NexTalk消息服务器设计

文档版本: 5.6 发布时间: 2014/10/15

架构简介

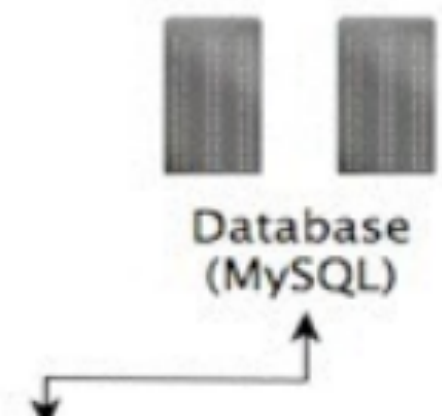
NexTalk 基于WEB标准协议架构设计，目标是为WEB站点或应用快速便捷的集成即时消息。

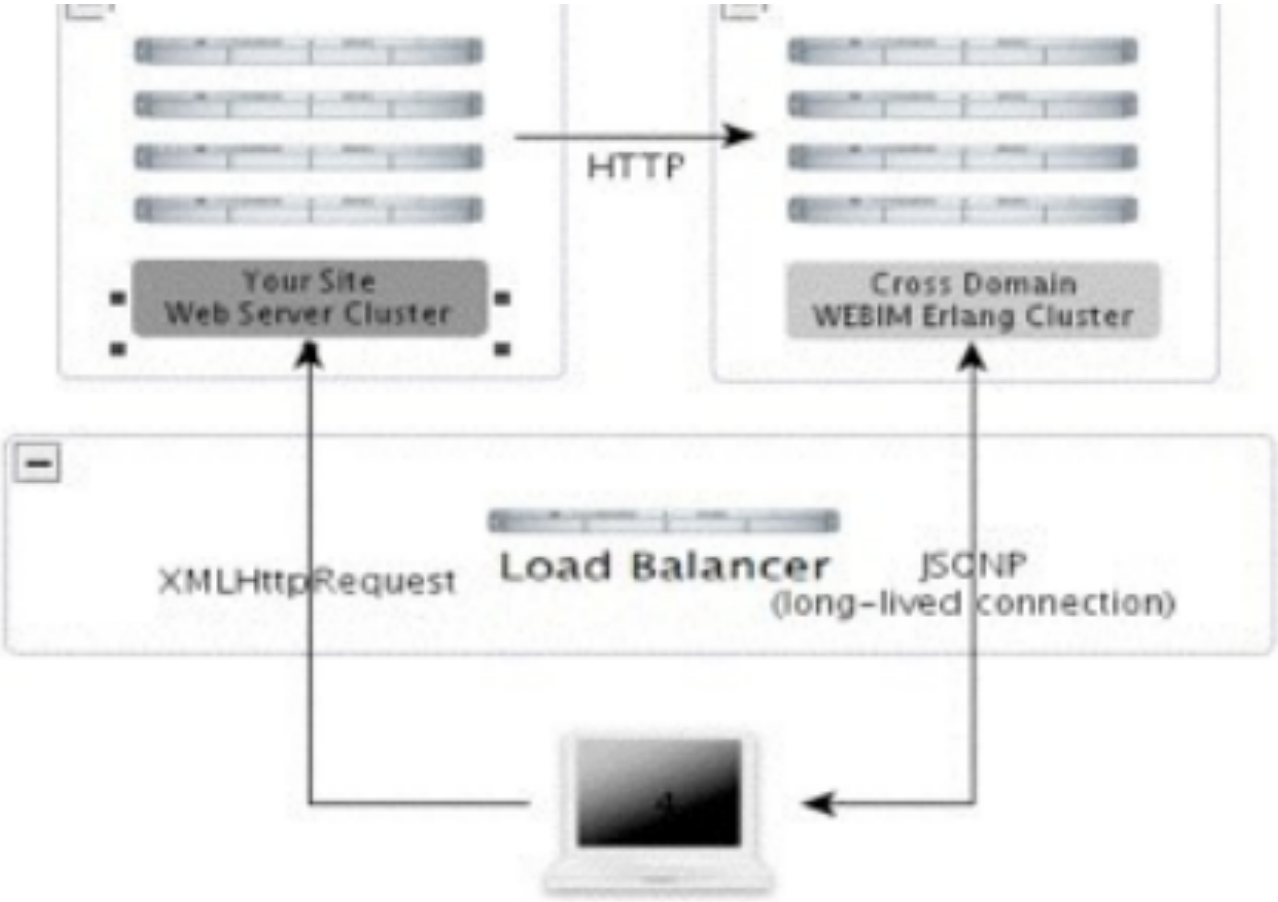
NexTalk架构上分离WebIM服务器与消息服务器，客户站点的WEB服务器与NexTalk提供的WebIM插件集成，NexTalk消息服务器直接负责浏览器长连接和消息路由。

