OTA 概要、详细设计文档

Revision:	1
Status:	Draft
Author:	Huang Xin
Function:	Netease, Inc.
Path and Filename:	
File Type:	Microsoft Word 2017

Revision History

Rev	Status*	Date	Author	Reason for change/description
1	D	2017/08/04	Huang Xin	First draft

* Statuses include: D: Draft, P: Pending review/frozen, U: Update needed, A: Approved

Approved by: **Date:**

1 概述

本文档详细描述 OTA(Over-The-Air)在智能音箱产品嵌入式软件中的实现。包含了通信协议、内容下载、系统更新、异常恢复四大组成部分。通信协议定义了嵌入式端和服务器的数据交互格式以及内容；内容下载提供 HTTP/HTTPS/FTP/等多种文件下载协议支持，用来从服务器获取系统更新内容；系统更新完成新版本替换旧版本的功能；异常恢复用来确保其它几个部分出现异常时不会导致系统不能正常使用。

2 术语

术语	说明

3 概要设计

3.1 背景信息

OTA 是一种为设备分发新软件、配置，乃至更新加密密钥的方法。OTA 的一项重要特征是，一个中心位置可以向所有用户发送更新，其不能拒绝、破坏或改变该更新，并且该更新为立即应用到频道上的每个人。用户有可能“拒绝”OTA 更新，但频道管理者也可以将其踢出频道。

区别于传统的嵌入式设备系统、软件更新方法，OTA 不需要有线连接，不需要用户进行配置，不需要用户下载，安装其它工具软件。由于没有有线连接，不依赖其它硬件设备，OTA 可以再网络连接正常的情况下同时对大批量设备进行系统更新。

本系统的 OTA 特指利用 OTA 技术，对智能音箱设备的 ROM 进行版本更新。ROM 包含启动代码、linux 内核、文件系统、应用软件等功能。

3.2 系统设计

3.2.1 存储系统分区规划

OTA 更新系统主要包含操作系统和文件系统两部分，操作系统包含了启动代码和 Linux 内核，文件系统包含根目录结构和应用软件，服务。R16 Tina Linux 平台上，启动代码和 Linux 内核整合在一起，存放在 boot 分区中，对应的镜像名称为 boot.img， 文件系统和应用软件整合在一起，存放在 rootfs 分区中，对应的镜像名称为 rootfs.img。OTA 就是对 boot 和 rootfs 分区进行版本更新。为了防止 OTA 损坏系统，需要对 boot 和 rootfs 提供一个安全的备份系统，保障在 OTA 损坏后，系统能从备份系统中恢复启动。OTA 自身也需要和启动代码交互数据，所以会增加一个分区来专门存放 OTA 配置信息。系统分区规划如下（详见表 1）,分区大小如下（详见表 2）：

