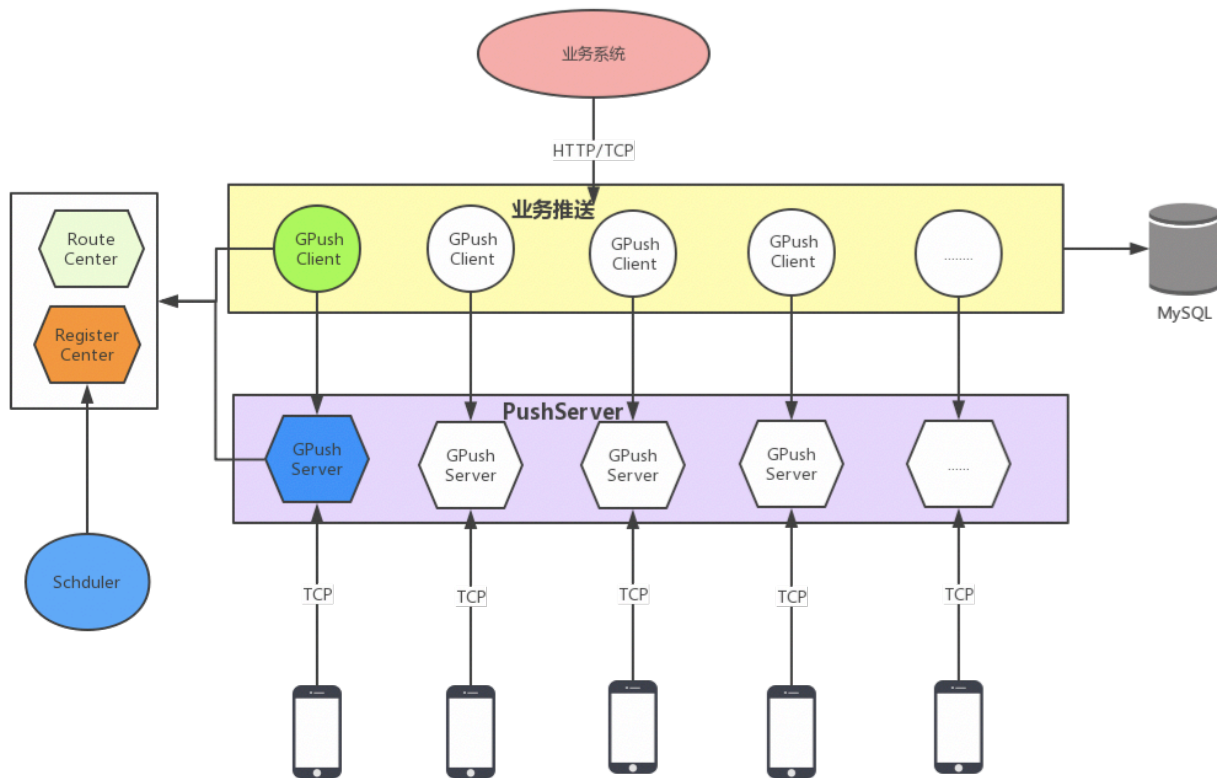


## 咕泡学院 JavaVIP 高级课程教案

# 实时推送平台

|        |                               |
|--------|-------------------------------|
| 主题     | 咕泡学院 Java VIP 高级课程教案-实时推送平台   |
| 主讲     | 大白                            |
| 适用对象   | 咕泡学院 Java 高级 VIP 学员及 VIP 授课老师 |
| 软件版本   | Consolas 字体                   |
| IDE 版本 | IntelliJ IDEA 2017.1.4        |

