Bionic Buffalo Tech Note #103

Introduction to Pegasus Client Support

last revised Tuesday 2003.04.08

Overview

The Time-Warner Pegasus SSP and LSC protocols are similar to some of the DSM-CC protocols. The similarity is sufficient to allow a single API to support both standards on the client. This Tech Note discusses extensions to the DSM-CC client API, to provide Pegasus support, and discusses how they are used by client applications.

Pegasus and DSM-CC Specifications

The DSM-CC specification (*ISO/IEC 13818-6*) defines four distinct protocols: User-User (U-U), User-Network (U-N), Download, and Switched Digital Broadcast Channel Change Protocol (SDB CCP). It also defines extensions to the MPEG protocols. Together, these protocols support provision, delivery, and use of various services, such as video-on-demand, broadcasting, file transfer, and internet access.

Time-Warner, as part of its Pegasus program, also defines several protocols which have some of these functions. The functionality of these Pegasus protocols is almost completely a strict subset of the functionality of the DSM-CC protocols. In particular,

> the Pegasus Session Setup Protocol (SSP) is based on the DSM-CC U-N Session protocol
>
> the Pegasus Lightweight Stream Control Protocol (LSCP) is different from the protocol used to support the DSM-CC U-U Stream class, but uses compatible semantics

There are specifications for SSP and LSCP. Please refer to the bibliography at the end of this Tech Note.

The various parts of DSM-CC are defined using different languages.

> The U-N Session Protocol, upon which SSP is based, is defined in terms of the Specification and Description Language (SDL), along with textual explanations, so that the interaction of client with server is described. However, no API is defined for clients or

servers implementing U-N.

The U-U Protocol is defined using the OMG Interface Definition Language (IDL), along with textual explanations. OMG provides formal mappings from IDL to protocols as well as to APIs, so there is an API for the U-U protocol, in addition to a well-defined over-the-wire behaviour. (The API is defined for a variety of programming languages, but this document, along with most client implementations and the examples in the DSM-CC specification, uses C.)

The CORBA specifications map IDL to the General Inter-ORB Protocol (GIOP). The mapping of GIOP to TCP/IP is called the Internet Inter-ORB Protocol (IIOP). In addition to the IIOP derived from the IDL mapping, DSM-CC allows other protocols, including Sun's ONC (Open Network Computing) RPC, to be used for remote object interaction, but IIOP is by far the most common.

