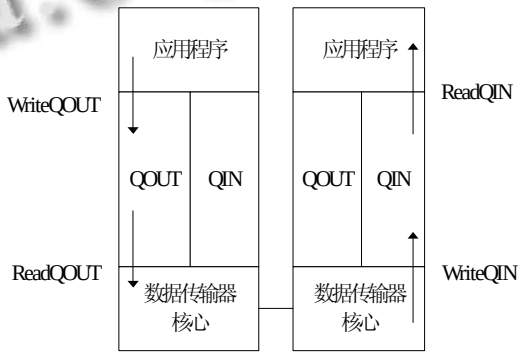

图 1　消息传输流程

消息队列每一项的数据结构定义：（如图 2 所示）

```
struct shmq{
int index; /*队列项索引*/
char udtype; /*消息类型*/
short flag;/*事务处理中标志消息是否已读*/
int pid; /*消息 PID*/
int txId;/*事务号*/
int nNextRead; /*链表中下一个队列项索引*/
int nNextWrite;/*链表中上一个队列项索引*/
int timeout; /*超时值，没有事务标志是 10*/
short bufHead; /*消息存放的第一个 BUF*/
short bufTail; /*消息存放的最后一个 BUF*/
int bufNum; /*消息所占 BUF 数*/
}
```

如图 2 所示，在一个消息中，往往包含了若干个数据包，在通信过程中，常常是将一个消息分段发送的。为了保证消息传输的效率，数据传输器借鉴了 TCP/IP 的滑动窗口协议，设计了一种滑动窗口机制来

保证传输的可靠性和效率,以及对通信进行流程控制[6]。如图 3 所示。

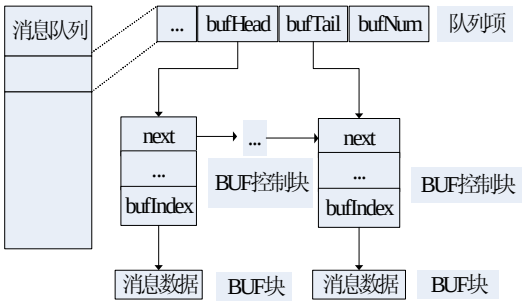

图 2 消息队列队列项

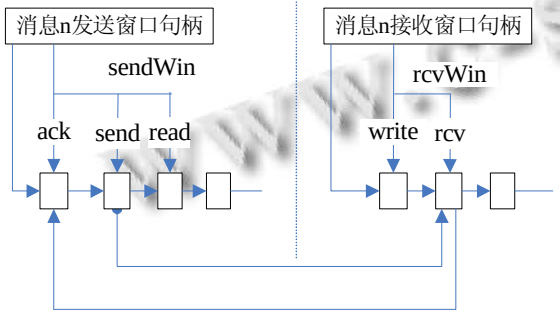

图 3 滑动窗口机制

3 消息的可靠性传输机制

消息可靠性分类（这样使得消息可靠性用户可以控制）

- SMALL_RAW_MSG

不加工的小消息，这种消息不进行可靠性处理，直接放到下行数据流（Down Stream）进行传输，只是在传输中出现错误时，通过事件通知来通知上层。这种消息的大小不能超过发送缓冲区的大小。

- NORMAL_MSG

这种消息不进行事务日志处理，因此它没有断点重传的能力，但它能保证消息在系统不发生异常的情况下，确保消息的可靠传输。

- LOG_MSG

这种消息需要进行事务日志处理，以保证在系统失败，崩溃等情况下，仍能可靠传输，具有断点续传的能力。

要保证可靠，需要把消息传输的有关数据和状态等日志信息保存到磁盘上，数据传输器重新启动后能够事务回滚，从磁盘上恢复继续传送。从数据结构上来看，就是在传输过程中需要保存消息数据和消息传输句柄信息等。

如何磁盘保存消息数据呢？根据消息数据的属性，有两种类型:file 和 buf。对于 file，发送端数据直接从文件中读，接收端直接写数据到一个文件中，因此不需要保存消息数据。而对于一个 buf 消息，有两种情况：<4K 时，消息的数据保存到一个大的临时文件的块中，因此，发送端需要保存一次数据，接收端在接收过程中根据发送窗口的情况也保存到一个块中；对于>4K 的 buf，数据传输器核心把它存放到一个文件中，把它作为一个 file 类似的方法来发送。只是，在接收端，用户接口不确定，因为这一消息可能很大，不能要求用户提供的 buf 要满足这种要求，因为，同样有很多很小的消息。这里主要看大于 4K 的 buf 消息有没有必要。如需要，可在用户接口中动态分配 buf，让用户去释放。这样做的缺点是用户接口不一致。

