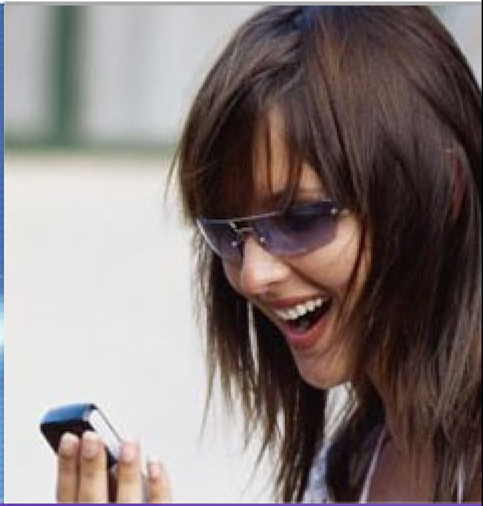
SLAM-TL1 FSM



Alpha Zhang

9/10/2010

Outline

- TL1
- Important CB
- SLAM-TL1 FSM

TL1

1). TL1 command sent to Slam, following parts should be contained:

```
 -- transaction ID(transId)
     TL1GenXtag(transId)
 -- command type (tl1Cmd)
     such as SM_ENT_REQUEST, SM_ED_REQUEST …
 -- value (rowKey, colArray)
 -- table ID
```

2). Slam Receive from TL1 agent via GA

```
-- Get data and put it to  m_buffer

-- pst initialization

        pst.srcEnt = ENTGA;

        pst.event=SM_CMD_CLASS_INT

        pst.spare1=NULL_EVENT_INT

-- SPstTsk(&pst, mBuf) Post message to system services
```

Important CB

```
typedef struct cmFsmCp
{
   CmEvalEvDepFp   evalEvDep;
   CmFsmEvFp       fsmEvFp;
   U32             maxInst;
   Region          region;
    Pool           pool;
    U8             numStates;
    U8             numEvents;
    CmFsmStateInfo *states;
    CmFsmEventInfo *events;
    CmFsmRow       *fsmMt;     /* FSM state matrix */
    CmHashListEnt   fsmCpHlEnt; /* global FSM control point hash list entry */
    CmHashListCp    instHlCp;   /* FSM instance hash list control point */
    CmQCp          instQCp;    /* FSM instance queue control point */
} CmFsmCp;
```

Important CB

CmFsmCp: (slam/base/sl_tl1.c)

cmFsmInstInit(): initialize FSM CP and insert it into slCb

```
fsmCp->evalEvDep    = slTl1FsmEvalDep;
fsmCp->fsmEvFp      = slTl1FsmDr;              slTl1Func
fsmCp->maxInst      = TELICA_MAX_TL1_1_INST;
fsmCp->numStates    = SLTL1_ST_MAX;
fsmCp->numEvents    = SLTL1_EV_MAX;
fsmCp->states       = slTl1States;// 二维数组首 "SLTL1_ST_IDLE",
fsmCp->events       = slTl1Events;//SLTL1_EV_TL1_ENT
fsmCp->fsmMt        = &slTl1_1_FsmMt[0][0];
fsmCp->region       = slCb.region;
fsmCp->pool         = slCb.pool;
fsmCp->entity       = ENTSLM;
```

Example:

```
CmFsmRow slTl1_1_FsmMt[SLTL1_ST_MAX][SLTL1_EV_MAX] =
{
   /* SLTL1_ST_IDLE */
  {
    {slTl1_start_lower,    SLTL1_ST_AWT_X10_DONE},  /* SLTL1_EV_TL1_ENT
```

Alcatel·Lucent

Important CB

```
typedef struct cmFsmEnt
{    U16          lastEvt;     /* last event received */
   U16          lastState;   /* last state */
   U16          state;       /* current state */
   U32          instId;      /* instance Id for this FSM instance */
   CmFsmCp       *fsmCp;      /* FSM control point */
   CmQCp         evtQCp;      /* event queue */Q
   CmHashListEnt instHlEnt;   /* FSM instance hash list entry */
   CmQEnt       instQEnt;    /* FSM instance queue entry */
   DateTime      timestamp;   /* create time, to order across related FSMs */
   CmTimer       timers[CM_FSM_MAX_TIMERS];
    Pst              pst;         /* pst stuct for the current event, if any */
   Buffer        *mBuf;       /* mBuf for the current event, if any */
} CmFsmEnt;
```

