ZEBRA

SDK设计说明书

修订历史

|  |  |  |  |
| --- | --- | --- | --- |
| 版 本 | 日 期 | 变更人 | 描述 |
| 1.0 | 2019-03-08 | 梁森明 | 初稿 |
| 1.1 | 2019-.7-25 | 梁森明 | 1）增加签名接口；2）增加函数原型和修改函数说明 |
|  |  |  |  |
|  |  |  |  |

目录

[1. 系统架构 4](#_Toc2951415)

[1.1. 系统架构图 4](#_Toc2951416)

[1.2. 节点构成和功能说明 4](#_Toc2951417)

[2. 参数说明 5](#_Toc2951418)

[2.1 启动参数说明 5](#_Toc2951419)

[2.2 通道类型参数 5](#_Toc2951420)

[2.3 事件类型参数 6](#_Toc2951421)

[3. 接口说明 6](#_Toc2951422)

[3.1底层接口 6](#_Toc2951423)

[3.1.1启动 start 7](#_Toc2951424)

[3.1.2停止 stop 7](#_Toc2951425)

[3.1.3创建通道createChannel 7](#_Toc2951426)

[3.1.4通道消除destroyChannel 7](#_Toc2951427)

[3.1.5加入通道joinChannel 8](#_Toc2951428)

[3.1.6监听通道 8](#_Toc2951429)

[3.1.7发送数据send\_message 8](#_Toc2951430)

[3.1.8发送事件postEvent 9](#_Toc2951431)

[3.2虚函数接口 9](#_Toc2951432)

[3.2.1获取节点ID 9](#_Toc2951433)

[3.2.2获取节点公私钥 9](#_Toc2951434)

[3.2.3获取节点类型 10](#_Toc2951435)

[3.2.4获取启动参数 10](#_Toc2951436)

[3.2.5获取数据内存最大值 10](#_Toc2951437)

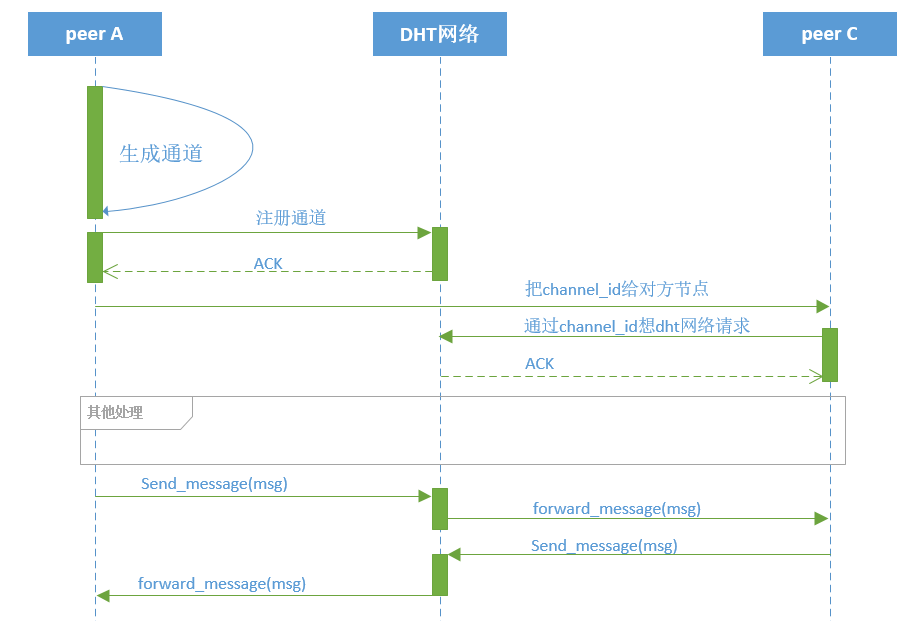
[3.2.6获取线程数量（client） 11](#_Toc2951438)

[3.2.7获取线程数量（server） 11](#_Toc2951439)

[3.2.8回调事件 11](#_Toc2951440)

# 系统架构

## 系统架构图



* Peer A生成通道；
* Peer A把通道注册到到dht网络中（同时应答）；
* 把channel id给到peer C（线上和线下都可以）；
* Peer C根据channel id向dht网络请求（同时应答）；
* 其他流程处理；
* Peer A和Peer C实现通过dht网络相互收发数据。

