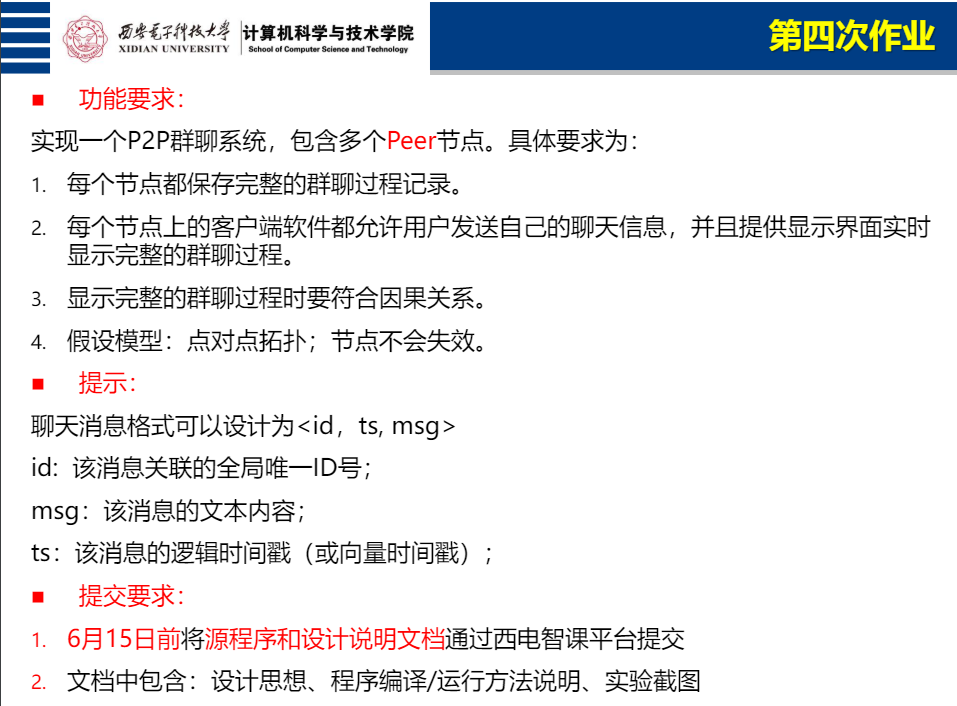
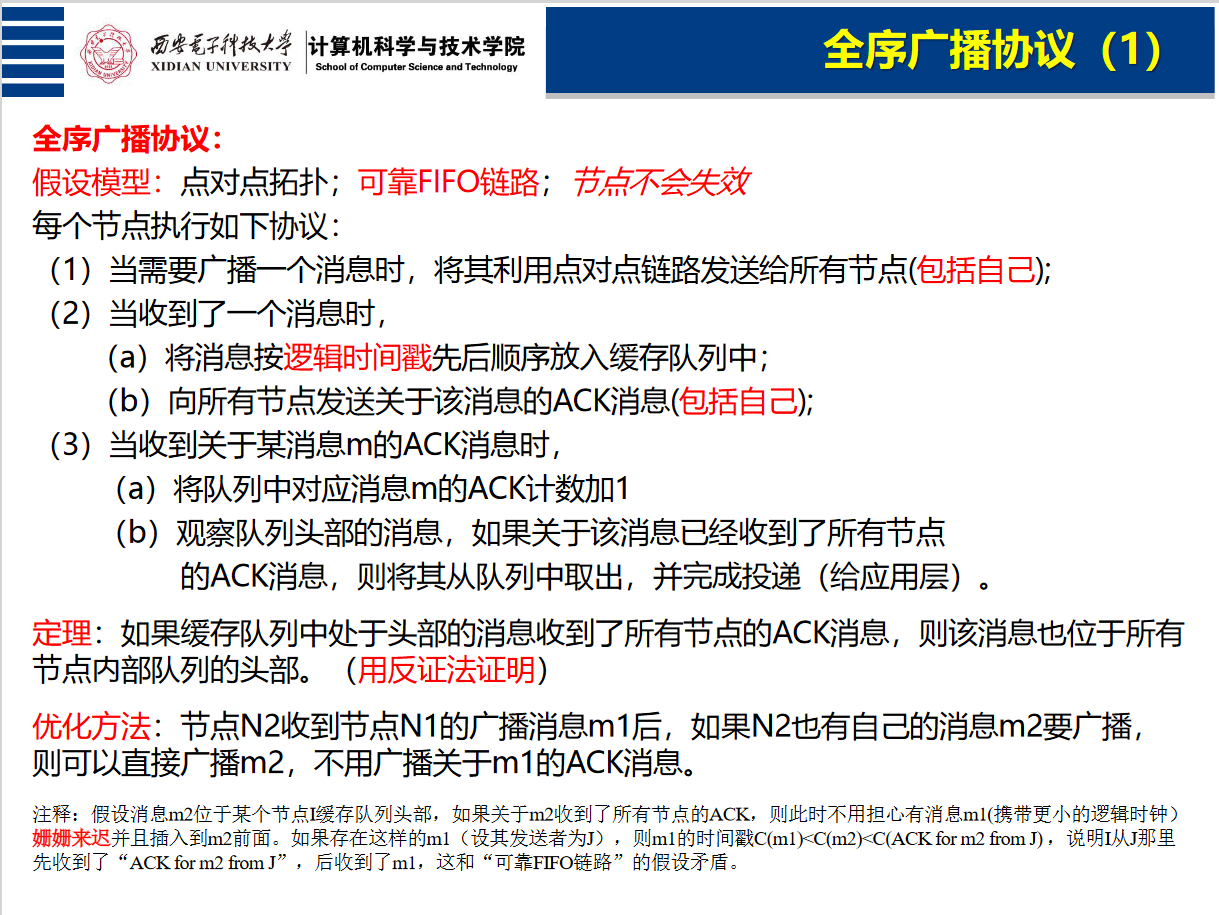
分布式设计报告

