文章编号:1671-4598(2022)12-0085-06 DOI:10.16526/j.cnki.11—4762/tp.2022.12.013 中图分类号:TP311.52 文献标识码:A

基于 B/S 的双 CCD 相机精度靶测控系统软件架构设计

高嘉韵[1]，倪晋平[1]，徐 飞[2]
(1. 西安工业大学 光电工程学院，西安 710000；2. 西安工业大学 计算机学院，西安 710000)

摘要：上位机软件是双 CCD 相机精度靶测控系统中非常重要的一部分；针对精度靶测控系统软件存在后期功能拓展难、维护效率低、使用升级复杂等问题，先进行了对该类软件的调查研究，然后分析上位机测控软件功能需求，根据软件的需求分析，设计了一种基于 B/S 架构的精度靶测控系统上位机软件实现方案；软件架构采用了浏览器和服务器方式专有的三层结构，在中间层设计了 TCP 服务器，提高软件的功能拓展性，降低了实现代码的耦合性；采用了 Netty 通信技术框架及其非阻塞技术实现了 TCP 服务器底层，设计了专有的解码方法接收下位机数据及发送控制指令；通过模拟测试，能够满足下位机频繁发送数据，上位机稳定接收和可靠运行的功能需求，软件的结构设计合理，扩展性和维护性良好。

关键词：B/S 架构；测控系统；TCP 服务器；Netty；上位机软件

Design of Software Architecture of Double CCD Camera Precision Target Measurement and Control System Based on B/S

GAO Jiayun[1], NI Jinping[1], XU Fei[2]
(1. School of Optical Engineering, Xi'an Technological University, Xi'an 710000, China;
2. School of Computer Science, Xi'an Technological University, Xi'an 710000, China)

Abstract: Upper computer software is a very important part of dual CCD camera accuracy target measurement and control system. the software of accuracy target measurement and control system has the problems of late functional expansion, low maintenance efficiency, complex use and upgrade etc. Firstly, the survey and research on this kind of software are finished, then the upper computer measurement and control software functional requirement are analyzed, according to the software requirement and analysis, the control system upper computer software of accuracy target measurement scheme based on B/S architecture is realized. The software architecture has three-layer proprietary structure by using the browser and server method, TCP server is designed in the middle layer to extend the software function, and reduce the code coupling. The framework of Netty communication technology and non-blocking technology are used to realize the bottom layer of the TCP server, a proprietary decoding method is designed to receive the data from the lower computer and send the control instruction. Through the simulation test, it can meet the functional requirements of frequent data transmission by the lower computer, stable reception and reliable operation by the upper computer, and the software has a reasonable structural design and the good scalability and maintainability.

Keywords: B/S architecture; measurement and control system; TCP server; Netty; upper computer software

0 引言

靶场试验测试作为武器研发、生产和测试的重要一环，主要对武器的各种性能指标进行的检测，以确定武器是否能够达到预期的指标要求。立靶密集度测试是枪弹生产检验的关键参数，双线阵交会精度靶密集度参数测量系统因较高的测量精度、使用方便以及相对较低的使用成本而被广泛使用[1-3]。系统包括三个部分，上位机软件、数据采集处理下位机软件和精度靶采集系统，其中上位机软件就是本文要设计实现的。

目前国内外对于该领域的测控软件通常采用 C/S（客户端/服务器）架构实现的，软件形成两层结构，服务器负责处理具有复杂逻辑的业务功能，客户端实现用户界面和简单的数据处理。这种架构软件实际使用中一般需要专门的客户端软件安装使用安装，处理出现的问题难度较大，也难以功能扩展，可能会出现开发一个全新的软件的情况[4]。如丁力等[5]设计 C/S 架构的控制管理系统有如下缺点：系统扩展性差，业务变更不灵活；兼容性差，开发工具不兼容，独立客户端安装；维护和升级成本高。

收稿日期:2022-08-30； **修回日期**:2022-09-29。

作者简介:高嘉韵(1997-),男,陕西榆林人,硕士研究生,主要从事 Java 软件开发方向的研究。
通讯作者:倪晋平(1965-),男,陕西乾县人,博士,教授,博士生导师,主要从事武器靶场测试方向的研究。

引用格式:高嘉韵,倪晋平,徐 飞.基于 B/S 的双 CCD 相机精度靶测控系统软件架构设计[J].计算机测量与控制,2022,30(12):85-90,97.

