

## CHAPTER 4

## Managing Identifiers and Properties of Overlay Networks

### 4.1. OVERVIEW

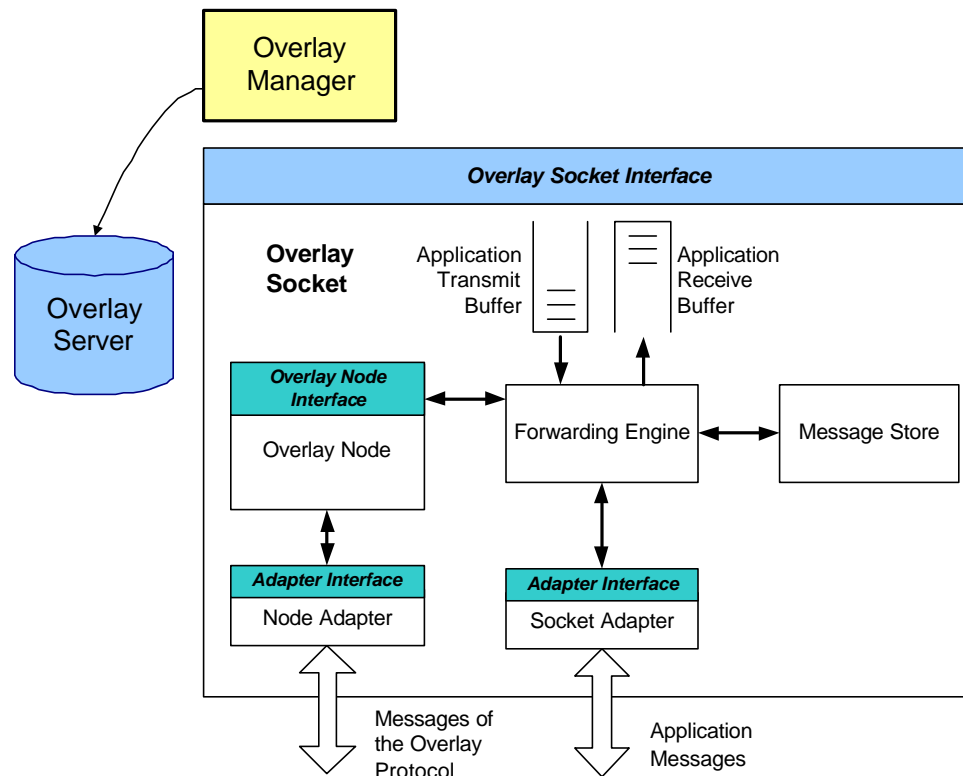


Figure 4.1. Overlay Manager and Overlay Server.

An overlay network is a collection of overlay sockets, where the overlay sockets are connected with an overlay protocol. An overlay network is uniquely identified by an overlay identifier, called Overlay ID. Each overlay network is associated a set of attributes which specify the properties of the overlay sockets that participates in the overlay network. These attributes include the type of overlay protocol (Hypercube, Delaunay triangulation, etc.) and the type of socket adapter (CO, CL, etc.).

A new overlay network is created by generating a unique overlay ID and by specifying the attributes of the overlay network that are associated with the overlay ID. To join an existing overlay network, a node must obtain the overlay ID and the attributes of the overlay network. The attributes of an overlay network must be known at the time when an overlay socket is created.

The overlay manager is a component that is responsible for creating new overlay networks, and for setting and retrieving attributes that are associated with an overlay network. The overlay manager is not part of an overlay socket (see Figure 4.1). From an

implementation perspective, the overlay manager creates an object that is called the configuration object. This object contains the attributes which specify the overlay socket. When an overlay socket is created, the configuration object is passed as a parameter of the constructor.

## 4.2. OVERLAY ID

The overlay ID is a string that identifies an overlay network. It can be used as a key to look up the attributes of an overlay network. The overlay ID should be a globally unique identifier. However, HyperCast does not assume or enforce a particular format of the overlay ID. The following are guidelines for creating unique overlay IDs.

### Examples:

- A suggested method to achieve global uniqueness of an overlay ID is to use the local IP address and a local timestamp from the process that created the overlay network. For example, a host with an IP address of 128.143.71.29 could create a unique overlay ID by concatenating a timestamp to this address (e.g. 128.143.71.29:997831668759, where the timestamp is a long integer)
- If hosts use a server to create an overlay ID, then the overlay ID should be the IP address of the server concatenated with a timestamp.
- If the overlay network employs IP multicast messages in its overlay protocol, then the IP multicast address and the port number can be used as the overlay ID. An example of an overlay protocol that uses IP multicast messages is the HC protocol.

