Direct Socket Stack

The Direct Socket Stack vertical application integration was introduced and defined in [MUP - D2.3], section 6.
It provides a simple and lean interface between the applications and the UNI proxy. This concept is illustrated in Figure 1.

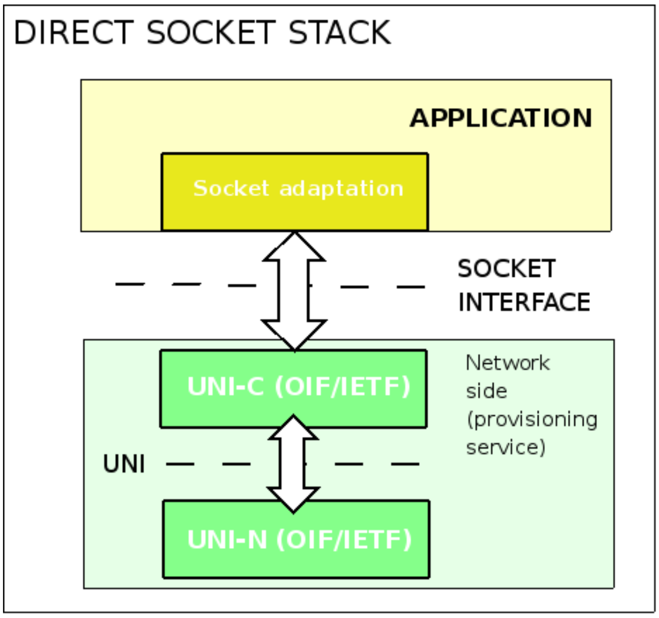

Figure 1 Socket stack vertical application integration model

This approach provides no traffic engineering capabilities due to the missing adaptation between application and socket layer. Removing the adaptation layer might however have the benefit that applications can reduce the latency of requesting a connection. This latency might be found in the other stack implementations, due to the added delay by the web service server. Providing a low latency connection setup technique might prove to be an important parameter for some applications. In the following sub section a description of the socket interface protocol is given together with an example that will highlight the use of the interface.

Characteristics

As a mean of communication this protocol uses messages which contain a varying number of objects carrying information. As operation of the protocol is based on request-reply communication, there are three types of messages:

- Request - REQ (1),
- Positive reply - ACK (2),
- Negative reply - NACK (3).

These types of messages apply to one of four defined commands:

- Path setup – CREATE (1),
- Path teardown – DELETE (2),
- RSVP Agent status – STATUS (3).

The messages mostly correspond to UNI functionality and Table 1 provides meaning of each message.

Table 1 Data types

CMD **TYPE**	1 – CREATE	2 – DELETE	3 – STATUS
1 - REQ	**path setup is requested:** **- from network side** **- from application side**	**- path teardown is requested from application side** **- network side informs about path deletion**	path(es) status is requested only from application side
2 - ACK	**- application agrees on path creation** **- network confirms path creation**	**- network confirms path deletion** **- application politely responses**	network responses with path status
3 - NACK	- application disagrees on path creation - network informs about path creation failure	- application informs about path mismatch - network informs about path mismatch	network informs about path mismatch

The RSVP agent activity upon receiving the management packet is shown on Figure 2. The most important on this scheme is that a particular request command generates a related response type. There are some details not shown on this figure. The diffrence is because STATUS REQ ask of status of particular path so there is no iteration over all paths but only sending STATUS NACK or STATUS ACK.