结合上述问题，精度靶测试系统上位机软件采用拓展性和维护性方便、简单、共享性强和访问简单的 B/S（浏览器/服务器）架构开发[6-7]。将用户界面等在浏览器端实现，为了不仅能简化系统的维护，还可以降低使用成本，将所有的功能逻辑都集中到了服务器端。

1 软件需求分析

1.1 系统概述

双线阵交会精度靶测控系统整体上可分为精度靶测控系统软件上位机部分、数据采集处理下位机部分和精度靶采集硬件系统三部分，如图 1 所示。其中将上位机部分习惯称为控制端，下位机和精度靶硬件采集系统整体称为测量端，从使用角度来说，一般情况下上位机和测量端之间间隔 100 米左右，通过网线进行连接，实现远程指令控制和数据传输的使用需求。

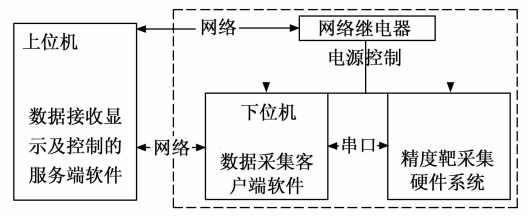

图 1 精度靶测控系统整体结构图

精度靶测控系统功能主要是实现对整个试验流程进行控制，需要上位机执行相应的控制命令，判断当前的任务或流程执行情况，以及是否进入下一个控制命令[8]；其次是接收下位机传输的数据。这里控制方面主要需要实现远程控制网络继电器操作以及任务开始、结束的控制，数据方面主要是对下位接收的数据进行一个直观显示和存储。

1.2 功能需求

精度靶系统对于上位机的要求是能够及时准确的接收下位的数据，同时对整个试验流程进行功能控制。从上述情况来看，上位机主要实现功能分为以下几个：

1）电源开关控制：上位机软件启动后，检测与网络继电器连接状态，向网络继电器发送控制命令，从而实现远程控制下位机和精度靶硬件部分的电源开关。

2）状态检测：下位机软件和精度靶硬件采集部分启动后，下位机软件获取和硬件的连接状态，将连接状态发送到上位机软件进行显示，方便使用者知道下位机和精度靶硬件采集部分的连接情况，便于出现异常状态问题的快速解决。

3）试验参数设置：该功能是精度靶测试试验前的相关参数初始化设置，包含试验任务的名称、组数、每组的次数和试验操作人员的相关信息设置，以及测试用的弹种数据包含弹径、弹长和弹速的设置。

4）测试试验：该功能包含了多个子功能，开始试验、数据显示、试验密集度参数计算、试验组数和结束试验。开始试验是使用者操作上位机向下位机发送开始试验的控制命令，下位机接收到指令后进入测试状态；数据显示是上位机软件把从下位机接收到的试验数据结果进行展示；试验密集度参数计算是由上位机完成的，在试验中，上位机接收到试验数据大于等于三的时候，程序会自动进行参数结果的计算，也可以手动选择计算；试验组数是在试验中进行试验组别切换；试验结束同开始试验，用户操作上位机向下位机发送试验结束指令，且该功能不设置前置条件。

5）历史数据：在试验中，上位机会把从下位机接收到的试验结果和密集度参数计算的结果全部保存到本地数据库，并且以任务创建时间为名称区别存储，通过任务创建时间或名称检索对应的历史试验数据，还可以选择所需试验任务数据导出到 Excel。