一、问题重述



二、实验原理

1、全序广播协议



采用一个服务端，多个客户端的方法，当客户端收到服务端传来的信息时，判断该信息属于ACK信息、新加入客户端信息还是message消息。

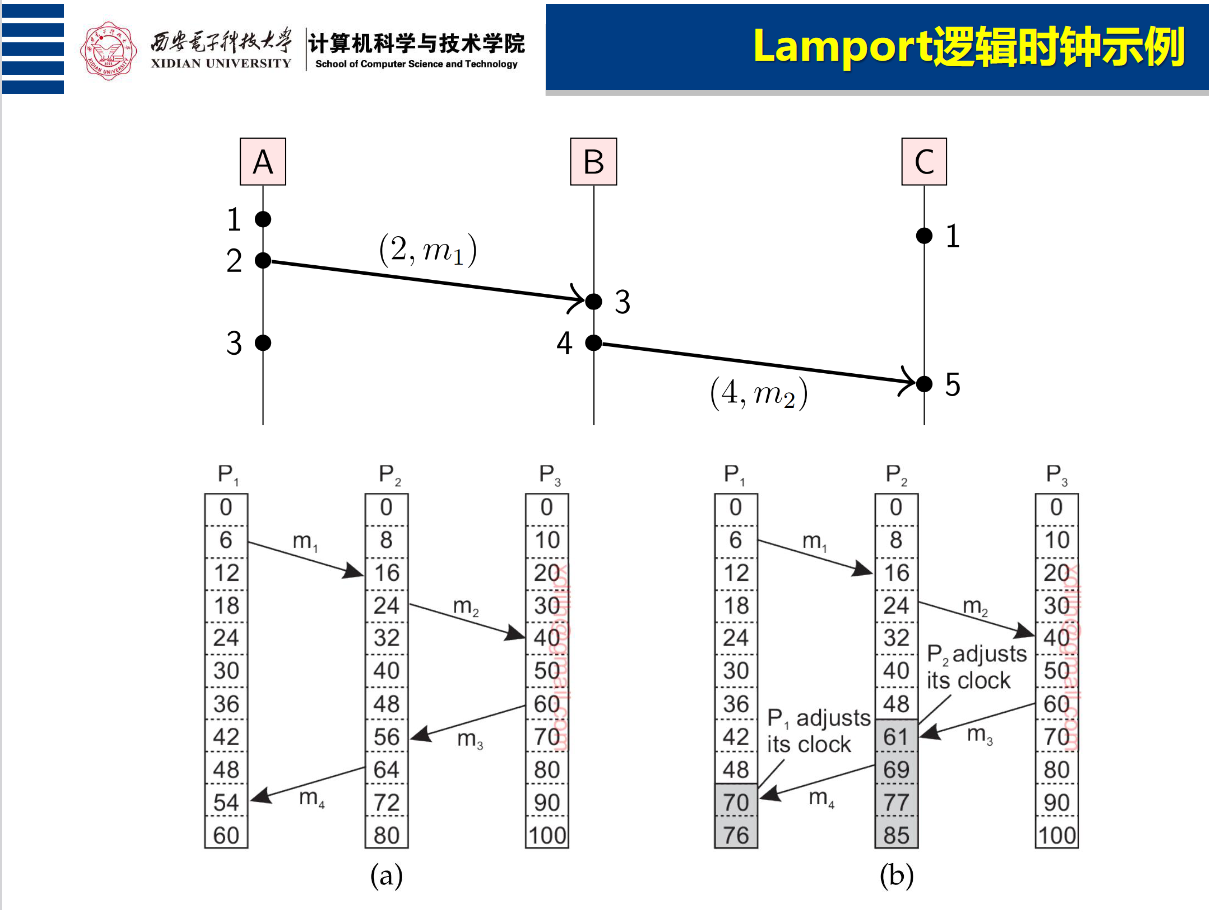
如果这条信息是属于ACK信息，将对应消息m的ACK+1,观察队列头部的消息，如果关于该消息已经收到了所有节点的ACK消息，则将其从队列中取出，并完成投递。