## 节点构成和功能说明

* 每个 zebra node 逻辑上区分两种类型节点
  1. DHT 节点：负责数据路由、读写路由表；
  2. peer 节点：peer 节点负责信息的收发；
* 两类节点运行同一套底层代码，通过执行程序的启动参数read only 属性来区分不同节点；
  1. read only=true，为peer节点
  2. read only=false，为DHT节点

# 参数说明

## 2.1 启动参数说明

启动参数是指在执行程序启动时需要输入的参数，这些参数可以设置系统一些属性，可以通过—help命令查看，如./zebra –help。

Zebra提供了一下启动参数：

|  |
| --- |
| --readOnly arg (=0) if true, only sends request and not handle request  from other nodes  --listenPort arg (=37053) listen port  --serviceThreads arg (=2) threads of serivce  --clientThreads arg (=1) threads of client  --maxMemory arg (=500) maximum usage in memory, MBytes  --bootstraps arg bootstrap nodes. eg: 172.16.0.1:37053  172.16.0.2:37053 |

* readOnly：设置zebra节点类型；=true：peer节点，=false：dht节点；
* listenPort：系统监听端口；
* serviceThreads：dht节点使用的线程数（默认2）；
* clientThreads：peer节点使用的线程数（默认1）；
* maxMemory：系统数据分配内存大小；
* bootstraps：dht节点的ip和端口。

## 通道类型参数

系统包括两种通道类型，分别如下：

* join：对应的值是0，该类型通道属于临时通道，用于辅助两个节点间建立通道关系，用户在创建通道时，create命令输入的通道类型参数便是该参数。
* chat：对应的值是1，该类型通道属于通信通道，由底层根据用户请求来创建。

## 2.3 事件类型参数

由于系统的操作都是通过异步的方式来实现，因此用户的操作不是根据返回值来判断，而是通过异步事件的方式来通知用户。以下是系统中提供给用户层的事件：

* BOOTSTRAP：引导完成通知事件，即系统启动完成之后会通过该事件通知用户，该事件会包括node id，用户收到该事件后首先判断本地是否已经保存了该node id，如果没有则把node id保存到本地。
* GETDHTNODES：路由表打印事件
* CREATECHANNEL：创建通道通知事件，当底层创建通道会通过该事件通知用户，事件内容包括channel id。
* CONNECTION：连接事件，即本地节点连接dht网络节点时产生的事件，告知用户是否连接成功。
* JOINCHANNEL：join请求事件，当有节点请求加入到已创建的通道时会产生该事件，用户接受到该事件后可以选择“同意”或“不同意”两种方式，同意代表接受请求节点加入自己的通道，不同意代表不允许请求节点加入自己的通道。
* ALLOWCHANNEL：同意join事件，该事件是由请求端收到。代表对方同意请求节点加入到通道中。
* DISALLOWCHANNEL：不同意join事件，该事件是由请求端收到。代表对方不同意加入到通道中。
* CHAT：消息通知事件。当系统中收到chat消息时会通过该事件通知用户，即正常通信的消息由该接口来接收数据。
* REPLY：用户请求应答事件，即用户调用接口运行结果通知，告诉用户请求的错误码。
* NOTIFY：其他通知事件。

# 接口说明

zebra是一套跨平台通信协议，可以在window、linux、arm平台上基zebra进行应用开发，zebra为应用提供统一的API。

## 3.1通用接口

通用接口是提供给用户端使用并实现了某个应用功能的接口。用户可以根据这些接口来实现应用开发。

### 3.1.1启动系统

函数原型：

void start()

函数说明：

系统启动接口，系统开始前首先需要调用该接口来初始化相关参数和启动相关功能模块。

### 3.1.2停止系统

函数原型：

void stop()

函数说明

系统停止接口，系统在停止时调用该接口停止相关服务。

### 3.1.3创建通道createChannel

函数原型：

int createChannel(int type, long expire, const std::string& name, std::string& channel\_id)

函数说明：

使用该函数创建通信通道。如果创建成功会通道回调事件的方式通知用户并在事件中返回channel id。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| type | int | 创建通道的类型（详见2.2） |
| expire | Long | 通道有效时间0代表长期有效 |
| name | String | 通道名称 |
| channel\_id | String | 通道返回channel id值 |

### 3.1.4通道消除destroyChannel

函数原型：

int destroyChannel(const std::string& channel\_id)