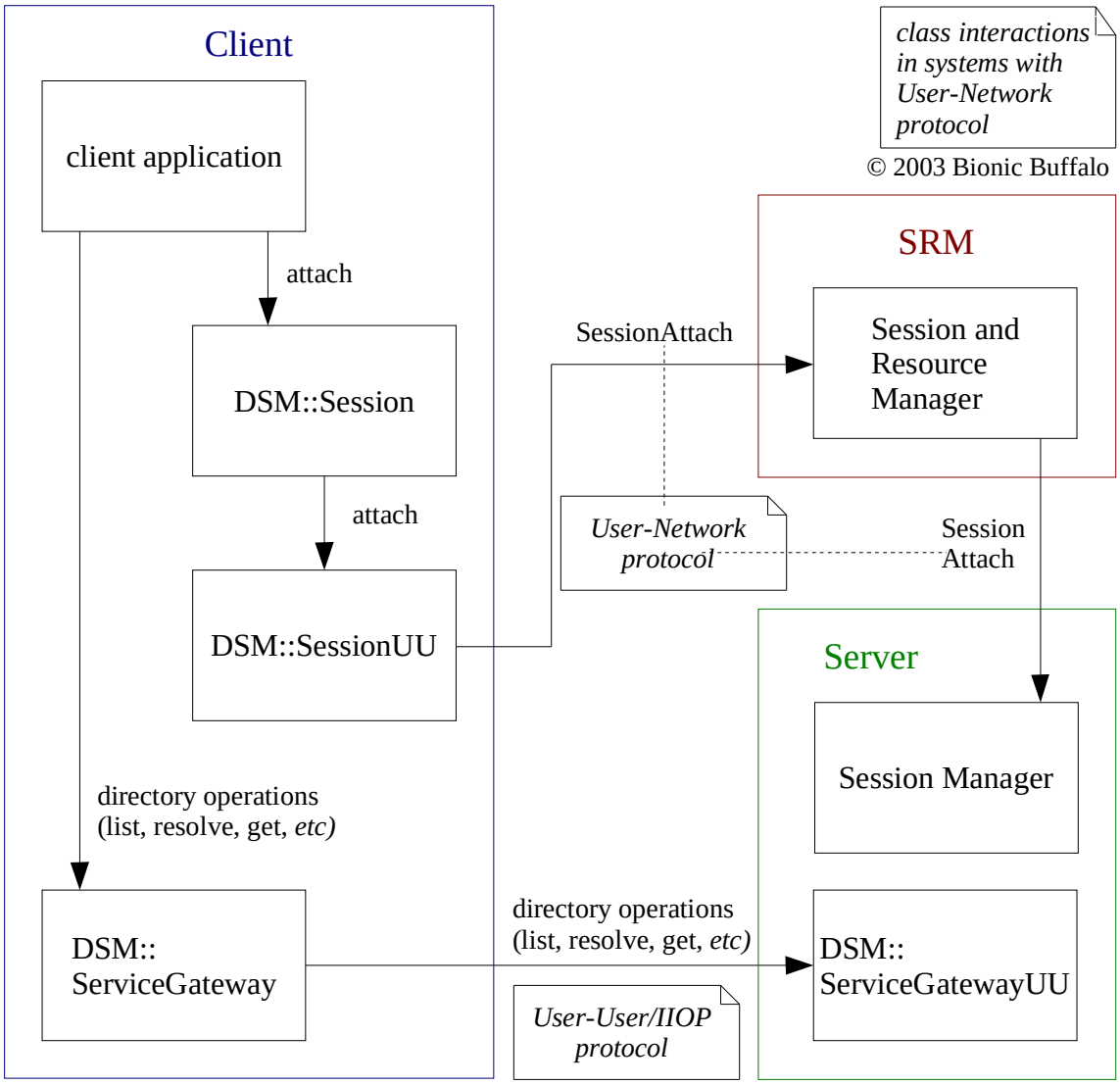
In the ensuing discussion, use is made of three terms: *server*, *client*, and *SRM*. We leave “server” and “client” with the common imprecise but adequate definitions. An SRM, or Session and Resource Manager, is a third type of entity which manages the network and its resources. The closest common analogue is the switching office in a telephone company: the client phone uses the SRM (switching office) to establish a connection, then carries on a conversation with the server. Although that might in some instances be an oversimplification, it should be an adequate model for this paper. Note that the SRM may be colocated with the server, or it may be in a separate machine or at some different location.