如果这条信息属于发送的信息,将消息按逻辑时间戳先后顺序放入缓存队列,观察缓存队列头部的消息，如果该消息尚未广播过ACK消息，则向所有节点发送关于该消息的ACK信息.

具体实现如下图代码所示



2、因果关系



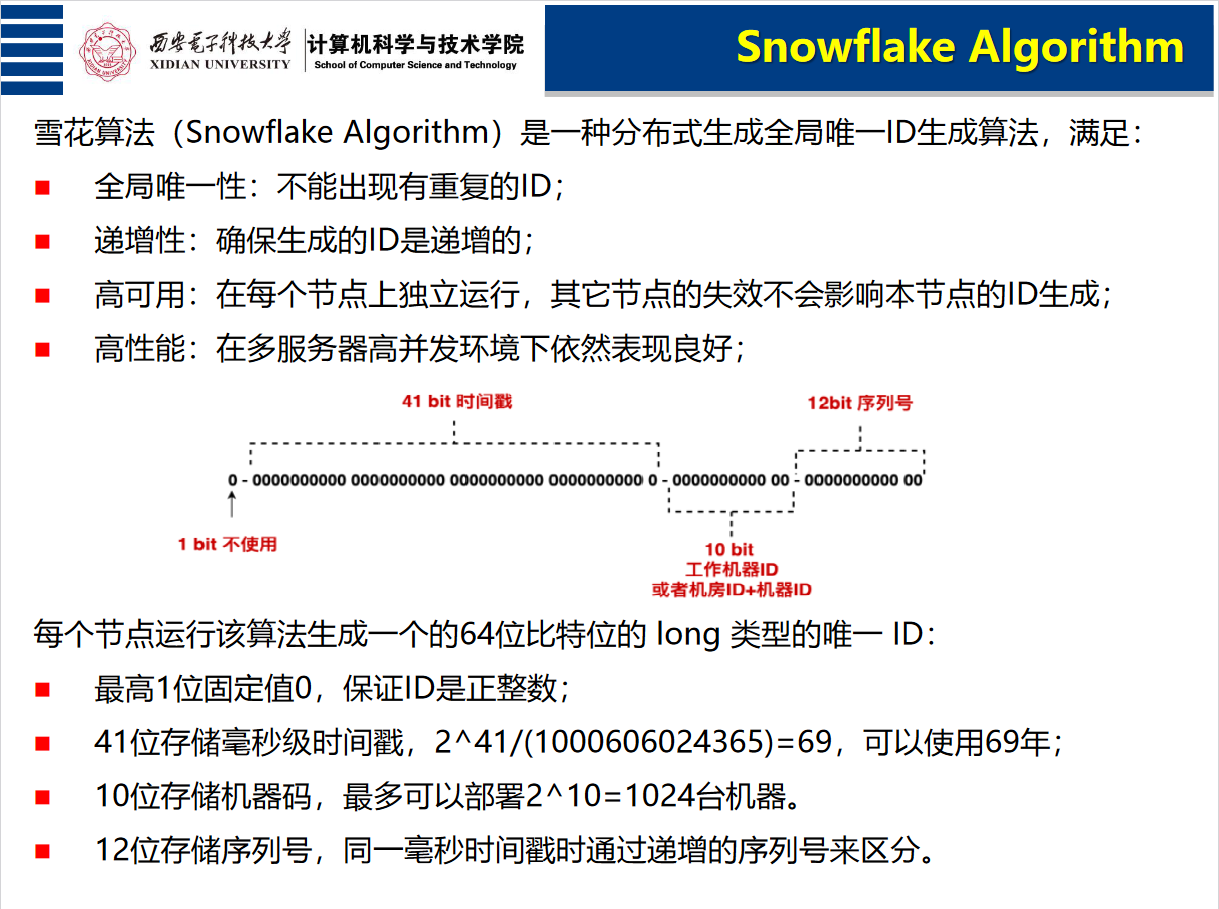
当客户端收到一条消息就将自己的时间戳与该信息的时间戳相比较，更新为更大的值实现因果关系的同一。

