**接入层介绍**

**1.接入层是什么？**

通常把跟客户端直连的服务器称为接入服务器，一个或多个接入服务器构成的接入层。

域名： [iot1.midea.com.cn](http://iot1.midea.com.cn/)

端口： 28870 、 28443

**1.1 接入层职责：**

1. 管理客户端的海量长连接
2. 建立与客户端通信的加密通道
3. 数据合法性、正确性校验 （接口参数合法、完整性，判断用户是否已登录）
4. 整合内部少量的长连接
5. Session管理（维护客户端的在线状态）
6. 实施初步攻防
7. 请求转发到逻辑层
8. 无状态设计、限流、高可用性保证、动态负载均衡。

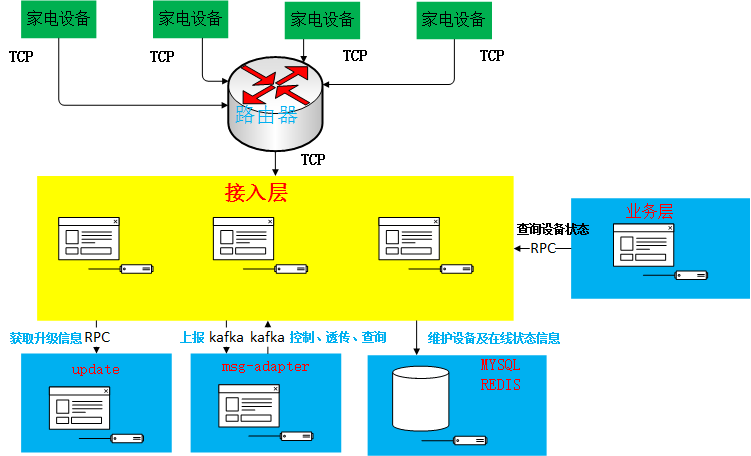
**1.2 接入层为什么不和业务层合并？**

功能越简单，改动越少的服务器，越容易做得稳定。接入服务器是直接跟设备连接的，它直接影响用户体验。接入服务器故障或者是重启肯定会影响到用户，而其他业务模块故障或者重启则不一定会影响用户体验。从这一点上说，应该尽量避免其他模块对接入服务器的不良影响。

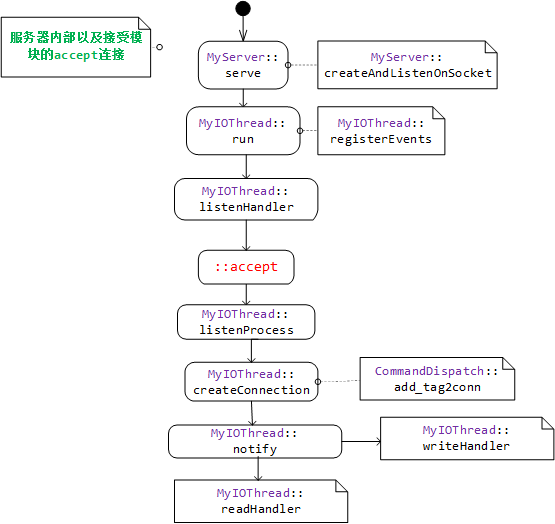
接入层独立出来有下好处:

1. 降低接入层与业务耦合度，减少低稳定度模块对高稳定度模块的不良影响。
2. 使业务层专注于业务处理，降低业务层设计的复杂度。
3. 接入层专注于消息转发，可以有效降低消息的丢失率，从而提高系统的稳定性。
4. 接入层以较小的代价大幅提高客户端接入体验。

