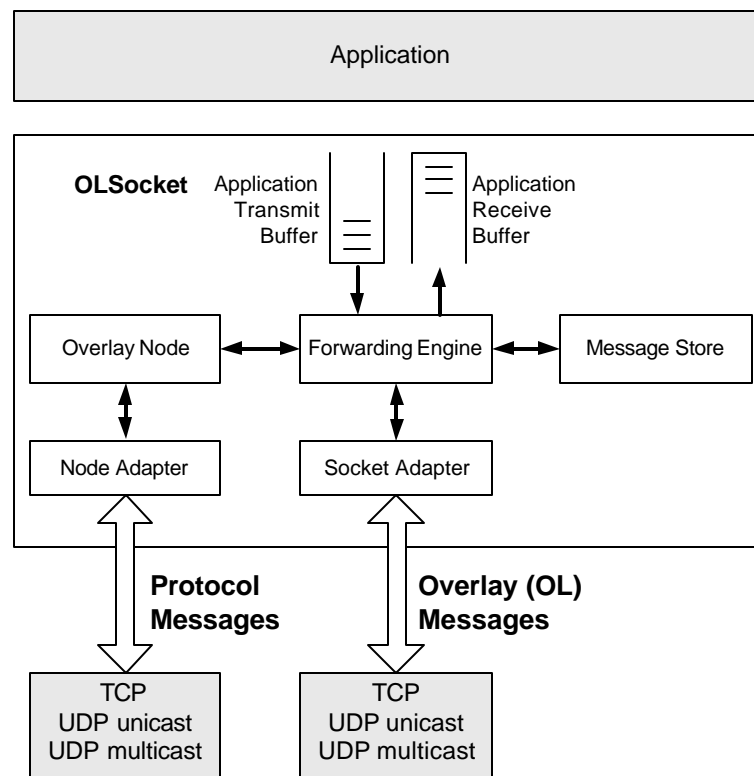


## CHAPTER 5

## Overlay (OL) Message Formats

## 5.1 OVERVIEW

Data between overlay sockets is exchanged as formatted messages. As shown in Figure 5.1, an overlay socket provides two endpoints for communications. The node adapter exchanges protocol messages with node adapters at other overlay sockets. The socket adapter exchanges overlay (OL) messages with socket adapters at other overlay sockets. The handling of protocol and overlay messages is completely separate.



**Figure 5.1.** Messages transmitted by the overlay socket.

- **Protocol Messages:** These are messages that are exchanged between overlay nodes. Protocol messages are mostly sent to neighbors in the overlay network. Some overlay protocols may define protocol messages that are broadcast to all nodes in the overlay network (e.g., Beacon messages in the HC protocol). The format of protocol messages are specific to a given overlay protocol, and are completely defined by the overlay node component of the overlay node (see Figure 5.1). Currently, there are two types of protocol messages, DT message and HC message.
- **Overlay (OL) Messages:** These are messages that are exchanged between socket adapters of overlay sockets that are neighbors in the overlay network, and are forwarded by the forwarding engine. All application data is transmitted as OL messages. Overlay messages can be multicast to all overlay sockets in an overlay network, unicast to one

specific overlay socket. In addition to transferring application data OL messages can be used for monitoring and control functions, e.g., to perform functions similar to traceroute or provide error reporting similar to ICMP. The message format of OL messages is the same across all versions of HyperCast 2.0 overlay sockets.

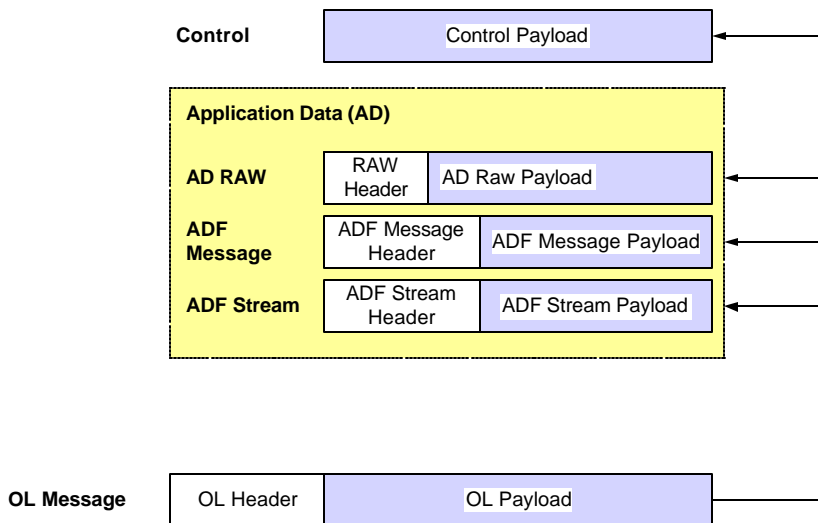
Currently, both protocol and overlay messages are transmitted using an Internet transport protocol, i.e., TCP, UDP unicast, or UDP multicast. In principle, non-Internet protocols can be used to provide transport services for overlay sockets. The adapters (see Figure 5.1) are responsible for encapsulating messages and interfacing to the underlying transport protocol. The transport addresses of the underlying transport protocol are maintained at the overlay node.