## 4.3. ATTRIBUTES OF AN OVERLAY SOCKET

Each overlay socket is characterized by a set of attributes. Since attributes specify the components of overlay sockets, the attributes must be known when a new overlay socket is created.

All attributes have a name and a value, both of which are strings. For example, the overlay protocol of an overlay socket is determined by an attribute with name "NODE." If NODE is set to "HC2-0," then the overlay socket will run the HC protocol. If NODE has value "DT2-0," then the overlay socket will run the Delaunay triangulation protocol.

We distinguish between two types of attributes: key attributes and configurable attributes. Key attributes are bound to an overlay ID and are determined when an overlay ID is created. Key attributes cannot be modified after an overlay ID has been created for an overlay network. An overlay socket that participates in an overlay network can modify configurable attributes.

Attributes can be further specified by additional attributes. For example, the attribute "HC2-0.Timeout" denotes an attribute called "Timeout" for the attribute HC2-0. We refer to attributes that describe parameters of another attribute as subattributes. Subattribute names follow a period. The name of the attribute that is described by the subattribute is on the left of the period. Subattributes can be nested, meaning that they can have subattributes themselves. For example, an attribute NameAttr1.NameAttr2.Attr3 specifies that Attr3 is a subattribute of an attribute with value NameAttr2, which is a subattribute of an attribute with value NameAttr1.

### 4.1.1 Key Attributes

All attributes that are essential for establishing overlay socket communication in an overlay network should be selected to be key attributes. Key attributes are specific to an overlay network with a given overlay ID. Key attributes are selected when the overlay ID is created for an overlay network. Key attributes cannot be modified after the overlay ID has been created. All overlay sockets that participate in the same overlay network must have identical key attribute values. If two overlay sockets have the same set of key attributes and have the same values assigned to all of their key attributes, then they should be able to communicate in an overlay network.

The set of key attributes is not fixed. The creator of an overlay ID is responsible for deciding which attributes will be key attributes and which will not. This is done by specifying the value of the key attribute "KeyAttributes," which contains a list of all key attributes.

The attributes in Table 4.1 are expected to be key attributes in most cases.

Attribute Name:	Valid Values:	Explanation:	Default Value:
KeyAttributes	List of key attributes, separated by commas	The list of key attributes. The attribute KeyAttributes is always a key attribute and does not need to be contained in the list.	Socket, Node, SocketAdapter
Socket	HCAST2-0	Version number of the overlay socket.	2-0
SocketAdapter	ScktAdptTCP ScktAdptUDP	Indicates the protocol used by socket adapter.	TCP2-0
Node	HC2-0 (hypercube) DT2-0 (Delaunay triangulation)	Acronym of the overlay protocol with version number.	DT2-0
NodeAdapter	NodeAdptUDPS NodeAdptUDPM multicast	Indicates the protocol used by socket adapter.	no default
NodeAdptUDPMulticast.UDPMulticastAddress	Multicast IP address/port number	IP multicast address used by NodeAdptUDPMulticast	224.228.19.78/9472

Table 4.1. Key attributes.

Possible key attributes (which are currently not implemented) include a certification authority, which assures the uniqueness of an overlay ID.

OverlayID can be used to obtain the key attributes of an overlay. The overlay requires the overlay ID and KeyAttribute, separated by commas. If the list of key attributes is empty, then there are no other key attributes associated with the overlay ID. Since different overlay sockets generally will not be able to communicate unless they run the same version of the overlay socket, the same overlay protocol, and the same socket adapter, all essential attributes should be defined as key attributes.

Subattributes of key attributes can be key attributes. Subattributes of configurable attributes should not be key attributes.

#### 4.1.2 Configurable Attributes

All attributes which are not key attributes are called configurable attributes. Configurable attributes specify parameters of an overlay socket, which are not considered to be essential for establishing communication between different overlay sockets in an overlay network. However, having different values for the configurable attributes of overlay sockets may have a significant impact on the performance of the overlay network. For example, two overlay sockets of the same overlay network which run the hypercube protocol (Node="HC2-0") may not fix the attribute for a timeout value "HC2-0.Timeout" and may select this attribute to be configurable. However, different timeout values for the overlay protocol of overlay sockets in the same overlay network may lead to instabilities of the overlay protocol.

Even though it may be useful to be able to change the values of a configurable attribute after an overlay socket is created, the HyperCast 2.0 implementation does not permit changes to attributes after an overlay socket has been created.

