OKKI

Socket.IO 协议分析及最小化实现

Ben

2019-09

一、Socket.IO 是什么？

一个基于事件的实时 B/S 双向通信库

1. Node.js 服务端
2. 浏览器 javascript 客户端

二、为什么用 Socket.IO ？

兼容所有浏览器以及不同浏览器使用不同的连接方式

重点是前端开发选择了 Socket.IO

现代浏览器：WebSocket

IE 8、IE 9：Ajax Polling

古老浏览器：Jsonp

三、在 OKKI 的使用场景

网页通知：新邮件通知、日程提醒等

四、Socket.IO 的特性（来自官网文档）

1. 可靠
2. 自动重连
3. 断线检测
4. 二进制消息
5. 命名空间：多路复用　　用不上
6. 房间：可实现消息群发　用不上

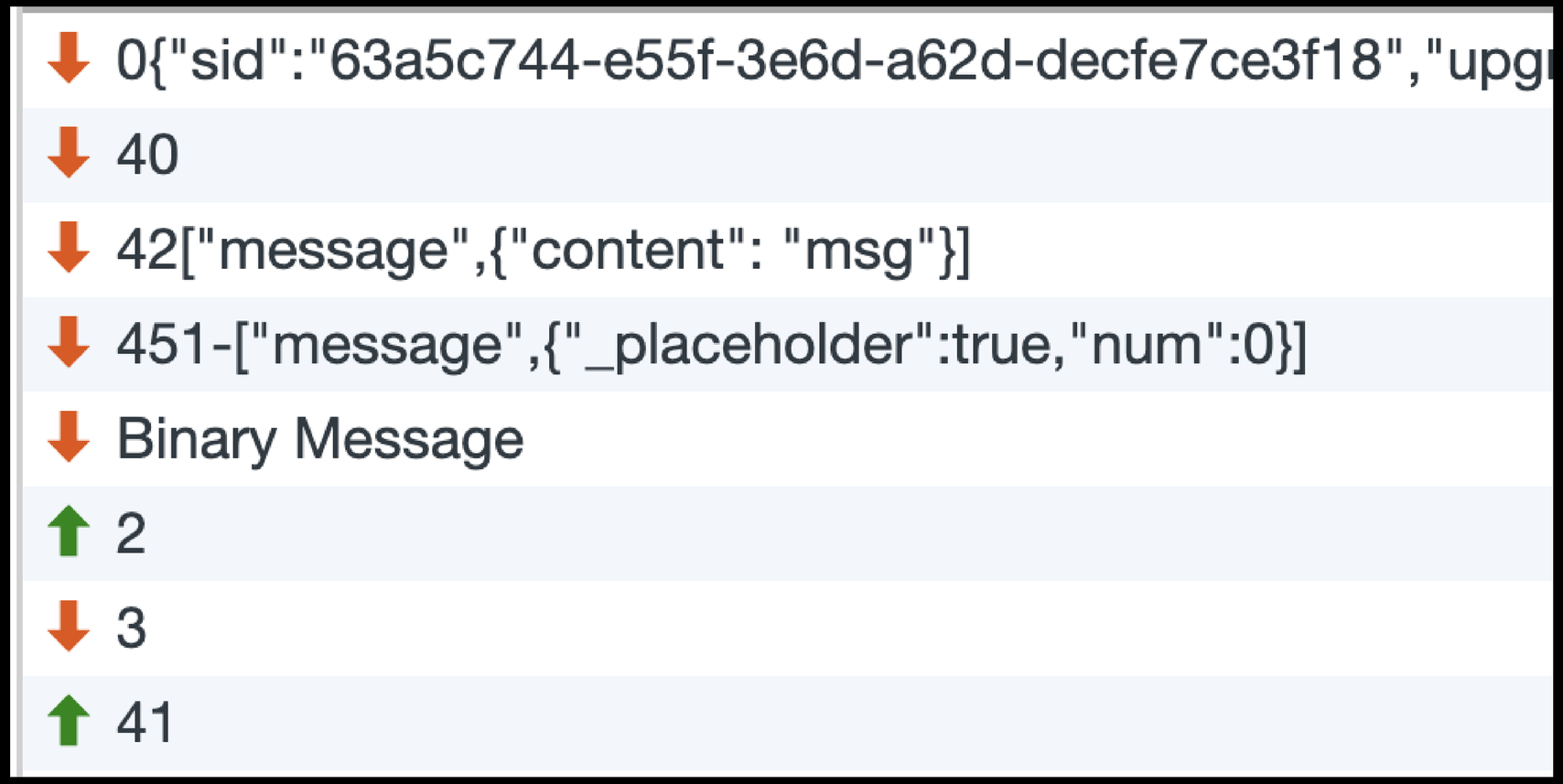
五、Socket.IO 的连接方式（可指定）

1. Websocket：只使用 Websocket