Alcatel·Lucent

Important CB

New control block: (slam/base/sl_tl1.c)

1). SlTl1TransIdCb

```
slGetNewTransIdCb(&transIdCb, transId);
 typedef struct slTl1TransIdCb
 {
          TsTransId      tl1TransId;
          U32             fsmInstId;
          CmHashListEnt tl1TransIdHl;
  } SlTl1TransIdCb;
```

2). slGetNewTl1Cb (&slTl1Cb, transId);

SlTl1Cb ----- It is context of TL1 command

Important CB

```
typedef struct slTl1Cb   {

U8             lmTl1Evt;
TsTableId      tableId;
TsTransId      transId;
TsSmiRowKey    *rowKey;
TsSmiColArray *colArray;
TL1_CMD_BUF_t tl1Cmd;
CmFsmEnt       fsmEnt;
U32            userState;
U32            userEvent;
void           *userVar;
SlTransCb      slTransCb;   /*slFindEvMt(), slGetNextTl1FsmInst () */
ClamData       clamData[TELICA_MAX_CLAM_ID+1];
U16clamCtr;
U8             addClamIdFlg;
U32            peerFsmInst;
Bool           replicatedFlag;
Bool           reTxFlag;
…
} SlTl1Cb;
```

→ Receive from TL1 agent

→ TL1 FSM variant

Alcatel·Lucent

Important CB

3). slFindEvMt(slTl1Cb);

```
typedef struct lmEvtMt
{
    U16       mtId;
    S8        *mtStr;
    LmKey     lmkey[LM_MAX_MT_KEYS];
    U8        rollBack;
    U8        numLevels;
    U16       mt[LM_MAX_LEVELS][LM_MAX_BROADCAST];
    slTl1Fp   slTl1Func;
    U8        lmTl1Class;
…
} LmEvtMt;
```

