第一章　算法实现

1.1　IB网络数据接收发模块简介

IB网络数据接收发模型和B/S（Browser/Server）模型很类似。在B/S模型中浏览器通过网址向服务器请求网站数据，服务器收到请求后会向浏览器传输所需网站内容。当然上述过程隐藏了很多B/S模型的实现细节，在这里不做详细讨论。IB网络数据接收发模型的网络节点之间的数据交换过程也会有同样的请求流程，但是与B/S 模型有很大不同的是任意一个IB节点既可以是浏览器也可以是服务器，便于节点间的数据交换。

IB网络数据接收发模块主要分为两个功能块：

* **IBDataRequestPacket模块：**IBDataRequestPacket（IB数据请求包）是一个网络数据包，用于IB节点向其他IB节点发送数据请求操作。该包包含了一些IB 数据请求的必要内容，比如：应该使用那块Queue Pair去操作IB数据，所需准备的缓冲区大小和请求什么类型的数据等。

* **IBWCEHandlingTheread模块：**该模块是一个IB 网络监听线程，用于监听IB节点的数据收发状态操作。

假设当前有两台IB主机：数据请求端A和数据发送端B。数据请求端A为了获取资源R，首先会向数据发送端B发送数据请求包（IBDataRequestPacket）请求资源R，同时数据请求端A会准备好用于数据操作的Queue Pair；当数据发送端B收到数据请求端A的数据请求包，它会根据数据请求包的Queue Pair直接向数据请求端发送其所需的数据R。但需要注意的是，IB数据请求包是通过TCP或UDP 的方式发送到数据发送端B，真正使用IB数据传输的过程是数据发送端B向数据接收端A传输资源R的时候。IB 网络数据接收发操作流程如图1.1。

1.2　IBDataRequestPacket模块

1.2.1　IBDataRequestPacket包内容

在表1.1中，IBDataRequestPacket包除了包含一些基本的信息外，还包含了包的额外的信息，表示IB节点所请求的数据类型：文件数据和消息数据。

表 1.1: 资源请求包的额外信息

SenderNodeName	包发送IB节点名
ReceiverNodeName	包接收IB节点名
RecvBufferSize	接收数据的缓存大小
ReceiverQueuePairID	接收数据的ID
PacketExtraInfo	包的额外信息

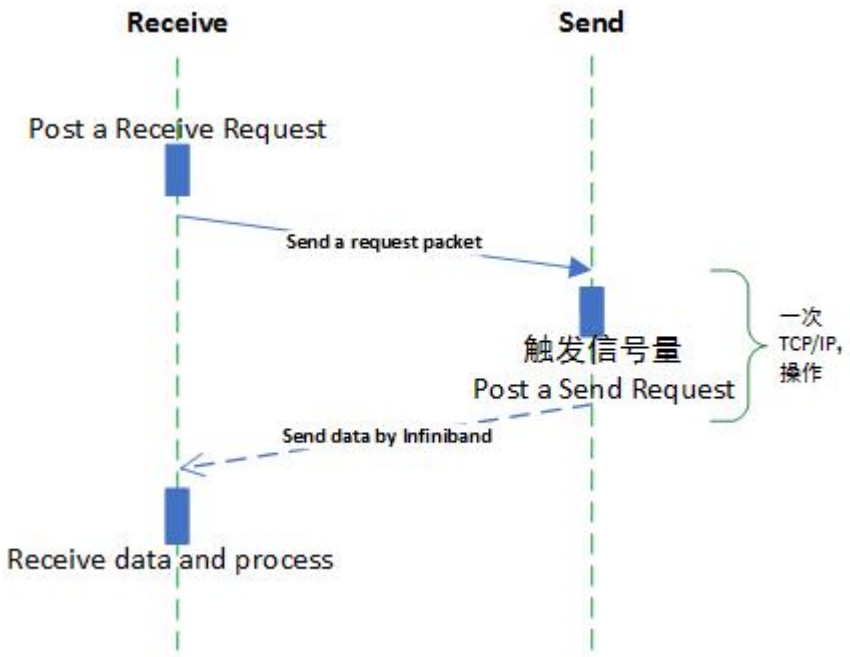

图 1.1: IB网络数据接收发操作

1.2.2 IBDataRequestPacket包处理

IB数据发送端在接收到IB数据请求端的IBDataRequestPacket过后，需要对包进行相应的一些列处理，处理流程如下：

1. 在整个网络系统中获取IB节点间的网络连接
2. 准备IB数据请求端所请求的数据
3. 通过网络连接向IB数据请求端发送数据

IBDataRequestPacket包处理算法伪码如Algorithm1。伪码的符号说明如下：
(1) HostNode：当前主机的网络节点（IB、UDP）信息
(2) Connection：当前主机之间的连接信息
(3) DataRequested：所请求的数据