2 软件设计

2.1 整体设计

基于 B/S 架构的上位机软件的整体结构如图 2 所示，依据整体框图的结构层次，进一步阐述层与层之间模块的设计和信息的交互。上位机软件整体分为上层、中层和下层，表示层位于最上层，是为了与用户进行交互的层面，用户在界面触发相关控制命令的按钮操作，随之用户的请求及数据由表示层接收，接着对数据进行处理，最后显示处理后的数据。业务逻辑层作为纽带存在于表示层和数据之间，需要依据业务的具体功能需求，实现相对应的业务功能。该软件系统数据来自与下位机的通信获得，所以设计在中间层需要搭建 TCP 服务器进行数据交互。数据层主要实现对数据库的增删改查，用于 TCP 服务器存储从下位机获取来的数据以及向历史数据查询功能模块提供数据支持。

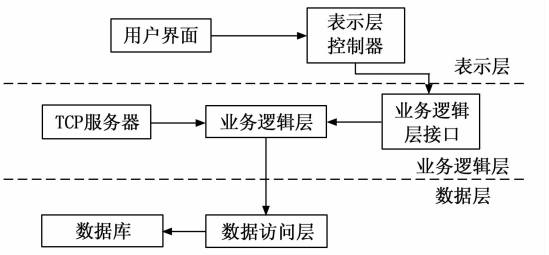

图 2 上位机软件系统整体框图

服务器和浏览器作为 B/S 软件架构的核心。其中浏览器作为用户直接操作的层面，主要包含功能操作、发送请求和获取数据，最后通过浏览器展示出来。服务器承载了大多数的业务分析数据处理，也是真正的实现界面应用层各模块的业务逻辑，例如在和数据库部分的交互中，专门设计了数据缓存处理，数据查询的优化等。更重要的是，在与下位机的交互中，提供可靠的数据传输，较好的传输性能。

根据软件系统整体设计数据流图，如图 3 所示，对上位机软件的功能需求进行分析整理，以技术出发的角度，上位机软件服务器部分设计实现了应用 Netty 通信框架、

HTTP 传输方式[9]、数据库等的搭建。其中，服务器的 Netty 通信框架的设计与实现应用是核心问题，保证了和下位机之间稳定的数据传输。Netty 多用于搭建数据通信底层部分，并且是基于 TCP 通信协议实现，Netty 因其非阻塞通信方式，实现了优异的性能而被广泛应用，也是目前大多数框架的主流或被服务端首选用来搭建软件底层框架。

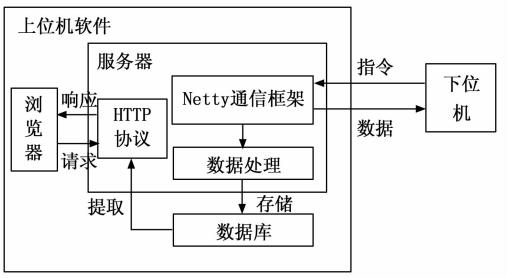

图 3　软件系统整体数据流图

2.2　相关技术

2.2.1　B/S 架构

目前软件常用的开发模式有两种，C/S 和 B/S[10]。C/S 模式，范围比较有限，大部分基于局域网环境使用，因为它的架构主要是由表示层和数据层构成的，所以使用方式必须是安装独有客户端使用，且难以一次开发实现多平台的使用。C/S 这种架构还有着一些其他缺点，举例来说，在开发者方面，客户端的升级和维护增加了工作量；对于用户来说，时常进行迭代也增加了使用成本。因此，需要简化客户端的安装部署工作。

B/S 模式下，选择了浏览器成为客户端，避免进行繁琐的客户端安装工作，使其可以专注于业务功能逻辑的开发实现。B/S 是一个由三个主要部分组成的 Web 应用程序：浏览器、服务器、数据库[11]。承担用户交互的主要应用软件，成为了浏览器部分，但是不承担业务部署和功能性开发，服务器端即可完成所有的业务逻辑。并且通过浏览器即可访问系统，浏览器的 UI 设计更加美观多样化，升级方便。

