Iraqi Journal of Statistical Science (9) 2006
P.P. [75-90]

Dealing with DirectPlay Environments under DirectX

Aseel Waleed A. Al-Niamey*

ABSTRACT

The idea of this project is based on programming the functions of DirectX, which will connect the last one with the windows, this work is needed because DirectX deals with the hardware and the buffers of the memory while the windows does not directly deal with it.

The work focuses on programming the DirectPlay function, which is considered as an interface programming to guarantee the arrival of the applications to the communication services.

DirectPlay supports the facilities of connecting the game with the internet directly by the Modem Link or indirectly by the Ethernet. Actually the game becomes more flexible and enjoying if it is played against a real player (or another user of the computer) instead of playing with the computer itself. Also DirectPlay supports a group of users connected with each other in an active way at the same time (for example, the Internet chatting or Ethernet chatting).

In this work, an Ethernet chatting is done by using a Client/Server applications to see the working of DirectPlay, then, we try to use an active and strong language to make this is possible, and therefor we use the Visual C++ language.

العمل ضمن بيئة برمجة الدالة تحت محرر DirectX

الملخص

إن فكرة البحث قائمة على أساس برمجة الدوال التي يوفرهـــا Microsoft DirectX والتي تربط الاخير مع نظام windows وذلك للتعامل مباشرة مع العتاد ومقاطع البيانات بخلاف نظام windows الذي لايتعامل معها مباشرة.

يرتكز البحث على برمجة الدالة Direct Play التي تعتبر واجهة برمجية تعمل علـــى وصول التطبيق إلى خدمات الاتصال. حيـــث وفـــر Direct Play إمكانيـــة ارتبـــاط الألعـــاب بالانترنيت بسهولة إما بصورة مباشرة عن طريق Modem Link أو غير مباشرة عن طريـــق شبكة محلية داخلية Ethernet، ومن المعروف أن الألعاب تكون اكثر فاعلية ومتعة إذا كانـــت

*Assistant Lecturer/Computer Science Dept. /College of Computer Sciences and Mathematics

Received: 18/ 5 /2005 _____Accepted: 6/ 2 / 2006

ضد لاعب حقيقي (مستخدم آخر للحاسبة) بدلاً من اللعب مع الحاسبة ذاتها. كما يوفر DirectPlay مجموعة من المستخدمين المرتبطين مع بعضهم بصورة متفاعلة في الوقت نفسه، كمثال على ذلك المحادثة التي تتم عن طريق الانترنيت (Internet chatting) أو المحادثة التي تتم بواسطة شبكة محلية داخلية (Ethernet chatting).

تم في هذا البحث عمل Ethernet chatting باستخدام التطبيق Server/Client لغرض مشاهدة التطبيق العملي للدالة DirectPlay وكان لابد من توظيف لغة برمجية كفوءة لتأدية مثل هذا العمل فكانت لغة Visual C++ هي الانسب.

1- Introduction

Multiplier gaming across computer networks is a relatively new idea, and the concept of being able to interact in a virtual world together with thousands of other players would have seemed like science fiction only a couple of years ago. Today, counterstrike is a competition sport much like any other, and multiplier games are common. Network programming is becoming nearly as central as graphics programming to the game programming process, and network efficiency is more and more important [2].

In general, the game needs a lot of computation power for higher speed graphics. then the window is a real problem in front of this work, because it is suffering from the low speed as a result of dealing with the buffers of the memory which is worked as a mirror for the locations of the data to be displayed instead of dealing with the data itself. For this reasons DirectX being the solution for this problem, because DirectX directly deal with the buffers that contain data. For the above reasons, it is necessary to develop programs that can work as an interface between DirectX (where Microsoft DirectX is a group of a low level application programs which help us to make the games and the Multi-Media applications with high level programming) and the windows.