具体实现如下

3、传递消息

采用socket传递消息

4、雪花算法



为Server提供信息的ID号

三、实验环境

1、软件版本

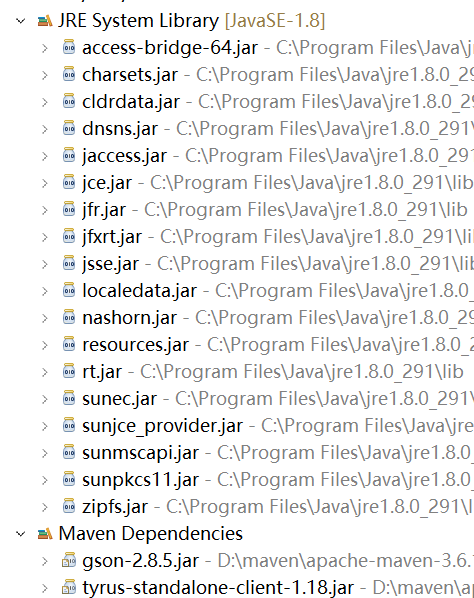
Windows11

Eclipse IDE for Java Developers - 2022-03

Java 1.8

Maven3.6.1

Gson2.8.5



2、pom环境

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.ls</groupId>

<artifactId>p2psystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>com.google.code.gson</groupId>