2.2.2　Netty 框架

Netty 是一个利用 Java 的高级网络的能力，隐藏其背后的复杂性而提供一个易于使用的 API 的客户端/服务器框架[12]。Netty 实现了多种通信协议的支持，因其使用简单，对非阻塞通信方式进行封装；功能强大，支持多种主流协议，内置集成了解码功能；稳定，目前已经修复了框架已发现的缺陷。所以，它已经成为大多框架和项目实现的底层，并被许多开发者青睐。Netty 框架的底层实现包括了一个非阻塞的 IO 架构，允许开发人员轻松地编写相关的应用程序，以达到优化网络编程的目的，虽然 Netty 的底层框架逻辑实现非常复杂，但是其 API 简单易用，开发者在编程中易于绕开业务层实现。其构成有三个模块：传输服务，支持的协议和核心模块[13]。

Netty 在设计上有以下几个优点：NIO 通信；无锁化的串行设计理念；高效的并发编程；高性能的序列化框架等[14]。本系统选择 Netty 主要是三方面的优势：并发高、传输可靠和封装好。

1）并发高：对比于 BIO（阻塞 IO），Netty 框架是基于非阻塞 IO 开发的网络通信框架。其单线程能处理的连接数比 BIO 要高出很多。当建立了一个连接之后，接着还有两个步骤，先是将客户端发过来的数据全部接收，然后是在服务器处理完请求给客户端响应。这也是 NIO 和 BIO 的一个主要区别。在 BIO 中，等候客户端发送数据这个过程是阻塞的，这样就造成了一个线程只能处理一个请求的情况，而且计算机的处理线程也不是无限的。但是在 NIO 中，当一个套接字建立好以后，Thread 类不会像 BIO 一样将其阻塞，而且将请求推到了选择器，由选择器去不停地遍历所有产生的套接字，当有一个套接字建立，选择器就会通知 Thread 类，然后交由 Thread 类处理后再响应给客户端，这样整个流程是非阻塞的。

2）传输快：Netty 的数据发送和接收主要是使用 bytebuffer，其使用对外内存直接对套接字进行读写操作。如果使用传统的堆内存对套接字进行读写，Java 虚拟机会将 buffer 直接复制一份到内存中后再写入套接字，多了一次缓冲区的内存复制操作。但是 Netty 提供了一种组合 ByteBuf，可以避免这样的情况，因为其并没有将 Buffer 组合起来，而是保存了它们的引用，本质上是避免了数据的复制，实现了零拷贝。Netty 中还使用了 FileChannel 的 transfer to 方法，此方法依赖于操作系统实现零拷贝机制。

3）封装好：编程过程中，从代码量来看，Netty 有很多自带处理方法，实现连接、请求和接入等[16]。如 ChannelHandler 及其实现类，ChannelHandler 接口定义了许多事件处理的方法，Netty 开发中需要自定义一个 Handler 类去实现 ChannelHandle 接口或其子接口或其实现类，然后我们可以通过重写这些方法去实现具体的业务逻辑。channelActive 通道就绪事件，channelRead 通道读取数据事件，exceptionCaught 通道发生异常的事件。

3　软件设计关键技术

在软件整体设计中提到，TCP 服务器的设计与实现是上位机系统中最关键部分。承担着与下位机通信，接收、发送、解析、保存数据等功能。其中主要就是可靠接收传输消息和及时发送消息指令，同时考虑到功能需求分析中下位机会频繁发送数据等问题，故选用了 Netty 通信框架搭建 TCP 服务器底层，通信架构如图 4 所示。在 Netty 通信框架的应用过程中主要研究了适应于本系统基于 Netty 的通信实现、传输中 TCP 粘包拆包处理和心跳监测机制的应用。

3.1　基于 Netty 的通信实现

Netty 被定义为是一个基于 Java 非阻塞 I/O（输入/输出）的异步事件驱动的网络应用框架[17-18]，提供了高层次的抽象来简化 TCP 服务器的编程，具有高性能和高可靠性的特点，且广泛应用于客户端与服务器之间长连接、高并发的场景。近几年，Netty 框架在计算机互联网行业热门起

