

电子科技大学

实验报告

学生姓名： 侯晨 学 号：202222081026 指导教师：丁 旭 阳

实验地点： 电子科技大学 实验时间：2023/12/6

一、实验室名称：

Linux 环境高级编程实验室

二、实验项目名称：

综合考查实验

三、实验学时：

4 学时

四、实验目的：

受某公司委托，需要为其开发一套分布式测试平台，该测试平台

要求如下：

(1) 测试任务由测试中心统一管理，当有测试任务时，由管理中心服务器向远端若干测试机进行任务分发；

(2) 测试中心需要跟踪远端测试机的测试任务执行情况，待远

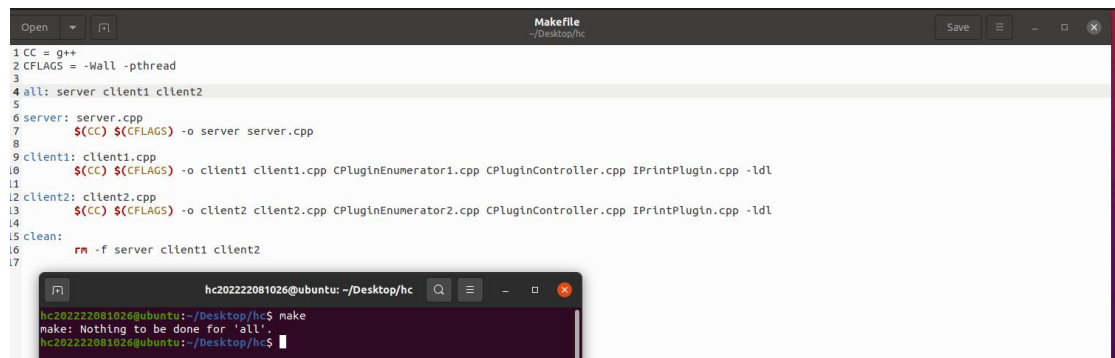
端测试机任务执行完成后，回收所有测试结果并进行分析；

(3) 支持测试中心对远端测试机的测试功能模块的动态更新；

(4) 支持测试中心对远端测试机测试任务的指定迁移。

五、实验内容：

首先进行 make：



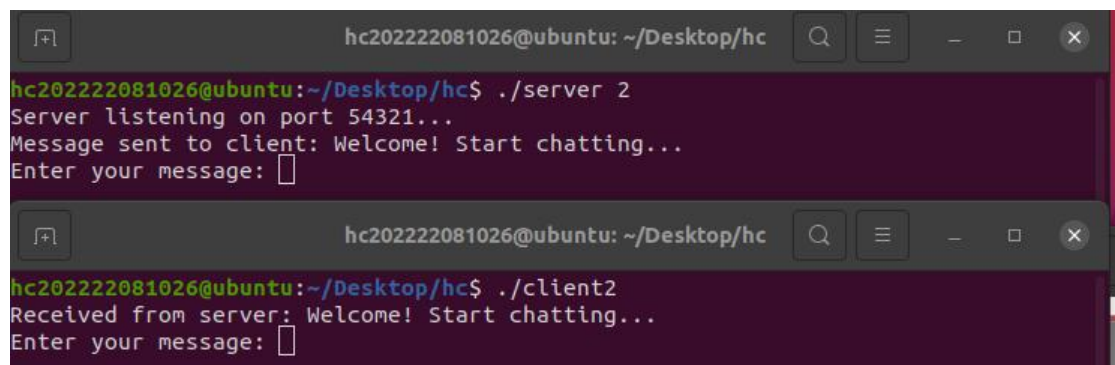
The image shows a Makefile editor window titled 'Makefile' with the following content:

```
1 CC = g++
2 CFLAGS = -Wall -pthread
3
4 all: server client1 client2
5
6 server: server.cpp
7     $(CC) $(CFLAGS) -o server server.cpp
8
9 client1: client1.cpp
10    $(CC) $(CFLAGS) -o client1 client1.cpp CPluginEnumerator1.cpp CPluginController.cpp IPrintPlugin.cpp -ldl
11
12 client2: client2.cpp
13    $(CC) $(CFLAGS) -o client2 client2.cpp CPluginEnumerator2.cpp CPluginController.cpp IPrintPlugin.cpp -ldl
14
15 clean:
16     rm -f server client1 client2
17
```

Below the editor is a terminal window showing the execution of the 'make' command:

```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ make
make: Nothing to be done for 'all'.
hc202222081026@ubuntu:~/Desktop/hc$
```

然后启动服务器与客户端：



The image shows two terminal windows. The top window is running the server:

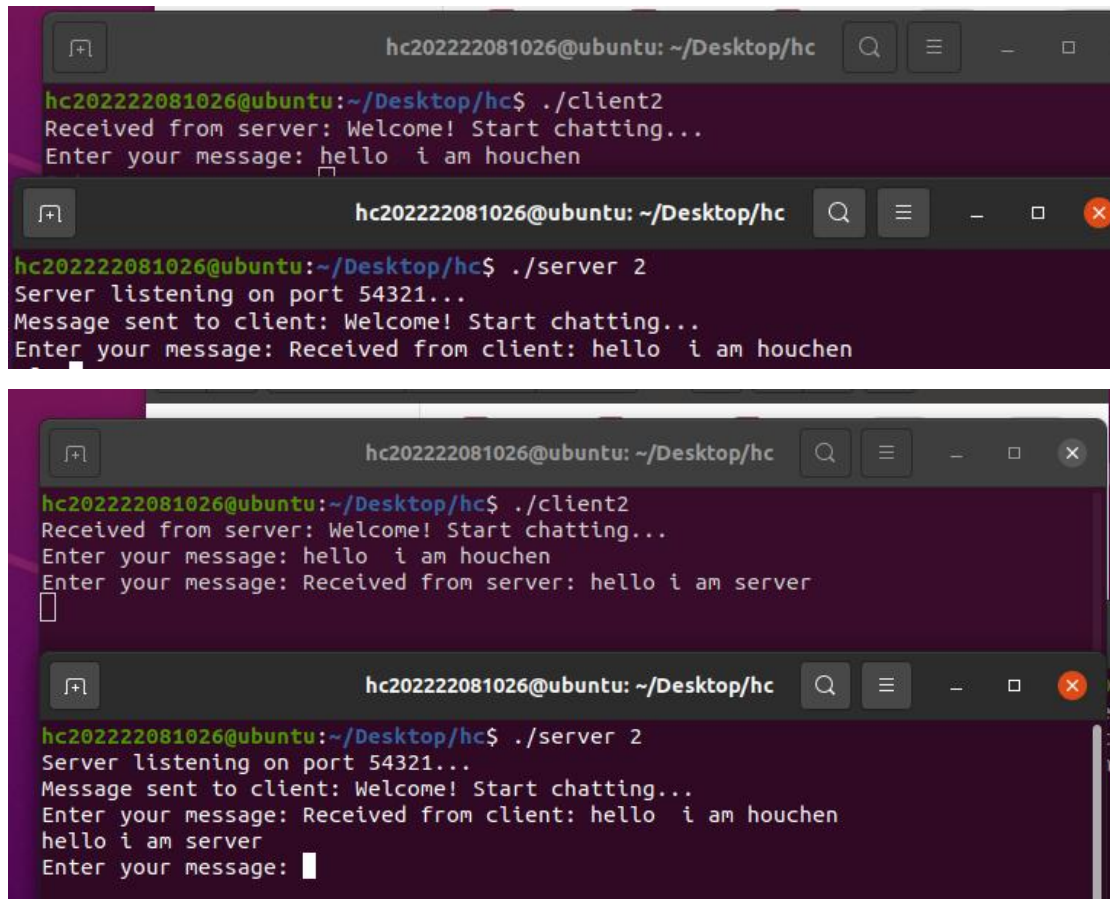
```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./server 2
Server listening on port 54321...
Message sent to client: Welcome! Start chatting...
Enter your message: 
```