## 一、整体架构

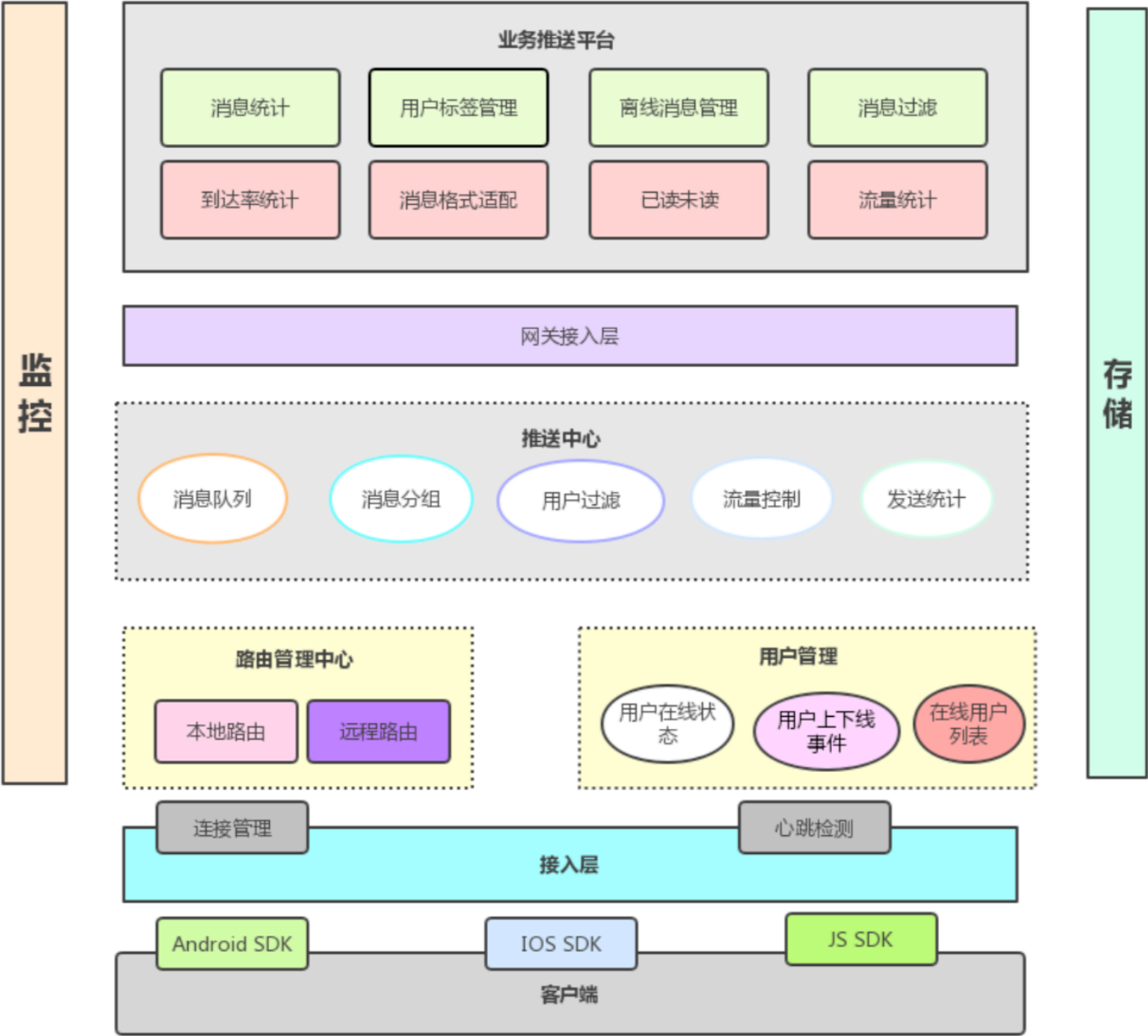


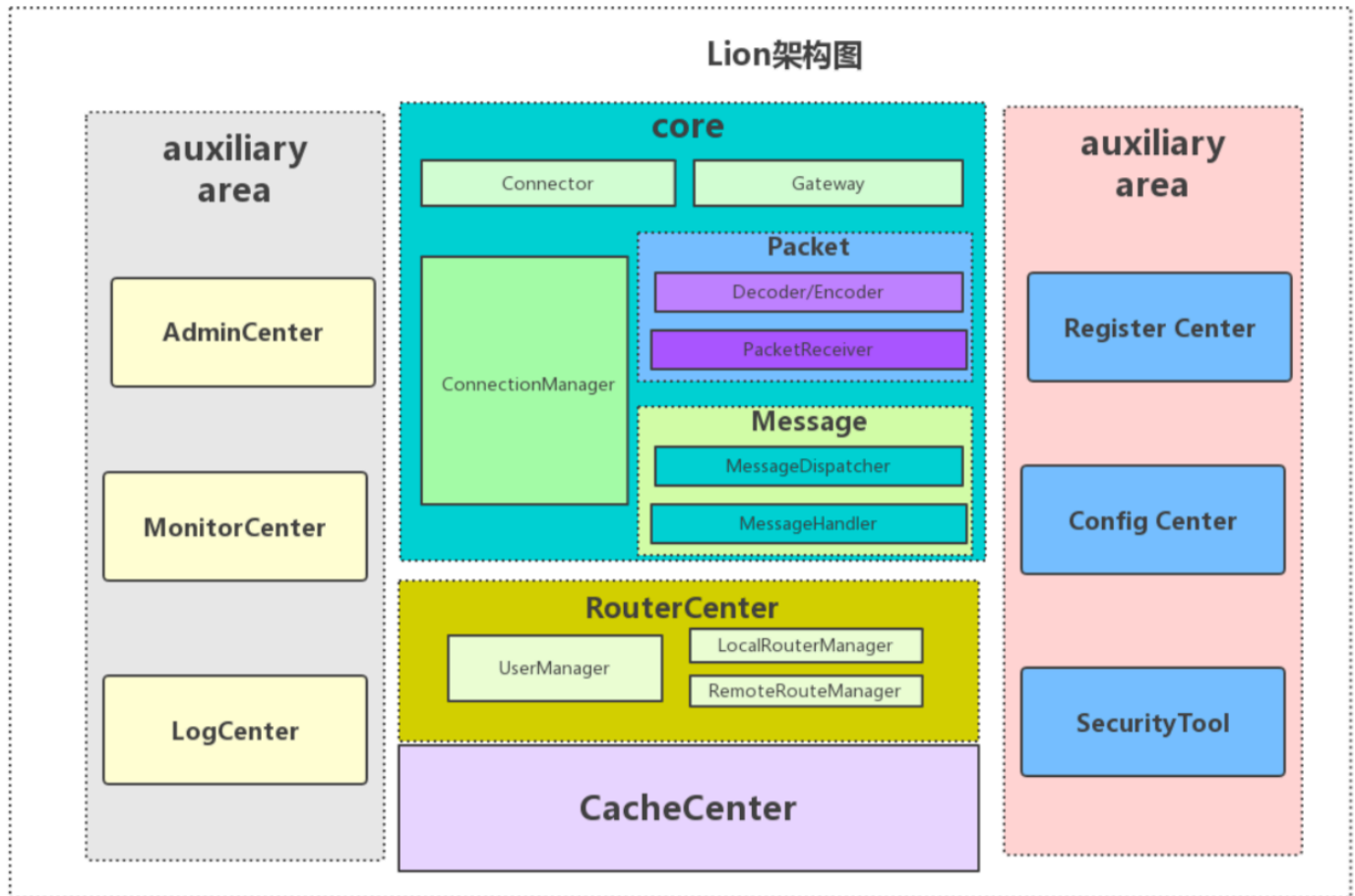
## 解释

1. 业务系统是要发送业务消息的服务，所有要推送的消息直接转给业务推送系统
2. 业务推送系统，负责消息推送，长链接的检查，离线消息存储，用户打标等
3. GPush Client 是我们的客户端已经接入的推送系统
4. 业务推送系统主要是为了隔离业务系统和各种推送系统，用户使用哪个长链接服务，业务系统不需要感知，统一有业务推送系统去选择、去切换

5. Scheduler 负责调度维护 GPush Server 集群，提供查询可用机器列表的接口

## 二、系统架构





## 解释

### 1. 最左侧三大组件分别是日志系统、监控系统、控制台治理服务

- **Log Center** 主要负责业务日志大输出，主要有链接相关日志、推送链路日志、心跳日志、监控日志等
- **Monitor Center** 主要用作系统状态监控，可用于系统调优，包括 jvm 内存，线程，线程池，系统堆栈，垃圾回收情况，内存泄漏情况等。

- AdminCenter 主要用于在控制台对单台机器进行控制与探查，比如查看连接数量，在线用户数，取消本级 ZK 注册，关闭服务等

## 2. 右侧三个分别是注册中心，配置中心和安全工具箱

- 注册中心主要负责注册长链接 ip:port, 网关 ip:port 以及监听各个节点变化，同时增加了缓存
- ConfigCenter 是 Lion Server 配置化的关键，贯穿到各个模块，非常重要
- Sercutity Box 主要实现了 RSA 加密，DES 加密，会话密钥生成及 Session 复用(用于快速重连)

## 3. Core 模块分别是长链接服务，网关服务，Packet 编解码及分发模块，Message 序列化及处理模块

- ConnectServer 用于维持和客户端之间的 TCP 通道，主要负责和客户端交互
- GatewayServer 用于处理 Lion Server 之间的消息交互比如踢人，发送 PUSH
- Packet 主要是协议部分的编解码和包的完整性校验，最大长度校验等
- PacketReceiver 主要负责消息的分发，分发是根据 Command 来的