Contrasting DSM-CC and Pegasus ISA

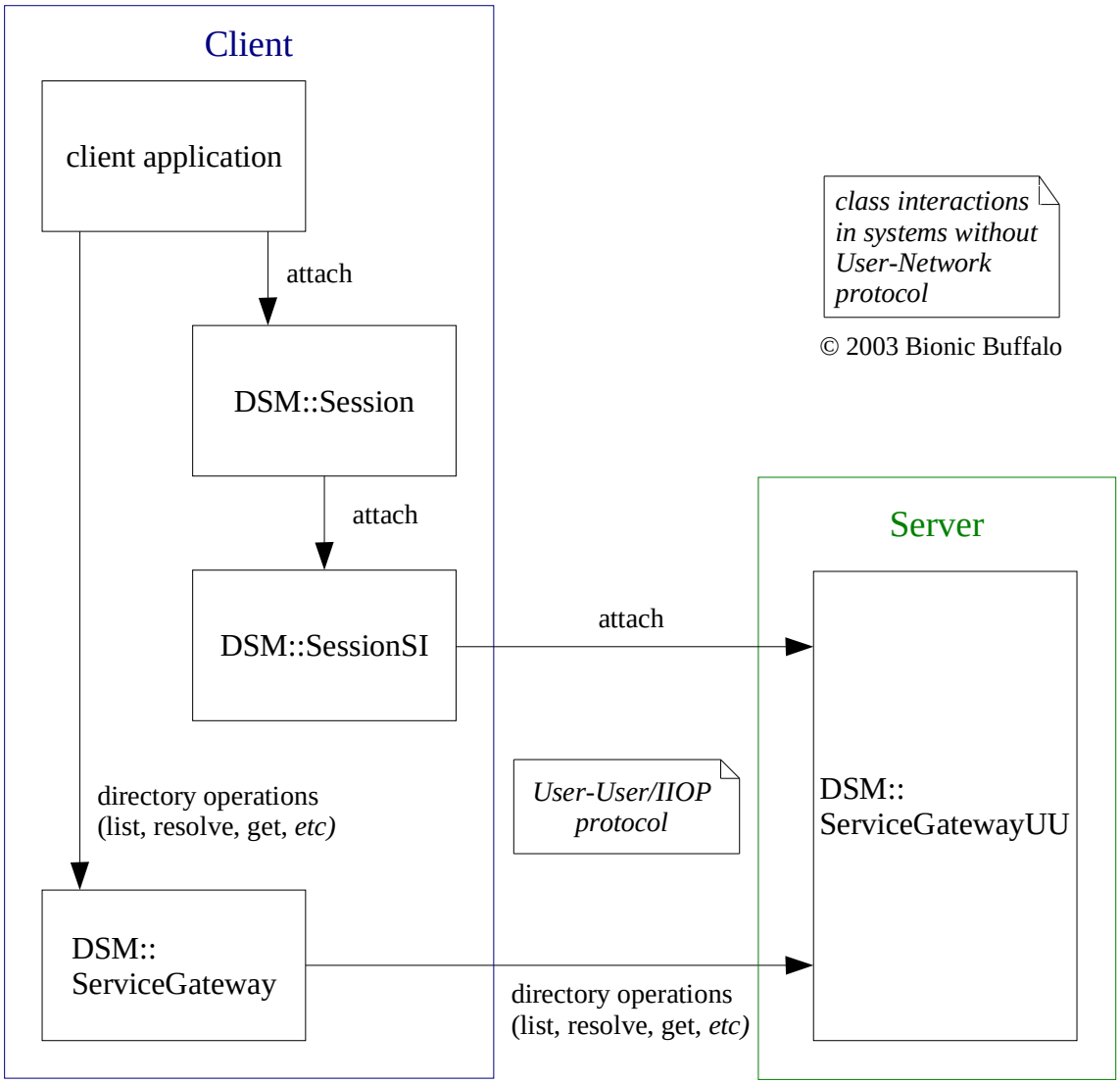
In addition to the LSCP and SSP protocols and specifications mentioned previously, there is a third Pegasus specification: *Pegasus Interactive Services Architecture*, whose subject matter is commonly called “ISA”. Although ISA claims some measure of DSM-CC compliance, in some ways it directly contradicts the DSM-CC standard.

In DSM-CC, there is a local `DSM::Session` object on the client. The client application uses the local `DSM::Session` object to connect to the server, returning a reference to a local `DSM::ServiceGateway` object. The `DSM::Session` object can do this in one of two ways:

In systems with U-N signalling, the `DSM::Session` object connects with a local `DSM::SessionUU` object, which performs the U-N signalling. The `DSM::SessionUU` object communicates with the SRM, using the U-N protocol, to set up a session between the local `DSM::ServiceGateway` object and a remote `DSM::ServiceGatewayUU` object on the server. (In the process of setting up the session, the SRM negotiates with the server's session manager, from which it obtains an object reference for the `DSM::ServiceGatewayUU` object. That object reference is relayed back to the client, giving the client access to the `DSM::ServiceGatewayUU` object.)

