Atitit 具体的im消息通讯协议xmpp lime sip等比较

目录

[1.1. 该协议定义了四种信封类型： 消息 贴纸 回话 命令 1](#_Toc11620)

[1.1.1. 认证 1](#_Toc29780)

[1.1.2. 多设备 2](#_Toc27503)

[1.1.3. 状态显示 2](#_Toc4362)

[1.2. 4.1. 连接，流的建立，身份验证以及断开连接。 6 2](#_Toc29540)

[2. Atitit xmpp协议概要 2](#_Toc14086)

[2.1. WebSocket太底层 3](#_Toc16260)

[2.2. XMPP 文本 标准化的im协议 3](#_Toc12680)

[2.3. 2SIP 多用于VOIP相关的模块，是一种文本协议 但非结构化 4](#_Toc26608)

[2.4. 协议如何制定？ 5](#_Toc21134)

[2.5. MQTT 二进制协议 并不是一个专门为IM设计的协议 5](#_Toc4001)

[2.6. 4私有消息协议 5](#_Toc11862)

[2.7. 5结论 使用json化XMPP +适度私有化+websocket/rest 5](#_Toc10845)

[2.7.1. Json化XMPP节：含义明确，标准规范 6](#_Toc24580)

[2.7.2. 简化使用xmpp子集，以及可能需要很少的扩展 6](#_Toc27466)

[2.8. MQTT 协议本身过于简单 6](#_Toc2161)

## 该协议定义了四种信封类型： 消息 贴纸 回话 命令

* **[消息](https://limeprotocol.org/" \l "message)** -提供内容在网络中节点之间的传输。
* **[通知](https://limeprotocol.org/" \l "notification)** -通知与消息相关的事件。
* **[命令](https://limeprotocol.org/" \l "command)** -允许操纵节点资源，例如服务器会话参数或与网络节点有关的信息。
* **[会话](https://limeprotocol.org/" \l "session)** -允许在节点之间配置和建立通信通道。

### 认证

* XMPP: 通过 [SASL RFC4422](https://tools.ietf.org/html/rfc4422" \t "_blank)
* MQTT: 协议未定义，通常的实现方式是定制 CONNECT 报文中的 username 和 password 字段，因为 MQTT 只提供了这两个字段可以自行发挥。比如定义 username 为 {"uname": "Shawn", "dtype": "mobile"}, password 为 JWT。(MQTT 5.0 里添加了 AUTH 消息类型)。然后在 MQTT 后台处理 CONNECT 消息进行认证。  
  在设计的时候是需要仔细考虑的部分。

### 多设备

* XMPP: 核心协议里有定义，通过 resource 实现。比如用户 [shawn@emqx.io](mailto:shawn@emqx.io" \t "_blank) 的两个设备分别是： [shawn@emqx.io](mailto:shawn@emqx.io" \t "_blank)/pc, [shawn@emqx.io](mailto:shawn@emqx.io" \t "_blank)/mobile。
* MQTT: 协议未定义。因为 MQTT 协议里，唯一标识设备使用的是 Client ID，所以设计的难点在于，如何定义 Client ID 的格式，并且能够跟 IM 需求里的 username 做对应，并且实现多终端同步。常见的做法是：
  + 限制用户名里不能有特殊字符 '@', '/';
  + 对于 ClientID 的格式为 'username@device', 比如用户 "shawn" 的两个 ClientID: "shawn@pc", "shawn@mobile"。
  + 对于用户 shawn 的这两个终端，通过订阅相同的主题来同步所有别人发来的聊天消息。比如都订阅主题：'p2p/talk/shawn'。并且向外发送消息时需要将消息复制到主题 'p2p/talk/shawn' 以同步发出的消息，这样做时，还需要忽略自己发出的同步消息。

### 状态显示

* XMPP: [XMPP Presence RFC 3921](https://tools.ietf.org/html/rfc3921" \t "_blank).
* MQTT: 协议未定义，可以通过订阅对方状态来实现：比如用户

