1 管理

1.1 缩略词

缩写	描述
LE	Location Engine
AP	Access Point
IP	Internet Protocol
LAN	Local Area Network
MPDU	802.11 Mac Protocol Data Unit
MU	Wi-FiMobile Unit
PDU	Protocol Data Unit
RSSI	Received Signal Strength Indication
UDP	User Datagram Protocol
Wi-Fi	Wireless Fidelity

2 接口基础

2.1 概述

AP（无线接入点）和LE(定位服务器）的通信通过UDP/IP实现。

AP以主动转发的方式上报终端信息报文：通过WAC开启定位功能后，即可主动向指定服务器的IP/Port发送当前扫描到的终端报文；LE在指定的端口上监听AP转发过来的报文。

2.2 接口信息

本协议中的信息即为AP转发终端消息：AP直接将终端消息转发给LE。

2.3 AP的MAC地址

AP的物理MAC地址通常与硬件绑定，并且唯一；可以通过AP转发终端消息将AP的MAC地址发至LE，并被LE使用。

2.4 协议标准

2.4.1 字节序

除非特殊说明，所有数字都以大字节序写入，如MAC地址。

2.4.2 IP 地址

IP地址为4字节长，标准IPV4地址，大字节序。例如：地址255.254.253.252表示为FF:FE:FD:FC。

3. 接口说明

3.1 协议数据单元结构

AP转发终端消息符合标准协议数据单元（PDU）结构。

PDU通用协议如下：

Header	Request ID	Code	Sub Code	Data Length	Data Payload
2 字节	2字节	1字节	1字节	2字节	0-2048字节

PDU协议说明：

PDU 字段	字节	描述	取值	注
Header	2	固定的头两个字节	0xCC83	
Request ID	2	一个请求ID序列号. 每一条服务器消息增加1.	任何值	1
Code	1	操作码	有效的码值	
Sub-Code	1	操作码的字码	有效的字码值	
Data Length	2	Data Payload字段的长度	0x0000-0x0FFF	2
Data Payload	0-2048	消息的有效载荷，结构依赖于Code字段的值		

注：

1. Data Length.该字段代表数据负载(不包括消息头)的字节数。例如，当没有数据时，该字段的值为0。

2. 发送复合报文时，Data Payload部分为多个消息报文叠加；Data Length应为消息类型报文长度的整数倍，如不对应即丢弃；并且，复合报文Data Payload长度不超过2048字节。

3.2 AP转发终端消息

3.2.1 单MU报文

描述

在 AP 接收到 MU 消息后直接转发给 LE 的消息。

PDU 说明

PDU字段	值
Header	0xCC83
Request ID	0
Code：	0xD6
Sub-Code：	0
Length：	40

数据负载结构

数据字段	字节	描述	取值	注
AP MAC Address	6	源AP的MAC地址	有效MAC地址	
Vendor ID	2	厂商ID	厂商ID	
MU-MAC Address	6	信号源的MAC地址	有效MAC地址	
Radio Type	1	接收消息的无线电类型： 0x01：802.11b 0x02：802.11g	0x01 –0x03	

		0x03：802.11a 其他：保留		
Channel	1	AP接收MU消息所在的信道	有效的信道号 1~255	
Is Associated	1	MU是否连接到AP： 0x01：是 0x02：否 其他：保留	0x01 –0x02	
AssociatedAP-Mac	6	为该终端关联到的AP的物理地址	有效MAC地址 非关联为0	
MU Type	1	MU的类型： 0x01：未知类型 0x02：Wi-Fi移动单元 0x03：流氓AP 其他：保留	0x01–0x03	
RSSI	1	测量到的以dBm为单位的RSSI整数值	十进制：-128到127	
Noise Floor	1	测量到的以dBm为单位的本地噪声整数值	十进制：-128到127	
Age	2	该测量值的测试时间与发送时间差；单位为s	0~65535	
MU-IPv4	4	MU的IP，见4.5.4	非关联时赋值为0	
Reserved	8	预留字段	赋值为0	

