**文件编号：LTE-HeNB\_MAC\_DL\_DYCS\_V0.5**

**TD-LTE HeNB协议栈软件系统**

**MAC下行数据发送代码**

**理解报告**

拟制：杜红艳

时间：2011.11.23

**中国科学院计算技术研究所**

**无线通信技术研究中心**

**软件组**

**LTE协议栈研发项目组**

修改记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文件编号 | 版本号 | 拟制人/  修改人 | 拟制/修改日期 | 更改理由 | 主要更改内容  （写要点即可） |
|  | 0.5 | 杜红艳 | 2010-11-23 | 建立 |  |
|  | 0.6 |  |  |  |  |
| 注1：每次更改归档文件（指归档到组内及研究室的文件）时，需填写此表。  注2：文件第一次归档时，“更改理由”、“主要更改内容”栏写“无”。 | | | | | |

修改列表：



本文档的程序或内容受版权法的保护，未经中科院计算所的书面许可，不得擅自泄漏、拷贝或复制本文档资料的全部或部分。

**目录**

**理解报告按照自己撰写的内容生成文档目录**

**MAC数据发送代码理解报告**

# 引言

## 编写目的

本文档将作为MAC接收模块的代码编写的依据，详细说明了本模块功能、结构、函数定义，以及与其它模块的接口。本说明书的读者为LTE HeNB端MAC接收模块设计、编码人员、测试人员、项目组负责人员、实验室主任及相关项目管理人员。

编写本说明书的目的在于

* 为开发人员提供依据。
* 为修改和维护本系统提供条件。
* 项目负责人将根据本文档计划和控制系统设计、开发的全过程。

## 定义

1. **MAC Media Access Control**
2. **SDU Segment Data Unit**
3. **PDU Protocal Data Unit**

## 参考资料

1. **《TD-LTE\_HeNB协议栈软件系统LowMAC控制模块详细设计说明书\_v1.0》**
2. **《TD-LTE\_HeNB协议栈软件系统MAC发送模块详细设计说明书\_v1.0》**
3. **《3GPP TS 36.321, "Medium Access Control (MAC) protocol specification", Rel. 9, V 9.3.0》**

# MAC数据发送过程

MAC数据发送流程：

1. 做上行调度和下行调度。

2. 根据调度结果取出相应MAC SDU及RAR消息。

3. 构造MAC PDU。

4. 做下行HARQ。

5. 递交给物理层MAC PDU及控制信息。

MAC运行机制：

Henb初始化创建数据收发线程，并调用收发处理函数完成数据的发送与接收，具体初始内容如下：

init\_lowmac(); 创建tLowMacRx 进程，调用lowmac\_ctrl()处理上下行帧。

init\_pdu\_rx(); 创建MACRx 进程，调用rx\_mac\_msg()接收ue发来的数据。

init\_henb\_send();创建tHenbMacTx进程，调用start\_henb\_send()来始henb下行数据处理。

init\_sched(); 初始化g\_dl\_phy\_res 全局变量(DlPhyResInfo类型)

init\_pdu\_construct();

## MAC数据发送相关函数调用关系

### init\_lowmac()

### init\_pdu\_rx()

说明：

deal\_puschtb\_ind函数会将将收到的上行数据发到 g\_to\_mac\_rx\_msgq 消息队列。

### init\_henb\_send()

说明：

1. init\_henb\_send()创建henb发送进程
2. start\_henb\_send()获取发送信号量g\_mac\_tx\_sem，并调用发送函数来发送数据。
3. get\_tx\_fn() 获取系统帧号、子帧号

### henb\_dl\_sched()

说明：

1. get\_phy\_resource\_info() 用于得到可用的物理资源信息(DlPhyResInfo类型)。(目前只得到dlsch 资源，存于g\_dl\_phy\_res.dlsch\_res\_lst )
2. alloc\_pch\_bch\_res()用于从RLC层得到pch/bch 数据(目前得到DlSchedResult.bch\_result类型数据)，并生成pch/bch调度结果。(目前只得到bch数据,没有pch数据)
3. build\_dl\_sched\_queue()得到下行调度队列(DlSchedQueue类型)。
4. do\_dl\_sched(DlSchedQueue \*dl\_sched\_queue\_p, DlSchedResult \*dl\_sched\_result\_p) 根据下行调度队列取得下行调度结果(DlSchedResult类型)。