$\longrightarrow$ The key of Event Matrix

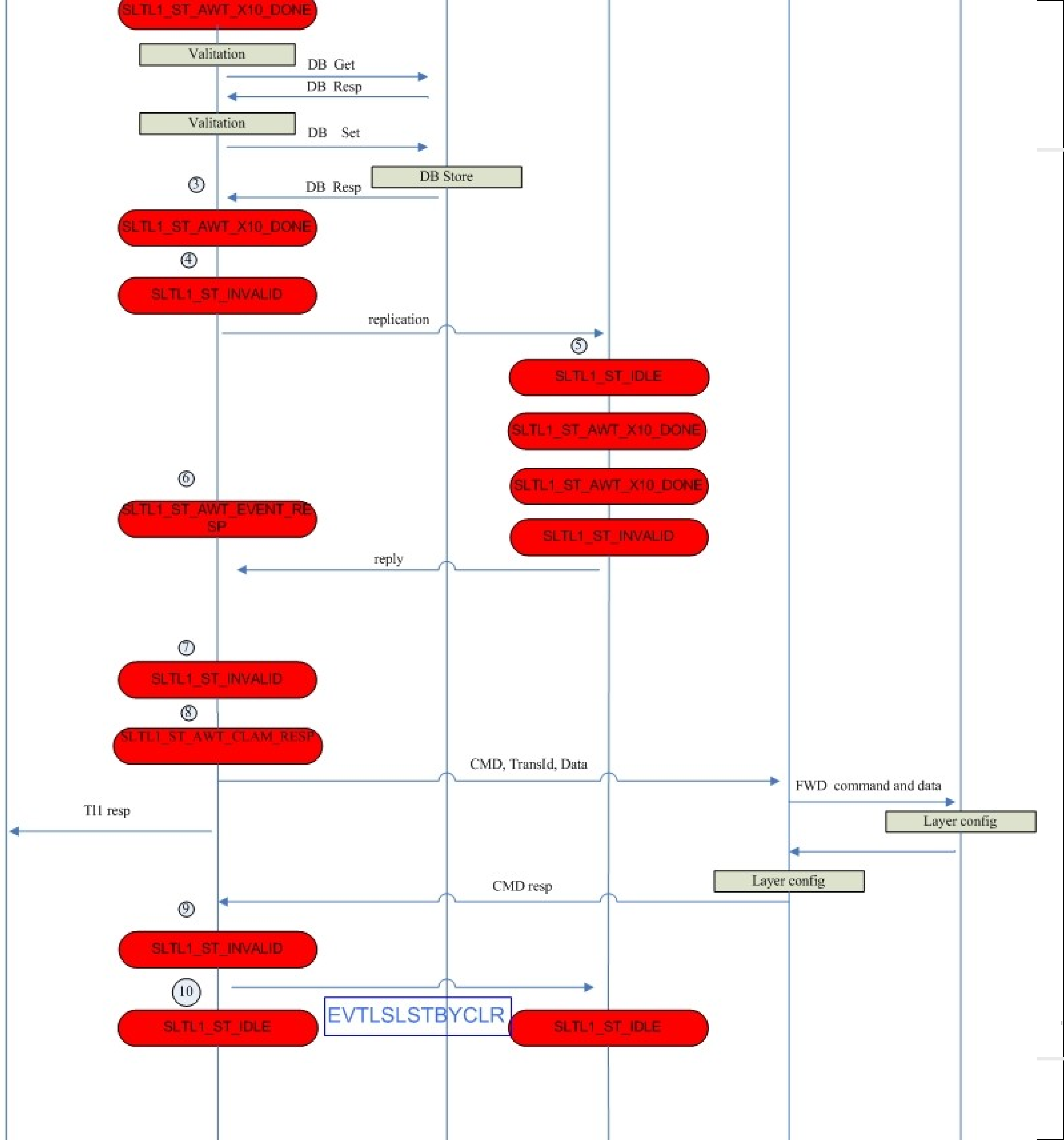
$\longrightarrow$ The entrance function

All the SLAM Event Matrixes are in lm_bdy1.c

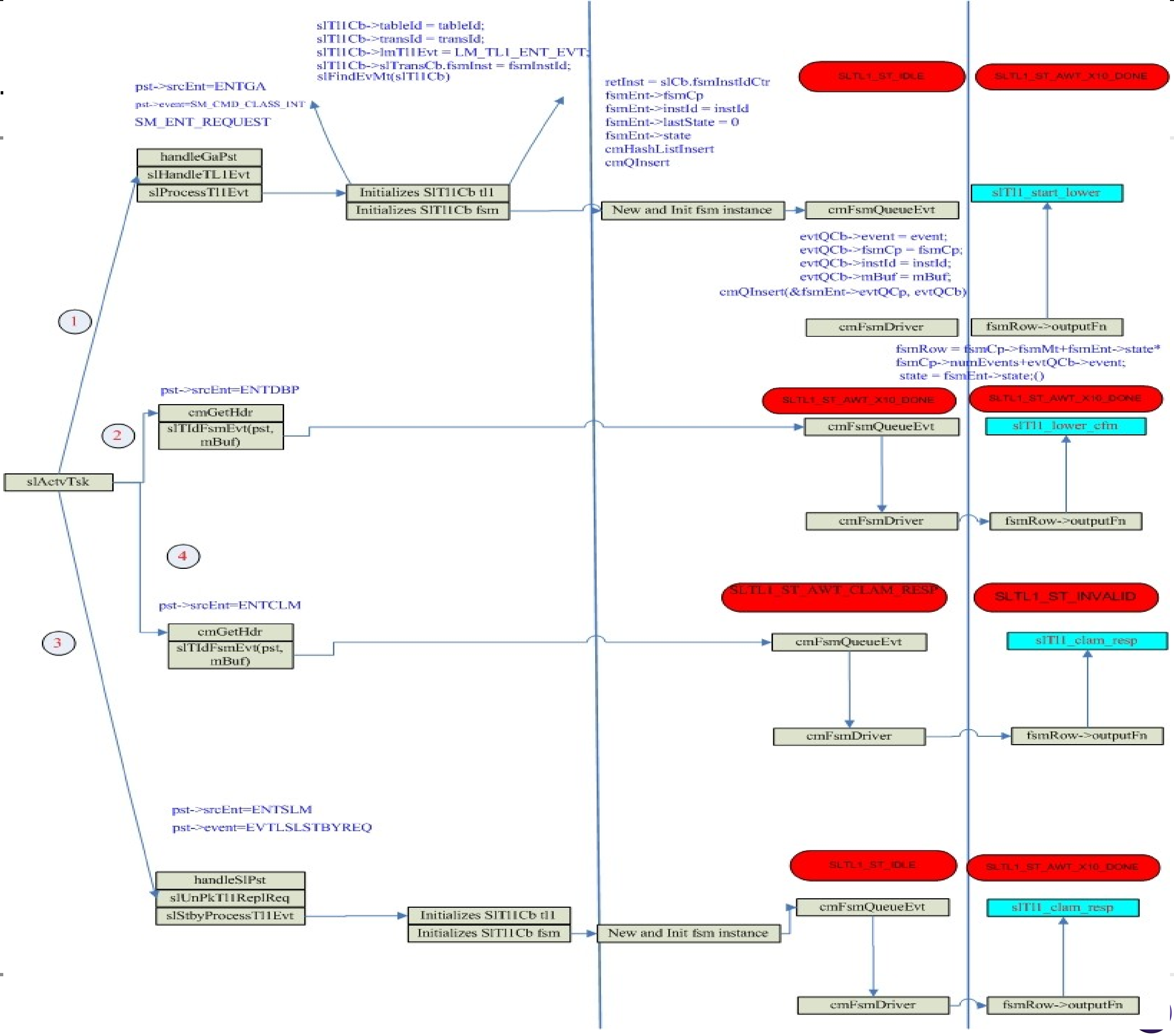
SLAM-TL1 FSM

- slActvTsk()
  - handleGaPst()
    - slHandleTL1Evt()
      - slProcessTl1Evt() transfer tl1cmd to event (tl1FsmEvt)
        - slGetNewTransIdCb()→ Get new Transaction ID
        - slGetNewTl1Cb() → Get slTl1Cb
        - slTl1CheckAllClams()→ Check clam status
        - slFindEvMt() → Find corresponding LmEvtMt
        - slGetNextTl1FsmInst() → Get new FSM instance ID
        - cmFsmInstInit() → Init FSM
        - cmFsmQueueEvt() → Start TL1 SLAM FSM(tl1FsmEvt)