The bottom window is running the client:

```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./client2
Received from server: Welcome! Start chatting...
Enter your message: 
```

链接成功！

相互打个招呼~

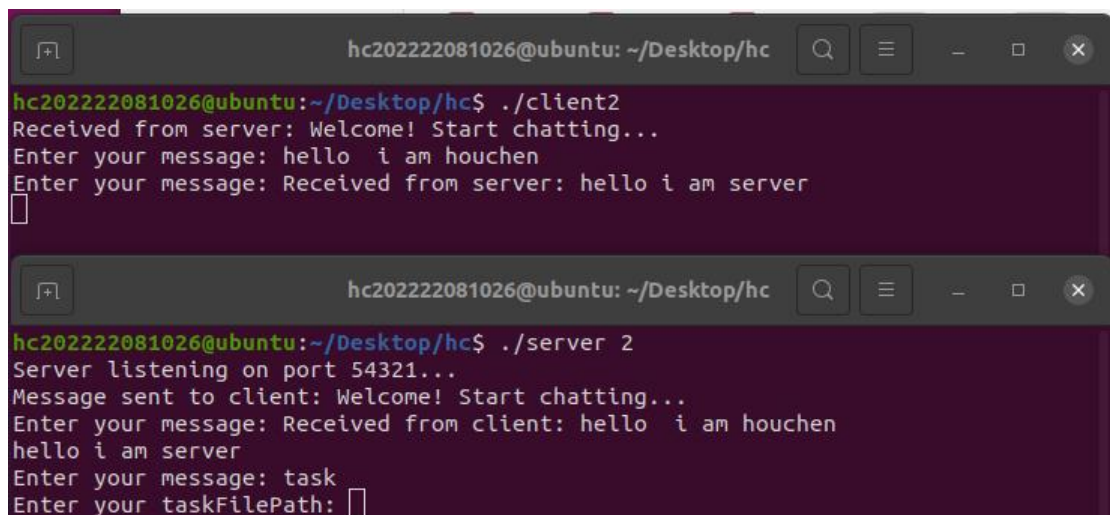


```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./client2
Received from server: Welcome! Start chatting...
Enter your message: hello i am houchen

hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./server 2
Server listening on port 54321...
Message sent to client: Welcome! Start chatting...
Enter your message: Received from client: hello i am houchen
```