**2.接入层系统框架**

****

**图1接入层系统逻辑图**

****

**图2****接受连接建立过程图**

**3.接入层技术难点**

**3.1高可用**

高可用保障

宕机部分机器没影响

模块和数据分离

接入层模块无状态

动态线性伸缩

冗余

降级、限流、切流量、可回滚

**1.避免单点故障**

a) 负载均衡技术

b) 热备（linux HA）

c) 多机房（同城灾备、异地灾备）

**2.应用的高可用性**

a) 故障监控（系统监控（cpu、内存）/链路监控/日志监控） 自动预警

b) 应用的容错设计、（服务降级、限流）自我保护能力

c) 数据量（数据分片、读写分离）

**3.分布式架构下的可伸缩设计**

a) 垂直伸缩 ：提升硬件能力

b) 水平伸缩 ：增加服务器

**4.灰度发布**

**3.2高并发、高性能、低延迟**

无状态、拆分、服务化、消息队列、数据易购、缓存银弹、并发化

要想高性能，需要尽量：避开网络开销（IO），避开海量数据，避开资源争夺。

a)减少并发锁操作

b)耗性能计算并发（性能瓶颈）

c)接口合理设计

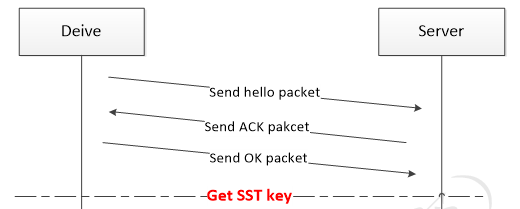
d) 数据库层面，常见的手段有：合理选择数据库的引擎，比如Mysql的InnoDB与MyISAM引擎；进行配置优化；可以考虑使用存储过程来处理复杂的数据逻辑；数据库集群，进行读写分离；合理设计数据库的表结构、索引等；分库、分表，降低单库、单表的数据量。

**4.接入层逻辑**

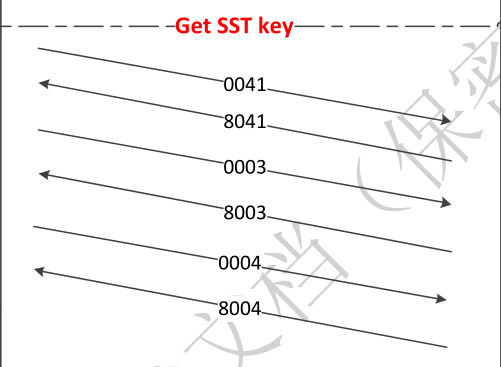
**4.1登录**

**1.安全模块获取sst key。 （非安全模块没有此步）**

安全模块，在获取登录随机串之前，需要获取sst key，交互如下：



**2.登录获取登录串**



再进行登录交互,get login key，交互如下：   （非安全模块的登录从这里开始）

家电申请ID：        0x0041/0x8041

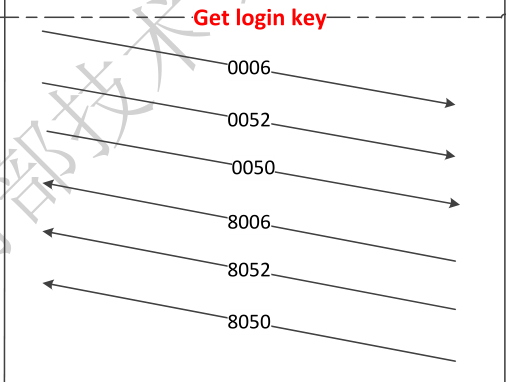
申请登录随机串：     0x0003/0x8003

家电登录认证：      0x0004/0x8004

1.0041设备ID申请，查询是否已经申请，若有了，则返回设备ID，否则生成新的设备ID。

2.0004登录认证，判断设备当前是否登录了，若登录了，则发起tickout给相应的接入层（10秒延迟）将这个旧的会话关闭，同时生成新的会话，保存在redis中。（redis中，同一个会话只保存一份会话信息）

**3.登录获取登录key**



**4.登出处理**

日志地址：<http://100.98.27.208:666/app/kibana#/dev_tools/console?_g=(refreshInterval:(pause:!t,value:0),time:(from:now-14m,to:now))>

