Capitolo 5

Protocolli di comunicazione

5.1 Scelte progettuali per la comunicazione tra i componenti del sistema

La comunicazione tra host remoti non è possibile senza la definizione di un protocollo, ossia di un insieme di regole per il formato dei messaggi e le modalità del loro invio. In questo capitolo si descrivono dettagliatamente i protocolli di comunicazione utilizzati dalle varie parti del sistema realizzato nell'ambito di questo progetto. La rassegna effettuata costituisce un riferimento indispensabile per la realizzazione dell'interfaccia utente, come si è detto nel capitolo precedente.

I messaggi scambiati devono trasportare una serie di dati che codificano il tipo di richiesta ed eventuali altre informazioni aggiuntive per quel tipo di richiesta. Un modo concettualmente molto semplice e genericamente valido per affrontare un problema di questo tipo è impostare un protocollo di tipo TLV (Type, Length, Value). In questo protocollo i messaggi hanno un

header di dimensione fissa, nel quale sono presenti i campi Type e Length che indicano sia il tipo di richiesta che la lunghezza dell'intero messaggio. Il successivo campo Value, di estensione variabile, può trasportare i dati aggiuntivi eventualmente richiesti. Per quanto questo metodo abbia il pregio di essere facilmente configurabile per essere calato in uno specifico progetto, si è preferito fare ricorso ad un'altra soluzione, che presenta maggiori vantaggi. La scelta è caduta, come si è visto nel capitolo 3, sull'adozione del linguaggio XML per tutta la messaggistica di controllo. La figura 5.1 mostra schematicamente i protocolli la cui descrizione è oggetto dei prossimi paragrafi, indicando per ciascuno il paragrafo di riferimento.

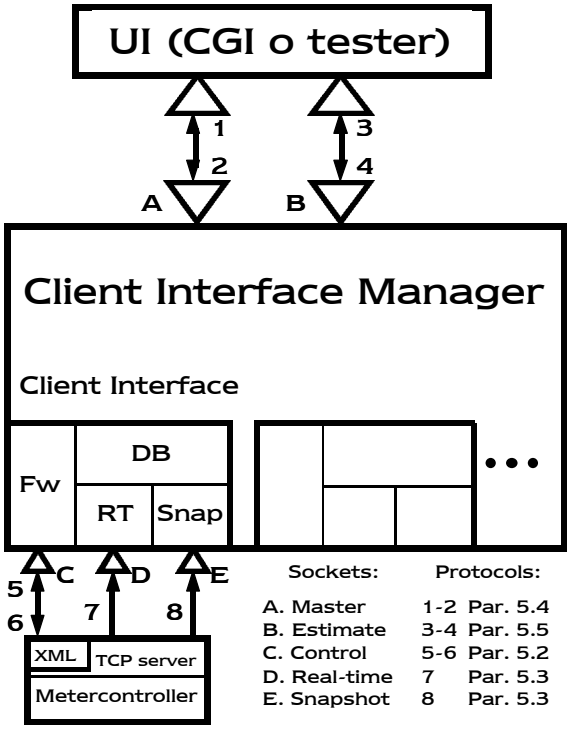

Figura 5.1 – Protocolli di comunicazione del sistema client-server

I protocolli di accesso al sistema sono essenzialmente due: quello utilizzato per le richieste sul master socket e quello per le richieste sullo estimate socket (rispettivamente i numeri 2 e 4 in figura 5.1). Analogamente, i protocolli di replica del sistema sono due: i messaggi di risposta e notifica inviati dal CIM sul master e sullo estimate socket (rispettivamente i numeri 1 e 3). I protocolli di comunicazione sul control socket dal CI al controller (numeri 5 e 6) sono un sottoinsieme di quelli del master socket, ma saranno trattati a parte per maggiore chiarezza espositiva. In questo capitolo si riportano per completezza anche i protocolli di trasferimento dei dati binari sul real-time e sullo snapshot socket (numeri 7 e 8), pur non rivestendo questi ultimi interesse nella realizzazione dell'interfaccia utente.

5.2 Protocolli di comunicazione tra Client Interface e controller

In questo paragrafo si approfondiscono i protocolli utilizzati per il control socket sia dal lato controller che dal lato Client Interface. Essi costituiscono una parte dei protocolli di accesso all'intero sistema, come si vedrà nel paragrafo 5.4, quindi è necessario conoscerli per poter progettare la User Interface.

5.2.1 Richiesta task al controller

Le richieste di esecuzione di task sono inviate dal modulo Client Interface al controller in formato XML. Viene prima inviato un intero indicante la lunghezza del messaggio XML per consentirne la corretta ricezione, quindi il messaggio XML stesso.

Segue l'elenco dei task possibili; i primi due gruppi sono i task che vengono messi in coda in attesa di una conferma (*commit*) che ne avvii l'esecuzione. I restanti vengono eseguiti immediatamente non appena se ne presenta la richiesta.

Task di tipo DB ACCESS (OPCODE 1):

- **Full connection:** consente di scaricare l'intero DB e connettersi alle misure in tempo reale
- **Full snapshot:** consente di scaricare l'intero DB inclusi cache e campione attuale, senza connettersi alle misure in tempo reale
- **Selective connection:** per un LSP, consente di scaricare lo storico e connettersi alle misure in tempo reale; può essere ripetuto per altri LSP
- **Selective snapshot:** per un LSP, consente di scaricare lo storico inclusi cache e campione attuale, senza connettersi alle misure in tempo reale; può essere ripetuto per altri LSP
- **Full last sample:** consente di scaricare l'ultimo campione per tutti gli LSP
- **Selective last sample:** consente di scaricare l'ultimo campione per un LSP; può essere ripetuto per altri LSP
- **Full N most recent samples:** consente di scaricare gli ultimi N campioni per tutti gli LSP, inclusi cache e ultimo campione
- **Selective N most recent samples:** per un LSP, consente di scaricare gli ultimi N campioni, inclusi cache e ultimo campione; può essere ripetuto per altri LSP
- **Full simple connection:** consente di connettersi alle misure in tempo reale (senza avere lo storico) per tutti gli LSP

- **Selective simple connection:** consente di connettersi alle misure in tempo reale (senza avere lo storico) per un LSP; può essere ripetuto per altri LSP

Task di tipo LSP DELETE (OPCODE 2):

- **Full delete:** consente di disconnettersi dalle misure in tempo reale per tutti gli LSP per i quali si è attualmente connessi, ma mantiene la connessione TCP Client Interface-controller
- **Selective delete:** consente di disconnettersi dalle misure per un LSP per cui si è attualmente connessi, ma mantiene la connessione Client Interface-controller. Può essere ripetuto per ogni LSP per cui si sia effettivamente connessi

Task di tipo CHANGE PARAMETERS (OPCODE 3):

- **Change window:** consente di modificare il parametro window
- **Change period:** consente di modificare il parametro period
- **Change window and period:** consente di modificare contemporaneamente entrambi i parametri

Task vari:

- **Info (opcode 4):** consente di ottenere immediatamente informazioni sugli attuali parametri di funzionamento (window, period, lista dei PHB, `HISTORY_LENGTH`) e sugli LSP attivi
- **Clear (opcode 7):** consente di svuotare la lista dei task in coda (in attesa del commit per essere eseguiti)
- **Commit (opcode 5):** dà il via all'esecuzione della coda dei task dei primi due gruppi (DB ACCESS, LSP DELETE)

- **Stop (opcode 6):** blocca l'esecuzione dei task in coda dopo l'arrivo di una richiesta commit; questo avviene durante la chiamata alla funzione `task_executor()`. Le code vengono svuotate. La richiesta stop non è valida se non durante l'esecuzione dei task in coda, ed il suo arrivo in un altro momento sarà segnalato dal controller come un errore
- **Abort (opcode 0):** annulla qualsiasi task attivo o in coda, abbatte la connessione Client Interface-controller.

Definizione del tipo di documento XML (DTD)

Il messaggio XML contenente le informazioni sul task deve essere inviato senza la seguente DTD (alla quale deve comunque fare riferimento), che viene aggiunta dal controller durante il parsing.

```
<!DOCTYPE TASK [
   <!ELEMENT TASK (CLNT_REQUEST?, REQUEST_ID, OPCODE, T_DATA?)>
      <!ELEMENT CLNT_REQUEST (FOR_CTRL, VALUE)>
         <!ELEMENT FOR_CTRL (#PCDATA)>
         <!ELEMENT VALUE (#PCDATA)>
      <!ELEMENT REQUEST_ID (#PCDATA)>
      <!ELEMENT OPCODE (#PCDATA)>
      <!ELEMENT T_DATA (DB_ACCESS | DELETE_LSP | NEW_PARAM)>
         <!ELEMENT DB_ACCESS (LSP, SNAPSHOT, N_SAMP?,
         STAY_CONN)>
            <!ELEMENT LSP (#PCDATA)>
            <!ELEMENT SNAPSHOT (#PCDATA)>
            <!ELEMENT N_SAMP (#PCDATA)>
            <!ELEMENT STAY_CONN (#PCDATA)>
         <!ELEMENT DELETE_LSP (#PCDATA)>
         <!ELEMENT NEW_PARAM (SUBCODE, WINDOW?, SAMPLE_RATE?)>
            <!ELEMENT SUBCODE (#PCDATA)>
            <!ELEMENT WINDOW (#PCDATA)>
            <!ELEMENT SAMPLE_RATE (#PCDATA)>
]>
```

L'elemento radice è TASK, ed il suo primo campo opzionale CLNT_REQUEST non è utilizzato dal controller, bensì dal Client Interface Manager (a tal proposito si veda il paragrafo 5.4.1).
L'elemento TASK contiene poi due campi obbligatori: REQUEST_ID, un identificatore numerico progressivo (non negativo e diverso da zero) di tutte le richieste effettuate al Client Interface Manager, e OPCODE, un numero che codifica l'operazione da svolgere secondo quanto indicato in precedenza. Queste sono informazioni sufficienti per tutti i task del gruppo task vari.
Un esempio di messaggio di tipo info è dunque:

```
<TASK>
        <REQUEST_ID>19</REQUEST_ID>
        <OPCODE>4</OPCODE>
</TASK>
```

Gli altri gruppi di task hanno bisogno di campi aggiuntivi, per i quali c'è l'elemento opzionale T_DATA. Il suo contenuto è uno (ed uno soltanto) tra i tre elementi DB_ACCESS, DELETE_LSP e NEW_PARAM.