The program can be defined as a Network Supervisor that can be used for session and gaming as well. The programmer wants to develop a server\client application. The server application is resided in the server computer and receives the information from the client of the LAN and manages the messages and information about users. The server program can save the time that each user spends using its space in the domain, this can be done by a database system. The server can prompt the user that spend more than one hour for warning him only. Another aspect of this Server\Client application is that each user can launch a session that is currently worked. The user can run the client application and see the list of all session, then choose one of them and enter it for conversation [7].

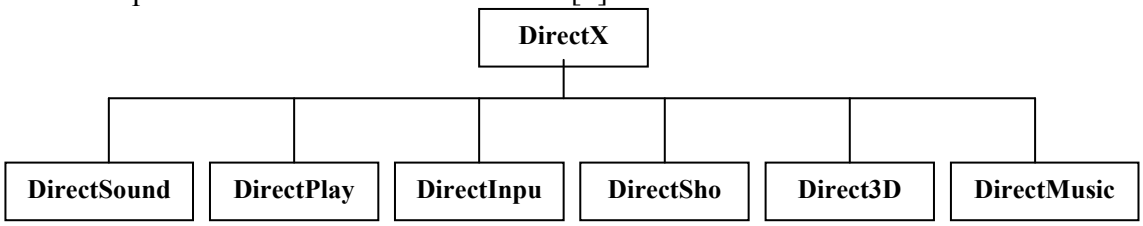
Both client and server programs are installed on the computer by a setup approach (setup.exe file), at client side the program is running in background of other applications and can not be realized and manipulated by user, but other utilities of it (session hosting or session participation, gaming, …etc) can be accessed directly from start menu of windows desktop [4].

1.1 An overview on DirectX:

Microsoft DirectX is a set of low-level Application Programming Interfaces (APIs) for creating games and other high-performance multimedia applications. It includes support for two-dimensional (2-D) and three-dimensional (3-D) graphics, sound effects and music, input devices, and networked applications such as multi-player games. Microsoft DirectX 9.0 (the newest version) is a major release primarily for graphics. It includes new tools, new features for graphics and Microsoft DirectShow, and enhancements for Microsoft DirectInput and Microsoft DirectPlay [3].

1.2 DirectX major components:

One of the main purposes of DirectX is to provide a standard way of accessing many different properties hardware devices. The major components of DirectX are as follows [6]:



1.2.1 DirectPlay:

DirectPlay is an API that simplifies application access to communication services as shown in figure (1). It provides a way for applications to communicate with each other independent of the underlying protocol, transport, or online service [7]. This is especially useful for games. Real players, each on his own personal computer, can be gathered and launched into a game, without the game developer worrying about the connections. Instead of forcing the developer to deal with the differences that each connectivity solution represents, DirectPlay provides well-defined, generalized communication capabilities. DirectPlay also shields developers from the underlying complexities of diverse connectivity implementations, freeing them to concentrate on producing great applications [6].

DirectPlay is Microsoft's own network library provided as a part of DirectX on the MS_Windows and XBox platforms. If a game is to be marketed solely for the PC or XBox markets, relying on the more advanced features of DirectPlay might be an option. Most games of today are primarily targeted for release on Play station-2, so for DirectPlay to be of

use, the programmer would have to duplicate much of its features on that platform manually [5].
The main use of DirectPlay in the context of XBox Live is as a game arbitrer, a central server keeping track of current games in progress and potential participants. It provides chat functionality for those looking for games, and services such as high score charts.