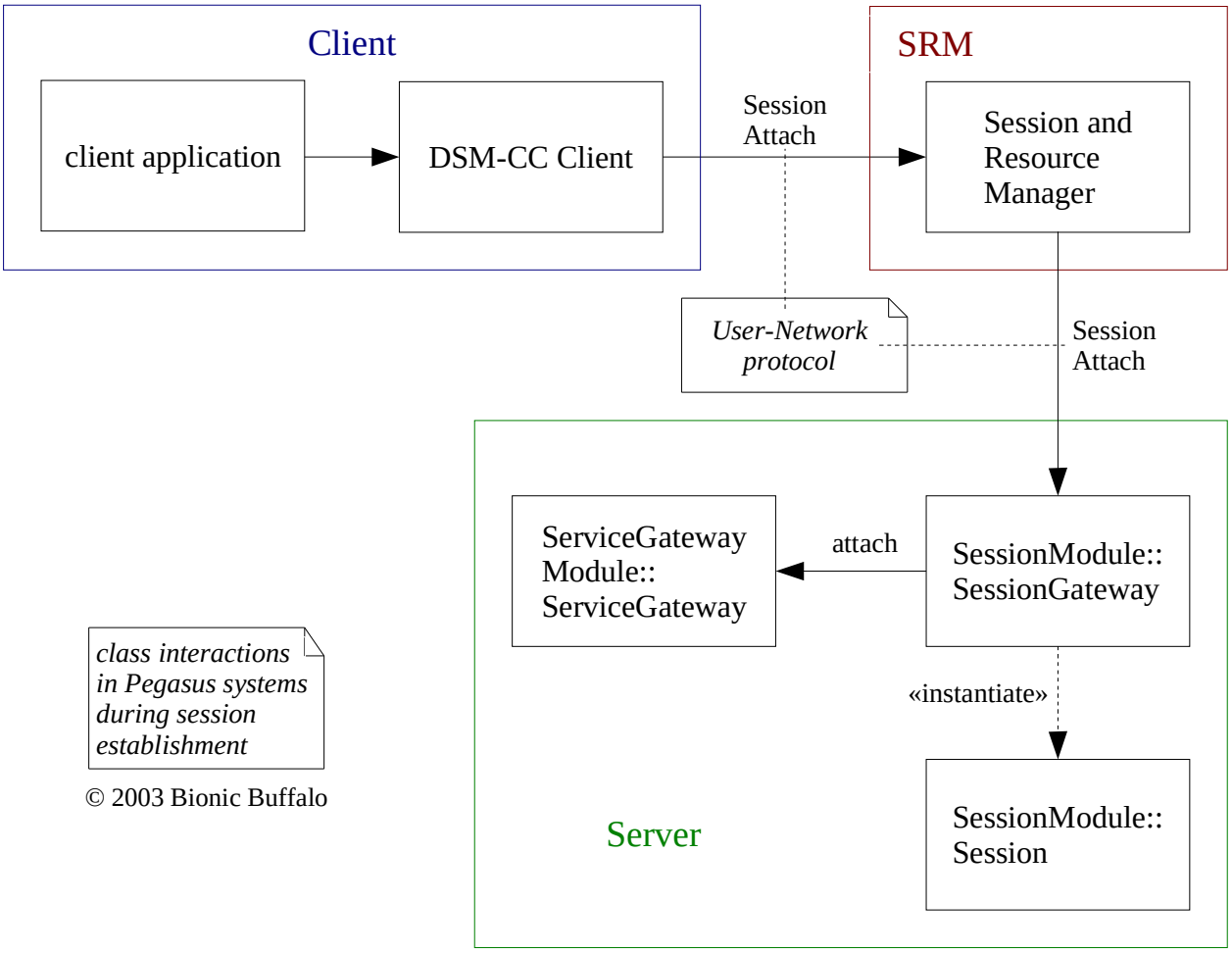


In systems without U-N signalling, the `DSM::Session` object connects with a local `DSM::SessionSI` object. The local `DSM::SessionSI` sets up a session between the local `DSM::ServiceGateway` object and a remote `DSM::ServiceGatewaySI` object on the server. The server's `DSM::ServiceGatewaySI` object performs some of the same functions as the SRM would in the U-N case.



In contrast, in the Pegasus model, *all* of the objects are on the server. The Pegasus ISA defines three main classes relating to sessions and connection: `SessionModule::Session`, `SessionModule::SessionGateway`, and `ServiceGatewayModule::ServiceGateway` (which inherits `DSM::SessionUU`). It also defines the DSM-CC Client, a component on the client system. In the Pegasus model, to establish a session with a server, the client application calls an uspecified API of the DSM-CC Client component. The DSM-CC Client component sends the U-N SessionSetup message to the SRM. The SRM negotiates with the server's `SessionModule::SessionGateway` object. The `SessionModule::SessionGateway` object creates a `SessionModule::Session` object, which represents the session. The `SessionModule::SessionGateway` attaches to a `ServiceGatewayModule::ServiceGateway` object on behalf of the client and the session. Unlike a DSM-CC ServiceGateway, a Pegasus ServiceGateway doesn't allow the client to browse the server for content. Indeed, it inherits `DSM::SessionUU`, but not any directory operations, so it is merely vestigial with respect to its DSM-CC counterpart.

Moreover, the client never gets object references, and there is no protocol for invoking methods directly by the client on a Pegasus ServiceGateway.



The other class with some similarity between the two architectures is the DSM::Stream object, which has the Pegasus stream object. The similarity is largely superficial. The DSM-CC concept of service objects, such as `DSM::Stream`, is radically different from the Pegasus concept of service objects. In DSM-CC, clients share use of single objects, while in Pegasus, there is an object for each client using a given service. For example, if a DSM-CC video server were to host the movie *Casablanca*, there would be a single *Casablanca* `DSM::Stream` object to be shared among all clients. A Pegasus video server, on the other hand, would instantiate a separate *Casablanca* stream object for each client viewing that movie.

A DSM-CC client gains access to service objects (such as streams and files) by first establishing a session, in a process which returns an object reference to a `DSM::ServiceGateway` object. The `DSM::ServiceGateway` object has directory operations, allowing the client to perform operations that browse the server's contents. When it finds an object the viewer likes, such as a *Casablanca* stream object, the client system merely issues a method invocation (remote procedure call) on the object's interface. For example, the client might invoke `DSM::Stream::play` (mapped to C as

`DSM_Stream_play()`). It is up to the servant to keep separate the state of each client's interaction. When the session ends, the object might issue a `close` operation to clean up object references and other resources, but the object itself doesn't go away. (There is a separate operation, `destroy`, to permanently destroy the object, and only the object's owner - not a mere, ordinary client - has sufficient privilege to do that.)

When a Pegasus client establishes a session, the client provides at session establishment time the name of a service. During the process, the `ServiceGatewayModule::ServiceGateway` object will issue an `attach` operation on the specified service object factory, which will instantiate a copy of the service object. The client doesn't talk to a pre-existing object, but rather has one created for it on the fly.

Implementation Models

DSM-CC User-User. The CORBA model implementing the U-U IDL normally would use an object request broker (ORB) to mediate interactions between applications and service objects:

application - ORB - object

or

application - local ORB - IIOP - remote ORB - object

Where they are needed (when the object is remote), the ORBs implement IIOP, and the location of the object is transparent to the application. In the case of U-U, the DSM-CC specification acknowledges that a complete, distributed object system, with attendant ORB, isn't practical in all implementations. Therefore, DSM-CC envisions local proxy objects as an alternative:

application - local proxy - IIOP - remote ORB - object

With local proxy objects, the over-the-wire IIOP protocol is implemented directly by the proxy, rather than by a local ORB. The proxy objects have the same interfaces (APIs) as the corresponding remote objects, so location transparency is preserved.

One class of special interest in this paper is the `DSM::Stream` class. An object of this class represents streaming audio, video, and data, with a time base and the capabilities of jumping directly to specified points, pausing, changing playback speed (fast or slow play), and other such functions. The client can control these operations, making the object a kind of virtual VCR or audio recorder with direct-access capability. Each `DSM::Stream` object contains a logical state machine, the stream state machine. Clients operate this virtual playback engine by invoking operations defined on the DSM:: Stream interface. The `DSM::Stream` object, with its state machine, exists on the server, and is operated remotely. Interaction is based on the same IIOP which is used with other classes.

application - local ORB - IIOP - remote ORB - DSM::Stream object

or

application - local proxy - IIOP - remote ORB - DSM::Stream object

In other words, the client application manipulates remote stream objects on the server.

DSM-CC User-Network. Bionic Buffalo implemented U-N by first defining the U-N services interfaces in IDL. In a conventional CORBA system, an application communicates with local and remote service objects by way of the ORB. In Bionic Buffalo's U-N system, the application communicates with service objects which implement the U-N protocols. The client normally doesn't see U-N server objects directly, or *vice versa*. Instead, the remote objects are seen only through the services of local objects. Thus, Bionic Buffalo's implementation of U-N is similar to that of U-U implementations which use proxy objects:

application - local U-N object - (U-N protocol) - remote U-N server

(Note: Normally, the application talks to local objects, but because the objects are implemented using CORBA it is also possible with Bionic Buffalo's product to communicate directly with remote objects. In other words, it is possible to have *application - local ORB - remote ORB - remote U-N object - U-N protocol - differently-remote U-N server*. This feature isn't expected to find normal use, but rather value in support and development situations.)

Pegaus Session Setup Protocol. The Pegasus specifications (at least, the SSP and LSCP ones considered here) define the over-the-wire interaction, but don't define an API for applications. They define protocols, not objects. To implement SSP and LSCP, Bionic Buffalo uses local and proxy objects.

For SSP, the Pegasus protocol is in fact a dialect of the DSM-CC U-N Session protocol. The model is the same as used for basic U-N:

application - local object - U-N protocol (SSP dialect) - U-N server

The local objects have been modified slightly from standard DSM-CC objects, to allow for the SSP dialect. (A modification wasn't, strictly speaking, necessary. However, it makes the API easier to use.) The same objects, with the same API, can be used for basic U-N, as for the SSP dialect.

Pegasus Lightweight Stream Control Protocol. As with the `DSM::Stream` class, LSCP envisions a playback engine on the server. Instead of regarding it as an object, however, the LSCP specification defines a client-server protocol. The playback engine embodies the same state machine as does a `DSM::Stream` object, and the messages controlling the playback operation have the same semantics as those of the corresponding IIOP messages. (However, the Pegasus playback engine lacks some operations not directly related to playback and the state machine, such as those for access control, security, and event management. Where Pegasus has equivalent functionality, these are handled in different ways.) The data formats, such as the one used for time and the time base, are a little different between Pegasus and DSM-CC. Altogether, the differences are slight enough that the API representing the `DSM::Stream` interface can be used to control a Pegasus playback engine.

Usually, modification of a Pegasus server is infeasible, so Bionic Buffalo's software uses a client proxy object to implement LSCP. The model is

application - local object - LSCP - Pegasus server - playback engine

In this implementation, the local object has the same interface as a DSM::Stream object (or, when used, a `DSM::Stream` proxy). Thus, the application doesn't see the difference between a Pegasus stream service and a DSM-CC stream object: one application can handle both.

Server Implementations. Bionic Buffalo's server implementations of DSM-CC and Pegasus are beyond the scope of this Tech Note, and are discussed elsewhere.

The Application Boot Process with Pegasus

Although DSM-CC offers various protocols (U-N, U-U, Download, SDB-CCP), the design allows the application itself to avoid working with them directly. Instead, an application finds a local object of the class `DSM::Session`, and uses that object's interfaces to find services and other resources on the network. There is no need for an application to be concerned for the protocols and operations which underpin the system. Pegasus, on the other hand, does not have the U-U abstraction layer with its interfaces to insulate the application in this way.

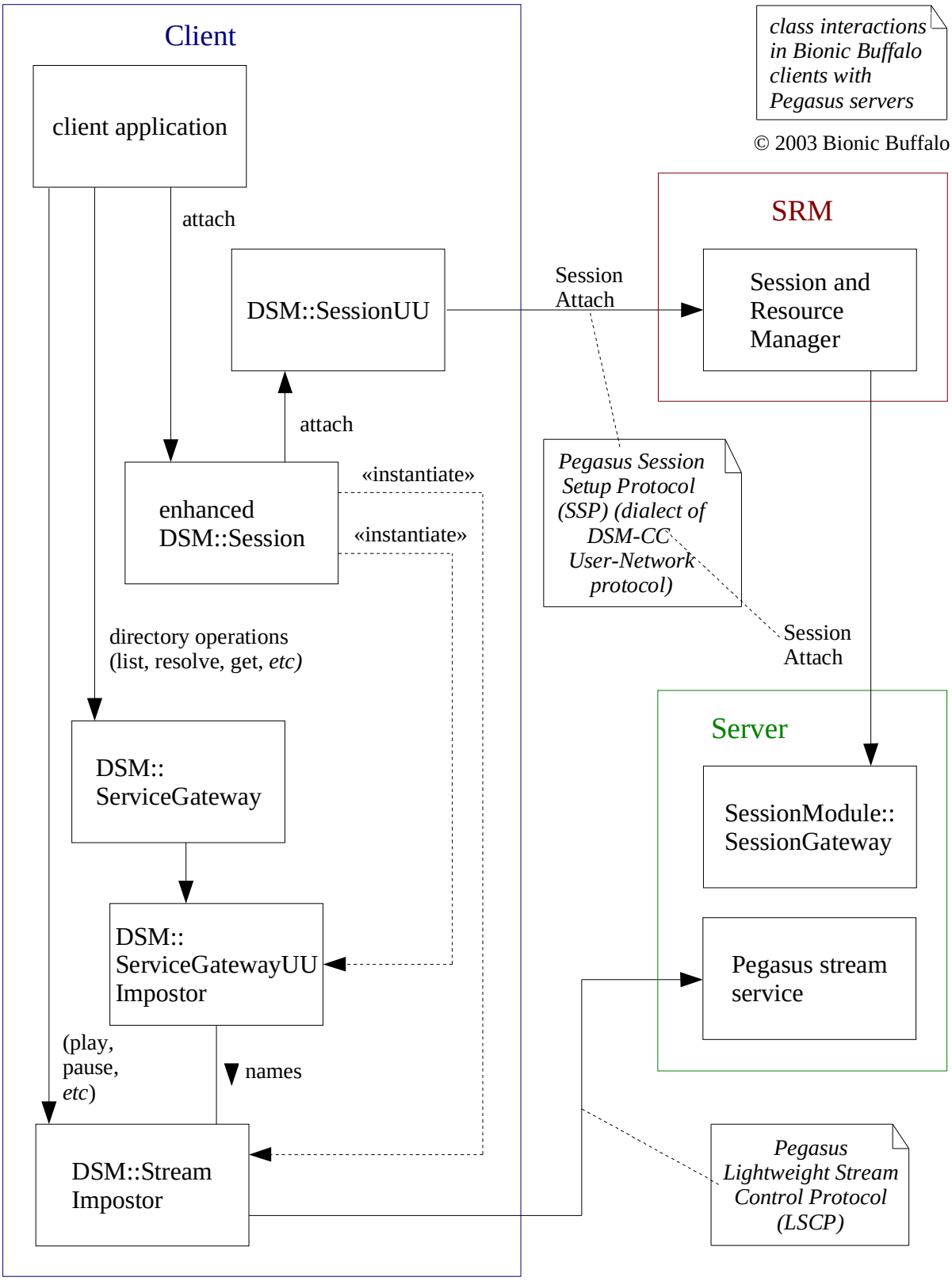
Bionic Buffalo clients have extended the functionality of the `DSM::Session` object, so it can interact with a Pegasus server, while the client sees the same API as used for ISO/IEC DSM-CC protocols.

The `DSM::Session` interface has two operations, `attach` and `detach`. The `attach` operation attaches or connects the client with a service domain (an object of class `DSM::ServiceGateway`), while `detach` releases the connection. A `DSM::ServiceGateway` is usually a remote object, and it has operations similar to those of a directory. The directory operations allow the client to browse for services in the same way one might browse the internet or a conventional disk filesystem. The objects in the directory may be of any kind (including kinds defined by the implementor), but three are especially important: `DSM::Stream` (streaming video, audio, and data), `DSM::File` (conventional data files), and `DSM::Directory` (tree-structured directories).

To allow transparent use of the usual DSM-CC interfaces in Pegasus environments, Bionic Buffalo has implemented adapter servants. These adapter servants are for classses that look like `ServiceGatewayUU` and `DSM::Stream` interfaces, but which in fact implement the Pegasus SSP and LSCP. They are impostors.

In Bionic Buffalo's DSM-CC, there is a configuration step before the Session object is used, when the application tells the Session object which U-N session object to use for session establishment. This is required because DSM-CC allows a client to talk to more than one SRM at a time, and there must be a way to specify which SRM is to be used. Each U-N session object is associated with a single SRM. If the application configures a U-N session object for the Pegasus U-N dialect, then the associated U-U Session object will know it will be connecting to a Pegasus server.

When a client application issues an attach request to the local Session object, and the Session object determines that the request is for a Pegasus server, then the Session object negotiates for a session with the SRM using a modified version of the `DSM::SessionUU` object. The modified SessionUU object can handle the special Pegasus dialect of the U-N protocol. Once the session has been negotiated, however, the local Session object takes a different step: it instantiates two new objects, which we will call PegasusServiceGateway and PegasusStream. The PegasusServiceGateway is a local impostor, pretending to be a remote ServiceGatewayUU object. The PegasusServiceGateway has one object in its directory: the PegasusStream object. The PegasusStream object is another local impostor, pretending to be a remote `DSM::Stream` object. The client application sees as ServiceGateway and Stream object, as it normally would in a DSM-CC environment, but those objects aren't really on the server. They are local, but the application cannot tell the difference. (CORBA makes little distinction between local and remote objects, from the client application's point of view.)


Client
client application
attach
DSM::SessionUU
attach
enhanced
DSM::Session
«instantiate»
«instantiate»
directory operations
(list, resolve, get, *etc)*
DSM::
ServiceGateway
DSM::
ServiceGatewayUU
Impostor
(play,
pause,
etc)
names
DSM::Stream
Impostor
class interactions in Bionic Buffalo clients with Pegasus servers
© 2003 Bionic Buffalo
SRM
Session and
Resource
Manager
Session
Attach
Pegasus Session Setup Protocol (SSP) (dialect of DSM-CC User-Network protocol)
Session
Attach
Server
SessionModule::
SessionGateway
Pegasus stream
service
Pegasus Lightweight Stream Control Protocol (LSCP)

Bibliography

The DSM-CC specification is owned by ISO/IEC (*http://www.iso.org*). It is an integral part of the MPEG-2 specification. The full name of the DSM-CC part is:

> *Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for Digital Storage Media Command and Control (DSM-CC)*

The reference number is ISO/IEC 13818-6. ISO/IEC specifications aren't free: they are available for purchase from national standards organizations and private publishers. (If you are in the U.S., you might try Global Engineering Documents, *http://global.ihs.com*.)

Time-Warner owns the Pegasus specifications. They are available on their web site, at *http://twcable.web.aol.com/Pegasus/pegasusArchive.htm*. The two client protocols and the server architecture are described in these documents:

> *Session Setup Protocol*, Version 2.1, 2000.11.03 (file name SSP21.pdf)
>
> *Lightweight Stream Control Protocol*, Version 1.0, 1999.06.10 (file name LSC10.pdf)
>
> *Pegasus Interactive Services Architecture*, Version 1.3, 2001.07.20 (file name ISA130.pdf)

For other DSM-CC documentation, see the Bionic Buffalo web site, at *http://www.tatanka.com/prod/info/dsmcc.html*.