Elemento DB_ACCESS:

Contiene le informazioni aggiuntive per i task del gruppo DB ACCESS. Combinando opportunamente i valori dei suoi campi si possono ottenere tutti i messaggi del suddetto gruppo. Contiene gli elementi:

- LSP: identificativo dello LSP per il quale si richiede l'operazione; zero nel caso si voglia un accesso al database di tipo full (ossia tutti gli LSP)
- SNAPSHOT: questo campo deve essere diverso da zero se si vuole un accesso di tipo snapshot, zero altrimenti
- N_SAMP: è un campo opzionale (nel controller è valido soltanto se si ha una snapshot, ossia se il campo SNAPSHOT è diverso da zero), che indica

il numero di campioni desiderati. Questo campo può avere valore compreso tra 1 (si desidera soltanto l'ultimo campione) ed il parametro `HISTORY_LENGTH` (si desiderano tutti i campioni del database). Il valore zero sarà interpretato nello stesso modo di `HISTORY_LENGTH`, ossia come una richiesta per tutti i campioni dello storico

- `STAY_CONN`: questo campo deve essere diverso da zero se si vuole rimanere connessi per le misure in tempo reale, zero altrimenti

Alla luce di queste convenzioni, un messaggio esemplificativo (di tipo *full simple connection*) è il seguente:

```
<TASK>
        <REQUEST_ID>59</REQUEST_ID>
        <OPCODE>1</OPCODE>
        <T_DATA>
                <DB_ACCESS>
                        <LSP>0</LSP>
                        <SNAPSHOT>0</SNAPSHOT>
                        <STAY_CONN>1</STAY_CONN>
                </DB_ACCESS>
        </T_DATA>
</TASK>
```

La tabella 5.1 riassume le combinazioni dei campi per ogni tipo di messaggio DB ACCESS:

DB ACCESS	Full conn	Sel. conn	Full snap	Sel. snap	Full simp	Sel. simp	Full last	Sel. last	Full N	Sel. N
`LSP_ID`	**0**	**ID**	**0**	**ID**	**0**	**ID**	**0**	**ID**	**0**	**ID**
`SNAPSHOT`	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**
`N_SAMPLES`	**0**	**0**	**0**	**0**	**–**	**–**	**1**	**1**	**N**	**N**
`STAY_CONN.`	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**

Tabella 5.1 – Campi per i messaggi DB ACCESS

Nella tabella, il valore ID indica che va messo l'identificatore dello LSP al quale si è interessati; il valore 1 indica un intero diverso da zero, eccetto che per il campo `N_SAMPLES`. In questo campo infatti il valore 1 indica proprio l'unità, poiché compare nei messaggi *last sample* in cui si è interessati solo ad un campione. Il valore N indica il numero dei campioni a cui si è interessati nei messaggi di tipo *N samples*; la sbarretta indica che il campo non è richiesto.

Elemento `DELETE_LSP`:

Serve per specificare i dati di un messaggio di tipo LSP DELETE. Contiene l'identificatore dello LSP per il quale si richiede l'operazione; zero nel caso si voglia disconnettersi da tutti gli LSP per i quali si è connessi. Ad esempio, per disconnettersi dallo LSP 16, il messaggio è il seguente:

```
<TASK>
        <REQUEST_ID>77</REQUEST_ID>
        <OPCODE>2</OPCODE>
        <T_DATA>
                <DELETE_LSP>16</DELETE_LSP>
        </T_DATA>
</TASK>
```

Elemento `NEW_PARAM`:

Vi risiedono le informazioni aggiuntive per i task del gruppo CHANGE PARAMETERS. Contiene gli elementi:

- `SUBCODE`: questo campo obbligatorio indica il tipo di operazione da svolgere: deve valere zero se non si apportano cambiamenti, 1 se si desidera cambiare il parametro window, 2 se si vuole cambiare il parametro period, 4 per cambiare entrambi

- `WINDOW`: (campo opzionale) nuovo valore in secondi per il parametro window; durante il parsing viene convertito in un `double` con la chiamata `strtod()`. Nel controller questo campo ha senso soltanto se `SUBCODE` vale 1 o 4
- `SAMPLE_RATE`: (campo opzionale) nuovo valore in secondi per il parametro period; durante il parsing viene convertito in un `double` con la chiamata `strtod()`. Nel controller questo campo ha senso soltanto se `SUBCODE` vale 2 o 4

Se ad esempio vogliamo cambiare il parametro period, il messaggio è il seguente:

```
<TASK>
     <REQUEST_ID>80</REQUEST_ID>
     <OPCODE>3</OPCODE>
     <T_DATA>
          <NEW_PARAM>
               <SUBCODE>2</SUBCODE>
               <SAMPLE_RATE>nuovo valore di period</SAMPLE_RATE>
          </NEW_PARAM>
     </T_DATA>
</TASK>
```

Nel controller, le informazioni relative ad ogni richiesta di task vengono salvate in un'apposita struttura dati di cui riportiamo la definizione:

```
struct db_access_task_struct
       {
       unsigned int lsp_id;
       int snapshot;
       unsigned int n_samples;
       int stay_connected;
       };

struct lsp_delete_task_struct
       {
```

```
        unsigned int lsp_id;
        };

struct ch_param_task_struct
        {
        int sub_code;
        double new_window;    //new window in seconds
        double new_period;    //new sampling period in seconds
        };

struct ctrl_task_struct
        {
        struct list_head list;
        unsigned int request_id;
        unsigned int ctrl_opcode;
        int error_code;
        union {//Struttura diversa a seconda dell'op_code
                struct db_access_task_struct db_access;
                struct lsp_delete_task_struct lsp_delete;
                struct ch_param_task_struct ch_param;
                } op_struct;
        };
```

5.2.2 Repliche del controller

Il modulo Client Interface riceve sul control socket le repliche da parte del controller alle richieste di esecuzione task, nonché le informazioni sui parametri correnti (all'inizio della connessione come default, poi in risposta ad una richiesta di tipo info). Queste repliche utilizzano XML, e, analogamente a quanto avviene per il protocollo esaminato nel precedente paragrafo, sono precedute da un intero indicante la lunghezza in byte del messaggio.

Definizione del tipo di documento XML (DTD)

Come per il protocollo di richiesta task, il controller invia un semplice messaggio XML che non include la specifica del tipo di documento. La DTD alla quale le repliche fanno riferimento è la seguente:

```
<!DOCTYPE REPLY [
    <!ELEMENT REPLY (REQ_ID, T_CODE, R_FLAG, R_CODE, INFO?)>
        <!ELEMENT REQ_ID (#PCDATA)>
        <!ELEMENT T_CODE (#PCDATA)>
        <!ELEMENT R_FLAG (#PCDATA)>
        <!ELEMENT R_CODE (#PCDATA)>
        <!ELEMENT INFO (CURRENT_WINDOW, CURRENT_PERIOD,
        VECT_DIM, PHB_NUM, LSPLIST)>
            <!ELEMENT CURRENT_WINDOW (#PCDATA)>
            <!ELEMENT CURRENT_PERIOD (#PCDATA)>
            <!ELEMENT VECT_DIM (#PCDATA)>
            <!ELEMENT PHB_NUM (#PCDATA)>
            <!ELEMENT LSPLIST (#PCDATA)>
]>
```

Sono presenti alcuni campi obbligatori, che vengono riempiti dal controller mediante la funzione `send_xml_reply()`:

- `REQ_ID`: contiene il valore del campo `REQUEST_ID` della richiesta di task alla quale la replica fa riferimento. Nel caso che ci sia stato un errore di convalida DTD o di parsing del messaggio XML, e non sia stato quindi possibile estrarre il valore del campo `REQUEST_ID`, il controller setta a zero questo campo
- `T_CODE`: contiene il valore del campo `OPCODE` della richiesta di task alla quale la replica fa riferimento
- `R_FLAG`: nel caso sia diverso da zero, indica che la replica fa riferimento ad un'inserzione in coda di un task in attesa del commit, altrimenti indica che la replica si riferisce all'esito vero e proprio della richiesta

- `R_CODE`: contiene un numero che codifica l'esito della richiesta; zero indica successo, mentre un numero diverso da zero indica che c'è stato un errore. Gli errori che possono essere notificati sono elencati in tabella 5.2.

All'arrivo di una richiesta di tipo commit, il controller attende l'esecuzione dei task in coda da parte della funzione `task_executor()`, quindi controlla l'esito della suddetta esecuzione. Se tutte le richieste sono state eseguite correttamente, viene inviata una sola notifica, con i campi `REQUEST_ID` e `OPCODE` del commit in `REQ_ID` e `T_CODE`, e con valore zero in `R_FLAG` e `R_CODE`. Se c'è stato qualche errore, vengono inviati messaggi relativi ad ogni task per il quale si è verificato errore, compresi tra due repliche con il `REQUEST_ID` del commit, entrambe con `R_CODE` pari a 100. La prima, che delimita l'inizio del gruppo di notifiche di errori, ha `R_FLAG` posto ad 1; la seconda, a conclusione del gruppo di notifiche, ha `R_FLAG` posto a 0.

