论实时应用开发的『正确』姿势

Rainbow—基于Tornado打造的长连接代理服务器

jeffkit@hoge.cn

关于@jeff_kit

- 珠三角技术沙龙化石组委
- 连续（未成功 -_|||）创业者
- 厚建云计算广州公司总经理
- 略懂Python，长大后想当全栈多 ~~妻~~ 栖工程师
- PyCon老朋友


懒


叮当
dingdone

关于实时（长连接）应用

智能手机推送

即时聊天

监控仪表板

etc …

关于Socket 编程

异步的编程模型

管理长连接的状态

保证消息传输的可靠性

不容易

寻找最偷懒的开发方式

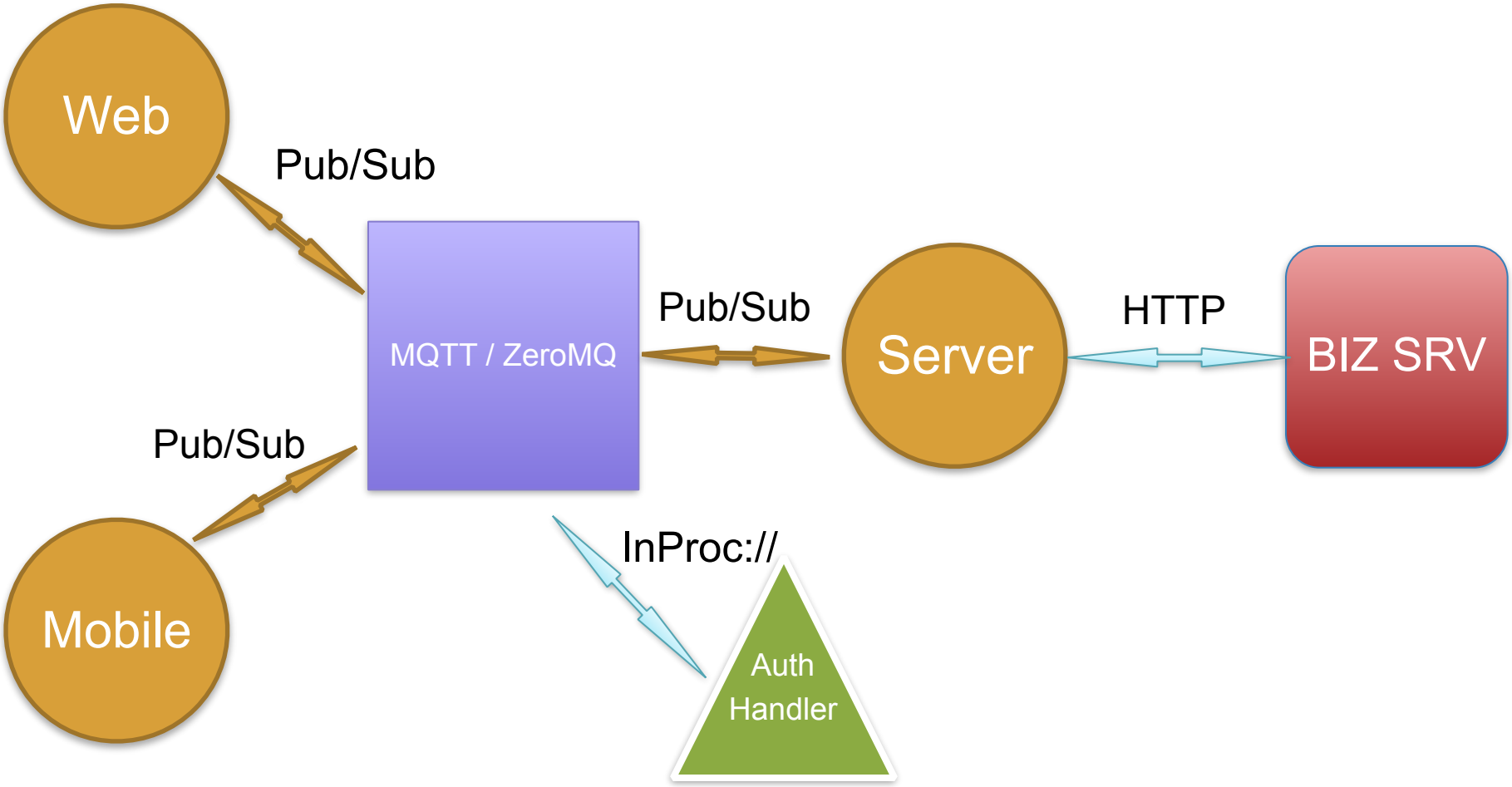
像HTTP应用那样简单!

客户端：Request/Response模式

服务端：写RESTFUL API

业务逻辑随时更新，不影响用户连接

MQTT ? ZeroMQ?


Web
Pub/Sub
MQTT / ZeroMQ
Pub/Sub
Server
HTTP
BIZ SRV
Pub/Sub
Mobile
InProc://
Auth
Handler

Why not!

