和云筹httplog系统

设计说明书

责任编撰：刘英先

审核/批准：和云筹-PMO

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **版本/状态** | **修改日期** | **修改人** | **说明** |
| V1.0 | 2016-12-07 | 刘英先 | 创建文档 |

目录

[一　引言 2](#__RefHeading__4761_213619708)

[二　项目概述 2](#__RefHeading__4763_213619708)

[三　方案的选定 3](#__RefHeading__4765_213619708)

[ELK系统的问题 3](#__RefHeading__8213_213619708)

[Dapper系统的问题 3](#__RefHeading__8215_213619708)

[实时部分选型 3](#__RefHeading__8325_213619708)

[http轮询的问题 3](#__RefHeading__8217_213619708)

[Java对websocket的支持 4](#__RefHeading__957_1486023100)

[下载部分的选型 4](#__RefHeading__8647_213619708)

[四　httplog的设计与分析 4](#__RefHeading__4767_213619708)

[Websocket与TCP 4](#__RefHeading__7825_213619708)

[Websocket 与http 5](#__RefHeading__7827_213619708)

[五　httplog的主要实现 6](#__RefHeading__651_2000936843)

[1.参数的获取 6](#__RefHeading__653_2000936843)

[2.请求与数据响应 6](#__RefHeading__655_2000936843)

[3.服务器端管道的建立 7](#__RefHeading__657_2000936843)

# 一　引言

在实际的开发调试以及运营期的线上问题处理时,大都会涉及到系统的日志,日志对于快速定位问题提供了很大的方便,但现实情况是:测试环境的日志及生产环境的日志需要通过运维小组成员来取,运维小组成员依据每位员工的权限从不同的环境取得日志后传给开发小组成员.这种模式增加了运维的工作量,同时也拉长了解决问题的时间,不利于问题的快速定位与解决.

本文从实际的情况出发,提出了一种使用httplog解决此问题的思路.

# 二　项目概述

　　本项目使用websocket 实时反馈日志,完成在浏览器上查看日志的一种方式,同时使用nginx 的功能来实现日志文件的下载.

项目支持基于浏览器(要求支持html5的websocket)的在线查看,也支持将历史日志通过http进行下载.

# 三　方案的选定

当前较为流行的日志分析系统主要包括ELK, google Dapper等,它们都在特定的领域表现出色.

## ELK系统的问题

ELK注重于日志收集统计与分析,近乎实时地将日志分析结果以图表的形式在浏览器中展示;但其部署较为麻烦,并且不能拿到日志文件,也就不能合乎运营的需求.

## Dapper系统的问题

Google Dapper 侧重于日志追踪,对于长链路的调用过程进行排序与汇总,并最终确定问题的源头.Google Dapper主要应用在分布式服务中,对于复杂调用链路服务编排的错误定位提供很好的支持,被称之为分布式系统日志追踪的神器. Google Dapper当前没有开源,仅限于其内部使用,已有公司依据其论文做了实现,如Twitter zipkin 系统,Ali 的 EagleEye 系统等,此类系统功能强大,但较为复杂,并且需要自已编码实现,成本较高,不适用于简小轻便的项目.