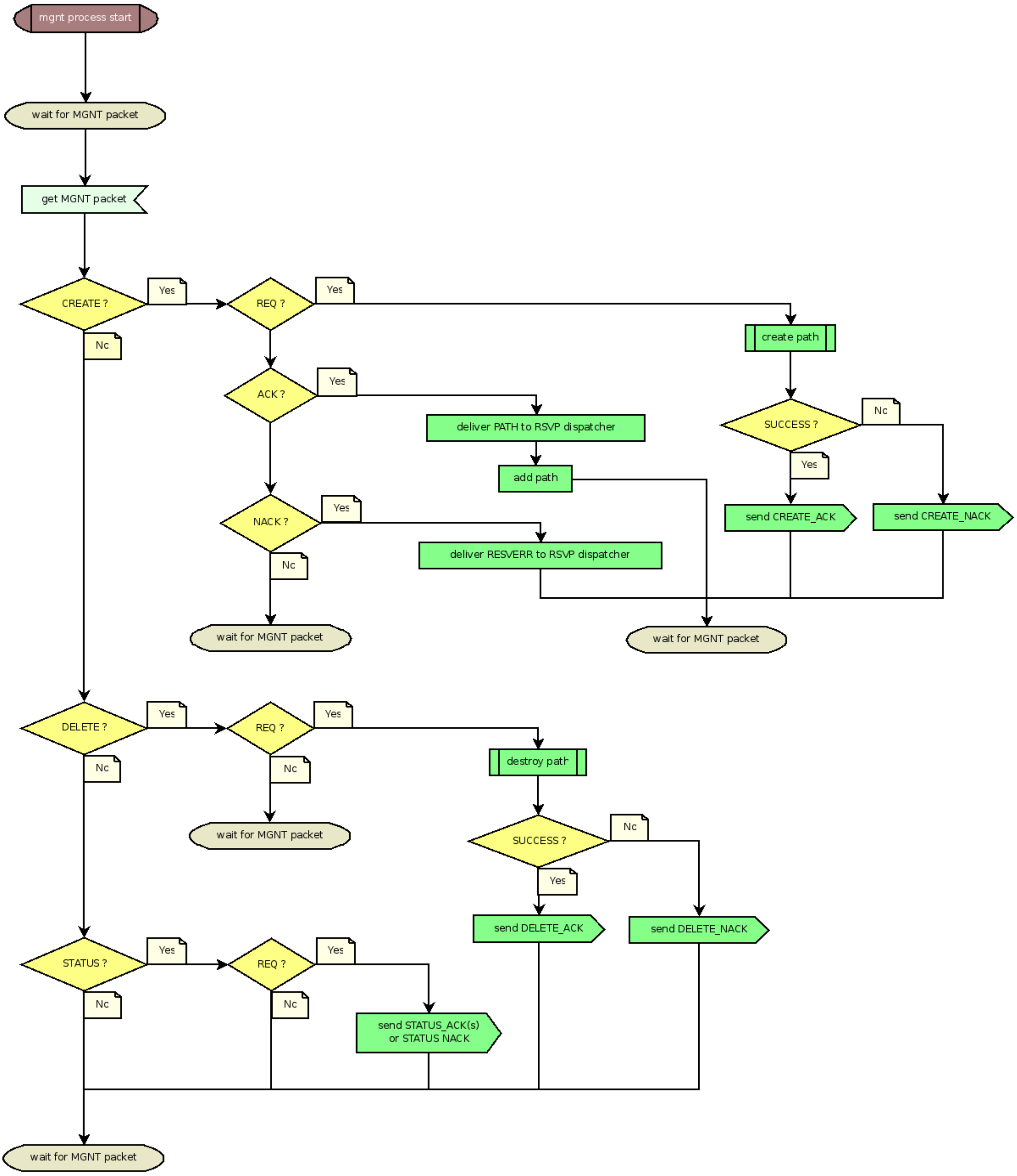

Figure 2 The RSVP agent management process activity diagram

Protocol message

Each request or reply is sent in a message. Each message can contain only one request or reply. The message structure illustrated in Figure 3 resembles a RSVP packet – it contains a constant-structure header and a variable set of objects.

