



jeremyxu 发表于 jeremy的技术...

👁 4.1K

分享



通过WebSocket传输文件

工作中需要将大量文件从一台服务器传输至另一台服务器，最开始是直接使用基础的TCP编程搞定的。但后来业务上要求两台服务器间只能走HTTP协议，而且还要保证传输过去的文件的完整性。想了下，最后基于WebSocket协议完成了该功能。

思路

1. 服务器端侦听某端口，接受WebSocket请求，后面可用nginx作反向代理，外部看到的将是80端口
2. 客户端连接服务器的WebSocket地址，连接成功后，首先传送一个NEW_FILE的数据包，里面带上要传输的文件名
3. 服务器端收到NEW_FILE包后，解析出文件名，并创建目标文件，再回复ACK_NEW_FILE的数据包
4. 客户端收到ACK_NEW_FILE的数据包后，检查回应的code，如是成功码，则启动一个线程，该线程负责将源文件的数据封装成多个FILE_DATA数据包，传送这些FILE_DATA数据至服务器端
5. 服务器端接收FILE_DATA数据包，解析出里面的文件数据，将文件数据写入文件
6. 客户端发送完源文件数据后，再传送一个FILE_END数据包，该数据包中带上源文件的MD5值
7. 服务器端收到FILE_END数据包后，比对源文件的MD5值与目标文件的MD5值，如相同，则认为传输成功，并返回ACK_FILE_END数据包，里面带上成功码
8. 客户端收到ACK_FILE_END数据包，检查回应的code，如是成功码，则认为传输成功，否则认为传输失败。

具体实现

以下为示例的简易代码，项目中的代码比这组织得更完善一些。该实现使用了WebSocket的Java实现 `Java-WebSocket` 与Java NIO。

`FilePacket.java`

```
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.charset.StandardCharsets;

/**
 * Created by jeremy on 16/6/11.
 */
public class FilePacket {
    public static final int P_NEW_FILE = 0x01;
    public static final int P_ACK_NEW_FILE = 0x02;
    public static final int P_FILE_DATA = 0x03;
    public static final int P_FILE_END = 0x04;
    public static final int P_ACK_FILE_END = 0x05;

    public static final int SUCCESS_CODE = 0;
    public static final int ERROR_CODE = -1;

    private static final int TYPE_LEN = 1;

    private int type;

    private final ByteBuffer buffer;
```

在这篇文章中：

思路

具体实现

注意事项

创作者年度总结
扫码立即查看



分享



```

public FilePacket(ByteBuffer buffer) {
    this.buffer = buffer;
}

public static FilePacket constructNewFilePacket(String fileName) {
    byte[] bytes = fileName.getBytes(StandardCharsets.UTF_8);
    ByteBuffer buffer = ByteBuffer.allocate(TYPE_LEN + 4 + bytes.length);
    buffer.order(ByteOrder.BIG_ENDIAN);
    buffer.put((byte)P_NEW_FILE);
    buffer.putInt(bytes.length);
    buffer.put(bytes);
    buffer.flip();
    return new FilePacket(buffer);
}

public static FilePacket constructAckNewFilePacket(int code) {
    ByteBuffer buffer = ByteBuffer.allocate(TYPE_LEN + 1);
    buffer.order(ByteOrder.BIG_ENDIAN);
    buffer.put((byte)P_ACK_NEW_FILE);
    buffer.put((byte)code);
    buffer.flip();
    return new FilePacket(buffer);
}

public static FilePacket constructFileEndPacket(String digest) {
    byte[] bytes = digest.getBytes(StandardCharsets.UTF_8);
    ByteBuffer buffer = ByteBuffer.allocate(TYPE_LEN + 4 + bytes.length);
    buffer.order(ByteOrder.BIG_ENDIAN);
    buffer.put((byte)P_FILE_END);
    buffer.putInt(bytes.length);
    buffer.put(bytes);
    buffer.flip();
    return new FilePacket(buffer);
}

public static FilePacket constructAckFileEndPacket(int code) {
    ByteBuffer buffer = ByteBuffer.allocate(TYPE_LEN + 1);
    buffer.order(ByteOrder.BIG_ENDIAN);
    buffer.put((byte)P_ACK_FILE_END);
    buffer.put((byte)code);
    buffer.flip();
    return new FilePacket(buffer);
}

public static FilePacket parseByteBuffer(ByteBuffer buffer){
    FilePacket p = new FilePacket(buffer);
    p.parseType();
    return p;
}

private void parseType() {
    this.type = (int)this.buffer.get();
}

public ByteBuffer getBuffer() {
    return buffer;
}

public int getType() {
    return type;
}
}