3.2.2 Compounded复合报文

描述

在AP接收到MU消息封装成复合报文后直接转发给LE的消息。

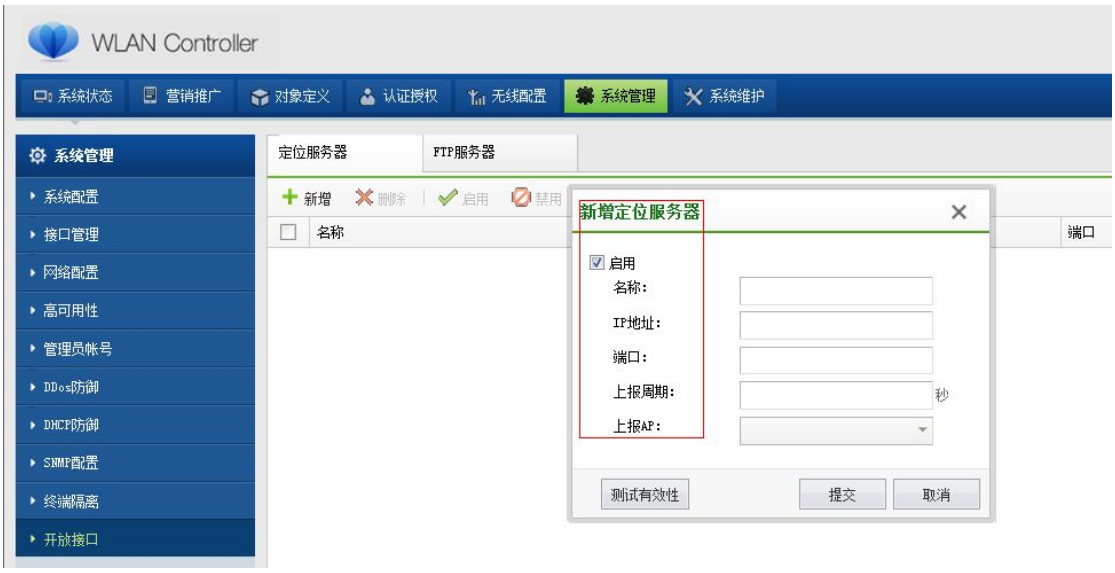
PDU说明

PDU字段	值
Header	0xCC83
Request ID	0
Code：	0xD8
Sub-Code：	0
Length：	4+N

数据负载结构

数据字段	字节	描述	取值	注
Numberof Compounded Messages	2	复合报文中报文单元的个数	0x0001-0xFFFF	
Reserved	2	0		
CompoundedMessages	N	完整的PDUs和Payloads	1–42	

4. 新增定位服务器



名称：填写任意名称，区分不同的定位服务器

IP 地址：定位服务器的 IP 地址

端口：定位服务器的 UDP 端口

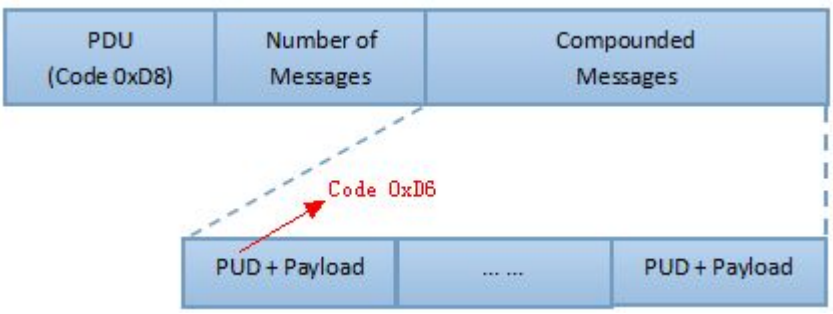
上报周期：AP 上报终端信息给定位服务器的周期

上报 AP：选择上报终端信息给定位服务器的 AP

5. AP上报终端消息数据结构

5.1 终端消息数据结构图

WAC 页面新增定位服务器，AP 通过 UDP 通信方式直接将终端信息转发给定位服务器，转发终端消息采用标准协议数据单元（PDU）结构。目前 AP 采用 PDU 结构复合报文转发终端信息给定位服务器，复合报文结构如 3.2.2 所示，复合报文数据负载包括若干单 MU 报文，单 MU 报文个数范围为 1-42。复合报文消息结构如下图所示：



5.2 终端消息数据结构

5.2.1 PDU通用头格式

PDU 通用头格式如下图所示，复合报文 PDU 头中的 code 字段为 0xD8，复合报文负载数据中 MU 报文 PDU 头中的 code 字段为 0xD6。

```
/* PDU 通用头格式 */
typedef struct PDU_header
{
    unsigned char    header[2];        /* 固定头 2 字节, 0xCC83   */
    unsigned char    request_id[2];    /* 请求 id ，暂补充为 0 */
    unsigned char    code;             /* 操作码（0xD8 或者 0xD6）*/
    unsigned char    sub_code;         /* 操作码-子码，暂补充为 0 */
    unsigned char    data_length[2];   /* Payload 数据负载长度 */
}PDU_header_t;
```

5.2.2 单MU报文数据负载部分

单MU报文数据负载部分如下图所示，如果STA关联上AP，数组assoc_ap[6]就是AP的MAC地址，否则就是全0。

```
/* 单 MU 报文数据负载部分 */
typedef struct MU_data_payload
{
     unsigned char      ap_mac[6];      /*   ap 的 mac 地址 */
    unsigned char      verdor_id[2];     /* 暂补充为 0 */
    unsigned char      sta_mac[6];       /* 终端的 mac 地址 */
    char          radio_type;            / * 射频类型 */
   unsigned char     channel;            /* 收包信道*/
   char          is_assoc;               /* 终端是否关联 */
   unsigned char     assoc_ap[6];        /* 关联 ap   */
   char          mu_mode;          /* 终端类型, 默认为 0x2（wifi 移动单元）*/
   char          rssi;                   /* 信号强度 */
   char          noise_floor;            /* 信道底噪 */
   unsigned char     age[2];             /*   age 时间差*/
   unsigned char     mu_ipv4[4];         /* 暂补充为 0 */
   unsigned char     reserved[8];        /* 暂补充为 0 */
}MU_data_payload_t;
```