确认双方通讯没有问题。即可下发任务。

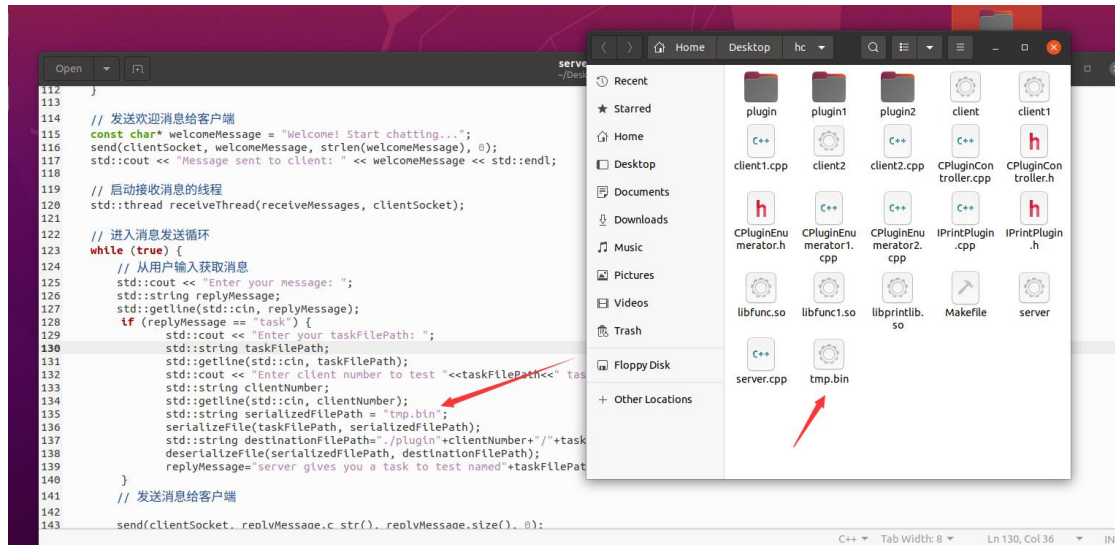
Server 输入 task，进入下发任务模式：



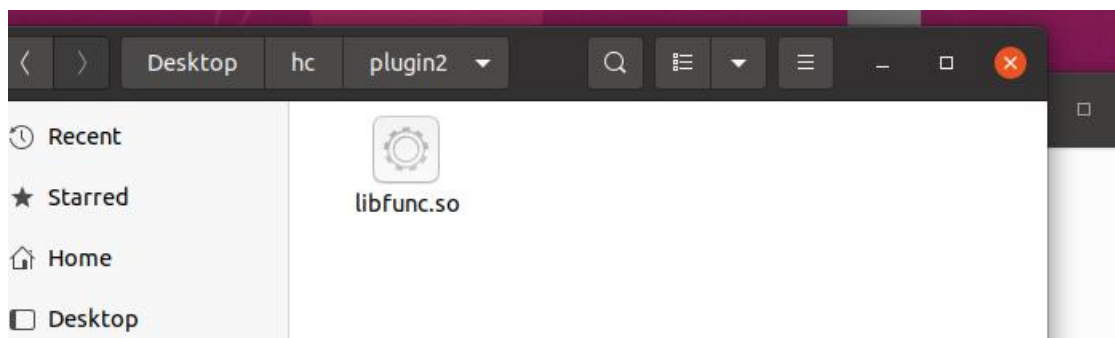
```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./client2
Received from server: Welcome! Start chatting...
Enter your message: hello i am houchen
Enter your message: Received from server: hello i am server

hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./server 2
Server listening on port 54321...
Message sent to client: Welcome! Start chatting...
Enter your message: Received from client: hello i am houchen
hello i am server
Enter your message: task
Enter your taskFilePath:
```

然后输入下发任务的文件地址，这里我们使用实验四中 4.6 的动态链接库的 so 文件作为插件。



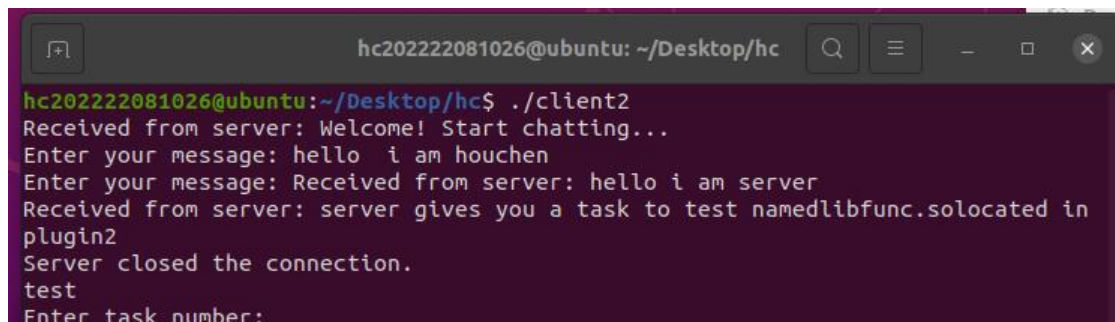
在输入客户机编号后，将这个中间文件反序列化到对应目录下./plugin2/下。



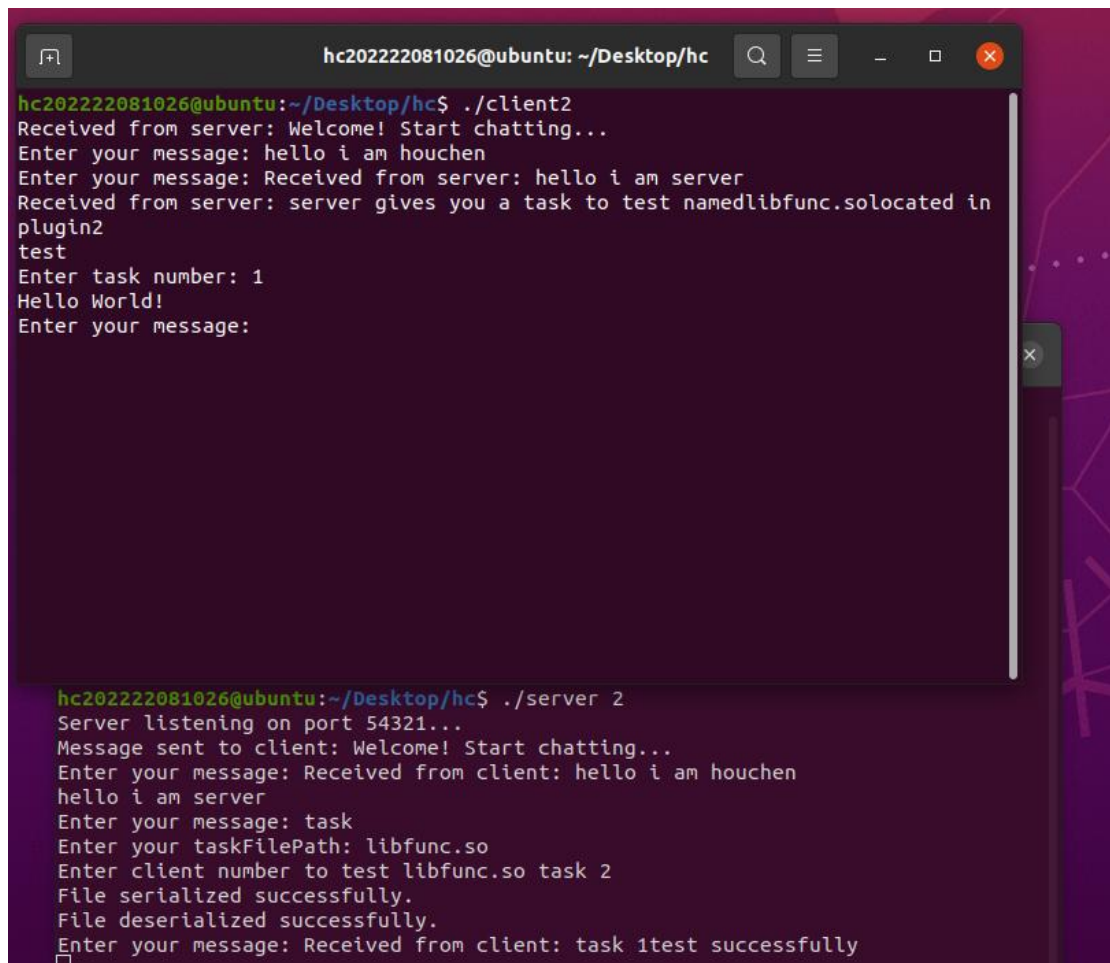
到此实现了：

- (3) 支持测试中心对远端测试机的测试功能模块的动态更新；
- (4) 支持测试中心对远端测试机测试任务的指定迁移。

然后当客户端到了空闲时刻，方便去测试这个任务时，就要输入任务执行关键字“test”