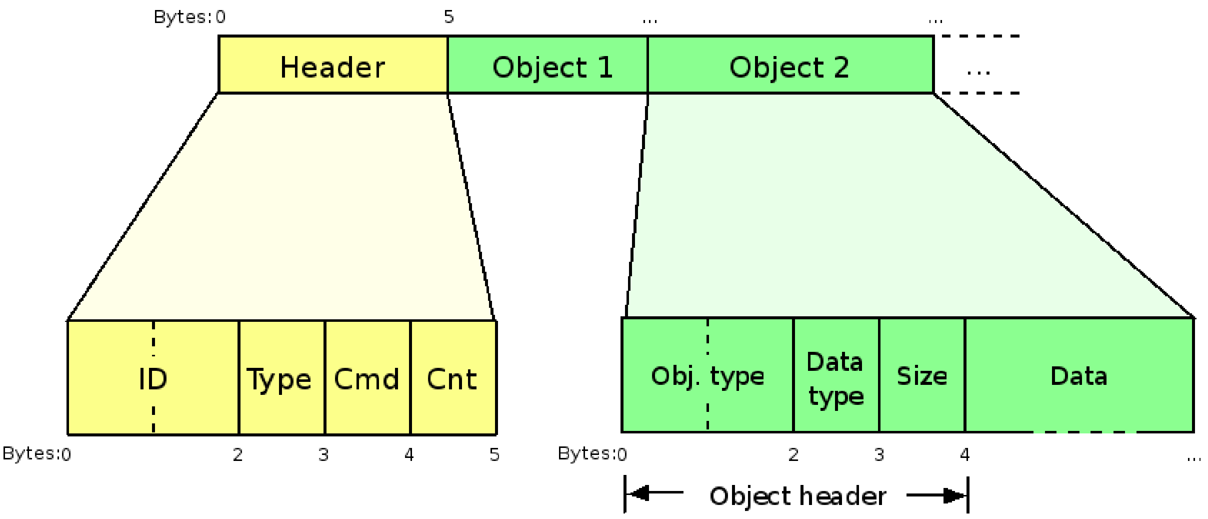

Figure 3 Message structure

A header contains information about message ID, message type, command and number of objects in a message. The message ID field is used to associate a reply message with a particular request message. The type and command were discussed before in “Characteristics”. The number of objects in the message is contained to help interpreting the message and to detect possible errors (e.g. when the message is too short). The number of objects can have a value of zero in the case of a STATUS message only: in STATUS-REQ, which doesn't need any data and STATUS-ACK, when the RSVP Agent has nothing to report. Objects are placed always after the header. Each object contains one data element – a number, text or undefined portion of data. This is called data type and it is defined in the “Data type” field in the object header. In the case the text or undefined data is enclosed in the object, the “Size” field is used to define its length (i.e. only the length of the data portion of the object). For the value transported to be useful, the “Object type” field specifies the meaning of data carried in the object. In the case of integer numbers being enclosed in the object, network byte order (big-endian) is always used.

Object types and data types

Table 2 defines the possible object types that can be used within the object structure shown in . Object types 1- 25 were defined in [MUP - D2.3] section 6.1.3, objects from 26 to 33 are added to provide extended functionality of the UNI Proxy.

Table 2 Object types

Obj. type	***Data type***	***Object contents***	***Relation to RSVP message fields***
1	3	RSVP Session destination address	SESSION/destination address
2	3	RSVP Session source address	SESSION/extended tunnel id SENDER TEMPLATE/tunnel sender addr
3	3	RSVP Agent's	HOP/neighbour address

Obj. type	*Data type*	*Object contents*	*Relation to RSVP message fields*
		outgoing interface	HOP/forward-reverse interface address
4	3	Path refresh period	TIME VALUES/refresh interval
5	5	Token bucket rate	Sender TSPEC (IETF RSVP) TOKEN FLOWSPEC (IETF RSVP)
6	5	Token bucket size	Sender TSPEC (IETF RSVP) TOKEN FLOWSPEC (IETF RSVP)
7	5	Peak data rate	Sender TSPEC (IETF RSVP) TOKEN FLOWSPEC (IETF RSVP)
8	7	Upstream label (for bidirectional paths)	UPSTREAM LABEL/label
9	1	LSP encoding type	GENERALIZED LABEL REQUEST/ encoding type
10	1	LSP switching type	GENERALIZED LABEL REQUEST/ switching type
11	2	LSP GPID	GENERALIZED LABEL REQUEST/ gpid
12	6	Path name	SESSION ATTRIBUTE/name
13	3	ERO address	ERO/address
14	1	ERO prefix length	ERO/prefix length
15	1	ERO strict/loose	ERO/strict/loose flag
16	3	RSVP hop address	HOP/TLV object
17	3	Error node address	ERROR SPEC/node address
18	1	Error flags	ERROR SPEC/error flags
19	1	Error code	ERROR SPEC/error code
20	2	Error value	ERROR SPEC/error value
21	3	Multi-part message packet number (0..N-1)	-
22	3	Multi-part message total packet count (N)	-
23	2	RSVP tunnel ID	SESSION/tunnel id
24	3	Traffic destination address	-
25	3	Traffic reverse destination address	-