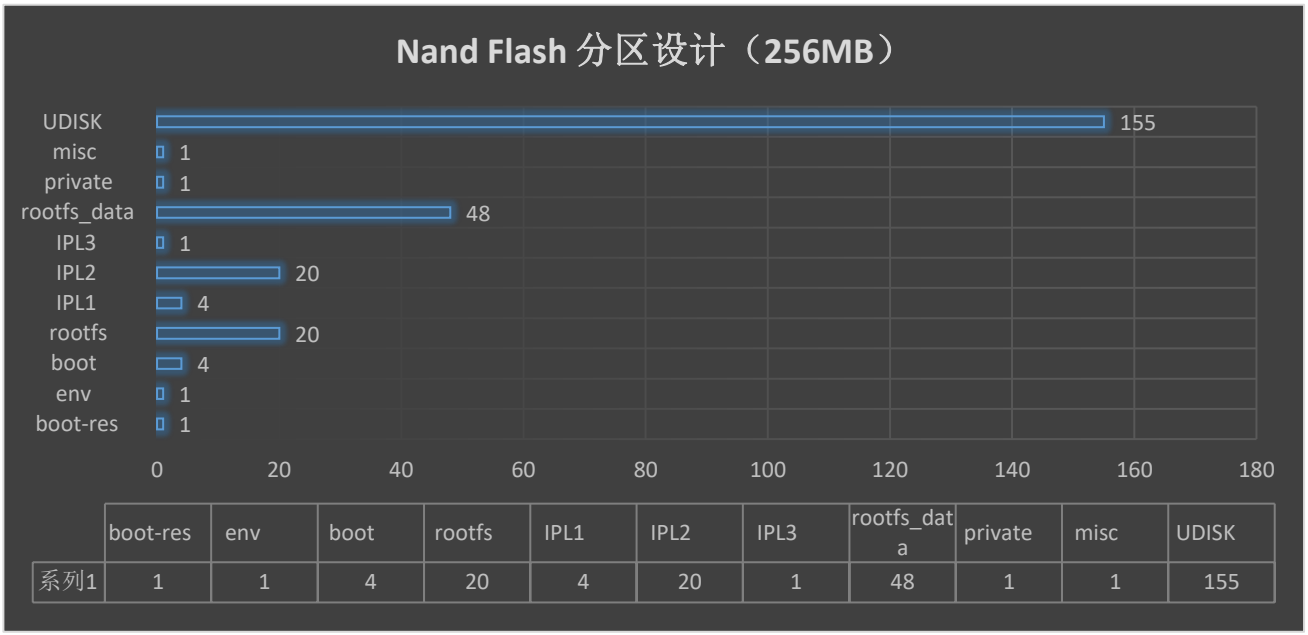


图1. 磁盘分区结构

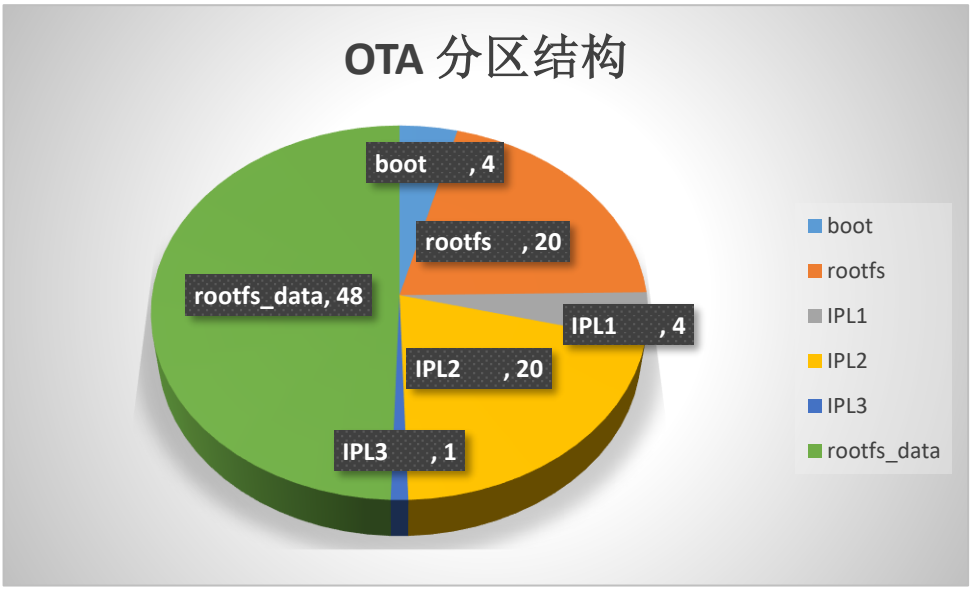


图2. OTA 分区结构

3.2.2 系统启动逻辑

R16 Tina Linux 平台启动代码由 BOOT0 和 UBOOT 两部分组成。系统上电后，BOOT0 首先从系统启动，BOOT0 启动成功后加载 UBOOT 到内存并运行。UBOOT 再加载 Linux 内核和文件系统，完成启动。OTA 相关逻辑在 UBOOT 实现。

UBOOT 启动后，首先计算 boot 和 rootfs 分区数据的 MD5 校验和，并和存放在 IPL3 分区中的参数进行比对。如果 MD5 校验和正确，直接运行 boot 分区中的内核和加载 rootfs 文件系统。如果 MD5 校验和不正确，表示 boot 或者 rootfs 分区中有数据损坏，系统将 IPL1 和 IPL2 中的数据加载到内存启动，并且设置系统进入到恢复模式。系统启动后，如果进入恢复模式，系统尝试重新 OTA 更新系统。

UBOOT 启动时，会将启动计数器加 1，当系统正常启动后，控制中心正常运行，此时会将计数器清零。如果 UBOOT 检测到计数器大于某一个数值，说明系统陷入无限重启模式，UBOOT 自动加载 IPL1 和 IPL2 进入恢复模式，再尝试修复系统。

当每次进行 OTA 升级后，UBOOT 在启动 Linux 内核前，对 rootfs_data, private, misc 分区进行擦除操作，清除旧的系统运行配置，保证 OTA 后的系统是一个干净完整的系统，避免版本不同引起的冲突导致系统无法正常运行。