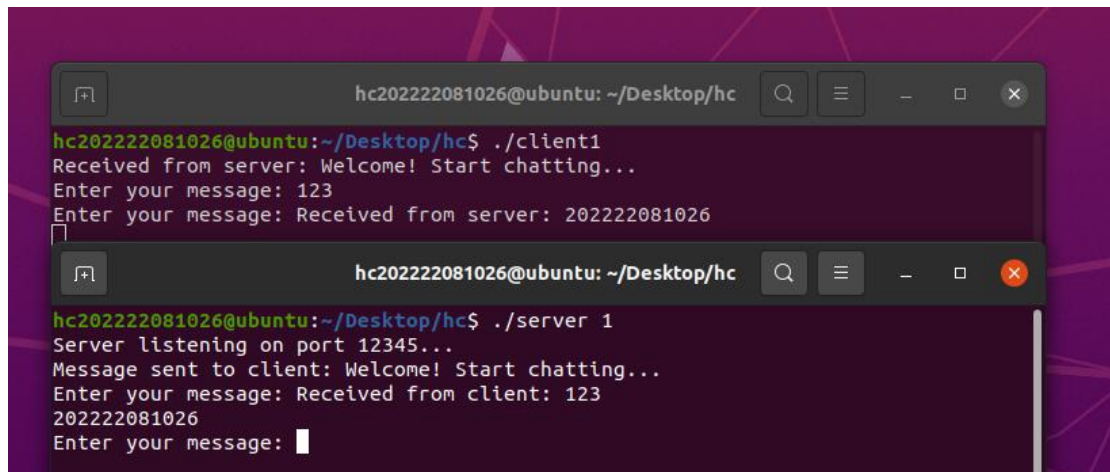
然后输入要执行的任务编号，这里是 1：

A terminal window titled 'hc202222081026@ubuntu: ~/Desktop/hc' showing the execution of a client-server program. The client process (./client2) receives a welcome message, sends 'hello i am houchen', receives a task description, and then sends '1' as the task number, resulting in 'Hello World!'. The server process (./server 2) listens on port 54321, receives the client's message, and successfully serializes and deserializes the task, finally reporting 'task 1test successfully'.

然后我们可以看到客户端 2 成功执行了任务 1，并且输出了 hello world!，同时将执行结果发送给 server，可以看到 server 收到了这个信息 Enter your message: Received from client: task 1test successfully。

然后我们尝试去将任务下发给客户端 1，

打招呼环节，以确定双方的信道通畅：

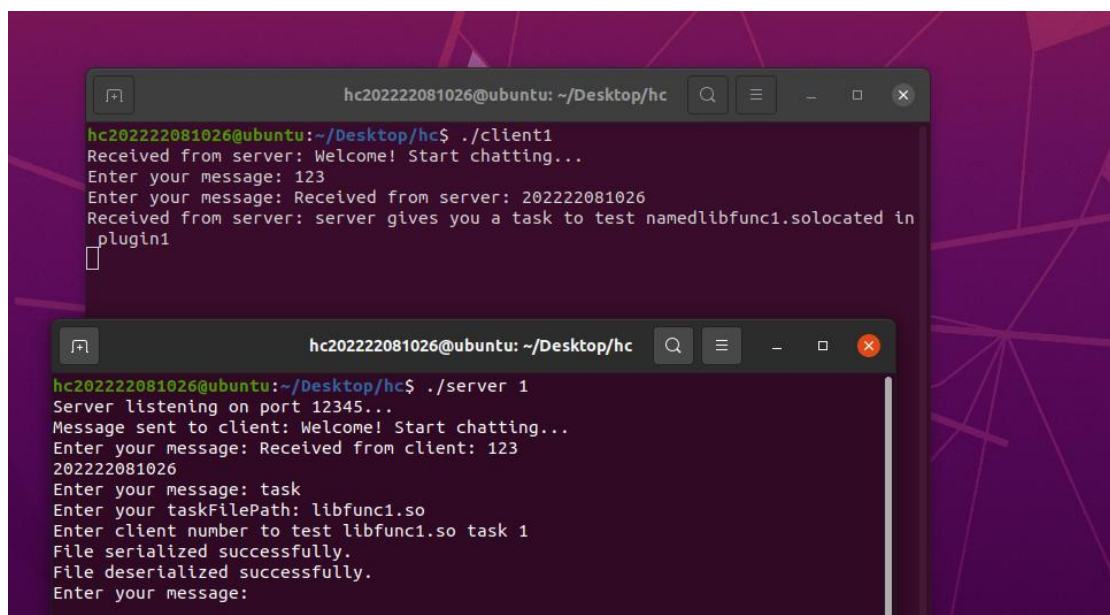


```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./client1
Received from server: Welcome! Start chatting...
Enter your message: 123
Enter your message: Received from server: 202222081026

hc202222081026@ubuntu:~/Desktop/hc$ ./server 1
Server listening on port 12345...
Message sent to client: Welcome! Start chatting...
Enter your message: Received from client: 123
202222081026
Enter your message: 
```