<artifactId>gson</artifactId>

<version>2.8.5</version>

</dependency>

<dependency>

<groupId>org.glassfish.tyrus.bundles</groupId>

<artifactId>tyrus-standalone-client</artifactId>

<version>1.18</version>

</dependency>

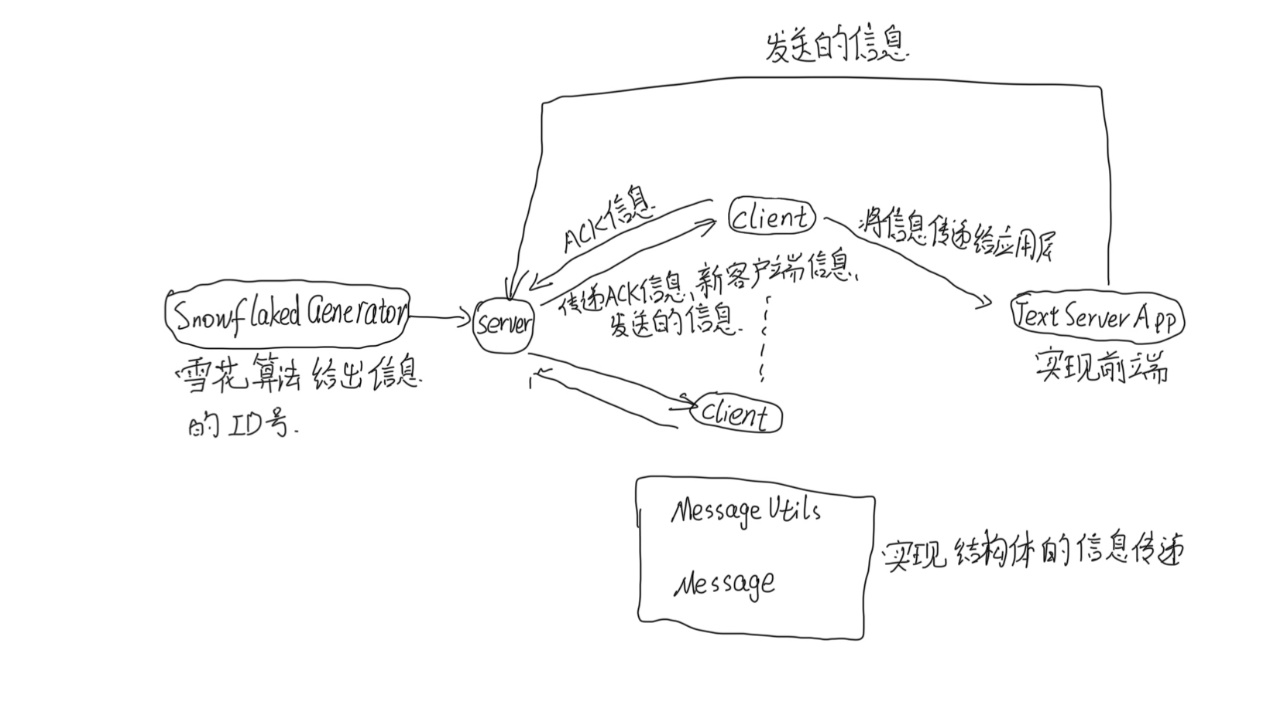
</dependencies>

</project>

```

四、设计思路

1、构架图



2、各个代码的用途

### Client

接受服务端的信息，处理终端和前端UI的信息，将最终信息传递给TextSaverApp进行展示。

### Server

不断接收来自PORT端口的信息，并将信息传给所有的Client节点，值得主义的是由于messageID是唯一的，所以需要进行锁操作，我采用了Java中的synchronized函数对此变量进行锁操作。

### TextSaverAPP

对文本框接收到的信息进行处理并传递给Server服务器；对从Client收到的信息进行展示。

### MessageUtils\Message

对信息进行定义并执行序列化和反序列化操作

### MessageQueue

对信息进行存储、删除、记录次数操作，并利用优先队列对信息进行排序。