日志解析： <http://confluence.msmart.com/pages/viewpage.action?pageId=21505576>

登出原因：

  "key" : "cli\_close",            模块主动断开连接

  "doc\_count" : 10926

  "key" : "tickout",             模块再次上线（上次连接未断开，可能的原因：1.网络原因断开；2.设备sn重复）

  "doc\_count" : 9082

  "key" : "timeout",             模块心跳包超时

  "doc\_count" : 2799

  "key" : "read\_errno\_104",          收到socket读错误码104（RST断链包）

  "doc\_count" : 983

  "key" : "baddata\_0x0040\_sign\_check\_error",                    0040指令签名校验失败

  "doc\_count" : 61

  "key" : "baddata\_0x0006\_sign\_check\_error",                    0006指令签名校验失败

  "doc\_count" : 36

  "key" : "baddata\_sst\_decode\_error",                          st数据包解析出错

  "doc\_count" : 16

  "key" : "baddata\_0x0003\_devicied\_already\_setted",                 0003指令设备ID重复设置

  "doc\_count" : 15

  "key" : "baddata\_0x0050\_sign\_check\_error",                    0050指令签名校验失败

  "doc\_count" : 12

  "key" : "baddata\_parse\_error",                            数据解析出错

  "doc\_count" : 9

**4.2 sn\_code服务提供查询模块在线状态和注册设备id**

接入层提供 ip , port: 15470，供业务层以RPC的方式调用查询设备的在线状态。

接口:  login.querystatus

参数:  deviceid

结果:  登录状态，登录时间，客户端ip

接口2:  login.querystatus\_sn

参数:  sn

结果:  登录状态，登录时间，客户端ip

接口3:  login.status

参数:  deviceid

结果:  登录状态，登录时间，客户端ip

接口4： login.getdeviceid

参数：   sn

结果：   deviceid

**4.3 模块上报指令**

0040，模块主动上报。

0044，模块上报（0020指令后），需要答复。

接入层收到该类指令后，通过kafka发送给msg\_adapter处理，接入层回复0044给模块。

**4.4 模块心跳维持长连接指令**

模块每隔40秒发送一次心跳包。（新版本，收到云端的其他数据包时，推迟发送心跳包）。

云端在140秒内未收到模块的任意有效数据包，会断开与模块的连接，离线原因输出为:timeout。

云端收到模块的心跳包，会更新redis缓存中会话的过期时间，目前更新的过期时间为：

    session\_expire=3600

**4.5 模块升级指令**

模块升级信息查询0x007F/0x807F

模块升级文件下载 0x0080/0x8080

大容量模块OTA查询0x00C1/0x80C1

接入层收到模块的升级请求后，将请求通过RPC发送给update服务，同步阻塞等待2秒钟，若没有收到回复，则超时，回复模块无升级信息。

（接入层重启，对update服务造成较大的压力）

Client第一次连上路由，15分钟检测是否有升级包，有则进行升级，升级成功模块重启，升级失败模块不重启；之后每24小时检测是否有升级包，有则进行升级，升级成功模块重启，升级失败模块不重启。