然后服务器发下任务，让客户端 1 完成任务 2，及实验四中 4.6 的 libfunc1.so 任务，内容是输出 hello china！

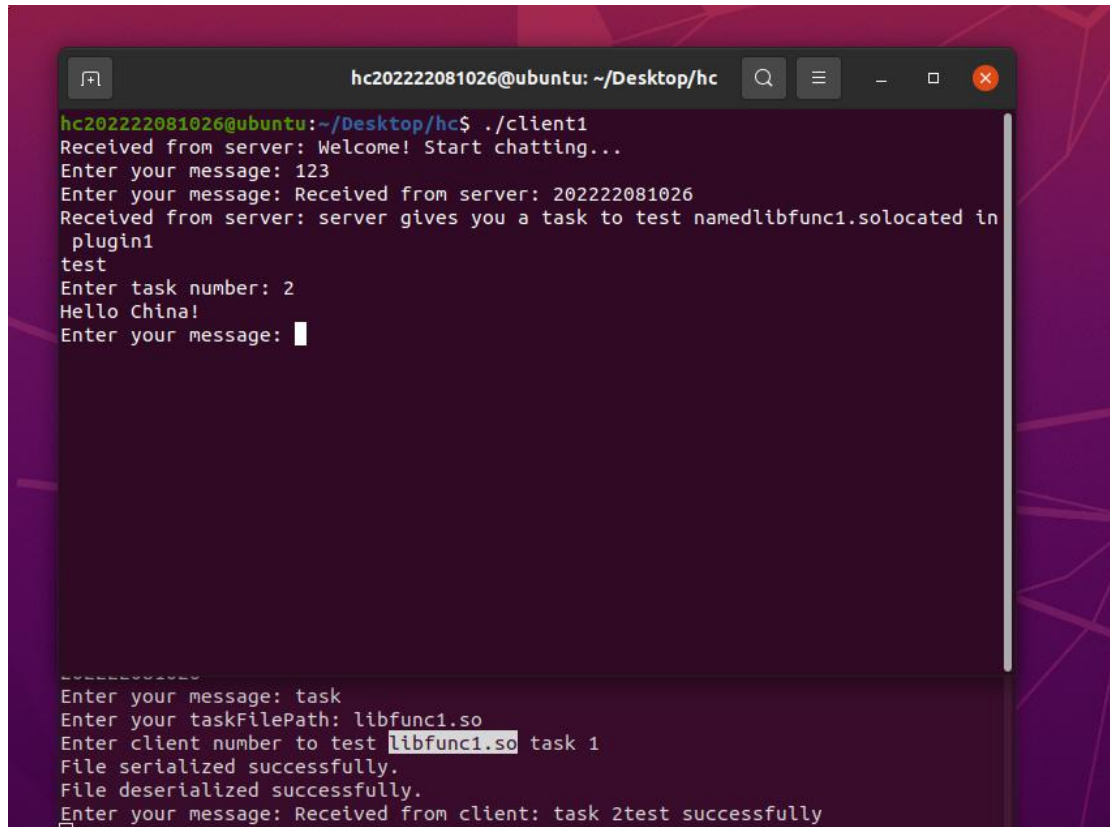
于服务器端输入任务关键字 task



```
hc202222081026@ubuntu:~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./client1
Received from server: Welcome! Start chatting...
Enter your message: 123
Enter your message: Received from server: 202222081026
Received from server: server gives you a task to test named libfunc1.so located in plugin1

hc202222081026@ubuntu:~/Desktop/hc$ ./server 1
Server listening on port 12345...
Message sent to client: Welcome! Start chatting...
Enter your message: Received from client: 123
202222081026
Enter your message: task
Enter your taskFilePath: libfunc1.so
Enter client number to test libfunc1.so task 1
File serialized successfully.
File deserialized successfully.
Enter your message: 
```

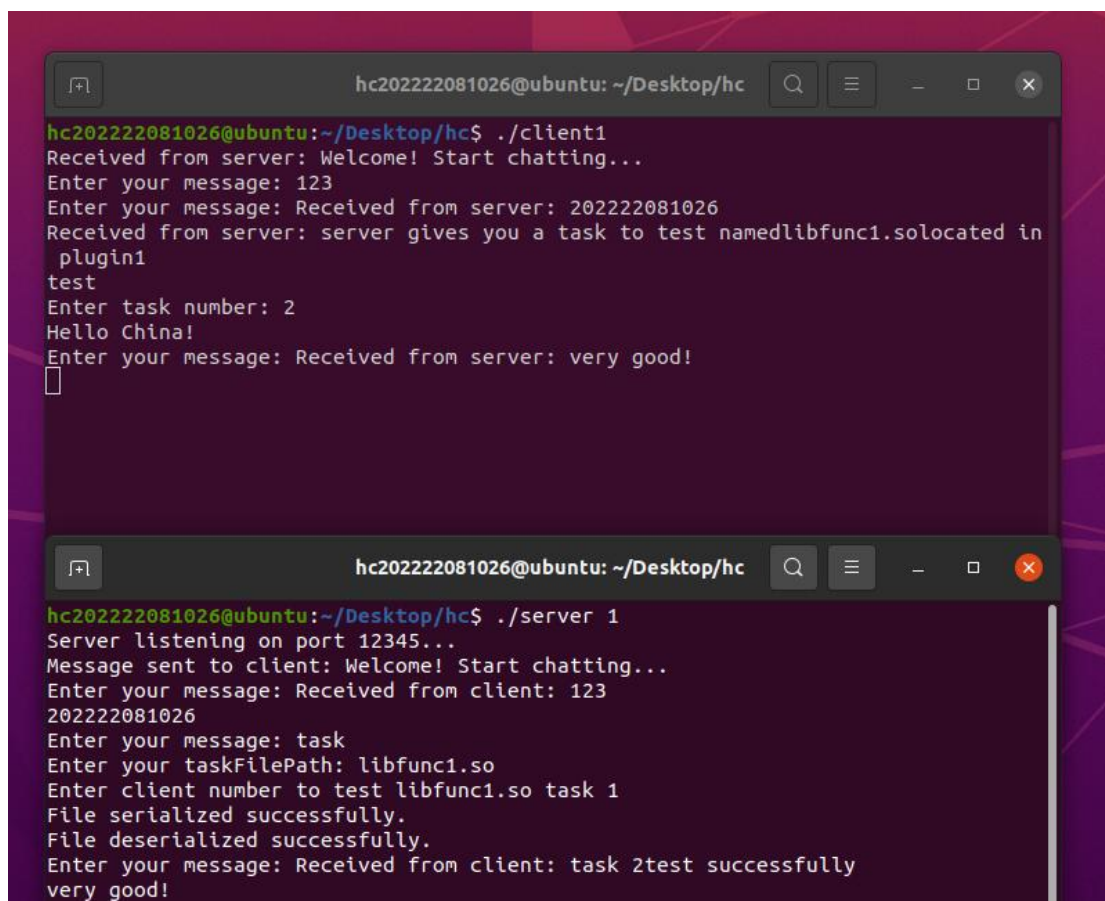
同时客户端 1 在空闲时间执行任务 libfunc1.so，这里的编号为 2，重复上述操作：



```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./client1
Received from server: Welcome! Start chatting...
Enter your message: 123
Enter your message: Received from server: 202222081026
Received from server: server gives you a task to test named libfunc1.so located in
plugin1
test
Enter task number: 2
Hello China!
Enter your message:
Enter your message: task
Enter your taskFilePath: libfunc1.so
Enter client number to test libfunc1.so task 1
File serialized successfully.
File deserialized successfully.
Enter your message: Received from client: task 2test successfully
```

可以看到客户端 1 成功执行了任务 2，输出了 hello china！ 并且将结果返回给服务器端。

服务器端对任务的执行结果非常的满足，进行分析过后给客户端 1 发送了 very good 消息！



```
hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./client1
Received from server: Welcome! Start chatting...
Enter your message: 123
Enter your message: Received from server: 202222081026
Received from server: server gives you a task to test namedlibfunc1.solocated in
plugin1
test
Enter task number: 2
Hello China!
Enter your message: Received from server: very good!
[]

hc202222081026@ubuntu: ~/Desktop/hc
hc202222081026@ubuntu:~/Desktop/hc$ ./server 1
Server listening on port 12345...
Message sent to client: Welcome! Start chatting...
Enter your message: Received from client: 123
202222081026
Enter your message: task
Enter your taskFilePath: libfunc1.so
Enter client number to test libfunc1.so task 1
File serialized successfully.
File deserialized successfully.
Enter your message: Received from client: task 2test successfully
very good!
```

