Описание задания

Реализация клиентского и серверного модулей, взаимодействующих посредством вымышленного бинарного протокола, с целью обеспечения примитивного вызова удаленных функций.

Язык программирования: Java
Платформа: J2SE 1.8
Протокол транспортного уровня: TCP

Описание протокола

В протоколе присутствуют команды и ответы на них. Команда передается клиентом серверу, в ответ на которую сервер пересылает ответ. Каждая команда идентифицируется уникальным, в рамках сессии, порядковым номером. Ответ содержит порядковый номер инициировавшей его команды. За генерацию и уникальность порядковых номеров ответственность несет клиент, сервер лишь прописывает в ответе тот порядковый номер, который получил в запросе.

Формат команды/ответа

Команда должна содержать обязательные поля:
- Порядковый номер (уникальный в сессии)
- Название сервиса
- Название функции (метода)
- Список объектов (параметры функции)

Ответ должен содержать:
- Порядковый номер команды (тот который был в запросе)
- Объект - результат выполнения метода

Порядковый номер, в данном случае, передается для того, чтобы была возможность через одно TCP соединение вызывать несколько методов одновременно (то есть можно вызвать один метод, а затем второй, не дождавшись ответа на первый, при этом оба метода буду выполнятся на сервере параллельно, и после их выполнения, клиент получит оба ответа).

Серверный модуль

Работает в многопоточном режиме. Обеспечивает прием команд от нескольких клиентских модулей. Логирует принятые команды и посланные ответы (объекты достаточно логировать в виде строки используя метод Object.toString()). Освобождение ресурсов происходит при TCP дисконнекте клиента.
Объектная модель приходящих команд и их обработки: Ровно один поток на accept клиентов, по одному потоку на каждого клиента для приема команд, а для обработки команд используется пул потоков. (под потоком здесь понимается поток исполнения - Thread)

Клиентский модуль

Запускает несколько потоков исполнения, в которых через одно соединения с некоторой регулярностью посылаются команды серверному модулю, ожидается ответ, при этом происходит логирование с какими параметрами был вызов команды, и какой результат вернулся.

Требования:

- Логирование должно производится при помощи log4j (http://jakarta.apache.org).
- Клиент обязательно должен создавать такую ситуацию, чтобы вызов нескольких методов происходил одновременно.
- При запуске сервера с командной строки, указывается локальный порт, который будет слушать сервер.
  Например так:
  java -cp classes com.mytest.MyServer 2323

  При этом сервер должен зачитывать конфигурационный файл (server.properties) такого вида:
  <Название сервиса>=<Имя класса>
  <Название сервиса>=<Имя класса>
  ....

  То есть на сервере может быть несколько сервисов с разными именами и с разным набором методов.
  При старте сервер создает ровно по одному объекту для каждого сервиса (указанного класса), и запоминает их по имени.
  Именно это название сервиса и будет использовать клиент при удаленном вызове.
  Для демонстрации достаточно будет одного сервиса.

- Клиент должен предоставлять некоторый интерфейс для удаленного вызова, чтобы его можно было легко использовать как библиотеку.
  Предлагается сделать примерно такой интерфейс для удаленного вызова:

```
public class Client {
    public Client(String host, int port) {...}
    public Object remoteCall(String serviceName, String methodName, Object[] params) {...}
}
```

  Метод remoteCall не должен быть синхронизованным, чтобы его можно было использовать из нескольких потоков одновременно.
  Реализация этого клиента должна устанавливать соединение с сервером, и при вызове метода remoteCall посылать команду на сервер и дожидаться ответа, метод должен быть блокирующим, то есть пока не пришел ответ от сервера возврат из метода не происходит.

- Специально обрабатывать на сервере ситуацию, когда запросили несуществующий сервис или метод, или указали параметры неправильных типов, или неправильное количество параметров, и передавать это клиенту. А на стороне клиента кидать специальный Exception.
- Специальным образом передавать возвращаемое значение в случае когда функция void.

Подсказки:

- Для передачи объектов использовать стандартную сериализацию.
- Для вызова методов сервиса на стороне сервера использовать reflection.
- Конфигурационный файл зачитывать при помощи Properties.
- Если возникнут проблемы с логированием можно использовать System.out.println()
- Использовать java.util.concurrent
- Для простоты - параметрами функции не могут быть примитивные типы, а могут быть только объекты реализующие Serializable.

Дополнительно:

При проверке задания будет создан класс

```
public class Service1 {

    public void sleep(Long millis) {
        Thread.sleep(millis.longValue());
    }

    public Date getCurrentDate() {
        return new Date();
    }

}
```

Этот класс будет зарегистрирован как сервис с именем service1 на сервере.

Клиент будт выглядеть примерно так:

```
public class MyClient {

    public static void main(String args[]) {
        Client client = new Client("localhost", 2323);
        for(int i=0;i<10;i++) {
            new Thread(new Caller(client)).start();
        }
    }

    private static class Caller implements Runnable {
        private Logger logger = Logger.getLogger(Caller.class);
        private Client c;
        public Caller(Client c) {
```

```
            this.c = c;
        }
        public void run() {
            while(true) {
                c.remoteCall("service1", "sleep", new Object[] {new Long(1000)});
                logger.info("Current Date is:" + c.remoteCall("service1", "getCurrentDate", new Object[] {}));
            }
        }
    }

}
```