ForCES TML Service Primitives

<draft-jhs-forces-tmlsp-00.txt> (to be submitted)

Jamal Hadi Salim, Weiming Wang

hadi@znyx.com, wmwang@mail.zjgsu.edu.cn

Nov 8, 2005

Modeling TML from PL point of view

- Identical to modeling an LFB:
  - TML Attributes
    - PL configures and query TML parameters by
  - TML Events
    - PL gets notifications from
  - TML capabilities
    - PL queries the TML capabilities by
  - TML channel
    - for PL message transmission
- TML Service Primitives (SP) provides standard interfaces for PL access to TML based on above model

Why standardize TML SP?

- PL standardized
- TML individually standardized
- all ForCES PL implementations should be portable across all TMLs (ForCES-PL Section 3.1.2)
- Thus, TML SP should be standardized.
- or, a private TML SP will make the PL implementation also private.

TML SP Design Principles

- hide implementation details regarding TML reliability, security, multicast, congestion control, etc from PL
- avoid TML to access information by reading into PL messages passed.
  - to immunize the SP from the change of PL message encoding

TML SP Design Objectives

- Make TML meet the PL requirements for the TML (ForCES-PL Section 4), like:
  - unicast, multicast and broadcast PL level mechanisms.
  - reliable/unreliable, in-order/agnostic, and timely message delivery requirements
  - Congestion Control
  - Preventing from DoS attack
  - both synchronous and asynchronous operations.
  - events notification to PL.

TML SP

- TML Open
- TML Close
- TML Config
- TML Query
- TML Send
- TML Receive

TML Open

- Syntax:

  fd TMLopen(void);

- Parameters:

  none

- Returns:

  fd: a file descriptor for the TML returned by the TML open

  fd>=0: success for the open, and with the fd returned.

  fd<0: failed for the open, and with the error code returned.

- Service Description:
  - PL connects to the TML by invoking the TML open call
  - TML should be ready for PL msg transportation after the call is executed.
  - specific to TML implementations on how TML should do to meet this

TML Close

- Syntax:

  TMLclose (fd);

- Parameters:

  fd: the file descriptor for the TML to be closed

- Returns:

  SUCCESS: TML close succeeds.

  others: TML close fails, and an error code returned

- Description:
  - PL actively disconnects from the TML by invoking this call
  - TML should be close to PL msg transportation after the call is executed.
  - specific to TML implementations on how TML should do to meet this

TML Config

- Syntax:

```
int TMLconfig (fd,
        int        td,          //input
        void    *tp );          //input
```

- Parameters:

  td: target descriptor for TML attributes to be configured.

  tp: target pointer pointing to attribute values

- Returns:

  SUCCESS: configuration succeeds.

  others: configuration fails, and an error code returned

TML Query

- Syntax:

```
int TMLquery ( fd,
        int       td,          //input
        void     *tp );        //input and output
```

- Parameters:

  td: target descriptor for TML attributes to be queried.

  tp: target pointer to pointing attribute values

- Returns:

  SUCCESS: query succeeds.

  others: query fails, with the error code returned

TML Send

- Syntax:

```
int TMLsend ( fd,
        _u32        msgDestID,          //input
          int       msgType,            //input
          int       msgPrio,            //input
          int       msgLen,             //input
          char      *msgBody,           //input
          int       timeout);           //input
```

- Parameters:
  - msgBody: a pointer to the PL message body in bit strings
  - timeout: to specify blocking or non-blocking mode for the sending, and in blocking mode, to specify a wait time for this sending process.

    timeout = NONBLOCK: non blocking mode
    = BLOCK: forever blocking mode
    = others: blocking for a period of time

TML Receive

- Syntax:

```
int TMLrecv(fd,
            int     *msgLen,        //output
            char    *msgBody,       //output
            int      timeout);      //input
```

- Parameters:
  - msgLen: a pointer pointing to the message length (in bytes) received.
  - msgBody: a pointer pointing to the PL message body (in bit string format) just received.
  - timeout: to specify blocking or non-blocking mode for the primitive; and in blocking mode, to specify a wait time for the primitive.