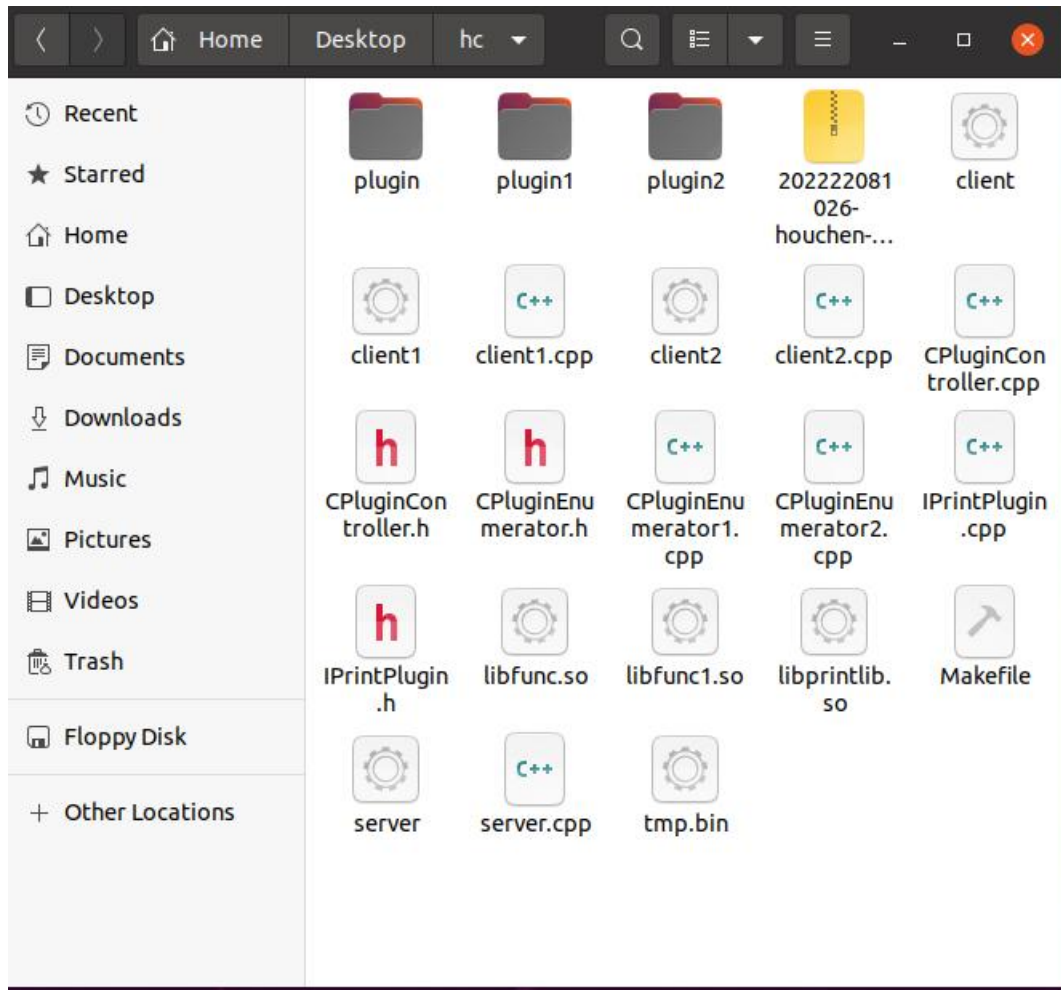
到此，代码完成了：

（1）测试任务由测试中心统一管理，当有测试任务时，由管理中心服务器向远端若干测试机进行任务分发；

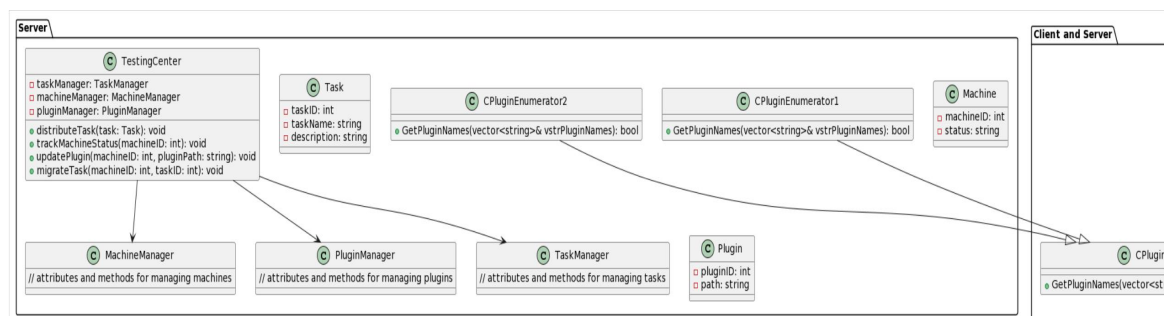
（2）测试中心需要跟踪远端测试机的测试任务执行情况，待远端测试机任务执行完成后，回收所有测试结果并进行分析；