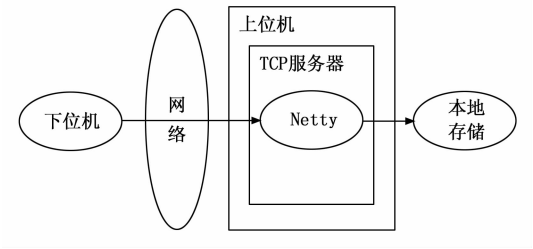

图 4 通信框架图

来，已经成为了 Java 网络编程框架的首选。

基本实现思路，通信的读写逻辑均是启动阶段实现连接数据的读写逻辑，其通过逻辑处理链 Pipeline 来添加逻辑处理器。首先，下位机连接成功回调处理器的 channelActive（）方法，当下位机和上位机分别接收到对方的消息回调自己的处理器的 channelRead（）方法；接着是上位机和下位机向对方写数据的时候调用 writeAndFlush（）方法，规定上位机和下位机之间交互信息的二进制数据传输载体为 ByteBuf。

Netty 服务器端的实现，是被动的接收发送请求，为了避免出现服务器之间多端口产生冲突的问题，将服务器的端口号固定且绑定为 6000。首先，创建 ServerBootstrap（）实例，设置并且绑定 Reactor（）线程池：EventLoopGroup，EventLoop 就是处理所有注册到该线程的选择器上的；设置并绑定服务器端的 Channel（）；TCP 链路建立时创建 ChannelPipeline（）方法，添加并设置 ChannelHandler（），用来处理网络事件；绑定监听端口并启动服务端；然后进入 Netty 独有流程，通过多路复用器轮询，进行网络事件通知，最后执行 Netty 系统和业务通过执行 pipline 中的方法最终调度调用 HandlerChannel（）方法，过程如图 5 所示。

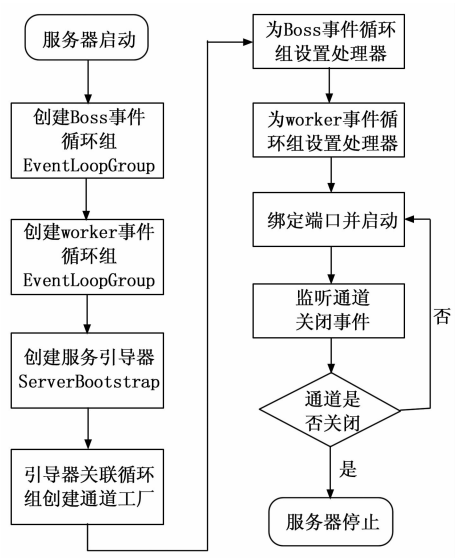

图 5 基于 Netty 服务器实现流程图

实现部分代码如下：

```
public void run() throws Exception{
  try{
   ServerBootstrap serverBootstrap = new ServerBootstrap();//创建服务器端的启动对象,配置参数
  serverBootstrap.group(groupBoss, groupWorker) //两个处理线程组设置
      .option(ChannelOption.SO_BACKLOG, 1024) // 线程队列并发数的设置
       .channel(NioServerSocketChannel.class)
       .childOption(ChannelOption.SO_KEEPALIVE, true) //保持活动的连接状态设置
       .childHandler((ChannelInit)(ch)->{});
       //服务器启动并绑定端口
  bind(serverBootstrap, PORT);
     // 对关闭通道进行监听
  cf.channel().closeFuture().sync();
        } catch (Exception e) {
    log.error(" netty 服务启动异常 " + e.getMessage());
        } finally {
    workerGroup.shutdownGracefully();
  bossGroup.shutdownGracefully();
    }
}
```