    timeout = NONBLOCK: non blocking mode
    = BLOCK: forever blocking mode
    = others: blocking for a period of time

TML Attributes (1)

- TML Multicast List
  - Data structure

```
struct McastList {
        _u32        groupID;  //the multicast PL level ID
        _u16        number;   //number of members
        _u32        memberIDlist[]; //the multicast PL level member ID
        Mcastlist *next;      //for more than one Mcastlist operations
                    } *tp;
```

  - Operations
    - td = ADD_MULTICAST_LIST with TMLconfig SP
      - to add a ‘number’ of FE/CE members to multicast groupID. if the groupID did not exist, then to create one.
    - td = DEL_MULTICAST_LIST with TMLconfig SP
      - to delete a ‘number’ of FE/CE members from multicast groupID. If the number is zero, it is to delete the group.
    - td = QUERY_MULTICAST_LIST with TMLquery SP
      - to query the groupID(input) for its members (outputs). If the groupID=0, it means to query all existing Mcastlist.

TML Attributes (1)

- Usage

  int TMLconfig(fd, ADD_MULTICAST_LIST, McastList *tp);

  int TMLconfig(fd, DEL_MULTICAST_LIST, McastList *tp);

  int TMLquery(fd, QUERY_MULTICAST_LIST, McastList *tp);

TML Attributes (2)

- TML Multicast TML specific parameter
  - Data structure

```
struct McastTMLPar {
                _u32      groupID;
                void      *Par;
        McastTMLPar       *next;
                          } *tp;
```

  - Operations
    - td = ADD_MULTICAST_PAR with TMLconfig SP
    - td = DEL_MULTICAST_PAR with TMLconfig SP
    - td = QUERY_MULTICAST_PAR with TMLquery SP
  - Description
    - This attribute is used for PL to configure TML specific parameters for TML multicast for a specific multicast group. For instance, for a UDP multicast TML, the parameter may include the UDP multicast IP address. Note that, the parameter data structure should be further defined by specific TMLs.

TML Events (1)

- TML Failure Event
  - Data structure (callback handle definition for the event)

    int callbackTMLFailure(void);

    tp = callbackTMLFailure;
  - Operations
    - td = SUB_TML_FAIL_EVENT with TMLconfig SP
      - to subscribe the event with a callback handle submitted to TML
    - td = UNSUB_TML_FAIL_EVENT with TMLconfig SP
      - to unsubscribe the event from TML
  - Usage

int TMLconfig(fd, SUB_TML_FAIL_EVENT, callbackTMLFailure);

int TMLconfig(fd, UNSUB_TML_FAIL_EVENT, callbackTMLFailure);

TML Events (2)

- TML Message Arrival Event
  - Data structure

    int callbackTMLMsgArrival(int *msgLen, char *msgBody);
  - Operations
    - td = SUB_TML_MSGARRIVAL_EVENT with TMLconfig SP
      - to subscribe the event with a callback handle submitted to TML
    - td = UNSUB_ TML_MSGARRIVAL _EVENT with TMLconfig SP
      - to unsubscribe the event from TML
  - Usage

    int TMLconfig(fd, SUB_ TML_MSGARRIVAL _EVENT, callbackTMLMsgArrival);
  - By subscribing this event, PL works in the asynchronous receiving mode.

TML Events (3)

- TML control message congestion event
  - Data structure

    int callbackTMLCtlMsgCongest(void);
  - Operations
    - td = SUB_TML_CTLMSGCONGEST_EVENT
    - td = UNSUB_ TML_CTLMSGCONGEST _EVENT

TML Events (4)

- TML redirect message congestion event
  - Data structure

    int callbackTMLRedMsgCongest(void);
  - Operations
    - td = SUB_TML_REDMSGCONGEST_EVENT
    - td = UNSUB_ TML_REDMSGCONGEST _EVENT

TML Events (5)