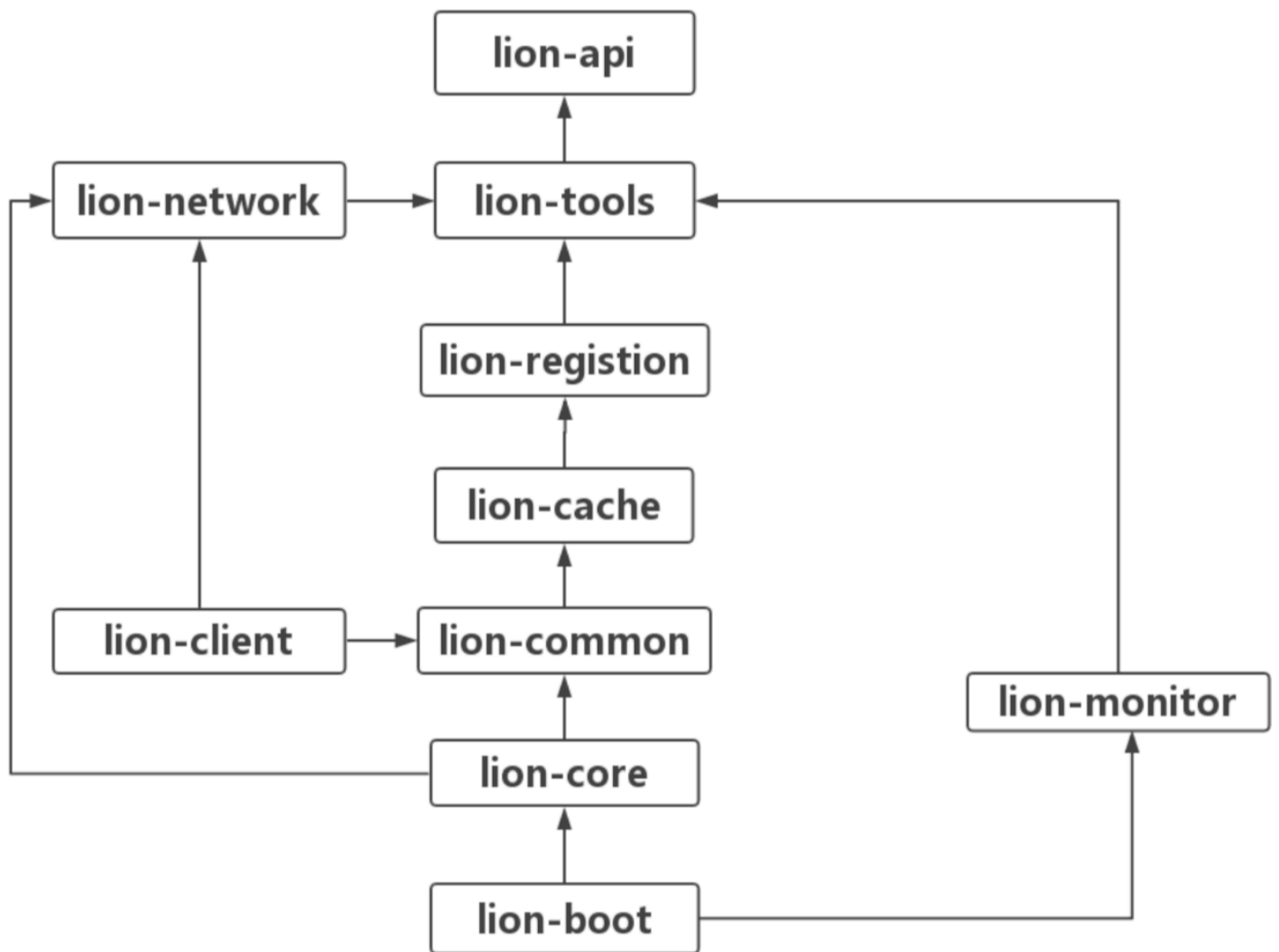
- **Connection/ConnectionManager** 主要负责链接管理，定时检查链接空闲情况，是否读写超时，如果链接断开发出相应的事件给路由中心去处理
- **Message** 部分是整个的业务核心处理了处理消息的序列化，还有压缩、加密等，**MessageHandler** 会根据不同消息独立处理自己所属的业务，主要有：心跳响应、握手及密钥交换、快速重连、绑定/解绑用户、http 代理、消息推送等

#### 4. 路由中心主要包括：本地路由，远程路由，用户在线管理三大块

- **LocalRouterManager** 负责维护用户+设备与链接(connection)之间的关系
- **RemoteRouterManager** 负责维护用户+设备与链接所在机器 IP 之间的关系
- **UserManager** 主要处理用户上下线事件的广播，以及单台机器的在线用户及数量的维护和查询