```

FileServer.java

```

import org.java_websocket.WebSocket;
import org.java_websocket.handshake.ClientHandshake;
import org.java_websocket.server.WebSocketServer;

import javax.xml.bind.DatatypeConverter;
import java.io.IOException;
import java.net.InetSocketAddress;

```





分享



```
import java.net.UnknownHostException;
import java.nio.ByteBuffer;
import java.nio.channels.ByteChannel;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.security.MessageDigest;
import java.util.EnumSet;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;

/**
 * Created by jeremy on 16/6/11.
 */
public class FileServer extends WebSocketServer {

    private ConcurrentMap<WebSocket, Map<String, Object>> clients = new ConcurrentMap<WebSocket, Map<String, Object>>();

    public FileServer(int port) throws UnknownHostException {
        super(new InetSocketAddress( port ));
    }

    @Override
    public void onOpen(WebSocket websocket, ClientHandshake clientHandshake) {
        clients.put(websocket, new HashMap<String, Object>());
    }

    @Override
    public void onClose(WebSocket websocket, int i, String s, boolean b) {
        clients.remove(websocket);
    }

    @Override
    public void onMessage(WebSocket websocket, String s) {
        // do nothing
    }

    @Override
    public void onMessage(WebSocket conn, ByteBuffer message) {
        FilePacket p = FilePacket.parseByteBuffer(message);
        Map<String, Object> params;
        ByteChannel fileChannel;
        MessageDigest md;
        switch (p.getType()) {
            case FilePacket.P_NEW_FILE:
                try{
                    int fileNameLen = p.getBuffer().getInt();
                    byte[] fileNameBytes = new byte[fileNameLen];
                    p.getBuffer().get(fileNameBytes);
                    String fileName = new String(fileNameBytes, StandardCharsets.UTF_8);
                    System.out.println("receive file request : " + fileName);
                    Path filePath = Paths.get("/tmp/otherdir", fileName);
                    fileChannel = Files.newByteChannel(filePath, EnumSet.of(StandardOpenOptions.CREATE, StandardOpenOptions.WRITE));
                    params = clients.get(conn);
                    params.put("fileChannel", fileChannel);
                    md = MessageDigest.getInstance("MD5");
                    params.put("md", md);
                    System.out.println("server accept file request: " + fileName);
                    FilePacket ackP = FilePacket.constructAckNewFilePacket(FilePacket.P_NEW_FILE, fileName, md);
                    conn.send(ackP.getBuffer());
                } catch (Exception e){
                    System.out.println("server deny file request");
                    FilePacket ackP = FilePacket.constructAckNewFilePacket(FilePacket.P_NEW_FILE, fileName, md);
                    conn.send(ackP.getBuffer());
                }
                break;
            case FilePacket.P_FILE_DATA:
                params = clients.get(conn);
                fileChannel = (ByteChannel) params.get("fileChannel");
                md = (MessageDigest)params.get("md");
                try {
                    p.getBuffer().mark();
                    md.update(p.getBuffer());
                    p.getBuffer().reset();
                    fileChannel.write(p.getBuffer());
                } catch (IOException e){

