改进的 TCP 应用层协议在远程实验系统中的应用

冯建文 董 剑*
(杭州电子科技大学 浙江 杭州 310018)

摘 要 TCP 协议以字节流的方式进行传输,无法对消息的边界进行保护。在大批量数据传输或者网络延迟较大时,可能产生分包和黏包现象。提出一种改进的 TCP 应用层协议及其处理机制。运用固定长度协议头和数据的封包技术,设计并实现自定义的应用层协议,并应用于远程实验系统中。实验表明,改进的 TCP 应用层协议能够实现大批量数据的准确传输,具有针对性强、灵活性高、兼容性好等优点。

关键词 自定义应用层协议 远程实验系统 TCP 兼容性 灵活性

中图分类号 TP3 **文献标识码** A DOI:10.3969/j.issn.1000-386x.2018.12.030

APPLICATION OF IMPROVED TCP APPLICATION LAYER PROTOCOL IN REMOTE EXPERIMENT SYSTEM

Feng Jianwen Dong Jian*
(*Hangzhou Dianzi University*, *Hangzhou 310018*, *Zhejiang*, *China*)

Abstract The TCP protocol is transmitted in byte stream mode. It cannot protect message boundaries. When mass data transmission or network delay is large, subcontracting and sticky packets may occur. This paper presented an improved TCP application layer protocol and its processing mechanism. Using fixed length protocol header and data packet technology, we designed and implemented a custom application layer protocol, and applied it to remote experiment system. Experiments show that the improved TCP application layer protocol can achieve accurate transmission of mass data, with strong pertinence, high flexibility and good compatibility.

Keywords Custom application layer protocol Remote experiment system TCP Compatibility Flexibility

0 引 言

近年来,伴随着计算机的普及和网络技术的不断发展,互联网提供的服务在人们的生活中越来越不可或缺。TCP/IP 协议是当前互联网中最主要的通信协议标准,是国际互联网络的基础,TCP 协议是一种面向连接的、可靠的、以字节流方式进行传输的协议[1]。由于面向字节流的协议是无边界的,在传输过程中,不保留数据的边界信息,这样就可能出现以下问题:当发送方连续进行发送操作时,接收方在一次接收操作中,可能会同时接收到发送方多次发送的数据;在接收端也可能一次无法完成所有数据的接收操作[2]。在客户端和服务端通信时,如果数据之间没有边界,那么服务器端无法确定需要经过几次接收操作才能完成一次数据交换。所以,需要设计应用层通信协议,对面向字节流的数据进行边界识别,来保证数据正确发送和接收。而往往在实现自己需要的特定功能时,对数据的安全性、灵活性等方面会有较高的要求,http、ftp、smtp 等已知协议可能难以满足需求,因此需要设计并实现自定义应用层协议。本文提出的自定义应用层协议的方法可适用于大部分应用程序的设计,实验结果证明此方法可以保证数据的准确性和实时性,并且代码灵活性高,针对性强。

1 TCP 协议

网络协议是为进行数据传输而制定的标准。发送

收稿日期:2018 - 05 - 25。国家自然科学基金项目(61877015);校级实践类课程教学改革项目(sjkg201801)。**冯建文**,副教授,主研领域:嵌入式系统,系统集成。**董剑**,硕士生。

方将特定信息封装到请求中发送给对方;接收方接收到来自发送方的信息后,按照相应协议解析,从而获取对方发送过来的原始信息。

通信协议包括三个要素:

(1) 语法:规定了信息的结构和格式;

(2) 语义:表明信息要表达的内容;

(3) 同步:规则通信内容和通信时间。

TCP 协议在不同领域的应用程序研发中被应用,当前互联网上进行 2 台计算机之间数据传输的主要方式就是应用了 TCP 协议[3]。在 TCP 协议中,通信双方分为客户端和服务器端,由于 TCP 是面向连接的,所以作为服务器端需要等待客户端的连接申请,连接成功后客户端和服务器端就可以互相通信,传输数据。客户端和服务器端通过套接字(socket)这种通信机制可以在网络中通信。