WebIM插件负责与网站的用户资料、好友关系、群组关系集成，访问站点的数据库。NexTalk在WebIM on GitHub上提供了丰富的程序开发库和框架集成包，包括PHP、Java、.Net、Python、Ruby等。

消息服务器负责处理跨域的长轮询(JSONP)或长连(WebSockt)，用户登录社区网站的WebIM后，用户浏览器直接与消息服务器建立连接，完全不占用站点的连接资源。

消息服务器采用Erlang并发编程语言开发，单服务器可支持10万以上在线连接，集群服务器可支持100万在线连接。





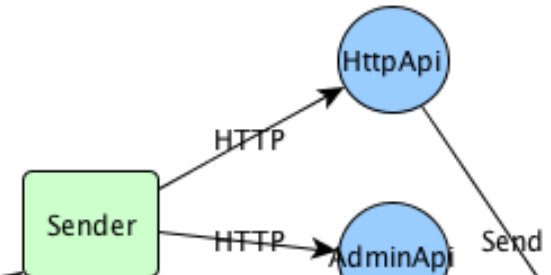
消息服务器设计

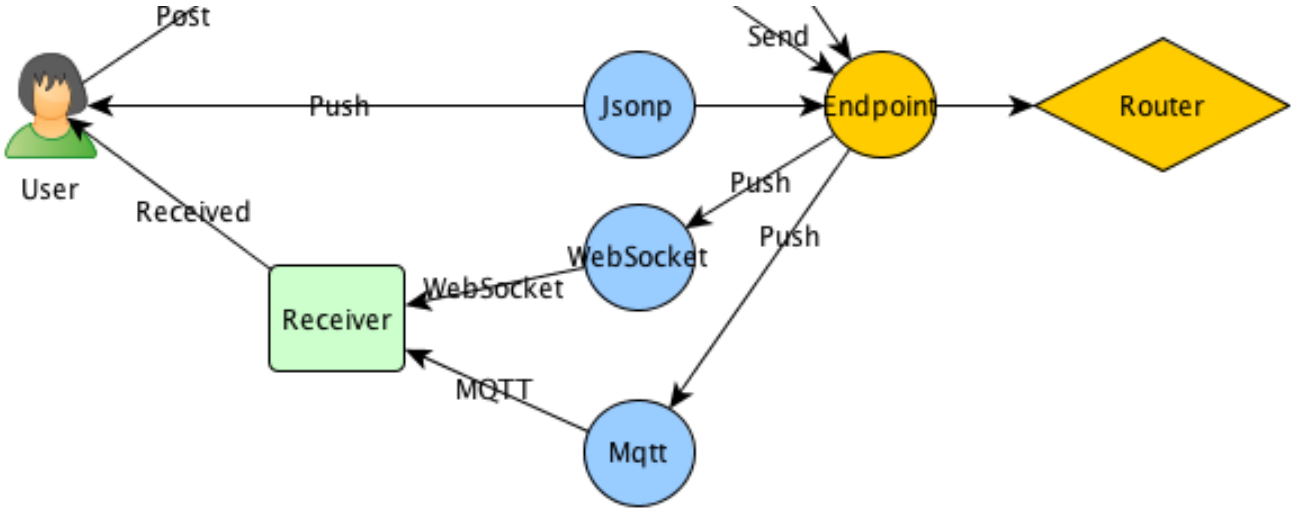
NexTalk消息服务器采用Erlang并发编程语言开发，低系统资源消耗，支持大量并发连接。