**4.6 业务层下发的0020指令(透传、控制、查询)**

接入层收到来自kafka的0020指令，判断模块是否在线，在线则发送给模块，否则丢弃消息。

待收到来自模块的8020指令，通过kafka发送上上层业务。

**5接入层业务日志解读**

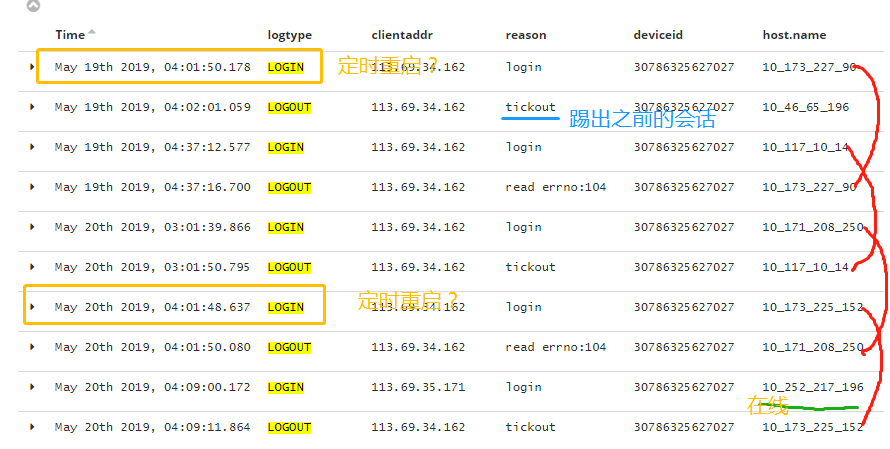
日志事件说明：

<http://confluence.msmart.com/pages/viewpage.action?pageId=21505576>

日志查询：

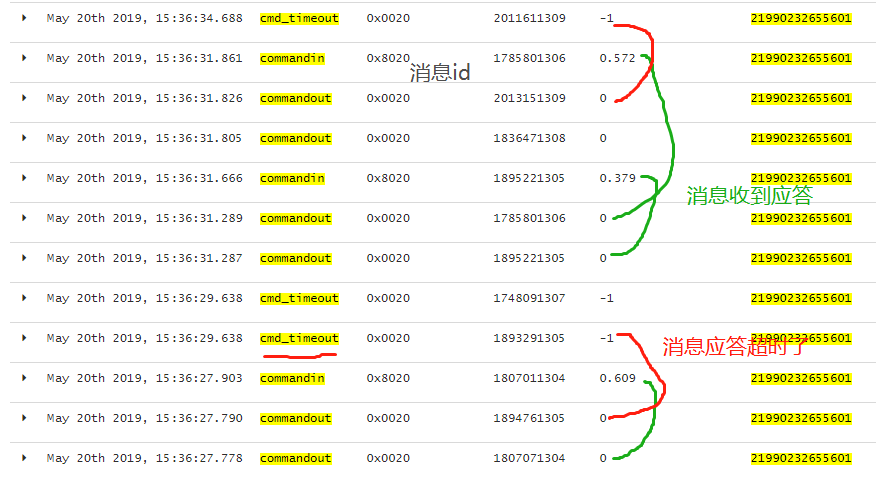
[http://100.98.27.208:666/app/kibana#/discover?\_g=(refreshInterval:(pause:!t,value:0),time:(from:now-14m,to:now))&\_a=(columns:!(msgtype,msgtag,sn,deviceid,expandtime),index:'0b446ae0-e12a-11e9-a186-cf08cb3e53ae',interval:auto,query:(language:kuery,query:'\*'),sort:!('@timestamp',desc))](http://100.98.27.208:666/app/kibana#/discover?_g=(refreshInterval:(pause:!t,value:0),time:(from:now-14m,to:now))&_a=(columns:!(msgtype,msgtag,sn,deviceid,expandtime),index:'0b446ae0-e12a-11e9-a186-cf08cb3e53ae',interval:auto,query:(language:kuery,query:'*'),sort:!('@timestamp',desc)))