图 1 中展示了 TCP 客户端与服务器端进行通信时套接字函数的调用流程。

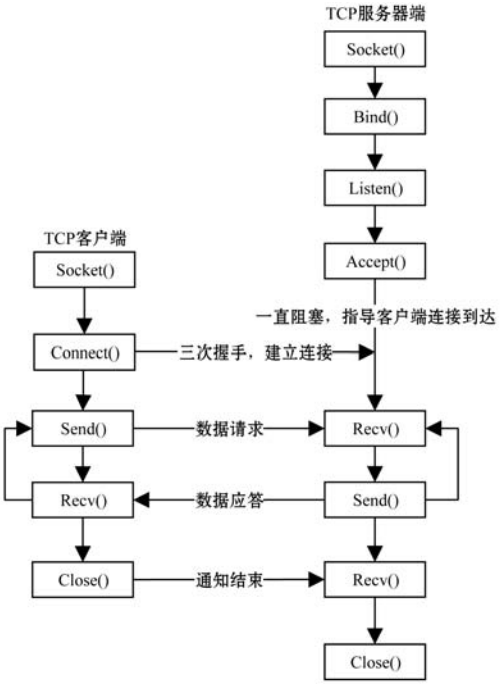

图 1　TCP 客户端/服务器端的套接字函数调用流程

服务器首先启动,然后监听客户的连接。当收到客户的请求时进行判断,如果客户连接成功,则双方可以进行数据的发送与接收,直到客户关闭客户端的连接,服务器也关闭相应的服务器端的连接,然后等待新的客户连接。

2　自定义 TCP 应用层协议

2.1　必要性

TCP 协议是以流的形式传输,在 TCP 流传输的过程中,由于面向字节流的协议是没有边界的,可能会出现分包与黏包的现象。因此,需要自定义应用层协议对数据进行处理。

分包是指接收方只接收了部分数据包。IP 分片、传输过程中丢失部分数据、接收缓冲区太小等都可能产生分包。

黏包是指发送方连续发送若干包数据,接收方接收后,后一包数据的头紧接着前一包数据的尾,无法分辨出每个数据包的界限。由于 TCP 协议面向连接的机制,客户端与服务器端会维持一个连接,数据在连接不断开的情况下,会不停地向服务器端发送数据包,可能产生黏包;当发送的网络数据包太小时,TCP 协议本身会启用 Nagle 算法将多个较小的数据包合并再发送。收到数据时服务器端可能由于无法确定数据包是否是客户端自己分开发送的而产生黏包。

2.2　TCP 的同步机制

由于远程实验系统自定义应用层协议是基于 TCP 的,应用层无法得知数据是否完全接收完毕,为了使接收方能正确理解发送方需要发送的数据,一般有三种方法:

(1) 双方约定一个固定的长度。发送方每次发送这一固定长度的数据,接收方每次都接收这么长,就不会造成偏差。这样完成的系统缺乏可扩展性和灵活性,而且会增加网络的负担,无论每次发送的有效数据是多大,都要按照定长的数据长度进行发送。

(2) 在数据的最后设置分隔符。接收方接收到分隔符就说明一次发送完成。这样对数据内容有要求,如果数据内容中含有分隔符,会造成一系列的错误。

(3) 在每个发送操作前加上数据包的长度。使用这种方法在接收方接收数据时,收到这一长度的数据量就算是一次接收完成。但是这种方法发送一次数据需要双方进行两次交互,分别发送长度和数据,加大了 CPU 的负荷,而且缺乏安全性。虽然 TCP 协议中有校验和,但是不同层次的校验覆盖范围不一致,因此自定义应用层协议中需要增加校验和这一字段,进一步提高数据的完整性。

3　改进的自定义应用层协议

3.1　协议优劣的标准

好的应用层协议一般具有以下特点:

（1）高效。快速打包解包减少对 CPU 的占用。

（2）简单、易于人的理解。

（3）易于扩展的。对可预知的变更，有足够的弹性用于扩展。

（4）容易兼容的。协议更新后，仍然可以使用新协议对旧协议发出的报文进行解析。

3.2 TCP 应用层传输协议的结构

封包技术就是在发送时对数据包进行处理，将包处理成协议头和包体。协议头是大小固定的结构体，其中有成员变量表示包体长度、包类型等，通过协议头中的内容可以判定接收方收到的数据包是否完整。