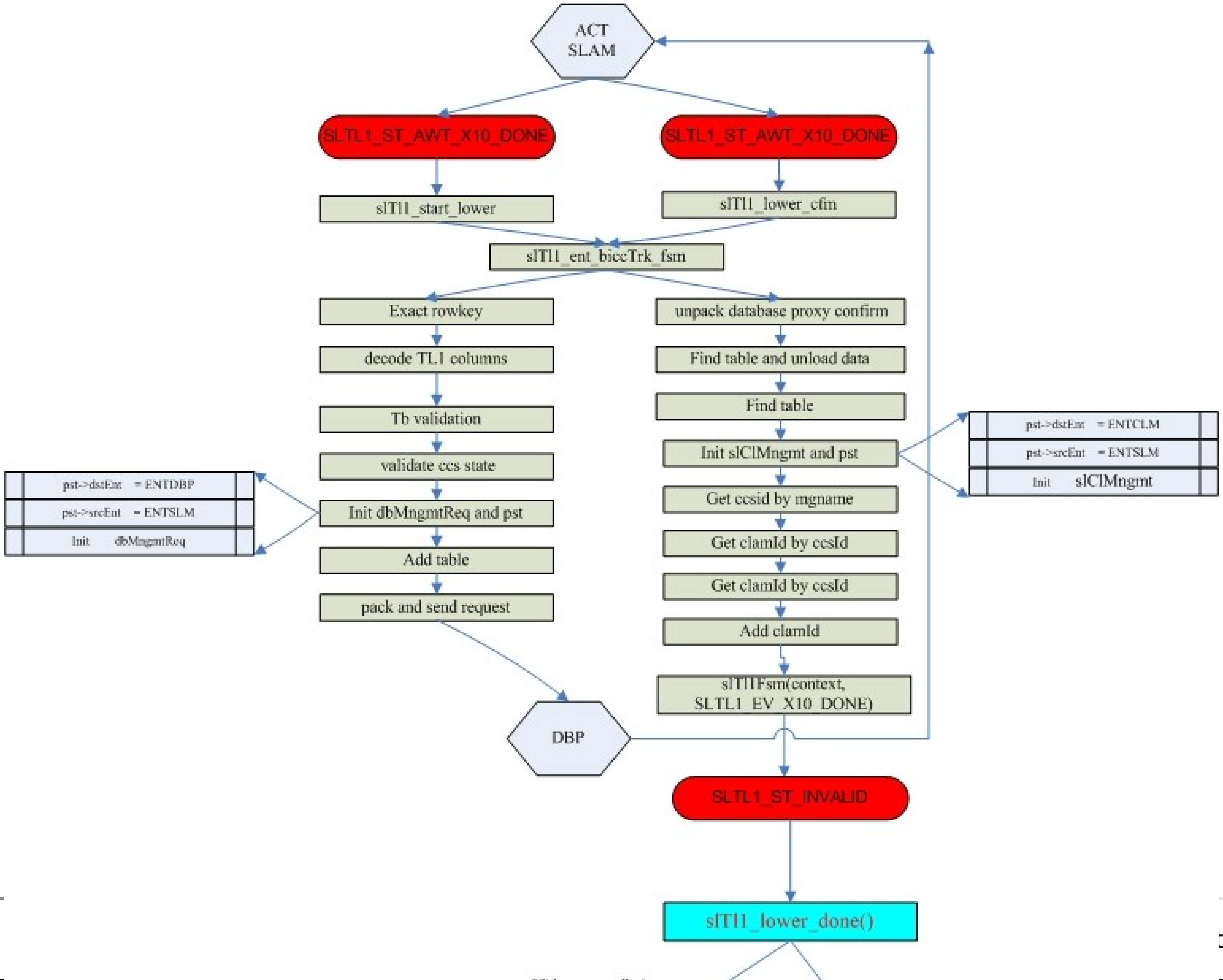
SLAM-TL1 FSM


SLTL1_ST_AWT_X10_DONE
Valitation
DB Get
DB Resp
Valitation
DB Set
DB Store
③
DB Resp
SLTL1_ST_AWT_X10_DONE
④
SLTL1_ST_INVALID
replication
⑤
SLTL1_ST_IDLE
SLTL1_ST_AWT_X10_DONE
SLTL1_ST_AWT_X10_DONE
⑥
SLTL1_ST_AWT_EVENT_RESP
SLTL1_ST_INVALID
reply
⑦
SLTL1_ST_INVALID
⑧
SLTL1_ST_AWT_CLAM_RESP
CMD, TransId, Data
FWD command and data
Tl1 resp
Layer config
Layer config
CMD resp
⑨
SLTL1_ST_INVALID
⑩
EVTLSLSTBYCLR
SLTL1_ST_IDLE
SLTL1_ST_IDLE

SLAM-


slTl1Cb->tableId = tableId;
slTl1Cb->transId = transId;
slTl1Cb->lmTl1Evt = LM_TL1_ENT_EVT;
slTl1Cb->slTransCb.fsmInst = fsmInstId;
slFindEvMt(slTl1Cb)
pst->srcEnt=ENTGA
pst->event=SM_CMD_CLASS_INT
SM_ENT_REQUEST
handleGaPst
slHandleTL1Evt
slProcessTl1Evt
Initializes SlTl1Cb tl1
Initializes SlTl1Cb fsm
retInst = slCb.fsmInstIdCtr
fsmEnt->fsmCp
fsmEnt->instId = instId
fsmEnt->lastState = 0
fsmEnt->state
cmHashListInsert
cmQInsert
New and Init fsm instance
cmFsmQueueEvt
SLTL1_ST_IDLE
SLTL1_ST_AWT_X10_DONE
slTl1_start_lower
evtQCb->event = event;
evtQCb->fsmCp = fsmCp;
evtQCb->instId = instId;
evtQCb->mBuf = mBuf;
cmQInsert(&fsmEnt->evtQCp, evtQCb)
cmFsmDriver
fsmRow->outputFn
fsmRow = fsmCp->fsmMt+fsmEnt->state*
fsmCp->numEvents+evtQCb->event;
state = fsmEnt->state;()
1
2
3
4
slActvTsk
pst->srcEnt=ENTDBP
cmGetHdr
slTIdFsmEvt(pst, mBuf)
SLTL1_ST_AWT_X10_DONE
SLTL1_ST_AWT_X10_DONE
cmFsmQueueEvt
slTl1_lower_cfm
cmFsmDriver
fsmRow->outputFn
pst->srcEnt=ENTCLM
cmGetHdr
slTIdFsmEvt(pst, mBuf)
SLTL1_ST_AWT_CLAM_RESP
SLTL1_ST_INVALID
cmFsmQueueEvt
slTl1_clam_resp
cmFsmDriver
fsmRow->outputFn
pst->srcEnt=ENTSLM
pst->event=EVTLSLSTBYREQ
handleSlPst
slUnPkTl1ReplReq
slStbyProcessTl1Evt
Initializes SlTl1Cb tl1
Initializes SlTl1Cb fsm
New and Init fsm instance
SLTL1_ST_IDLE
SLTL1_ST_AWT_X10_DONE
cmFsmQueueEvt
slTl1_clam_resp
cmFsmDriver
fsmRow->outputFn

