分布式系统

应用架构设计

编制： 王 科 威

审核：

版本： V1.0.0

2020年5月18日

目录

[修改历史 I](#_Toc41290392)

[一 引言 1](#_Toc41290393)

[1.1 编写目的 1](#_Toc41290394)

[1.2 规范说明 1](#_Toc41290395)

[二 定义 1](#_Toc41290396)

[2.1 协议格式 1](#_Toc41290397)

[2.2 协议定义 1](#_Toc41290398)

# 修改历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **说明** | **作者** | **审核** | **日期** |
| V1.0.0.0 | 初稿 | 王科威 |  | 2020-05-25 |

# 引言

## 编写目的

在视频数据越来越多的情况下，如何做到投入最少的IT资源、提供更高的用户数据和分析能力？分布式系统应用架构即是通过采集这些非结构化数据源，并将它们汇聚到一起存储到公有或私有云上，帮助用户思考能用数据做什么？从中获取有价值的数据及知识，并有可能在未来提供现在还无法想象的云服务。

## 规范说明

### 命名规范

所有名称都使用小写字母表示。

|  |  |  |
| --- | --- | --- |
| ***编号*** | ***名称*** | ***说明*** |
| 1 | xmq | 消息分发服务 |
| 2 | cms | 资源管理服务 |
| 3 | web | WEB后台服务 |
| 4 | das | 动态IPv4地址解析服务 |
| 5 | dhslb | 设备接入负载均衡服务 |
| 6 | dhs | 视频监控设备接入服务 |
| 7 | xmslb | 流媒体负载均衡服务 |
| 8 | xms | 私有流媒体服务 |
| 9 | rtsp | RTSP流媒体服务 |
| 10 | rtmp | RTMP流媒体服务 |
| 11 | hls | HLS流媒体服务 |
| 12 | gbs | GB28181信令服务 |
| 13 | gbxms | GB28181流媒体服务 |
| 14 | dss | 数据同步服务 |
| 15 | dbe | 数据库引擎服务 |
| 16 | ais | AIS航海定位服务 |
| 17 | tss | 时间同步服务 |
| 18 | alm | 报警订阅服务 |
| 19 | helmet | 安全帽检测算法服务 |
| 20 | phone | 打手机检测算法服务 |
| 21 | sleep | 睡觉检测算法服务 |
| 22 | face | 人脸检测算法服务 |
| 23 | fight | 打架检测算法服务 |

### ID标识规范

所有ID标识使用UUID规范。

### 用户数据规范

* 用户数据格式按照Google protocol buffers开源项目的sytex2格式要求进行串行化处理。
* 网络通信协议字段都使用小写字母。

### 端口号规范

|  |  |  |
| --- | --- | --- |
| ***名称*** | ***端口号*** | ***说明*** |
| xmq | TCP:60531 | XMQ监听端口号 |
| web | 80/8080 | WEB服务端口号 |
| das | UDP:60531 | 地址解析端口号 |
| xms | TCP:60820 | 流媒体连接端口号 |
| dhs | TCP:60927 | 视频监控设备组件连接端口号 |
| alm | TCP:50531 | 报警数据推送订阅端口号 |

# 系统架构设计

## 系统架构图



## 系统结构图说明

### 平台应用层

### 平台处理层

### 分布式服务层

#### 消息分发服务（XMQ）

1. 消息分发服务是分布式服务层中所有服务间网络通信的桥梁。

* 该服务不解析任何用户数据，只负责在各个服务间转发消息。
* 该服务作为桥梁，同时面向两侧用户，一个是Worker用户，另一个是Client用户。

1. 消息分发服务在分布式服务层中是*唯一的*。
2. 消息分发服务打开UDP服务协助其他服务确认IPv4地址信息。

#### 资源管理服务（CMS）

资源管理服务是分布式服务层中所有服务的管理工具。分布式服务层中的服务以名称作为标识向资源管理服务注册、心跳和注销，该服务自动监视各服务的在线状态，超过90s没有状态更新的服务作离线处理。资源管理服务支持服务在线状态查询功能。