发送时通过封包技术将协议头和数据内容组成一个数据包，其中协议头中有包类型、包长度、校验和等。接收方先读取协议头，根据协议头中的数据长度循环接收数据，直到接收到的数据大小等于协议头中的数据长度字段，此时接收完全。然后可以根据协议头中的包类型等字段，使用相应的协议进行解包。由于 TCP 协议三次握手机制，可以保证数据从发送缓冲区到接收缓冲区是有序无误的，而应用程序从缓冲区读入的时候，无法完全保证数据安全性，所以应用上层还是要做 TCP Sokcet 的数据校验。设计的通信协议如图 2 所示。

图 2 通信协议设计

（1）协议头版本：便于后期更新、维护。

（2）数据包类型：可以指定数据包的作用，便于解析数据部分的内容。

（3）数据包长度：指的是数据包的总长度。

（4）CS 校验：TCP 校验无法覆盖到应用进程与 TCP 协议栈间的信息交互错误。远程实验系统对数据的可靠性要求较高，因此自定义应用层协议中必须包含数据的完整性校验。

（5）预留：预留一块空间，便于后期增加内容，提高协议的可扩展性和兼容性。

3.3 处理机制

该自定义应用层协议工作时的处理机制如图 3 所示。

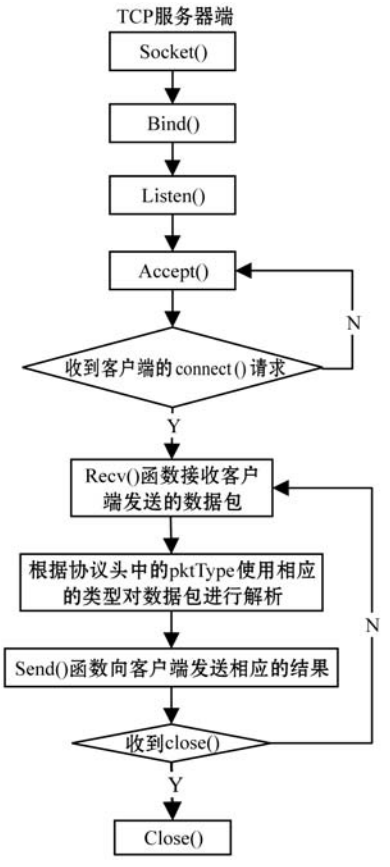

图 3 自定义应用层协议服务器端数据传输流程图

首先，服务器启动，然后监听客户的连接。当收到客户端发来的 connect( ) 请求后建立连接，接着 Recv( ) 函数接收客户端发送的数据包，先对固定协议头大小的数据使用协议头进行解析，然后根据协议头中的 pktType、totalLen 等字段使用相应的协议进行解析，发送对应的结果，接着继续接收下一个数据包直到收到客户端的 Close( ) 请求关闭连接。

4 远程实验系统中的应用

4.1 远程实验系统概述

远程实验系统由客户端、服务器端和 ARM 客户端三个模块组成，其整体结构如图 4 所示。

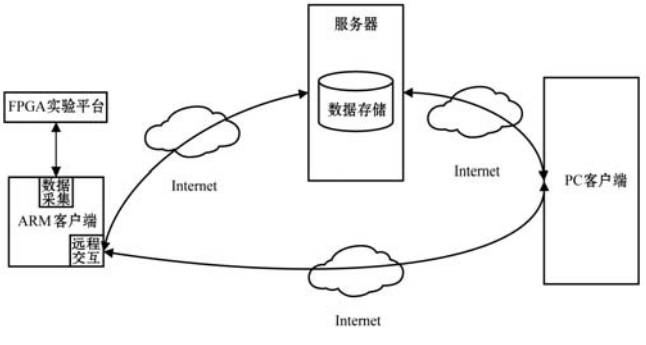

图 4 远程实验系统结构图

(1) PC 客户端　给用户提供实验接口,引导用户进行实验,并将实验数据形象地展现给客户。

(2) 服务器端　负责对用户数据、实验数据进行管理,对数据进行解析或者封装,是 PC 客户端和 ARM 客户端交互的桥梁。

(3) ARM 客户端　ARM 客户端对 FPGA 实验平台进行动态配置,采集实验数据并将数据最终传输到客户端显示。

4.2　TCP 应用层协议的定义

根据远程实验系统的结构,可以将协议头部分定义为一个结构体,数据部分定义为一个结构体并且包含协议头部分。不同包类型的结构如表 1 所示。

表 1　包类型结构图

包类型	pktType	数据部分长度
登录包	01	32 B
配置文件包	02	2 092 B
实验数据包	03	38 B

协议头设计:

```
typedef struct PacketHeader
{
    unsigned short version;              //协议头版本号
    unsigned short pktType;              //数据包类型
    unsigned int totalLen;               //数据包长度
    unsigned int checkSum;               //CS 校验
    char  reverse[24];                   //预留
}PacketHeader;
```