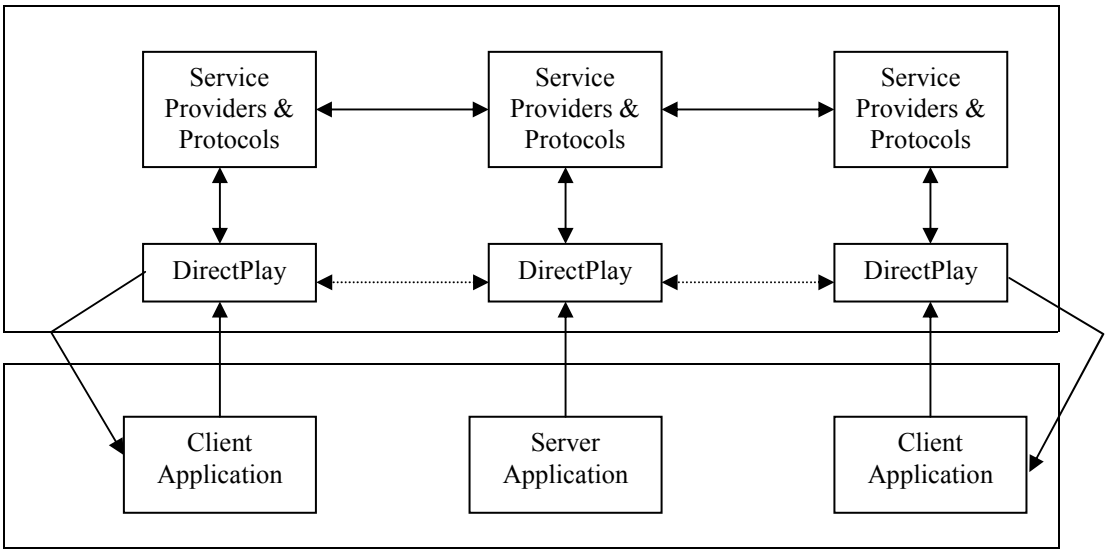


Figure1: General Definition of DirectPlay

1.2.2 Architecture of DirectPlay

The DirectPlay API is a network abstraction and distributed object system that applications can be written to. The API defines the functionality of the abstract DirectPlay network, and all the functionality is available to the user's application regardless of whether the actual underling network supports it. In cases where the underling network does not support a method, DirectPlay contains all the code necessary to emulate it. Example includes group messaging and guaranteed messaging [6].

DirectPlay's service-provider architecture insulates the application from the underlying network it is running on, as shown in figure (2). The application can query DirectPlay for specific capabilities on the underlying network, such as latency and bandwidth, and adjust its communication accordingly [7].
The first step in using DirectPlay is to select which service provider to use. The service provider determines what type of network or protocol will be used for communications. The protocol can range from TCP/IP over the Internet, to an IPX local area network, to a serial cable connection between two computers [5].

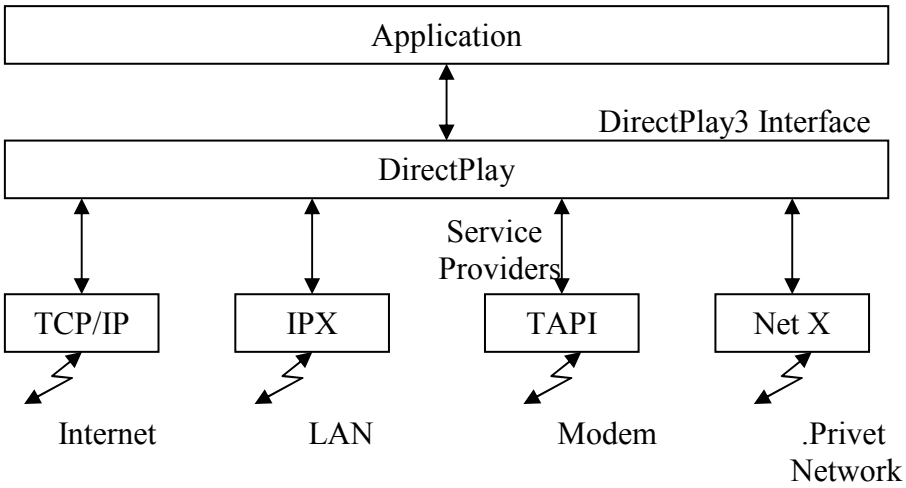


Fig.2: Direct Play service-provider architecture

1.2.3 Session Management

A DirectPlay session is a communications channel between several computers. Before an application can start communicating with other computers, it must be part of a session. An application can do this in one of two ways: It can enumerate all the existing sessions on a network and join one of them, or it can create a new session and wait for other computers to join it. Once the application is part of session, it can create a player and exchange messages with all the other players in the session [3].