在 Netty 框架的设计与实现应用的过程中，主要实现了基础的数据通信交互后，因为 TCP 使用的是一种“数据流”的方式传输，还需要设计并且解决基于 TCP 数据传输过程中出现的粘包拆包的问题。

3.2 TCP 数据处理

TCP 是一种“数据流”的协议，处理数据包以流的方式，所以在接收数据的时候不能完整的得到一条数据称为 TCP 粘包拆包。Netty 这里提供了消息定长度，将传输消息大小固定不够的空位用空格补齐然后发送；符号分割即用特殊符合进行标识，接收方遇到标识符就知道是下一条数据；发送长度，每次发送数据的时候将这条数据的长度一并发送，这样对方解析的时候根据长度来判断数据接收是否完整。本软件采用了换行符的处理方式，结合 Netty 提供的多种解码器来解决这种问题。并且设计了自己的处理方法，顺序遍历 ByteBuffer 中的刻度字节[19]，查找标志位“/n”。

不断地从 TCP 缓冲区中读取数据，每次都需要判断这是不是一个完整的包读取完后，假如当下读到的消息不能够拼成完整的业务数据包，那就先把该数据保留，再从 TCP 缓冲区中读出待得到完整的消息包。数据内容正确需要进行业务处理，推向前台页面或者存入数据库，流程如图 6 所示。

实现部分代码如下：

```
class Solution(SocketChannel ch){
ch.pipeline().addLast(new IdleStateHandler(5,0,0));//换行符处理
ch.pipeline().addLast(new LineBaseFrameDecoder(1024));
```

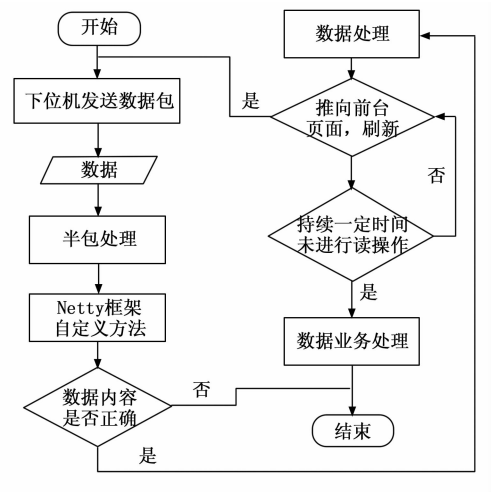

图 6　TCP 数据处理流程图

```
    ch.pipeline().addLast(new StringDecoder());//编码
    ch.pipeline().addLast(new StringEncoder());//解码
    ch.pipeline().addLast(new MyServerHandler());
}
```

实现了 TCP 传输中粘包拆包问题的解决方法后，还需要通过对 Netty 框架提供的心跳机制功能接口进行研究，来实现状态检测管理功能模块。

3.3　状态监测管理

在使用的过程中，会出现网络不稳定的情况，而在 TCP 通信中，发生网络问题就会导致上下位机连接中断。没有通信连接，那就无法发现相互掉线，这时候需要通过心跳机制实现状态监测。类似一端发送一个数据包内容给对面，另一端收到后马上返回一个响应包，这样的一个数据交互就是心跳交互[20]。通俗来说，就是定时发送一个自定义的心跳包，让对方知道己方的存在，来确保连接的有效性。

下位机每间隔 1 s 向上位机发送一个包含特殊内容数据的消息 Ping，上位机收到后马上返回一个消息 Pong，于是双方通过这样一对心跳消息来确定 TCP 的连接是否断开，如图 7 所示。