Valore	Descrizione
0	Nessun errore, successo dell'operazione per cui si effettua la notifica
-1	In un msg. LSP DELETE lo LSP richiesto non esiste più
-2	In un msg. LSP DELETE il client non era connesso per quello LSP
-3	Richiesta LSP DELETE per un singolo LSP con full connection attiva
1	In un msg. DB ACCESS lo LSP richiesto non esiste più
2	CHANGE PARAMETERS non eseguibile perché una simile è già in corso
3	Richiesta DB ACCESS di connessione a un LSP per cui si è già connessi
4	Richiesta DB ACC. di connessione a un singolo LSP con full conn. attiva
5	Richiesta DB ACCESS di full connection con full connection già attiva
11	Errore nel parsing del msg. XML
12	Errore nella convalida DTD del msg. XML
13	Msg. scartato, diverso da stop o abort durante l'esecuzione task in coda
14	Msg. stop scartato, arrivato al di fuori dell'esecuzione task in coda
15	Msg. commit scartato, arrivato con le code dei task vuote
16	Valore non legale del campo `OPCODE`
17	In un msg. CHANGE PARAM., valore non legale del campo `SUBCODE`
18	Msg. clear scartato, arrivato con le code dei task vuote

19	Msg. DB ACCESS inutile, con i due campi `SNAPSHOT` e `STAY_CONN`. a 0
20	Msg. scartato, il campo `N_SAMP` è maggiore di `HISTORY_LENGTH`
21	Errori nei task della coda eseguiti dopo il commit
50	Esecuzione dei task bloccata con successo su richiesta a mezzo msg. stop
90	Errore di cache full nello svolgimento della funzione `task_executor`

Tabella 5.2 – Codici di risposta del controller

Esempio di messaggio di replica (indica inserzione in coda di un task di tipo DB ACCESS):

```
<REPLY>
        <REQ_ID>7</REQ_ID>
        <T_CODE>1</T_CODE>
        <R_FLAG>1</R_FLAG>
        <R_CODE>0</R_CODE>
</REPLY>
```

In risposta ad una richiesta di task di tipo info, nella replica sono presenti, oltre alla notifica del successo, le stesse informazioni richieste, per le quali c'è l'elemento opzionale INFO.

Elemento `INFO`:

Il controller riempie i campi di questo elemento con le informazioni aggiornate sui parametri di funzionamento e sulla lista degli LSP attivi:

- `CURRENT_WINDOW`: attuale valore del parametro window, variabile di tipo `double` stampata come stringa con la conversione `sprintf("%g", …)`
- `CURRENT_PERIOD`: attuale valore del parametro period, variabile di tipo `double` stampata come stringa con la conversione `sprintf("%g", …)`
- `VECT_DIM`: attuale valore di `HISTORY_LENGTH`, dimensione temporale del database (e dunque massimo numero di campioni che si può avere in uno storico)

- `PHB_NUM`: stringa contenente l'attuale elenco dei Per Hop Behaviour in uso, assieme al valore numerico con il quale sono codificati nel Metercontroller (ad esempio: "0 BE, 1 EF, 2 AF11 …")
- `LSPLIST`: stringa contenente gli identificativi degli LSP attivi, separati da uno spazio (ad esempio: "300 512 2000 2010")

5.3 Protocolli di trasferimento dati binari

In questo paragrafo vengono descritti i protocolli in uso sui socket real-time e snapshot, che riguardano la comunicazione dei dati binari di traffico dal Metercontroller al modulo Client Interface. Si noti che questi protocolli costituiscono un lavoro abbastanza definitivo, e che la loro conoscenza non è necessaria per il progetto della User Interface.

5.3.1 Dati real-time

Sul socket real-time viaggiano messaggi nella sola direzione dal sampler al modulo Client Interface. Tali messaggi, che non fanno uso di XML, contengono i timestamps ed i campioni delle misure sugli LSP, e sono delimitati da marcatori di sei caratteri che ne consentono il parsing.
Ogni messaggio comincia con il marcatore `_START` e termina con `_END__`. Dopo il marcatore di partenza i marcatori possibili sono tre:

- `_ERASE`, che si ha in corrispondenza di un cambiamento di window e/o sample rate nel sistema di misura (e quindi il messaggio contiene i nuovi valori di tali parametri, e segnala la necessità di provvedere ad aggiornare il database locale del Client Interface)

- `TARRAY` o `TSTAMP`, che contengono i timestamps ed i campioni e si differenziano tra loro per avere un singolo timestamp o un array di timestamps

Messaggio _ERASE:

`_START`
`_ERASE`
`<2*sizeof(double) bytes>`, nuovi valori di window e period
`_END__`

Messaggio TARRAY:

`_START`
`TARRAY`
`<2*HISTORY_LENGTH*sizeof(long int) bytes>`, array di `HISTORY_LENGTH` timestamps
CAMPIONI, vedi sotto
`_END__`

Messaggio TSTAMP:

`_START`
`TSTAMP`
`<2*sizeof(long int) bytes>`, un timestamp
CAMPIONI, vedi sotto
`_END__`

Formato dei campioni

Ci sono due possibili marcatori che indicano la presenza di campioni: `_LSP__` e `_CLSP_`. Il primo è per i dati in tempo reale veri e propri, l'altro è per i campioni in cache. Dopo tali marcatori seguono campioni di un solo LSP (e quindi ci sarà un marcatore per ogni LSP di cui si trasmettono i dati), ed un altro marcatore ci dice se tale LSP è nuovo (`_NEW__`) o era già

presente (_ACTV_), o se è stato rimosso (_DELE_) secondo il seguente schema (NB: un array di campioni di un LSP è costituito dai campioni per ciascuno degli otto PHB):

CLSP
<sizeof(unsigned long) bytes>, id del corrente LSP
_NEW__ *oppure* _ACTV_
<sizeof(unsigned long) bytes>, numero di array di campioni in cache per il corrente LSP
<8*sizeof(unsigned long)*numero di array di campioni in cache bytes>, campioni veri e propri per ogni PHB

_LSP__
<sizeof(unsigned long) bytes>, id del corrente LSP
_NEW__ *oppure* _ACTV_ *oppure* _DELE_
<8*sizeof(unsigned long)>, campioni veri e propri per ogni PHB, assente nel caso il precedente marcatore sia _DELE_

5.3.2 Dati snapshot

Sul socket snapshot viaggiano messaggi nella sola direzione dal sampler al modulo Client Interface. Tali messaggi contengono i dati binari di snapshot per gli LSP richiesti, e, come per i dati real-time, sono delimitati da marcatori di sei caratteri che ne consentono il parsing. Ogni messaggio comincia con il marcatore _START e termina con _END__.
I marcatori possibili sono i seguenti, presentati assieme ai dati che li seguono; un array di campioni di un LSP è costituito dai campioni per ciascuno degli otto PHB (classi di servizio):

START
DELALL (è stata richiesta la disconnessione da tutti gli LSP)
_LSP__

`<sizeof(unsigned long) bytes>` id del corrente LSP
`_SNAP_` (invio snapshot per il corrente LSP)
`<sizeof(unsigned long) bytes>` numero di array di campioni per il corrente LSP
`<numero array campioni*numero di classi di servizio(8)* sizeof(unsigned long) bytes>` campioni veri e propri
`NOSNAP` (non è stata richiesta alcuna snapshot per questo LSP)
`_DELE_` (è stato richiesto di scollegarsi da questo LSP)
`_END__`

La figura 5.3 descrive l'albero dei possibili messaggi di snapshot.

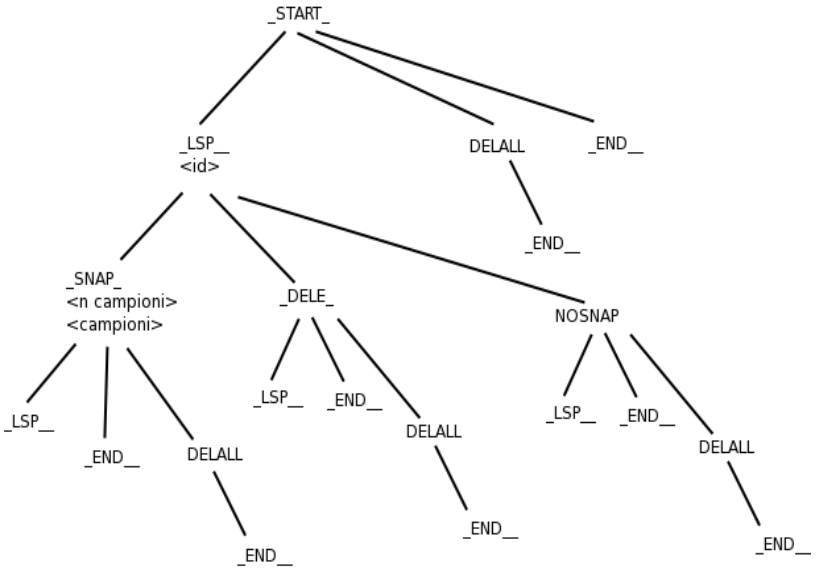

Figura 5.3 – Albero dei messaggi snapshot

I marcatori `_SNAP_`, `_DELE_` e `NOSNAP` seguono sempre un marcatore `_LSP__` ed il suo id; Il marcatore `DELALL` è sempre seguito da un marcatore `_END__`.

5.4 Protocolli di accesso al sistema di misura

In questo paragrafo si esaminano i protocolli di accesso al Client Interface Manager relativi al master socket. Nel paragrafo 5.4.1 si esamina il formato delle richieste di operazioni inviate dalla UI al CIM, nel 5.4.2 quello delle repliche e notifiche inviate dal CIM alla UI. La conoscenza di questi protocolli è fondamentale per la progettazione della User Interface.