## 4.1. 连接，流的建立，身份验证以及断开连接。 6

****LIME协议****（Lightweight Messaging Protocol的缩写）是一种基于JSON的消息传递协议

，受XMPP协议启发。它的目标是简单但可扩展，没有太多的冗长但仍提供良好的可读性。

# Atitit xmpp协议概要

目录

1. Xep XMPP扩展协议 1

1.1. XEP-0295：XMPP的JSON编码 1

2. XMPP节： 1

2.1. 核心节：<message><presene><iq> 1

2.2. 节属性： 2

2.3. 支持通用属性：from，to，type，id。 2

3. 核心节： 3

3.1. 1）、presence 3

3.2. 2）、message 4

3.3. ）、iq info/query（信息与查询），他给xmpp童心提供请求和响应机制 4

3.4. 、error在错误提示节中， 5

4. xmpp连接： 6

4.1. 连接，流的建立，身份验证以及断开连接。 6

## WebSocket太底层

## **XMPP 文本 标准化的im协议**

XMPP 是老牌的即时聊天协议，像 [Google Hangouts](https://searchmobilecomputing.techtarget.com/definition/Google-Hangouts" \t "https://www.jianshu.com/p/_blank), [WhatsApp Messenger](https://searchmobilecomputing.techtarget.com/definition/WhatsApp" \t "https://www.jianshu.com/p/_blank) 等即时聊天程序都是基于 XMPP 的。

Jabber是一个开放源码形式组织产生的网络即时通信协议。XMPP原本是为即时通讯而量身定制，但由于XML Stanza本身是XML元素，在基于XML灵活发展的特性下，使得XMPP也可以适用其他方面，已经得到了IETF的批准。XMPP与[IMPP](https://zh.wikipedia.org/w/index.php?title=IMPP&action=edit&redlink=1" \o "IMPP（页面不存在）)、[PRIM](https://zh.wikipedia.org/w/index.php?title=PRIM&action=edit&redlink=1" \o "PRIM（页面不存在）)、[SIP](https://zh.wikipedia.org/wiki/%E4%BC%9A%E8%AF%9D%E5%8F%91%E8%B5%B7%E5%8D%8F%E8%AE%AE" \o "会话发起协议)（[SIMPLE](https://zh.wikipedia.org/wiki/SIMPLE" \o "SIMPLE)）合称四大IM协议主流，在此4大协议中，XMPP是最灵活的。

1XMPP

****优点：****协议开源，可拓展性强，在各个端(包括服务器)有各种语言的实现，开发者接入方便；  
****缺点：****缺点也是不少，XML表现力弱、有太多冗余信息、流量大，实际使用时有大量天坑。

简介：基于XML协议的通讯协议，前身是Jabber，目前已由IETF国际标准化组织完成了标准化工作。  
优点：协议成熟、强大、可扩展性强、目前主要应用于许多聊天系统中，且已有开源的Java版的开发实例androidpn。  
缺点：协议较复杂、冗余（基于XML）、费流量、费电，部署硬件成本高。

SIP (Session Initiation Protocol)

SIP是一个应用层协议,是用在类似VOIP这样的场合,

用来建立,修改,中止会话,同时在多人会议中他也能在已有会话中加入新的会话.

基本上SIP是VOIP中的信令协议,它处理呼叫建立,呼叫转移和产生CDR(Call Detail Record,供通话计费用).

XMPP (Extensible Messaging Presence Protocol)

XMPP是一个为即时通讯和请求响应业务服务的XML协议.

最早由Jabber开源社区在1999年开发,2002年XMPP工作组为了更适合即时通讯对Jabber进行了扩展.

## 2SIP 多用于VOIP相关的模块，是一种文本协议 但非结构化

SIP协议多用于VOIP相关的模块，是一种文本协议，由于我并没有实际用过，所以不做评论，但从它是文本协议这一点几乎可以断定它的流量不会小。

 SIP的信令和消息传送是基于文本的，不太好解析,或者说解析起来缺少规律性,在新增数据消息体的时候缺少继承性,需要开发新的代码来封装和解析,原有代码的继承性比较差。

MQTT gcm

SIP信令由header和body两部分组成，也就是说，SIP报文格式的header已经包含了部分内容,类似于HTTP,与具体的上层应用直接关联，而不是通用的报文格式；