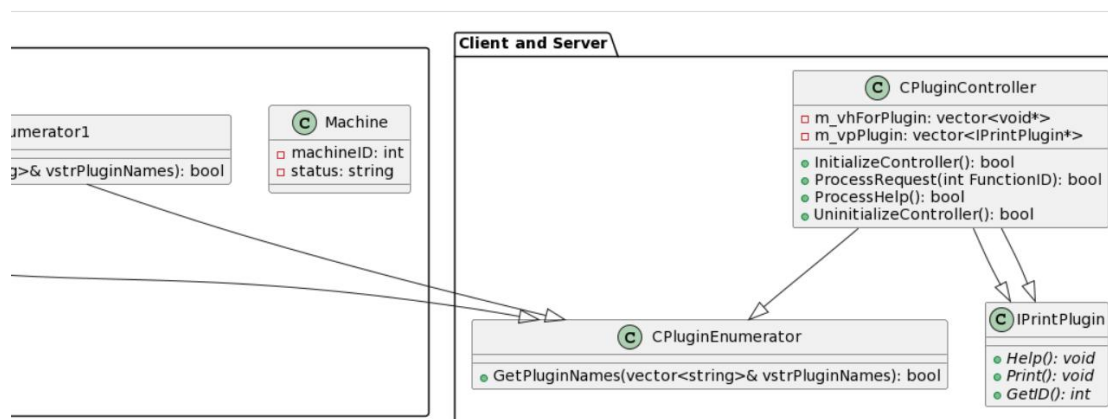
实验圆满完成。

文件截图：



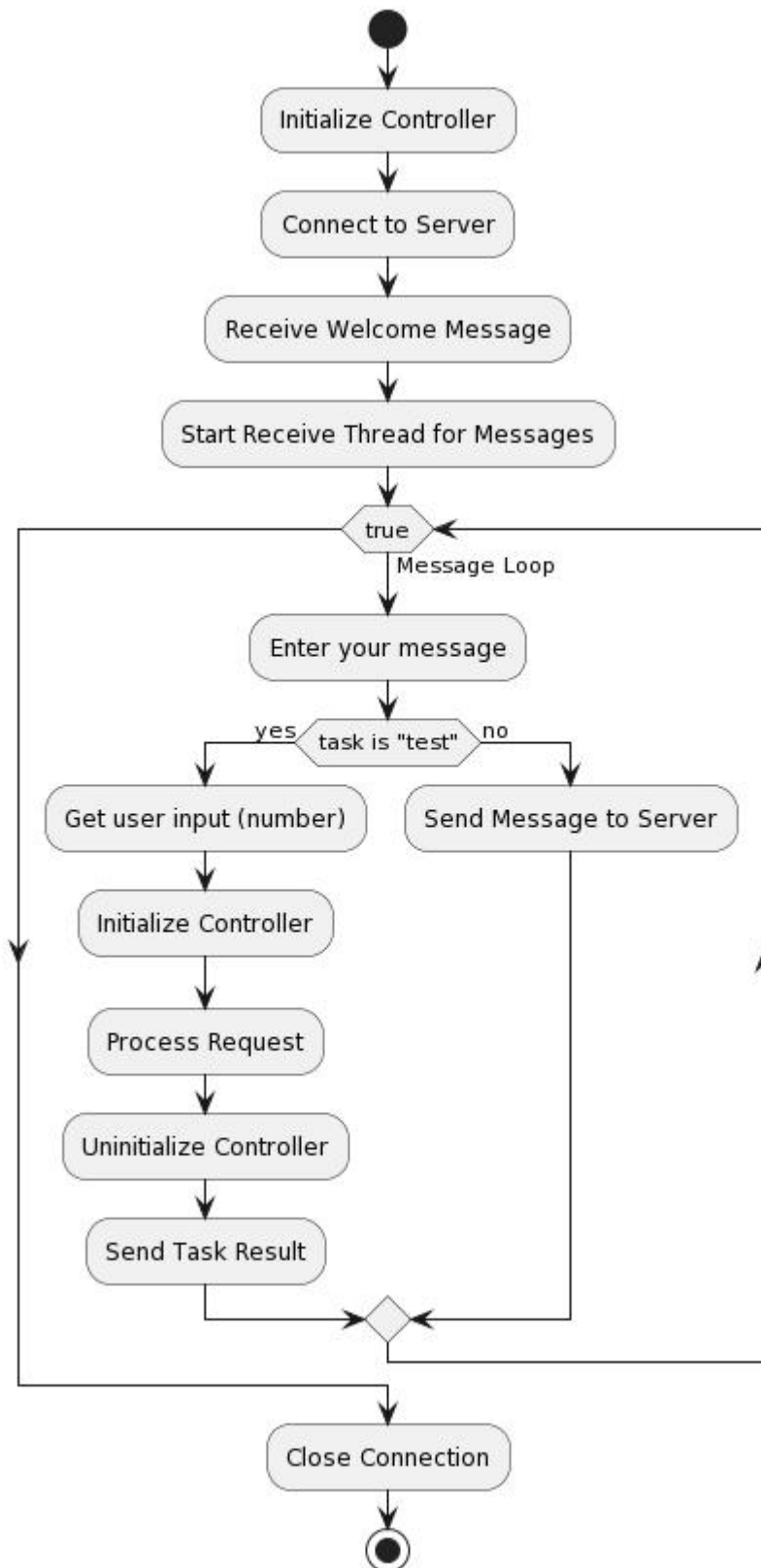
为了更好的理解代码的逻辑，下面是类图：



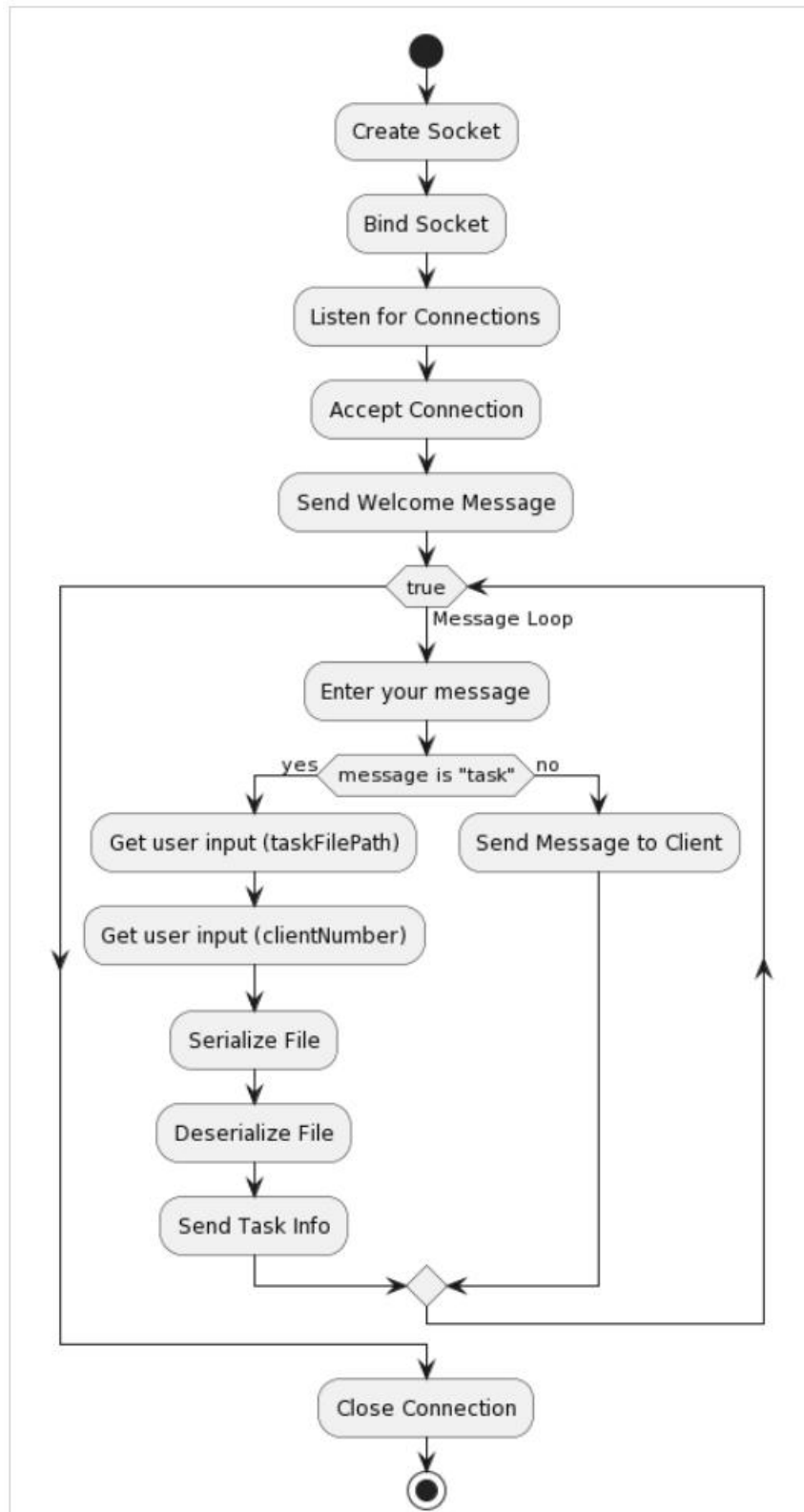


流程图：

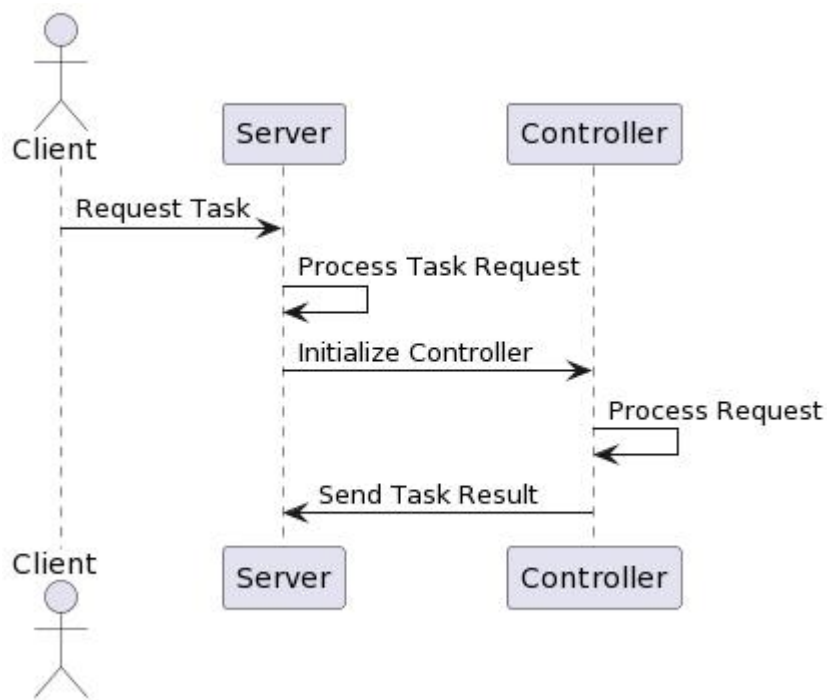
Client



Server:



时序图：



六、实验步骤：

1. 系统架构和组件说明

详细描述测试平台的系统架构和各个组件的功能，包括：

测试中心服务器（Manager）

远端测试机（Controller）

插件（Plugin）

2. 编译和运行

使用提供的 `makefile` 文件编译服务器、客户端 1 和客户端 2。分别运行服务器、客户端 1 和客户端 2。

3. 任务分发和执行

在测试中心服务器上创建测试任务，指定测试机和测试插件。测试中心服务器将任务分发给相应的远端测试机。远端测试机接收任务，执行测试。

4. 任务状态跟踪

测试中心服务器跟踪远端测试机的任务执行状态。当远端测试机完成任务时，测试中心服务器回收测试结果。

5. 动态更新插件

在测试中心服务器上更新测试插件。测试中心服务器将更新通知发送给相应的远端测试机。

远端测试机接收更新通知，动态加载新插件并执行任务。

6. 测试任务迁移

在测试中心服务器上指定迁移任务。测试中心服务器将迁移任务通知发送给相应的远端测试机。远端测试机接收迁移任务通知，暂停当前任务并执行迁移。