### process\_dlsch\_result()

说明：

1. **process\_dlsch\_result**(DlSchedResult \*dlsched\_result\_p, MacToLowMac \*tx\_p)根据下行调度结果得到lowmac使用的发送数据。主要是调用 RLC层获取数据接口，得到调度结果所需要大小的逻辑信道数据。
2. **process\_dl\_sch**(DlSchedSchLst \*sch\_lst\_p, MacToLowMac \*tx\_p)根据 Dlsch传输信道调度大小调用RLC接口获得待发送数据。
3. 用到的RLC层接口定义如下：

INT32 **get\_rlc\_pdu**(LchType lch\_type, Rnti rnti, LchResultLst \*lc\_sch\_list\_p, LcRlcPduList \*lc\_rlc\_pdu\_list\_p) 根据调度数据大小得到下行逻辑信息数据。

INT32 **pack\_rlc\_pdu\_node**(Rnti rnti, UINT8 lc\_id, UINT8 \*pack\_location\_p, TxPduNode \*rlc\_pdu\_node\_p) 将rlc\_pdu\_node\_p指向数据打包成传输信道发送的数据(pack\_location\_p指向)

### process\_c\_rnti\_node()

说明：

1. **process\_c\_rnti\_node**(DlSchedSchResultNode \*dlsch\_node\_p,MacToLowMac \*tx\_p)根据Dlsch调度结果(处理CRNTI, SPS CRNTI, TCRNIT 类型数据)得到Dlsch发送数据, 。

INT32 **construct\_ue\_pdu**(InputData \*input\_data\_p, UINT16 alloc\_size, UINT8 \*\*pdu\_data\_p) 根据

### process\_other\_node()

说明：

get\_rar() 根据全局变量g\_ra\_rnti\_table[rnti] 中RARnti信息，添加随机接入的UE实体，

### do\_dlharq()

说明：

1. 执行harq的相关函数定义如下：

INT32 **do\_dlharq**(UINT16 rnti, UINT8 ndi, UINT32 tb\_size, UINT8 \*pdu\_p, TbHarqParam \*tbharqparam\_p)

INT32 **do\_newtran\_dlharq**(DlHarqInfo \*dlharq\_p, UINT32 tb\_size, UINT8 \*pdu\_p,TbHarqParam \*tbharqparam\_p) 根据调度信息

INT32 **do\_retran\_dlharq**(DlHarqInfo \*dlharq\_p, UINT8 \*pdu\_p, TbHarqParam \*tbharqparam\_p)

INT32 **create\_sf\_timer**(SFTIMERFUNC \*func\_p, INT32 arg, UINT32 expires)创建帧定时器

INT32 **do\_wait\_timer\_expiration**(DlProcInfo \*proc\_p) 为进程帧定时器超时后执行的函数，函数将进程数据内pdu清空，并设置进程状态为重传。

### tx\_mac\_to\_lowmac()

说明：

1. INT32 **tx\_dlsch\_mac\_pdu**(DlschMacPduLst \*dlsch\_mac\_pdu\_lst\_p, UINT16 tx\_fn,UINT8 tx\_sub\_fn) 将 dlsch mac数据存入g\_data\_mem\_p[BUF\_NUM][USER\_NUM]指向内存，并转化为LowmacTxBuffer类型全局变量 g\_lowmac\_tx\_buffer[10] 类型数据(lowmac模块数据缓冲区)；
2. INT32 **tx\_ctrl\_msg**(UlGrantInfoLst \* ul\_grant\_info\_lst\_p, UINT16 tx\_fn, UINT8 tx\_sub\_fn) 将UlGrant信息存入g\_ctrlmsg\_mem\_p[BUF\_NUM][USER\_NUM]指向内存，并转化为LowmacTxBuffer类型全局变量 g\_lowmac\_tx\_buffer[10] 类型数据(lowmac模块数据缓冲区)；。