Algorithm 1 processV
Require: $HostNode$
Ensure:
1: **function** PROCESSV($HostNode$)
2: $Connection \leftarrow findConnection2Receiver(HostNode, IP, NodeName)$
3: $DataRequested \leftarrow prepareIBData()$
4: $sendRawBuffer(DataRequested, ID) \leftarrow Connection$

IB数据准备工作涉及到两种类型：文件数据和消息数据，因此IBDataRequestPacket模块在处理数据过程中会有不同的数据准备方式。Algorithm2描述了整体的数据准备工作。

Algorithm 2 prepareIBData

Require:

Ensure: *DataRequested*

1: **function** PREPAREIBDATA()
2: **if** $PacketExtraInfo$ **is** $PacketExtraInfoOnMessage$ **then**
3: **return** $prepareIBMessageData(PacketExtraInfo)$
4: **else**
5: **return** $prepareIBFileData(PacketExtraInfo)$

1.2.3 IBDataRequestPacket数据接收准备

IB数据请求端在发送IBDataRequestPacket前，需要对IB网络底层进行数据的接收准备操作，根据指定的QueuePair ID来配置相关Queue Pair。其具体操作过程请参照IBConnection和QueuePair模块。

1.3 IBWCEHandlingTheread模块

IBWCEHandlingTheread模块是一个独立的线程，负责处理IB网络节点间的数据发送与接收状态工作。

1.3.1 IBWCEHandlingTheread主功能

IBWCEHandlingTheread实现主功能迭代过程中，会不断调用ib_poll_cq函数（IB 底层函数）是网络来检查在completion queue里面是否有新的work reuqests 请求，并返回当前completion queue状态。IBWCEHandlingTheread 模块主要使用其中的两种状态：

* **IB_NOT_FOUND**：当前在completion queue中没有新的work requests请求被取出。

* **IB_SUCCESS**：当前在completion queue中有新的work requests请求被取出。

ib_poll_cq同时会获取Work Completion的信息，如果Work Completion的类型是IB_WC_SEND表示当前需要进行发送数据操作，如果是IB_WC_RECV 类型则进行数据接收操作。算法的伪码如Algorithm3：

1.3.2 IBWCEHandlingTheread发送数据

IB发送数据的流程是在其他功能块完成的，当其他功能块发送完成IB数据，会触发IB的信号从而通知IBWCEHandlingTheread，IBWCEHandlingTheread再数据一条日志语句“Success to send data with IB node”。

1.3.3 IBWCEHandlingTheread接收数据

IBWCEHandlingTheread收到文件接收的信号时会进行数据接收的相关工作。在数据接收的流程中，首先会取出MemoryBlock中的缓存数据，根据当前接收的数据类型分别进行文

Algorithm 3 runV

Require:

Ensure:

```
1: function RUNV()
2:     while ThreadIsRunning() do
3:         while true do
4:             IBState ← ib_poll_cq(CompletionQueue)
5:             SucceededRequest ← ib_poll_cq(CompletionQueue)
6:             if IBState=IB_NOT_FOUND then
7:                 break
8:             else if IBState=IB_SUCCESS then
9:                 MemoryBlock ← fetchMemoryBlock(SucceededRequest)
10:                if SucceededRequest=IB_WC_SEND then
11:                    sendData(MemoryBlock)
12:                else if SucceededRequest=IB_WC_RECV then
13:                    receiveData(MemoryBlock)
```

件的存储或者消息的处理。处理文件数据会先在包的额外信息（CPacketExtraInfoOnFile）中取出文件所需保存的路径，然后直接将缓冲区的文件数据存储到上述路径中；处理消息数据时，首先会通过字符串流保存缓冲区的内容，然后再逆序列化出字符串流中的基础包类（CPacket），最后调用包的处理函数。Algorithm4是IBWCEHandlingTheread接收数据伪码表示：

Algorithm 4 receiveData

Require: $WorkCompletion, MemoryBlock$

Ensure:

```
1: function RECEIVEDATA(WorkCompletion, MemoryBlock)
2:     pBuffer ← MemoryBlock
3:     if PacketExtraInfo is PacketExtraInfoOnMessage then
4:         DeserializedStream ← pBuffer
5:         Packet ← Deserialize(DeserializedStream)
6:         processV(Packet)
7:     else
8:         FilePath ← PacketExtraInfo
9:         ofstream ← OutputFile(FilePath)
10:        write(pBuffer)
```