消息服务器的核心是MQTT协议，基于MQTT定义的Topic匹配方式进行消息路由。主要模块包括HTTPD模块、JSONPD模块、 WebSocket模块、路由模块、MQTT模块、认证模块和统计模块。

消息服务器提供了多协议的外部接口，包括HTTP REST接口、JSONP轮询接口、WebSocket接口、MQTT客户端接口。

消息服务器支持多服务器集群，可平滑扩展到100万连接用户规模。





线上用户、访客、群组、服务可表述为一个Endpoint进程，消息通过HTTP接口发送，推送给Jsonp、Websocket、MQTT等接收接口。

消息路由设计

NexTalk5版本的消息路由机制，基于MQTT协议设计，以方便支持手机和桌面。

MQTT协议规范详见: http://mqtt.org

Address设计

用户、访客、群组(聊天室)、服务表述为一个Endpoint对象，Endpoint对象唯一的寻址标识为:

```
domain/{domain}
    /{node}/{nodeId}
        /client/{clientId}
```

举例:

```
/domain/nextalk.im/uid/1/client/chrome
/domain/nextalk.im/vid/219392/client/android
```

domain	域名
node	节点(uid,vid,gid,sid枚举)
nodeId	用户ID、访客ID、群组ID等
clientId	登录终端标识，例如Android,浏览器

Queue设计

Queue名称	Queue结构
用户Queue	/domain/${domain}/uid/${uid}
群组Queue	/domain/${domain}/gid/${gid}
服务Queue	/domain/${domain}/sid/${sid}
访客Queue	/domain/${domain}/vid/${vid}

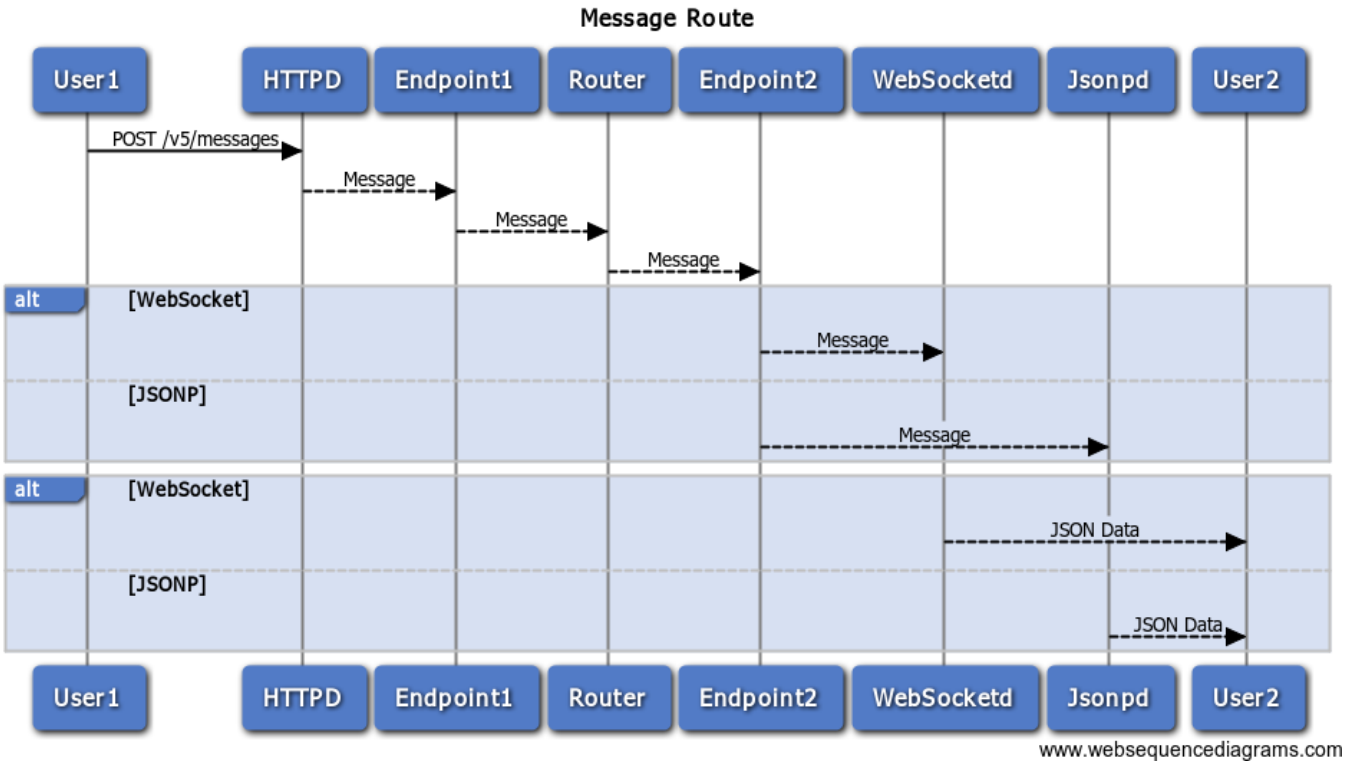
路由设计

```
JSONP <————————————                  ————————————Desktop
                   |                 |
MQTT <———————————— Router <———HTTP————————Browser
                   |                 |
WebSocket <————————                  ————————————Mobile
```