函数说明：

删除不需要的通道。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| channel\_id | string | 通道id |

### 3.1.5加入通道joinChannel

函数原型：

int joinChannel(const std::string& channel\_id, const std::string& description)

函数说明：

通过该函数把节点加入到已有的通道中，由请求加入方调用，同时需要通道创建方同意才能加入。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| channel\_id | string | 通道id，由通道创建方提供 |
| description | string | 请求描述说明 |

### 3.1.6监听通道

函数原型：

int listenChannel(const std::string& channel\_id)

函数说明：

监听已有通道是否正常，如果发现通道异常，如通信异常，系统会自动修复异常问题，保证节点间能正常通信。一般在通道创建或通道重连时使用。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| channel\_id | string | 通道id |

### 3.1.7发送数据send\_message

函数原型：

int send\_message(const std::string& channel\_id, const std::string& data)

函数说明：

用户发送数据接口，用户使用本地保存的不同通道id值，把数据发送到不同节点上。用户可以根据业务需求定义不同到数据协议格式，最后封装成string类型由该函数发出去。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| channel\_id | string | 通道id |
| data | string | 用户数据 |

### 3.1.8发送事件postEvent

函数原型：

int postEvent(peersafe::imapi::eventCode event)

函数说明：

该函数属于回调函数，由系统底层调用，用户根据回调函数获取对应的事情，再根据不同的事件做不同的处理。3.1.1—3.1.7所述接口的结果都是通过该回调函数回调对应的事件给用户。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| event | peersafe::imapi::eventCode | 事件（详见2.2） |

## 3.2回调接口

回调接口由一个纯虚类提供，应用层代码必须继承并实现该接口，用于完成系统底层与用户之间的数据交互。

### 3.2.1获取节点ID

函数原型：virtual void node\_id(std::string& id)

函数说明：

该函数的任务是从本地文件或者数据库获取本地节点的node id，同时把该node id值赋给形参 &id。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| id | string | 返回node id |

### 3.2.2获取节点公私钥

函数原型：virtual void key\_pair(std::string& public\_key, std::string& private\_key)

函数说明：

从本地数据库获取本地节点的公私钥对，获取过程中需要传入本地节点node id，如果程序是初次使用，还没有对应的公私钥对，那么底层会自动生成并回调给用户端保存。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| public\_key | string | 节点公钥 |
| private\_key | string | 节点私钥 |

### 3.2.3获取节点类型

函数原型：virtual bool readOnly()

函数说明：

该函数返回命令行输入的readOnly属性值。

参数：无

### 3.2.4获取引导节点地址

函数原型：virtual std::size\_t bootstraps(std::string& nodes)

函数说明：

提供dht节点的ip地址和端口号。该ip地址和端口是在peer节点启动的时候有启动参数提供的。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| id | string | 返回node id  格式eg: 10.100.0.100:37053;10.100.0.101:37053 |

### 3.2.5获取数据内存最大值

函数原型：virtual std::size\_t usingMaxMemory()

函数说明：

返回数据内存最大值。该值来自启动参数。

参数：无

### 3.2.6获取线程数量（client）

函数原型：virtual int client\_threads()

函数说明：

返回client节点设置的线程数量，该值来自启动参数。

参数：无

### 3.2.7获取线程数量（server）

函数原型：virtual int service\_threads()

函数说明：

返回server节点设置的线程数量，该值来自启动参数。

参数：无

### 3.2.8回调事件

函数原型：virtual void onEvent(eventCode code, event\* evn)

函数说明：

系统底层会根据用户的请求或者处理过程回调不同的结果给用户，用户需要实现该回调函数，根据回调事件的不同做出不同的处理，如，对于返回事件BOOTSTRAP，用户需要判断本地是否已经保存了node id，如果没有保存则把返回的node id保存到本地数据库中。

参数：

|  |  |  |
| --- | --- | --- |
| 传入参数 | 类型 | 说明 |
| code | eventCode | 事件码（详见2.3） |
| evn | event\* | 事件内容 |

### 3.2.9数据签名

函数原型：

virtual bool disableSignature() const

函数说明：

该函数作用于单包数据的签名，默认是每包数据都签名，直接返回true代表关闭该功能，同时通信性能会得到提升。

# 4.参考例子

详见clever\_pdu\_app仓库develop-zpdu-v2 分支的src/zebra/example目录。