#### 5. Cache Center 是 Lion 的缓存部分，目前只支持 Redis, 支持双写，主备，hash 等特性

### 三、模块划分



## 模块说明

**Lion-client:** 服务端 SDK, 主要提供发送 Push 的接口给其他业务使用

**Lion-boot:** 是服务端启动入口模块, 主要控制 server 启动、停止流程

**Lion-api:** 定义了 Lion 相关核心接口及协议, 还包括对外暴露的 SPI 接口



**Lion-netty:** 主要提供 netty 相关的一些基础类，像 NettyServer, NettyClient

**Lion-tools:** Lion 用到的一些工具类，包括线程池，加密，配置文件解析等等

**Lion-regsiton:** zookeeper 的 client，包括 path 的定义，节点定义，数据监听等

**Lion-cache:** redis 缓存模块，支持单机模式和 3.x 集群模式，包括用户路由，上下线消息等

**Lion-common:** 定义了 Lion-client 模块和 Lion-core 模块都会用到的类，主要是消息、路由等

**Lion-core:** sever 核心模块，包括接入服务，网关服务，路由中心，推送中心等

**Lion-monitor:** 服务监控模块主要监控 JVM，线程池，JMX，服务状态统计，性能统计等

## 四、协议格式

### 协议说明

- lion 使用的为自定义私有协议，定长 Header + body，其中 header 部分固定 13 个字节。
- 心跳固定为一个字节，值为 -21。

## Header 说明

| 名称        | 类型    | 长度 | 说明                         |
|-----------|-------|----|----------------------------|
| length    | int   | 4  | 表示 body 的长度                |
| cmd       | byte  | 1  | 表示消息协议类型                   |
| checkcode | short | 2  | 是根据 body 生成的一个校验码          |
| flags     | byte  | 1  | 表示当前包启用的特性，比如是否启用加密，是否启用压缩 |
| sessionId | int   | 4  | 消息会话标识用于消息响应               |
| lrc       | byte  | 1  | 纵向冗余校验，用于校验 header         |