2. Polling：使用 Ajax 或 Jsonp 长轮询

3. Upgrade: 先发起一个 Polling 请求，获得 sid 后，尝试发起 Websocket，成功后关闭 Polling

六、Socket.IO 的消息交互



连接成功后发送配置信息

发送 40 表示成功

发送文本消息

发送二进制消息

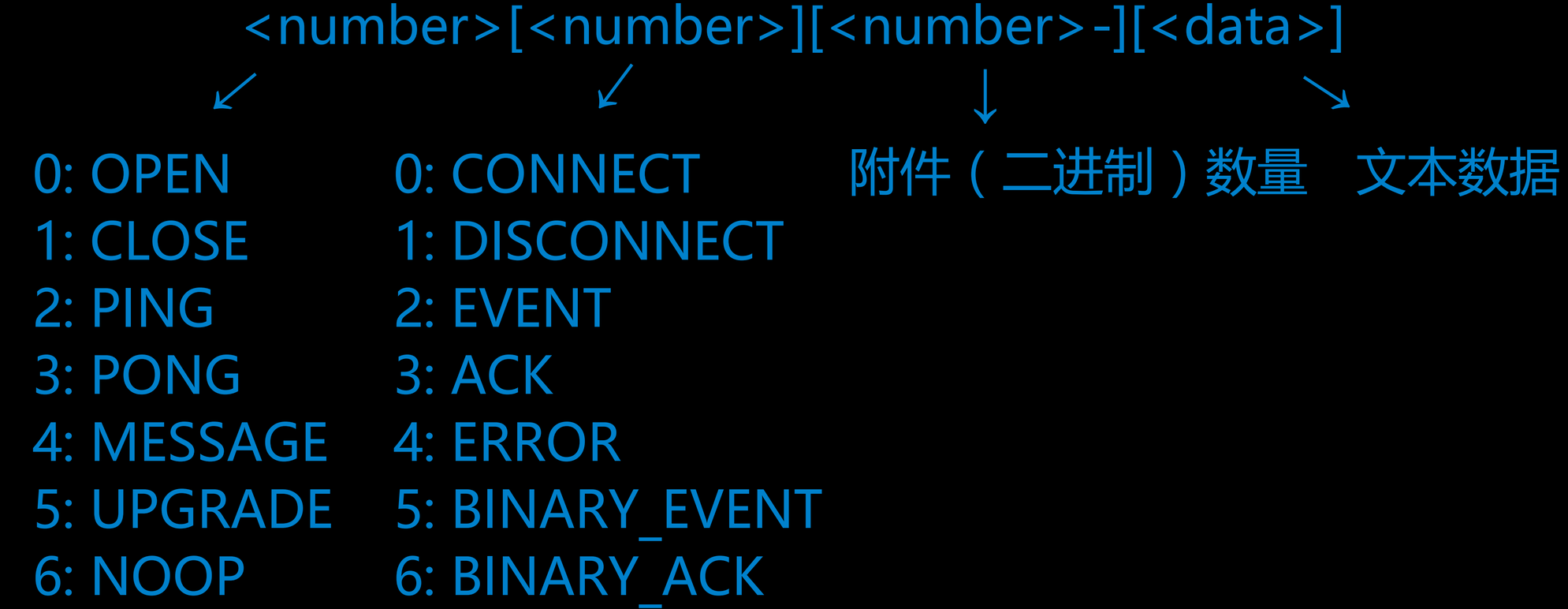
消息附件

Ping

Pong

连接断开前发送

七、消息格式


<number>[<number>][<number>-][<data>]
0: OPEN
1: CLOSE
2: PING
3: PONG
4: MESSAGE
5: UPGRADE
6: NOOP
0: CONNECT
1: DISCONNECT
2: EVENT
3: ACK
4: ERROR
5: BINARY_EVENT
6: BINARY_ACK
附件（二进制）数量
文本数据