当使用非 OTA 模式烧写系统时(PhoenixSuit 等软件)，IPL3 没有任何 OTA 配置参数。UBOOT 启动后检测到这种情况，首先会计算 boot、rootfs、IPL1、IPL2 分区的 MD5 校验和，然后判断 boot、rootfs 分区与 IPL1、IPL2 分区的校验和是否相同。如果相同，UBOOT 根据当前的 boot、rootfs 分区内容重新计算 OTA 参数信息并且写入 IPL3 分区，然后正常启动。

3.2.3 OTA 更新实现

OTA 操作可以简单分为将升级包下载到本地，将升级包写入磁盘两部分。接收到服务器发送的 OTA 升级请求后，根据服务器提供的 URL 下载文件到本地并校验文件是否正确。如果下载文件无误，将升级包写入 Nand Flash 对应的分区后，再将分区的数据读出进行校验，如果正确则重启系统完成 OTA 升级。

升级包由 boot.img、rootfs.img、ota.md5 三个文件组成，ota.md5 是 boot.img 和 rootfs.img 的 MD5 校验和。OTA 调用操作系统工具 dd 将文件写入对应分区。

3.2.4 恢复出厂模式功能

系统可以通过特殊按键，在重启的后恢复系统到出厂模式。当 UBOOT 检测到特殊按键生效时，会擦除 rootfs_data, private, misc, UDISK，分区，清除所有用户数据。并且利用 IPL1 和 IPL2 分区覆盖 boot 和 rootfs 分区后再启动。这样整个系统运行环境恢复到了和出厂时一模一样的设置。

分区名	功能	OTA 功能
boot-res	存放启动代码资源（图片、声音等）	OTA 不操作该分区
env	存放 UBOOT 环境变量	OTA 不操作该分区
boot	UBOOT 启动代码和 Linux 操作系统内核	OTA 自动更新
rootfs	存放根文件系统	OTA 自动更新
IPL1	存放 boot 分区备份	OTA 失败后加载
IPL2	存放 rootfs 分区备份	OTA 失败后加载
IPL3	存放 OTA 参数	OTA 参数存储
rootfs_data	overlay 分区，存储用户在文件系统中的修改	OTA 更新后擦除
private	私有分区，存放操作系统用配置信息	OTA 更新后擦除
misc	存放 misc_upgrade 用参数	OTA 更新后擦除
UDISK	用户数据分区	OTA 不操作该分区

表1. 分区配置信息

分区名	Size(M/Bytes)
boot-res	1
env	1
boot	4
rootfs	20
IPL1	4
IPL2	20
IPL3	1
rootfs_data	48
private	1
misc	1
UDISK	Left

表2. 分区大小信息

3.2.5 安全性（扩展）

保证 OTA 升级包数据的安全性，防止破解，暂不支持。

4 详细设计

4.1 通信协议

OTA 利用 D-Bus 总线和控制中心进行通信，利用通信协议传送命令和数据。通信协议包含控制中心发送到 OTA 模块的控制命令和 OTA 发送到控制中心的状态信息协议。

4.1.1 控制命令协议

控制中心控制 OTA 模块协议，用于控制 OTA 模块下载、更新系统。控制中心获悉服务器有新版本需要进行 OTA 更新时，通过控制命令协议通知 OTA 模块进行操作。

4.1.1.1 CMD_OTA_NOTIFY 协议

控制中心控制命令通知协议，协议格式如下：

```
{
        "version":      x,      // 版本
        "otaCmd":       x,      // 命令
        "otaMode":      x,      // 模式，强制升级，普通升级
        "otaFileInfo":  {
                "url":  "",     // URL with base64
                "md5":  "",     // md5 not used base64
                "size": x       // ota file size
        }
}
```

otaCmd 支持以下命令：

- **OTA_CMD_DOWNLOAD：** 从服务器下载升级包
- **OTA_CMD_USED_LOCAL_IMAGE：** 从本地下载的升级包进行 OTA 升级
- **OTA_CMD_EXEC：** 进行 OTA 升级

otaMode 支持以下模式：

- OTA_MODE_NORMAL： 普通模式
- OTA_MODE_FORCE_NOW： 强制升级模式
- OTA_MODE_RECOVERY： 恢复模式

4.1.1.2 CMD_SYSTEM_STANDBY 协议

系统启动正常命令，通知 OTA 更新启动计数。

4.1.2 OTA 状态反馈协议

该协议用于通知控制中心当前 OTA 模块的状态。

4.1.2.1 CMD_OTA_STATUS 协议

协议格式如下：

```
{
        "status":       2,
        "val":  0
}
```

status 支持以下命令：

- OTA_CURRENT_VERSION：通知控制中心当前系统版本号
- OTA_CURRENT_SETUP_MODE：通知控制中心当前系统模式，普通模式/恢复模式
- OTA_CURRENT_REBOOT_TIME：通知控制中心当前系统启动次数（未成功运行）
- OTA_DOWNLOAD_FILE：通知控制中心当前升级包下载结果
- OTA_DOWNLOAD_PROGRESS：通知控制中心当前升级包下载进度