```





分享



```

        try {
            fileChannel.close();
        } catch (IOException ignore) {
        }
        conn.close();
    }
    break;
case FilePacket.P_FILE_END:
    params = clients.get(conn);
    fileChannel = (ByteChannel) params.get("fileChannel");
    md = (MessageDigest) params.get("md");
    try {
        byte[] digest = md.digest();
        String localDigest = DatatypeConverter.printHexBinary(digest);
        int digestBytesLen = p.getBuffer().getInt();
        byte[] digestBytes = new byte[digestBytesLen];
        p.getBuffer().get(digestBytes);
        String remoteDigest = new String(digestBytes, StandardCharsets.UTF_8);
        System.out.println("receive file end, digest : " + remoteDigest);
        FilePacket ackP;
        if (localDigest.equals(remoteDigest)) {
            System.out.println("file digests are same, send success ack");
            ackP = FilePacket.constructAckFileEndPacket(FilePacket.SUCCESS);
        } else {
            System.out.println("file digests are not same, send error ack");
            ackP = FilePacket.constructAckFileEndPacket(FilePacket.ERROR);
        }
        conn.send(ackP.getBuffer());
    } finally {
        try {
            fileChannel.close();
        } catch (IOException ignore) {
        }
    }
    break;
}

@Override
public void onError(WebSocket websocket, Exception e) {

}

public static void main(String[] args) throws UnknownHostException, InterruptedException {
    FileServer s = new FileServer(8888);
    s.start();
    System.out.println("FileServer started on port: " + s.getPort());

    Thread.sleep(Long.MAX_VALUE);
}
}

```

FileClient.java

```

import org.java_websocket.client.WebSocketClient;
import org.java_websocket.drafts.Draft_17;
import org.java_websocket.handshake.ServerHandshake;

import javax.xml.bind.DatatypeConverter;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.channels.ByteChannel;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.security.MessageDigest;
import java.util.EnumSet;
import java.util.concurrent.atomic.AtomicBoolean;

/**
 * Created by jeremy on 16/6/11.

```





分享



```

*/
public class FileClient implements Runnable{

    private final String wsUrl;
    private final Path filePath;
    private WebSocketClient wsclient;

    private volatile AtomicBoolean running = new AtomicBoolean(false);

    public FileClient(String wsUrl, Path filePath) {
        this.wsUrl = wsUrl;
        this.filePath = filePath;
    }

    public static void main(String[] args) throws InterruptedException {
        FileClient fileClient = new FileClient("ws://127.0.0.1:8888", Paths.get(""));
        fileClient.start();

        fileClient.await();
    }

    private void await() {
        while(running.get()){
            synchronized (running) {
                try {
                    running.wait(2000L);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    private void start() {
        Thread t = new Thread(this);
        t.start();
        running.set(true);
    }

    public void run() {
        try {
            wsclient = new WebSocketClient(new URI(this.wsUrl), new Draft_17()) {
                @Override
                public void onOpen(ServerHandshake serverHandshake) {
                    String fileName = FileClient.this.filePath.getFileName().toString();
                    System.out.println("request send file : " + fileName);
                    FilePacket p = FilePacket.constructNewFilePacket(fileName);
                    this.send(p.getBuffer().array());
                }

                @Override
                public void onMessage(String s) {
                    // do nothing
                }

                @Override
                public void onMessage(ByteBuffer bytes) {
                    FilePacket p = FilePacket.parseByteBuffer(bytes);
                    int code;
                    switch (p.getType()) {
                        case FilePacket.P_ACK_NEW_FILE:
                            code = (int)p.getBuffer().get();
                            if(FilePacket.SUCCESS_CODE == code){
                                System.out.println("server accept file request");
                                startSendFileData();
                            }
                            break;
                        case FilePacket.P_ACK_FILE_END:
                            code = (int)p.getBuffer().get();
                            if(FilePacket.SUCCESS_CODE == code){
                                System.out.println("server save file sucessfully");
                                wsclient.close();
                            }
                            break;
                    }
                }
            }
        }
    }
}