7.1 两种消息类型

1. Packet 不带附件，为文本消息
2. Packet 带附件，主体为文本消息，附件为二进制消息

7.2 附件的处理

多个文本数据可以跟多个附件同时发送，每个二进制消息需要在文本消息里加个占位：

{ “_placeholder” : true, “num” : 0}

num 表示附件的序号，从 0 开始递增

使用 Websocket 时，需要发两帧（Frame）；

使用 Polling 时，多个消息拼在一起

7.3 服务端消息编码（Encode）

1.（Websocket 和 Poling）二进制消息加前缀 0x04

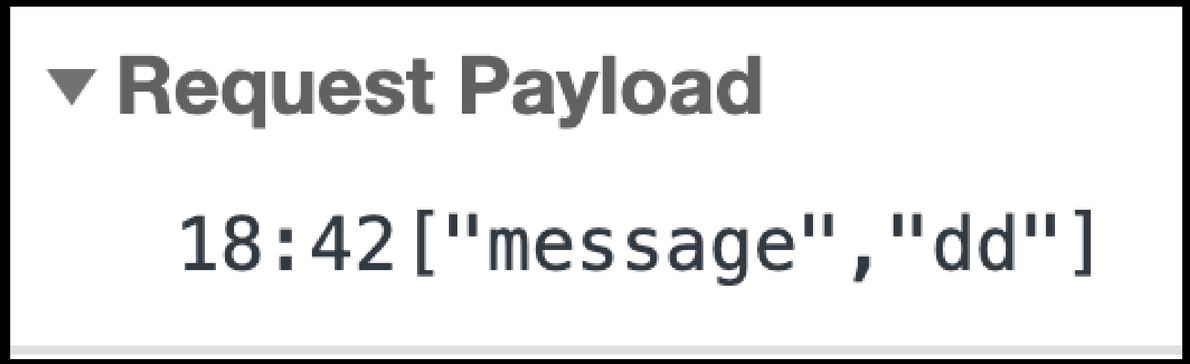
2. Polling 消息加前缀，格式如下：

文本消息：0x00 <packet 长度> 0xff
二进制消息：0x01 <packet 长度（加上 0x04 的前缀后再计算）> 0xff

如果文本消息的长度为 114，则编码后，其前缀为 0x00 0x01 0x01 0x04 0xff

7.4 客户端消息编码（Encode）

在 Polling 的情况下，客户端向服务端发送的消息需要加前缀，格式为<消息长度><:>，如下：



八、使用 Netty 实现最小化版本

根据 CRM 的需求，以下特性不实现：

1. Namespace
2. Room
3. Ack（确认消息）
4. Jsonp

8.1 注意事项

Netty Websocket 不支持带参数的 uri

8.2 核心逻辑

1. SocketIoServer，server初始化、启动和停止。
2. SocketIoRequest 解析来自客户端的 Socket.IO http 请求，区分 Websocket 和 Polling
3. SocketIoMessage，向客户端发送消息。
4. MessageType，定义了三种：AUTO（仅byte[]类型作为附件发送），BIN（所有数据类型作为附件发送），TEXT（所有类型作为文本发送）。
5 .Transport，包括 WEBSOCKET 和 POLLING。
6. Packet，消息体。
7. EncoderHandler，根据 Socket.IO 的消息格式， 作以下转换：

   transport=WEBSOCKET 时，转为 TextWebSocketFrame BinaryWebSocketFrame
   transport=POLLING，转为 FullHttpResponse。

8.3 Polling

1. 记录 sid 的状态：CONNECTING, CONNECTED, OPEN

CONNECTING: 发送配置信息，然后变为 CONNECETED

CONNECTED: 发送 40，然后变为 OPEN

OPEN: 读取阻塞队列

当客户端发送关闭请求时，清空相关缓存；定期检测不活跃的 sid

2. 待发送的消息放在本地阻塞队列，由 Pending 请求读取

THANKS

www.okki.com