**TD-LTE Small Cell基站协议栈软件**

**使用说明书**

目录

[1. 系统设计 1](#_Toc353125078)

[1.1 系统概述 1](#_Toc353125079)

[1.1.1 Small cell介绍 1](#_Toc353125080)

[1.1.2 系统架构 1](#_Toc353125081)

[1.2 系统模块划分 3](#_Toc353125082)

[1.3 系统运行设计 6](#_Toc353125083)

[1.3.1 线程设计 6](#_Toc353125084)

[1.3.2 处理流程 8](#_Toc353125085)

[1.4 RRM状态机 10](#_Toc353125086)

[1.5 核心数据管理 12](#_Toc353125087)

[1.5.1 核心数据管理总体设计 12](#_Toc353125088)

[1.5.2 核心数据管理接口设计 15](#_Toc353125089)

[1.5.3 核心数据管理结构设计 18](#_Toc353125090)

[2.系统测试 29](#_Toc353125091)

[2.1 概述 29](#_Toc353125092)

[2.2 Uu口数据面联合测试 29](#_Toc353125093)

[2.2.1 测试目的与内容 29](#_Toc353125094)

[2.2.2 软硬件需求 30](#_Toc353125095)

[2.2.3 测试场景 30](#_Toc353125096)

[2.2.4 测试方案 31](#_Toc353125097)

[2.2.5 UM模式下测试介绍 31](#_Toc353125098)

[2.2.6 AM模式下测试介绍 32](#_Toc353125099)

[2.3 S1与Uu控制面联合测试 34](#_Toc353125100)

[2.3.1 测试目的与内容 34](#_Toc353125101)

[2.3.2 软硬件需求 35](#_Toc353125102)

[2.3.3 测试场景 35](#_Toc353125103)

[2.3.4 测试方案 35](#_Toc353125104)

[2.3.5 遗留问题 43](#_Toc353125105)

[2.4 X2AP控制面联合测试 43](#_Toc353125106)

[2.4.1 测试目的与内容 43](#_Toc353125107)

[2.4.2 软硬件需求 44](#_Toc353125108)

[2.4.3 测试场景 44](#_Toc353125109)

[2.4.4 测试方案 44](#_Toc353125110)

[2.4.5 遗留问题 53](#_Toc353125111)

[2.5 协议栈联合测试 53](#_Toc353125112)

[2.5.1 测试目的与内容 53](#_Toc353125113)

[2.5.2 软硬件需求 53](#_Toc353125114)

[2.5.3 测试场景 54](#_Toc353125115)

[2.5.4 测试方案 54](#_Toc353125116)

# 系统设计

## 1.1 系统概述

### 1.1.1 Small cell介绍

Small Cell是低功率的无线接入[节点](http://baike.baidu.com/view/47398.htm)，工作在授权的、非授权的频谱，可以覆盖10米到200米的范围，相比之下，宏蜂窝的覆盖范围可以达到数公里。移动运营商正在为增长的数据流量发愁，很多运营商认为分流移动数据是高效使用无线频谱资源的好办法。Small Cell是3G数据分流的重要成分，很多运营商认为Small Cell是管理LTE A频谱的有效办法，而不是只是使用宏蜂窝。

小基站具有如下特点，使得其应用必然越来越广泛。

1. 低成本：结构简单，与传统基站相比，价格低廉，用户可承担；
2. 安装简便：支持即插即用，用户可以自行安装Small cell终端，只需要运营商进行激活；
3. 宽带接入：基于IP协议，采用扁平化的基站架构，可以通过现有的DSL、cable或光纤等宽带手段接入移动运营商的网络；
4. 低功率：发射功率为10～100mW，与Wi-Fi接入点类似；
5. 基于蜂窝移动网络标准：基于任何移动蜂窝通信技术，包括现有的3G标准及即将成熟的UMB、LTE；
6. 支持多种标准化协议：独立于网络连接方式，支持连接运营商核心网的多种接口；
7. 自动配置、自动优化；

针对小基站的系统特点和实现特点，小基站可以有如下应用场景：

1. 基础应用场景，解决家庭，企业，地下通道等室内场所的信号覆盖问题；
2. 升级应用场景，完成应急通信组网，可以实现：
   * 更加灵活的网络规划、网络优化、低成本的网络补盲；
   * 更加快捷的灾后无线通讯恢复手段；
   * 应对会议、机会等突发性的容量变化，灵活进行补充覆盖，弥补通讯车覆盖局限，控制通信成本。

### 1.1.2 系统架构

LTE HeNB协议栈软件实现LTE系统中接入网功能，通过3GPP规范的UU接口与终端设备连接，通过3GPP规范的S1接口与HeNB Gateway或者直接与核心网连接。HeNB逻辑架构如图1.1所示。HeNB共包含三种组网方式：

1. HeNB GW作为HeNB与EPC的中继节点，HeNB通过S1接口与HeNB GW相连，HeNB GW通过S1接口接入EPC；
2. HeNB直接与EPC连接，HeNB通过S1-U接入S-GW，通过 S1-MME接入MME；
3. HeNB GW作为HeNB控制面的中继节点，HeNB通过S1-MME与HeNB GW相连，HeNB GW通过S1-MME接入MME；而HeNB通过S1-U直接接入S-GW。

不管HeNB与核心网连接过程中使用哪种接入方式，HeNB协议栈接口功能不会改变。



图1.1 E-UTRAN HeNB逻辑架构

根据3GPP LTE协议规定，暂不支持HeNB之间或HeNB与eNB间的X2接口连接。

ICT-TD-LTE HeNB协议栈软件系统根据3GPP LTE Uu接口以及 S1接口规范，完成各协议层功能，系统功能架构如图1.2所示：

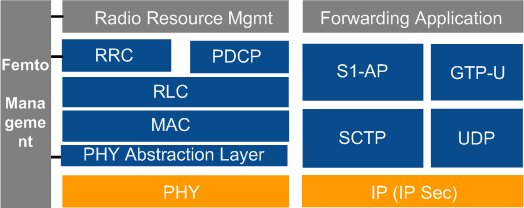


图1.2 LTE HeNB协议栈架构

其中，RRC层完成Uu口控制功能，PDCP/RLC/MAC完成Uu口L2层数据处理功能，物理层抽象层为MAC与PHY的接口，通过API完成MAC与PHY的交互；S1-AP完成S1接口控制功能，GTP-U完成S1接口数据传输功能；RRM完成对整个HeNB协议栈的无线管理，如接入控制，动态资源分配等。

## 1.2 系统模块划分

LTE HeNB协议栈系统采用层次化的设计模型，将各个协议层次作为单独的子系统实现，各个子系统之间通过预定义的APIs实现交互，系统采用“分层”、“高内聚，低耦合”的理念结合跨层优化，平台无关性等技术，满足LTE标准对基站协议栈软件的要求。ICT-LTE HeNB系统模块划分如图1.3所示：

图1.3 LTE HeNB系统模块划分

1. ICT-LTE HeNB协议栈软件子系统

主要完成Uu接口与S1接口各协议层功能，包括：

1. RRC子系统

Uu口控制平面处理中心，完成空中接口部分控制面RRC连接管理、系统信息广播、寻呼、移动性管理功能、测量管理以及ASN.1消息编解码等功能。

1. PDCP子系统

协议栈L2层数据处理的最高层，完成PDCP层协议规定的数据收发和递交，PDCP层用户数据和信令的加解密、控制面信令完整性保护以及数据面头压缩功能。

1. RLC子系统

逻辑链路控制层，通过对TM、UM、AM模式的数据收发的支持，完成对不同QoS参数的RLC数据传输功能。

1. MAC子系统

完成上下行数据调度、随机接入、MAC PDU数据收发、MAC层HARQ管理以及MAC层控制信息的处理等功能。

1. S1-AP子系统

完成S1接口控制平面功能，包括E-RAB管理、UE上下文管理、连接状态移动性管理、寻呼、NAS消息传输等功能。

1. L1-API子系统

主要完成协议栈与物理层进行数据交互时的编解码过程。

1. ICT-LTE HeNB协议栈管理子系统
2. RRM子系统

RRM子系统为整个软件系统的控制中心，通过RRM状态机，根据Uu口以及S1接口的消息以及当前协议状态，触发空口RRC层以及S1接口S1-AP不同过程的执行。完成无线承载控制、无线准入控制、功率控制、拥塞控制、移动性管理、无线资源分配等功能。

1. 协议栈管理子系统

协议栈管理子系统完成对整个ICT-LTE HeNB软件系统的初始化、各层配置以及错误处理等功能。

1. ICT-LTE HeNB协议栈支撑子系统

基站设备集成、制造商部署协议栈的重要不利因素在于该类系统软件的软硬件依赖性。为此，大量的成本被用于针对特定硬件平台和底层操作系统服务平台的软件修改。ICT-LTE协议栈支撑子系统实现协议栈软件系统的平台无关性，通过虚拟环境的技术将硬件和底层服务例程与协议栈核心部分透明的隔离开来，使得协议栈系统核心部件透明的访问操作系统特定接口。

1. 物理层抽象子系统

使用PHY适配器屏蔽底层硬件平台和物理层实现的细节，使ICT-LTE HeNB协议栈核心部分能透明地使用硬件平台和物理层所提供的服务。

1. 操作系统抽象子系统

使用OS适配器屏蔽不同操作系统的实现细节，使ICT-LTE HeNB协议栈核心部分能透明地使用操作系统的相关操作。

1. 网络传输子系统

网络传输子系统需满足无线网络层协议所需要的网络传送服务，通过封装底层SOCKET通信细节，为无线网络层提供传输服务。网络传送模块将利用SELECT机制实现IO多路复用，同时管理SCTP socket和GTP-U相关sockets，并对SCTP协议、GTP-U协议提供一个抽象的、简单的应用接口。GTP-U需要参考3GPP TS 29.281。GTP-U采用Tunnel Endpoint ID复用的方式，因此，一个GTP-U实例便足够了(一个IP地址一个GTP-U 实体)。GTP-U需要采用全双工UDP，以防部分数据丢失。网络传输层将同时支持IPV4和IPV6双协议栈。

## 1.3 系统运行设计

### 1.3.1 线程设计



图1.4 LTE HeNB协议栈软件系统线程关系图

LTE HeNB协议栈系统线程模型如图1.4所示，系统共**7个线程，4个数据缓冲区**，分别完成数据收发以及控制信息处理过程，具体描述如下：

**线程1：Uu发送线程**

当获得发送信号量g\_uu\_tx \_sem后，启动Uu口数据发送过程，MAC调度模块决定每个UE对应的逻辑信道上可以发送的数据后，调用RLC模块提供的接口，获取RLC PDU并组装成MAC TB调用L1api发送函数通过PHY发往Uu；【from 1ms中断、pdcp\_txmq】【to Uu】

对于MAC层提前调度的说明，MAC层获取调度机会时将准备后续N帧要发送的数据，并调用L1api消息构造函数，准备发送给PHY的数据（该数据中包含SFN/SF字段，DCI，对应的下行数据等）。

**线程2：Uu接收线程**

该线程循环检查消息队列g\_uu\_rx\_msgq，如果收到消息，则启动Uu数据接收过程，通过MAC接收、RLC接收以及PDCP接收函数，完成数据解包操作；【from 接收中断】

* 对于控制信息(RRC消息)，通过调用RRC接收模块的接口，放入RRM消息队列（rrm\_rxmq），待RRM进行处理；【to rrm\_rxmq】
* 对应数据(IP包)，PDCP完成数据解析后，放入S1数据发送消息队列（cndata\_txmq），待S1接口Gtp-u进行处理；【to cndata\_txmq】

**线程3：PDCP数据处理线程**

该线程检查PDCP数据发送消息队列（pdcp\_txmq）：

如果有来自网络GTP-U数据端口的IP数据，则该线程对缓存中的IP数据包进行处理（头压缩、加密、完整性保护等），并构造为PDCP PDU，放入RLC缓存，待下层处理；

如果有来自RRM调用的下行RRC消息发送，则进行PDCP层相关处理，放入RLC缓存，等待PHY发送时机。

**线程4：接收核心网数据线程**

当S1接口从网络SCTP端口或GTP-U端口监听到有数据接收时，启动S1数据接收过程；【from CN】

* 若是SCTP端口数据，则调用S1AP数据接收模块提供的接口，放入RRM消息队列（rrm\_rxmq），待RRM控制线程进行处理；【to rrm\_rxmq】
* 若是GTP-U端口数据，则调用GTP-U数据接收接口接收IP数据，并放入PDCP数据发送消息队列（pdcp\_txmq），待PDCP数据处理线程进行处理；【to pdcp\_txmq】

**线程5：SCTP发送线程**

用于发送S1-MME接口的信令，该线程检查S1数据发送消息队列（sctp\_txmq）：

* 当有S1AP要发送给核心网的消息时，该线程读取缓存中的数据，并通过对应的SCTP端口发往核心网；

**线程6：GTP-U数据发送线程**

用于发送S1-U接口的数据，该线程检查S1数据发送消息队列（cndata\_txmq）：

* 当有来自Uu接收线程处理完成的数据时，通过对应的GTP-U端口发往核心网；

**线程7：RRM控制线程**

RRM控制线程通过监听RRM消息队列（rrm\_rxmq），控制Uu口RRC层以及S1接口S1AP的行为；

* 当获取RRC消息时，通过调用RRC子系统提供的过程处理接口进行处理；
* RRM在处理完上行RRC消息后，如果需要，构一个下行RRC响应消息，并挂到PDCP数据发送消息队列（pdcp\_txmq）
* 当获取到S1AP消息时，通过调用S1AP子系统提供的过程处理接口进行处理，如果需要向核心网回复响应消息，则调用相应接口构造消息，并将构造好的消息放入S1数据发送消息队列（sctp\_txmq）。

Note：polling中断的线程待讨论，看是由MAC线程或者其他线程来执行；

另外RRM控制线程polling 中断端口，中断发生时进行相应的处理。

1. 1ms 帧中断

该中断到来时进行以下处理：

* + 1. 调用l1api\_recv\_msg函数接收PHY消息；
    2. 解析消息内容更新本地SFN变量；
    3. 调用超时处理函数（这里考虑到状态机可能用SFN来定时）
    4. post下行用户面发送信号量g\_uu\_tx \_sem
    5. other

1. UU上行数据接收中断

该中断到来时进行以下处理：

1. 调用l1api\_recv\_msg函数接收PHY消息；
2. 解析接收到的消息并放到g\_uu\_rx\_msgq消息队列（此处可以考虑收到消息后直接调用mac->rlc->pdcp/rrc处理消息）
3. 控制面响应消息中断

该中断到来时进行以下处理：

1. 调用l1api\_recv\_msg函数接收PHY消息；
2. 解析接收到的消息并放到g\_l1\_cplan\_resp\_msgq消息队列（此处可以考虑收到响应消息后直接进行处理）
3. S1系统中SCTP与GTP-U数据发送线程是否合并，根据实际系统测试性能决定；

### 1.3.2 处理流程

LTE HeNB子系统通过上述7个线程，完成Uu口与S1接口控制信令的处理以及数据的收发过程。

1. 信令交互过程

当Uu口协议栈接收到来自UE的RRC信令消息后，RRC接收模块调用ASN.1消息编解码器对消息进行解码并将解析后的消息放入RRM消息队列，RRM控制线程监听到RRC消息后，通过状态机进行判断，调用RRC子系统提供的相应过程处理函数进行处理，若需要构造RRC消息发送给UE，则通过RRC发送模块将对应的SRB0/SRB1/SRB2信令递交给RLC或PDCP待发送，处理完成后，通过状态机对UE的状态进行跳转并触发其他对应过程的执行；

当S1接口协议栈收到核心网的S1消息后，S1AP消息接收模块通过调用ASN.1消息编解码器对消息进行解码，并将解析后的消息放入RRM消息队列，RRM控制线程监听到S1AP消息后，通过状态机进行判断，调用S1AP子系统提供的相应过程处理函数进行处理，若需要构造S1AP消息发送给MME，则通过S1AP发送模块将对应的S1AP信令发送，处理完成后，状态机对UE状态进行跳转，并控制其他对应过程的执行。

1. 上行数据接收过程

当Uu口协议栈接收到来自物理层的接收中断后，启动数据接收过程，Uu接收线程从PHY读取接收的上行数据TB，依次调用MAC接收、RLC接收以及PDCP接收函数，对数据进行解包，并放入S1数据发送缓存；

S1发送线程通过监听S1数据发送缓存信息，当有新的IP数据包到达时，启动数据发送过程，根据该数据对应的RNTI/RBID寻找对应的UE以及GTP-U TEID，将对应的上行IP数据包发送给S-GW；

1. 下行数据发送过程

当S1接收线程监听到GTP-U端口监听到有数据到达时，启动下行数据接收过程，根据GTP-U TEID找到对应的UE/RBID，并将接收的IP数据包放入PDCP数据接收缓存；

PDCP数据处理线程监听到PDCP数据接收缓存有新数据到达时，启动PDCP数据处理过程，经过头压缩、加解密将对应的IP数据包构造成PDCP PDU，并放入RLC的数据缓存；

当Uu口协议栈接收到来自物理层的数据发送中断后，启动下行数据发送过程，通过MAC调度模块首先决定每个UE对应的逻辑信道上可以发送的数据，MAC发送模块启动RLC数据发送过程，将构造好的RLC PDU构造成MAC PDU，递交给物理层通过空口发送。

## 1.4 RRM状态机

LTE HeNB协议栈软件系统RRM状态机以UE为索引，通过对Uu口以及S1接口信令过程的处理，完成对UE的管理。根据UE入网注册、解注册流程(图1.5)，以及该过程中异常处理过程，设计RRM状态机。

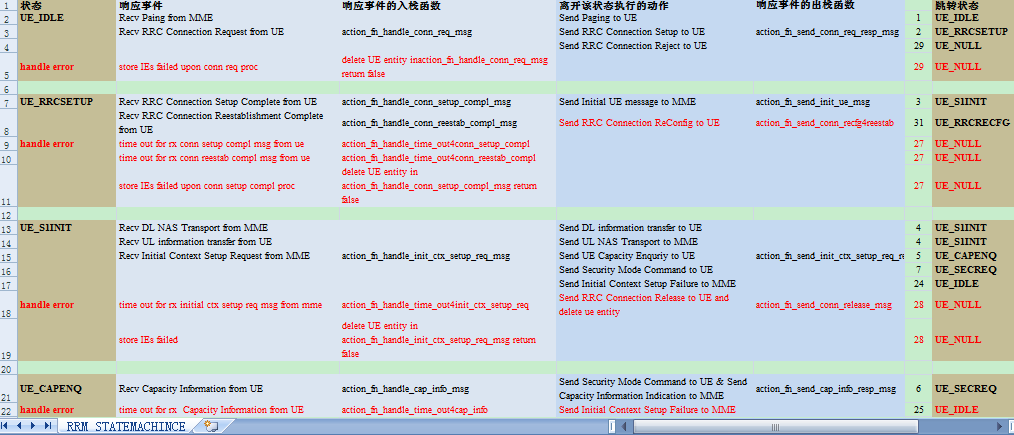
RRM状态转移图设计如图1.6所示。根据状态转移图，表1.1详细描述了RRM状态机的实现机制，分别对各状态下，所响应的事件，对应的跳转状态。



图1.5 终端入网注册/解注册

图1.6 LTE HeNB RRM状态转移图

表1.1 LTE HeNB RRM状态表



（见SM table.xls）

## 1.5 核心数据管理

### 1.5.1 核心数据管理总体设计

核心数据管理模块所执行的功能就是将与核心数据结构相关的许多复杂而繁琐的操作封装成若干个接口供其他子系统调用，这样一来可以避免其他子系统在实现过程中对核心数据结构组织形式的过多考虑，二来可以避免当核心数据结构的组织形式发生重大改变时对其他子系统的影响。

在模块概述中已经阐述了核心数据模块所需完成的功能，根据对核心数据管理模块的功能要求，我们将核心数据管理模块分为5个子模块：RB查询模块、RB创建模块、RNTI查询模块、RNTI创建模块、初始化/清理模块。

其中初始化模块由于初始化各种全局变量。

RB创建模块用于创建RB级别的实体。

RB查询模块用于查询RB/LC级别的信息

RNTI创建模块用于创建UE实体

RNTI查询模块用于查询UE相关信息

1. 系统初始流程

现有系统中，核心数据模块初始化时只需初始化RNTI相关数据结构。



图1.7 核心数据模块初始化流程图

* 1. UE/RNTI创建流程

现有系统在UE创建时，只有SRB0可以初始化，其他参数的初始化需等相关模块添加。



图1.8 UE/RNTI实体创建流程

* 1. RB/LC创建流程

此处所指的RB/LC创建流程是指除了SRB0-LC0外的其他RB-LC的创建。另外，在现有系统中，默认RB的RB ID和RB所对应的LC的LC ID从数值上相等。如下图所示。



图1.9 RB/LC创建流程

### 1.5.2 核心数据管理接口设计

1．函数描述

|  |  |  |
| --- | --- | --- |
| **函数名** | **功能** | **子模块：部分** |
| get\_lcid\_from\_rbid | 取得指定UE的指定无线承载对应的逻辑信道的LC ID | RB查询模块 |
| get\_rbid\_from\_lcid | 取得指定UE的指定逻辑信道对应的无线承载的RB ID | RB查询模块 |
| get\_rlc\_mode\_from\_lcid | 取得指定UE的指定逻辑信道中的RLC实体的传输模式（成功情况下） | RB查询模块 |
| get\_rlc\_mode\_from\_rbid | 取得指定UE的指定逻辑信道中的RLC实体的传输模式（成功情况下） | RB查询模块 |
| get\_rlc\_entity | 取得指定UE的指定逻辑信道中的RLC实体的指针 | RB查询模块 |
| get\_rb\_type | 取得指定UE的指定无线承载的类型（成功情况下） | RB查询模块 |
| get\_pdcp\_entity | 取得指定UE的指定无线承载中的PDCP实体的指针 | RB查询模块 |
| get\_lch\_entity | 取得指定UE的指定LC ID对应的逻辑信道实体指针 | RB查询模块 |
| get\_rb\_entity | 取得指定UE中指定RB ID对应的RB实体的指针 | RB查询模块 |
| get\_rnti\_type | 根据RNTI值来返回RNTI的类型,类型分为RARNTI，SPSCRNTI和CTCRNTI | RNTI查询模块 |
| get\_rnti\_entity | 从Henb系统中查找到RNTI值对应的RNTI实体的指针 | RNTI查询模块 |
| get\_ue\_state | 从系统中得到rnti对应的UE的状态信息 | RNTI查询模块 |
| create\_rb\_entity | 在RRC决定创建一个RB实体时，根据RRC给予的rb\_config\_node\_p解析出RB的各种参数，再根据这些参数创建RB实体、LCH实体、PDCP实体和RLC实体等，并且将这些实体的指针挂在相应的索引位置中 | RB创建模块 |
| delete\_rb\_entity | 删除RB实体以及该RB实体相对应的LCH实体、PDCP实体、RLC实体等实体，释放这些实体对应的内存空间 | RB创建模块 |
| modify\_lch\_sched\_param | 根据给出的参数修改UE中某个逻辑信道的调度参数 | RB创建模块 |
| re\_establish | 按照RRC命令对某个UE进行相应的重建立过程 | RB创建模块 |
| get\_available\_c\_rnti | 从保存未使用RNTI值的队列数组中获取当前可用而且未用的C-RNTI或者Temp-C-RNTI的值 | RNTI创建模块 |
| recycle\_c\_rnti | 回收C-RNTI或者Temp-C-RNTI | RNTI创建模块 |
| get\_available\_sps\_rnti | 从管理可用的RNTI值的队列数组中获取当前可用而且未用的SPS-RNTI的值 | RNTI创建模块 |
| recycle\_sps\_rnti | 回收SPS-CRNTI | RNTI创建模块 |
| create\_ue\_entity | 根据给出的RNTI值创建一个普通的RNTI实体，并且初始化这个RNTI实体，然后将这个RNTI放置在RNTI指针数组和RNTI链表之中 | RNTI创建模块 |
| turn\_tc2c\_rnti | 该rnti标示的UE成功接入后， 将t-c-rnti变为c-rnti；将UE实体从g\_tmp\_rnti\_ue\_lst中删除，链接到g\_c\_rnti\_ue\_lst中 | RNTI创建模块 |
| turn\_c2sps\_rnti | UE开始半静态调度时，将其C-RNTI保存，使用当前的SPS-RNTI作为标示进行数据传输，将UE实体从g\_c\_rnti\_ue\_lst转移到g\_sps\_rnti\_ue\_lst | RNTI创建模块 |
| turn\_sps2c\_rnti | UE结束半静态调度时，将其SPS-RNTI回收，恢复使用之前的C-RNTI作为标示进行数据传输，将UE实体从g\_sps\_rnti\_ue\_lst转移到g\_c\_rnti\_ue\_lst | RNTI创建模块 |
| delete\_ue\_entity | UE与HeNB断开连接后删除该UE实体，内存释放C-RNTI回收 | RNTI创建模块 |
| suspend\_ue | 在某些重建立情况下，将某个UE的所有RB进入悬停状态 | RNTI创建模块 |
| resume\_ue | 在某些重建立情况下，将UE的某些RB/LCH退出悬停状态 | RNTI创建模块 |
| init\_rnti\_mgmt | 初始化各个RNTI指针数组、负责RNTI号管理的队列数组，以及与配合这些数组的由于标识的全局变量 | 初始化模块 |
| cleanup\_rnti\_mgmt | 清理所所有RNTI链表、清理所有RNTI指针数组、还原所有管理RNTI号的队列数组以及相对应的全局变量 | 初始化模块 |

1. 函数调用关系

本系统所有函数均提供外部模块作为接口存在，存在调用关系的是：

查询模块：



图1.10 RB查询模块和RNTI查询模块之间的调用关系

如图所示，查询模块是逐级调用的，底层的查询函数调用上层的查询函数找到上层实体，再在上层实体中查询到所需信息。

创建模块需随着其他模块的添加而改进，暂无相互调用关系。

### 1.5.3 核心数据管理结构设计

在基站侧，核心数据结构以UE为单位进行组织，用于存储UE上下文信息，包括数据、管理消息和各种控制信息，供各模块全局访问，在Uu口通过RNTI索引，在S1接口，通过eNB UE S1APID进行索引，完成对UE的管理。

当一个UE与基站建立空口连接之后，基站将为该UE建立一个UeRntiMsg实体，存放UE上下文信息，并通过指针数组及链表两种交叉索引方式对UE实体进行管理，其中，数组索引可以通过CRNTI/eNB UE S1APID快速定位具体UE，从而获取UE信息；链表索引方式用于MAC调度时，对存在的UE进行遍历。

根据不同状态的UE所使用的CRNTI的不同，共有3个不同的RNTI链表：UeRntiMsgLst，TmpRntiMsgLst，SemiRntiMsgLst，分别用来链接以CRNTI，TC-RNTI，SEMI-RNTI为标志的UE。

其中，g\_c\_tc\_msg\_table用以直接索引以C\_RNTI或者TC\_RNTI为标记的终端，使用该表进行索引时，数组下标+ MIN\_C\_RNTI = 待索引的C\_RNTI；g\_sps\_ue\_table用以直接索引以SPS\_RNTI为标记的终端，使用该表时，数组下标+ MIN\_SPS\_C\_RNTI = 待索引的SPS\_RNTI。一个配置了SPS功能的终端，同时既有SPS\_RNTI和C\_RNTI标识，但其使用SPS功能时，只能属于SemiRntiMsgLst索引链表；SPS配置取消后，又由UeRntiMsgLst索引。

当UE与核心网建立S1连接后，基站会为UE分配一个唯一的eNB UE S1APID标识，在S1接口，通过该标识对UE相关的核心数据信息进行索引。由于eNB UE S1APID的取值范围为0…224-1，为了方便管理，eNB UE S1APID采用与CRNTI相同的数值。当UE与核心网建立好S1连接后，基站会为每个UE分配一个唯一的eNB UE S1APID，基站端收到核心网的数据后，通过eNB UE S1APID首先找到对应的CRNTI值，然后根据CRNTI索引到UE对应的核心数据信息。



图1.11 UE索引结构

每个UE对应的UeRntiMsg核心数据信息以RNTI/eNB UE S1APID为标示，需要通过该RNTI进行索引的信息均保存在该结构中。核心数据包含如下内容：

表1.2 UE上下文核心数据结构信息

|  |  |  |  |
| --- | --- | --- | --- |
| **UE上下文信息(UeRntiMsg)** | | | **Description** |
| RntiType rnti\_type;  RntiType pre\_rnti\_type;  UINT16 rnti\_value;  UINT16 pre\_rnti\_value; | | | UE与eNB空口间的唯一标示符，记录UE的RNTI值以及类型； |
|  | | |  |
| UeState ue\_state; | | | 记录UE当前状态，状态机根据UE当前状态以及事件进行相应操作； |
| UeId ue\_id; | | | UE临时网络标示符信息，40bit； |
|  | | |  |
|  | | |  |
| UEAggregateMaximumBitrate\_t ueaggregatemaximumbitrate; | | | 定义UE非GBR承载在上、下行链路的最大比特速率； |
| UESecurityCapabilities\_t uesecuritycapabilities; | | | UE安全能力，记录UE支持的加解密、完整性保护算法列表； |
| UeSecurityParams ue\_security\_params; | | | UE安全上下文信息，包括Kenb、加解密算法、完整性算法、用户面数据加解密密钥、控制面数据加解密密钥、完整性保护密钥； |
| TraceActivation\_t \*traceactivation\_p; | | | UE跟踪功能相关信息； |
| HandoverRestrictionList\_t\*handoverrestrictionlist\_p; | | | UE切换限制列表； |
| UERadioCapability\_t \*ueradiocapability\_p; | | | UE无线接入能力信息； |
| SubscriberProfileIDforRFP\_t\*subscriberprofileidforrfp\_p; | | | RAT或Frequency选择优先级信息； |
| CSFallbackIndicator\_t \*csfallbackindicator\_p; | | | CS Fallback信息； |
| SRVCCOperationPossible\_t \*srvccoperationpossible\_p; | | | 表示UE和MME在单射频情况下对于语音功能切换支持的能力； |
| CSGMembershipStatus\_t \*csgmembershipstatus\_p; | | | 表示UE是否属于某一给定的CSG； |
| void \*s1\_entity\_p; | UINT32 mme\_ue\_s1ap\_id; | | MME侧在S1接口用于唯一标示UE的标示符； |
| UINT32 enb\_ue\_s1ap\_id; | | eNB侧在S1接口用于唯一标示UE的标示符； |
| UINT32 sctp\_stream\_id; | | UE在S1接口SCTP连接信息； |
| UINT32 mme\_connection; | | UE对应的MME信息； |
| ListType erab\_tobe\_setup\_lst;  ListType erab\_tobe\_mod\_lst;  ListType erab\_tobe\_release\_lst;  ListType erab\_fail\_lst;  erab\_t \*e\_rab\_info[MAX\_E\_RAB\_NUM]; | | E-RAB相关信息； |
| INT32 last\_rx\_s1\_msg; | | | 记录UE对应的上一条S1消息类型； |
| void \*rrc\_entity\_p; | | EstabCause estab\_cause;  NasMsg last\_nas\_msg; /Gmmei reg\_mme;  UINT32 rrc\_trans\_id;  UINT32 sel\_plmn\_id; | Uu口RRC实体相关信息； |
| RadioBearInfo \*radio\_bear\_info[RB\_NUM];  UlLchInfo \*ul\_lcid\_info[LCH\_NUM];  DlLchInfo \*dl\_lcid\_info[LCH\_NUM];  DlLchLst dl\_lch\_lst; | | | Uu口数据面管理信息，采用指针数组的方式对RB、上下行逻辑信道进行索引，同时采用链表的方式对下行逻辑信道进行索引； |
| CeList dl\_ce\_list;  void \*ul\_harq\_info\_p;  DlHarqInfo \*dl\_harq\_info\_p;  void \*ul\_sched\_param\_p;  void \*dl\_sched\_param\_p;  void \*dl\_sps\_param\_p;  void \*ul\_sps\_param\_p;  void \*measure\_msg\_p; | | | 记录UE MAC相关参数，包括CE信息，HARQ参数，动态调度参数，SPS调度参数，测量相关参数； |



图1.12 UE上下文索引关系图

1. **E-RAB管理**

在S1接口，需要对E-RAB进行管理，提供数组的索引方式，通过数组的下标，可以快速定位到具体的E-RAB。

* 保持E-RAB的QoS参数信息，供MAC调度模块使用；
* 记录E-RAB对于的GTP-U端口信息，用于与S-GW间传输上下行数据；
* 记录E-RAB对应的RB信息，当接收到核心网的数据后，通过查找对应的RB，将数据递交给Uu口对应的RB列表中；

typedef struct {

NodeType ln;

E\_RAB\_ID\_t e\_rab\_id;

E\_RABLevelQoSParameters\_t e\_rab\_level\_qos\_params;

/\* EPC GTP-U information \*/

TransportLayerAddress\_t tnl\_epc;

UNIT32 teid\_epc;

/\* Local GTP-U information \*/

TransportLayerAddress\_t tnl\_local;

UINT32 teid\_local;

/\* NAS PDU \*/

NAS\_PDU\_t \*nas\_pdu\_p;

/\* Radio Bearer ID and GTP TEID have one-to-one mapping \*/

UINT32 rb\_id;

} E\_Rab\_t;

1. **RB管理**

在Uu口，需要对无线承载数据进行管理，对于RB的管理，提供数组的索引方式，通过数组的下标，可以快速定位到具体的RB。

* PDCP通过该管理可以对接收到的IP数据包进行相应的处理，并通过LCID将处理后的数据提交给相应的RLC实体进行处理。
* PCDP通过该管理，可以将经过处理的上行数据通过EPS Bear ID提交给S1子系统从S1接口进行发送；
* 核心网数据到来时，可以通过该管理，将EPS Bearer上的数据映射到对于RB上待Uu口进行发送；

typedef struct{

RbType rb\_type;

void \*pdcp\_entity\_p; /\* This struct defined in PDCP subsystem \*/

UINT8 rb\_id;

UINT8 eps\_id;

UINT8 lc\_id;

UINT8 suspend\_flag;

} RadioBearInfo;

1. **逻辑信道管理**

对于逻辑信道的管理，提供指针数组和链表两种交叉索引方式，其中数组索引的目的是为了在知道LCID的情况下，快速定位到对应的LCID数据节点；链表索引的目的是为了在调度、获取数据的时候，更方便地对所有的LCID进行遍历。

* RLC通过该管理可以对PDCP提交的数据进行处理；并提供给MAC访问数据的接口；
* MAC通过该管理可以根据各逻辑信道的优先级进行调度信息，并通过RLC的接口访问数据；

/\* Uplink logic channel information \*/

typedef struct {

NodeType ln;

UINT8 lc\_id;

UINT8 rb\_id;

RlcTransMode rlc\_mode;

void \*rlc\_entity\_p; /\* This struct defined in RLC. It is different for TM/AM/UM\*/

} UlLchInfo;

/\* Downlink logic channel information \*/

typedef struct {

NodeType ln;

UINT8 lc\_id;

UINT8 rb\_id;

RlcTransMode rlc\_mode;

UINT32 byte\_size;

void \*rlc\_entity\_p; /\* This struct defined in RLC. It is different for TM/AM/UM\*/

void \*sched\_param\_p;

} DlLchInfo;

1. **UE状态信息**

/\*UE State\*/

typedef enum {

UE\_IDLE = 0, /\*UE not connected with eNB \*/

UE\_RRCSETUP = 1, /\* eNB tx RRC Connection Setup to UE \*/

UE\_S1INIT = 2, /\* eNB tx Init UE Message to MME\*/

UE\_CAPENQ = 3, /\* eNB tx UE Capacity Enquriy to UE \*/

UE\_SECREQ = 4, /\* eNB tx Security Mode Command to UE \*/

UE\_RRCRECFG = 5, /\* eNB tx RRC Connection ReConfig to UE\*/

UE\_SETUPED = 6 /\* UE connected with eNB and MME\*/

}UeState;

1. **UE ID信息**

数据包经过HeNB在核心网和空口进行传输时需要维持UE Id与RNTI之间的映射关系。

typedef struct InitialUE\_Identity {

InitialUE\_Identity\_PR present;

union InitialUE\_Identity\_u {

S\_TMSI\_t s\_TMSI;

BIT\_STRING\_t randomValue;

} choice;

/\* Context for parsing across buffer boundaries \*/

asn\_struct\_ctx\_t \_asn\_ctx;

} InitialUE\_Identity\_t;

typedef InitialUE\_Identity\_t UeId;

1. **MAC处理信息**

CE list 用于存放下行发送时需要处理的CE

/\*

\* CE list.

\*/

typedef struct {

ListType lh;

UINT32 byte\_size;

} CeLst;

/\* CE Node \*/

typedef struct {

NodeType ln;

UINT16 ce\_msg\_len;

UINT8 lcid;

void\* ce\_msg\_p;

} CeRntiNode;

1. **常量定义**

#define ERR\_NO\_MEM -2000

#define ERR\_FUNC\_PARAM -1999

#define ERR\_SYS\_PARAM -1998

#define ERR\_BIT\_PACK -1997

#define NUM\_ZERO 0

#define NUM\_ONE 1

#define CORE\_ALL\_RIGHT 0

/\*define about RNTI.\*/

#define MAX\_UE\_NUM 4

#define MAX\_RA\_RNTI\_NUM MAX\_UE\_NUM

#define MAX\_C\_RNTI\_NUM (MAX\_UE\_NUM \* 2)

#define MAX\_SPS\_RNTI\_NUM MAX\_UE\_NUM

#define MIN\_RA\_RNTI 0x0001

#define MIN\_C\_RNTI 0x003D

#define MIN\_SPS\_RNTI (MIN\_C\_RNTI + MAX\_C\_RNTI\_NUM)

#define MAX\_RA\_RNTI (MIN\_RA\_RNTI + MAX\_RA\_RNTI\_NUM - 1)

#define MAX\_C\_RNTI (MIN\_C\_RNTI + MAX\_C\_RNTI\_NUM - 1)

#define MAX\_SPS\_RNTI (MIN\_SPS\_RNTI + MAX\_SPS\_RNTI\_NUM - 1)

#define P\_RNTI 0xFFFE /\* Paging RNTI \*/第一版未加入，第二版内容

#define SI\_RNTI 0xFFFF /\* System information RNTI \*/同上

/\*define about RB/LCH.\*/

#define MAX\_RB\_NUM 32

#define SRB0\_LCID 0x00 /\*SRB0 LCID type\*/

#define SRB1\_LCID 0x01 /\*SRB1 LCID type\*/

#define SRB2\_LCID 0x02 /\*SRB2 LCID type\*/

/\* Following LCIDs are used in MAC RX PDU Processing Procedure\*/

#define MIN\_DTCH\_LCID 0x03 /\* Minimal identity of DTCH\*/

#define MAX\_DTCH\_LCID 0x0A /\* Maxmum identity of DTCH\*/

#define PADDING\_LCID 0x1F /\* Padding Logical channel \*/

#define DTCH\_LCID\_NUM (MAX\_DTCH\_LCID - MIN\_DTCH\_LCID + 1)

#define DL\_CCCH\_LCID 0x00 /\* Downlink CCCH LCID type \*/

#define DL\_MIN\_IDEN\_LCID 0x01 /\* Minimal DL identity Logical channel \*/

#define DL\_MAX\_IDEN\_LCID 0x10 /\* Maxmum DL identity Logical channel \*/

#define DL\_CONTEN\_RESOL\_LCID 0x1C /\* UE Contention Resolution Identity \*/

#define DL\_TIME\_ADVANCE\_LCID 0x1D /\* Timing Advance \*/

#define DL\_DRX\_CMD\_LCID 0x1E /\* DRX Command \*/

#define UL\_CCCH\_LCID 0x00 /\* Uplink CCCH LCID type \*/

#define UL\_MIN\_IDEN\_LCID 0x01 /\* Minimal UL identity Logical channel \*/

#define UL\_MAX\_IDEN\_LCID 0x10 /\* Maxmum UL identity Logical channel \*/

#define UL\_PHR\_LCID 0x1A /\* Power Headroom Report \*/

#define UL\_C\_RNTI\_LCID 0x1B /\* C-RNTI \*/

#define UL\_TRUNC\_BSR\_LCID 0x1C /\* Truncated Buffer Status Report \*/

#define UL\_SHORT\_BSR\_LCID 0x1D /\* Short Buffer Status Report \*/

#define UL\_LONG\_BSR\_LCID 0X1E /\* Long Buffer Status Report \*/

#define PADDING\_LCID 0x1F

/\* Padding Logical channel \*/

#define LC\_NUM (DL\_MAX\_IDEN\_LCID - DL\_MIN\_IDEN\_LCID + 1)

/\*Downlink Logical channel pool size \*/

#define MAX\_AMBR\_GROUPS\_NUM 4

#define LCH\_NUM 11

#define RB\_NUM 11

1. **全局变量定义**

/\* available c rnti and Temporary c rnti table.\*/

UINT16 g\_c\_rnti\_table[(2 \* MAX\_UE\_NUM + 1)];

/\* available semi-persistance rnti table.\*/

UINT16 g\_sps\_rnti\_table[MAX\_UE\_NUM + 1];

/\* header pointer of the g\_c\_rnti\_table.\*/

INT32 g\_c\_rnti\_head\_flag = 0;

/\* rear pointer of the g\_c\_rnti\_table.\*/

INT32 g\_c\_rnti\_rear\_flag = 2 \* MAX\_UE\_NUM;

/\* header pointer of the g\_c\_rnti\_table.\*/

INT32 g\_sps\_rnti\_head\_flag = 0;

/\* rear pointer of the g\_c\_rnti\_table.\*/

INT32 g\_sps\_rnti\_rear\_flag = MAX\_UE\_NUM;

/\* table for ue entire message, with rnti type c-rnti & t-c-rnti,

used to index a UE with the c-rnti.\*/

UeRntiMsg \*g\_c\_tc\_msg\_table[2 \* MAX\_UE\_NUM];

/\* table for ue entire message, with rnti type sps-c-rnti,

used to index a UE with the sps-rnti.\*/

UeRntiMsg \*g\_sps\_ue\_table[MAX\_UE\_NUM];

/\* global msg list for UE entity with rnti type c-rnti.\*/

CRntiUeLst g\_c\_rnti\_ue\_lst;

/\* global msg list for UE entity with rnti type t-c-rnti.\*/

TmpRntiUeLst g\_tmp\_rnti\_ue\_lst;

/\* global msg list for UE entity with rnti type sps-c-rnti.\*/

SemiRntiUeLst g\_sps\_rnti\_ue\_lst;

RlcConfig \*g\_default\_rlc\_config\_p[3];

LogicChanConfig \*g\_default\_logic\_chan\_config\_p[3];

# 2.系统测试

## 2.1 概述

根据LTE HeNB项目研发进度，TD-LTE HeNB协议栈软件系统测试共分为4个步骤：

1）在PC机环境下对Uu口数据面（L2层）进行联合测试，确保协议栈软件系统数据传输基本功能的正确性；

2）在PC机环境下对控制面RRC层以及S1AP信令进行联合测试，确保协议栈信令处理过程的正确性；

3）在双PC环境下对整个协议栈软件系统（Uu口+S1接口）进行联合测试，确保整个协议栈软性系统运行的正确性；

4）物理层联合测试，与PicoChip物理层平台进行联合调试，对系统性能进行评估与分析。

测试进度安排如表2.1所示。

以下从测试目的与内容、软硬件需求以及测试场景等方面分别介绍LTE HeNB软件系统各个阶段的测试方案。

**表2.1 测试进度表**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试阶段** | **开始时间** | **完成时间** | **备注** |
| Uu口数据面（L2层）联合测试 | 2011.10.10 |  |  |
| 控制面RRC层以及S1AP信令联合测试 | 2011.10.24 |  |  |
| 协议栈软件系统（Uu口+S1接口）联合测试 | 2011.11.7 |  |  |
| 物理层联合测试 | 2011.11.28 |  |  |

## 2.2 Uu口数据面联合测试

### 2.2.1 测试目的与内容

在与物理层平台集成前，Uu口数据面联合测试将重点测试数据面L2层数据收发功能，对L2各子层功能的正确性进行验证，包括PDCP数据传输、RLC UM与AM两种模式数据传输、MAC层随机接入、上下行调度等功能。

由于本阶段测试侧重功能性的验证，测试将在PC机上完成，暂不在开发板进行测试。

测试内容包括：

* HeNB L2协议栈数据收发过程，测试协议栈运行的正确性与稳定性；
* 测试UM模式下数据传输性能；
* 测试 AM模式下数据传输性能；

### 2.2.2 软硬件需求

1. **硬件需求**
   * PC两台
2. **软件需求**
   * 操作系统： Linux (Kernel 2.6.20)
   * 发行版本： CentOS 5.2
   * 编译环境： gcc-3.4.5-glibc-2.3.6
   * LTE HeNB协议栈软件系统一套
   * LTE Testing UE协议栈软件系统一套

### 2.2.3 测试场景

测试中，使用两台PC，分别执行LTE HeNB协议栈以及LTE Testing UE协议栈软件，用于测试Uu口L2层数据传输功能。测试时，在开发PC上构建应用程序产生IP数据包，并发送至PDCP层，经过协议栈处理后，在LowMAC通过Socket传输到目标PC。测试场景物理连接关系以及逻辑关系如图2.1所示。

图2.1 HeNB协议栈数据面测试场景

### 2.2.4 测试方案

1. 底层通信链路搭建

HeNB协议栈下行数据或者Testing UE的上行数据到达其对应虚拟物理层后，通过建立Socket连接，将数据发送至对端接收。

1. 底层时序模拟

在虚拟物理层，通过sim\_phy.c来模拟物理层功能，构建定时器模拟物理层给L2的TTI中断，通过信号量g\_sys\_sem触发lowmac线程的执行(触发MAC调度、MAC数据接收、发送过程)。

1. 系统初始化过程

系统运行时，首先进行系统初始化过程：在HeNB端以及Testing UE端分别建立一个或者多个RB，用于发送或者接收数据。

1. 数据收发模拟

HeNB协议栈L2层建立成功后，采用两种形式的产生IP数据包：

* 1. 构解包模块

负责在发送端构造具有随机性且可以重现的数据包，在接收端验证这些包在传输过程中的正确性。构解包模块由于自构包测试，在发送端，调用构包函数构造数据包（数据包的长度、内容均由数据包的序号sn和可修改的密钥共同决定）；在接收端，根据同样的密钥对数据包进行解析，判断数据包是否正确、接收是否连续。采用这样的复杂构包方法可以达到既能够重现场景又能够使数据包的参数具有随机性的特点，这两条对于模拟各种实际业务情况是很关键的。

* 1. 实际业务测试

通过VLC应用程序传输视频，测试UM模式下的数据传输功能；通过FTP文件传输测试AM模式下的数据传输功能；

1. MAC层功能测试

在UM以及AM模式下数据传输功能验证成功的基础上，进一步对MAC层功能进行测试，分别包括随机接入功能、HARQ功能、SPS功能等。

### 2.2.5 UM模式下测试介绍

1）测试场景

图2.2 HeNB协议栈数据面UM模式下测试场景

2）测试方案

根据图2.2，测试场景一共需要四台机器，分别两台windows，两台Linux。其中Win1代表VLC的发送端，负责将数据发送到协议栈，Win2代表VLC的接收端，负责从协议栈中实时的接收数据，然后再客户端上显示出来。而Linux1中运行HeNB的下行发送，其从特定端口监听到数据后，将数据递交到PDCP，然后从PDCP将数据灌入协议栈，通过整个下行的发送过程，到达虚拟的物理层，然后虚拟的物理层通过socket发送到UE(Linux2)，UE接收到数据后，按照UE的下行接收，将从虚拟物理层接收到的数据包进行逐层解析，最后递交到PDCP。最后PDCP处理后的数据再通过socket发送到一直在进行监听的VLC接收端（Win2）。

### 2.2.6 AM模式下测试介绍

1）测试场景

图2.3 HeNB协议栈数据面AM模式下测试场景

2)测试方案

由于AM是确认传输模式，一般用于质量要求较高的文件传输。在测试场景中使用FTP工具，而在Linux中进行ip包的抓取与回收。首先在准备好HeNB和UE两个协议栈之外，还需要两个内核的驱动程序，分别用于ip数据包的抓取与放回，这两个程序必须放到不同的内核为2.6.20-2.6.28的Linux操作系统中。

首先，需要在Win1 中添加路由route add 10.21.1.188 10.21.1.141，意思是目标为10.21.1.188的数据包下一个路由ip为10.21.1.141，而在Linux1中有这样一条路由route add 10.21.1.188 gw 192.168.0.2，所以当数据包的目的地址为10.21.1.188时，它的下一个路由ip为192.168.0.2，这样就寻找与路由同一网段的虚拟网卡192.168.0.1，而数据处理过程会在Driver\_tx中进行处理，处理完后，将数据发送到某一设定的本机端口A，然后HeNB协议栈监听此端口A，接收到数据后递交给PDCP,数据进入数据面协议栈，最后到达虚拟物理层，然后通过socket到达10.21.1.189机器上运行的UE协议栈，协议栈将数据包处理后到达DPCP，然后由PDCP再将数据递交到本机的某一设定端口B，Driver\_rx监听端口B的数据，收到数据后交给虚拟网卡192.168.0.2，然后再由路由递交到网上，最后到达FTP服务端Win2（10.21.1.188）。TCP的确认过程可以由Win2直接到达Win1，不需要走协议栈，而图中的上行数据是AM模式下的状态PDU。

3）注意事项

1. Linux升级内核时，一定要确定内核版本为2.6.20-2.6.28之间，其他版本的驱动无法安装，目前确定20和21的版本完全可行。

2. 升级版本时，make menuconfig一定要配置好，一个是cpu的型号，还有一个是频率

3. 我们这一批清华同方的机器好像是用的集成的网卡和显卡，升级内核时驱动安装不好，有可能需要自己进行驱动安装，在安装驱动之前一定要明确硬件的型号及安装方式。如果图形界面无法显示，可以先将/etc/X11/xorg.conf文件中 section “device” 中的driver “intel”改为driver “vesa”，这样设置以后可以有图形界面，但是分辨率只有两个，驱动没有安装好，在官网上也没有找到驱动，目前我也没有解决。网卡驱动可以查看网卡信息，然后下载并且安装，否则无法连通外网。

4. 驱动必须放在两台机器上，在bin目录下编译后，运行./ue.test, 运行完后要卸载模块，即rmmod xxx.ko

5. Win1必须和Linux1在同一网段，Win2必须和Linux2在同一网段。

## 2.3 S1与Uu控制面联合测试

### 2.3.1 测试目的与内容

在与Uu口数据面以及物理层硬件平台集成前，需要对控制面RRC层以及S1AP信令进行测试，通过Uu口、S1接口的信令交互，为UE建立、修改或者释放对应的RB。本阶段测试侧重对RRM状态机功能、RRC信令处理以及S1AP信令处理功能进行验证，在PC机上完成测试。

测试内容包括：

* 寻呼；
* RRC初始连接建立；
* 初始安全激活；
* RRC连接重配置；
* RRC连接释放；
* UE上下文建立、修改与释放；
* E-RAB建立、修改与释放；
* S1建立；

### 2.3.2 软硬件需求

1. **硬件需求**
   * PC两台
2. **软件需求**
   * 操作系统： Linux (Kernel 2.6.20)
   * 发行版本： CentOS 5.2
   * 编译环境： gcc-3.4.5-glibc-2.3.6
   * LTE HeNB协议栈软件系统一套
   * LTE Testing UE协议栈软件系统一套
   * LTE Testing GW协议栈软件系统一套

### 2.3.3 测试场景

测试中，使用两台PC，执行LTE HeNB协议栈，LTE Testing UE协议栈软件以及LTE Testing GW相关功能。测试时，在HeNB以及Testing UE协议栈RRC层打桩，Uu信令消息不通过L2层处理，直接通过Socket传输至对端。测试场景物理连接关系以及逻辑关系如图2.4所示。

图2.4 HeNB协议栈控制面测试场景

### 2.3.4 测试方案[[1]](#footnote-1)

#### 2.3.4.1 终端入网过程

表 2.2 终端入网成功与失败过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | | **Henb 🡨🡪SimMME（S1）** | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** | ***RRCConnectionRequest*** |  |  |  |
| ***RRCConnectionSetup*** | ***RRCConnectionReject*** |  |  |
| ***RRCConnectionSetupComplete*** | ***Timeout***  ***RRCConnectionSetupComplete*** | ***InitialUeMessage*** | ***InitialUeMessage***  ***（tx failed）*** |
| **2** | ***DlInfomationTransfer*** |  | ***DownlinkNasTransport*** |  |
| ***UlInfomationTransfer*** |  | ***UplinkNasTransport*** |  |
| **3** | ***UECapabilityEnquiry*** |  | ***InitialContexSetupRequest*** |  |
| ***UECapabilityInformation*** | ***Timeout***  ***UECapabilityInformation*** |  | ***InitialContexSetupFailure*** |
|  | ***no ue sec alg match with*** |  | ***InitialContexSetupFailure*** |
| **4** | ***SecurityModeCommand*** |  |  |  |
| ***SecurityModeComplete*** | ***Timeout***  ***SecurityModeComplete*** |  | ***InitialContexSetupFailure*** |
|  | ***SecurityModeFailure*** |
| **5** | ***RRCConnectionReconfiguration*** |  |  |  |
| ***RRCConnectionReconfigurationComplete*** | ***Timeout RRCConnectionReconfigurationComplete*** | ***InitialContexSetupResponse*** | ***InitialContexSetupFailure*** |

备注：

1. 成功入网过程：

**Henb :Rx Conn Setup Req🡪Tx Conn Setup🡪Rx Conn Setup Compl🡪Tx Init Ue Msg🡪鉴权和NAS安全🡪Rx Init Ctx Setup Req🡪Tx Ue Cap Req🡪Rx Ue Cap Info🡪Tx Sec Mod Cmd🡪Rx Sec Mod Compl🡪Tx Recfg🡪Rx Recfg Compl🡪Tx Init Ctx Setup Resp**

1. 失败入网过程：

* 信令失败
  + **Henb :Rx Conn Setup Req🡪Tx Conn Reject(Del Ue entity)**
  + **Henb :Rx Conn Setup Req🡪… …🡪Rx Init Ctx Setup Req(no ue security** [**algorithm**](app:ds:algorithm) **match)**
  + **Henb :Rx Conn Setup Req🡪… …🡪Rx Sec Mod Fail(send initial ctx setup failure)**
* 超时失败
  + **Henb :Rx Conn Setup Req🡪Tx Conn Setup🡪Timeout Rx Conn Setup Compl(Del Ue entity)**
  + **Henb :Rx Conn Setup Req🡪… …🡪Tx Ue Cap Req🡪 Timeout Rx Ue Cap Info(send initial ctx setup failure)**
  + **Henb :Rx Conn Setup Req🡪… …🡪Tx Sec Mod Cmd🡪 Timeout Rx Sec Mod Compl(send initial ctx setup failure)**
  + **Henb :Rx Conn Setup Req🡪… …🡪Tx Recfg🡪 Timeout Rx Recfg Compl(send initial ctx setup failure)**

#### 2.3.4.2 业务建立过程

表 2.3 业务建立过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | | **Henb 🡨🡪SimMME（S1）** | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** | ***RRCConnectionReconfiguration*** |  | ***ERABSetupRequest*** |  |
| ***RRCConnectionReconfigurationComplete*** | ***Timeout RRCConnectionReconfigurationComplete*** | ***ERABSetupResponse*** | ***ERABSetupResponse*** |

#### 2.3.4.3 业务修改过程

表 2.4 业务修改过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | | **Henb 🡨🡪SimMME（S1）** | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** | ***RRCConnectionReconfiguration*** |  | ***ERABModifyRequest*** |  |
| ***RRCConnectionReconfigurationComplete*** | ***Timeout RRCConnectionReconfigurationComplete*** | ***ERABModifyResponse*** | ***ERABModifyResponse*** |

#### 2.3.4.4 业务释放过程

表 2.5 业务释放过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | | **Henb 🡨🡪SimMME（S1）** | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** | ***RRCConnectionReconfiguration*** |  | ***ERABReleaseCommand*** |  |
| ***RRCConnectionReconfigurationComplete*** | ***Timeout RRCConnectionReconfigurationComplete*** | ***ERABReleaseResponse*** | ***ERABReleaseResponse*** |

备注：

1. ERAB成功流程：

* **Henb :Rx Erab Setup/Modify/Release Req🡪Tx Conn Recfg🡪Rx Conn Recfg Compl🡪Tx Erab Setup/Modify/Release Resp**

1. ERAB失败流程：

* **Henb :Rx Erab Setup/Modify/Release Req**🡪**Tx Conn Recfg**🡪**Timeout Rx Conn Recfg Compl(send Erab Setup/Modify/Release Resp)**
* **Henb :Rx Erab Setup/Modify/Release Req**🡪 **Tx Conn Recfg failed (add Erab node to failed lst && send Erab Setup/Modify/Release Resp)**

#### 2.3.4.5 UE上下文修改过程

表 2.6 UE上下文修改过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | | **Henb 🡨🡪SimMME（S1）** | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***UeContextModificationRequest*** |  |
|  |  | ***UeContextModificationResponse*** | **UeContextModificationFailure** |

备注：

1. 成功流程：

**Henb :Rx UE Context Modification Request🡪Tx UE Context Modification Response**

1. 失败流程：

**Henb :Rx UE Context Modification Request 🡪Tx Ue Context Modification Failure**

#### 2.3.4.6 终端注销过程

表 2.7 终端注销过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | | **Henb 🡨🡪SimMME（S1）** | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***UeContextReleaseCommand*** |  |
| ***RRCConnectionRelease*** |  | ***UeContextReleaseResponse*** |  |

备注：

**Henb :Rx Ue Ctx Release Cmd🡪Tx RRC Release && Tx Ue Ctx Release Resp**

#### 2.3.4.7 S1建立过程

表 2.8 S1建立过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | | **Henb 🡨🡪SimMME（S1）** | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***S1SetupRequest*** |  |
|  |  | ***S1SetupResponse*** | ***S1SetupFailure*** |
|  |  |  | ***Timeout S1SetupResponse*** |

备注：

1. 成功流程：

**Henb :Tx S1 Setup Req🡪Rx S1 Setup Resp**

1. 失败流程：

**Henb :Tx S1 Setup Req🡪Rx S1 Setup Failure（ReTx）**

**Henb :Tx S1 Setup Req🡪Timeout Rx S1 Setup Resp（ReTx）**

#### 2.3.4.8 MME配置更新过程

表 2.9 MME配置更新过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | **Henb 🡨🡪SimMME（S1）** | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***MMEConfigurationUpdate*** |  |
|  |  | ***MMEConfigurationUpdateAcknowledge*** | ***MMEConfigurationUpdateFailure*** |
|  |  |  |  |

备注：

1. 成功流程：

**Henb :Rx MME Configuration Update🡪Tx MME Configuration Update Acknowledge**

1. 失败流程：

**Henb :Rx MME Configuration Update🡪Tx MME Configuration Update Failure**

#### 2.3.4.9 ENB配置更新过程

表 2.10 ENB配置更新过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | **Henb 🡨🡪SimMME（S1）** | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***ENBConfigurationUpdate*** |  |
|  |  | ***ENBConfigurationUpdateAcknowledge*** | ***ENBConfigurationUpdateFailure*** |
|  |  |  | ***Timeout***  ***ENBConfigurationUpdateAcknowledge*** |

备注：

1. 成功流程：

**Henb :Tx ENB Configuration Update🡪Rx ENB Configuration Update Acknowledge**

1. 失败流程：

**Henb :Tx ENB Configuration Update🡪Rx ENB Configuration Update Failure（ReTx）**

**Henb :Tx ENB Configuration Update🡪Timeout Rx ENB Configuration Update Acknowledge（ReTx）**

#### 2.3.4.10 重置过程

表 2.11 重置过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | **Henb 🡨🡪SimMME（S1）** | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***Reset（MME init）*** |  |
|  |  | ***ResetAcknowledge（MME init）*** |  |
|  |  | ***Reset（ENB init）*** |  |
|  |  |  | ***ResetAcknowledge（ENB init）*** |  |
|  |  |  |  | ***Timeout***  ***ResetAcknowledge***  ***（ENB init）*** |

备注：

1. 成功流程：

**MME init**：

**Henb :Rx Reset🡪Tx Reset Acknowledge**

**ENB init：**

**Henb ：Tx Reset🡪Rx Reset Acknowledge**

1. 失败流程：

**ENB init：**

**Henb :Tx Reset🡪Timeout Rx Reset Acknowledge（ReTx）**

#### 2.3.4.11 错误指示过程

表 2.12 错误指示过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | **Henb 🡨🡪SimMME（S1）** | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***ErrorIndication（MME init）*** |  |
|  |  | ***ErrorIndication（ENB init）*** |  |

备注：

PS：此过程是在其他过程出错情况下辅助测试的。

1. 成功流程：

**MME init**：

**Henb :Rx Error Indication**

**ENB init：**

**Henb ：Tx Error Indication**

#### 2.3.4.12 超载过程

表 2.13 超载过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | **Henb 🡨🡪SimMME（S1）** | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***OverloadStart*** |  |
|  |  | ***OverloadStop*** |  |

备注：

1. 成功流程：

**超载开始：**

**Henb :Rx Overload Start**

**超载停止：**

**Henb ：Rx Overload Stop**

#### 2.3.4.13 UE性能指示过程

表 2.14 UE性能指示过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | **Henb 🡨🡪SimMME（S1）** | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***UECapbilityInfoIndication*** |  |

备注：

1. 成功流程：

**Henb :Tx UE Capbility Indication**

#### 2.3.4.14 位置上报过程

表 2.15位置上报过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **SimUe🡨🡪Henb（Uu）** | **Henb 🡨🡪SimMME（S1）** | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** |  |  | ***LocationReportingControl*** |  |
|  |  | ***LocationReport*** | **LocationReportFailureIndication** |

备注：

1. 成功流程：

**Henb :Rx Location Reporting Control🡪Tx Location Report**

1. 失败流程：

**Henb :Rx Location Reporting Control🡪Tx Location Report Failure Indication**

#### 2.3.4.15 直接信息传输过程

表 2.16直接信息传输过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **Henb（LTE） 🡨🡪SimMME（S1）🡨🡪enb（other2g、3g）** | | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** | ***ENBDirectInformationTransfer*** |  |  |  |
|  |  |  | ***MMEDirectInformationTransfer*** |  |

备注：

1. 成功流程：

**Henb :Tx ENB Direct Information Transfer**

**Henb：Rx MME Direct Information Transfer（no handle）**

#### 2.3.4.16 配置信息传输过程

表 2.17 配置信息传输更新过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **Henb1 🡨🡪SimMME（S1）🡨🡪Henb2** | | | |
| **Success** | **Failed** | **Success** | **Failed** |
| **1** | ***ENBConfigurationTransfer（Req）*** |  |  |  |
|  |  |  | ***MMEConfigurationTransfer（Req）*** |  |
|  |  |  | ***ENBConfigurationTransfer（Rep）*** |  |
|  | ***MMEConfigurationTransfer（Rep）*** |  |  |  |

备注：

1. 成功流程：

**Request：**

**Henb1 Tx ENB Configuration Transfer🡪Henb2 Rx MME Configuration Transfer**

**Reply：**

**Henb2 Tx ENB Configuration Transfer🡪Henb1 Rx MME Configuration Transfer**

#### 2.3.4.17 多UE测试场景

表 2.18 基站中4个UE同时测试过程

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Proc** | **Ue 1** | **Ue 2** | **Ue 3** | **Ue 4** |
| **RRC Establishment (SRB1)** | Success | Success | Success | Failed |
| **Ue Capability Enquiry** | Success | Success | Success |  |
| **Init Security Active** | Success | Success | Failed |  |
| **RRC Reconfiguration (SRB2)** | Success | Success |  |  |
| **Erab Setup** | Success | Failed |  |  |
| **Erab Modify** | Success |  |  |  |
| **Erab Release** | Success |  |  |  |
| **Ue Ctx Release** | Success |  |  |  |

备注：

1. 4个UE同时进行测试，同时可以将所有的正常流程测通；
2. 表17中有一个UE进行正常流程测试，其他UE在不同流程进行异常处理；

### 2.3.5 遗留问题

1.PAGING、CDMA2000、跟踪过程、警告消息、KILL过程、lppa等过程还未测试。

## 2.4 X2AP控制面联合测试

### 2.4.1 测试目的与内容

本测试目的是测试X2AP软件子系统控制面信令处理功能。本阶段测试侧重对X2连接建立、配置更新、重置及一些移动性管理等信令流程，在PC机上完成测试。

测试内容包括：

* X2连接建立；
* ENB配置更新；
* 重置；
* 负载指示；
* 错误指示；
* 资源状态报告；
* 小区激活；
* 无线链路失败和切换失败报告；
* 移动配置改变；

### 2.4.2 软硬件需求

1. **硬件需求**
   * PC两台
2. **软件需求**
   * 操作系统： Linux (Kernel 2.6.20)
   * 发行版本： CentOS 5.2
   * 编译环境： gcc-3.4.5-glibc-2.3.6
   * LTE HeNB协议栈软件系统一套
   * LTE Testing GW协议栈软件系统一套

### 2.4.3 测试场景



### 2.4.4 测试方案

#### 2.4.4.1 X2建立过程

表 2.19 X2建立过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***X2SetupRequest*** |  |
| ***X2SetupResponse*** | ***X2SetupFailure*** |
|  | ***TimeOut***  ***X2SetupResponse*** |

备注：

1. X2建立成功过程：

**Henb1 :Tx X2 Setup Request🡪Rx X2 Setup Response**

1. X2建立失败过程：

* 信令失败
  + **Henb1 :Tx X2 Setup Request🡪Rx X2 Setup Failure(according timetowait reinitiating X2 Setup procedure)**
* 超时失败
  + **Henb1 :Tx X2 Setup Request🡪 Timeout Rx X2 Setup Response(reinitiating X2 Setup procedure)**

#### 2.4.4.2 eNB配置更新过程

表 2.20 eNB配置更新过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***ENBConfigurationUpdate*** |  |
| ***ENBConfigurationUpdateAcknowledge*** | ***ENBConfigurationUpdateFailure*** |
|  | ***TimeOut***  ***ENBConfigurationUpdateAcknowledge*** |

备注：

1. eNB配置更新成功过程：

**Henb1 :Tx ENB Configuration Update🡪Rx ENB Configuration Update Response**

1. eNB配置更新失败过程：

* 信令失败
  + **Henb1 :Tx ENB Configuration Update🡪Rx ENB Configuration Update Failure(according timetowait reinitiating ENB Configuration Update procedure)**
* 超时失败
  + **Henb1 :Tx ENB Configuration Update🡪 Timeout Rx ENB Configuration Response(reinitiating ENB Configuration Update procedure)**

#### 2.4.4.3 X2 RESET过程

表 2.21 X2 RESET过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***ResetRequest*** |  |
| ***ResetResponse*** |  |
|  | ***TimeOut***  ***ResetResponse*** |

备注：

1. X2 RESET成功过程：

**Henb1 :Tx Reset Request🡪Rx Reset Response**

1. X2 RESET失败过程：

* 超时失败
  + **Henb1 :Tx Reset Request🡪 Timeout Rx Reset Response(reinitiating Reset procedure)**

#### 2.4.4.4 LOAD INDICATION过程

表 2.22 LOAD INDICATION过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***Load Information*** |  |

备注：

1. LOAD INDICATION成功过程：

**Henb1 :Tx Load Information**

#### 2.4.4.5 ERROR INDICATION过程

表 2.23 ERROR INDICATION过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***Error Indication*** |  |

备注：

ERROR INDICATION是在处理其他X2消息出错时选择发送的。

1. LOAD INDICATION成功过程：

**Henb1 :Tx Error Indication**

#### 2.4.4.6 资源状态报告过程

表 2.24资源状态报告过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***ResourceStatusRequest*** |  |
| ***ResourceStatusResponse*** | ***ResourceStatusFailure*** |
|  | ***TimeOut***  ***ResourceStatusResponse*** |
|  | **ResourceStatusUpdate** |  |

备注：

该过程分为两个过程：

* Henb1发起的Resource Status Reporting Initiation过程
* Henb2发起的Resource Status Reporting过程

1. 成功过程：

**Henb1 :Tx Resource Status Request🡪Rx Resource Status Response🡪Rx Resource Status Update**

1. 失败过程：

* 信令失败
  + **Henb1 : Tx Resource Status Request🡪 Rx Resource Status Failure**
* 超时失败
  + **Henb1 :Tx Resource Status Request🡪 Timeout Rx Resource Status Response(reinitiating Resource Status Reporting Initiation procedure)**

#### 2.4.4.7 小区激活过程

表 2.25小区激活过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***CellActivationRequest*** |  |
| ***CellActivationResponse*** | ***CellActivationFailure*** |

备注：

1. 成功过程：

**Henb1 :Tx Cell Activation Request🡪 Rx Cell Activation Response**

1. 失败过程：

* 信令失败
  + **Henb1 : Tx Cell Activation Request🡪 Rx Cell Activation Failure**

#### 2.4.4.8 Mobility Settings Change过程

表 2.26 Mobility Settings Change过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***MobilityChangeRequest*** |  |
| **MobilityChangeAcknowledge** | ***MobilityChangeFailure*** |

备注：

1. 成功过程：

**Henb1 :Tx Mobility Change Request🡪 Rx Mobility Change Acknowledge**

1. 失败过程：

* 信令失败
  + **Henb1 : Tx Mobility Change Request🡪 Rx Mobility Change Failure**

#### 2.4.4.9 MobilityRobustness Optimisation过程

表 2.27无线链路失败指示过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***RadioLinkFailureIndication*** |  |

备注：

1. 成功过程：

**Henb1 :Tx Radio Link Failure Indication**

表 10切换报告过程

|  |  |  |
| --- | --- | --- |
| **Proc** | **Henb1🡨X2🡪Henb2** | |
| **Success** | **Failed** |
| **1** | ***HandoverReport*** |  |

备注：

1. 成功过程：

**Henb1 :Tx Handover Report**

### 2.4.5 遗留问题

小区激活模块只将信令跑通、功能代码未完善。

## 2.5 协议栈联合测试

### 2.5.1 测试目的与内容

完成Uu口数据面以及控制面单独测试后，开展协议栈联合测试，本阶段测试完成对终端从入网、业务建立到数据收发整个过程的处理，验证LTE HeNB协议栈软件系统功能的正确性。

测试内容包括：

* 寻呼；
* RRC初始连接建立；
* 初始安全激活；
* RRC连接重配置；
* RRC连接释放；
* UE上下文建立、修改与释放；
* E-RAB建立、修改与释放；
* S1建立；
* 数据收发；

### 2.5.2 软硬件需求

1. **硬件需求**
   * PC三台
2. **软件需求**
   * 操作系统： Linux (Kernel 2.6.20)
   * 发行版本： CentOS 5.2
   * 编译环境： gcc-3.4.5-glibc-2.3.6
   * LTE HeNB协议栈软件系统一套
   * LTE Testing UE协议栈软件系统一套
   * LTE Testing GW协议栈软件系统一套

### 2.5.3 测试场景

测试中，使用三台PC，分别执行LTE HeNB协议栈，LTE Testing UE协议栈软件以及LTE Testing GW相关功能。测试场景物理连接关系以及逻辑关系如图2.5所示。

图2.5 HeNB协议栈联合测试场景

### 2.5.4 测试方案

#### 2.5.4.1 单UE业务建立过程测试

1）在单UE的联合测试中，使用vlc实时播放工具测试UM模式下的数据传输，具体的测试场景如图2.6所示

图2.6 HeNB协议栈联合测试UM模式场景

2）测试场景说明

本测试中共需要五台机器，两台windows系统、三台linux系统，其中Win1运行VLC程序的服务端，负责视频流的输出；Win2运行VLC程序的客户端，负责视频流的接收；Linux1运行网关协议栈，同时为了将数据转向虚拟网卡，这里需要运行一个接收本机外网数据包并将其发送到虚拟网卡的脚本程序，即udp\_route.py；Linux2运行基站协议栈；Linux3运行终端协议栈；

3）测试场景运行过程

1.启动MME，即在debug/s1ap\_tool/pysctp目录下，运行./mme.py 19000 10.21.1.188，后面的参数19000代表MME的端口，可以设置为用户可用的任意端口；0.21.1.188代表基站协议栈所在主机的ip地址。

2.启动udp\_route，即在debug/s1ap\_tool/pysctp目录下，运行./upd\_route.py 12233，后面的参数代表接收VLC数据的端口号，此端口号必须与VLC客户端发送的目的地址设置的端口号一致。

3.启动UE协议栈，在bin目录下运行./lte.ue.host -P 19000 -p 18004 -O 18003 -A 10.21.1.188 -V 17000 -v 10.21.1.141 10.21.1.189;其中-P 后的19000代表MME的端口号（与启动MME时的端口号一致），-p后的18004代表协议栈接收数据时监听的端口号（与基站发送的端口号一致），-O和-A后面的18003和10.21.1.188代表协议栈发送数据的目的端口号和地址（与基站的接收端口和ip一致），-V和-v表示VLC客户端的端口号和ip地址，最后的10.21.1.189代表运行MME的机器的ip地址。

4.启动HeNB协议栈，在bin目录下运行./lte.host -P 19000 -p 18003 -O 18004 -A 10.21.1.188 10.21.1.189,各参数的意义与ue运行时一致，只是没有-V和-v两个参数。

**注意：在两台协议栈进行通信时，有可能出现解包失败，这可能是由于不同机器使用的网络字节序不同而导致的大小端问题，所以在测试时最好用同一类机器；运行MME时需要建立虚拟网卡，所以在运行MME之前必须保证已经将tun模块加载到内核。**

4）数据走向

首先，设置VLC服务端，其数据的目的信息为MME所在的机器（10.21.1.189:12233），脚本dup\_toute监听12233的端口，接收到数据后，将数据给192.168.1.\*：2150，mme建立了192.168.1.1的虚拟网卡，接收所有的2150端口的数据，然后将数据通过GTP发送到HeNB的S1层，S1层监听GTP的端口，接收到数据后发到一个消息队列中，数据面PDCP层不断地从消息队列取数据，然后递交到下层，最后到达虚拟物理层。通过socket及配置的对端的套接字将数据发送给UE，UE接收到数据后按照逆过程将数据一层层解析递交到PDCP，然后PDCP通过设定的端口递交给VLC的客户端，从而实现视频的实时传输。

目前的测试场景连续跑了12个小时，没有任何问题。

1. 下表中黄色区域表示正常入网流程；下表中白色区域表示该过程的失败流程；下表中信令对齐方式代表信令发送的方向； [↑](#footnote-ref-1)