This chapter is about OL messages. The message formats of overlay messages are described in the sections, which contain the overlay protocols.

**Note:** This chapter describes many message formats which are not used in HyperCast 2.0. Specifically, the ADF packet format is not used in HyperCast 2.0.

### 5.2 OVERLAY MESSAGES

All overlay (OL) messages have a header and a payload field. The header of an OL message specifies, among others, the message format, the delivery mode, and source and destination addresses. All addresses in the OL message header are logical addresses. Recall that logical addresses are defined by the overlay topology and the overlay protocol. In the DT protocol, logical addresses are a pair of coordinates (x,y), where x and y are numbers. In the HC protocol, logical addresses are a 32-bit binary number.



**Figure 5.2.** Types of OL Messages.

The payload of an HyperCast 2.0 OL message can have a different type, each type has its own header format.

- **Application Data (AD) Messages:** These messages perform delivery of application-level data. There are three basic types of AD messages:
  - **Raw AD messages:** Raw AD messages send unformatted application level data, and have a minimal header. If Raw AD messages are used, the application

program is responsible for maintaining sequence numbers, performing acknowledgments, and others.

**Note:** In HyperCast 2.0, only Raw AD messages are used.

The notion of ADF messages is tightly coupled to the message store, which is not part of HyperCast 2.0. The discussion of ADF messages is not relevant to the HyperCast 2.0 software.

- **ADF messages:** ADF messages are formatted AD messages. Each ADF message has a unique identifier, and may contain control information such as acknowledgements. ADF messages may contain application data or control information (such as acknowledgments).

ADF messages are used if the overlay sockets provides services such as retransmission, or others. Each ADF message carries an identifier for a finite state machine, that can interpret the format of that message.

- **ADF stream messages:** ADF stream messages are formatted AD messages, which define a sequence of formatted AD messages. The difference to an ADF message is that, instead of a message identifier, the ADF stream message has a stream identifier and a sequence number. ADF messages are used to implement streaming services.

Each ADF stream message carries an identifier for a finite state machine, that can interpret the format of the message and associate the message with other message from the same stream.

- **Control Messages:** Control messages are sent through the overlay to provide maintenance, control, and management. They can be used to implement utilities such as ping, traceroute, and can provide reporting functions similar to ICMP. An important role of control message is to help with congestion and rate control services.

**Note:** HyperCast 2.0 does not specify or implement control message formats.

The delivery mode of an OL message can be specified as multicast, Flood, Unicast, or Anycast. A multicast OL message is transmitted from the source to all overlay sockets in the overlay network, using routing trees that are embedded in the overlay network. A unicast message is a point-to-point message between any two overlay sockets in an overlay network. A flood message is transmitted using a simple flooding algorithm.

The design of the OL message header is intended to be similar to the IPv6 packet format. Specifically, the OL header uses the notion of extension headers, to concatenate multiple headers. Currently the defined OL message headers are shown in Table 5.1. Of these, only the RAW format is used in HyperCast 2.0. Future OL message headers are listed in Table 5.2.

In HyperCast, any number of extension headers can be concatenated. Especially, using extension headers, multiple ADF messages may be transmitted in the same OL message.

Table 5.1. OL Message header extensions defined in HyperCast 2.0.

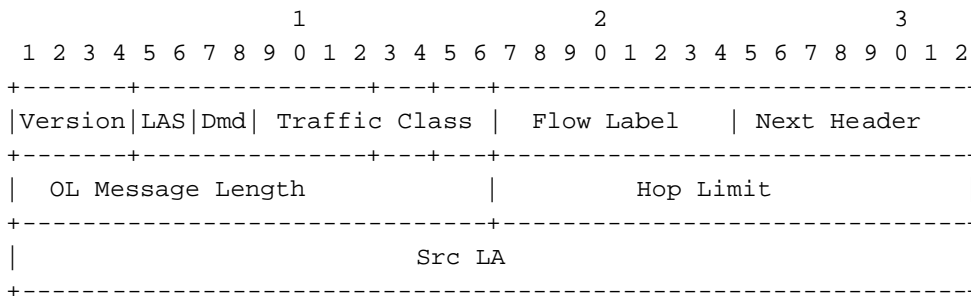
Next Header (hex)	Next Header	Used in HyperCast 2.0?
0x00	No Header to follow (If the first Next Header field is set to 0x00, the OL message consists only of the Header)	Yes
0x01	8 bytes MessageID ADF Message	No
0x02	10 bytes MessageID ADF Message	No
0x03	8 bytes StreamID ADF Stream	No
0x04	10 bytes StreamID ADF Stream	No
0x05	Raw	Yes

Table 5.2. Future OL Message header extensions.