消息流程

```
title Message Route
User1->HTTPD: POST /v5/messages
```

```
Endpoint1-->Router: Message
Router-->Endpoint2: Message
alt WebSocket
 Endpoint2-->WebSocketd: Message
else JSONP
 Endpoint2-->Jsonpd: Message
end
alt WebSocket
 WebSocketd-->User2: JSON Data
else JSONP
 Jsonpd-->User2: JSON Data
end
```



HTTP接口设计

WebIM插件与消息服务器间的HTTP REST接口，WebIM插件转发用户上下线、现场、消息、状态 给消息服务器，从服务器读取当前在线用户、群组成员等信息。HTTP接口列表如下:

标识	名称	方法	URL	说明

ONLINE	用户上线接口	POST	/v5/presences/online	
OFFLINE	用户下线接口	POST	/v5/presences/offline	
SHOW	现场变更接口	POST	/v5/presences/show	
PRESENCES	读取现场接口	GET	/v5/presences	
MESSAGES	发送消息接口	POST	/v5/messages	

STATUSES	发送状态接口	POST	/v5/statuses	
MEMBERS	读取群组成员接口	GET	/v5/rooms/{roomId}/members	
JOIN	加入群组接口	POST	/v5/rooms/{roomId}/join	
LEAVE	离开群组接口	POST	/v5/rooms/{roomId}/leave	

HTTP基本认证

WebIM插件与消息服务器间通过HTTP基本认证。用户名为网站域名(Domain)，口令为网站申请的APIKEY

ONLINE(用户上线接口)

概述

WebIM插件通过该接口转发用户上线请求，同时把用户的好友关系、群组关系发送给消息服务器。

请求URL

```
HTTP POST /v5/presences/online
```

请求参数

名称	类型	是否必须	描述
domain	string	是	域名
name	string	是	用户uid或name
nick	string	是	用户昵称
show	string	否	available, unavailable,busy, away, chat, dnd, invisible
status	string	否	用户当前状态信息
buddies	string	是	用户好友uid列表，逗号分割。例如: 1,2,3,vid:abc
rooms	string	否	用户群组id列表，逗号分割。例如: room1,room2,room3…

成功返回

HTTP 200, JSON数据:

```
{“success”: true,
 “ticket”: “uid:demo:0df37e76e75f9d66a6c7”,
 “server”: “http://nextalk.im:8080/v5/packets”,
 “jsonpd”: “http://nextalk.im:8080/v5/packets”,
 “websocket”: “http://nextalk.im:8080/v5/wsocket”,
 “presences”: {“uid1”: “available”, “2”: “away”}
}
```

数据说明:

属性	类型	是否必须	描述
success	boolean	是	请求成功
ticket	string	是	浏览器与消息服务器通信的令牌
server	string	否	消息服务器轮询接口，用于兼容老版本
jsonpd	string	是	消息服务器Jsonpd接口地址
websocket	string	是	消息服务器WebSocket接口地址
presences	string	是	返回好友Presence状态数据，属性是好友uid，值