## 数据结构

### 物理资源数据结构(lowmac层定义)

/\* DlPhyResInfo用于保存物理资源信息\*/

typedef struct {

DlschResLst dlsch\_res\_lst;/\*链表，子结点结构为DlPhyResNode \*/

PchResInfo pch\_res\_info;

BchResInfo bch\_res\_info;

}DlPhyResInfo;

/\* DlPhyResNode 用于保存表示dlsch物理资源dlsch\_res\_lst的子结点\*/

typedef struct {

NodeType ln;

UINT32 rb\_start;

UINT32 rb\_len;

}DlPhyResNode;

/\* BchResInfo保存Bch物理资源信息\*/

typedef struct {

UINT32 rb\_start;

UINT32 rb\_len;

} BchResInfo;

/\* PchResInfo保存Pch物理资源信息\*/

typedef struct {

} PchResInfo;

### 调度数据结构

1. 调度结果

/\* DlSchedResult用于保存调度结果\*/

typedef struct {

DlSchedSchLst sch\_result;

SchedPchResult pch\_result;

SchedBchResult bch\_result;

} DlSchedResult;

/\* SchedPchResult 保存调度得到的Pch资源\*/

typedef struct {

UINT16 rb\_start;

UINT16 rb\_len;

UINT8 mcs;

UINT32 data\_len;

} SchedPchResult;

/\* SchedBchResult 保存调度得到的Bch资源\*/

typedef struct {

UINT16 rb\_start;

UINT16 rb\_len;

UINT8 mcs;

UINT32 data\_len;

} SchedBchResult;

typedef struct {

NodeType ln;

LchResultLst lch\_result\_lst;

UINT32 size;

UINT16 rb\_start;

UINT16 rb\_len;

UINT16 rnti;

UINT8 mcs;

UINT8 ndi;

} DlSchedSchResultNode;

1. 下行调度队列

typedef struct {

CrntiDlSchedQ c\_queue; /\*CRNTI queue\*/

TcrntiDlSchedQ tc\_queue; /\*TCRNTI queue\*/

RarntiDlSchedQ ra\_queue; /\*RARNTI queue\*/

}DlSchedQueue;

/\* DlSchedQueue.c\_queue 列表指向结点 \*/

typedef struct {

NodeType ln;

UINT32 priority;

LcSchedQ lc\_queue;

UINT32 buf\_sz;

UINT16 rnti;

UINT8 harq\_ndi;

RntiType rnti\_type;

} DlSchedQueueNode;

### PDU构造数据结构

typedef struct {

LcRlcPduList lc\_rlc\_pdu\_lst\_p;

CeList ce\_lst\_p;

UINT16 rnti;

} InputData;

typedef struct {

ListType lh;

UINT32 data\_size;

} LcRlcPduList;

typedef struct {

NodeType ln;

LchType lc\_type;

TxPduList pdu\_list;

UINT32 pdu\_num;

UINT32 data\_size;

UINT8 lc\_id;

UINT8 reserved[3];

} LcRlcPduNode;

typedef struct {

ListType lh;

UINT32 byte\_size;

} CeList;

typedef struct {

NodeType ln;

UINT8 \*ce\_msg\_p;

UINT16 ce\_msg\_len;

UINT8 lcid;

UINT8 reserved;

} CeNode;

### LowMAC发送数据结构

typedef struct {

DlschMacPduLst dlsch\_pdu\_lst;

UlGrantInfoLst ul\_grant\_info\_lst;

PchMacPdu pch\_mac\_pdu;

BchMacPdu bch\_mac\_pdu;

AckLst \*ack\_lst\_p;

UINT16 tx\_frame\_num;

UINT8 tx\_subframe\_num;

} MacToLowMac;

**typedef** ListType DlschMacPduLst;

typedef struct {

NodeType ln;

DciInfo dci\_info;

PduData mac\_pdu;

} DlschMacPduNode;

typedef struct {

NodeType ln;

UINT32 size;

UINT32 rb\_start;

UINT32 rb\_len;

UINT16 rnti;

UINT8 mcs;

UINT8 ndi;

UINT8 rv;

} UlGrantInfoNode;

typedef struct {

UINT32 rb\_start;

UINT32 rb\_len;

UINT16 rnti;

TbHarqParam harq\_param;

UINT8 mcs;

UINT8 ndi;

} DciInfo;

typedef struct {

UINT32 data\_len;

UINT8 \*data\_addr;

} PduData;

typedef struct {

UINT32 data\_size;

UINT32 ctrlmsg\_size;

UINT8 \*data\_buffer\_p;

UINT8 \*ctrlmsg\_buffer\_p;

} LowmacTxBuffer;

### 随机接入相关数据结构

typedef struct {

NodeType ln;

UINT16 bytes\_size; /\* total size of all rar node \*/

UINT16 ra\_rnti; /\* ra-rnti value \*/

ListType rar\_node\_lst; /\* 列表中结点为 RarNode 类型\*/

UINT8 backoff\_flag; /\* no backoff\_ind exist backoff\_ind \*/

UINT8 backoff\_ind; /\* backoff indication value \*/

}RaRntiMsgNode;

typedef struct {

NodeType ln;

RarInfo \*rar\_msg\_p; /\* Pointer to RAR message of the RA-RNTI \*/

UINT8 rapid; /\* The RAPID of the RAR message \*/

} RarNode;

typedef struct {

RARUlGrant \*ul\_grant\_p; /\* pointer to schedule result \*/

INT32 ta; /\* time alignment value \*/

UINT16 tc\_rnti; /\* temporary rnti sent to construct rar pdu \*/

} RarInfo;

### HARQ相关数据结构

**typedef** **struct** {

DlProcInfo \*dlproc\_p; /\* pointer to the first process of this RNTI\*/

INT32 ack\_waiting\_time;/\* the time of waiting ACK \*/

UINT8 total\_proc\_num; /\* the total number of processes used \*/

UINT8 retran\_tb\_size; /\* the total size of TB in all retransimission\*/

UINT8 retran\_proc\_num;/\* the number of retransimission process \*/

UINT8 idle\_proc\_num; /\* indicate the number of idle process \*/

UINT8 ndi;

UINT8 dl\_max\_harq\_tx; /\* the max number of retransimission \*/

UINT8 max\_sps\_proc\_num;/\* the max number of this UE DL HARQ process \*/

UINT8 reserved;

} DlHarqInfo;

**typedef** **struct** {

UINT32 dl\_tx\_time; /\* the TX time of the TB in this process \*/

UINT8 \*pdu\_p; /\* the pointer to the RLC pdu information list \*/

UINT32 tb\_size; /\* the TB size in Byte in this process \*/

UINT8 current\_tx\_num; /\* the tx times of DL data \*/

UINT8 rv; /\* RV \*/

UINT8 state; /\* process state:0 idle;1 wait;2 need re-TX \*/

UINT8 reserved;

} DlProcInfo;

typedef struct {

UINT8 process\_id; /\* process number for current TB \*/

UINT8 irv; /\* redundancy version for current TB \*/

} TbHarqParam;

# 其他说明

本文档基于2011.11.24日前ltehenb MAC层实现。

Pch/bch , bch下行调度结果总是一定？

Pch、Mch数据未处理？只处理pdsch信道发送数据？

新重传的设置只根据调度结果设置？

待进一步理解的数据变量

TDD\_MACTX\_TIMER\_CONF2[g\_sub\_frame\_num]

TDD\_MACTX\_SUBFRAME\_OFFSET\_CONF2[g\_sub\_frame\_num]

static FrameNumLst g\_frame\_num\_to\_mactx\_lst; /\*发送子帧信息链表\*/

# 附录LTE 协议栈的信道映射



图 .1 LTE 协议栈的信道映射