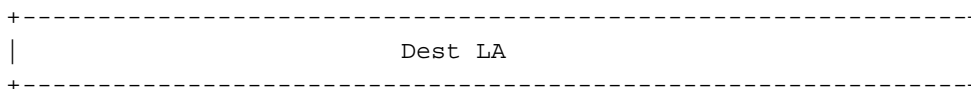
Next Header
Hop-by-Hop Options header
Jumbo OL Messages (Messages with large Authentication Headers)
Control headers
Source Routing Headers

### 5.3 FORMAT OF OL MESSAGES

This section describes the OL message format. All OL messages have a header that is at least twelve bytes long and a payload field. Multiple payloads can be concatenated by using extension headers, as indicated in the Next Header field.



Only if Dmd=0x2:



Only if Dmd=0x1:

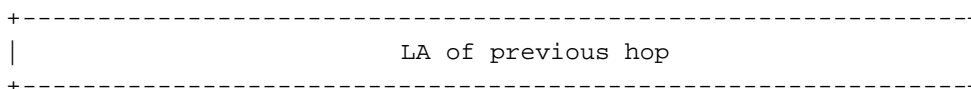


Figure 5.3. OL Message Header.

---

<b>Version (4 bit):</b>	Version of the protocol. The current version is 0x0
<b>LAS (2 bit):</b>	Size of the logical address field
<b>Dmd (2 bit)</b>	Delivery mode. Currently defined delivery modes are: <ul style="list-style-type: none"> <li>0x0 Multicast</li> <li>0x1 Flood</li> <li>0x2 Unicast</li> <li>0x3 Anycast (not implemented)</li> </ul>
<b>Traffic Class (8 bit):</b>	Specifies Quality of Service (QoS) class ( <i>not used</i> )
<b>Flow Label (16 bit)</b>	Assigned for QoS purposes ( <i>not used</i> )
<b>OL Message Length (16 bits)</b>	Length of entire payload following this header
<b>Next Header (8 bit)</b>	Specifies the type of the following header. Currently the following headers are allowed: <ul style="list-style-type: none"> <li>• <b>Raw</b> ( 0x05 ) Identifies an AD RAW message. This data is interpreted by the application program.</li> <li>• <b>AD Message</b> ( 0x01 , 0x02 ) Contains a formatted AD message that uses the Message ADF format. There are two versions, with different lengths for the Message ID field.</li> <li>• <b>AD Stream</b> ( 0x03 , 0x04 ) Contains a formatted AD message or an ADF message that uses the Stream ADF format. There are two versions, with different lengths for the Stream ID field.</li> </ul>
<b>Hop Limit (8 bit):</b>	Corresponds to Time-to-Live (same as in IP data)
<b>SrcLA ((LAS+1)*4 bytes)</b>	Logical address of the source of the OL Message
<b>DestLA ((LAS+1)*4 bytes)</b>	Logical address of the destination of the OL Message

**5.4 FORMAT OF AD RAW MESSAGES**

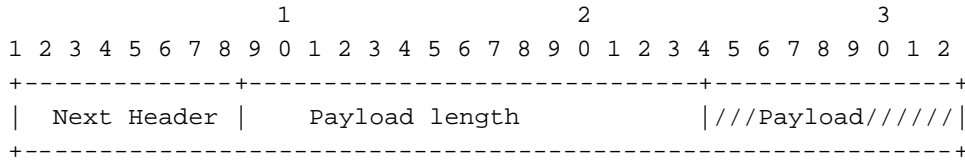


Figure 5.4. Message Header of AD Raw Message.

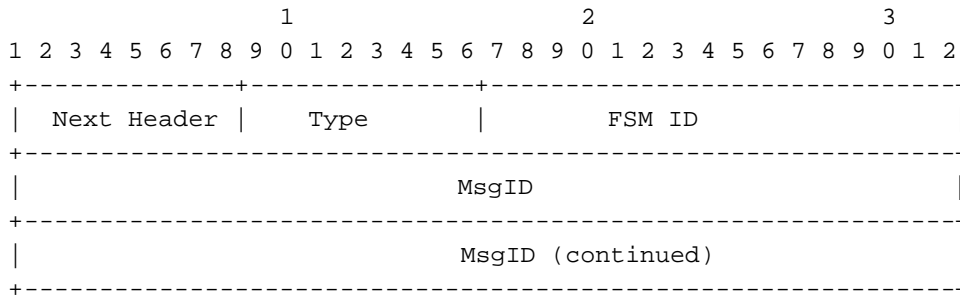
**Payload Length (16 bits)** Length of Payload field in bytes.