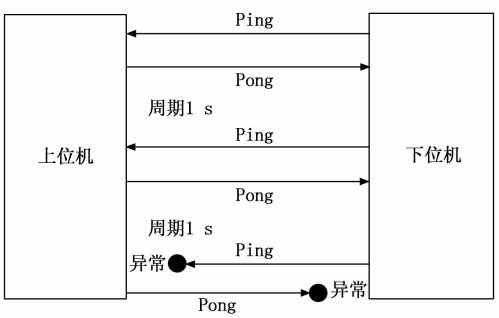

图 7　心跳机制

在本文中，依据功能需求，基于心跳机制的原理，借助 Netty 提供的 IdleStateHandler 类，来实现处理下位机的连接状态。

```
    public userEventTriggered
    (ChannelHandlerContext ctx, Object evt){
    if(evt instanceof IdleStateEvent){
        IdleStateEvent e =( IdleStateEvent) evt;
        switch(e.state()){
          case READER_IDLE:
            handlerReaderIdle(ctx);
            break;
         case WRITER_IDLE:
            handleWriterIdle(ctx);
            break;
         case ALL_IDLE:
            handlerAllIdle(ctx);
            break;
      }
    }
}
```

其中需要通过创建通用类负责接收心跳，客户端发送心跳，在上文中提到过，使用 Netty 提供的 IdleStateHandler 类实现心跳的关键，根据不同的 idle 类型发生不同的 idle 类事件，而该事件的捕捉就是通过上文的 userEventTriggered 实现的。

心跳机制的实现，下位机就需要按照设定周期频繁的发送数据包，上位机就必须能够响应和处理下位机发送来的数据请求。这里，Netty 框架提供的 NettyServerHandler 类又实现了多个重要的方法，设计使用非公平锁机制提高系统的吞吐量，更好地处理与客户端的连接、断开和数据读取。非公平锁是指多个线程获取锁的顺序，并不是按照申请锁的顺序，有可能申请的线程比先申请的线程优先获得锁，收发效率较高。

4　通信测试

对本系统软件数据通信进行测试和试用，上位机软件测试环境为 Intel（R）Core（TM）i3-8100，内存 8 GB，CPU3.6 GHz，硬盘 1 T，操作系统 Windows 7 64 位；下位机操作系统为 Windows7，上下位机之间为千兆网络连接，测试环境框架如图 8 所示。

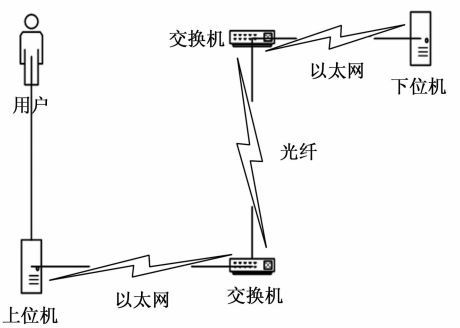

图 8　通信测试环境框架图

4.1 模拟测试

本系统上位机软件关键部分在于 TCP 服务器的设计，采用 Netty 通信框架设计，实现稳定可靠的上下位机数据传输，满足上位机接收处理下位机频繁的数据发送需求。TCP 服务器部分属于底层开发，所以采用 WebSocket 技术通过 Web 页面进行模拟测试的数据展示。WebSocket 是一种新的协议在 HTML5 下，其还是基于 TCP 的协议本质上说。实现了全双工通信在浏览器端与服务器之间，模拟 Socket 协议，可以双向发送或接受信息，实现数据推送，主动由服务端推送向浏览器端；完成一次握手，由浏览器和服务器，即可以建立长时间的连接在两者之间，实现数据传输，更加方便我们进行通信模拟测试。如图 9 所示，对下位机部分进行了模拟设计实现，充分达到真实的使用情况，实现心跳机制发送状态数据，以及模拟坐标数据发送。

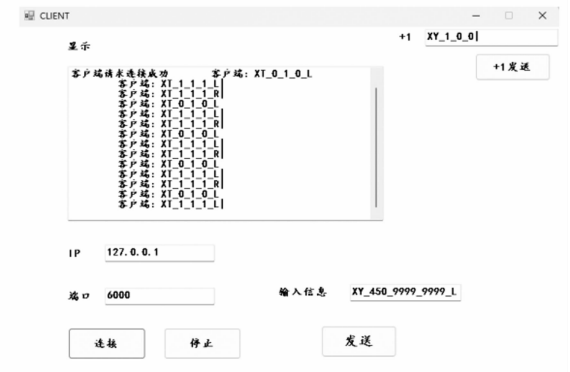

图 9 下位机模拟程序

模拟下位机实现状态数据发送，以周期 1 s 的时间模拟下位机向上位机发送状态的协议内容和手动发送模拟的测量试验结果坐标数据。上位机通过浏览器页面展示由下位机发送的心跳状态数据和坐标试验数据，TCP 服务器模拟接收数据测试结果如图 10 所示。