5.4.1 Richiesta operazioni al CIM

Il Client Interface Manager interroga periodicamente il master socket per sapere se ci sono richieste di operazioni da svolgere. Nel caso ve ne siano, le riceve e ne effettua il parsing. Tali richieste possono riguardare il CIM stesso (creazione di un modulo Client Interface, cancellazione di un modulo Client Interface, disconnessione completa e chiusura) oppure un controller pilotato da un modulo Client Interface già attivo (esecuzione task, come spiegato nel paragrafo 5.2.1). Nel secondo caso il CIM passa la richiesta al corretto modulo CI che provvederà ad inviarla al controller di sua competenza. Entrambe le tipologie di operazione sono comunque segnalate tramite un unico modello di documento XML.
La ricezione sul master socket avviene in questo modo: se il Client Interface Manager trova il descrittore pronto in lettura, legge prima `sizeof(int)` byte, che devono indicare la lunghezza del messaggio XML seguente, poi riceve il messaggio stesso sfruttando l'informazione della lunghezza, ottenuta in precedenza.

Definizione del tipo di documento XML (DTD)

Il messaggio XML contenente le informazioni sulla richiesta deve essere inviato senza la seguente DTD (alla quale deve comunque fare riferimento), che viene aggiunta dal Client Interface Manager durante il parsing.

```
<!DOCTYPE TASK [
     <!ELEMENT TASK (CLNT_REQUEST, REQUEST_ID, OPCODE?,
     T_DATA?)>
          <!ELEMENT CLNT_REQUEST (FOR_CTRL, VALUE)>
               <!ELEMENT FOR_CTRL (#PCDATA)>
               <!ELEMENT VALUE (#PCDATA)>
          <!ELEMENT REQUEST_ID (#PCDATA)>
          <!ELEMENT OPCODE (#PCDATA)>
          <!ELEMENT T_DATA (DB_ACCESS | DELETE_LSP | NEW_PARAM)>
               <!ELEMENT DB_ACCESS (LSP, SNAPSHOT, N_SAMP?,
               STAY_CONN)>
                    <!ELEMENT LSP (#PCDATA)>
                    <!ELEMENT SNAPSHOT (#PCDATA)>
                    <!ELEMENT N_SAMP (#PCDATA)>
                    <!ELEMENT STAY_CONN (#PCDATA)>
               <!ELEMENT DELETE_LSP (#PCDATA)>
               <!ELEMENT NEW_PARAM (SUBCODE, WINDOW?,
               SAMPLE_RATE?)>
                    <!ELEMENT SUBCODE (#PCDATA)>
                    <!ELEMENT WINDOW (#PCDATA)>
                    <!ELEMENT SAMPLE_RATE (#PCDATA)>
]>
```

Come si può vedere, la DTD è quasi identica a quella definita per il protocollo di richiesta task al controller; la differenza è nel fatto che qui il campo `CLNT_REQUEST` è obbligatorio ed il campo `OPCODE` è opzionale. Il motivo della analogia è il fatto che il messaggio XML inviato dal modulo Client Interface al controller (secondo il protocollo visto nel paragrafo 5.2.1) è in realtà lo stesso messaggio XML ricevuto dal Client Interface Manager (secondo questo protocollo), che viene inoltrato intatto. Il motivo

della differenza è il fatto che il CIM può ricevere anche messaggi non destinati ad un controller, nei quali sono presenti soltanto gli elementi `CLNT_REQUEST` e `REQUEST_ID`. Quest'ultimo è il solito identificatore numerico progressivo della richiesta.

Elemento CLNT_REQUEST:

Questo elemento contiene i dati riguardanti l'operazione che deve svolgere il Client Interface Manager. Le operazioni possibili sono quattro: creare un'interfaccia CI, cancellare un'interfaccia CI, inoltrare una richiesta di task ad un'interfaccia CI, uscire.

- `FOR_CTRL`: questo campo contiene l'indirizzo del controller al quale si riferisce l'operazione corrente; nel caso si richieda al CIM di uscire, il valore di questo campo è ininfluente e non viene preso in considerazione. Il formato dell'indirizzo è una stringa con lo standard *dotted quad* (*ddd.ddd.ddd.ddd*) per IPv4.
- `VALUE`: contiene un numero, il codice dell'operazione che il Client Interface Manager deve eseguire:
  - value 0: il CIM chiude tutte le interfacce attive ed esce; il campo `FOR_CTRL` non è utilizzato
  - value 1: il CIM crea, se ha slot liberi, un modulo Client Interface che tenta di connettersi al controller identificato dall'indirizzo presente in `FOR_CTRL` (nota: non si possono creare due interfacce CI verso lo stesso controller)
  - value 2: il CIM cancella, se presente, il modulo Client Interface connesso al controller indicato dall'indirizzo presente in `FOR_CTRL`
  - value 3: il messaggio è una richiesta di task verso il controller indicato dall'indirizzo presente in `FOR_CTRL`; il Client Interface

Manager passa il messaggio al modulo Client Interface giusto (se ne esiste uno connesso a tale controller), che provvederà ad inoltrarlo al controller

I possibili messaggi XML per il Client Interface Manager (messaggi di controllo) sono dunque esemplificabili come segue:

```
<TASK>
        <CLNT_REQUEST>
                <FOR_CTRL>191.2.200.3</FOR_CTRL>
                <VALUE>1</VALUE>
        </CLNT_REQUEST>
</TASK>
```

(crea un modulo Client Interface verso il controller di indirizzo 191.2.200.3)

```
<TASK>
        <CLNT_REQUEST>
                <FOR_CTRL>191.2.200.3</FOR_CTRL>
                <VALUE>2</VALUE>
        </CLNT_REQUEST>
</TASK>
```

(cancella il modulo Client Interface attivo verso il controller di indirizzo 191.2.200.3)

```
<TASK>
        <CLNT_REQUEST>
                <FOR_CTRL>irrilevante</FOR_CTRL>
                <VALUE>0</VALUE>
        </CLNT_REQUEST>
</TASK>
```

(cancella qualsiasi modulo Client Interface attivo ed esce)

Nei messaggi di richiesta task devono essere presenti anche i campi `REQUEST_ID`, `OPCODE` ed eventualmente `T_DATA` secondo quanto definito nel protocollo descritto nel paragrafo 5.2.1, altrimenti i messaggi non saranno ritenuti validi quando perverranno al controller. Un esempio di richiesta di inoltro task è il seguente:

```
<TASK>
        <CLNT_REQUEST>
                <FOR_CTRL>191.2.200.3</FOR_CTRL>
                <VALUE>3</VALUE>
        </CLNT_REQUEST>
        <REQUEST_ID>30</REQUEST_ID>
        <OPCODE>1</OPCODE>
        <T_DATA>
                <DB_ACCESS>
                        <LSP>0</LSP>
                        <SNAPSHOT>1</SNAPSHOT>
                        <N_SAMP>HISTORY_LENGTH</N_SAMP>
                        <STAY_CONN>1</STAY_CONN>
                </DB_ACCESS>
        </T_DATA>
</TASK>
```

(si richiede al Client Interface Manager di inoltrare una richiesta di task di tipo *full connection* al controller di indirizzo 191.2.200.3; nota: deve esserci già un modulo Client Interface attivo verso quel controller)

5.4.2 Repliche del CIM

Il Client Interface Manager invia sul master socket sia le repliche da parte dei moduli Client Interface (che giungono dai controller ai quali i moduli sono connessi) sia le repliche ai messaggi di gestione come ad esempio creare o eliminare un'interfaccia CI. L'invio delle notifiche è effettuato in

modo non bloccante, ossia il CGI connesso al Client Interface Manager può perdere alcune repliche se non è abbastanza rapido nella ricezione (cosa comunque assai poco probabile dato che CGI e Client Interface Manager sono in realtà sulla stessa macchina). Il CIM invia un intero – `sizeof(int)` bytes – indicante la lunghezza del messaggio, quindi il messaggio vero e proprio. Come per gli altri protocolli di controllo, anche qui si utilizza XML.

Definizione del tipo di documento XML (DTD)

Il Client Interface Manager invia un semplice messaggio XML che non include la specifica del tipo di documento. La DTD alla quale le notifiche fanno riferimento è la seguente:

```
<!DOCTYPE NOTIFY [
    <!ELEMENT NOTIFY (TYPE, REPLY)>
        <!ELEMENT TYPE (FROM_CTRL | CMAN_CODE)>
            <!ELEMENT FROM_CTRL (#PCDATA)>
            <!ELEMENT CMAN_CODE (#PCDATA)>
        <!ELEMENT REPLY (REQ_ID, T_CODE?, R_FLAG?, R_CODE?,
        INFO?)>
            <!ELEMENT REQ_ID (#PCDATA)>
            <!ELEMENT T_CODE (#PCDATA)>
            <!ELEMENT R_FLAG (#PCDATA)>
            <!ELEMENT R_CODE (#PCDATA)>
            <!ELEMENT INFO (CURRENT_WINDOW, CURRENT_PERIOD,
            VECT_DIM, PHB_NUM, LSPLIST)>
                <!ELEMENT CURRENT_WINDOW (#PCDATA)>
                <!ELEMENT CURRENT_PERIOD (#PCDATA)>
                <!ELEMENT VECT_DIM (#PCDATA)>
                <!ELEMENT PHB_NUM (#PCDATA)>
                <!ELEMENT LSPLIST (#PCDATA)>
]>
```

Si configurano due situazioni distinte a seconda che il messaggio di notifica sia inviato dal Client Interface Manager a seguito di una richiesta di gestione oppure provenga da un controller come replica ad una richiesta di esecuzione task. Nel primo caso è ovviamente il Client Interface Manager a generare ed inviare il messaggio XML; nel secondo, il modulo Client Interface riceve un messaggio di replica da parte di un controller secondo il protocollo visto nel paragrafo 5.2.2, provvede ad incapsularlo tra i tag `<NOTIFY>` aggiungendo anche il campo `TYPE`, infine passa il nuovo messaggio così creato al Client Interface Manager che lo inoltra al CGI. L'elemento `TYPE` serve proprio per discriminare i due casi. L'elemento `REPLY` contiene invece le informazioni aggiuntive sulla replica.