5.2.3 单MU报文格式

单MU报文格式如下图所示。

```
/* 单 MU 报文格式 */
typedef struct PDU_MU_info
{
    PDU_header_t       pdu_header;          /* 消息头,code = 0xD6 */
   MU_data_payload_t   mu_data_payload;      /* 数据负载部分. */
}MU_info_t;
```

5.2.4 复合报文数据负载部分

复合报文数据负载部分如下图所示，数据负载包括若干单MU报文，单MU报文个数范围为1-42。

```
/* 复合报文数据负载部分 */
typedef struct Compounded_data_payload
```

```
{
    u_char      mu_num[2];          /* MU 报文个数（1 - 42）*/
    u_char      reserved[2];        /*  暂补充为 0 */
    MU_info_t   sta_node[0];        /*  完整的  PDUs 和  Payloads */
}Com_data_payload_t;
```

5.2.5 复合报文格式

复合报文格式如下图所示。

```
/*  复合报文格式  */
typedef struct Compounded_info
{
    PDU_header_t          pdu_header;          /*  消息头  code = 0xD8 */
    Com_data_payload_t    Com_data_payload;    /*  数据负载  */
}Compounded_info_t;
```

6. 定位服务器解析终端消息

6.1 创建UDP的socket，并绑定IP地址与端口

将WAC页面设置定位服务器的IP地址与端口赋值给ip_addr与port，然后使用创建UDP的socket绑定该IP地址与端口，如下图所示。

```
//创建udp的socket
s = socket(AF_INET, SOCK_DGRAM, 0);
memset(&addr_server, 0, sizeof(addr_server));
addr_server.sin_family = AF_INET;
addr_server.sin_addr.s_addr = inet_addr(ip_addr);
addr_server.sin_port = htons(port);
//绑定指定服务器的ip地址与端口
bind(s, (struct sockaddr *)&addr_server, sizeof(addr_server));
```

6.2 解析AP转发的终端消息

终端消息所有字段字节长度不超过2048，将接收到的终端消息保存在缓存buf中，使用第5节定义的Compounded_info_t类型指针指向该buf，这样就可以解析出终端消息中的所有数据。首先获取复合报文的头部，头部中的code字段为0xD8；然后获取复合报文的数据负载，从中得到mu报文的个数以及获取复合报文中的mu报文；最后依次获取单MU报文的头部与数据负载，头部中的code字段为0xD6。解析复合报文终端消息参考第7节。

```
memset(buf, 0 , 2048);
n = recv(s, buf, 2048, 0);
parse_sta_info = (Compounded_info_t *)buf;
```

7. 解析复合报文终端消息

```
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <stdio.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>

int main(int argc , char *argv[])
{
    int s,n;

    char ip_addr[32] = "127.0.0.1";

    int port = 0;
    int opt;

    char buf[2048];

    struct sockaddr_in addr_server;

    while ((opt = getopt(argc, argv, "s:p:")) != -1)
    {
        switch (opt)
        {
          case 's':
                strcpy(ip_addr,optarg);
                break;
          case 'p':
                port = atoi(optarg);
                break;
          default:
                break;
      }
    }

    //创建udp的socket
    s = socket(AF_INET, SOCK_DGRAM, 0);

    if(s < 0)
        return -1;

    memset(&addr_server, 0, sizeof(addr_server));
    addr_server.sin_family = AF_INET;
    addr_server.sin_addr.s_addr = inet_addr(ip_addr);
    addr_server.sin_port = htons(port);
```

```
    //绑定指定服务器的ip地址与端口
    bind(s, (struct sockaddr *)&addr_server, sizeof(addr_server));

    while(1)
    {
        int i = 0, mu_sum = 0;
        Compounded_info_t    * parse_sta_info = NULL;
        PDU_header_t         * parse_header_info = NULL;
        Com_data_payload_t * parse_com_payload_info = NULL;
        MU_info_t            * parse_mu_info = NULL;
        MU_data_payload_t    * parse_mu_payload_info = NULL;

        memset(buf, 0 , 2048);
        n = recv(s, buf, 2048, 0);

        //获取AP上报的终端消息
        parse_sta_info = (Compounded_info_t *)buf;

        //获取复合报文的pdu头部
        parse_header_info = (PDU_header_t *)&parse_sta_info->pdu_header;

        //获取复合报文的数据负载
        parse_com_payload_info = (Com_data_payload_t *)&parse_sta_info->Com_data_payload;

        //获取复合报文数据负载中的MU报文个数
        mu_sum = *(unsigned short *)parse_com_payload_info->mu_num;

        //获取复合报文数据负载中的MU报文
        parse_mu_info = (MU_info_t *)&parse_com_payload_info->sta_node[0];

        for (i = 0; i < mu_sum; i++)
        {
            //获取单MU报文的pdu头部
            parse_header_info = parse_mu_info[i].pdu_header;

            //获取单MU报文的数据负载
            parse_mu_payload_info = parse_mu_info[i].mu_data_payload;
        }
    }

    return 0;
}
```