```



创作者年度总结
扫码立即查看



分享



```

        @Override
        public void onClose(int i, String s, boolean b) {
            stop();
        }

        @Override
        public void onError(Exception e) {
            stop();
        }
    };

    wsclient.connect();
} catch (URISyntaxException e) {
    e.printStackTrace();
    stop();
}

private void stop(){
    running.set(false);
    synchronized (running){
        running.notify();
    }
}

private void startSendFileData() {
    Runnable runnable = new Runnable() {
        public void run() {

            try {
                ByteChannel fileChannel = Files.newByteChannel(FileClient.this
                ByteBuffer buffer = ByteBuffer.allocate(1 + 4096);
                buffer.order(ByteOrder.BIG_ENDIAN);

                MessageDigest md = MessageDigest.getInstance("MD5");

                int bytesRead = -1;

                buffer.clear();//make buffer ready for write
                buffer.put((byte)FilePacket.P_FILE_DATA);

                while((bytesRead = fileChannel.read(buffer)) != -1){
                    buffer.flip(); //make buffer ready for read
                    buffer.mark();
                    buffer.get(); //skip a byte
                    md.update(buffer);
                    buffer.reset();
                    FileClient.this.wsclient.getConnection().send(buffer);
                    buffer.clear(); //make buffer ready for write
                    buffer.put((byte)FilePacket.P_FILE_DATA);
                }

                byte[] digest = md.digest();
                String digestInHex = DatatypeConverter.printHexBinary(digest)
                System.out.println("send file finished, digest: " + digestInHex
                FilePacket p = FilePacket.constructFileEndPacket(digestInHex)
                FileClient.this.wsclient.getConnection().send(p.getBuffer());
            } catch (Exception e) {
                wsclient.close();
            }
        }
    };

    new Thread(runnable).start();
}
}

```



注意事项

1. 为了清除内存byte数组拷贝，全部使用的是Java NIO的Buffer，所以要注意 `flip` 、 `clear` 、 `mark` 、 `reset` 、 `compact` 的用法，用惯了Netty的Buffer，再

用Java NIO的Buffer还真是不习惯

2. 服务器端与客户端传输了int，为了避免大小端问题，最好显式设置
 ByteOrder, `buffer.order(ByteOrder.BIG_ENDIAN);`
3. 为了提高文件操作效率，全部使用Java NIO File API，特别要注意打开文件的方式，`ByteChannel fileChannel = Files.newByteChannel(FileClient.this.filePath, EnumSet.of(StandardOpenOption.READ));`，这个跟Old File API有些不一样，在打开文件Channel时必须指定Channel的操作方式，详见 `java.nio.file.StandardOpenOption`



本文参与[腾讯云自媒体分享计划](#)，欢迎正在阅读的你加入，一起分享。
发表于 2018-05-10

分享



其他

举报



jeremy的技术点滴
198 篇文章 · 43 人订阅

订阅专栏

- 实例变量的懒初始化
- 处理一个NodeJS程序内存泄露的问题
- WEB界面测试实践之Selenium WebDriver
- Netty框架研究
- 试用docker功能

我来说两句

0 条评论

[登录](#) 后参与评论

上一篇：[教程 | 基于遗传算法的拼图游戏解决方案](#)
下一篇：[1分钟链圈 | 跌跌跌！比特币再度跌破1万美元，加密货币全线大跌；Facebook发话了，将屏蔽其平台上所有数字加密货币广告](#)

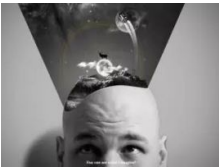
相关文章

来自专栏 钱塘大数据

理工男图解零维到十维空间，烧脑已过度，受不了啦！

让我们从一个点开始，和我们几何意义上的点一样，它没有大小、没有维度。它只是被想象出来的、作为标志一个位置的点。它什么也没有，空间、时间通通不存在，这就是零维度。

386 3 0



来自专栏 前端桃园

知识体系解决迷茫的你

最近在星球里群里有小伙伴说道自己对未来的路比较迷茫，一旦闲下来就不知道自己改干啥，今天我这篇文章就是让你觉得一天给你 25 个小时你都不够用，觉得睡觉都是浪费...