Elemento TYPE:

L'elemento `TYPE` contiene uno ed uno solo tra gli elementi `FROM_CTRL` e `CMAN_CODE`.

- `FROM_CTRL`: è presente nel caso di una replica proveniente da un controller connesso ad un modulo Client Interface, e contiene l'indirizzo del controller medesimo, in formato *dotted quad* (*ddd.ddd.ddd.ddd*) per IPv4.
- `CMAN_CODE`: è presente nel caso di una replica del Client Interface Manager, e contiene un valore numerico che codifica l'esito dell'operazione richiesta. I possibili valori assunti da questo campo sono elencati, assieme al loro significato, nella sottostante tabella 5.3.

Valore	Descrizione
	Codici di errore generici
1	Errore nel parsing del msg. XML
2	Errore nella convalida DTD del msg. XML
3	Errore nella ricezione del msg. XML nella funzione `recv()`

4	Errore dovuto a timeout nella ricezione del msg. XML
5	Valore non legale del campo `VALUE` (codice operativo per il CIM)
6	Valore non valido del campo `FOR_CTRL` (indirizzo del controller)
	Codici di replica alla richiesta di creazione CI
11	***Successo*** nella creazione di un modulo CI
12	Indirizzo IP del controller non valido nella richiesta di creazione CI
13	Richiesta scartata, massimo numero di CI attivi già raggiunto
14	Richiesta scartata, esiste già un CI attivo verso il controller indicato
	Codici di replica alla richiesta di cancellazione CI
18	***Successo*** nella cancellazione di un modulo CI
19	Indirizzo IP del controller non valido nella richiesta di cancellazione CI
20	Richiesta scartata, non esiste alcun CI attivo verso il controller indicato
	Codici di replica alla richiesta di inoltro task ad un controller
24	***Successo*** nell'inoltro di un messaggio al corretto modulo CI
25	Indirizzo IP del controller non valido nella richiesta di inoltro messaggio
26	Richiesta scartata, non esiste alcun CI attivo verso il controller indicato
27	Richiesta di inoltro scartata, il CI non ne ha ancora finita una precedente
	Codici di errore di socket del CIM
50	Procedura di recupero ultimata sul master socket
51	Procedura di recupero ultimata sullo estimate socket
	Codici di errore di socket di un modulo CI
7	Errore fatale di socket, il modulo CI è uscito automaticamente
52	Errore sul real-time socket, sincronizzazione real-time parser iniziata
53	Errore sullo snapshot socket, sincronizzazione snapshot parser iniziata

Tabella 5.3 – Codici di replica del CIM sul master socket

Elemento `REPLY`:

Si può notare che l'elemento `REPLY` è esattamente quello definito nel protocollo visto nel paragrafo 5.2.2, ossia è l'elemento che contiene le informazioni sulle repliche di un controller. Nel caso di una replica proveniente da un controller, vi si trovano le informazioni sull'esito dello svolgimento di un task ed eventualmente i campi per la risposta al messaggio *info*, secondo quanto stabilito in precedenza. Né il modulo Client Interface né il Client Interface Manager modificano questa parte.

L'elemento `REPLY` è usato parzialmente anche nell'altro caso (replica del Client Interface Manager). Difatti, quando il Client Interface Manager invia un messaggio di replica ad un'operazione a lui richiesta, sfrutta l'elemento `REPLY` per trasmettere informazioni aggiuntive. Per la precisione, sfrutta il campo `REQ_ID` per allegare l'identificatore della richiesta alla quale sta replicando (oppure 0 quando l'identificatore non è disponibile, come nel caso di errore di parsing o DTD), e riempie (quando possibile) il campo `R_CODE` con l'indirizzo IP del controller verso il quale era stata effettuata la richiesta. Anche nelle notifiche di errore di socket di un modulo CI viene indicato mediante questo campo qual è il modulo interessato, specificando l'indirizzo del controller verso il quale era connesso.

Esempi di messaggi:

```
<NOTIFY>
	<TYPE>
		<FROM_CTRL>192.1.2.3</FROM_CTRL>
	</TYPE>
	<REPLY>
		<REQ_ID>12</REQ_ID>
		<T_CODE>1</T_CODE>
		<R_FLAG>1</R_FLAG>
		<R_CODE>0</R_CODE>
	</REPLY>
</NOTIFY>
```

(il Client Interface Manager ha inoltrato al CGI la risposta del controller di indirizzo 192.1.2.3 alla richiesta task di id #12; la richiesta era di tipo DB ACCESS, è stata messa in coda correttamente)

```
<NOTIFY>
	<TYPE>
		<FROM_CTRL>192.1.2.3</FROM_CTRL>
```

```
        </TYPE>
        <REPLY>
                <REQ_ID>15</REQ_ID>
                <T_CODE>7</T_CODE>
                <R_FLAG>0</R_FLAG>
                <R_CODE>0</R_CODE>
        </REPLY>
</NOTIFY>
```

(il Client Interface Manager ha inoltrato al CGI la risposta del controller di indirizzo 192.1.2.3 alla richiesta task di id #15; la richiesta era di tipo clear, è stata eseguita correttamente)

```
<NOTIFY>
        <TYPE>
                <CMAN_CODE>2</CMAN_CODE>
        </TYPE>
        <REPLY>
                <REQ_ID>0</REQ_ID>
        </REPLY>
</NOTIFY>
```

(risposta del Client Interface Manager; la richiesta non rispettava la specifica della DTD)

```
<NOTIFY>
        <TYPE>
                <CMAN_CODE>14</CMAN_CODE>
        </TYPE>
        <REPLY>
                <REQ_ID>5</REQ_ID>
                <R_CODE>161.80.77.1</R_CODE>
        </REPLY>
</NOTIFY>
```

(risposta del Client Interface Manager alla richiesta di id #5; non è stato possibile creare un modulo Client Interface verso il controller di indirizzo 161.80.77.1 perché c'era già un modulo attivo verso quel controller)

```
<NOTIFY>
        <TYPE>
                <CMAN_CODE>18</CMAN_CODE>
        </TYPE>
        <REPLY>
                <REQ_ID>34</REQ_ID>
                <R_CODE>165.29.17.1</R_CODE>
        </REPLY>
</NOTIFY>
```

(risposta del Client Interface Manager alla richiesta di id #34; è stato cancellato correttamente il modulo Client Interface verso il controller di indirizzo 165.29.17.1)

```
<NOTIFY>
        <TYPE>
                <CMAN_CODE>25</CMAN_CODE>
        </TYPE>
        <REPLY>
                <REQ_ID>76</REQ_ID>
                <R_CODE>212.7.8.9</R_CODE>
        </REPLY>
</NOTIFY>
```

(risposta del Client Interface Manager alla richiesta di id #76; non è stato possibile inoltrare il task verso il controller di indirizzo 212.7.8.9 poiché non vi era un modulo Client Interface attivo verso quel controller)

```
<NOTIFY>
        <TYPE>
                <CMAN_CODE>7</CMAN_CODE>
        </TYPE>
        <REPLY>
                <REQ_ID>0</REQ_ID>
                <R_CODE>123.4.5.6</R_CODE>
        </REPLY>
</NOTIFY>
```

(notifica del Client Interface Manager; il modulo Client Interface attivo verso il controller di indirizzo 123.4.5.6 è uscito per errori sui socket di controllo o di dati)

5.5 Protocolli di accesso alle operazioni di stima

La seconda coppia di protocolli di accesso al Client Interface Manager è quella riguardante lo estimate socket. Nel paragrafo 5.5.1 è descritto il protocollo per le richieste al CIM, tramite il quale si possono impostare le soglie per le misure sugli LSP e far eseguire le operazioni nell'ambito delle stime statistiche. Nel paragrafo 5.5.2 si parla invece della messaggistica utilizzata dal CIM per le repliche e le notifiche (sempre sullo estimate socket).

5.5.1 Richiesta operazioni di stima al CIM

Il Client Interface Manager interroga periodicamente lo estimate socket (essendo l'unico thread a ricevere dati su tale socket) per sapere se ci sono richieste di operazioni di stima da svolgere. Nel caso ve ne siano, le riceve e ne effettua il parsing. Se le richieste sono destinate ad un modulo Client Interface, provvede poi a passarle, a patto che esista il CI desiderato. La ricezione avviene in questo modo: se il Client Interface Manager trova il descrittore pronto in lettura, legge prima `sizeof(int)` byte, che devono indicare la lunghezza del messaggio XML seguente, poi riceve il messaggio stesso sfruttando l'informazione della lunghezza.

Richieste possibili nell'ambito delle operazioni di stima:

- **Estimators list (est_opcode 1):** si richiede al Client Interface Manager di inviare la lista degli stimatori supportati
- **Add estimator (est_opcode 2):** si richiede ad un modulo Client Interface attivo di agganciare un certo stimatore ad un determinato LSP
- **Adjust estimator (est_opcode 3):** si richiede ad un modulo Client Interface attivo di modificare i parametri di uno stimatore agganciato ad un determinato LSP
- **Remove estimator (est_opcode 4):** si richiede ad un modulo Client Interface attivo di rimuovere uno stimatore agganciato ad un determinato LSP
- **Set threshold (est_opcode 5):** si richiede ad un modulo Client Interface di configurare le soglie per un certo LSP
- **LSP e/t info (est_opcode 6):** si richiede ad un modulo Client Interface di fornire per un certo LSP le soglie e la lista degli stimatori su di esso configurati.