SLAM-TL1 FSM


ACT SLAM
SLTL1_ST_AWT_X10_DONE
SLTL1_ST_AWT_X10_DONE
slTl1_start_lower
slTl1_lower_cfm
slTl1_ent_biccTrk_fsm
Exact rowkey
decode TL1 columns
Tb validation
validate ccs state
Init dbMngmtReq and pst
Add table
pack and send request
pst->dstEnt = ENTDBP
pst->srcEnt = ENTSLM
Init dbMngmtReq
unpack database proxy confirm
Find table and unload data
Find table
Init slClMngmt and pst
Get ccsid by mgname
Get clamId by ccsId
Get clamId by ccsId
Add clamId
slTl1Fsm(context, SLTL1_EV_X10_DONE)
pst->dstEnt = ENTCLM
pst->srcEnt = ENTSLM
Init slClMngmt
DBP
SLTL1_ST_INVALID
slTl1_lower_done()

Common function

//slam/base/sl_tl1.c

```
slTl1_start_lower() (SlTl1Cb *context)  // Kicks off the SLAM Unit functions.
{
  …
 (context->slTransCb.mtInfo->slTl1Func)(context); /*call matrix event--LmEvtMt*/
      slTl1Fsm() //Wrapper function to send TL1 FSM an EVENT
  …slSendTl1Resp(context,TSSMI_GEN_ERROR……) //send TL1 response

}
```

Function:
a). Do validation for parameters.
b). Update DB and prepare data which will be forwarded to CLAM.

❖ slDecodeSmiCols() → Decode row to table structure
❖ tbValidateRow() → Validate the columns via the column definitions
❖ slInitPstDbpMngmt () → Init DBP Management
❖ cmTbAddTbl () → Add table
❖ cmPkDbMngmtReq() → Pack data to DBP

Common function

```
slTl1_lower_cfm ()  /*Get data from DB and fill ClamData structure.*/

{
  …
 (context->slTransCb.mtInfo->slTl1Func)(context); /*call matrix event--LmEvtMt*/
  …
}
SlTl1Cb

{   …

    ClamData      clamData[TELICA_MAX_CLAM_ID+1];

}
```

Fetch data from DB and fill this structure

cmUnpkDbMngmtCfm() → Unpk data from dbMngmtCfm
cmTbFndDlnkTbl() → Find the row from cmTbCb
slInitPstClmMngmt() → sets up SlClMngmt struct for a CLAM
cmTbAddTbl() → Add data into table
slAddClamId() → Fill ClamData

Common function

slTl1_lower_done():

when active SP complete transaction with DBP (receive event DBP final confirm). Active SP post event EVTLSLSTBYREQ to standby SP. The corresponding function is **slReplCmdToStandbySlam()**

During this state:

The sameTL1 FSM will be execute on Standby SP.

State change:

when standby SP complete transaction with DBP (receive event DBP final confirm). Standby SP will post event EVTLSLSTBYRSP to active SP The corresponding function is **slSendRespToActSlam()**

Common function

when standby SP complete transaction with DBP (receive event EVTLSLSTBYRSP from standby SP).

slTl1_standby_resp():

1. If active SP receives a bad response from standby SP, standby SP will failed and reboot. Alarm name--SLAM_FAM_REPL_FLD_EVT

2. Send a response to TL1 agent (active SP) slSendTl1Resp()

3. send to corresponding CLAM. slSendToClamList()

State change:

When active SP receives response from all CLAM (SLTL1_EV_ACT_CLAM_RESP)

Common function

slTl1_clam_resp():
On active SP:
When receive event SLTL1_EV_ACT_CLAM_RESP from CLAM.
1. Post event “EVTLSLSTBYCLR” to standby slSendClrTrans()
2. Set TL1 state to SLTL1_ST_IDLE
3. Free all control block slTl1FsmCleanup(context)
    -- SlTl1Cb
    -- table list
    …

On standby SP:
When standby receive event EVTLSLSTBYCLR from active SP:
release all allocated memory
 -- SlTl1Cb
 -- table list
 -- transaction CP

My deepest gratitude goes first to Alpha, my mentor. Without his patience and guidance I won’t learn so much. Second, I want to pay my sincere thanks to Sophia who give me chances to take different tasks and to learn more. Also, I greatly appreciate to Colin, Paolo and all the colleagues in our team who helped me a lot during the work.