Obj. type	***Data type***	***Object contents***	***Relation to RSVP message fields***
26	3	Data forward/reverse interface ID	Control out interface identifier HOP/forward-reverse interface identifier
27	3	Source TNA	GENERALIZED UNI/source TNA CALL ID/source TNA
28	3	Destination TNA	GENERALIZED UNI/destination TNA
29	3	Call ID - part 1	CALL ID/local identifier
30	3	Call ID - part 2	CALL ID/local identifier
31	2	Sender LSP ID	SENDER TEMPLATE/lsp id
32	1	Setup priority	SESSION ATTRIBUTE/setup priority flag
33	1	Hold priority	SESSION ATTRIBUTE/hold priority flag
34	2	Path state	-
35	1	Protection service level	GENERALIZED UNI/protection service level
36	5	**Ethernet committed information rate**	**Sender TSPEC (OIF UNI2.0Eth.), FlowSpec (OIF UNI2.0Eth.)**
37	5	**Ethernet committed burst size**	**Sender TSPEC (OIF UNI2.0Eth.), FlowSpec (OIF UNI2.0Eth.)**
38	5	**Ethernet excess information rate**	**Sender TSPEC (OIF UNI2.0Eth.), FlowSpec (OIF UNI2.0Eth.)**
39	5	**Ethernet excess burst size**	**Sender TSPEC (OIF UNI2.0Eth.), FlowSpec (OIF UNI2.0Eth.)**
40	2	**Ethernet granuality**	**Sender TSPEC (OIF UNI2.0Eth.), FlowSpec (OIF UNI2.0Eth.)**
41	1	**Ethernet index**	**Sender TSPEC (OIF UNI2.0Eth.), FlowSpec (OIF UNI2.0Eth.)**
42	1	**Ethernet profile**	**Sender TSPEC (OIF UNI2.0Eth.), FlowSpec (OIF UNI2.0Eth.)**

Table 3 depicts the different data types that can be used in the object structure.

Table 3 Data types

Data type	***Data size (bytes)***	***Description***
1	1	Unsigned char

Data type	*Data size (bytes)*	*Description*
2	2	Unsigned short
3	4	Unsigned integer
4	8	Unsigned long integer
5	4	Float
6	“size”	Text
7	“size”	Undefined data

It is assumed, that objects of each type always appear with the corresponding data type. In the case of integer and float values this field might seem redundant, but in case of future protocol revisions it will help older implementation parse messages containing unknown objects. When the program which parses the message knows the type and size, it can easily skip the unknown object and analyse further ones. Even if the data type is not known, the “Data size” field provides enough information to skip the object.

Table 4 Required objects in messages

Message	*Required objects*
CREATE-REQ	1, (8), 23
CREATE-ACK/NACK	23
DELETE-REQ	23
DELETE-ACK/NACK	23
STATUS-REQ	---
STATUS-ACK	21, 22, 23 (if any)

Depending on particular command and message types some objects may be optional, others may be required.

For most message types only RSVP tunnel ID object is obligatory. The CREATE-REQ message has to include a RSVP session destination address object and for bidirectional path an upstream label object also. If there is lack of RSVP session source address or RSVP Agent's outgoing interface objects in the CREATE REQ message then related RSVP fields are filled with an interface address used by the RSVP agent.

Traffic destination address and traffic reverse destination address objects are only used when data labelling module (the mpls-linux packet) is activated in the RSVP agent.

There is also a new Path state object which is used only in STATUS ACK messages. It contain information about path state in the agent (see Table 4).

Table 5 Path state object values

Path state value	***Description***
0	Null state
1	Initiated state
2	Path send
6	Path up
11	Path down

Path state 'Null' and 'Initiated' should not appeare in STATUS ACK message when the agent is work correctly. We will see 'Path send' status only when there will not be any RESV responses. 'Path up' means that path is establishes and working. 'Path down' is returned after any path termination.