- OTA_VERIFY_FILE：通知控制中心文件校验结果
- OTA_VERIFY_PARTITION：通知控制中心分区校验结果
- OTA_DECOMPRESS_FILE：通知控制中心当前开始解压文件
- OTA_UPGRADE_READY：通知控制中心 OTA 准备完毕
- OTA_UPGRADE_START：通知控制中心当前开始 OTA 更新系统
- OTA_UPGRADE_PARTITION：通知控制中心分区更新结果
- OTA_RECOVERY_START：通知控制中心开始恢复 OTA 升级
- OTA_REBOOT_SYSTEM：通知控制中心当前 OTA 结束，准备重启系统
- OTA_SUCCESED：通知控制中心当次 OTA 升级成功
- OTA_ERR_CODE：通知控制中心当前操作发生异常以及原因

4.2 交互逻辑图

4.2.1 普通 OTA 数据交互

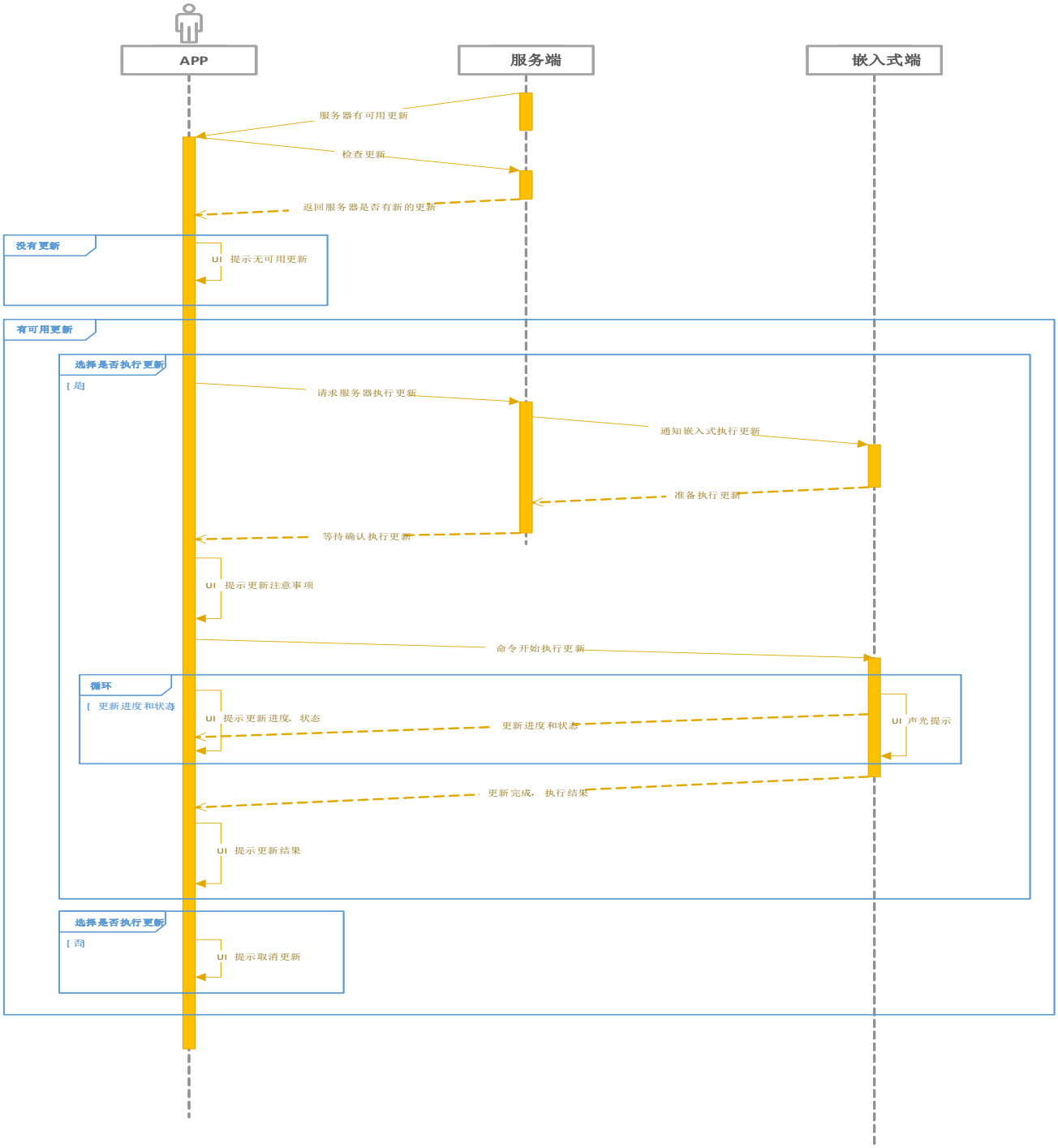


图3. 普通 OTA 数据交互图

4.2.2 强制 OTA 数据交互

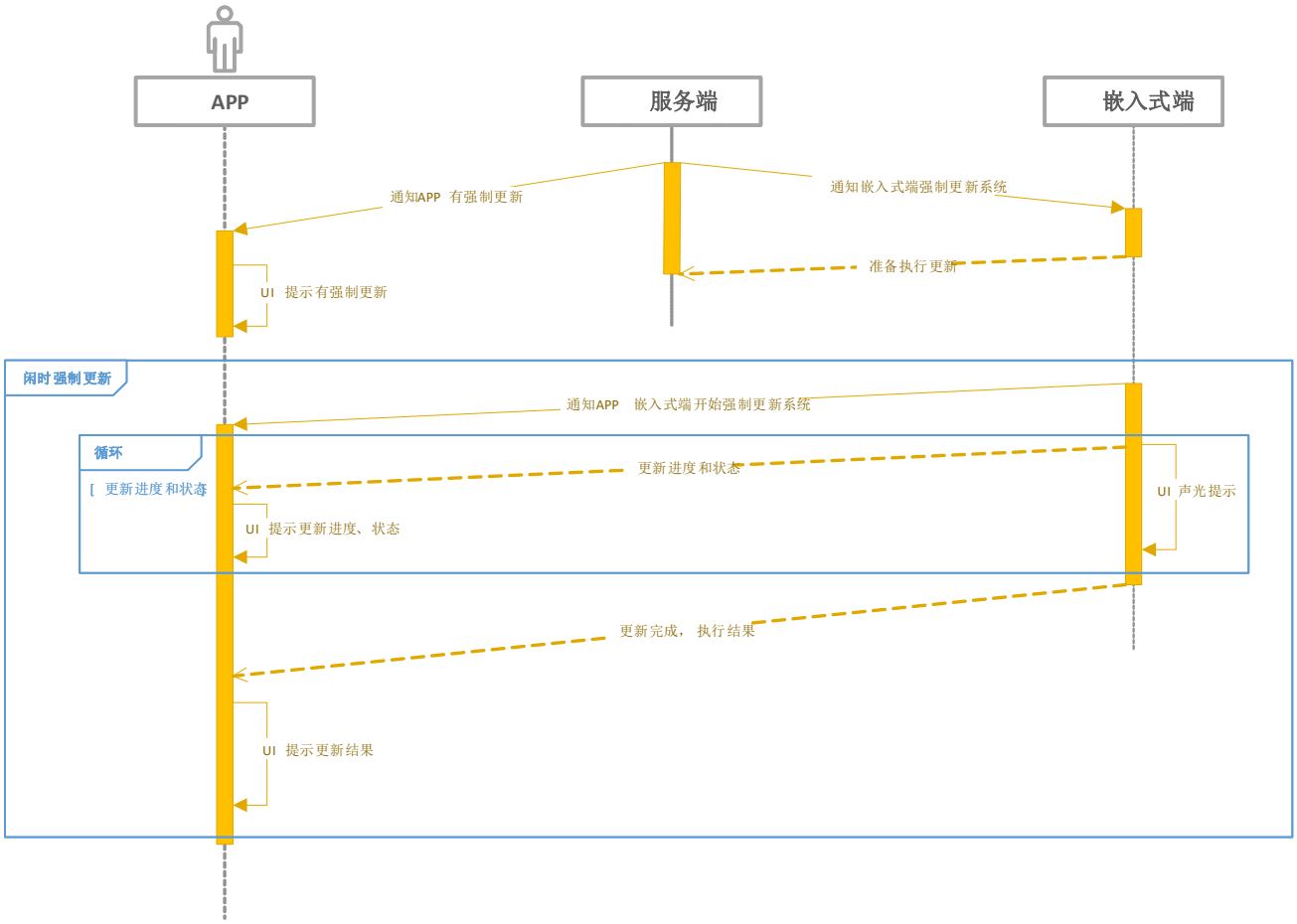
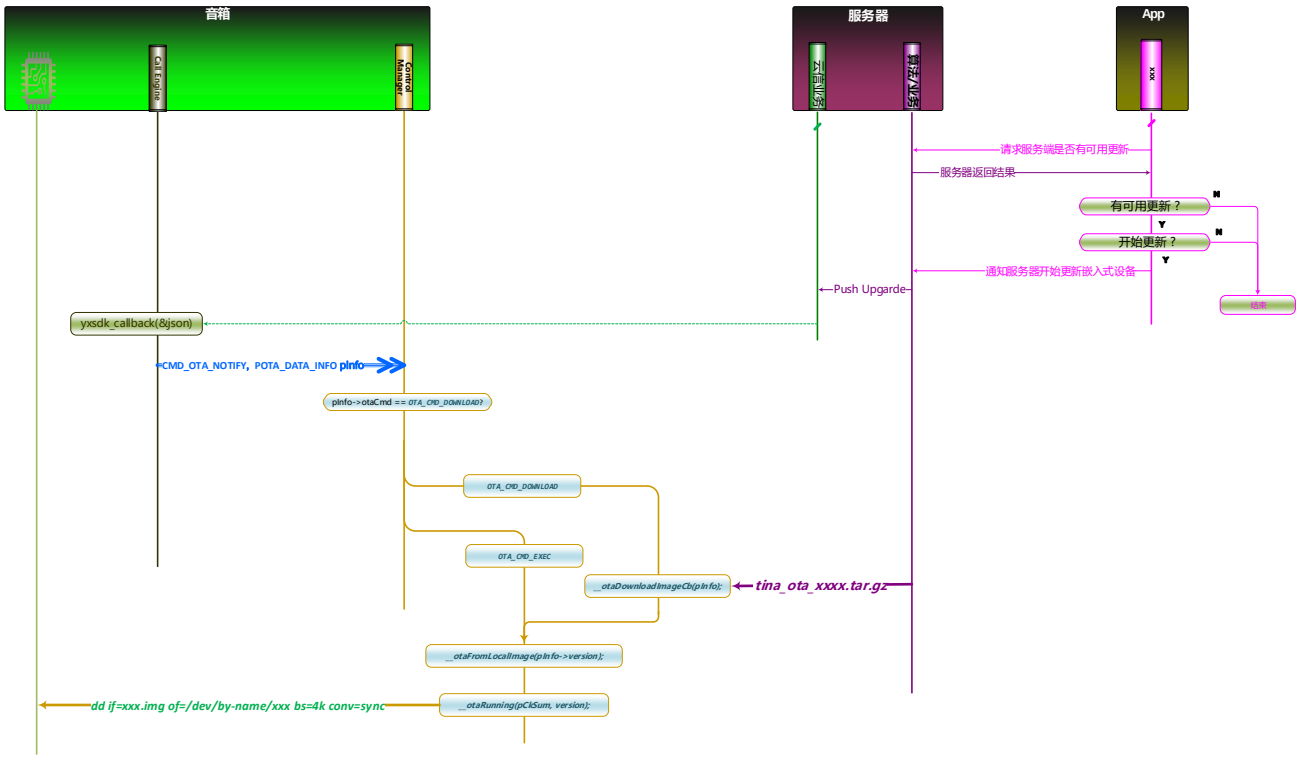