I task possibili nell'ambito della stima non sono ancora tutti operativi (le funzioni per la loro esecuzione non rientrano in questo progetto), ma il supporto per la loro attivazione (protocolli, funzioni di parsing e strutture per il salvataggio delle informazioni sui task) è pronto.

Definizione del tipo di documento XML (DTD)

Il messaggio XML contenente le informazioni sulla richiesta deve essere inviato senza la seguente DTD (alla quale deve comunque fare riferimento), che viene aggiunta dal Client Interface Manager o dai moduli Client Interface durante il parsing.

```
<!DOCTYPE EST_REQUEST [
   <!ELEMENT EST_REQUEST (EST_REQUEST_ID, EST_OPCODE,
   REQ_DATA?)>
      <!ELEMENT EST_REQUEST_ID (#PCDATA)>
      <!ELEMENT EST_OPCODE (#PCDATA)>
      <!ELEMENT REQ_DATA (FOR_CI, FOR_LSP, FOR_PHB, EST_DATA?,
      L_UP_TH?, L_LO_TH?)>
         <!ELEMENT FOR_CI (#PCDATA)>
         <!ELEMENT FOR_LSP (#PCDATA)>
         <!ELEMENT FOR_PHB (#PCDATA)>
         <!ELEMENT EST_DATA (EST_NAME, PROTOTYPE?, E_UP_TH?,
         E_LO_TH?)>
            <!ELEMENT EST_NAME (#PCDATA)>
            <!ELEMENT PROTOTYPE (PARAMETER*)>
               <!ELEMENT PARAMETER (NAME?, TYPE?, DESCRIPTION)>
                  <!ELEMENT NAME (#PCDATA)>
                  <!ELEMENT TYPE (#PCDATA)>
                  <!ELEMENT DESCRIPTION (#PCDATA)>
            <!ELEMENT E_UP_TH (#PCDATA)>
            <!ELEMENT E_LO_TH (#PCDATA)>
         <!ELEMENT L_UP_TH (#PCDATA)>
         <!ELEMENT L_LO_TH (#PCDATA)>
]>
```

L'elemento obbligatorio `EST_REQUEST_ID` è il solito identificativo numerico della richiesta; l'elemento obbligatorio `EST_OPCODE` contiene il codice numerico dell'operazione da svolgere, secondo quanto definito in precedenza in questo paragrafo.

Elemento `REQ_DATA`:

L'elemento facoltativo `REQ_DATA` contiene le informazioni aggiuntive sull'operazione da svolgere, eccetto il caso del messaggio *estimators list*, dove non compare in quanto non necessario.
Contiene gli elementi:

- `FOR_CI` (obbligatorio): indirizzo IPv4 del controller che identifica il CI di interesse, in formato *dotted quad*
- `FOR_LSP` (obbligatorio): identificativo dello LSP di interesse
- `FOR_PHB` (obbligatorio): il valore di questo elemento indica a quali PHB si applica il corrente messaggio. Si tratta di un numero intero, detto *PHB mask*, compreso tra 1 e 255, che in fase di ricezione viene interpretato dal CI come un `unsigned long`. Da esso viene ricavata una maschera di otto bit, corrispondenti agli otto Per Hop Behaviour supportati, nella quale uno o più bit posti ad 1 indicano che il messaggio è da applicarsi ai corrispettivi PHB
- `EST_DATA` (facoltativo): vedi sotto
- `L_UP_TH` (facoltativo): soglia superiore per le misure sullo LSP (compare solo nel messaggio *set threshold*); un valore negativo in questo campo disattiva la soglia
- `L_LO_TH` (facoltativo): soglia inferiore per le misure sullo LSP (compare solo nel messaggio *set threshold*); un valore negativo in questo campo disattiva la soglia

Elemento EST_DATA:

L'elemento facoltativo `EST_DATA` contiene le informazioni aggiuntive sullo stimatore che si vuole agganciare, configurare o rimuovere. Non compare nei messaggi *LSP e/t info* e *set threshold*, poiché non serve. Contiene gli elementi:

- `EST_NAME` (obbligatorio): nel caso di messaggio *add estimator*, contiene il nome dello stimatore (il Client Interface Manager si crea all'avvio una lista degli stimatori disponibili che può essere richiesta con il messaggio *estimators list* ed include appunto un campo per il nome).

Nel caso di messaggio *adjust estimator* o *remove estimator*, contiene invece l'identificativo che è fornito dal modulo Client Interface al CGI al momento dell'agganciamento di uno stimatore ad uno LSP e che lo individua univocamente (a questo proposito si veda il paragrafo 5.5.2).

- `PROTOTYPE` (facoltativo): vedi sotto
- `E_UP_TH` (facoltativo): soglia superiore per lo stimatore; non compare nel messaggio *remove estimator*
- `E_LO_TH` (facoltativo): soglia inferiore per lo stimatore; non compare nel messaggio *remove estimator*

Elemento PROTOTYPE:

L'elemento facoltativo `PROTOTYPE` contiene le informazioni aggiuntive sui parametri richiesti per avviare l'esecuzione di uno stimatore. Non compare nel messaggio *remove estimator*. Contiene un numero arbitrario (variabile a §seconda del tipo di stimatore) di elementi `PARAMETER`, ognuno dei quali contiene a sua volta i tre seguenti campi:

- `NAME`: nome del parametro in questione, facoltativo (per ora non utilizzato dal CIM in fase di ricezione ma soltanto in fase di invio, vedi paragrafo 5.5.2)
- `TYPE`: tipo di parametro, facoltativo (*idem* come per `NAME`)
- `DESCRIPTION`: valore del parametro, obbligatorio: con questo campo si specifica il valore da dare ad un parametro, nelle richieste add e adjust estimator. Il CIM utilizza questo campo in fase ricezione, ed in fase di invio lo utilizza con un doppio significato, sia come descrizione leggibile del parametro (in caso di replica a messaggio estimators list) che come valore configurato per il parametro (in caso di replica a messaggio *LSP e/t info*, vedi paragrafo 5.5.2)

Esempi di messaggi:

Segue un esempio di messaggio XML per ogni tipo di richiesta nell'ambito della stima:

```
<EST_REQUEST>
        <EST_REQUEST_ID>28</EST_REQUEST_ID>
        <EST_OPCODE>1</EST_OPCODE>
</EST_REQUEST>
```

(messaggio *estimators list*)

```
<EST_REQUEST>
     <EST_REQUEST_ID>36</EST_REQUEST_ID>
     <EST_OPCODE>2</EST_OPCODE>
     <REQ_DATA>
          <FOR_CI>191.26.37.1</FOR_CI>
          <FOR_LSP>18</FOR_LSP>
          <FOR_PHB>255</FOR_PHB>
          <EST_DATA>
               <EST_NAME>Nome del tipo di stimatore</EST_NAME>
               <PROTOTYPE>
                    <PARAMETER>
                         <DESCRIPTION>Valore per il primo
                         parametro</DESCRIPTION>
                    </PARAMETER>
                    <PARAMETER>
                         <DESCRIPTION>Valore per il secondo
                         parametro</DESCRIPTION>
                    </PARAMETER>
                    …
               </PROTOTYPE>
               <E_UP_TH>Soglia superiore per lo
               stimatore</E_UP_TH>
               <E_LO_TH>Soglia inferiore per lo
               stimatore</E_LO_TH>
          </EST_DATA>
     </REQ_DATA>
</EST_REQUEST>
```

(messaggio *add estimator*, si richiede di agganciare lo stimatore descritto allo LSP di id 18 nel modulo Client Interface attivo verso il controller di indirizzo 191.26.37.1)

```
<EST_REQUEST>
     <EST_REQUEST_ID>40</EST_REQUEST_ID>
     <EST_OPCODE>3</EST_OPCODE>
     <REQ_DATA>
          <FOR_CI>191.26.37.1</FOR_CI>
          <FOR_LSP>18</FOR_LSP>
          <FOR_PHB>255</FOR_PHB>
          <EST_DATA>
               <EST_NAME>Identificatore univoco dello stimatore
               agganciato</EST_NAME>
               <PROTOTYPE>
                    <PARAMETER>
                         <DESCRIPTION>Nuovo valore per il primo
                         parametro</DESCRIPTION>
                    </PARAMETER>
                    <PARAMETER>
                         <DESCRIPTION>Nuovo valore per il secondo
                         parametro</DESCRIPTION>
                    </PARAMETER>
                    …
               </PROTOTYPE>
               <E_UP_TH>Nuova soglia superiore per lo
               stimatore</E_UP_TH>
               <E_LO_TH>Nuova soglia inferiore per lo
               stimatore</E_LO_TH>
          </EST_DATA>
     </REQ_DATA>
</EST_REQUEST>
```

(messaggio *adjust estimator*, si richiede di modificare i parametri dello stimatore identificato dal nome univoco in questione, agganciato allo LSP di id 18 nel modulo Client Interface attivo verso il controller di indirizzo 191.26.37.1)

```
<EST_REQUEST>
     <EST_REQUEST_ID>44</EST_REQUEST_ID>
     <EST_OPCODE>4</EST_OPCODE>
     <REQ_DATA>
          <FOR_CI>191.26.37.1</FOR_CI>
          <FOR_LSP>18</FOR_LSP>
          <FOR_PHB>255</FOR_PHB>
          <EST_DATA>
               <EST_NAME>Identificatore univoco dello stimatore
               agganciato</EST_NAME>
          </EST_DATA>
     </REQ_DATA>
</EST_REQUEST>
```