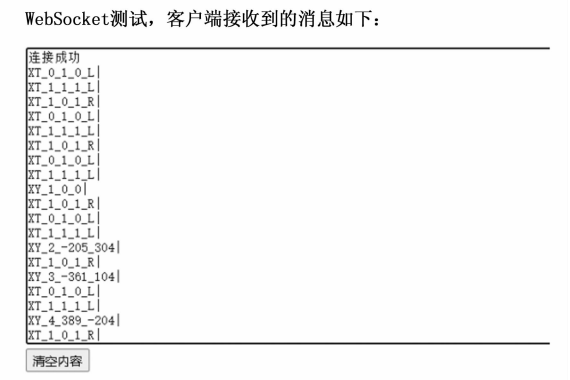

图 10 上位机数据接收测试图

通过模拟测试，下位机所发送的状态检测数据和试验坐标数据上位机的 TCP 服务器都能正常接收，且全部显示到测试页面，满足基本的功能需求。

4.2 性能测试

最后对上位机接收数据处理性能进行对比测试。根据功能需求分析所提出的技术指标要求，需要上位机对接收数据的处理不能低于每秒钟一个请求。创建两组不同周期数据发送的对比测试，第一组设计 1 000 个样本数据，以周期 1 s 发送数据，测试服务器接收处理请求的性能；第二组同样设计 1 000 个样本数据，发送速率比第一组提高 10 倍以周期 100 ms 发送数据，测试服务器接收处理请求的性能。这里采用 JMeter 开源测试软件，模拟一个下位机以周期 1 s 的时间向上位机发送数据判断上位机 TCP 服务器的接收情况，如图 11 所示。由测试吞吐量结果可知，每秒能够处理请求的次数约为 1，能够满足功能需求。

图 11 第一组性能测试结果

第二组性能测试将数据发送速率设为 100 ms，对服务器处理性能测试，测试结果如图 12 所示。

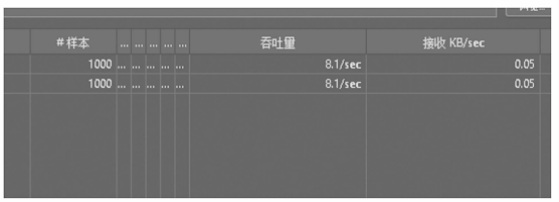

图 12 第二组性能测试结果

从上述的两组对比性能测试中，在 1 000 个样本数据的情况下，将数据发送速率从 1 s 提高 10 倍的测试结果看，第一组的吞吐量测试结果为服务器每秒处理请求约 1 个，第二组的吞吐量测试结果为服务器每秒处理请求约 8 个，满足性能要求以及技术指标。

4.3 实际测试

进行了模拟测试和性能测试后，再对上位机软件进行系统测试，测试的内容为将上位机软件置于真实的使用环境，测试上位机软件是否能达到效果。

把上位机软件置于试验测试的界面，如图 13 所示。从弹丸发射开始计算时间起始，上位机界面显示模拟坐标计算为时间结束，来衡量数据通信和软件系统响应的情况。同时作为对性能测试的一个补充测试。

通过对整个精度靶的系统联调测试，可以得到在实际使用环境中的结果。从图中的测试结果可以看到，从弹丸发射开始计时到最后上位机显示坐标位置整个过程时间发生都在 1 s 以内，忽略掉计时操作的时间以及系统的响应时间，数据的通信部分能够满足相应的技术指标要求，整体上

（下转第 97 页）