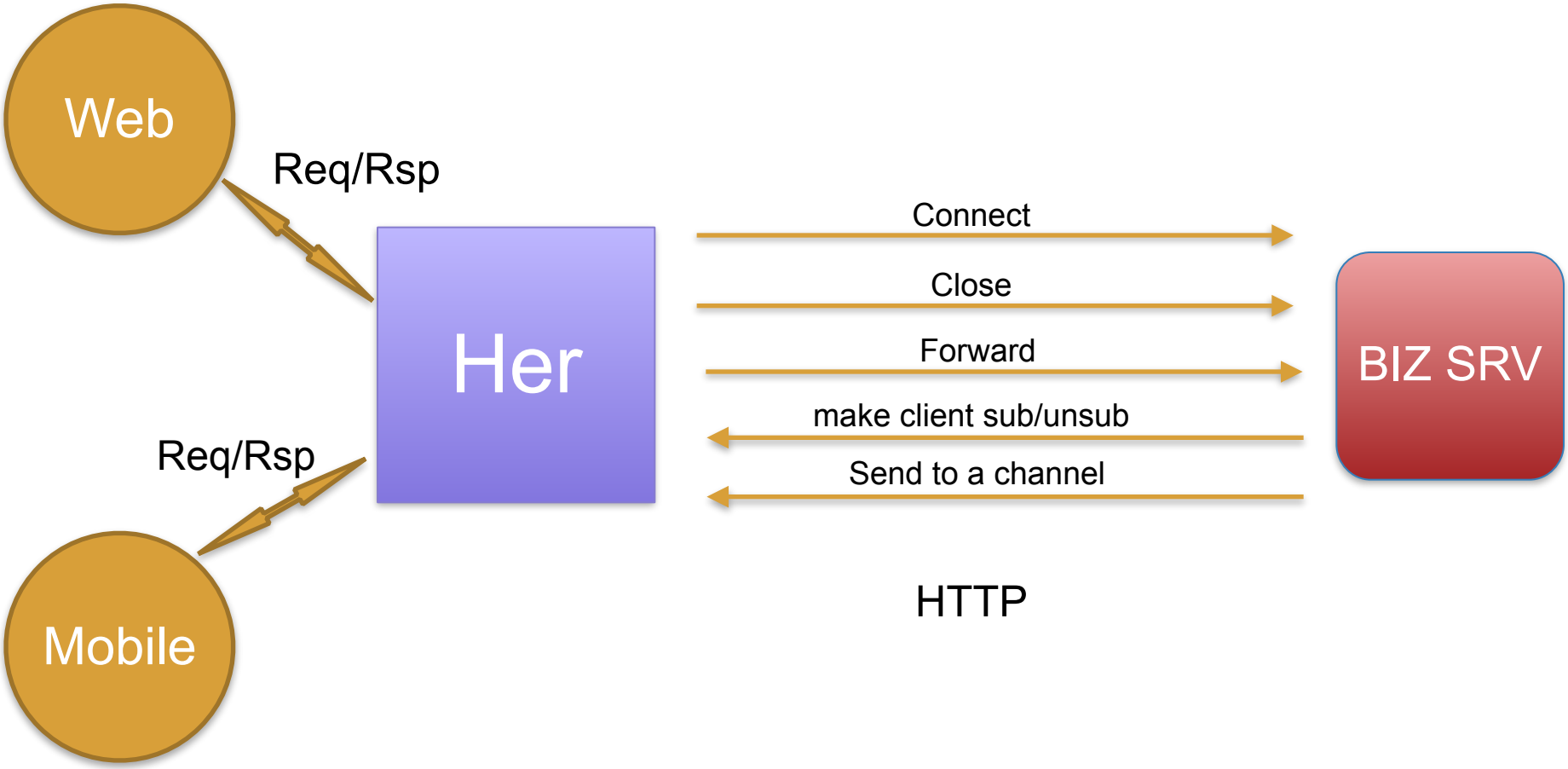
编写专有的鉴权系统

Pub/Sub系统所有客户端平等

以致于服务端没有上帝视角

服务端不能控制客户端

心中的她...


Web
Req/Rsp
Her
Req/Rsp
Mobile
Connect
Close
Forward
make client sub/unsub
Send to a channel
BIZ SRV
HTTP

消息的关键概念

消息类型，类比HTTP的URL

消息内容，类比HTTP的参数

基于WebSocket

TCP+，轻

但，完整：心跳，装拆消息包

支持Web接入

成熟、流行

借鉴MQTT的QoS

消息Quality of Service 三个级别

级别 0：最多只发一次，不管是否成功

级别 1：保证至少成功送达一次

级别 2：保证有且只成功送达一次

协议格式

Bit	7	6	5	4	3	2	1	0
Byte1	CMD				DUP	QOS		NA
Byte2	Message_type(Optional)							
Byte3								
Byte4	Message_ID(Optional)							
Byte5	Data(Optional)							

命令类型

PACKET_SEND

PACKET_ACK # for QoS=1

PACKET_REC # for QoS=2

PACKET_REL # for QoS=2

PACKET_COM # for QoS=2

站在Tornado的肩膀上

面向高并发的异步IO Socket开发框架

在FriendFeed，Facebook内久经考验

内置WebSocket实现、定时器等

她叫Rainbow

服务器：

- Coding.net：https://coding.net/u/jeff/p/Rainbow/git

客户端SDK:

- Objective-c:https://coding.net/u/jeff/p/rainbow-objective-c/git
- Java:https://coding.net/u/jeff/p/rainbow-java/git
- Javascript:https://coding.net/u/jeff/p/rainbow-js/git

服务端SDK：

- Django:https://coding.net/u/jeff/p/rainbow-django/git

这样写客户端

```
[[RainbowEngine defaultEngine] sendMessageWithQos:1
                               mesType:1 content:@"test" timeout:15
                               success:^(UInt16 msgType, NSString *content) {
                               // 处理成功的响应
                               }
                               failure:^(NSError *error) {
                               // 处理错误的响应
                               }];
```

Objective-C

这样写客户端

python阻塞型:

```
>>> rsp_type, content = client.send(msg_type=1, {"id": 100}, qos=1)
>>> rsp_type
101
>>> content
{"name": "jeff"}
```

python非阻塞型:

```
>>> def callback(rsp_type, content):
        print rsp_type, content

>>> client.send_async(msg_type=1, {'id': 100}, qos=1, callback=callback)
```

老湿，整一聊天Demo吧

pip install rainbow-server

rainbow.ini

```
[main]
security_token = qwe123
connect_url = http://localhost:8000/rainbow/connect/
close_url = http://localhost:8000/rainbow/close/
forward_url = http://localhost:8000/rainbow/message/{message_type}/
```

rainbow-server -f rainbow.ini

pip install rainbow-django

服务端代码

用户连接上服务器

```
@rainbow_connect
def connect(request):
    return RainbowResponse({'status': 'success'})
```

用户关闭连接或退出

```
@rainbow_close
def close(request):
    return RainbowResponse({'status': 'success'})
```

用户修改名字

```
@rainbow(msg_type=0)
def update_name(rbrequest):
    name = rbrequest.data['message']
    rsp = RainbowResponse({'status': 'ok', 'identity': rbrequest.identity},
                          rbrequest)
    subscribe('room', rbrequest.identity)
    rsp.context['user'] = name

    send('room', 1001, {'name': name, 'identity': rbrequest.identity})
    return rsp
```

用户发送信息

```
@rainbow(msg_type=1)
def send_message(rbrequest):
    msg = rbrequest.data['message']
    send('room', 1002, {'name': rbrequest.context['user'],
                        'message': msg,
                        'identity': rbrequest.identity})
    return RainbowResponse({'status': 'ok'}, rbrequest)
```

主要JS代码

```
var host = DEMO_HOST;
var rainbow = new Rainbow(host);
rainbow
        .on("open", function () {
          ChatMng.__status__ = 1;
          ChatMng.log("Rainbow is opened.");
        })
        .on("message", function (msgType, data) {
          if (1001 == msgType) {
            ChatMng.log(data["name"] + " enter this room, welcome!");
          }
          else if (1002 == msgType) {
            ChatMng.push(data);
          }
        })
        .on("close", function (err) {
          err && ChatMng.log("Rainbow closed with error: " + err.message);
          ChatMng.__status__ = 0;
          ChatMng.identity = ChatMng.nickname = null;
          ChatMng.refreshStatus();
        })
        .on("error", function (err) {
          ChatMng.log("Rainbow occured an erorr: " + err.message);
        });
      this.rainbow = rainbow.open();
      ChatMng.log("Connecting Rainbow service...");
    },
```

服务器集群

零配置，内网自动发现机制

TCP Port：1984 - ~

UDP Port：2014 - ~

网内支持多个独立集群，cluster_name

http://rainbow/serverinfo/

BenchMark（1G内存单核）

连接数	内存	qos=0 cpu%	qos=0 内存%	qos=1 cpu%	qos=1 内存%	qos=2 cpu%	qos=2 内存%
4800	15.4	0	15.4	100	18	100	18
5600	19.4	27	19.4	100	21.5	100	21.5
7200	23.0	36	23.0	100	26.7	100	26.8
8800	28.1	33	28.1	100	32.3	100	32
11200	34.9	60	34.9	100	41.3	100	41.3
13600	43.0	68.0	43	100	50.7	100	50.7
15000	54.4	75	54.4	100	54.4	100	55.7
18000	56.8	86	56.8	100	65	100	66
21536	64.6	100	64.6	100	78	100	78

感谢

同事们: 智丰、升爷、亮、理天、Nick

社区伙伴:老潘、邱文武、婷姐@techparty

微信群：广州技术宅饭醉团伙

人生苦短 python 當歌

讨论