是Show

失败返回

```
HTTP 400 {“status: “error”, “message: “bad request”}
HTTP 401 {“status”: “error”, “message”: “authenticati
on failed"}
```

OFFLINE(用户下线接口)

概述

WebIM插件通过该接口转发用户下线请求。

请求URL

```
HTTP POST /v4/presences/offline
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
ticket	string	是	网站与消息服务器通信令牌

成功返回

```
HTTP 200 {“status”: “ok”}
```

失败返回

```
HTTP 401
HTTP 404
```

SHOW(现场变更接口)

概述

WebIM插件转发用户现场变更请求，用户现场枚举包括:

值	名称
available	下线
away	离开
chat	聊天中
dnd	忙碌
invisible	隐身
unavailable	下线

请求URL

```
HTTP POST /v5/presences/show
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
ticket	string	是	网站与消息服务器通信令牌
nick	string	否	用户昵称

show	string	是	用户现场show
status	string	否	用户当前状态

成功返回

```
HTTP 200 {“status”: “ok”}
```

PRESENCES(读取现场信息接口)

概述

WebIM插件向消息服务器发送读取用户现场信息请求，传入逗号分割的用户id列表。

请求URL

```
HTTP GET /v5/presences
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
ids	string	是	逗号分割的用户id列表，例如: “1,2,vid:328”

成功返回

HTTP 200 JSON数据:

```
{
    “uid1”: “available”,
    “uid2”: “dnd”,
    “vid:123”: “away”
}
```

MESSAGES(发送消息接口)

概述

WebIM插件向消息服务器转发用户即时消息，由消息服务器来路由分发到目的用户。

请求URL

```
HTTP POST /v5/messages
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
ticket	string	是	网站与消息服务器通信令牌
to	string	是	目的用户，消息接收者
body	string	是	即时消息内容
nick	string	否	用户当前昵称
style	string	否	消息css
timestamp	string	是	即时消息发送时间
type	string	否	即时消息类型：chat

成功返回

HTTP 200:

```
{“status”: “ok”}
```

PUSH(推送消息接口)

概述

WebIM插件的消息推送接口，5.2版本新增。该接口与消息转发接口相同，但用户无需上线获得 Ticket，即可使直接推送消息。

请求URL

```
HTTP POST /v5/messages/
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
from	string	是	消息发送者
to	string	是	目的用户，消息接收者
body	string	是	即时消息内容
nick	string	否	用户当前昵称
style	string	否	消息css
timestamp	string	是	即时消息发送时间

成功返回

HTTP 200

```
{“status”: “ok”}
```

STATUSES(发送状态接口)

概述

WebIM插件向消息服务器转发当前输入状态消息，例如：“正在输入…”。

请求URL

```
HTTP POST /v5/statuses
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
apikey	string	是	网站与消息服务器通信apikey
ticket	string	是	网站与消息服务器通信令牌
to	string	是	目的用户，状态接收者
nick	string	是	用户昵称
show	string	是	例如“用户a正在输入…"

成功返回

HTTP 200

```
{“status”: “ok”}
```

MEMBERS(读取群组成员接口)

概述

WebIM插件向消息服务器请求，读取群组在线成员现场列表。

请求URL

```
HTTP GET /v5/rooms/${roomId}/members
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
ticket	string	是	网站与消息服务器通信令牌
roomId	string	是	群组gid

成功返回

HTTP 200 JSON数据:

```
{
“uid1”: “available”,
“uid2”: “dnd”
}
```

JOIN(加入群组接口)

概述

WebIM插件向消息服务器转发加入群组请求

请求URL

```
HTTP POST /v5/rooms/${roomId}/join
```

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
ticket	string	是	网站与消息服务器通信令牌
roomId	string	否	群组gid

成功返回

HTTP 200:

```
{“status”: “ok”}
```

Leave(离开群组接口)

概述

WebIM插件向消息服务器转发离开群组请求。

HTTP POST /v5/rooms/${roomId}/leave

请求参数

名称	类型	是否必须	描述
domain	string	是	网站域名
ticket	string	是	网站与消息服务器通信令牌
roomId	string	否	群组gid

成功返回

HTTP 200:

```
{“status”: “ok”}
```

推送接口

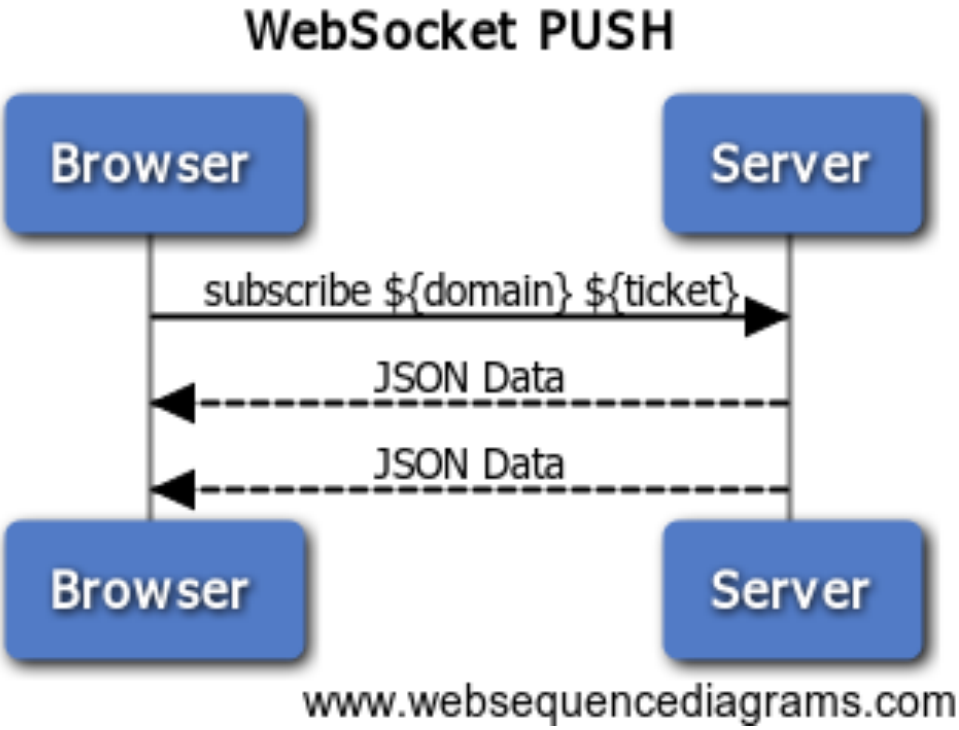
WebSocket

用户上线后，前端JS代码检测浏览器是否支持WebSocket，如支持直接与消息服务器尝试建立 WebSocket长连接。

WebSocket长连接建立后，浏览器与消息服务器间的交互流程:

```
title WebSocket PUSH

Browser->Server: subscribe ${domain} ${ticket}
Server-->Browser: JSON Data
Server-->Browser: JSON Data
```



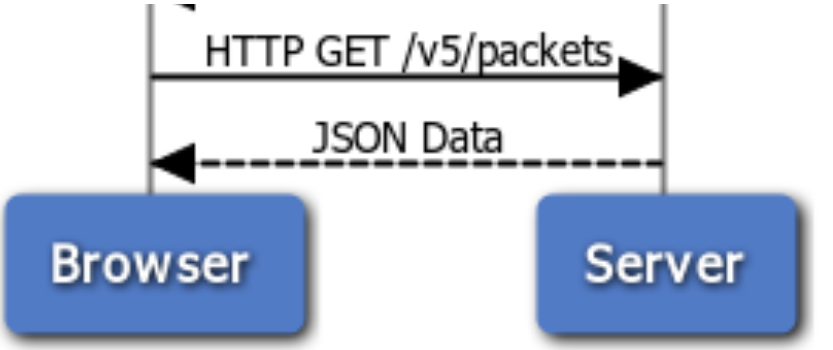
JSONP

如前端JS检测浏览器不支持WebSocket，会退化通过JSONP长轮询与消息服务器通信.

JSONP长轮询交互流程:

```
title JSONP PUSH

Browser->Server: HTTP GET /v5/packets
Server-->Browser: JSON Data
Browser->Server: HTTP GET /v5/packets
Server-->Browser: JSON Data
```



www.websequencediagrams.com