以用户登录数据包为例,其数据包结构如下:

```
typedef struct ClientLoginPacket
{
    PacketHeader header;
    char userName[16];                   //用户名
    char pwd[16];                        //用户密码
}ClientLoginPacket;
```

以配置文件包为例,其数据包结构如下:

```
typedef struct FileDataPacket
{
    PacketHeader header;
    char filePath[32];                   //文件路径
    int fileLen;                         //文件总长度
    int len;          //本次发送的数据包中,数据的长度
    char data[2048];              //本次发送的文件内容
    int id;                              //客户端 id
} FileDataPacket;
```

4.3　登录的具体实现

登录数据传输流程图如图 5 所示。

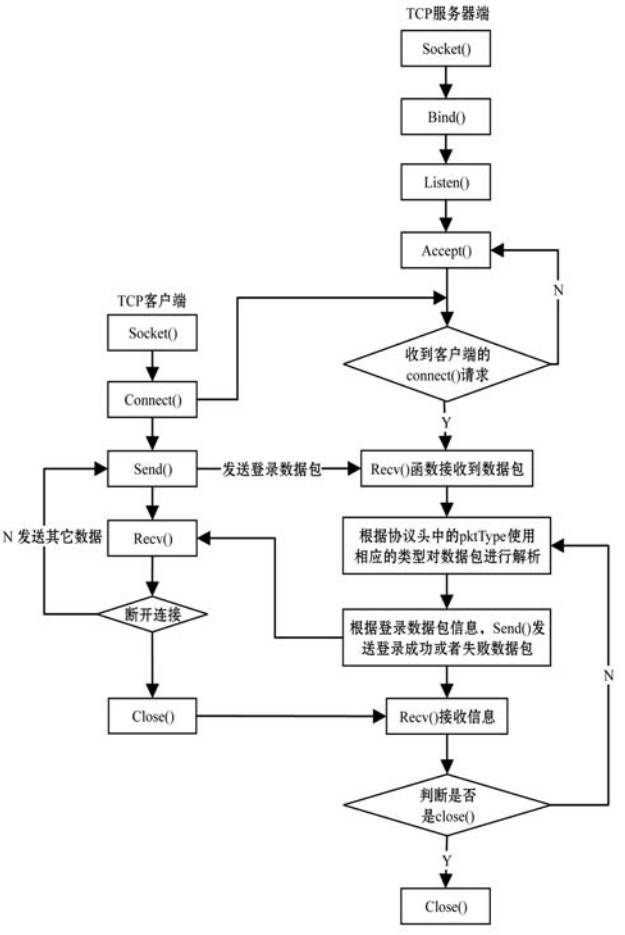

图 5　登录数据传输流程图

首先启动服务器端,调用 bind( ) 和 listen( ) 这两个函数,然后等待连接。当客户端调用 connect( ) 函数连接成功后发送数据,当服务器端接收到来自客户端的数据时,对数据进行处理,代码如下:

```
while(pIoContext->m_nRecvLen >= PKT_HEADER_LEN)
{
    PacketHeader *header
        =(PacketHeader*)pIoContext->m_szRecvPkt;
    if(pIoContext->m_nRecvLen >= header->totalLen)
    {
      pIOCPModel->_DoRecv(pHandleContext, pIoContext);
      memcpy(pIoContext->m_szRecvPkt,
            pIoContext->m_szRecvPkt+header->totalLen,
            pIoContext->m_nRecvLen-header->totalLen);
      pIoContext->m_nRecvLen -= header->totalLen;
    }
    else
      break;
}
```

其中 m_szRecvPkt 是一个缓冲区,保存已收到的数据内容,m_nRecvLen 是已收到的数据长度。代码表示收到消息后,检测收到的数据长度是否大于一个协

议头的长度,如果小于一个协议头的长度,那么表示数据包没有接收完成,继续接收,否则使用数据协议头对数据进行解析。再检测数据协议头中数据长度字段的大小,如果收到的数据长度大于协议头中数据长度字段 totalLen 的长度,说明登录数据包接收完成,否则,还没有接收完,需要继续接收。