7. 结果分析

测试中心服务器收集所有远端测试机的测试结果。对测试结果进行分析和汇总。

8. 清理和关闭

测试完成后，关闭服务器、客户端 1 和客户端 2。

七、总结及心得体会：

总结

通过本次项目/实验，我学到了许多关于分布式系统和网络编程的知识。在这个项目中，我成功地实现了一个分布式测试平台，该平台具有以下主要特征：

任务管理与分发：测试中心服务器能够有效地管理和分发测试任务，通过网络与远端测试机进行通信，实现任务的分布式执行。

任务状态跟踪：我成功地设计了任务状态跟踪系统，使得测试中心服务器能够实时监控远端测试机的任务执行情况。

动态更新插件：实现了插件的动态更新功能，测试中心服务器能够向远端测试机推送新的插件，实现系统的可维护性和可扩展性。

测试任务迁移：设计了测试任务迁移系统，使得测试中心服务器能够根据需要将任务从一个测试机迁移到另一个测试机，提高系统的灵活性。

心得体会

在项目中，我面临了一些挑战，其中一些主要包括：

网络通信：通过网络进行通信是一个复杂的任务，我深入理解了套接字编程和网络协议，学到了如何有效地在分布式系统中传递数据。

动态更新和迁移：实现插件的动态更新和测试任务的迁移涉及到动态加载和卸载模块的知识，这对我来说是一个全新的领域。

系统设计：需要仔细设计系统架构，确保各个组件能够协同工作。

同时，需要考虑系统的健壮性和容错性，以防止单点故障和数据丢失。

通过克服这些挑战，我不仅加深了对分布式系统的理解，还提高了在网络编程和系统设计方面的技能。这次经历让我对实际项目的开发和维护有了更深刻的认识，为我的职业发展打下了坚实的基础。

报告评分:

指导教师签字: