Openair LTE Core Network Control Plane

Training session

OAI workshop, Beijin, 04.25.2017

OPENAIR-CN training
Plan

- Plan
  - Scope
  - Software architecture
  - Tools
  - 3GPP Rel 10 implementation status
  - MME internal interfaces
  - What is missing

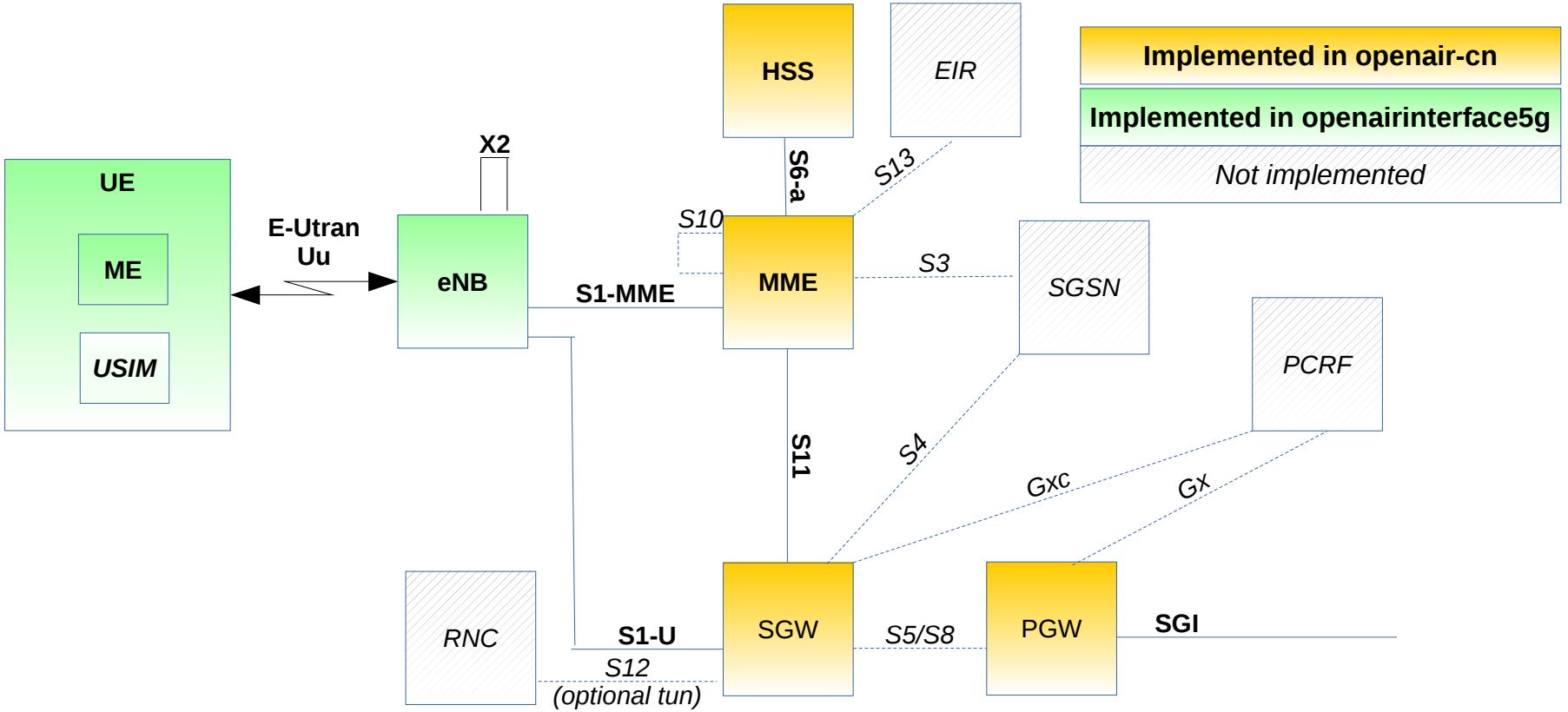
SCOPE

OPENAIR-CN training
Scope

- Targeted audience: Code contributors, testers, users of openair-cn.
- Objectives
  - Save time for willing contributors who want to understand what is in openair-cn
    - What is implemented
    - What is missing
    - Through some parts of UE attach scenario example, explain/show :
      - internals
      - How is organized the code
      - How are implemented the procedures
      - API.

LTE core network scope
Scope : Reminder 3GPP LTE Core Network entities

- LTE network entities involved in openair-cn :



SOFTWARE ARCHITECTURE

LTE Software Architecture
External Open Source libraries

- Asn1c (tool)
- NwGTPv2-c from Amit Chawre
- FreeDiameter
- OpenSSL
- Liblfds (lock free containers)
- Bstr
- Mscgen (tool)
- Complete list can be found in EPC user guide.

LTE Software Architecture
ITTI design

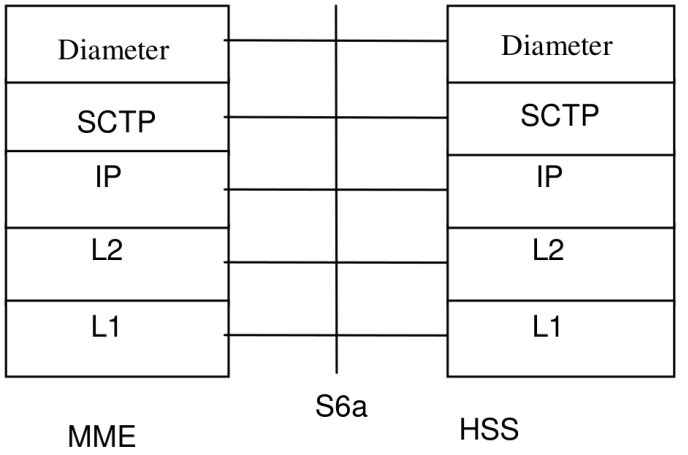
- ITTI means InTer Task Interface
  - It is a kind of lightweight middleware providind services to the application.
- Goals and interests
  - Provide services abstracted from the underlying operating system.
    - This will make the porting to other operating system easier.
- List of services
  - Timer facilities
  - Asynchronous Inter-task message facilities.
  - I/O events facilities (sockets, pipes, files).
  - Each protocol instance or interface adapter has been assigned its own ITTI task.
- MME and SPGW user space executables use ITTI.
- In HSS, ITTI is not used :
  - Actually use threading architecture provided by Diameter library.

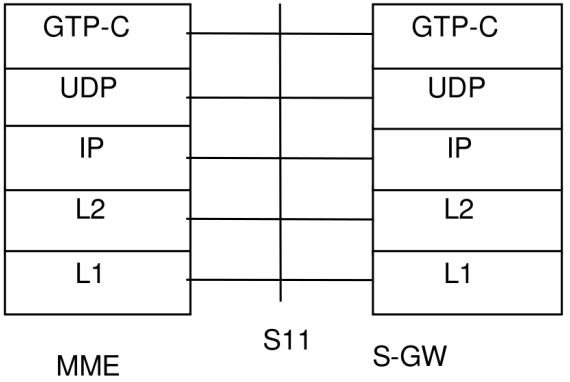
LTE Software Architecture
MME ITTI design

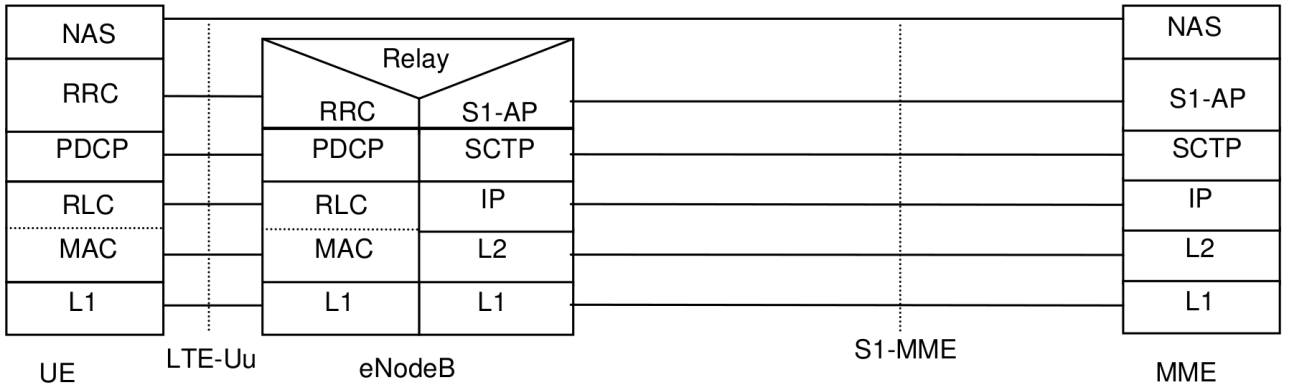
- In MME and SPGW
  - Each protocol instance or interface adapter has been assigned its own task (InTerTask Interface).
  - Each task is wake-up by events (messages, IO events, timer events).

LTE Software Architecture
MME ITTI design - Tasks split

Protocol layer	Application layer	ITTI task name
SCTP	-	TASK_SCTP
S1AP	-	TASK_S1AP
-	MME_APP	TASK_MME_APP
NAS	-	TASK_NAS_MME
UDP	-	TASK_UDP
GTP-C	S11 (interface adapter, lib nwGTPv2c wrapped)	TASK_S11
Diameter	S6a (interface adapter, lib freeDiameter wrapped)	TASK_S6A







LTE Software Architecture
ITTI Services

- Task services
  - int **itti_create_task**(task_id_t task_id, void *(*function) (void *), void *args_p)
  - void **itti_wait_ready**(int wait_tasks);
  - void **itti_mark_task_ready**(task_id_t task_id);
  - void **itti_exit_task**(void);
  - void **itti_terminate_tasks**(task_id_t task_id);
  - const char ***itti_get_task_name**(task_id_t task_id);
  - void **itti_wait_tasks_end**(void);

LTE Software Architecture
ITTI Services

- **Message services**
  - MessageDef ***itti_alloc_new_message** (task_id_t origin_tid, MessagesIds msg_id);
  - int **itti_send_broadcast_message** (MessageDef *mesg_p);
  - int **itti_send_msg_to_task** (task_id_t tid, instance_t inst, MessageDef *mesg);
  - void **itti_send_terminate_message** (task_id_t tid);
  - void **itti_receive_msg** (task_id_t tid, MessageDef **rx_msg);
  - void **itti_poll_msg**(task_id_t task_id, MessageDef **rx_msg);
  - const char ***itti_get_message_name** (MessagesIds msg_id);

LTE Software Architecture
ITTI Services

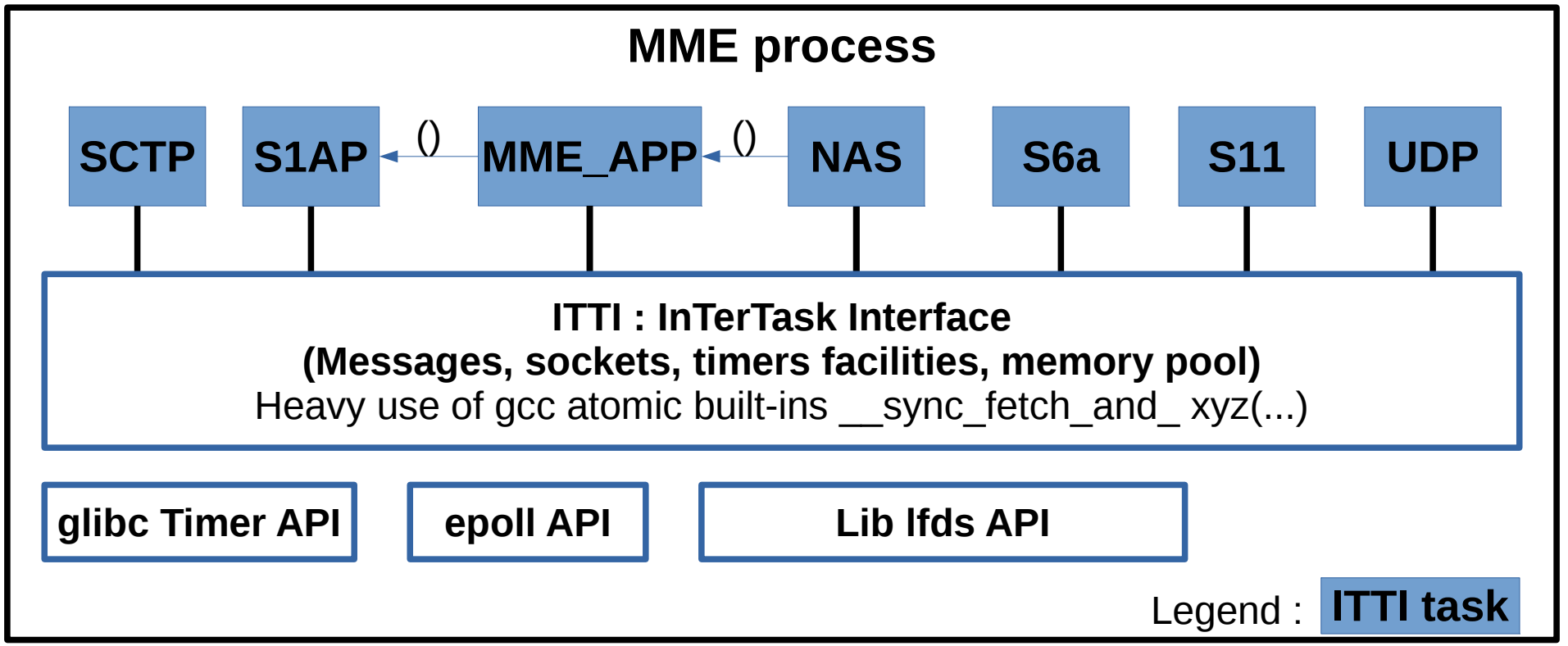
- **IO events services**
  - void **itti_subscribe_event_fd** (task_id_t tid, int fd);
  - void **itti_unsubscribe_event_fd** (task_id_t tid, int fd);
  - int **itti_get_events** (task_id_t tid, struct epoll_event **events);

LTE Software Architecture
ITTI Services

- Timer services
  - int **timer_setup** (uint32_t interval_sec, uint32_t interval_us, task_id_t tid, int32_t instance, timer_type_t type, void *timer_arg, long *timer_id);
  - int **timer_remove** (long timer_id);
  - int **timer_init** (void);

LTE Software Architecture
MME ITTI design

- Each ITTI task has to create a thread infinitly looping on a epoll_wait() function waiting for ITTI events.



LTE Software Architecture Internals : ITTI design

- ITTI task pthread template, example TASK_UDP

```
int                                     nb_events = 0;
struct epoll_event                     *events = NULL;

itti_mark_task_ready (TASK_UDP);
while (1) {
  MessageDef                           *received_message_p = NULL;
  itti_receive_msg (TASK_UDP, &received_message_p);
  if (received_message_p != NULL) {
    switch (ITTI_MSG_ID (received_message_p)) {
    case UDP_INIT:
         ...
      break;
    case TERMINATE_MESSAGE:
        itti_exit_task ();
      break;
    default:
        OAILOG_DEBUG (LOG_UDP, "Unkwnon message ID %d:%s\n",
             ITTI_MSG_ID (received_message_p), ITTI_MSG_NAME (received_message_p)) ;

  on_error:
    rc = itti_free (ITTI_MSG_ORIGIN_ID (received_message_p), received_message_p);
  }
  nb_events = itti_get_events (TASK_UDP, &events);
  if ((nb_events > 0) && (events != NULL)) {
    udp_server_flush_sockets (events, nb_events);
  }
}
```

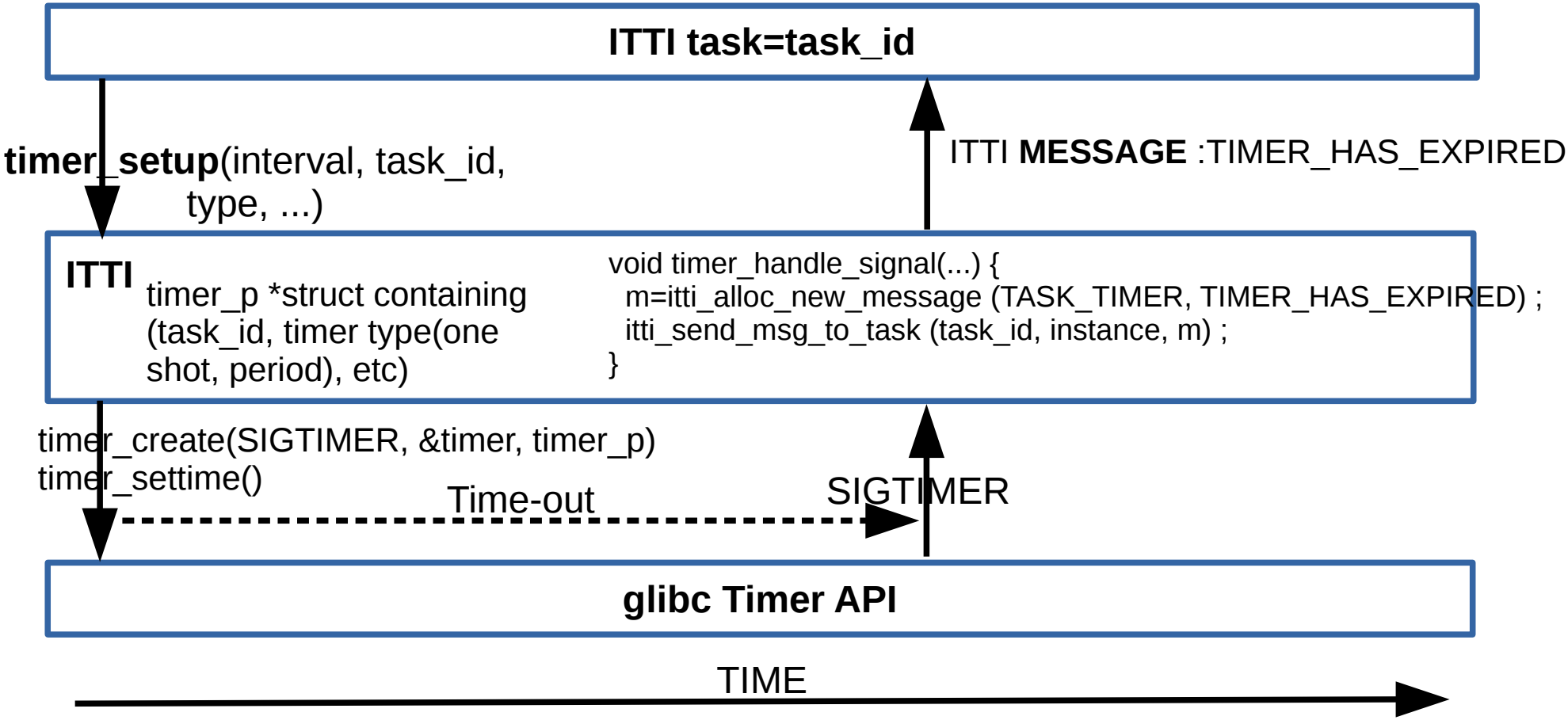
Blocking on rx event

Message processing

FD events processing

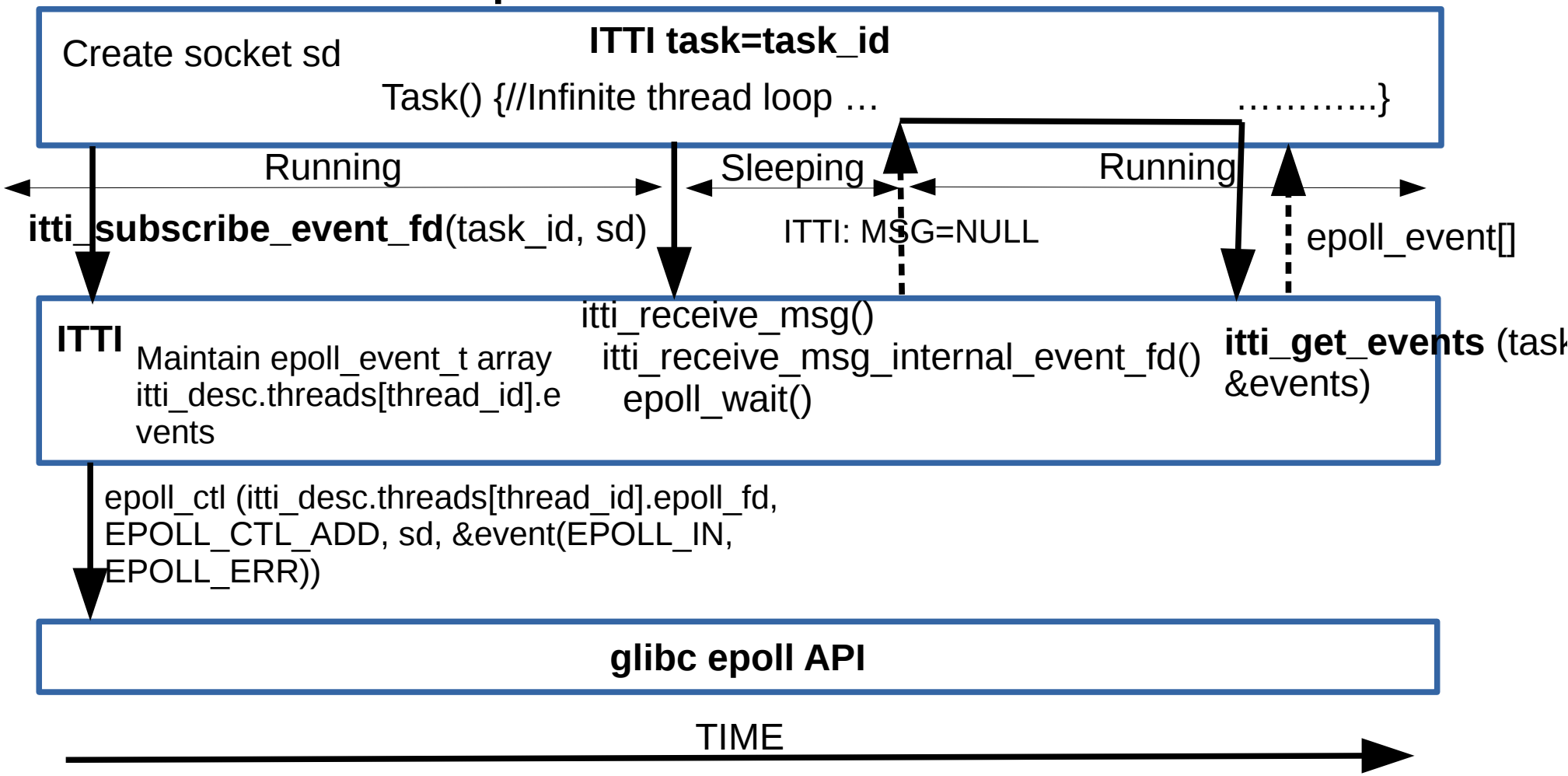
LTE Software Architecture Internals : ITTI timers

- ITTI timer internals



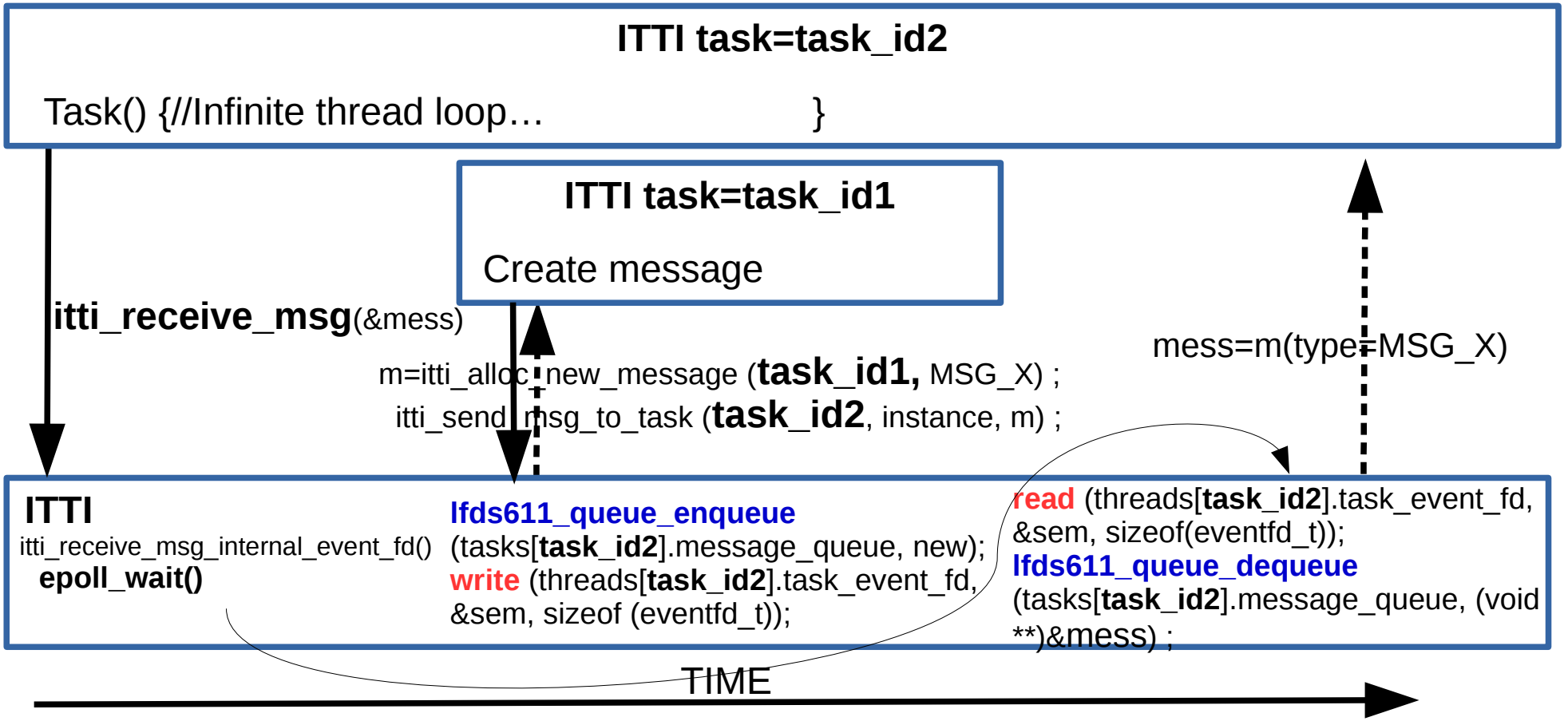
LTE Software Architecture
Internals : ITTI File descriptor facility

- ITTI file descriptors internals (see for ex TASK_UDP)



LTE Software Architecture
ITTI Message facility

- ITTI Message internals



TOOLS

Tools

- **Helpers for developpers, testers**
  - **Mscgen**
  - **Logs**
  - **3GPP requirements (use Logs)**
  - **MME Scenario player**

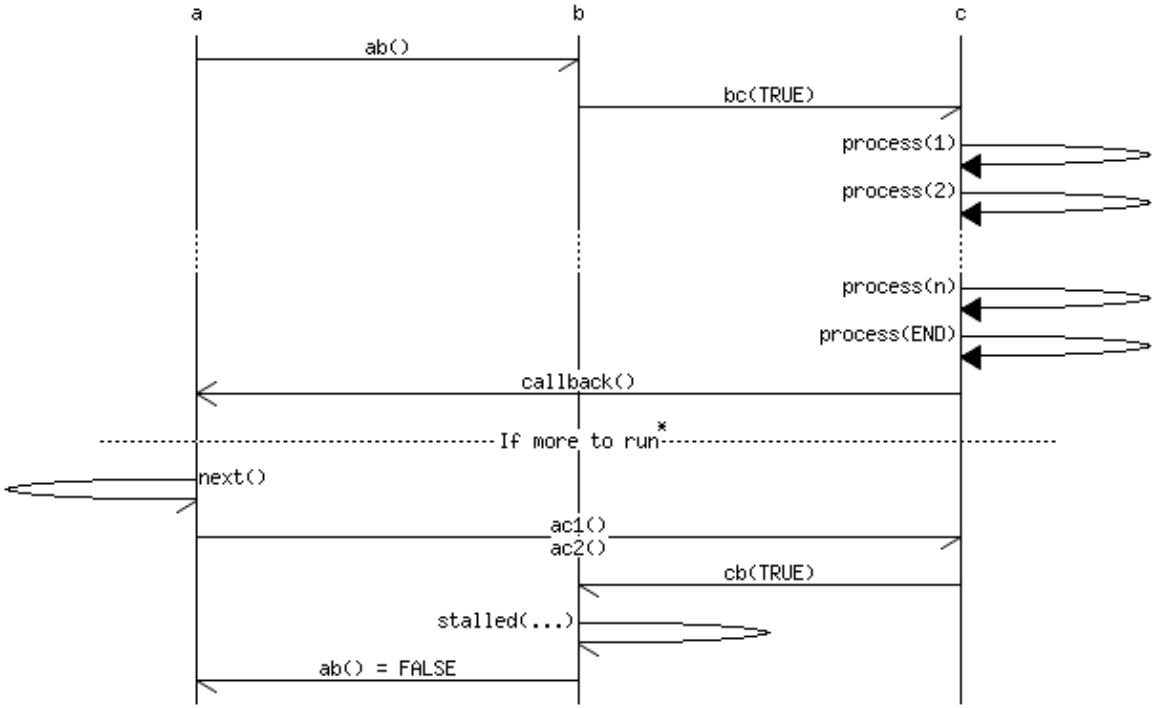
TOOLS
MSCGEN

- http://www.mcternan.me.uk/mscgen/
- Mscgen is a small program that parses **Message Sequence Chart descriptions** and produces PNG, SVG, EPS or server side image maps (ismaps) as the output.
- MSCs are popular in Telecoms to specify how protocols operate although **MSCs need not be complicated to create or use.**

TOOLS
MSCGEN

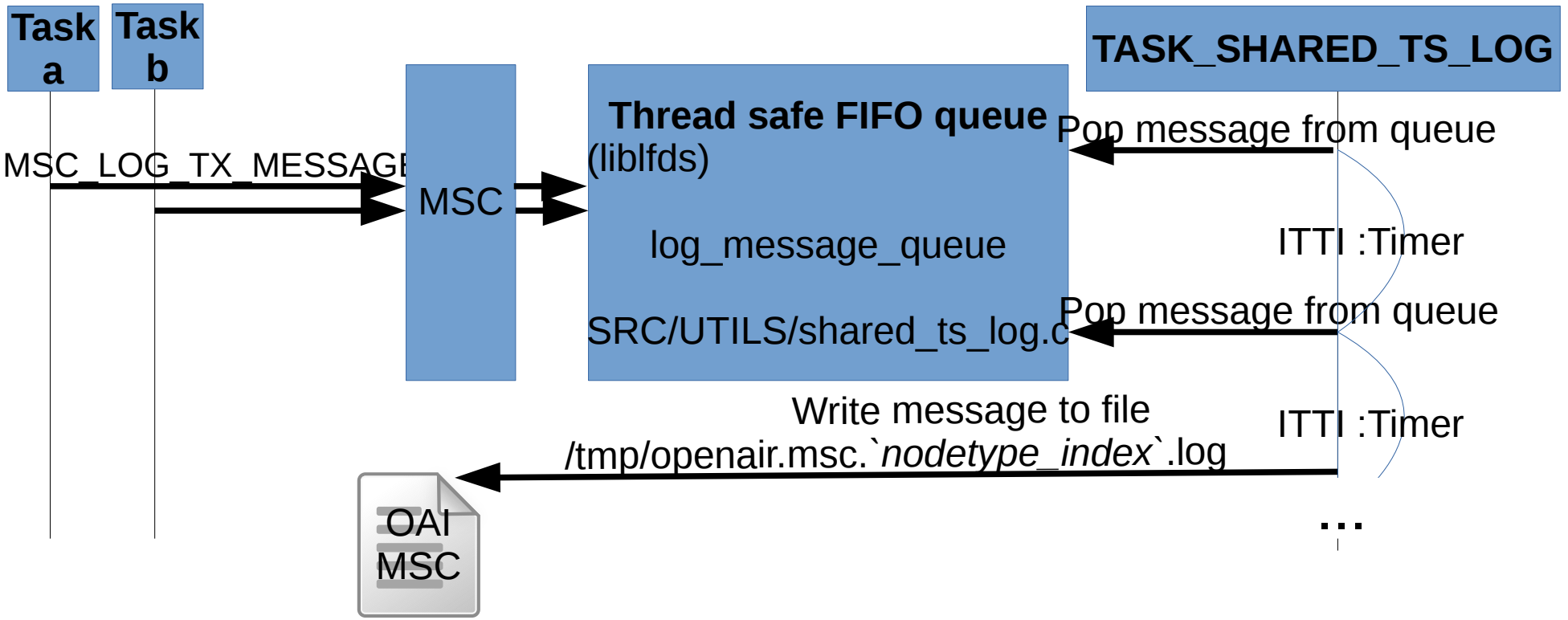
Declaration of protocol entities

```
# MSC for some fictional process
msc {
  hscale = "2";

  a,b,c;

  a->b [ label = "ab()" ] ;
  b->c [ label = "bc(TRUE)"];
  c=>c [ label = "process(1)" ];
  c=>c [ label = "process(2)" ];
  ...;
  c=>c [ label = "process(n)" ];
  c=>c [ label = "process(END)" ];
  a<<=c [ label = "callback()"];
  ---  [ label = "If more to run", ID="*" ];
  a->a [ label = "next()"];
  a->c [ label = "ac1()\nac2()"];
  b<-c [ label = "cb(TRUE)"];
  b->b [ label = "stalled(...)"];
  a<-b [ label = "ab() = FALSE"];
}
```



Message arc

TOOLS
MSCGEN : MSC thread-safe design



TOOLS
MSCGEN : MSC Declaration of entities

- Declaration of node types, entities (SRC/UTILS/MSC/msc.h):

```
typedef enum {
  MIN_MSC_ENV = 0,
  MSC_E_UTRAN = MIN_MSC_ENV,
  MSC_MME,
  MSC_SP_GW,
  MAX_MSC_ENV
} msc_env_t;
```

```
typedef enum {
  MIN_MSC_PROTOS = 0,
  MSC_NAS_UE = MIN_MSC_PROTOS,
  MSC_S1AP_ENB,
  MSC_GTPU_ENB,
  MSC_GTPU_SGW,
  MSC_GTPC_SGW,
  MSC_GTPC_MME,
  MSC_S1AP_MME,
  MSC_MMEAPP_MME,
  MSC_NAS_MME,
  MSC_NAS_EMM_MME,
  MSC_NAS_ESM_MME,
  MSC_S11_MME,
  MSC_S6A_MME,
  MSC_SGW,
  MSC_HSS,
  MAX_MSC_PROTOS,
} msc_proto_t;
```

TOOLS
MSCGEN: MSC API

- MSC API
  - **MSC_INIT**(node_type, max_protocol_entities) // open msc log file, init all variables.
  - **MSC_START_USE**() // to be called after MSC_INIT() by each producer thread (liblfds issue)
  - **MSC_END**() // stop use mscgen logging : close msc log file
- MSC log message primitives:
  - **MSC_LOG_RX_MESSAGE** (msc_rx_entity, msc_tx_entity, binary_stream, string_format, args…)
  - **MSC_LOG_RX_DISCARDED_MESSAGE** (msc_rx_entity, msc_tx_entity, binary_stream, string_format, args…)
  - **MSC_LOG_TX_MESSAGE** (msc_tx_entity, msc_rx_entity, binary_stream, string_format, args…)
  - **MSC_LOG_TX_MESSAGE_FAILED** (msc_tx_entity, msc_rx_entity, binary_stream, string_format, args…)
  - **MSC_LOG_EVENT** (msc_entity, string_format, args…)
- MSC are activated by setting MESSAGE_CHART_GENERATOR to true in openair/BUILD/XYZ/CmakeLists.template

TOOLS
MSCGEN : « OAI MSC » output file format 1/3

- Format of MSC intermediate log file

  Format : MSC file is a text file with space separated values.

  - **Protocol entities declaration**

Item	0	1	2	3
Field Description	Uniq item number	Keyword '[PROTO]'	Protocol entity identifier (integer b10)	Protocol entity display name

  - **Event declaration**

Item	0	1	2	3	4
Field Description	Uniq item number	Keyword '[EVENT]'	Protocol entity Identifier	Time	String to display

TOOLS
MSCGEN : « OAI MSC » output file format 2/3

Format : MSC file is a text file with space separated values.

- **Message declaration**

Item	0	1	2	3	4
Field Description	Unig item number	Keyword '[MESSAGE]'	Protocol entity identifier 1	Message arc	Protocol entity identifier 2

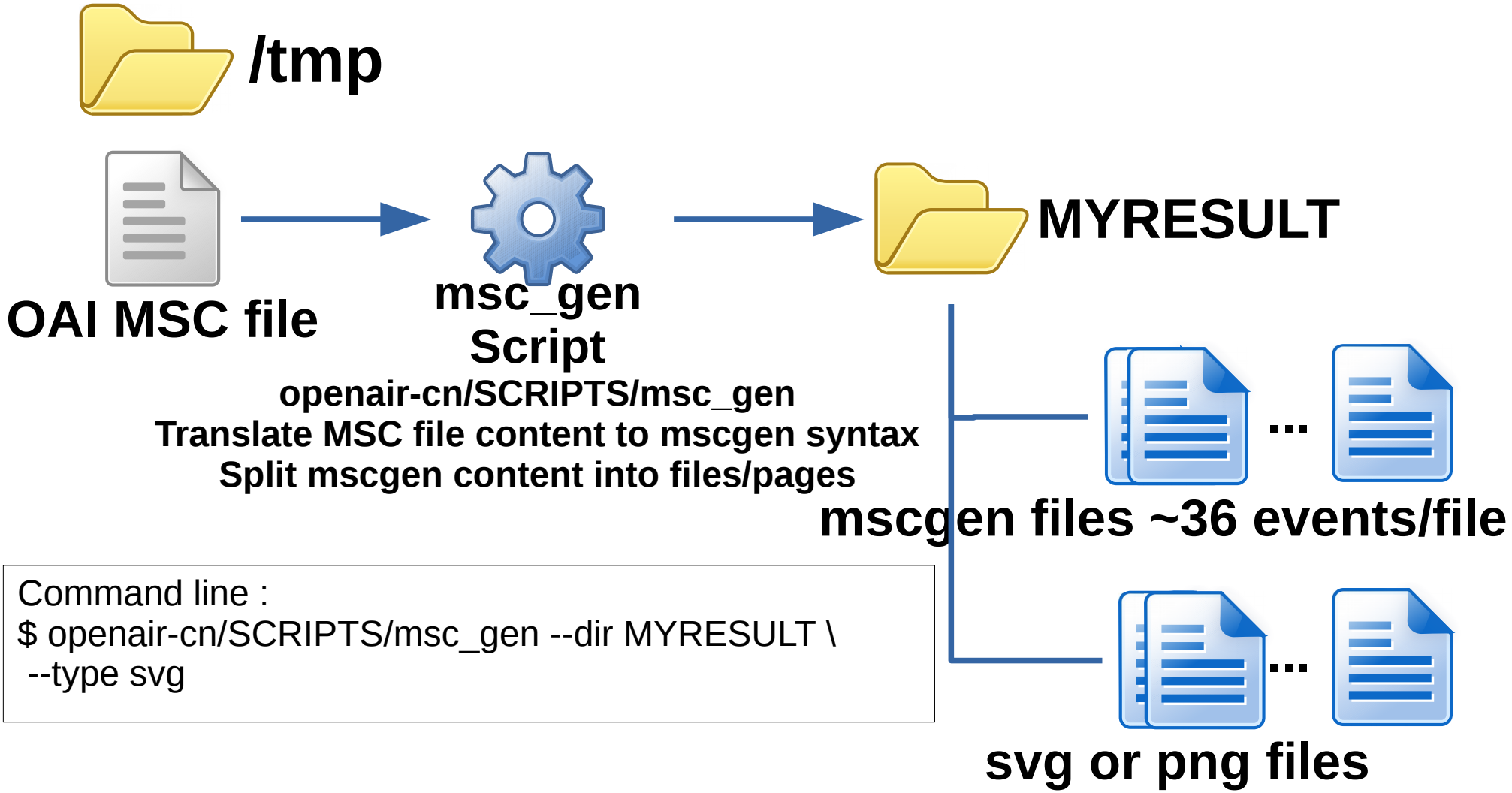
Item	5	(6)	7 or (6)
Field Description	Length of binary stream	Binary stream if length != 0, else field not present.	String to display

TOOLS
MSCGEN : « OAI MSC » output file format 3/3

- « OAI MSC » file example :

```
0 [PROTO] 1 S1AP_ENB
1 [PROTO] 2 GTPU_ENB
2 [PROTO] 6 S1AP_MME
3 [PROTO] 7 MME_APP
4 [PROTO] 8 NAS_MME
5 [PROTO] 9 NAS_EMM
6 [PROTO] 10 NAS_ESM
7 [PROTO] 12 S11_MME
8 [PROTO] 13 S6A
9 [PROTO] 14 SGW
10 [PROTO] 15 HSS
11 [EVENT] 6 0005:8966590 Event SCTP_NEW_ASSOCIATION assoc_id: 6
12 [MESSAGE] 6 <- 1 0 0005:8987890 S1Setup/Originating message assoc_id 6 stream 0
13 [MESSAGE] 6 -> 1 0 0005:8988690 S1Setup/successfulOutcome assoc_id 6
```

TOOLS
MSCGEN output files creation



Command line :
$ openair-cn/SCRIPTS/msc_gen --dir MYRESULT \
 --type svg

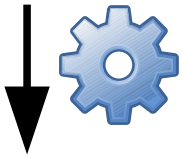
TOOLS
MSCGEN file creation

```
0 [PROTO] 1 S1AP_ENB
1 [PROTO] 2 GTPU_ENB
2 [PROTO] 6 S1AP_MME
...
11 [EVENT] 6 0005:8966590 Event SCTP_NEW_ASSOCIATION assoc_id: 6
12 [MESSAGE] 6 <- 1 0 0005:8987890 S1Setup/Originating message assoc_id 6 stream 0
13 [MESSAGE] 6 -> 1 0 0005:8988690 S1Setup/successfulOutcome assoc_id 6
```

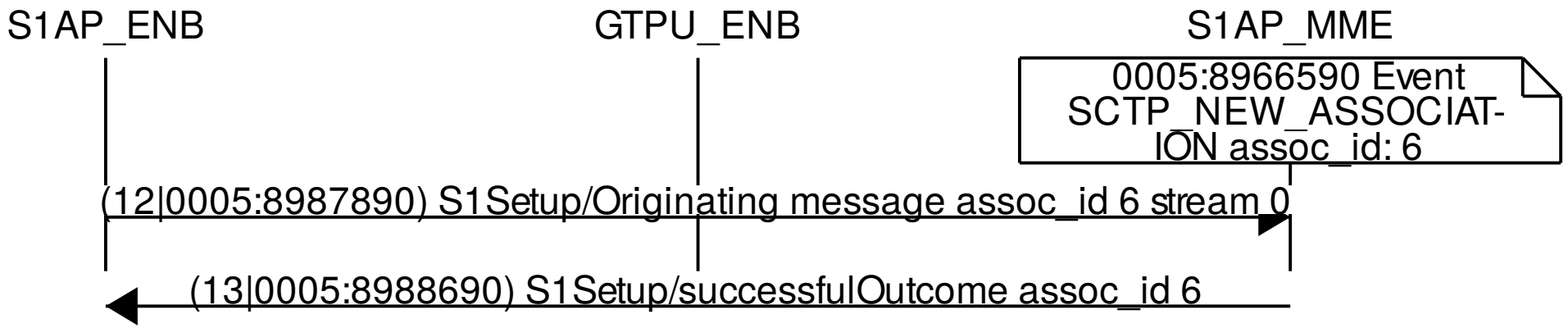

```
msc {
width = "2048";
   S1AP_ENB, GTPU_ENB, S1AP_MME, MME_APP, NAS_MME, NAS_EMM, NAS_ESM, S11_MME,
S6A, SGW, HSS;
  S1AP_MME note S1AP_MME [ label = "0005:8966590
 Event SCTP_NEW_ASSOCIATION assoc_id: 6", textcolour="black" ] ;
  S1AP_MME<=S1AP_ENB [ label = "(12|0005:8987890) S1Setup/Originating
message assoc_id 6 stream 0", linecolour="black" , textcolour="black" ] ;
  S1AP_MME=>S1AP_ENB [ label = "(13|0005:8988690) S1Setup/successfulOutcome
assoc_id 6", linecolour="black" , textcolour="black" ] ;
```

TOOLS
MSCGEN file creation

```
msc {
width = "2048";
   S1AP_ENB, GTPU_ENB, S1AP_MME, MME_APP, NAS_MME, NAS_EMM, NAS_ESM, S11_MME,
S6A, SGW, HSS;
  S1AP_MME note S1AP_MME [ label = "0005:8966590
 Event SCTP_NEW_ASSOCIATION assoc_id: 6", textcolour="black" ] ;
  S1AP_MME<=S1AP_ENB [ label = "(12|0005:8987890) S1Setup/Originating message
assoc_id 6 stream 0", linecolour="black" , textcolour="black" ] ;
  S1AP_MME=>S1AP_ENB [ label = "(13|0005:8988690) S1Setup/successfulOutcome
assoc_id 6", linecolour="black" , textcolour="black" ] ;
```

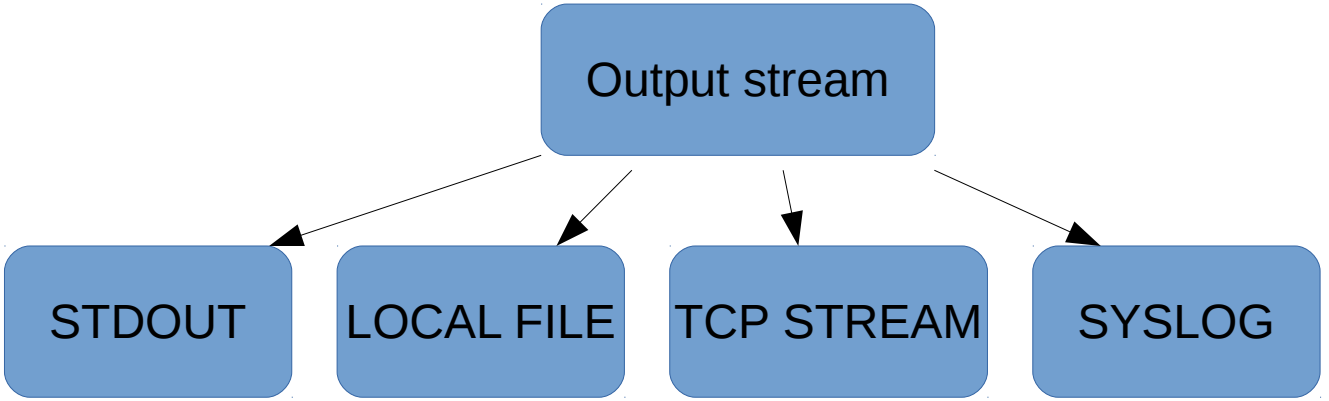


mscgen called by msc_gen



TOOLS
Logging 1/2

- Same architecture as MSC
  - Thread safe FIFOs (possibility to bypass this, usefull if you can't see the logs that appear between the last flush and the exception, if any !)
  - 1 backgroung thread poping logs and flushing them to
    - STDOUT (choice CONSOLE)
    - File
    - TCP server
    - Syslog
  - Log levels : TRACE, DEBUG, INFO, NOTICE, WARNING, ERROR, CRITICAL, ALERT (may be too much ?).



TOOLS
Logging config section 2/2

```
LOGGING :
{
    # OUTPUT choice in { "CONSOLE", "`path to file`", "`IPv4@`:`TCP port num`"}
    # `path to file` must start with '.' or '/'
    # if TCP stream choice, then you can easily dump the traffic on the remote or
local host: nc -l `TCP port num` > received.txt
    OUTPUT              = "CONSOLE";
    # THREAD_SAFE choice in { "yes", "no" } means use of thread safe intermediate
buffer then a single thread pick each message log one
    # by one to flush it to the chosen output
    THREAD_SAFE         = "yes";
    # COLOR choice in { "yes", "no" } means use of ANSI styling codes or no
    COLOR               = "yes";
    # Log level choice in { "EMERGENCY", "ALERT", "CRITICAL", "ERROR", "WARNING",
"NOTICE", "INFO", "DEBUG", "TRACE"}
    SCTP_LOG_LEVEL    = "TRACE";
    S11_LOG_LEVEL     = "TRACE";
    GTPV2C_LOG_LEVEL  = "TRACE";
    UDP_LOG_LEVEL     = "TRACE";
    S1AP_LOG_LEVEL    = "TRACE";
    NAS_LOG_LEVEL     = "TRACE";
    MME_APP_LOG_LEVEL = "TRACE";
    S6A_LOG_LEVEL     = "TRACE";
    UTIL_LOG_LEVEL    = "TRACE";
    MSC_LOG_LEVEL     = "ERROR";
    ITTI_LOG_LEVEL    = "ERROR";
};
```

TOOLS
3GPP requirements 1/2

- Objective
  - Testing/debug purpose
    - Leave a trace of a **3GPP specification requirement** (MUST/MAY/SHOULD/ COULD) **hit** in the code, ease the maintenance of the code.
    - Could use the trace or sequence of traces to **diagnostic** a successful test passed or failled.
    - Higher level trace than raw logging (TRACE→EMERGENCY) that should tell how/why things (bugs/success/malfunctions) occur.
- Actually relies on logging
  - **Macro** to be **activated** in openair-cn/BUILD/MME/CmakeLists.template by setting TRACE_3GPP_SPEC to true.

TOOLS
3GPP requirements 2/2

- Example : will log
  - **Hit 3GPP TS 24_301R10_5_4_2_4__1 : AUTHENTICATION RESPONSE received, stop T3460, check RES**

Extract from openair-cn/SRC/COMMON/3gpp_requirements.h

```
#    define REQUIREMENT_3GPP_SPEC(pRoTo, sTr) OAILOG_SPEC(pRoTo, sTr) // NOTIC LOG LEVEL
```

Extract from openair-cn/SRC/NAS/3gpp_requirements_24.301.h :

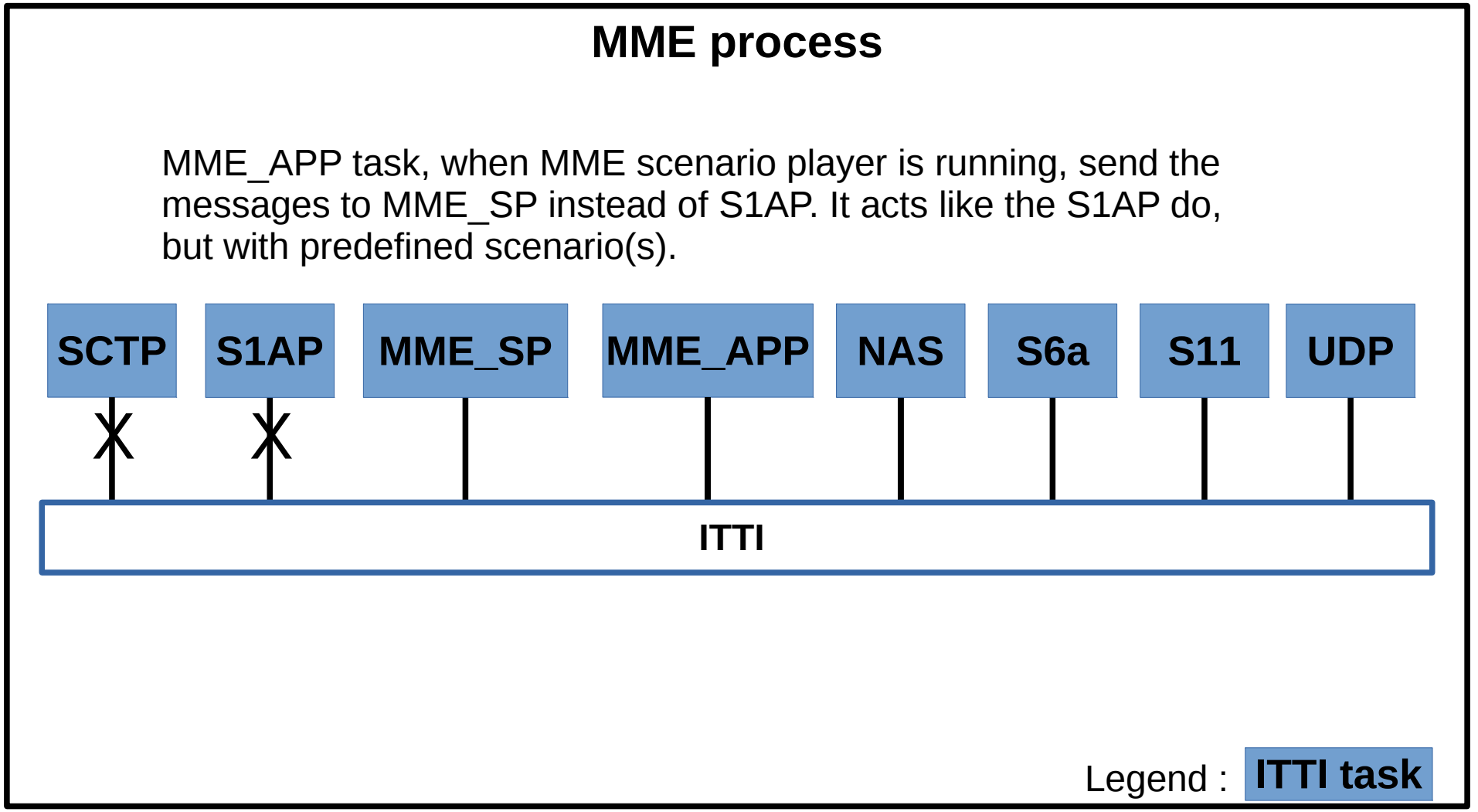
```
#define REQUIREMENT_3GPP_24_301(rElEaSe_sEcTiOn__OaImark) REQUIREMENT_3GPP_SPEC(LOG_NAS,
"Hit 3GPP TS 24_301"#rElEaSe_sEcTiOn__OaImark" : "rElEaSe_sEcTiOn__OaImark##_BRIEF"\n")

#define R10_5_4_2_4__1 "Authentication completion by the network                                    \
   Upon receipt of an AUTHENTICATION RESPONSE message, the network stops the timer T3460 and checks the  \
   correctness of RES (see 3GPP TS 33.401 [19])."
#define R10_5_4_2_4__1_BRIEF "AUTHENTICATION RESPONSE received, stop T3460, check RES"
```

```
if (emm_ctx) {
  // Stop timer T3460
  REQUIREMENT_3GPP_24_301(R10_5_4_2_4__1);
  emm_ctx->T3460.id = nas_timer_stop (emm_ctx->T3460.id);
```

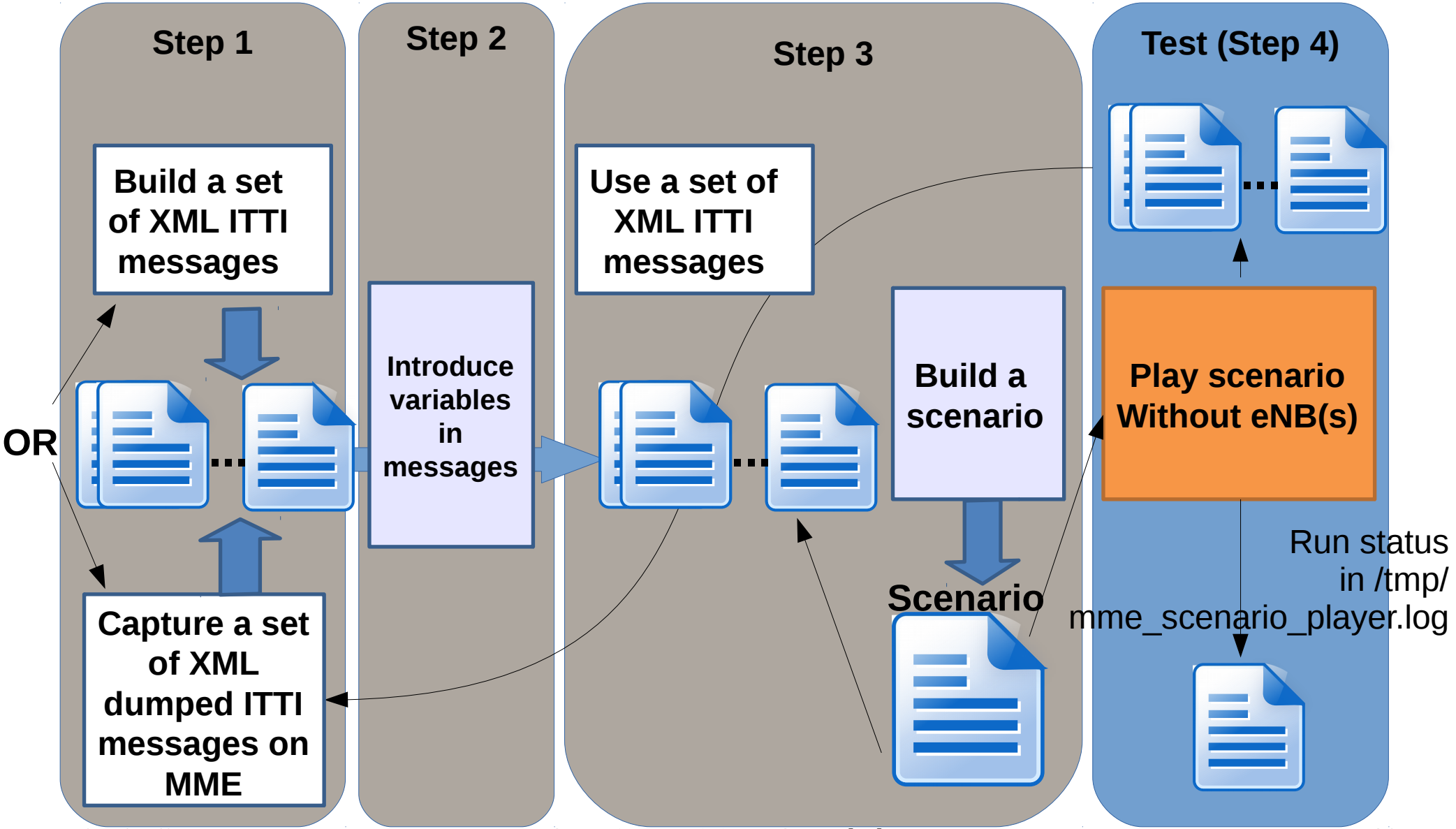
TOOLS
MME Scenario player

- Motivation
  - We have to develop a scenario player for covering all (as many as possible) test cases.
    - Do non regression testing.
    - Capture buggy/suspicious scenarios and replay them for debbuging.

LTE Software Architecture
MME ITTI design



TOOLS
MME Scenario Player, Howto build a scenario



TOOLS
MME Scenario player, Scenarios

- Scenarios **can** contains
  - Rx message(s) with or without timing constraints
  - Tx message(s) with or without timing requirements
  - Variable(s) declaration(s)
    - Variable types : uint64, hex stream, ascii stream
  - Test(s) of variable(s)
    - <jcond var_name="MME_UE_S1AP_ID" cond="ne" value="0xFFFFFFFF" label="checked_mme_ue_s1ap_id_invalid"/>
  - Labels like goto of BASIC
    - <label name="checked_mme_ue_s1ap_id_invalid" />

TOOLS
MME Scenario player, Scenarios

- Scenarios **can** contains
  - Labels like goto of BASIC
    - <label name="checked_mme_ue_s1ap_id_invalid" />
  - Special tags :
    - <usim lte_k="fec86ba6eb707ed08905757b1bb44b8f" sqn_ms="FF9BB4000E0C"/>
    - <compute_authentication_response_parameter/> < !-- Warning implicit variables (TODO) -->
    - <update_emm_security_context seea="$ITTI_NAS_DOWNLINK_DATA_REQ.NAS.SECURITY_MODE_COMMAND.TYPE_OF_CIPHERING_ALGORITHM" seia="$ITTI_NAS_DOWNLINK_DATA_REQ.NAS.SECURITY_MODE_COMMAND.TYPE_OF_INTEGRITY_PROTECTION_ALGORITHM" ul_count="$NAS_UPLINK_SEQUENCE_NUMBER"/>
  - Timing delays
    - <sleep seconds="10" useconds="0" />
  - Scenario(s)

    See example: https://gitlab.eurecom.fr/oai/openair-cn/blob/master/TEST/MME/all.xml

TOOLS
MME Scenario player, Variables

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ITTI_NAS_DOWNLINK_DATA_REQ>
  <ACTION>WAIT_RECEIVE</ACTION>
  <ITTI_SENDER_TASK>TASK_MME_APP</ITTI_SENDER_TASK>
  <ITTI_RECEIVER_TASK>TASK_MME_SCENARIO_PLAYER</ITTI_RECEIVER_TASK>
  <TIMESTAMP>ANY</TIMESTAMP>
  <mme_ue_s1ap_id>#ITTI_NAS_DOWNLINK_DATA_REQ.MME_UE_S1AP_ID</mme_ue_s1ap_id>
  <enb_ue_s1ap_id>$ENB_UE_S1AP_ID</enb_ue_s1ap_id>
  <plain_nas_message>
    <security_header_type>0x000000</security_header_type>
    <protocol_discriminator>EPS_MOBILITY_MANAGEMENT_MESSAGE</protocol_discriminator>
    <message_type>IDENTITY_REQUEST</message_type>
    <identity_type_2>1</identity_type_2>
  </plain_nas_message>
</ITTI_NAS_DOWNLINK_DATA_REQ>
```

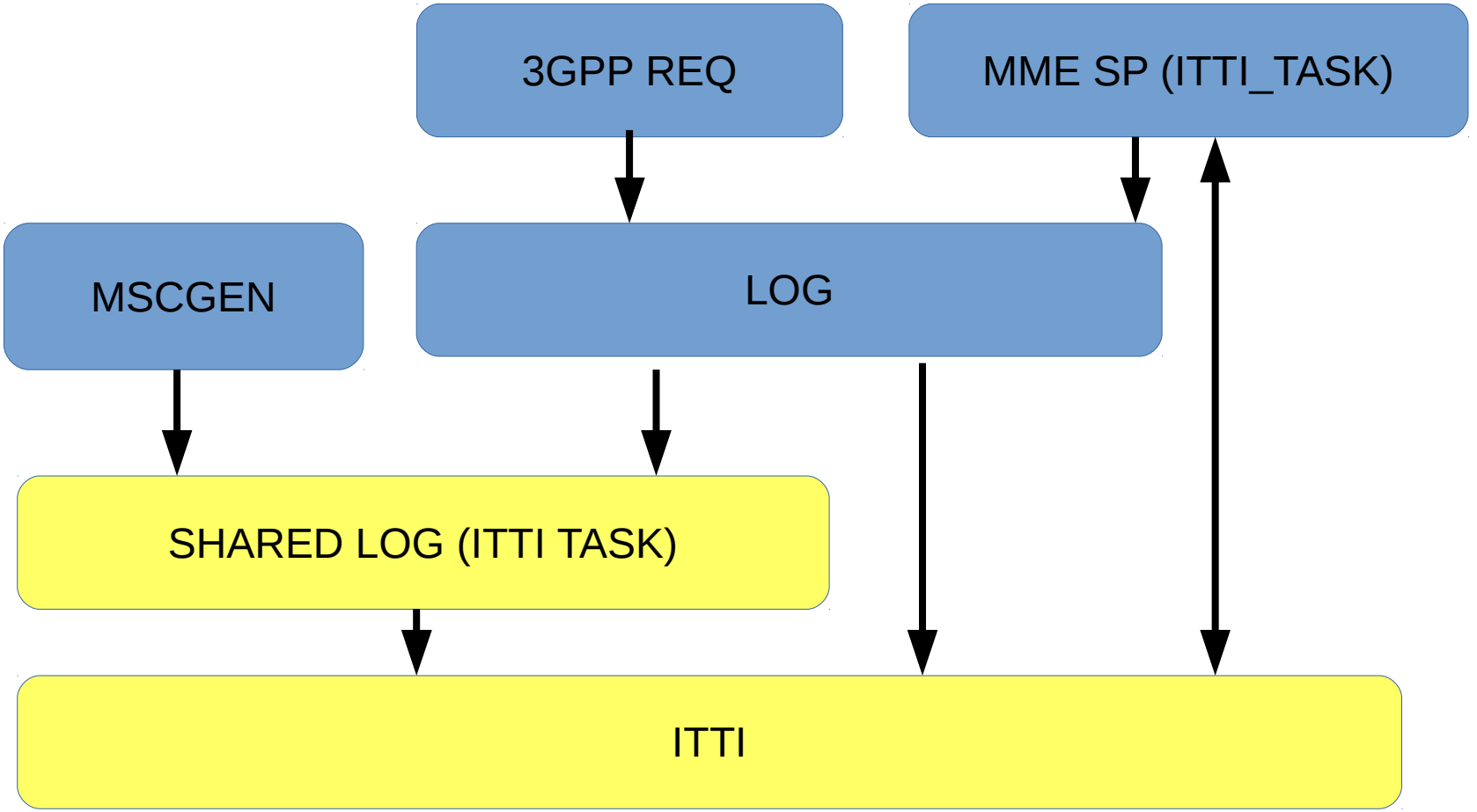
« # » means that variable ITTI_NAS_DOWNLINK_DATA_REQ.MME_UE_S1AP_ID will take the value received in message

« **$** » means that the scenario will fail if the enb_ue_s1ap_id field is not equal to **ENB_UE_S1AP_ID variable**

The scenario will fail if field « identity_type_2 » is not present and if value received is not « 1 »

TOOLS
Dependancies

- Interest of the figure : Show dependancy with ITTI



3GPP Rel 10 implementation Status

Disgression on Release 14

- Work to be done :
  - Evaluate the delta with Release 10 in core network for NB-IoT support
  - Area of collaboration/contribution

3GPP Implementation Status
S1AP Elementary Class 1 Procedures

Handover Preparation	-	Reset	-
Handover Resource Allocation	-	**S1 Setup**	**Yes**
Path Switch Request	Will be shortly	**UE Context Release**	**Yes**
Handover Cancellation	-	UE Context Modification	-
E-RAB Setup	**Yes**	ENB Configuration Update	-
E-RAB Modify	-	MME Configuration Update	-
E-RAB Release	-	Write-Replace Warning	-
Initial Context Setup	**Yes**	Kill	-

3GPP Implementation Status
S1AP Elementary Class 2 Procedures

Handover Notification	-	Trace Start	-
E-RAB Release Indication	-	Trace Failure Indication	-
Paging	Will be	Location Reporting Control	-
Initial UE Message	**Yes**	Location Reporting Failure Indication	-
Downlink NAS Transport	**Yes**	Location Report	-
Uplink NAS Transport	**Yes**	Overload Start	-
NAS non delivery indication	**Yes**	Overload Stop	-
Error Indication	-	eNB Direct Information Transfer	-
UE Context Release Request	**Yes**	MME Direct Information Transfer	-
DownlinkS1 CDMA2000 Tunneling	- out of scope	eNB Configuration Transfer	-
Uplink S1 CDMA2000 Tunneling	- out of scope	MME Configuration Transfer	-
UE Capability Info Indication	**Yes~**	Cell Traffic Trace	-
eNB Status Transfer	-	Downlink UE Associated LPPa transport	-
MME Status Transfer	-	Uplink UE Associated LPPa Transport	-
Deactivate Trace	-	Downlink Non UE Associated LPPa Transport	-

3GPP Implementation Status
NAS EMM Specific Procedures

Specific procedures	Status
combined attach	-
attach	**Yes**
detach	**Yes**
combined detach	-
normal tracking area updating	-
combined tracking area updating	-
periodic tracking area updating	- (Soon from from Radisys with Facebook collaboration)

3GPP Implementation Status
NAS EMM Connection Management Procedures

EMM connection management procedures	Status
service request	- Soon, from Radisys with Facebook collaboration
paging procedure	- Soon, from Radisys with Facebook collaboration
transport of NAS messages (SMS)	-
generic transport of NAS messages	-

3GPP Implementation Status
NAS EMM Common Procedures

Common procedures	Status
GUTI reallocation	-
authentication	**Yes**
security mode control	**Yes**
identification	**Yes**
EMM information	**- Soon, from Radisys with Facebook collaboration**

3GPP Implementation Status
GTPv2 Path Management Messages

Messages	Status
Echo Request/Response	-
Version Not Supported Indication	-

3GPP Implementation Status
GTPv2 Tunnel Management Messages

Messages	Status
Create Session Request/Response	**Yes S11**
Create Bearer Request/Response	**Yes S11**
Bearer Resource Command/Failure Indication	-
Modify Bearer Request/Response	-
Delete Session Request/Response	**Yes S11**
Delete Bearer Request/Response	-
Downlink Data Notification/Acknowledge/Failure Indication	- **Req Paging**
Delete Indirect Data Forwarding Tunnel Request/Response	-
Modify Bearer Command/Failure indication	-
Update Bearer Request/Response	-
Delete Bearer Command/Failure indication	-
Create Indirect Data Forwarding Tunnel Request/Response	-
Release Access Bearers Request/Response	**Yes S11**
Stop Paging Indication	-
Modify Access Bearers Request/Response	-

3GPP Implementation Status
GTPv2 Mobility Management Messages

- Many message for S10

Messages	Status
Forward Relocation Request/Response	-
Forward Relocation Complete Notification/Acknowledge	-
Context Request/Context Response	-
Identification Request	-
Forward Access Context Notification/acknowledge	-
Detach Notification/Acknowledge	-
Change Notification Request/Response	-
Relocation Cancel Request/Response	-
Configuration Transfer Tunnel	-
RAN Information Relay	-

3GPP Implementation Status
S6a Location Management Procedures

Procedures	Status
Update Location	Yes
Cancel Location	-
Purge UE	-

3GPP Implementation Status
S6a Subscriber Data Handling Procedures

Procedures	Status
Insert Subscriber Data	-
Delete Subscriber Data	-

3GPP Implementation Status
S6a Misc Procedures

Authentication Procedures	Status
Authentication Information Retrieval	Yes

Fault Recovery Procedures	Status
Reset	-

Notification Procedures	Status
Notification	-

3GPP Implementation Status
S6a Misc Procedures

Authentication Procedures	Status
Authentication Information Retrieval	Yes

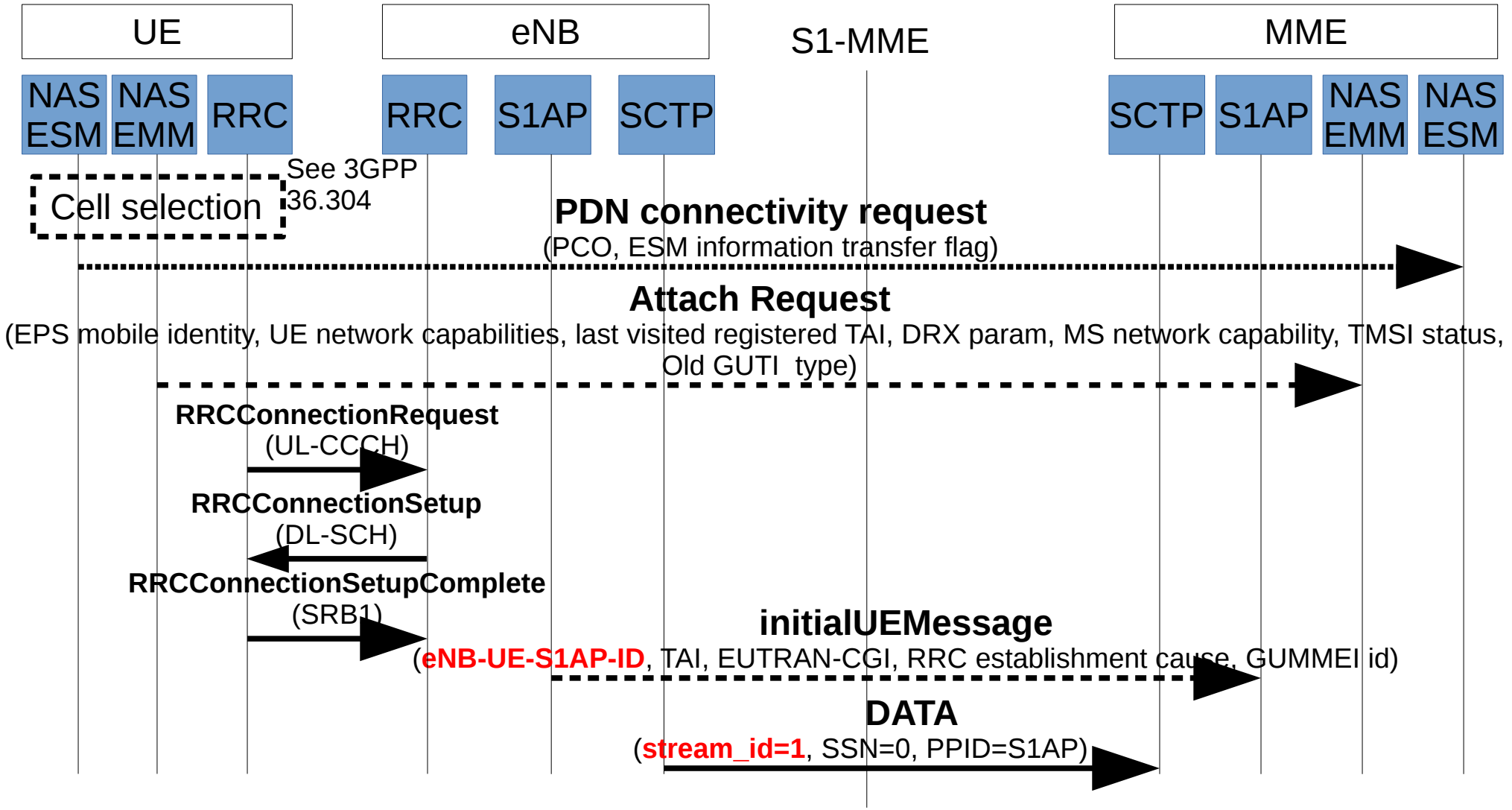
Fault Recovery Procedures	Status
Reset	-

Notification Procedures	Status
Notification	-

MME Internal interfaces

UE attach procedure

UE Attach initialUEMessage



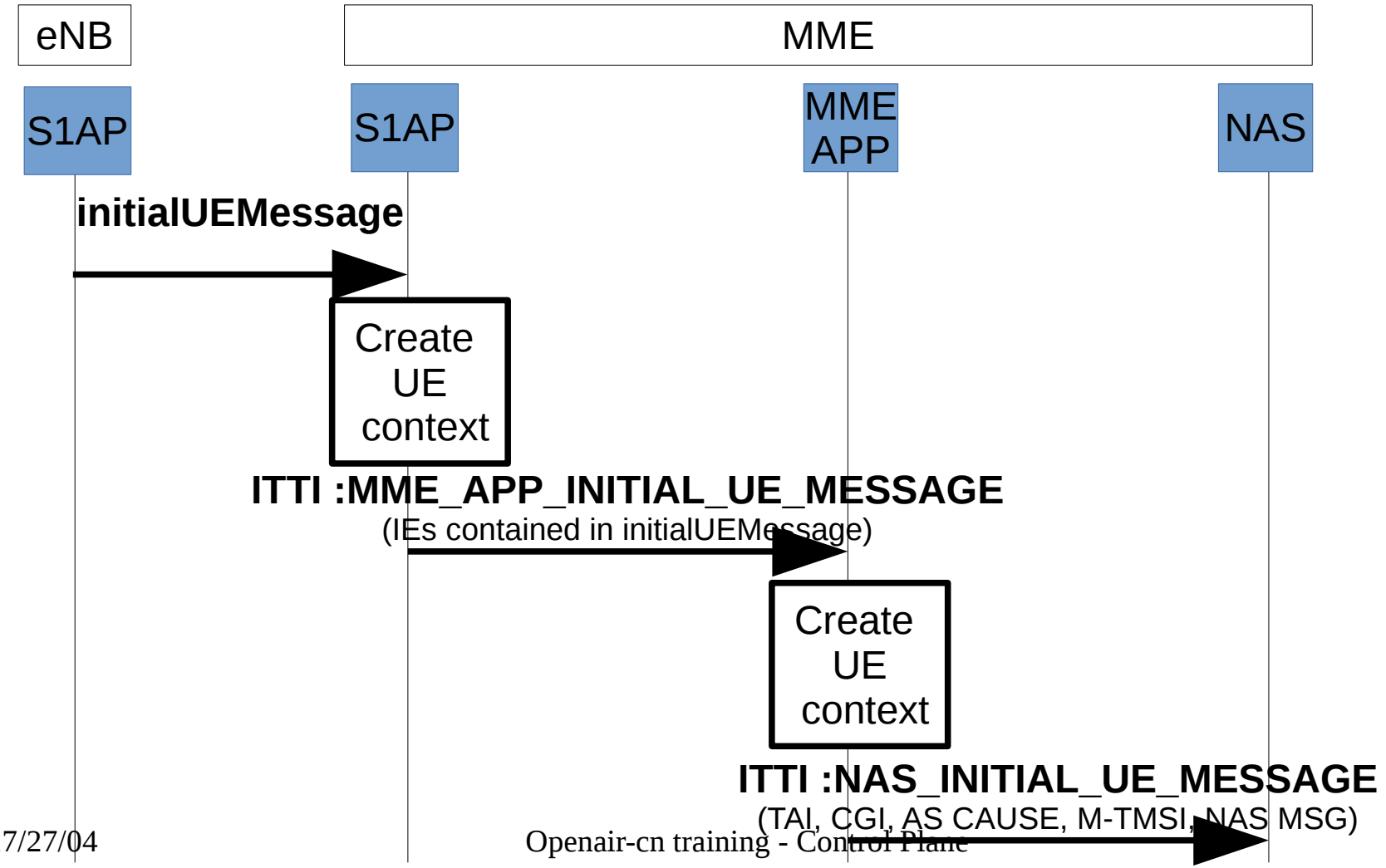
UE Attach – initialUEMessage - S1AP internals

- When S1AP receives a request from an unknown UE, it creates a UE entry and add it in a collection in the enb_description_t struct.

```
typedef struct ue_description_s {
  struct enb_description_s *enb;  ///< Which eNB this UE is attached to
  enum s1_ue_state_s        s1_ue_state;       ///< S1AP UE state
  enb_ue_s1ap_id_t          enb_ue_s1ap_id:24;///< Unique UE id in eNB
  mme_ue_s1ap_id_t          mme_ue_s1ap_id; ///< Unique UE id in MME

  sctp_stream_id_t          sctp_stream_recv; ///< eNB -> MME stream
  sctp_stream_id_t          sctp_stream_send; ///< MME -> eNB stream

  s11_teid_t                s11_sgw_teid;

  /* Timer for procedure outcome issued by MME that should be answered*/
  long outcome_response_timer_id;
} ue_description_t;
```

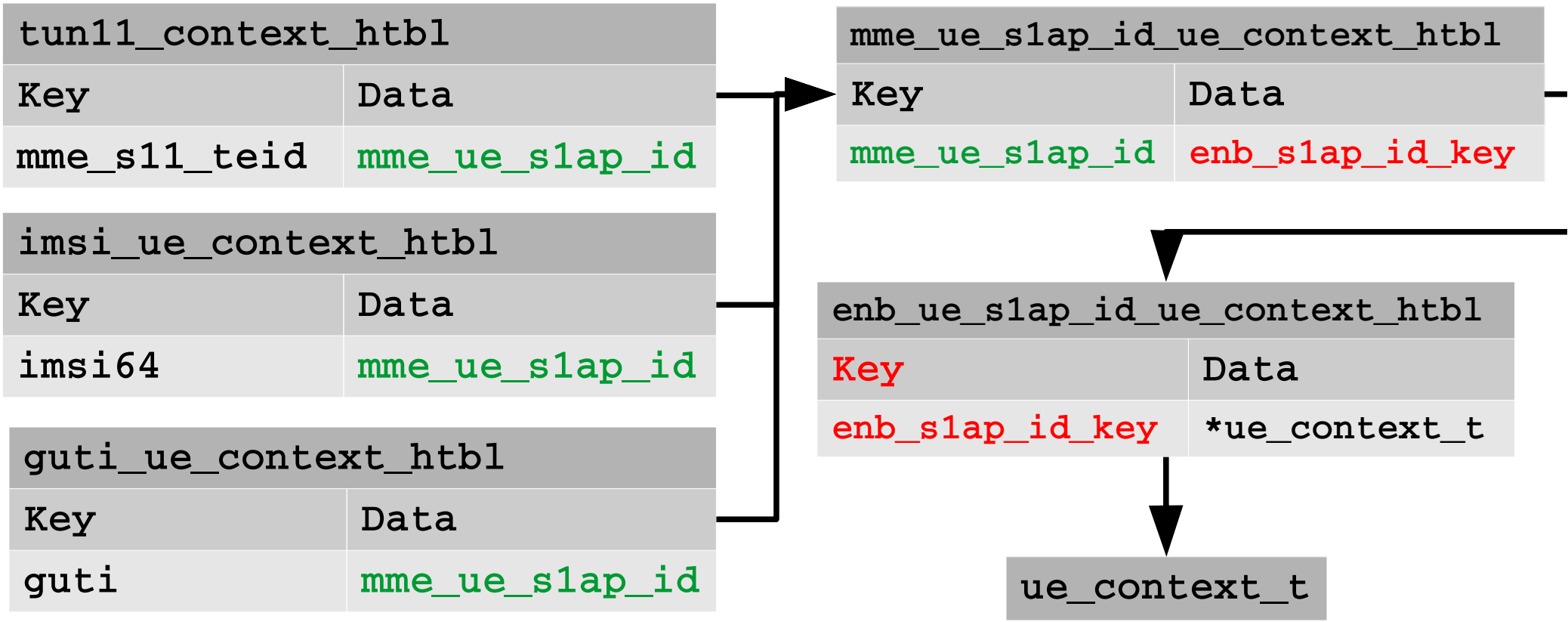
- mme_ue_s1ap_id is set to INVALID_MME_UE_S1AP_ID
- sctp_stream_send is electd from range [1..NB max instreams]
- s1_ue_state set to S1AP_UE_WAITING_CSR (Create Session Request)

UE Attach – initialUEMessage



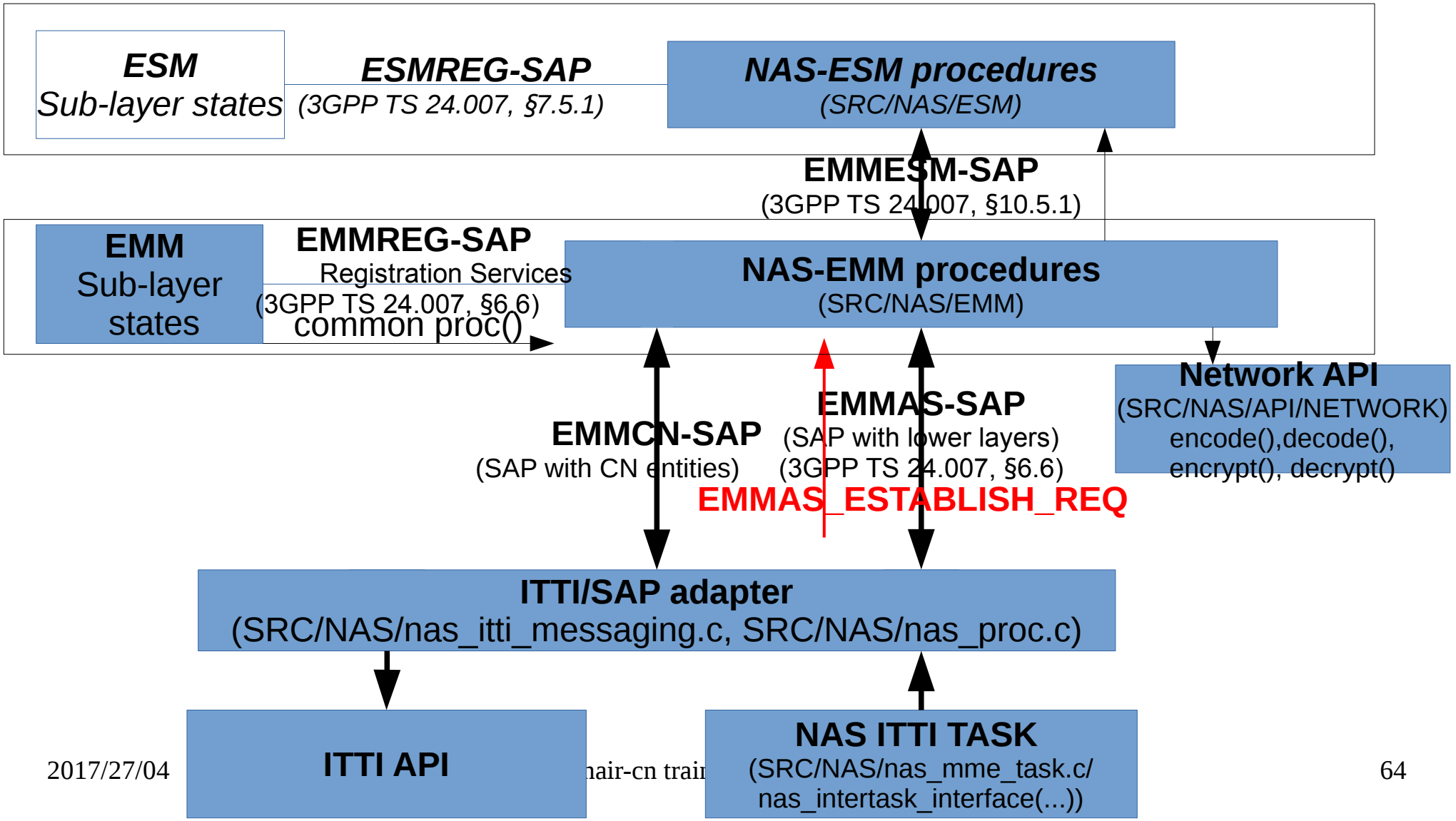
UE Attach – initialUEMessage
MME_APP internals

- Then the UE context is referenced by collections with different keys (mme_app_ue_context.c/mme_insert_ue_context()), because each entity in MME has its own key :

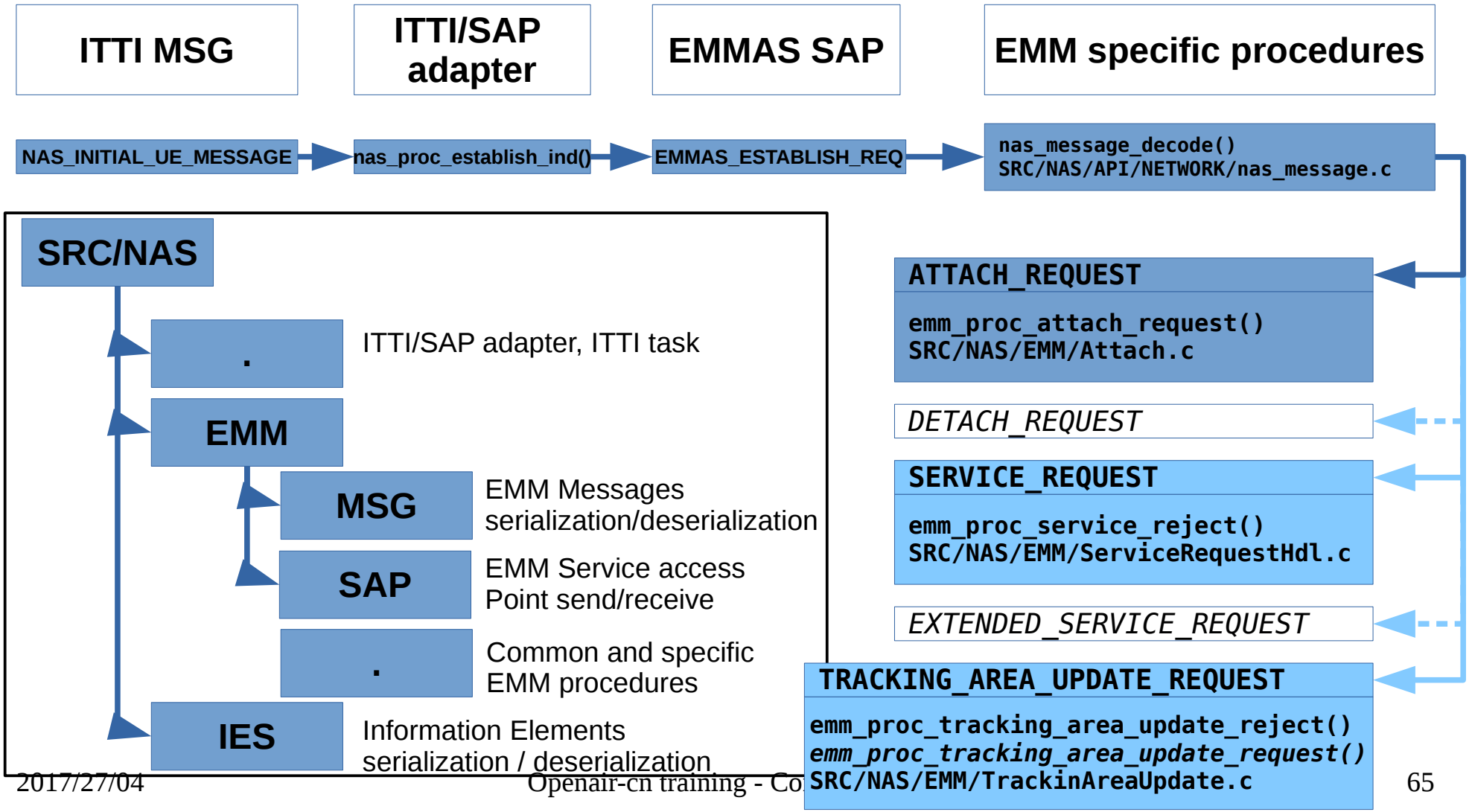


UE Attach – initialUEMessage

- The reception of the message NAS_INITIAL_UE_MESSAGE triggers the message EMMAS_ESTABLISH_REQ



ITTI NAS_INITIAL_UE_MESSAGE to EMM specific procedures



Few words on ITTI S6A task

- **Role : ITTI to freeDiameter API adapter**

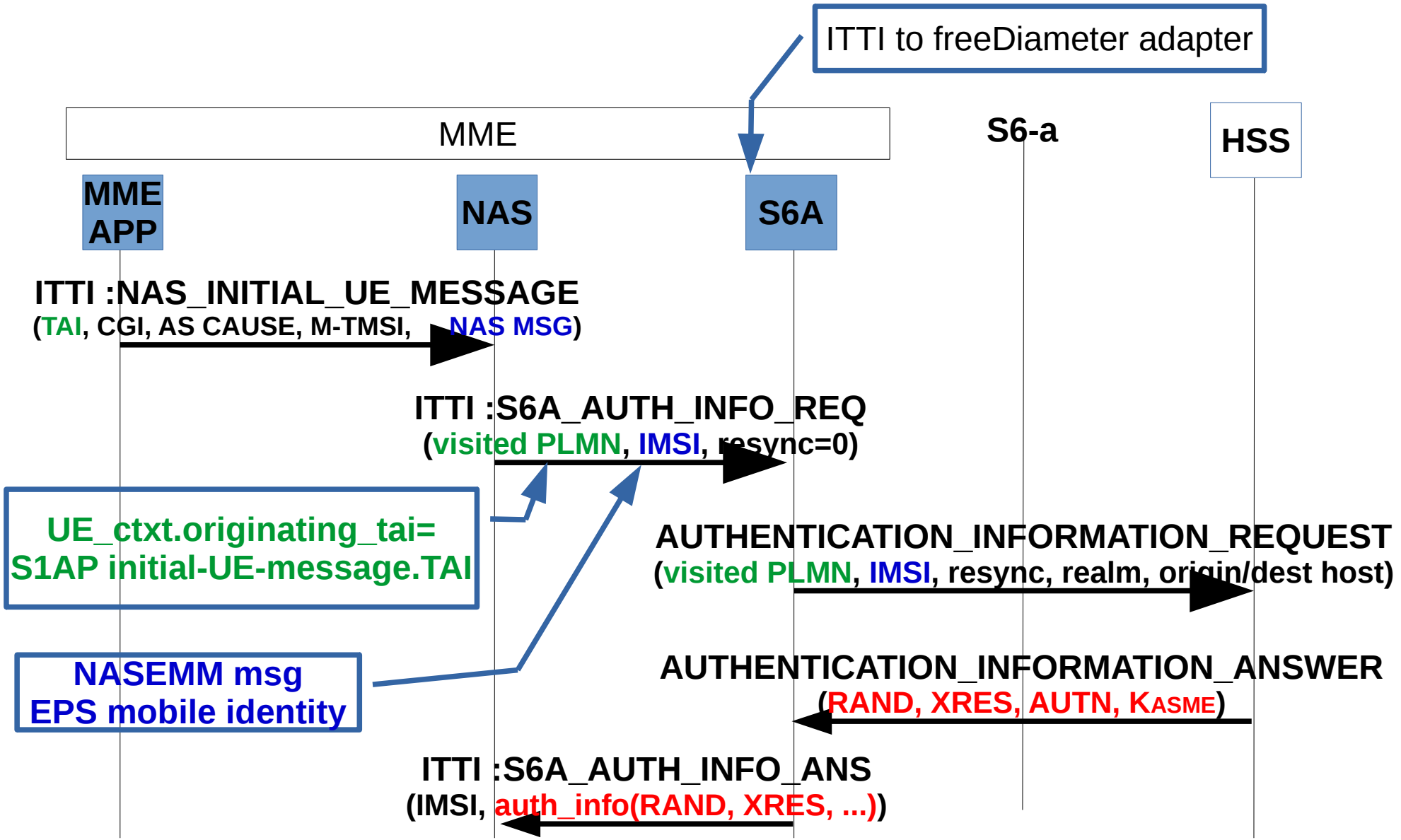
Use freeDiameter library, BSD 3 clauses licence (http://www.freediameter.net/hg/freeDiameter/archive/1.2.0.tar.gz)

- **Implemented send/receive messages:**
  - Authentication Information Request/Answer
  - Update Location Request/Answer
- **How to send a message:**
  (ex :SRC/S6A/auth_info.c/s6a_generate_authentication_info_req(...))
  - **Create message:** fd_msg_new(...)
  - **Setting Attribute-Value Pairs:** fd_msg_avp_new(...), fd_msg_avp_setvalue(...), fd_msg_avp_add(...).
  - **Sending message :** fd_msg_send(...)

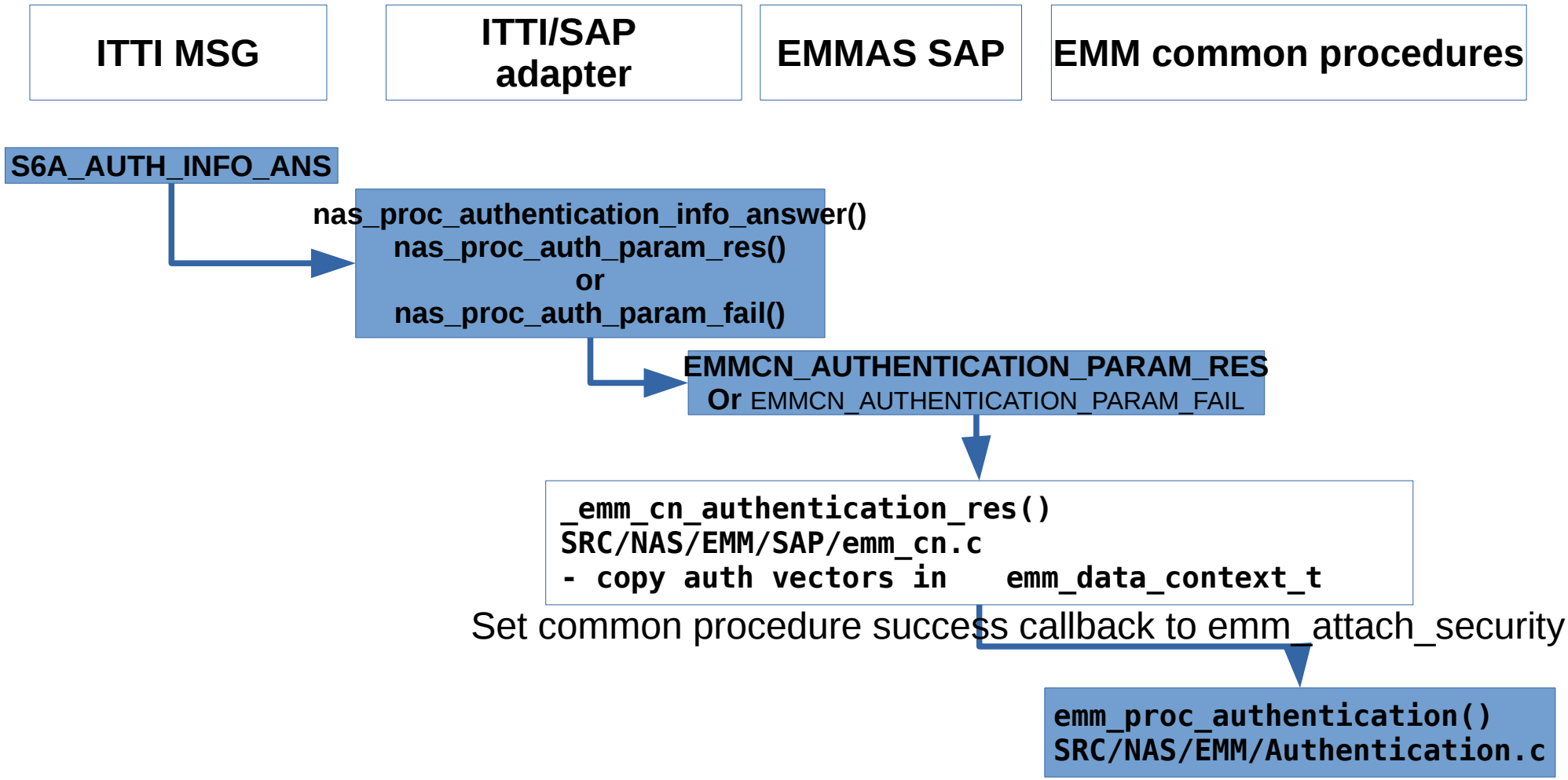
Few words on ITTI S6A task

- **How to receive a message: use registered callbacks:**
  - **Register callback:** fd_disp_register(…) for each message, passing incoming messages to extensions registered callbacks. Ex: SRC/S6A/s6a_dict.c/s6a_fd_init_dict_objs(...)
  - **Get message:** fd_msg_answ_getq(…). Ex: SRC/S6A/auth_info.c/s6a_aia_cb(...)
  - **Get Attribute-Value Pairs:** fd_msg_search_avp(...), fd_msg_avp_hdr(...), fd_msg_browse(...)

AUTHENTICATION MSC in core network



ITTI S6A_AUTH_INFO_ANS to NAS-EMM

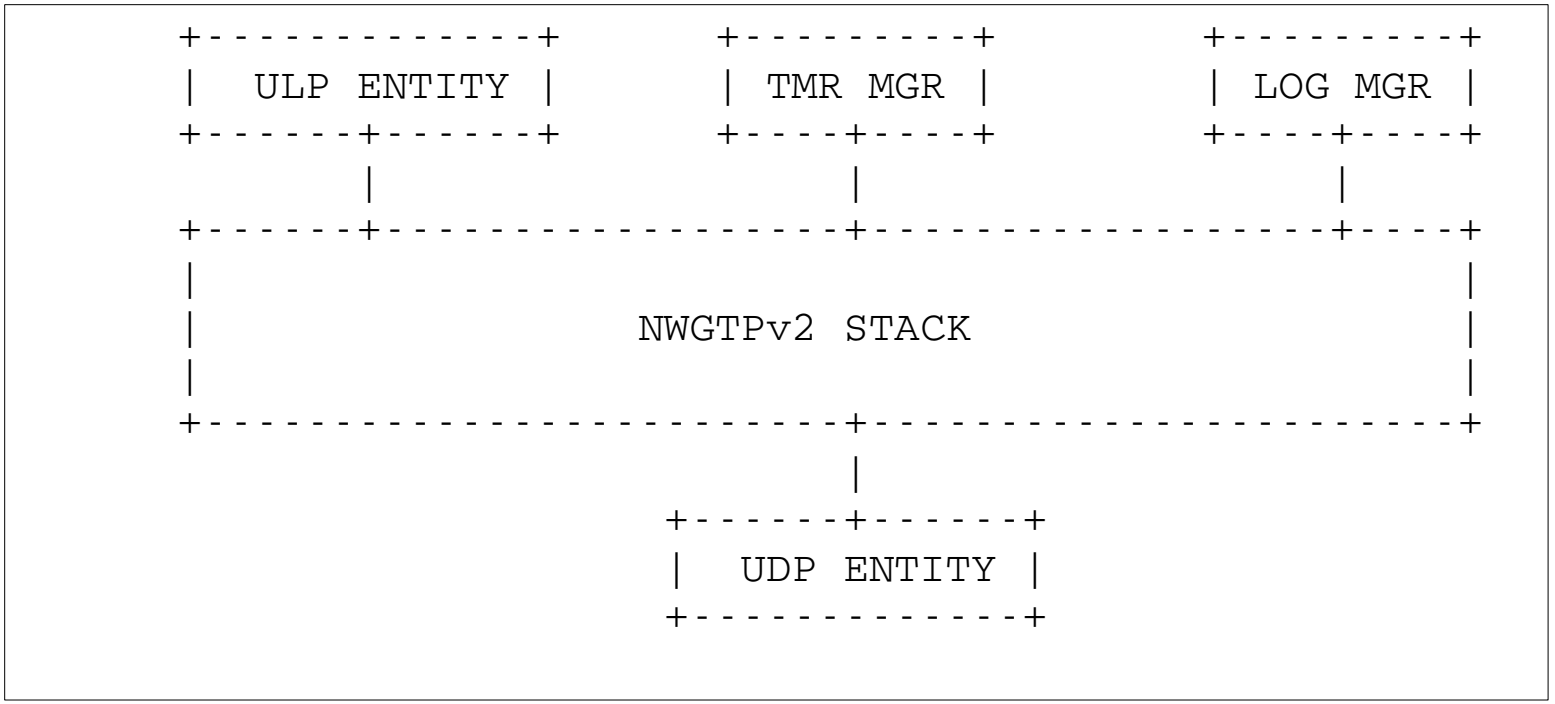


S11 task

- **Role : ITTI/GTPv2-C adapter**

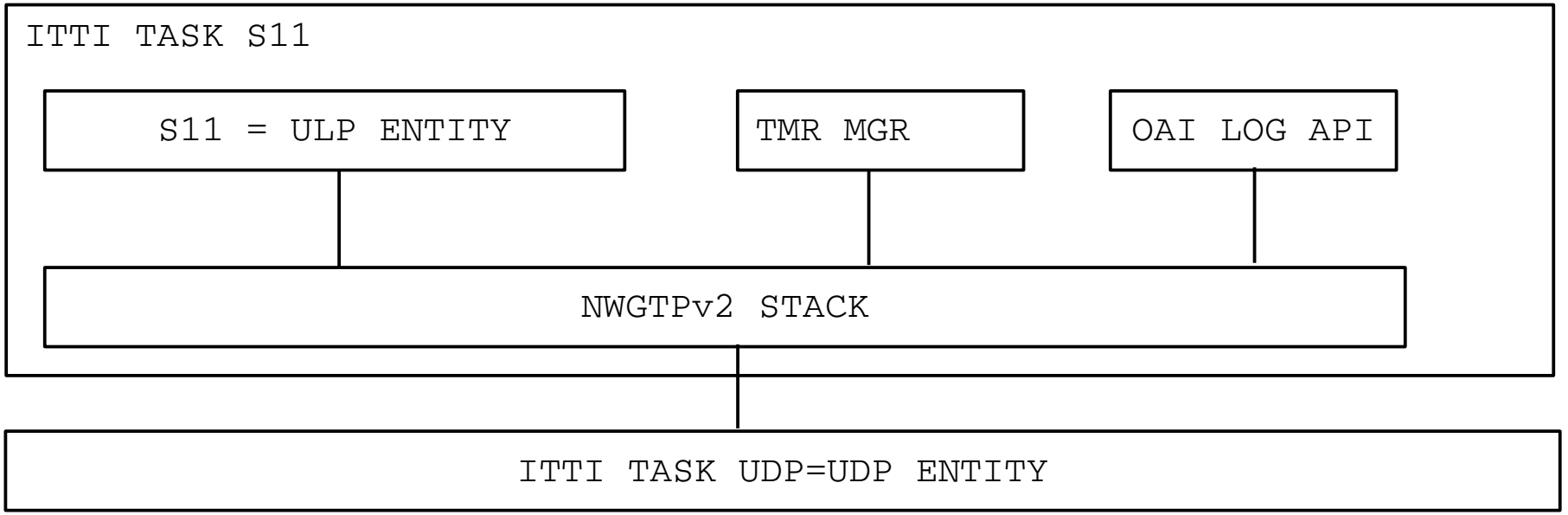
Use nwGtpv2c library, BSD 3 clauses licence

- **Out of the box architecture** (see nwGtpv2c readme file)



Few words on S11 task

- **Final architecture**



Few words on S11 task

- **How to send a message:**
  (ex :SRC/S11/s11_mme_sesson_manager.c/s11_mme_create_session_request(...))
  - **Create message:** nwGtpv2cMsgNew(...)
  - **Adding a already serialized IE:** nwGtpv2cMsgAddIe(…)
  - Otherwize write the IE de/serializer functions, ex s11_pdn_type_ie_set()/s11_pdn_type_ie_get(), etc
  - **Sending message :** nwGtpv2cProcessUlpReq(...)

Few words on S11 task

- **How to read a received message: use msg parser API**

```
s11_mme_handle_create_session_response(NwGtpv2cStackHandleT * stack_p,
NwGtpv2cUlpApiT * pUlpApi) {
...
  //Create a new message parser
  rc = nwGtpv2cMsgParserNew (*stack_p, NW_GTP_CREATE_SESSION_RSP,
s11_ie_indication_generic, NULL, &pMsgParser); // default IE indication

  // Cause IE
  rc = nwGtpv2cMsgParserAddIe (pMsgParser, NW_GTPV2C_IE_CAUSE,
NW_GTPV2C_IE_INSTANCE_ZERO, NW_GTPV2C_IE_PRESENCE_MANDATORY,
      s11_cause_ie_get, &resp_p→cause);
  // same for all needed IEs
  …
  // Run the parser
  rc = nwGtpv2cMsgParserRun (pMsgParser, (pUlpApi->hMsg), &offendingIeType,
&offendingIeInstance, &offendingIeLength);

  // IEs are in resp_p→cause, ...

  // Clean
  rc = nwGtpv2cMsgParserDelete (*stack_p, pMsgParser);
  rc = nwGtpv2cMsgDelete (*stack_p, (pUlpApi->hMsg));
```

Questionable

Few words on S11 task

- **How to read a message**

```
  // Set ULP entity
  ulp.hUlp = (NwGtpv2cUlpHandleT) NULL;
  ulp.ulpReqCallback = s11_mme_ulp_process_stack_req_cb;
  DevAssert (NW_OK == nwGtpv2cSetUlpEntity (s11_mme_stack_handle, &ulp));
```

```
static NwRcT s11_mme_ulp_process_stack_req_cb(NwGtpv2cUlpHandleT hUlp,
  NwGtpv2cUlpApiT * pUlpApi)
{
  int                                     ret = 0;
  switch (pUlpApi->apiType) {
  case NW_GTPV2C_ULP_API_TRIGGERED_RSP_IND:
    OAILOG_DEBUG (LOG_S11, "Received triggered response indication\n");

    switch (pUlpApi->apiInfo.triggeredRspIndInfo.msgType) {
    case NW_GTP_CREATE_SESSION_RSP:
      ret = s11_mme_handle_create_session_response (&s11_mme_stack_handle, pUlpApi);
      Break;
     ...
```

What is missing

Missing/ToDo

- MME
  - More test cases/scenarios for MME scenario player.
  - S1AP causes, S11 causes.
  - **Rework NAS ESM sublayer**, report true ESM causes, most of all modification procedures.
  - NAS EMM procedures : TAU, Service request, Paging.
  - X2 HandOver.
- SPGW
  - SGW/PGW split not considered.
  - Better integration of nwGTPv2-C.
    - Presence of IEs (Mandatory/conditional/optional)
  - Simplify internal procedures in SPGW.

Missing

- Tools
  - 3GPP requirements
    - Spread its use ? and exploit traces to enforce testing scenarios ?
  - HSS import/populate subscriber tool, instead of relying on SQL import/export.
- Organization
  - Scale for wider contributions
    - Tools and Process (inputs from Radisys, example of Redmine, etc).
    - Test tools, continuous integration testbed, set a reference testbed for control plane and userplane.
- Documentation
  - Doxygen still to do.
  - Example modelize with UML tool like papyrus, etc.
- NB-IoT Rel 14 compliance

Thank You