(messaggio *remove estimator*, si richiede di rimuovere lo stimatore identificato dal nome univoco in questione, agganciato allo LSP di id 18 nel modulo Client Interface attivo verso il controller di indirizzo 191.26.37.1)

```
<EST_REQUEST>
     <EST_REQUEST_ID>45</EST_REQUEST_ID>
     <EST_OPCODE>5</EST_OPCODE>
     <REQ_DATA>
          <FOR_CI>212.212.212.212</FOR_CI>
          <FOR_LSP>99</FOR_LSP>
          <FOR_PHB>255</FOR_PHB>
          <L_UP_TH>nuova soglia superiore</L_UP_TH>
          <L_LO_TH>nuova soglia inferiore</L_LO_TH>
     </REQ_DATA>
</EST_REQUEST>
```

(messaggio *set threshold*, si fissano le soglie per lo LSP di id 99 nel modulo Client Interface attivo verso il controller di indirizzo 212.212.212.212)

```
<EST_REQUEST>
     <EST_REQUEST_ID>51</EST_REQUEST_ID>
     <EST_OPCODE>6</EST_OPCODE>
     <REQ_DATA>
          <FOR_CI>192.255.2.112</FOR_CI>
          <FOR_LSP>78</FOR_LSP>
```

```
        <FOR_PHB>255</FOR_PHB>
    </REQ_DATA>
</EST_REQUEST>
```

(messaggio *LSP e/t info,* si richiedono informazioni riguardo allo LSP di id 78 dal modulo Client Interface attivo verso il controller di indirizzo 195.255.2.112)

5.5.2 Repliche del CIM e superamento soglie

Le repliche, di ogni tipo, alle richieste di operazioni nell'ambito della stima vengono effettuate sullo estimate socket utilizzando XML. L'invio del messaggio è effettuato in due tempi come per gli altri protocolli che utilizzano XML: prima si inviano `sizeof(int)` bytes che rappresentano la lunghezza del messaggio XML successivo, quindi si invia il messaggio stesso. Come nel caso del protocollo di replica sul master socket, sono utilizzate funzioni `send()` in modalità non bloccante, per non mettere a rischio il funzionamento ciclico del CIM e dei moduli CI.

Le repliche possono essere inviate sia dal Client Interface Manager (risposta ad una richiesta *estimators list*, notifica di errori di ricezione, parsing o convalida, o impossibilità di inoltrare una richiesta verso il modulo Client Interface opportuno) che dai moduli Client Interface (risposta ad una richiesta *LSP e/t info*, notifica successo operazioni *add, adjust, remove estimator*, *set LSP threshold*, notifica errori nelle medesime). La comunicazione degli eventi di superamento soglia è invece effettuata dal parser real-time, al momento del calcolo di un nuovo valore per la stima da parte delle apposite funzioni, non ancora del tutto operative.

Definizione del tipo di documento XML (DTD)

Il messaggio XML contenente le informazioni sulla richiesta viene inviato senza la seguente DTD (alla quale fa comunque riferimento). I campi utilizzati sono variabili a seconda del tipo di replica o notifica che si sta effettuando.

```
<!DOCTYPE EST_REPLY [
   <!ELEMENT EST_REPLY (EST_REQ_ID, EST_REPLY_CODE,
   EST_REPLY_DATA?)>
      <!ELEMENT EST_REQ_ID (#PCDATA)>
      <!ELEMENT EST_REPLY_CODE (#PCDATA)>
      <!ELEMENT EST_REPLY_DATA (FROM_CI?, FROM_LSP?, FROM_PHB?,
      UNIQUE_ID?, LIST*, LSP_UP_TH?, LSP_LO_TH?)>
         <!ELEMENT FROM_CI (#PCDATA)>
         <!ELEMENT FROM_LSP (#PCDATA)>
         <!ELEMENT FROM_PHB (#PCDATA)>
         <!ELEMENT UNIQUE_ID (#PCDATA)>
         <!ELEMENT LIST (E_NAME, E_DESCRIPTION?, PROTOTYPE?,
         EST_UP_TH?, EST_LO_TH?)>
            <!ELEMENT E_NAME (#PCDATA)>
            <!ELEMENT E_DESCRIPTION (#PCDATA)>
            <!ELEMENT PROTOTYPE (PARAMETER*)>
               <!ELEMENT PARAMETER (NAME?, TYPE?, DESCRIPTION)>
                  <!ELEMENT NAME (#PCDATA)>
                  <!ELEMENT TYPE (#PCDATA)>
                  <!ELEMENT DESCRIPTION (#PCDATA)>
            <!ELEMENT EST_UP_TH (#PCDATA)>
            <!ELEMENT EST_LO_TH (#PCDATA)>
         <!ELEMENT LSP_UP_TH (#PCDATA)>
         <!ELEMENT LSP_LO_TH (#PCDATA)>
]>
```

L’elemento obbligatorio `EST_REQ_ID` è l’identificativo numerico della richiesta alla quale si sta replicando, viene posto a zero nel caso che ci siano stati errori di parsing, ricezione o convalida; l’elemento obbligatorio

`EST_REPLY_CODE` classifica il tipo di replica, come è mostrato nella sottostante tabella 5.4.

Valore	Descrizione
	Codici di errore
1	Errore nel parsing del msg. XML
2	Errore nella convalida DTD del msg. XML
3	Errore nella ricezione del msg. XML nella funzione `recv()`
4	Errore dovuto a timeout nella ricezione del msg. XML
5	Valore non legale del campo `EST_OPCODE` (codice operativo stime)
6	Valore non valido del campo `FOR_CI` (indirizzo del controller)
7	Richiesta scartata, non esiste alcun CI attivo verso il controller indicato
8	Inoltro impossibile, il CI non ha ancora eseguito una precedente richiesta
9	Il contenuto di un elemento `E_NAME` supera la lunghezza consentita
10	Il contenuto di un elemento `E_PROTOTYPE` supera la lunghezza consentita
50	Lo LSP richiesto non esiste, oppure il CI non è collegato per quello LSP
51	*Add estimator* impossibile, esiste già uno stimatore identico per quello LSP
52	*Add estimator* impossibile, il tipo di stimatore richiesto non esiste
	Notifiche
100	Il messaggio contiene la risposta ad una richiesta *estimators list*
101	Il messaggio contiene la risposta ad una richiesta *LSP e/t info*
102	***Successo*** nell'esecuzione di una richiesta *add estimator*
103	***Successo*** nell'esecuzione di una richiesta *adjust estimator*
104	***Successo*** nell'esecuzione di una richiesta *remove estimator*
105	***Successo*** nell'esecuzione di una richiesta *set LSP threshold*
	Superamento soglie
200	Soglia superiore per le misure sullo LSP superata
201	Soglia inferiore per le misure sullo LSP superata
202	Soglia superiore per lo stimatore superata
203	Soglia inferiore per lo stimatore superata

Tabella 5.4 – Codici di replica del CIM sullo estimate socket

Elemento `EST_REPLY_DATA`:

L'elemento facoltativo `EST_REPLY_DATA` contiene le informazioni aggiuntive sulla replica. Non compare in alcune notifiche di errore del Client Interface Manager in quanto non necessario o non realizzabile.

Contiene gli elementi (tutti facoltativi):

- `FROM_CI`: è assente nel caso del messaggio *estimators list*; indirizzo IPv4 del controller di interesse, in formato *dotted quad*
- `FROM_LSP`: è assente nel caso del messaggio *estimators list*; identificativo dello LSP di interesse
- `FROM_PHB`: questo campo è stato inserito per contenere le informazioni riguardo al PHB per il quale si sta effettuando la notifica (ad esempio in caso di superamento soglie); nel caso tale informazione non sia disponibile, viene posto a zero o omesso
- `UNIQUE_ID`: presente soltanto in due casi: nelle repliche di successo conseguenti al messaggio di richiesta di task *add estimator* contiene l'identificatore univoco che permette di riferirsi in seguito (in messaggi di tipo *adjust* e *remove estimator*) allo stimatore appena agganciato; nelle notifiche di superamento soglie di stima identifica lo stimatore agganciato al quale si riferisce la notifica stessa
- `LIST`: vedi sotto
- `LSP_UP_TH`: soglia superiore attualmente configurata per le misure sullo LSP (compare solo nelle repliche al messaggio *LSP e/t info*)
- `LSP_LO_TH`: soglia inferiore attualmente configurata per le misure sullo LSP (compare solo nelle repliche al messaggio *LSP e/t info*)

Nel capitolo precedente si è parlato della funzione `cm_notify()`, che copre tutte le possibili casistiche di replica e notifica sullo estimate socket. L'ultimo parametro da passare a tale funzione è un puntatore ad una stringa per gli eventuali dati aggiuntivi. La stringa deve contenere la sezione del messaggio XML interna all'elemento `EST_REPLY_DATA`, esclusi gli elementi `FROM_CI` e `FROM_LSP`.

Elemento `LIST`:

L'elemento facoltativo `LIST` può comparire più volte, e contiene un elemento di una lista di stimatori. Nel caso di replica ad un messaggio *estimators list*, contiene un elemento della lista degli stimatori disponibili per il Client Interface Manager. Nel caso di replica ad un messaggio *LSP e/t info*, contiene un elemento della lista degli stimatori configurati per quello LSP. Viene usato anche nelle notifiche di sucesso di operazioni riguardanti uno stimatore e nel superamento soglia di uno stimatore; in tali casi compare solo una volta e contiene le informazioni dello stimatore di interesse.