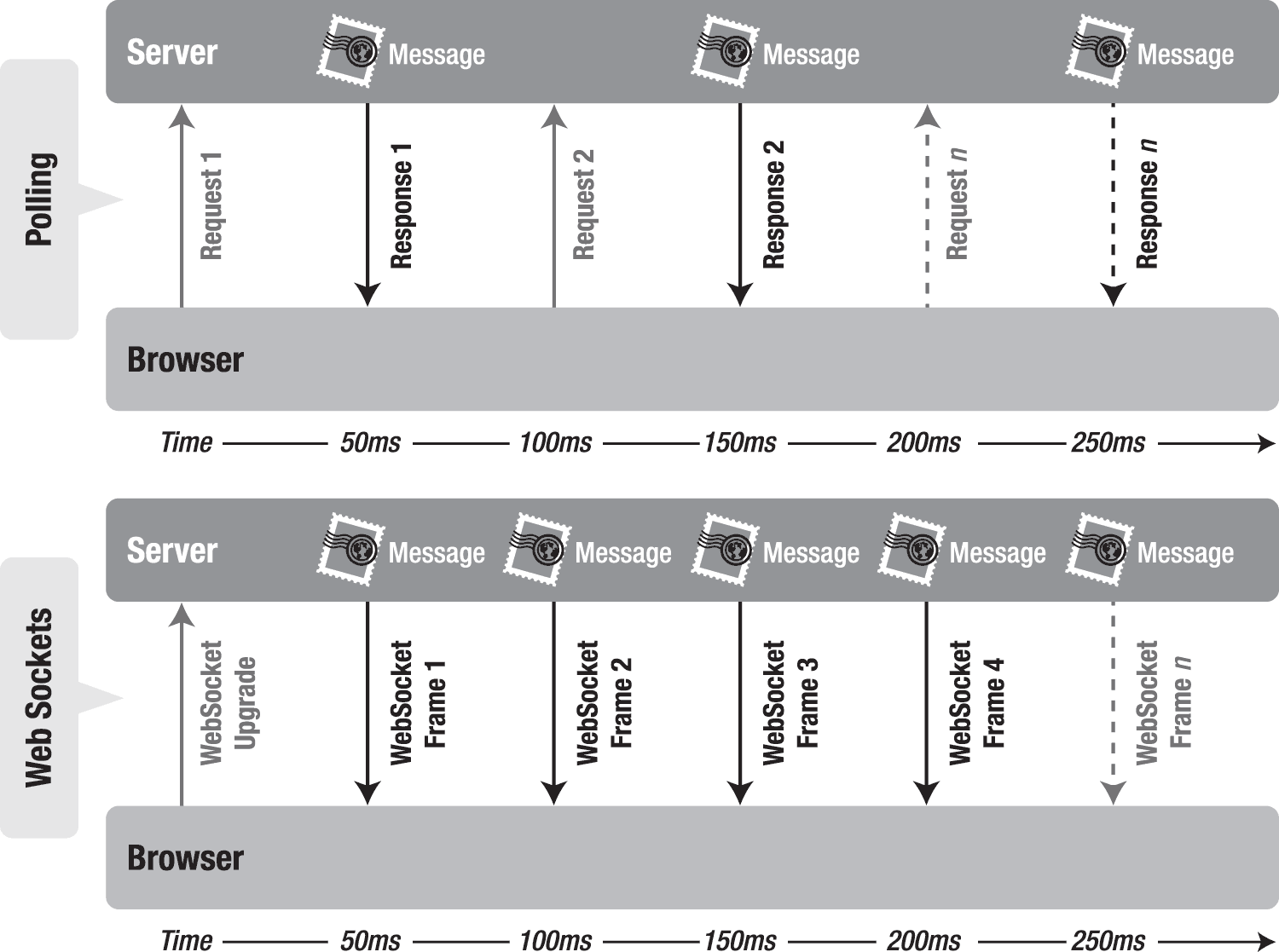
httplog 的实现分为实时日志输出部分与历史日志下载两个部分.以下简称实时部分和下载部分.

## 实时部分选型

由于要求浏览器支持,对于实时的实现可以使用websocket或者webRTC来实现,webRTC主要应用于web多媒体,功能非常强大,但主要限于Google主推,并非RFC的标准,所以本文选择了标准化的websocket来实现.

### *http轮询的问题*

使用http轮询的方式可以解决此问题,并且被所有的浏览器支持.但其实时性较差,服务器压力大.对于长时间没有输出的情况,会造成服务器资源的大量浪费,优化空间有限.websocket与http轮询的比较如下图所示



### *Java对websocket的支持*

Java对websocket的支持也很好,Java标准化委员会通过JSR356 来标准化Java API for websocket.

## **下载部分的选型**

静态文件的下载是web server最为基本的功能,Apache和Nginx都可以完成基于http的目录列示及文件下载.　由于Nginx设计的先进性,在大量并发请求时以epoll方式来处理,性能较好,已被广泛应用于各类web应用.同时也支持websocket代理,在本系统中使用了Nginx做为web文件服务器及websocket的反向代理服务.

# 四　httplog的设计与分析

httplog实时输出使用websocket来实现,日志信息可以实时地输出到浏览器.websocket作为html5的标准,为程序开发提供了方便的接口.

## Websocket与TCP

要实现websocket数据的读写,首先要解决的是它的协议问题,websocket协议是建立在TCP协议的基础上的,首先要确定其与TCP协议的关系.

websocket与TCP协议的关系如下图1所示:

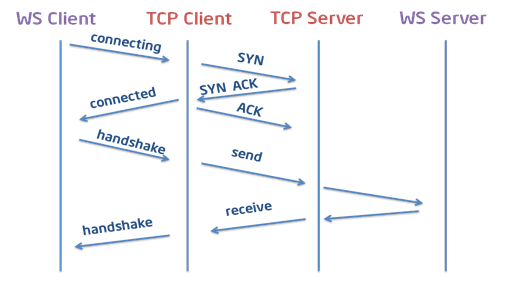
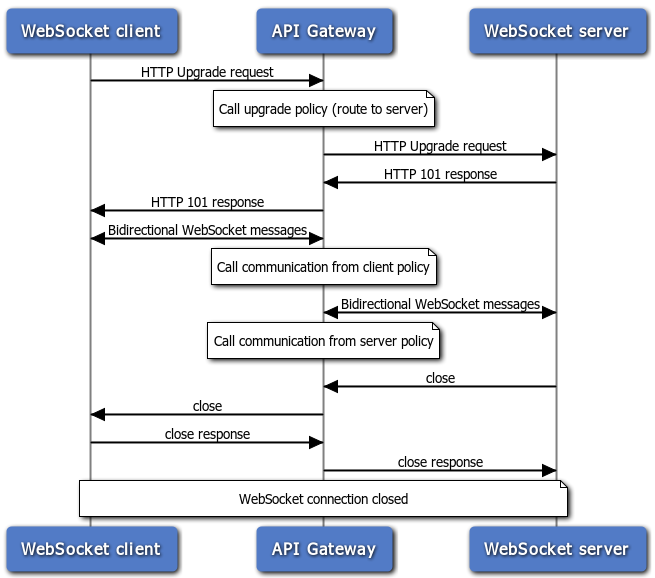


图１[1]

## Websocket 与http

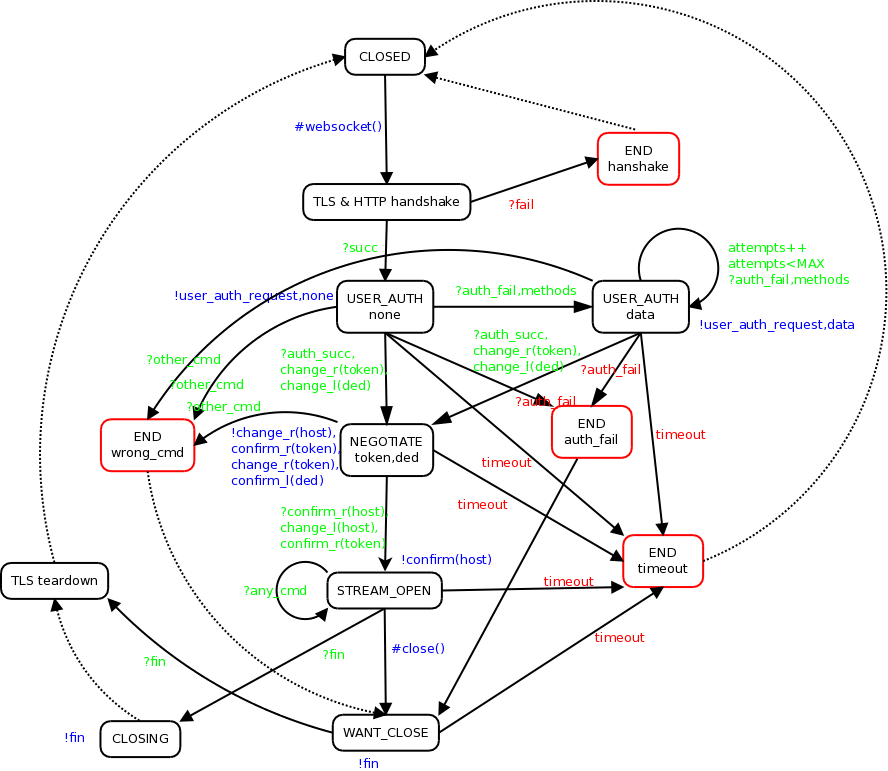
websocket是通过将http协议升级的方式来实现的,所以依赖于http 协议,其与http协议之间的调用关系如下图2所示:

图２ [1]

从以上图所示,其过程三次握手和四次挥手,与TCP协议大致相同,其先将http升级到websocket,websocket server向客户端发送http 101响应,完成连接后,客户端和服务端双向传输数据(Bidirectional Message),完整的关闭连接需要四次挥手,其实现简化为任何一端可以关闭连接(one side closes channel).

**websocket的生命周期**

理解websocket就需要了解其整个生命周期的过程及其状态的转换,在不同的生命阶段其使用状态机制来控制,



# 五　httplog的主要实现

## 参数的获取

页面启动后通过url在client取得相关参数,主要包括项目名称对应的日志目录,日志文件名称.

## 请求与数据响应

　　在client端中处理服务器发送的数据或者指令,统一归口于doEventActiong来处理,主要包括onMessage的处理及onError的处理,并控制最大输出行数,大于此threshold时将重新刷新数据.

## 服务器端管道的建立

　　服务端通过调用linux原生tail 并获取其InputStream,将其桥接到WebSocket的管道中,完成日志的传送.