**Next Header (8 bit)** Specifies the type of the next header.

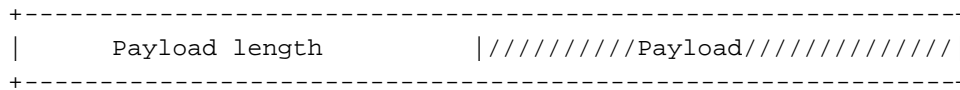
**5.5 FORMAT OF ADF MESSAGES**

ADF messages are associated with a Finite State Machine (FSM), which knows how to process and interpret the ADF message. The FSM is identified by a FSM Identifier. For each FSM, there may be different ADF message types. These are identified by the Type field. Note that the Type field is specific to the FSM ID.

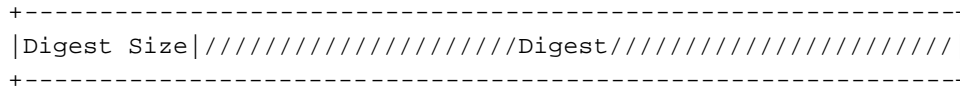
**Recall:** The finite state machines are part of the message store which is not included in HyperCast 2.0.



Only if Type == 0x80 or 0x81 or 0x82



Only if Type == 0x0B



Only if Type == 0x01 ... 0x0A 0x0C \_0x0F

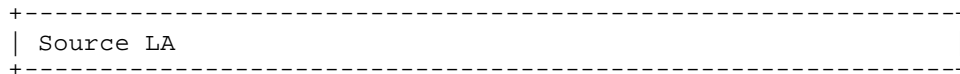


Figure 5.5. Message Header of ADF Message.

<b>Next Header (8 bit)</b>	Specifies the type of the next header.
<b>Payload Length (16 bits)</b>	Length of Payload field in bytes.
<b>FSM ID (16 bits)</b>	<p>Identifies a Finite State machine, which processes this message. The finite state machine is implemented in the message store. Each finite state machine provides a certain service, which is defined in the message store.</p> <ul style="list-style-type: none"> <li>• 0x00: No service (best effort delivery)</li> <li>• 0x01: Local acknowledgements (Local ACK)</li> <li>• 0x02: Global acknowledgments (Global ACK)</li> <li>• 0x03: Stateful flooding</li> <li>• 0x03: Synchronization</li> <li>• 0x04: Incast (reverse multicast)</li> </ul>
<b>Type (8 bits)</b>	<p>ADF Message type. The interpretation of the message type may be dependent on the FSM ID. The following types are defined for all FSM's:</p> <ul style="list-style-type: none"> <li>• <b>Payload Data</b> (Type = 0x80): The message contains payload data</li> <li>• <b>Retransmission</b> (Type = 0x81): The message contains a retransmission of payload data</li> <li>• <b>Mergeable Payload</b> (Type = 0x82): The payload of this message is "mergeable". This type is used for transmission of Incast messages.</li> </ul> <p><b>The FSM specific types are specified in Table</b></p>

For each service, different types of ADF\_Control messages are necessary.

Table 5.3. Interpretation of Type field in ADF message header.

Service	FSM ID	Definition and Interpretation of Type field
No Service	0x00	None defined:
Local ACK	0x01	0x01: Local NACK = Request for Retransmission
		0x02: Local ACK
		0x03: Request Local ACK = Request for sending a Local ACK
		0x04: Local_Reset = Resets state machine for this message
Global ACK	0x02	0x05: Global NACK = Request for Retransmission
		0x06: Global ACK (full)

Service	FSM ID	Definition and Interpretation of Type field
		0x07: Global ACK (partial)
		0x08: Global ACK (partial done)
		0x09: Request Global ACK = Request for sending a Global ACK
		0x0a: Global_Reset = Resets Message
Stateful Flooding	0x03	None
Synchronization	0x04	0x0b: QuerySyncAll = Requests Information on Message Store of Neighbor
		0x0c: QueryMsgSync = Requests State Information for a specific list of messages
		0x0d: HaveIt = Sent in response to a QueryMsgSyncAll, QueryMsgSync, or sent without any request
		0x0e: Don'tHaveIt = Sent in response to a QueryMsgSyncAll, QueryMsgSync, or sent without any request
		0x0f: NACK_SYNC
Incast	0x05	None

**MsgID (8 or 10 bytes):**

Message Identifier. This should be a unique number, for example, a random number, or “Physical address + Transport layer Port number + Counter”. Uniqueness is not enforced.

**Sender LA:**

Logical address of the overlay socket which originally generated this message. This field is used for control data (such as acknowledgments) which are sent to the sender of an ADF message. The Sender LA is needed to calculate the next hop (parent) in the tree with Sender LA as root. The LAS size is known from the OL Header.