### SnowflakeIdGenerator

生成信息的ID号

五、Commands

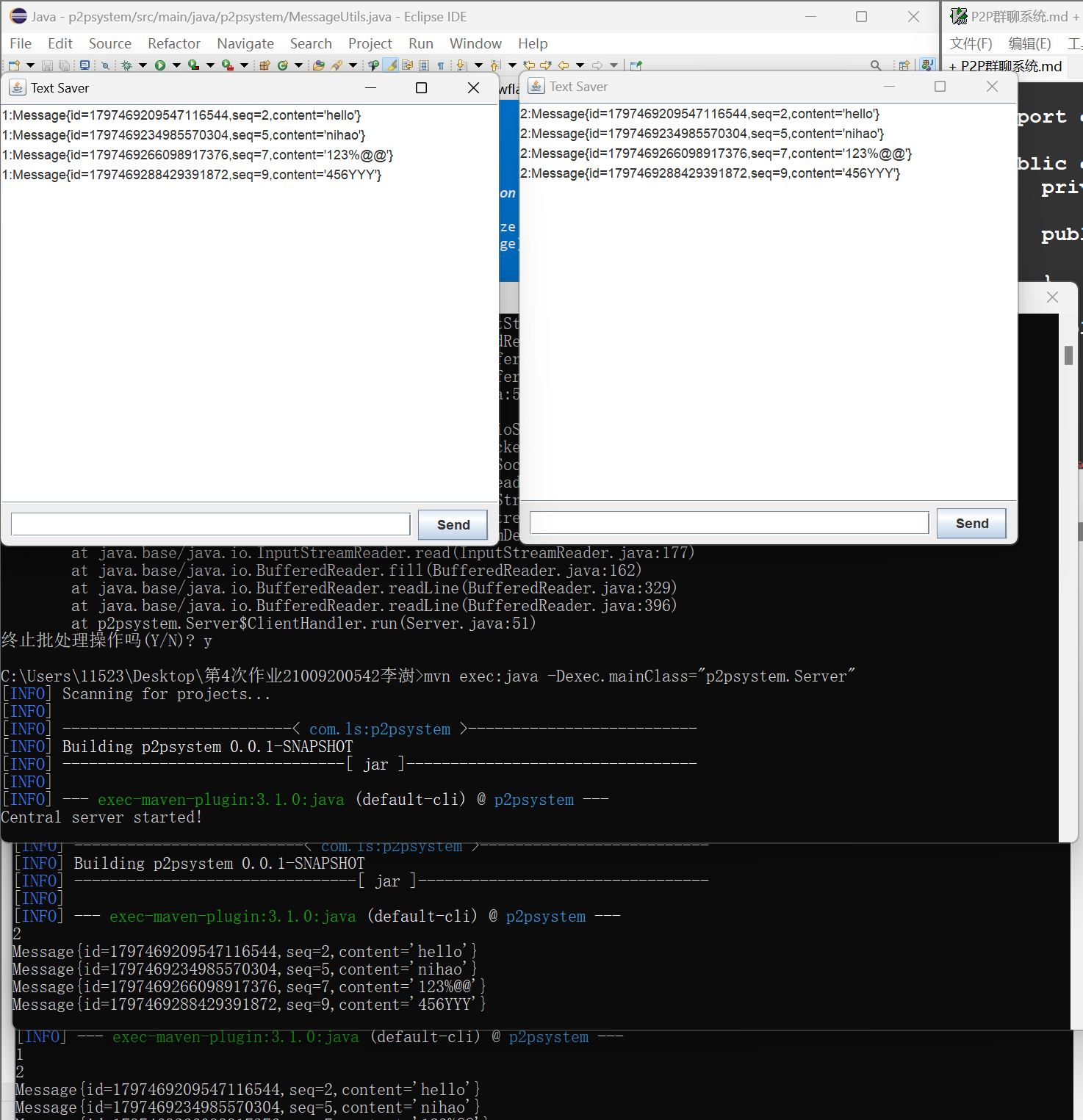
mvn clean

mvn compile

mvn exec:java -Dexec.mainClass="p2psystem.Server"

mvn exec:java -Dexec.mainClass="p2psystem.Client"

六、实现效果



当我们在文本框中输入想发送的信息后，点击Send该信息会在Client各自的显示界面中出现，对于不同文本框输入的值，我们可以发现Id号是严格递增的符合预期，Seq代表的逻辑序号也是递增的，符合我们实现的原理。可以看到在后台的命令指示符中各自的程序也都是良好的运行，符合预期！

七、实现中遇到的问题及解决方案

1、区分不同客户端并动态加入客户端

每有新的客户端加入就向服务端发送信息，记录该客户端的加入并把该信息发送给每个客户端，将目前客户端的数量当作编号

2、死锁问题

对messageID来讲是只能有一个来进行写操作，我们采用了synchronized方法对线程进行锁操作，防止进程之间的冲突

3、序列号归属问题

我注意到如果将序列号的增加与应用端分开将导致书写代码的复杂，因此将序列号的增加融合进应用层并与客户端同步

4、保存消息的队列

采用优先队列的方法，在更新优先队列时重新覆盖实现消息的存储