*资源管理服务在分布式服务层中是唯一的。*

#### WEB后端服务（WEB）

*WEB后端服务同时具备Worker模型和Client模型二者的特性。*

#### 动态IPv4地址解析服务(DAS)

该服务通过监听UDP的指定端口数据以获取发送者的IPv4地址信息，并将获取到IPv4地址信息返回给发送者。

#### 视频监控设备接入负载均衡服务（DLB）

该服务每间隔5s向资源管理服务查询在线的视频监控设备接入服务。在每次设备配置时选取出最合适的处理者。

*视频监控设备接入负载均衡服务在分布式服务层中是唯一的。*

#### 海康视频监控设备接入服务（HKD）

#### 大华视频监控设备接入服务（DHD）

#### 流媒体负载均衡服务（SLB）

该服务每间隔5s向资源管理服务查询在线的流媒体分发服务。在每次流配置时选取出最合适的处理者。

*视频监控设备接入负载均衡服务在分布式服务层中是唯一的。*

#### 私有流媒体分发服务（XMS）

#### RTSP流媒体分发服务（RTSP）

#### RTMP流媒体分发服务（RTMP）

#### HLS流媒体分发服务（HLS）

#### GB28181服务（GB28181）

#### 数据同步服务（DSS）

#### 串口设备接入服务（AIS & MCS）

*串口设备接入服务在分布式服务层中可以多次注册，但名称必须不同。*

* AIS
* MCS

### 分布式数据存储层

### 数据ETL层

## 分布式服务层网络架构图



## 分布式服务层网络架构说明

分布式服务层网络架构由3种不同的角色组成，分别为：Majordomo、Worker和Client。Majordomo使用桥接的思想在逻辑上将Worker和Client分离成面向内部服务和外部的不同服务，并采用不同的数据格式进行交互和转换。Worker之间和Worker与Client之间也可以直接使用发布订阅的方式直接进行数据交互。

### Majordomo模型

首先，Majordomo只负责数据的交换，就如同邮局一样，Majordomo只按照发送ID和接收ID进行数据转发，不解析任何数据内容。

其次，从逻辑结构上来说，Majordomo可以是点对点的互联，也可以是上下级的级联。无论是互联还是级联结构，Majordomo模型都必须具备Worker模式的特性才能实现。

### Worker模型（面向内部）

Worker模型是分布式服务层网络架构中面向内部的服务。与Client模型不同之处在于Worker模型必须实现注册、心跳和注销等业务逻辑。

### Client模型（面向外部）

Client模型是分布式服务层网络架构中面向外部的服务。

### Pub模型

Pub模型即可以向分布式服务层网络结构的内部发送订阅数据，也可以向分布式服务层网络结构的外部发送订阅数据。

### Sub模型

Client模型只能通过Sub模型订阅分布式服务层内部数据。

## 分布式数据处理层架构图

## 分布式数据处理层架构说明

# 分布式服务层通信协议

基于系统中存在的3中不同角色，以Majordomo模型为桥，通信协议分为Client模型使用的面向外部服务协议和Worker模型使用的面向内部服务协议。在特定条件下，Majordomo也具有Worker模型的特性。

## Majordomo模型与Worker模型

### 协议格式

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CommID* | *Placeholder* | *Module* | *From* | *To* | *Router(1)* | *…* | *Router(n)* | *Placeholder* | *Router(1)* | *…* | *Router(n)* | *Command* | *Message* |

### 数据定义

#### 固定头部

* CommID：Worker模型ID标识。
* Placeholder：空。
* Module：模型标识，即“worker”/“client”。
* From：数据发送端名称。
* To：数据接收端名称。

#### 可变路由地址

* 消息路由地址表

#### 可变数据内容

* Placeholder：可变数据起始标识。
* 消息路由地址表。
* Command：命令标识。
* Message：消息数据。

## Client模型与Worker模型

### 协议格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Placeholder*** | ***Module*** | ***From*** | ***Command*** | ***Data*** |

### 数据定义

* Placeholder：空。
* Module：模型标识，只能是“worker”。
* From：数据发送端名称。
* Command：命令字符串。
* Data：数据

# 内部通信时序图

## 面向外部

### Client <--> XMQ

#### 服务查询



Data：”Query”表示查询请求，以”;”分隔的服务名称表示查询应答。

### Client <--> XMQ <--> XMQ

#### 服务查询



### Client <--> XMQ <--> Worker

#### 数据转发



#### 数据订阅



### Client <--> XMQ <--> XMQ <--> Worker

#### 下行转发



## 面向内部

### CMS与XMQ配对



### 组件注册/查询



### CMS注册



### WEB查询/配置

#### 单域



#### 跨域



### WEB推送



# 应用层通信协议定义

## IPv4地址协议

*syntax = "proto2";*

*package msg;*

*message DASMessage*

*{*

*enum Command*

*{*

*DAS\_REQ = 1;*

*DAS\_REP = 2;*

*}*

*required Command command = 1;*

*optional bytes request = 2;*

*optional bytes response = 3;*

*optional int32 status = 4;*

*}*

## 组件协议

*syntax = "proto2";*

*package msg;*

*message Component*

*{*

*enum Type*

*{*

*XMQ = 1;*

*CMS = 2;*

*WEB = 3;*

*DHSLB = 4;*

*DHS = 5;*

*XMSLB = 6;*

*XMS = 7;*

*RTSP = 8;*

*RTMP = 9;*

*HLS = 10;*

*GBS = 11;*

*GBXMS = 12;*

*DSS = 13;*

*DBE = 14;*

*AIS = 15;*

*TSS = 16;*

*ALM = 17;*

*HELMET = 18;*

*PHONE = 19;*

*SLEEP = 20;*

*FACE = 21;*

*FIGHT = 22;*

*}*

*required Type type = 1;*

*required bytes name = 2;*

*required bytes ipv4 = 3;*

*optional bytes id = 4;//组件ID标识*

*optional bytes nickname = 5; ;//组件别名,可由用户自定义*

*}*

*message ComponentMessage*

*{*

*enum Command*

*{*

*SIGNIN\_REQ = 1;*

*SIGNIN\_REP = 2;*

*SIGNOUT\_REQ = 3;*

*SIGNOUT\_REP = 4;*

*QUERY\_REQ = 5;*

*QUERY\_REP = 6;*

*}*

*required Command command = 1;*

*repeated Component components = 2;*

*optional int32 status = 3;*

*}*

## 设备协议

*syntax = "proto2";*

*package msg;*

*message Camera*

*{*

*enum Type*

*{*

*ANALOG = 1;*

*DIGIST = 2;*

*}*

*required Type type = 1;*

*required bytes id = 2;//摄像机ID标识，由设备组件生成*

*required int32 index = 3; //摄像机索引号*

*required bool enable = 4;*

*required bytes nickname = 4; //摄像机名称*

*required bytes ip = 5;*

*}*

*message Device*

*{*

*enum Factory*

*{*

*HIKVISION = 1;*

*DAHUA = 2;*

*EATON = 3;*

*}*

*enum Type*

*{*

*DVR = 1;*

*NVR = 2;*

*IPC = 3;*

*}*

*required bytes device = 1;//设备ID标识，由用户自定义*

*optional bool enable = 2;*

*optional Factory factory = 3;*

*optional Type type = 4;*

*optional bytes username = 5;*

*optional bytes userpwd = 6;*

*optional bytes ip = 7;*

*optional int32 port = 8;*

*optional bytes nickname = 9; //设备名称*

*repeated Camera cameras = 10;*

*}*

*message DeviceMessage*

*{*

*enum Command*

*{*

*NEW\_REQ = 1;*

*NEW\_REP = 2;*

*DELETE\_REQ = 3;*

*DELETE\_REP = 4;*

*QUERY\_REQ = 5;*

*QUERY\_REP = 6;*

*}*

*required Command command = 1;*

*repeated Device devices = 2;*

*optional int32 status = 3;*

*}*

## 算法协议

*syntax = "proto2";*

*package msg;*

*message Algorithm*

*{*

*enum Type*

*{*

*HELMET = 1;*

*PHONE = 2;*

*SLEEP = 3;*

*FIGHT = 4;*

*FACE = 5;*

*}*

*required Type type = 1;*

*required int32 gpu = 2;*

*required bytes id = 3; //算法ID标识，由用户自定义*

*required bytes camera = 4; //摄像机ID标识*

*required float dectect = 5;*

*required float track = 6;*

*optional float seconddetectforphone = 7; //仅PHONE算法使用*

*optional float similarforface = 8; //仅FACE算法使用*

*optional bool inoutforface = 9; //仅FACE算法使用*

*}*

*message AlgorithmMessage*

*{*

*enum Command*

*{*

*NEW\_REQ = 1;*

*NEW\_REP = 2;*

*DELETE\_REQ = 3;*

*DELETE\_REP = 4;*

*QUERY\_REQ = 5;*

*QUERY\_REP = 6;*

*}*

*required Command command = 1;*

*repeated Algorithm algorithms = 2;*

*optional int32 status = 3;*

*}*

## 人员协议

*syntax = "proto2";*

*package msg;*

*message Crew*

*{*

*required bytes id = 1; //人员ID标识，由用户自定义*

*optional bytes name = 2;*

*optional bytes job = 3;*

*repeated bytes pictures = 4;*

*}*

*message CrewMessage*

*{*

*enum Command*

*{*

*NEW\_REQ = 1;*

*NEW\_REP = 2;*

*DELETE\_REQ = 3;*

*DELETE\_REP = 4;*

*QUERY\_REQ = 5;*

*QUERY\_REP = 6;*

*}*

*required Command command = 1;*

*repeated Crew crews = 2;*

*optional int32 status = 3;*

*}*

## 报警协议

*syntax = "proto2";*

*package msg;*

*message Range*

*{*

*required int32 x = 1;*

*required int32 y = 2;*

*required int32 w = 3;*

*required int32 h = 4;*

*}*

*message Alarm*

*{*

*enum Type*

*{*

*HELMET = 1;*

*PHONE = 2;*

*SLEEP = 3;*

*FIGHT = 4;*

*FACE = 5;*

*OFFDUTY = 6;*

*}*

*required Type type = 1;*

*required bytes camera = 2; //摄像机ID标识*

*required bytes timestamp = 3;*

*required bytes picture = 4;*

*repeated Range ranges = 5;*

*optional bytes idforface = 6; //人员ID标识，仅FACE报警使用*

*optional bool inoutforface = 7; //出入状态，仅FACE报警使用*

*optional bytes component = 8; //组件ID标识*

*optional bytes id = 9; //报警推送ID标识，由用户自定义*

*}*

*message AlarmMessage*

*{*

*enum Command*

*{*

*ALARM\_PUSH = 1;*

*ALARM\_ACK = 2;*

*}*

*required Command command = 1;*

*optional Alarm alarm = 2;*

*}*

## 状态消息

*syntax = "proto2";*

*package msg;*

*message State*

*{*

*required int32 state = 1; //实时航行状态*

*}*

*message Controller*

*{*

*required bool operation = 1; //自动/手动标识*

*optional bool state = 2; //航停标识*

*}*

*message StatusMessage*

*{*

*enum Command*

*{*

*SET\_REQ = 1;*

*SET\_REP = 2;*

*QUERY\_REQ = 3;*

*QUERY\_REP = 4;*

*}*

*required Command command = 1;*

*optional State state = 2;*

*optional Controller controller = 3;*

*optional int32 status = 4;*

*}*

## 抓图消息

*syntax = "proto2";*

*package msg;*

*message Image*

*{*

*required bytes device = 1; //设备ID标识*

*required int32 camera = 2; //摄像机索引号*

*optional bytes image = 3;*

*}*

*message CaptureMessage*

*{*

*enum Command*

*{*

*CAPTURE\_REQ = 1;*

*CAPTURE\_REP = 2;*

*}*

*required Command command = 1;*

*optional Image image = 2;*

*optional int32 status = 3;*

*}*

## 时钟消息

*syntax = "proto2";*

*package msg;*

*message ClockMessage*

*{*

*required bytes clock = 1;*

*}*

## AIS消息

*syntax = "proto2";*

*package msg;*

*message PositionA*

*{*

*required int32 repeatindicator = 1;*

*required bytes mmsi = 2;*

*required int32 navigationalstatus = 3;*

*required float rot = 4;*

*required float sog = 5;*

*required int32 positionaccuracy = 6;*

*required bytes longitude = 7;*

*required bytes latitude = 8;*

*required float cog = 9;*

*required int32 trueheading = 10;*

*required int32 timestamp = 11;*

*required int32 specialindicator = 12;*

*required int32 spare = 13;*

*}*

*message ShipStatic*

*{*

*required int32 repeatindicator = 1;*

*required bytes mmsi = 2;*

*required int32 aisversion = 3;*

*required int32 imo = 4;*

*required bytes callsign = 5;*

*required bytes name = 6;*

*required int32 typeofshipcargo = 7;*

*required bytes dimensions = 8;*

*required int32 epfd = 9;*

*required int32 month = 10;*

*required int32 day = 11;*

*required int32 hour = 12;*

*required int32 minute = 13;*

*required float draught = 14;*

*required bytes destination = 15;*

*required int32 dte = 16;*

*required int32 spare = 17;*

*}*

*message StandardSAR*

*{*

*required int32 repeatindicator = 1;*

*required bytes mmsi = 2;*

*required int32 altitude = 3;*

*required float sog = 4;*

*required int32 positionaccuracy = 5;*

*required bytes longitude = 6;*

*required bytes latitude = 7;*

*required float cog = 8;*

*required int32 timestamp = 9;*

*required int32 altitudesensor = 10;*

*required int32 spare = 11;*

*required int32 assignmodeflag = 12;*

*required int32 raimflag = 13;*

*required int32 commstateselectorflag = 14;*

*required int32 communicationstate = 15;*

*}*

*message PositionB*

*{*

*required int32 repeatindicator = 1;*

*required bytes mmsi = 2;*

*required int32 spare = 3;*

*required float sog = 4;*

*required int32 positionaccuracy = 5;*

*required bytes longitude = 6;*

*required bytes latitude = 7;*

*required float cog = 8;*

*required int32 trueheading = 9;*

*required int32 timestamp = 10;*

*required int32 spare = 11;*

*required int32 bunitflag = 12;*

*required int32 bdisplayflag = 13;*

*required int32 bdscflag = 14;*

*required int32 bbandflag = 15;*

*required int32 bmessage22flag = 16;*

*required int32 modeflag = 17;*

*required int32 raimflag = 18;*

*required int32 commstateselectorflag = 19;*

*required int32 syncstate = 20;*

*required int32 slotincrement = 21;*

*required int32 numberofslots = 22;*

*required int32 keepflag = 23;*

*}*

*message AidsToNavigation*

*{*

*required int32 repeatindicator = 1;*

*required bytes mmsi = 2;*

*required int32 navigationtype = 3;*

*required bytes name = 4;*

*required int32 positionaccuracy = 5;*

*required bytes longitude = 6;*

*required bytes latitude = 7;*

*required bytes dimensions = 8;*

*required int32 epfd = 9;*

*required int32 utctimestamp = 10;*

*required int32 epfd = onoffpositionindicator = 11;*

*required int32 atonflag = 12;*

*required int32 raimflag = 13;*

*required int32 virtualflag = 14;*

*required int32 modeindicator = 15;*

*required int32 spare = 16;*

*required bytes extendname = 17;*

*required int32 stufbits = 18;*

*}*

*message AISMessage*

*{*

*enum Type*

*{*

*POSITION\_A = 1;*

*POSITION\_B = 2;*

*SHIP\_STATIC = 3;*

*STANDARD\_SAR = 4;*

*AIDS\_TO\_NAVIGATION = 5;*

*}*

*required Type type = 1;*

*required bytes component = 2;*

*optional PositionA positiona = 3;*

*optional ShipStatic shipstatic = 4;*

*optional StandardSAR standardsar = 5;*

*optional PositionB positionb = 6;*

*optional AidsToNavigation aidstonavigation = 7;*

*}*

# 系统数据定义

## 参数数据定义

### UUID标识

各类应用程序需要使用的UUID标识由应用程序自己创建，并将其存储于XML文件中，XML格式如下所示：

<?xml version="1.0" encoding="utf-8"?>

<Component>

<Helmet>

<Name>Helmet</Name>

<ID>49e97e32-d9f6-4bc5-a6fc-ee9646e7ff41</ID>

</Helmet>

</Component>

## 消息数据定义

消息数据是在网络通信中传输的字节流。字节流使用Google Protocol Buffers开源项目进行串行化。

## 音/视频数据定义

音/视频数据是通过编码算法进行压缩过的字节流数据。

### 流数据定义

### 帧数据定义

## 报警数据定义

报警数据是经过AI算法计算过后产生的图片输出和位置信息数据。

# 音视频编/解码架构设计

## 编码架构设计

## 解码架构设计

### 音频解码设计

### 视频解码设计