L'elemento `LIST` contiene gli elementi (tutti facoltativi eccetto `E_NAME`):

- `E_NAME`: a seconda dei casi, nome oppure identificatore univoco dello stimatore in questione
- `E_DESCRIPTION`: descrizione dell'algoritmo di stima; compare solo nella replica al messaggio *estimators list*
- `PROTOTYPE`: è la stessa cosa dell'elemento `PROTOTYPE` definito nel protocollo di richiesta operazioni di stima nel paragrafo 5.5.1. Per quanto riguarda l'elemento `PARAMETER`, nel caso di replica al messaggio *estimators list*, sono presenti tutti i suoi sottoelementi con la descrizione completa del prototipo della funzione di stima, parametro per parametro; nel caso di replica al messaggio *LSP e/t info* contiene solo i campi `NAME` e `DESCRIPTION`, rispettivamente con nome del parametro e valore attualmente configurato per quel parametro.
- `EST_UP_TH`: quando compare, soglia superiore configurata per lo stimatore in questione
- `EST_LO_TH`: quando compare, soglia inferiore configurata per lo stimatore in questione

Esempi di messaggi:

```
<EST_REPLY>
     <EST_REQ_ID>7</EST_REQ_ID>
     <EST_REPLY_CODE>5</EST_REPLY_CODE>
</EST_REPLY>
```

(risposta alla richiesta n°7: il codice operazione non era valido)

```
<EST_REPLY>
     <EST_REQ_ID>11</EST_REQ_ID>
     <EST_REPLY_CODE>100</EST_REPLY_CODE>
     <EST_REPLY_DATA>
          <LIST>
               <E_NAME>Media</E_NAME>
               <E_DESCRIPTION>Media aritmetica</E_DESCRIPTION>
               <PROTOTYPE>
                    <PARAMETER>
                         <NAME>Parametro1</NAME>
                         <TYPE>Tipo parametro1</TYPE>
                         <DESCRIPTION>Descrizione
                         parametro1</DESCRIPTION>
                    </PARAMETER>
                    <PARAMETER>
                         <NAME>Parametro2</NAME>
                         <TYPE>Tipo parametro2</TYPE>
                         <DESCRIPTION>Descrizione
                         parametro2</DESCRIPTION>
                    </PARAMETER>
               </PROTOTYPE>
          </LIST>
          <LIST>
               <E_NAME>Lineare</E_NAME>
               <E_DESCRIPTION>Descrizione h.r.</E_DESCRIPTION>
               <PROTOTYPE>
                    <PARAMETER>
                         <NAME>Parametro1</NAME>
                         <TYPE>Tipo parametro1</TYPE>
                         <DESCRIPTION>Descrizione
                         parametro1</DESCRIPTION>
                    </PARAMETER>
```

```
                    <PARAMETER>
                         <NAME>Parametro2</NAME>
                         <TYPE>Tipo parametro2</TYPE>
                         <DESCRIPTION>Descrizione
                         parametro2</DESCRIPTION>
                    </PARAMETER>
               </PROTOTYPE>
          </LIST>
     </EST_REPLY_DATA>
</EST_REPLY>
```

(risposta alla richiesta n°11, di tipo *estimators list*; la replica contiene la lista degli stimatori supportati dal CIM)

```
<EST_REPLY>
     <EST_REQ_ID>14</EST_REQ_ID>
     <EST_REPLY_CODE>101</EST_REPLY_CODE>
     <EST_REPLY_DATA>
          <FROM_CI>168.30.192.1</FROM_CI>
          <FROM_LSP>2010</FROM_LSP>
          <LIST>
               <E_NAME>Identificatore univoco stim. 1</E_NAME>
               <PROTOTYPE>
                    <PARAMETER>
                         <NAME>Parametro1 configurato</NAME>
                         <TYPE>Tipo parametro1</TYPE>
                         <DESCRIPTION>Valore parametro1
                         configurato</DESCRIPTION>
                    </PARAMETER>
                    <PARAMETER>
                         <NAME>Parametro2 configurato</NAME>
                         <TYPE>Tipo parametro2</TYPE>
                         <DESCRIPTION>Valore parametro2
                         configurato</DESCRIPTION>
                    </PARAMETER>
               </PROTOTYPE>
               <EST_UP_TH>Soglia sup. stim. 1</EST_UP_TH>
               <EST_LO_TH>Soglia inf. stim. 1</EST_LO_TH>
          </LIST>
          <LIST>
               <E_NAME>Identificatore univoco stim. 2</E_NAME>
```

```
            <PROTOTYPE>
                <PARAMETER>
                    <NAME>Pa rametro1 configurato</NAME>
                    <TYPE>Tipo parametro1</TYPE>
                    <DESCRIPTION>Valore parametro1
                    configurato</DESCRIPTION>
                </PARAMETER>
                <PARAMETER>
                    <NAME>Pa rametro2 configurato</NAME>
                    <TYPE>Tipo parametro2</TYPE>
                    <DESCRIPTION>Valore parametro2
                    configurato</DESCRIPTION>
                </PARAMETER>
            </PROTOTYPE>
            <EST_UP_TH>Soglia sup. stim. 2</EST_UP_TH>
            <EST_LO_TH>Soglia inf. stim. 2</EST_LO_TH>
        </LIST>
        <LSP_UP_TH>Soglia sup. LSP</LSP_UP_TH>
        <LSP_LO_TH>Soglia inf. LSP </LSP_LO_TH>
    </EST_REPLY_DATA>
</EST_REPLY>
```

(risposta alla richiesta n°14, di tipo *LSP E/T info* per lo LSP 2010 ed il CI connesso al controller di indirizzo 168.30.192.1: la replica contiene gli stimatori agganciati allo LSP 2010 e le soglie configurate per lo LSP 2010)

```
<EST_REPLY>
    <EST_REQ_ID>22</EST_REQ_ID>
    <EST_REPLY_CODE>102</EST_REPLY_CODE>
    <EST_REPLY_DATA>
        <FROM_CI>192.168.30.2</FROM_CI>
        <FROM_LSP>700</FROM_LSP>
        <UNIQUE_ID>Identificatore univoco</UNIQUE_ID>
    </EST_REPLY_DATA>
</EST_REPLY>
```

(risposta alla richiesta n°22, di tipo *add estimator*, da parte del CI connesso al controller di indirizzo 192.168.30.2: lo stimatore richiesto è stato

agganciato con successo allo LSP 700; la replica contiene l'identificatore univoco per il nuovo stimatore)

```
<EST_REPLY>
     <EST_REQ_ID>30</EST_REQ_ID>
     <EST_REPLY_CODE>103</EST_REPLY_CODE>
     <EST_REPLY_DATA>
          <FROM_CI>192.168.30.3</FROM_CI>
          <FROM_LSP>3020</FROM_LSP>
     </EST_REPLY_DATA>
</EST_REPLY>
```

(risposta alla richiesta n°30, di tipo *adjust estimator*, da parte del CI connesso al controller di indirizzo 192.168.30.3: i parametri dello stimatore richiesto, agganciato allo LSP 3020, sono stati modificati con successo)

```
<EST_REPLY>
     <EST_REQ_ID>37</EST_REQ_ID>
     <EST_REPLY_CODE>104</EST_REPLY_CODE>
     <EST_REPLY_DATA>
          <FROM_CI>192.168.30.4</FROM_CI>
          <FROM_LSP>4570</FROM_LSP>
     </EST_REPLY_DATA>
</EST_REPLY>
```

(risposta alla richiesta n°37, di tipo *remove estimator*, da parte del CI connesso al controller di indirizzo 192.168.30.4: lo stimatore richiesto, agganciato allo LSP 4570, è stato rimosso con successo)

```
<EST_REPLY>
     <EST_REQ_ID>44</EST_REQ_ID>
     <EST_REPLY_CODE>105</EST_REPLY_CODE>
     <EST_REPLY_DATA>
          <FROM_CI>192.168.30.5</FROM_CI>
          <FROM_LSP>320</FROM_LSP>
     </EST_REPLY_DATA>
```

```
</EST_REPLY>
```

(risposta alla richiesta n°44, di tipo *set LSP threshold*, da parte del CI connesso al controller di indirizzo 192.168.30.5: le soglie per lo LSP 320 sono state modificate con successo)

```
<EST_REPLY>
     <EST_REQ_ID>0</EST_REQ_ID>
     <EST_REPLY_CODE>200</EST_REPLY_CODE>
     <EST_REPLY_DATA>
          <FROM_CI>192.168.30.11</FROM_CI>
          <FROM_LSP>2000</FROM_LSP>
     </EST_REPLY_DATA>
</EST_REPLY>
```

(Notifica di un evento di superamento soglia superiore nelle misure sullo LSP 200, da parte dei parser del CI connesso al controller di indirizzo 192.168.30.11; si noti che il campo `EST_REQ_ID` è posto a zero)

```
<EST_REPLY>
     <EST_REQ_ID>0</EST_REQ_ID>
     <EST_REPLY_CODE>203</EST_REPLY_CODE>
     <EST_REPLY_DATA>
          <FROM_CI>192.168.30.101</FROM_CI>
          <FROM_LSP>7050</FROM_LSP>
          <UNIQUE_ID>Identificatore univoco</UNIQUE_ID>
     </EST_REPLY_DATA>
</EST_REPLY>
```

(Notifica di un evento di superamento soglia inferiore nello stimatore identificato dal campo `UNIQUE_ID` , da parte del CI connesso al controller di indirizzo 192.168.30.101; si noti che il campo `EST_REQ_ID` è posto a zero)

Si noti che la sezione del modulo CI riguardante l'esecuzione delle operazioni di stima non è ancora stata del tutto implementata al momento della

scrittura, e non rientra negli scopi previsti per questa tesi di laurea. Pertanto, pur essendo pronto il supporto per il funzionamento del protocollo (sia in fase di ricezione che di invio), alcuni di questi messaggi esemplificati non sono ancora operativi.