245 4 0

30人，只有我一个前端
懂了点vue，现在开发结



来自专栏 Ken的杂谈

【系统设置】CentOS 修改机器名

191 3 0

来自专栏 haifeiWu与他朋友们的专栏

复杂业务下向Mysql导入30万条数据代码优化的踩坑记录

从毕业到现在第一次接触到超过30万条数据导入MySQL的场景（有点low），就是在顺丰公司接入我司EMM产品时需要将AD中的员工数据导入MySQL中，因此楼主负...



333 4 0

分享



来自专栏 FSociety

SQL中GROUP BY用法示例

GROUP BY我们可以先从字面上来理解，GROUP表示分组，BY后面写字段名，就表示根据哪个字段进行分组，如果有用Excel比较多的话，GROUP BY比较类...

i_no	dept_no	from_date	to_date
01	d005	1986-06-26	9999-01-
02	d007	1996-08-03	9999-01-
03	d004	1995-12-03	9999-01-
04	d004	1986-12-01	9999-01-
05	d003	1989-09-12	9999-01-
06	d005	1990-08-05	9999-01-
07	d008	1989-02-10	9999-01-
08	d005	1998-03-11	2000-07-

5.2K 2 0

来自专栏 钱塘大数据

中国互联网协会发布：《2018中国互联网发展报告》

在2018中国互联网大会闭幕论坛上，中国互联网协会正式发布《中国互联网发展报告2018》（以下简称《报告》）。《中国互联网发展报告》是由中国互联网协会与中国互联...

2017年中国互联网基础设施-域名与网站	03
2017年中国互联网基础设施建设情况	04
2017年中国移动互联网应用与服务状况	05
2017年中国电子商务发展状况	06
2017年中国网络游戏发展状况	07
2017年中国网络文学发展状况	08
2017年中国网络音乐发展状况	09
2017年中国网络视频发展状况	10
2017年中国社交平台发展状况	11
2017年中国网络出行服务发展状况	12
2017年中国网络健康服务发展状况	13
2017年中国网络经济状况	14

140 5 0

来自专栏 腾讯高校合作

【倒计时7天】2018教育部-腾讯公司产学合作协同育人项目申请即将截止!

168 2 0

来自专栏 怀英的自我修炼

考研英语-1-导学

英二图表作文要重视。总体而言，英语一会比英语二难点。不过就写作而言，英语二会比英语一有难度，毕竟图表作文并不好写。



127 1 0

来自专栏 腾讯社交用户体验设计

ISUX Xcube智能一键生成H5

524 2 0

来自专栏 微信公众号：小白课代表

不只是软件，在线也可以免费下载百度文库了。

不管是学生，还是职场员工，下载各种文档几乎是不可避免的，各种XXX.docx，XXX.pptx更是家常便饭，人们最常用的就是百度文库，豆丁文库，道客巴巴这些下载...



478 3 0

社区

专栏文章

互动问答

技术沙龙

技术快讯

团队主页

开发者手册

智能钛AI

活动

原创分享计划

自媒体分享计划

资源

在线学习中心

技术周刊

社区标签


开发者实验室

关于

社区规范

免责声明

联系我们



扫码关注云+社区
领取腾讯云代金券

分享

热门产品

热门推荐

更多推荐

微信分享

QQ分享

微博分享

复制链接

二维码

实名认证

云服务器

区块链技术

消息队列

网络加速

关系型数据库

域名解析

云存储

主机

数据可视化

CDN 加速

视频转码

图片文字识别

MySQL 数据库

安全证书

语音识别

短信群发平台

文字识别

视频点播

数据安全审计

小程序开发


网络安全

学生机

域名备案

网站备案

域名监控



Copyright © 2013 – 2020 Tencent Cloud. All Rights Reserved. 腾讯云 版权所有 京ICP备11018762号京公网安备 11010802020287