完全接收到数据后对数据进行处理的代码如下:

```
PacketHeader * header =
(PacketHeader * ) pIoContext -> m_szRecvPkt;
switch (header -> pktType == CLIENT_LOGIN_PACKET)
{
    ClientLoginPacket * clientLoginPacket =
    (ClientLoginPacket * ) pIoContext -> m_szRecvPkt;
}
```

根据数据协议头中的数据包类型字段 pktType 确定数据包是登录数据包,然后使用登录数据包对收到的数据进行解析,然后对其数据内容进行判断,符合条件则登录成功,向客户端发送登录成功消息,否则登录失败。

5 性能测试

通过多线程的方式,启动多个线程并发发送不同的文件,查看服务器端接收文件的情况,如表 2 所示,所有测试包的正确性为 100% 。

表 2 测试结果表

数据类型	客户端数量	平均包大小/KB	平均用时/ms
登录	1	0.06	0
登录	50	0.06	0
登录	100	0.06	0
配置文件	1	102 196	14 523
配置文件	50	29 779	4 271
配置文件	100	30 057	5 662
实验数据	1	0.08	0
实验数据	50	0.08	0
实验数据	100	0.08	0

表 2 中的登录数据包和实验数据包平均包过小,平均用时接近 0 ms。由表 2 可知,对于大批量文件的传输,本文方法解决了由数据量过大或者网络延迟过高造成的分包和黏包问题,保证了数据传输的准确性。

6 结 语

通过在远程实验系统中使用改进的应用层协议,数据传输提高了准确性、实时性。从实验结果可以看到,使用这种改进的应用层协议使得打包解包更加快捷、准确,减少了 CPU 的占用;从程序代码来看,结构清晰、易于理解,便于数据解析;由于数据协议头中有版本号字段和预留字段,使得协议具有更好的扩展性和兼容性。

本文提出的改进的应用层协议的设计方法具有普遍性,对于不同情况的应用程序,经过修改均适用。

参 考 文 献

[1] 邱正师. 基于 KRTS 的实时视觉处理系统与模板匹配算法研究[D]. 山东:山东大学 2017.

[2] 徐镇河. 征服 C/C ++ 企业软件开发核心技术[M]. 北京:科学出版社,2008.

[3] 龙昱程. 基于 TCP 协议的应用层协议设计[J]. 信息通信,2015(5):69 - 70.

[4] 谭靖, 杨为民, 张百平,等. 基于自定义协议的网络地理信息系统[J]. 计算机工程, 2008, 34(13):248 - 250.

[5] 陈健苇. 基于 Winsock 的局域网聊天室设计与实现[J]. 科学技术与应用,2013(2):134 - 136.

[6] 王大治, 王世卿, 李曼. 基于自定义协议的中间件开发方法研究[J]. 计算机应用研究, 2002, 19(10):59 - 61.

[7] 刘佳,郑华,刘洋,等. 基于 TCP 和 UDP 混合协议的远程控制软件的设计与实现[J]. 计算机应用与软件, 2010, 27(3):127 - 130.

[8] Kim M J, Cloud J, ParandehGheibi A, et al. Network coded TCP (CTCP)[EB]. eprint arXiv:1212.2291,2012.

[9] 崔莉,鞠海玲,苗勇. 无线传感器网络研究进展[J]. 计算机研究与发展,2005,42(1):163 - 174.

[10] Ma Y, Wang L, Cui J, et al. Design of P2P application layer protocol[C]//International Conference on Mobile Ad-Hoc and Sensor Networks. IEEE, 2017:175 - 178.

[11] 罗军舟, 黎波涛, 杨明. TCP/IP 协议及网络编程技术[M]. 北京:清华大学出版社,2004.

[12] Sundararajan J K, Jakubczak S, Medard M, et al. Interfacing network coding with TCP: an implementation[C]//INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15 - 19 March 2010, San Diego, CA, USA. IEEE 2010.

[13] 牟璇,王俊峰. TCP 自适应压缩传输方案研究[J]. 计算机应用与软件,2013,30(11): 279 - 282.

[14] 陈佳. 应用层协议快速识别的研究与实现[D]. 北京:北京邮电大学, 2010.