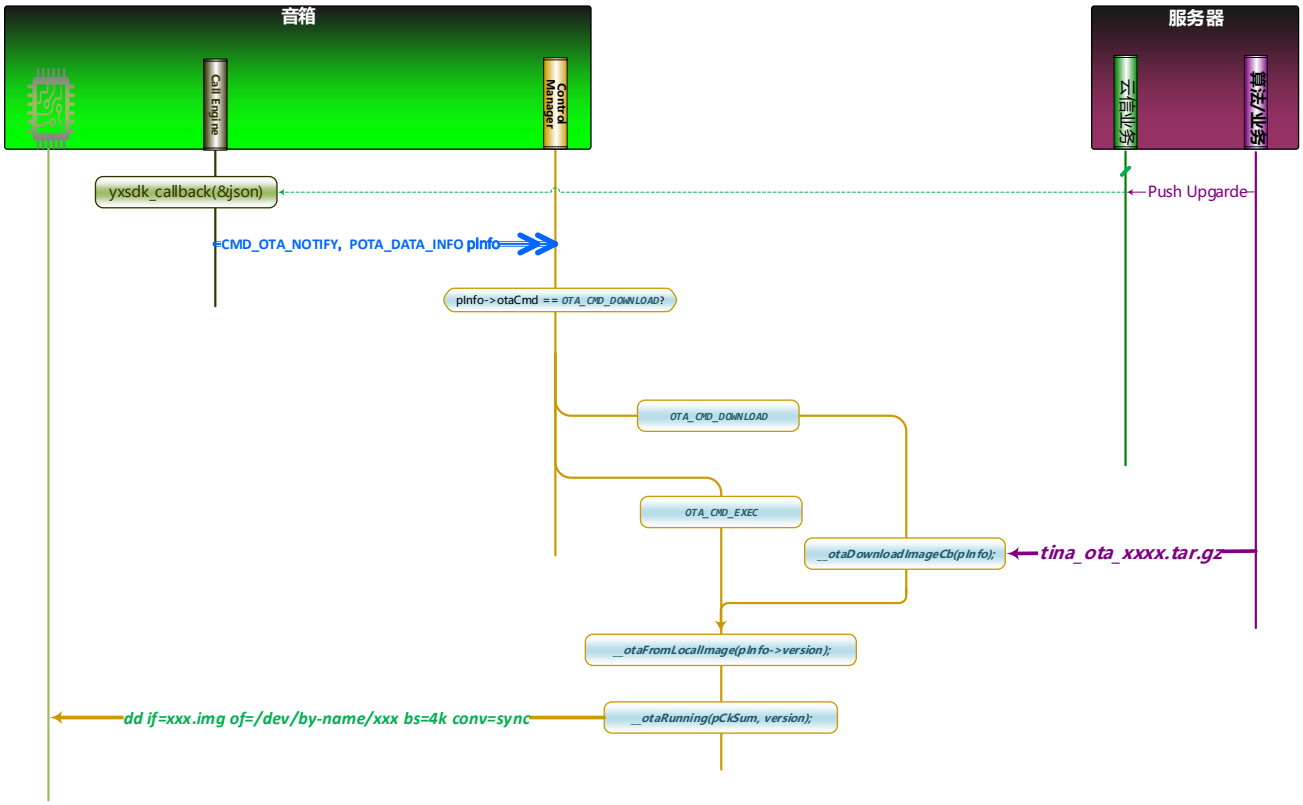
图4. 强制 OTA 数据交互图

4.3 程序流程图图

4.3.1 普通 OTA 更新程序流程图



4.3.2 强制 OTA 更新程序流程图



4.4 软件设计

4.4.1 常量定义

4.4.1.1 通用常量宏定义

```
#define SIZE_1K                     (1024)                  ///< 1K 字节大小
#define MD5_CHKSUM_LEN              (16)                    ///< MD5 校验和长度
#define MD5_CHKSUM_STR_LEN          (MD5_CHKSUM_LEN * 2 + 1)    ///< MD5 校验和字符串格式长度
#define NO_OTA_STATUS_VAL_TAG       (0)                     ///< 状态默认值
```

4.4.1.2 系统启动模式常量定义

```
/**
 * 系统启动模式定义
 */
typedef enum
{
    NORMAL_SETUP                = 0x1234,                   ///< 普通启动模式
    RECOVERY_SETUP,                                         ///< 恢复模式
    SYSTEM_OTA,                                             ///< OTA 升级模式
} SETUP_MODE;
```

4.4.1.3 错误码常量定义

```
/**
 * 错误码
 */
typedef enum
{
    ERR_NO_INIT_IPL3 = CMD_OTA_NOTIFY + 1,          ///< (0xF101) 未初始化 OTA 参数
    ERR_BAD_IPL3,                                   ///< OTA 参数异常
    ERR_BAD_FILE_SIZE,                              ///< 文件大小不正确
    ERR_MD5_FILE,                                   ///< 计算文件 MD5 校验和异常
    ERR_MD5_CHECK_SUM,                              ///< MD5 校验和不正确
    ERR_OTA_WRITE_BOOT,                             ///< 写入 BOOT 分区异常
    ERR_OTA_WRITE_ROOTFS,                           ///< 写入 ROOTFS 分区异常
    ERR_OTA_WRITE_PARAMS,                           ///< 写入 OTA 参数异常
    ERR_OTA_DOWNLOAD_FILE,                          ///< 下载文件失败
```

```
    ERR_VERIFY_PARTITION_MD5,                   ///< 校验分区 MD5 异常
    ERR_OTA_PRE_STATR,                          ///< 先前已经启动了一个未完成的 OTA 任务
    ERR_OTA_YET_CUR_VER,                        ///< 当前版本已经更新
    ERR_OTA_NOT_READY,                          ///< OTA 未准备好，暂时不能进行 OTA 升级
} OTA_ERROR;
```

4.4.1.4 OTA 状态命令常量定义

```
/**
 * OTA 状态命令定义
 */
typedef enum
{
    OTA_CURRENT_VERSION         = 0,            ///< 当前系统版本号
    OTA_CURRENT_SETUP_MODE,                     ///< 当前系统启动模式
    OTA_CURRENT_REBOOT_TIME,                    ///< 当前系统未正常启动次数
    OTA_DOWNLOAD_FILE,                          ///< OTA 升级包下载状态
    OTA_DOWNLOAD_PROGRESS,                      ///< OTA 升级包下载进度
    OTA_VERIFY_FILE,                            ///< 文件校验结果
    OTA_VERIFY_PARTITION,                       ///< OTA 分区校验结果
    OTA_DECOMPRESS_FILE,                        ///< OTA 当前进度：解压文件
    OTA_UPGRADE_READY,                          ///< OTA 更新准备完成
    OTA_UPGRADE_START,                          ///< OTA 更新开始
    OTA_UPGRADE_PARTITION,                      ///< OTA 写入分区数据
    OTA_RECOVERY_START,                         ///< OTA 恢复模式启动
    OTA_REREQ_OTA_NOW,                          ///< OTA 请求立即执行 OTA 更新系统
    OTA_REBOOT_SYSTEM,                          ///< OTA 准备重启系统
    OTA_SUCCESED,                               ///< OTA 执行成功
    OTA_UNKNOWN_CMD,                            ///< 位置 OTA 状态命令
    OTA_ERR_CODE,                               ///< OTA 过程异常
} OTA_STATUS_TYPE;
```

4.4.1.5 OTA 模式常量定义

```
/**
 * OTA 模式
 */
typedef enum
{
    OTA_MODE_NORMAL             = 0x2345,       ///< 普通 OTA 更新模式
    OTA_MODE_FORCE_NOW,                         ///< 强制 OTA 更新模式
    OTA_MODE_RECOVERY,                          ///< OTA 恢复模式
```

```
} OTA_MODE;
```

4.4.1.6 OTA 控制命令常量定义

```
/**
 * OTA 操作命令
 */
typedef enum
{
    OTA_CMD_DOWNLOAD            = 1234,                 ///< 下载 OTA 升级包
    OTA_CMD_USED_LOCAL_IMAGE,                           ///< 从上次备份的 OTA 升级包升级
    OTA_CMD_EXEC,                                       ///< 开始 OTA 更新系统
} OTA_CMD;
```

4.4.2 结构体定义

4.4.2.1 OTA 升级包信息数据结构

```
/**
 * OTA 级包信息
 *
 */
typedef struct
{
    char            url[SIZE_1K];                       ///< URL, Base64 编码
    char            md5[MD5_CHKSUM_STR_LEN];            ///< OTA 升级包 MD5 校验和
    unsigned int    size;                               ///< OTA 升级包文件大小
} OTA_FILE_INFO, *POTA_FILE_INFO;
```

4.4.2.2 控制中心控制命令协议数据结构

```
/**
 * 控制中心控制命令协议
 *
 * @see OTA_CMD
 * @see OTA_MODE
 * @see OTA_FILE_INFO
 */
```

```
typedef struct
{
    int             version;                                    ///< OTA 版本号
    OTA_CMD         otaCmd;                                     ///< OTA 命令
    OTA_MODE        otaMode;                                    ///< OTA 模式
    OTA_FILE_INFO   otaFileInfo;                                ///< OTA 升级包信息
} OTA_DATA_INFO, *POTA_DATA_INFO;
```

4.4.2.3 OTA 状态协议数据结构

```
/**
 * OTA 状态信息协议
 *
 * @see OTA_STATUS_TYPE
 */
typedef struct
{
    OTA_STATUS_TYPE     status;                                 ///< 状态命令
    int                 val;                                    ///< 状态值
} OTA_RSP_STATUS, *POTA_RSP_STATUS;
```

5 Review Information

Topic of Review: **Date**:

Related Documents:

Meeting Moderator:

Meeting Scribe:

Invitees	Attended (Y/N)

Approval required from anyone who didn't attend?

If so, from whom?

Agenda/Issues discussed:

1.
2.
3.
4.

Outcome:

Action items

Item	Owner	Date Due	Status	Closed?