Configurable attributes may specify subattributes of a key attribute. A listing of the format and the values of all attributes are given in a separate document.

#### 4.1.3 Creating Overlay Networks

The creation of an overlay network ties an overlay ID to a set of key attributes. Since all overlay sockets that join an overlay group require the same key attributes, the attributes and their values must be made available to the applications that start the overlay sockets. Applications can learn about the attributes of an overlay network in various ways. For example, key attributes can be assumed to be "well-known," can be communicated by email, or can be looked up at a server.

In the HyperCast software, the attributes of an overlay ID are handled by a component called the overlay manager, which initializes an overlay socket. The overlay manager is external to an overlay socket. The overlay manager creates an object, called configuration object, which contains the attributes of the overlay socket. Then, it passes this object to the procedure which creates and initializes an overlay socket.

The overlay manager reads attributes from a configuration file, with the default name "hypercast.prop." The HyperCast software also provides a HTTP server, called the overlay server, which can be used by the overlay manager to retrieve attributes for overlay networks. The overlay server, whose function is described in the next section, can be used to create overlay IDs and to store attributes of overlay networks.

If the "hypercast.prop" file for a certain overlay ID contains the attribute `OverlayServer = HTTP`, then the overlay manager assumes that there exists an overlay server. The location of the overlay server is a URL specified in the subattribute `HTTP.HttpServer0`. The values read from the overlay server override the values in the file "hypercast.prop."

Figure 4.2 describes the process to create the overlay network.

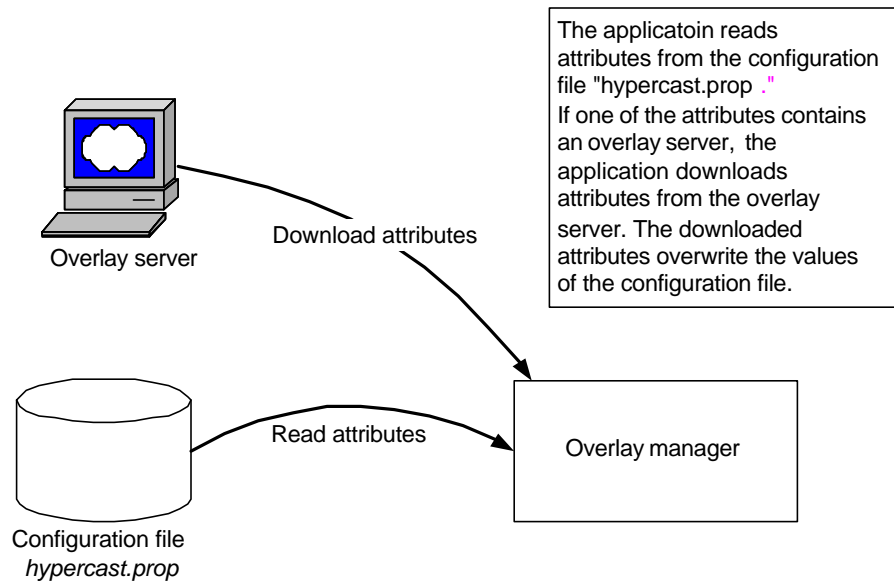


Figure 4.2. The overlay manager reads configuration parameters for an overlay socket from a configuration file and/or downloads configuration parameters from an overlay server. The configuration parameters are called attributes.

#### 4.4. OVERLAY SERVER

The overlay server is an http server which can

- create new overlay IDs and store associated attributes,
- test if an overlay ID already exists, and
- retrieve the attributes associate with a given overlay ID.

In this section, we describe the interactions of the overlay manager with the overlay server. We emphasize that all overlay sockets can be started without an overlay server.

Ideally, the overlay server should be implemented as a distributed service, similar to the domain name system of the Internet. However in HyperCast 2.0, multiple overlay servers do not communicate with each other.