**5.1接入层上下线案例分析**



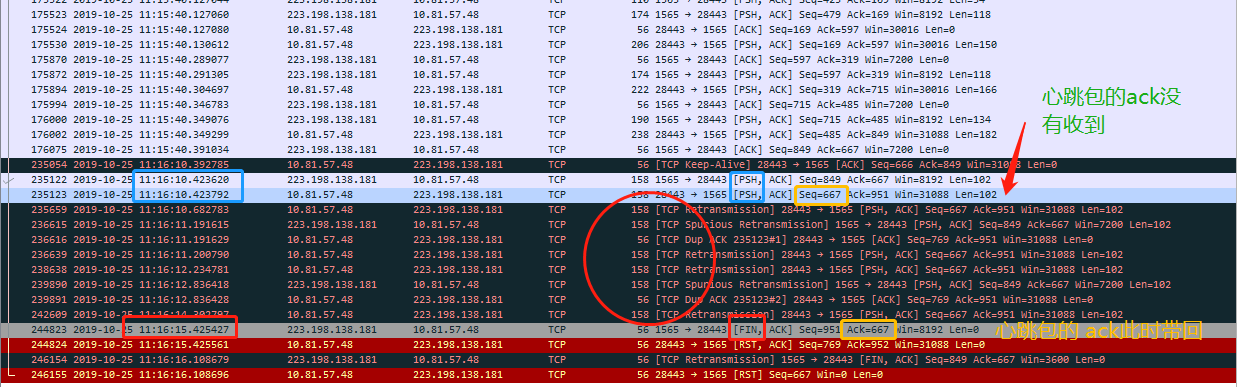
  冰箱每天凌晨4点重启，云端无法感知重启，故下线原因为tickout。

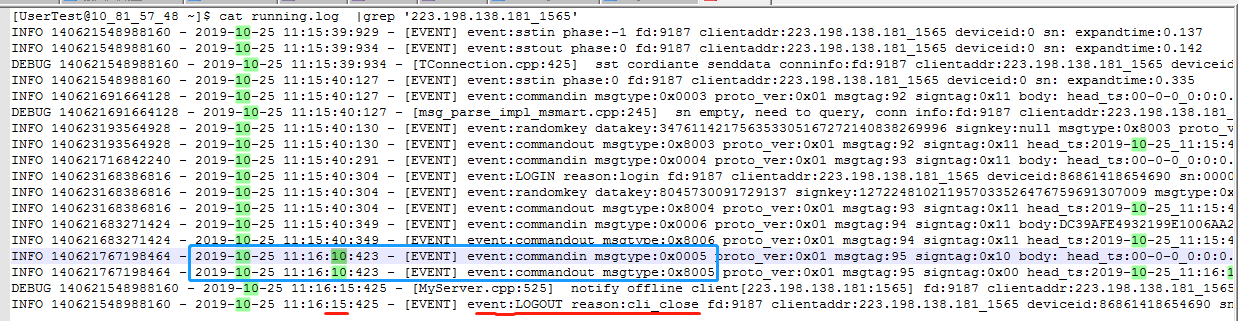
**5.2接入层指令收发案例分析**



5.3模块主动断线案例抓包分享

示例一：心跳包的服务端回包，模块未收到，5秒钟后网络恢复，模块的fin包到达，断链，原因为cli\_clsoe

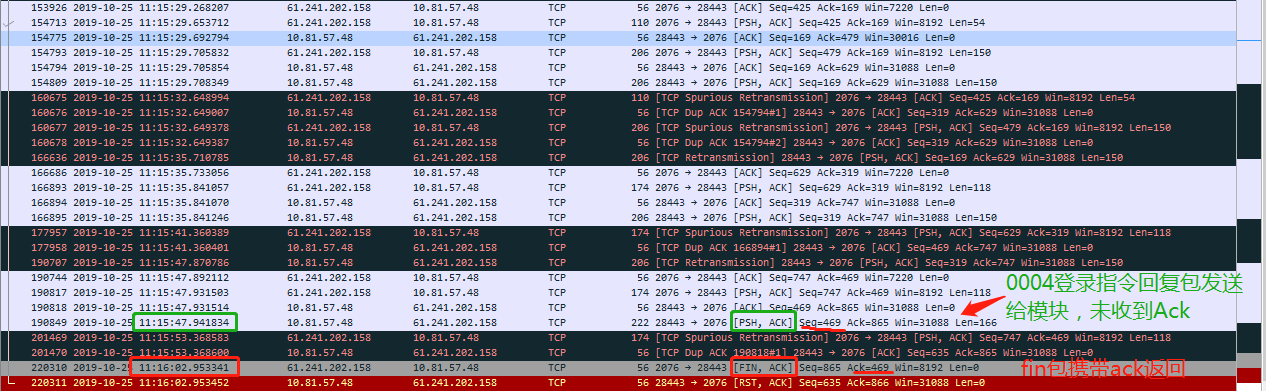




这说明，心跳包发送时网络异常，网络是在5秒钟后恢复了

然而，msmart的心跳机制，模块在发送心跳包后，等待5秒后发起断链机制，是否可以优化？

示例二：0004的服务端回包，模块未收到，6秒钟后模块重发，再9秒后模块的fin包到达（此fin包携带ack，说明模块收到了之前的0004回复包），断链，原因为cli\_clsoe



**6.开发者平台、开放平台**

开发者平台：

<http://developer.msmartlife.com/#/product/myProducts/hardwareProducts/products>

开放平台：

<http://openadmin.smartmidea.net/#/product/dataQuery/queryEquipmentInfo>

账号申请及权限开通找： **梁洋波**