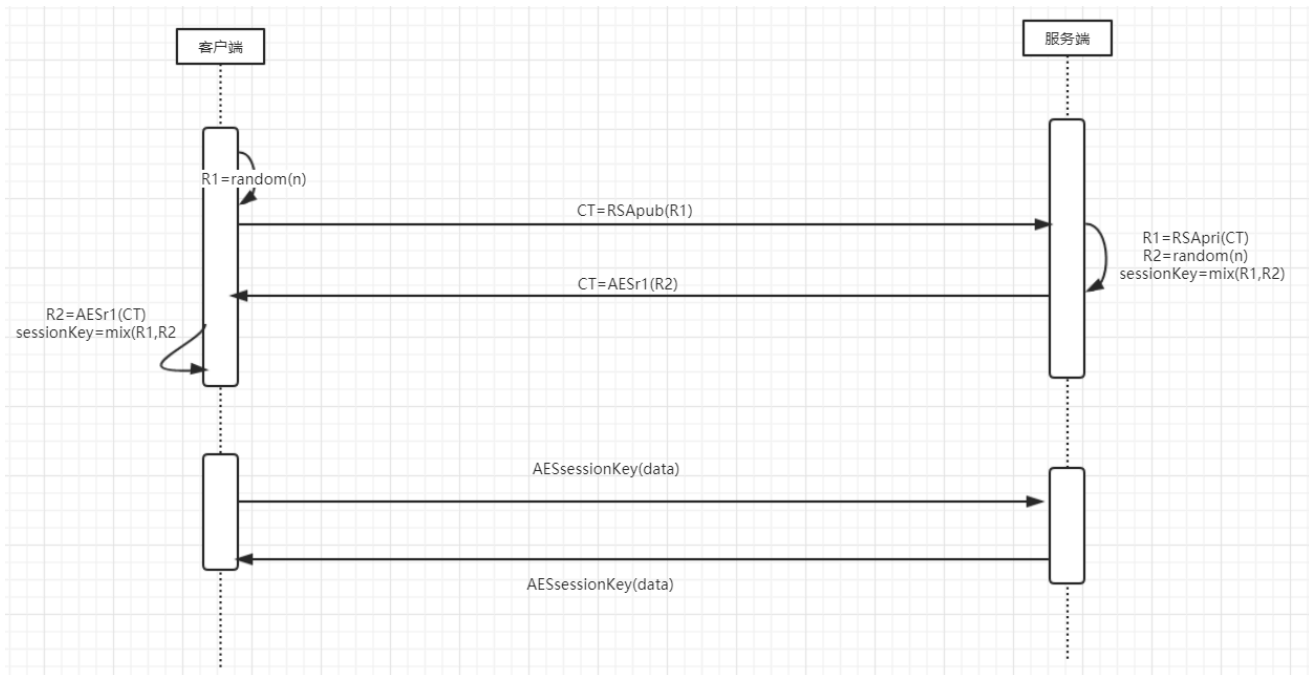
## 五、密钥交换

### 交换方案

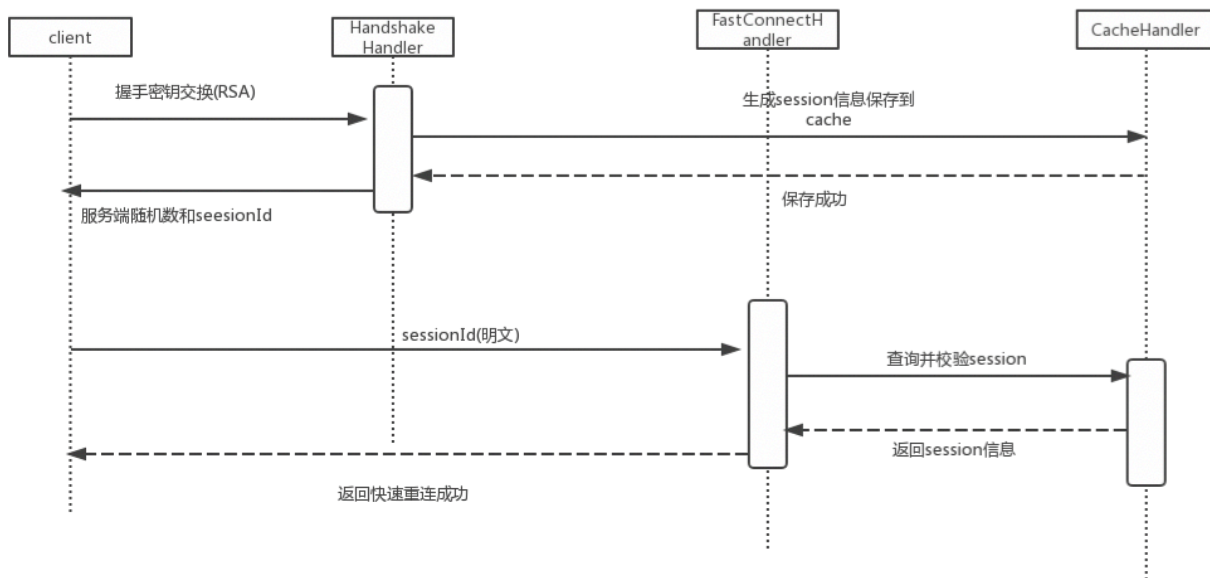
1. 算法上同时使用了非对称加密算法(RSA)和对称加密算法(AES)。
2. 在客户端预埋好由服务端生成好的公钥。
3. 客户端生成随机数 R1，并通过 RSA 公钥加密后发送给服务端。
4. 服务端用 RSA 私钥解密客户端发送的数据取得 R1,同时生成随机数 R2,并以 R1 为 Key 使用 AES 加密 R2 然后把加密后的数据发送到客户端。
5. 客户端以 R1 为 Key 使用 AES 解密服务发送的数据取得 R2

6. 此时客户端和服务的同时都拥有随机数  $R1$ 、 $R2$ ，然后使用相同的混淆算法生成会话密钥(sessionKey)，之后传输的数据都以此值作为 AES 加密的密钥。

## 交换流程



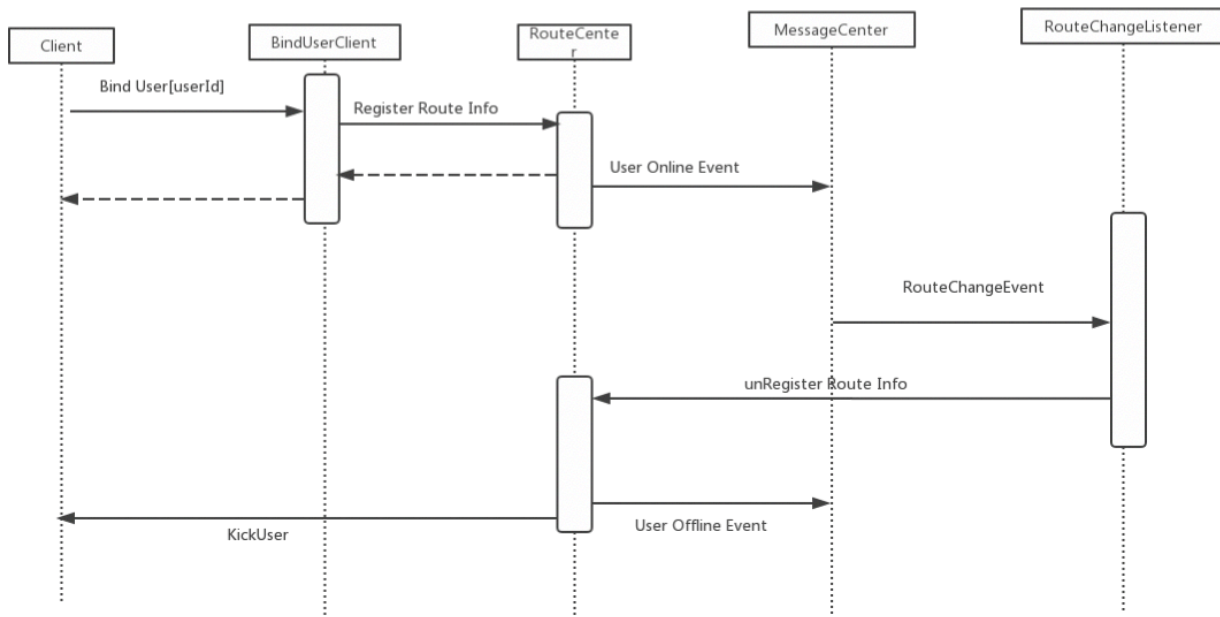
## 六、握手及快速重连



## 解释

1. tcp 连接建立后，第一个消息就是握手或快速重连消息。
2. Handshake 的目的是为了生成会话密钥，同时会把生成好的密钥存储到 **redis**，并把 **key** 返回给客户端，为快速重连使用
3. FastConnect 是基于 Handshake 生成的 **sessionId** 来配合使用的，目的是为了减少 **RSA** 加密的使用次数，特别是网络较差的情况，毕竟 **RSA** 加密相对还是比较耗时的，客户端只需把 **sessionId** 传给服务端，其就能从 **cache** 中取出上次会话信息，恢复到上次握手成功之后的会话状态，这个过程不需要任何加密和密钥交换，相对会比较快速。