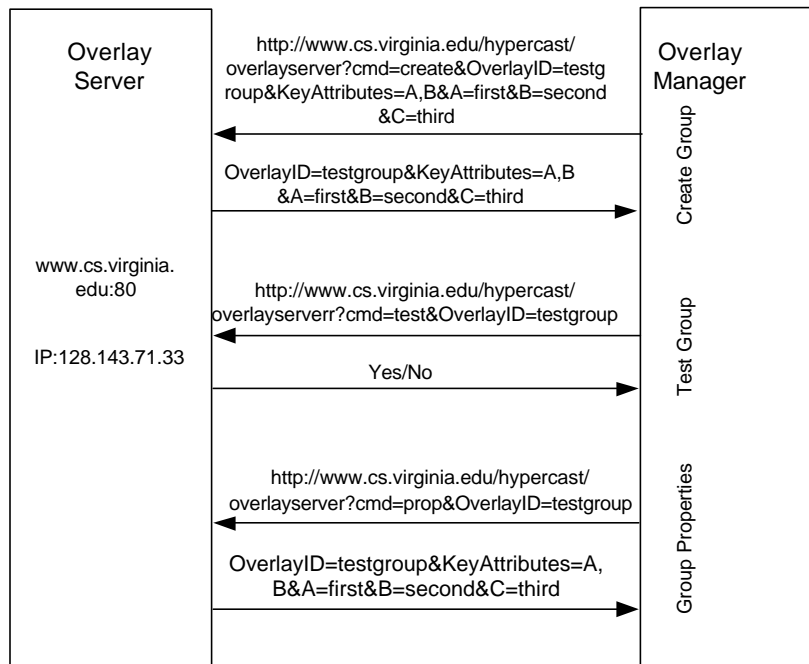
The interaction of the overlay manager with an overlay server is that of a web client with a web server, which executes a CGI script.

The properties of an overlay server are specified by a set of attributes:

Attribute Name:	Valid Values:	Explanation:
Overlay Server	<i>HTTP</i> or " "	Determines if an overlay server is used.
HTTP.HttpServer0	<URL of a server>	URL of primary overlay server.
HTTP.HttpServer1	<URL of server>	URL of secondary overlay server.

All interactions with the overlay server are implemented as "GET"-style CGI queries. "GET"-style CGI queries add name/value pairs as part of the URL; the URL of a query is the URL of the script followed by a '?' and the parameters which are entered as name/value pairs. A name/value pair is written as "Name=value." Multiple name/value pairs are separated by an ampersand ('&').

The first parameter of a CGI query to the overlay server is a parameter “cmd,” which indicates the command to be performed. There are three commands: create, test, and prop.



**Figure 4.3.** Interactions of overlay manager and overlay server.

#### 4.1.4 Creating an overlay network

A query to the overlay server with the command “cmd = create” asks the overlay server to create a new overlay network. As part of this query, the overlay ID and attributes are transmitted as parameters. If the query does not supply an overlay ID, then the overlay server creates an overlay ID. The query must supply all key attributes and may contain configurable attributes.

##### Example:

- `http://www.cs.virginia.edu/hypercast/overlayserver?cmd=create&OverlayID=testgroup&KeyAttributes=A,B&A=first&B=second&C=third`

The URL of the overlay server is to the left of the “?”. This query asks the overlay server to create a group with overlay ID “testgroup,” key attributes ‘A’ and ‘B’, where A=“foo” and B=“bar,” and configurable attribute C=“third.”

- `http://www.cs.virginia.edu/hypercast/overlayserver?cmd=create&OverlayID=&KeyAttributes=A,B&A=first&B=second&C=third`

This query provides an empty string as overlay ID. Here, the overlay server creates the overlay ID.

The overlay server responds to a CGI-query with an HTML document. If the creation of the overlay network was successful, the server replies with an HTML document that contains the overlay ID and the attributes that were set by the overlay server. If the

creation was not successful, the server replies with an HTML document which begins with the string "ERROR," followed by a message which describes the error.

#### 4.1.5 Testing for existence of overlay networks

A query to the overlay server with the command "cmd = test" is an attempt to verify if a specific overlay ID has been created. The overlay server replies to a test query with "YES" or "NO," where "YES" indicates that the group exists, and "NO" indicates that the group does not exist at the overlay server.

##### Example:

- `http://www.cs.virginia.edu/hypercast/overlayserver?cmd=test&OverlayID=testgroup`

This query asks if an overlay network with overlay ID "testgroup" has been created.

#### 4.1.6 Retrieving attributes for an overlay network

The command "cmd=prop" requests group attributes for an overlay ID from the overlay server. The reply contains a series of name/value pairs encoded as *name=value*, separated by ampersands ('&'s). The response contains all attributes stored at the server. If the overlay ID is not known at the overlay server, the server replies with an HTML document which begins with the string "ERROR," followed by a message which describes the error.

##### Example:

The query:

```
http://www.cs.virginia.edu/hypercast/overlayserver?cmd=prop&OverlayID=testgroup
```

could result in the response:

```
OverlayID=&KeyAttributes=A,B&A=first&B=second&C=third
```