- TML DoS attack alert event
  - Data structure

    int callbackTMLDoSAlert(void);
  - Operations
    - td = SUB_TML_DOSALERT_EVENT
    - td = UNSUB_ TML_DOSALERT _EVENT

TML Capabilities (1)

- Supported TML Type
  - Data structure

```
struct TMLType {
        _u16    number;   //number of supported TML types
        _u16    tmlType[]; //supported TML type list
        } *tp;
```

tmlType	Definitions
1	RFCxxxx, IPTML1(TCP+UDP)
2	RFCxxxx, IPTML2(TCP+DCCP)
3	RFCxxxx, IPTML3(SCTP+UDP)
4	RFCxxxx, Ethernet TML
5	RFCxxxx, ATM TML
.....	.....

  - Operation
    - td = QUERY_SUPPORTED_TMLTYPE with TMLquery SP

e.g., int TMLquery(fd, QUERY_SUPPORTED_TMLTYPE, TMLTYPE *tp);

TML Capabilities (2)

- Current working TML Type
  - Data structure

    struct CurrentTMLType {

    _u16 currentTMLType;

    } *tp;
  - Operation
    - td = QUERY_CURRENT_TMLTYPE with TMLquery SP
    - CEM/FEM is responsible to specify current working TML type

Theory of Operation (1)

- PL Message Receiving Mode
  - synchronous mode
    Using TMLrecv( ) SP
    with BLOCK and NONBLOCK options
  - asynchronous mode
    Using TML Message Arrival Event
    always nonblock

Theory of Operation (2)

- CE to FEs Multicast
  1. CE forms a multicast list with groupID and FE memberIDs
  2. CE sends the list to the CE TML by TMLconfig SP
  3. CE sends the list to individual FEs by CE PL Configure messages
     - actually to configure the attributes of FE protocol LFBs
  4. FEs sends the list to their TMLs by use of TMLconfig SP.
  5. In the same way as 1 to 4, CE configures the mulitcast specific parameters (like in UDP case, the multicast IP address) to the CE-TML and the FE-TMLs.

– Usually, after above configuration, TMLs are able to construct multicast platforms while without any further TML level information exchanges among the TMLs.

Theory of Operation (3)

- FE to CEs Multicast (the necessity is under discussion)
  1. Master CE forms a multicast list with groupID and CE memberIDs
  2. Master CE sends the list to the FE doing the multicast by CE PL Configure messages
     - actually to configure the attributes of FE protocol LFBs
  3. Master CE sends the list to the CE TML by TMLconfig SP
  4. Master CE sends the list to alternative CEs by means of CE-CE management tools (out of scope)
  5. In the same way as 1 to 4, CE configures the mulitcast specific parameters (like in UDP case, the multicast IP address) to the CE-TML and the FE-TMLs.

– also no need to for TML level message exchanges

Theory of Operation (4)

- SP to meet the TML requirements of reliable/unreliable, in-order/agnostic, and timely message delivery:
  1. PL sends messages to TML by means of TMLsend, in which msgType and msgPrio are explicitly included.
  2. TML should then be able to construct mechanisms to meet the above requirements.
     - Note, PL should not interfere into the TML construction like channels for control msg or for redirect msg, which is implementation specific.

Theory of Operation (5)

- SP to meet the TML requirements of congestion Control and Preventing from DoS attack:
  1. SP provides TMLevent for congestion control and DoS attack alert
  2. PL is then able to participate to some extent in the work to meet above TML requirements.

TML SP be separated from PL in document

- Viewed at PL level, FE TML is quite like a logical functional block, and is actually a sub-block of FE Protocol LFB
  - attributes, events, capabilities, ...
- like the reasons to move FEPO, FEO and TMLs out of PL, a separate document is a better choice.
  - its development is quite independent of the PL
  - may make the PL document tidy, reasonable in length, more focus, and the PL work faster also.

Acknowledgement:

- Research is partly funded by:
  - NSF China (60273061, 60573116)
  - National Hi-Tech R&D Project (2005AA121310)
  - ZJ NSF (RC02063), ZJ Sci&Tec Project (2005C21013)

Thank You!