## 七、连接绑定用户



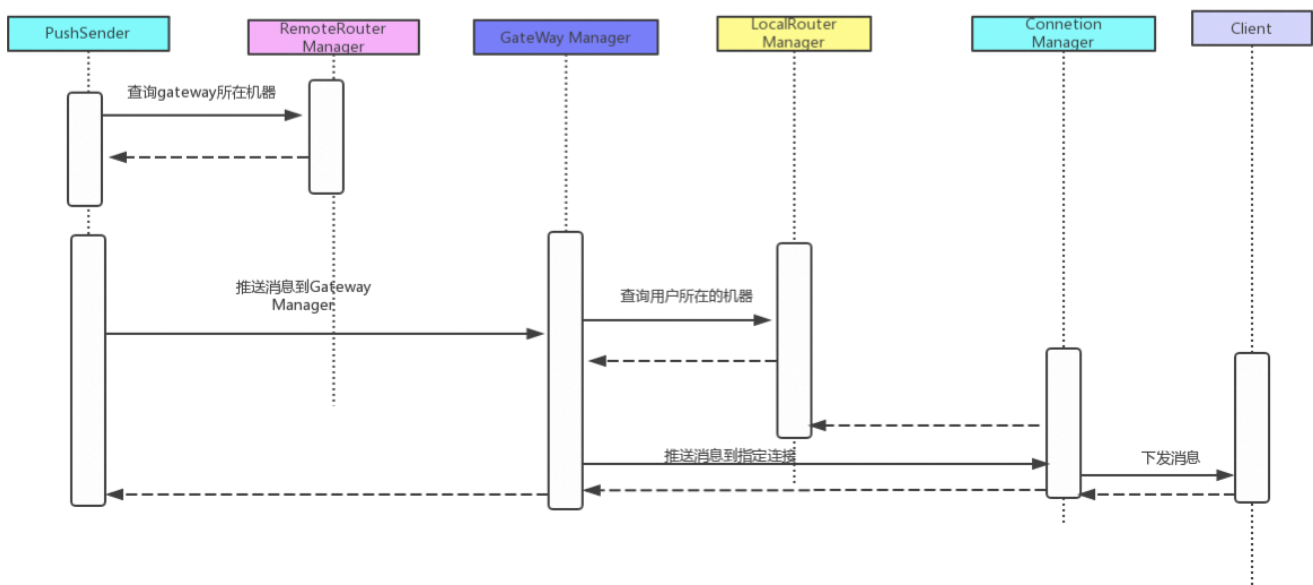
### 解释

1. 握手成功后表示安全通道已经建立，但同时还要给连接设置用户甚至标签，只有这样业务才能更好的去识别用户（没有用户的业务是另外一回事）。
2. 设置用户非常简单，只需把其存储到 **session** 即可，但因为要支持集群的，就必须把用户的位置信息（或者叫路由信息）发布出去，让集群里的其他机器能够通过 **userId** 来查找用户的位置（在哪台机器），因为客户端的 **TCP** 连接在哪台机器，那么和这个客户端的所有数据传输都必须经过这台机器的这个连接！
3. 路由中心有两部分组成：本地路由和远程路由，本地路由数据结构为 **userId->connection** 的 **map**，数据存储在本地内存；远程路由数据结构为

userId->connServer 机器 ip, 数据存储在 cache 一个用户发信息必须先查远程路由, 找到用户在哪台机器, 然后把消息发给这台机器, 让其去本地路由查找 connection 并通过查找到的 TCP 连接把消息发给用户。

4. MessageCenter 之前使用的 redis 提供的 pub/sub 实现的, 也可以自己搭建 MQ 来实现, 最新版踢人已经不再使用 pub/sub, 而是直接使用 udp 网关实现。
5. 踢人: 之所以会有踢人的情况是根据业务需要来的, 有些业务系统是不允许同一个用户在多个设备同时在线的, 或者只允许不同类型的终端同时在线, 比如 QQ, 手机和 PC 可以同时在线, 但同一个帐号在两台 PC 登录时其中一个肯定会被踢下线, push 的踢人要表达的也是同一个意思。

## 八、消息推送流程



## 解释

1. **PushSender** 是消息推送的入口，它的实现在 **push-client** 模块属于服务端 SDK，主要包含有 **GatewayClient**，**RemoteRouterManager**；  
**RemoteRouterManager** 用于定位用户在哪台机器，有没有在线等，而 **GatewayClient** 用于把要发送的业务消息发给用户 TCP 连接所在的机器。
2. **GatewayServer** 负责接收 **GatewayClient** 发送过来的消息，然后到 **LocalRouterManager** 查找用户的 **Connection**，然后把消息交由其下发。
3. **ConnectionServer** 负责维持所有连接到当前机器的客户端连接，所以消息最终还是由其下发（图比较简单，但能表达核心流程）。

## 九、广播推送与流控

按推送用户范围来划分，目前支持三种方式的推送：

1. 单用户推送，推送消息给指定的某个用户。
2. 批量推送，业务自己圈定一批用户，推送同一条消息给圈定的用户。
3. 全网推送，推送消息给所有的在线用户。这里所说的广播推送指的就是第三种用户范围的推送。

为什么广播推送要控制流量？

因为要推送消息给全网在线用户，用户量可能非常大，为了防止瞬时流量过大，所有加了入了防过载保护：流量控制。

流量控制的使用

- 单任务流量控制
- 全局流量控制

## 十、LoadBlance 服务

### LoadBlance 的作用

- Scheduler 是针对 client 提供的一个轻量级的负载均衡服务
- 每次客户端在链接 Lion server 之前都要调用下该服务
- 以获取可用的 Lion server 列表,然后按顺序去尝试建立 TCP 链接,直到链接建立成功

### 对外提供的接口定义

接口类型 : HTTP

Method : GET

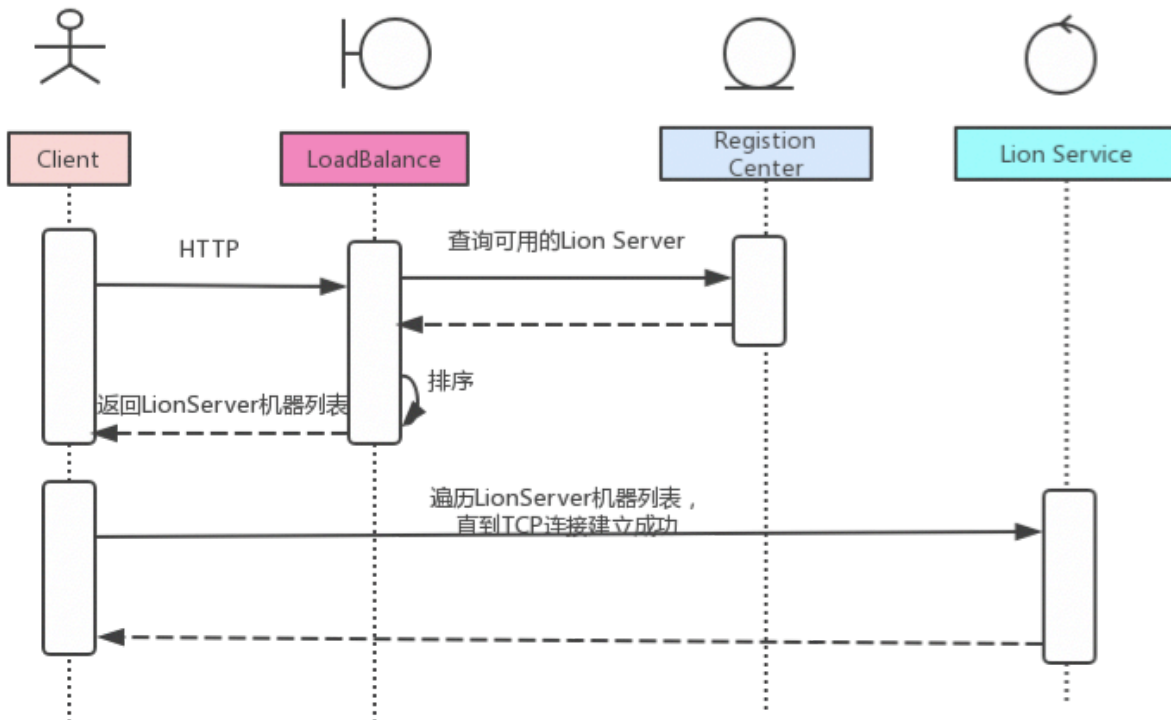
参数 : 无

返回值格式 : ip:port,ip:port

content-type : text/plain;charset=utf-8



## 客户端建立连接流程



## 解释

1. 服务部署可以集成 Tomcat 或自己实现一个 HttpServer 比如基于 Netty 实现
2. Lion server 集群列表可以从注册中心查询，目前提供的有 ZK 查询客户端
3. 如果要想实现负载均衡可以考虑使用以下几种方式实现：

随机，每次从 Lion server 列表随机选取一个地址返回给客户端

轮播，每次把 Lion server 列表依次返回给客户端

按链接数量排序，链接数少的排最前面