我们提出一种基于日志的事务部分回滚策略，即通过该事务所做日志信息中的内容来回滚事务。部分回滚的目的是为了撤销事务在保存点之后对数据传输器的影响，使事务恢复到保存点的状态。我们主要参考 ARIES 算法，并对其进行一定的改进以适应需要。首先介绍一下实现部分回滚需要用到的一些数据结构。ARIES 算法的事务管理器在运行中维护一张事务表，用以记录事务的执行状态。数据传输器对其进行一定的扩充以适应保存点的设置和撤销[7]。具体结构如下：

TxnID：事务标识，用于在系统中唯一标识一个事务

State：记录事务是在运行、提交或放弃状态

LastLSN：事务所写最近一个日志的日志序列号（LSN）

UndoNxtLSN：回滚时下一个需处理日志记录的 LSN

SaveList：指向保存点结构链表的指针，记录该事务所建立的所有保存点

保存点只在事务活动时有效，当在时刻 T 撤销或回滚到保存点 S 时，S 以及它以后建立的保存点都被撤销。我们设置了保存点结构体 Savepoint 用以记录保存点信息，具体结构如下：（如图 4 所示）

Spname：保存点的名字。

LogLsn：当前事务所写的最近一个日志的逻辑地址，通过它的大小可以得知保存点的建立时间。

Prev：指向该事务的保存点链表中的下一个保存点结构的指针。

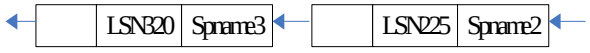

图 4 保存点链表

部分回滚的算法的总体思想为：首先根据保存点名在保存点链表中找到相应的保存点结构，在保存点链表中删除掉在该保存点后所建立的保存点。然后从当前事务所写的最近的一个日志开始对该事务所写的日志向前扫描，如果日志的 LSN（日志的逻辑地址）比保存点中的 LogLsn 大，则表明该日志是事务在保存点之后写入的，因此我们根据日志内容来回滚消息，同时写入补偿日志 CLRs 以撤销事务所写日志对数据传输器的影响。如果日志的 LSN 小于或等于保存点中的 LogLsn，则表明日志是在该保存点设置之前写入的，不需要回滚操作，停止扫描。事务回到了设置保存点时的状态，完成操作[8]。

```
/*回滚伪代码*/
RollBack(name, TxnID, type)
{
/*从事务表中获得 TxnID 对应的事务结构指针*/
entry = FindEntry(TxnTable, TxnID);
if(type != All)/*不回滚整个事务*/
{
position = entry->SaveList;
/*找到所在保存点结构*/
while(position != NULL)
{
if(position == NULL)
{
return;
}
position = position -> prev;
}
if(position == NULL)
{
return;
}
temp = entry->SaveList;
while(temp->name != position->name)
{
DestroySavepoint(temp->name, TxnID);
temp = temp->prev;
}
}
/*回滚事务所写日志*/
RollBackLog(entry, position);
/*回滚临时段*/
RollBackTempSegment(entry, position);
}
```

4 结语

本文从辽河流域水环境综合监管系统数据交换平台项目的实际需求出发，设计并实现了一个简单实用的数据传输器组件，目前该组件已在实际项目中试用。今后将进一步研究如何对该组件进行传输过程监控，及时发现传输错误，解决问题。

参考文献

1 邢诒俊,廖廷悟,龙天威.数据交换平台的实现.Financial computer of huanan,2010,12:45–47.
2 李磊,刘嘉勇.关于对 QQ 文件传输特点的研究.信息安全与通信保密,2011,4:36–37.
3 IBM WebSphere MQ V7 技术白皮书.IBM(中国)有限公司,2008.
4 Gu R, Keen M, Kuehlthau E, McWilliam L, Ward D. Getting Started with WebSphere MQ File Transfer Edition V7. International Technical Support Organization, August 2009.
5 王建,江婷.浅谈消息中间件 IBM WebSphere MQ.微型机与应用,2010,5:6–8.
6 陈德栋.消息中间件消息可靠传输机制的实现.成都:电子科技大学,2002.
7 姚建中,孙建伶,董金祥.SQL3 保存点和部分回滚的设计和实现.小型微型计算机系统,2001,22(3): 313–316.
8 邹现军,彭智勇,王黎维.基于 MVCC 的保存点和事务部分回滚功能的设计与实现.海军工程大学学报,2004,16(5): 21–24.