而XMPP所有信息都是采用XML在流管道之间透明传送。

SIP的连接建立通道与数据传送通道是各自独立的，连接建立在SIP client与Server之间，而数据传送通道是在Client--Client之间直接进行的。

这个对视频、语音和文件传送业务很合适，但是不适合其他形式的应用。

XMPP的控制和数据通道是一体的，Clent只与Server建立连接，而Client与Client之间是没有之间连接的。Client之间传送的通道是：Client1---〉Server1---〉Server2---〉Client2。这种方式看起来扩展性差，server压力很大，但是能够实现很好的业务功能，比如留言、广播、群聊、状态更新、Blog、微博、数据共享等等。

这种C-S模型，很多业务的控制在Server上完成，新功能的增加在server上实现，在server上定义新的XML对象和逻辑，客户端只要负责XML数据流的解析和呈现就可以了, 所以，终端实现简单

## 协议如何制定？

IM协议选择原则一般是：易于拓展，方便覆盖各种业务逻辑，同时又比较节约流量。后一点的需求在移动端IM上尤其重要。常见的协议有：XMPP、SIP、MQTT、私有协议。（更多关于即时通讯应用的协议选择，请参见《如何选择即时通讯应用的数据传输格式》：[http://www.52im.net/thread-276-1-1.html](http://www.52im.net/thread-276-1-1.html" \t "http://www.52im.net/_blank)）

## MQTT 二进制协议 并不是一个专门为IM设计的协议

****优点：****协议简单，流量少；  
****缺点：****它并不是一个专门为IM设计的协议，多使用于推送。

****MQTT****

MQTT（Message Queue Telemerty Transport）[2]是一种二进制协议，主要用于服务器和那些低功耗的物联网设备（IoT）之间的通信

## 4私有消息协议

市面上几乎所有主流IM APP都是是使用私有协议，一个被良好设计的私有协议优点非常明显。  
  
****优点：****高效，节约流量(一般使用二进制协议)，安全性高，难以破解；  
****缺点：****在开发初期没有现有样列可以参考，对于设计者的要求比较高。

## 5结论 使用**json化XMPP +适度私有化+websocket/rest**

一个好的协议需要满足如下条件:高效，简洁，可读性好，节约流量，易于拓展，同时又能够匹配当前团队的技术堆栈。基于如上原则，我们可以得出: 如果团队小，团队技术在IM上积累不够可以考虑使用XMPP或者MQTT+HTTP短连接的实现。反之可以考虑自己设计和实现私有协议。

### ****Json化XMPP节：含义明确，标准规范****

核心节：<message><presene><iq>  
1、<stream>  xmpp以其标记开始。  
2、<presence>用来表明用户状态。可以来广播或“发布-订阅”；<status>网络状态、<show>可用状态、<priority>资源优先级。  
3、<iq>一种请求/响应机制。  
4、<message>用于两个jabber用户发送消息。<body>是消息内容；他包含<subject/>（消息标题）<body/>（消息内容）<thread/>(跟踪会话线索)子标签。

简单会话：  
<stream:stream>    创建一个xmpp流  
<iq type=’get’>    请求elizabeth的花名册，也就是她的所有已存储的联系人列表。  
<query xmlns=’jabber:iq:roster’/>  
</iq>  
<presence/>             通知服务器她已经在线并可以访问。  当他注意到darcy在线的时候，就会发个message节，  
<message to=’darcy@pemberley.lib’  
from=’ellizabaeth@longbourn.lit/ballroom’  
type=’chat’>  
<body>sdf</body>  
</message>  
<presence type=’unavailable’/>  告诉服务器，这是不可访问的要关闭  
</stream:stream>

### 简化使用xmpp子集，以及可能需要很少的扩展

## MQTT 协议本身过于简单

另一方面虽然 MQTT 协议本身过于简单，放在实时聊天协议里的话，多数功能是要自己定制协议来实现的，但总归可以用。

IM 功能

XMPP 在 IM 的各个功能上的定义非常完善，而 MQTT 需要在基础协议之上自行实现。

WebSocket子协议STOMP详解 - FEINIK的个人主页 - OSCHINA - 中文开源技术交流社区