Each session has one computer that is designated as the host. The host is the owner of the session and is the only computer that can change the session's properties as shown in figure (3).

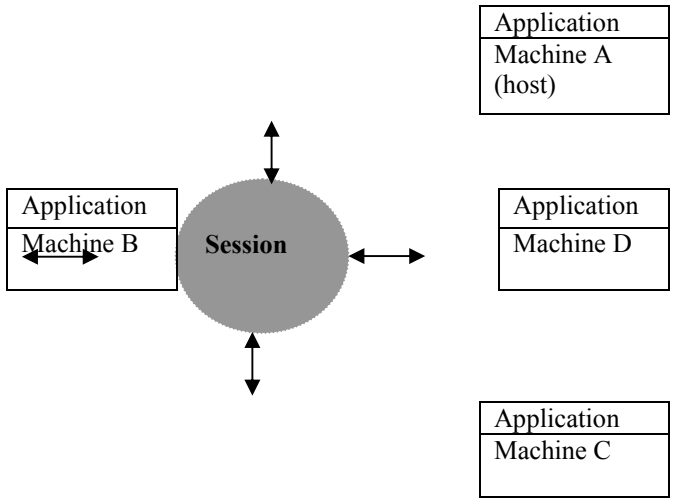


Figure 3: DirectPlay session model

2- DirectPlay Communications

DirectPlay's default mode of communications is peer-to-peer. In this model, the session complete state is replicated on all the computers in the session as shown in figure (4). This means that the session description data, the list of players and groups, and the names and remote data associated with each session are duplicated on every computer [4]. When one computer changes something, it is immediately propagated to all the other computers.

DirectPlay's alternative mode of communications is Client/server. In this model, only the server stores the session complete state, and each client has only a subset of the session's state as shown in figure (5). Each client has only the information that is relevant to that computer and receives that information from the server. When one computer changes something, it propagates the change to the server. The server then determines which client it must inform of the change [7].

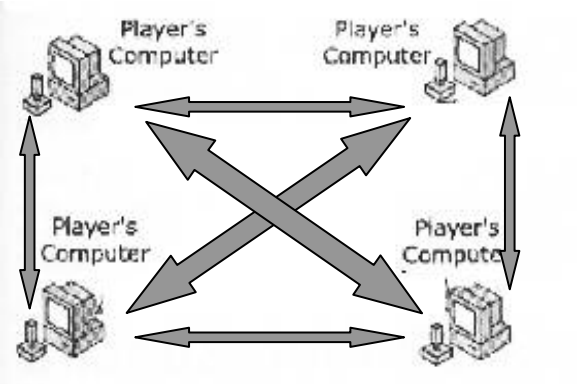


Figure 4: Peer to Peer model

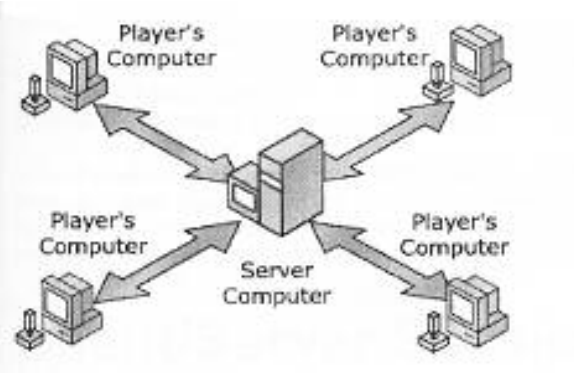


Figure 5: Client/Server model

3- Client/Server Session

The work is a Client/Server Chat Program. A Client/Server Session consists of a collection of players, or clients connected to a central server. As far as Microsoft DirectPlay is concerned, a client has no knowledge of any other client, only the server. The massaging needed to run the game is between the individual clients and the server. DirectPlay does not provide direct client-to-client massaging, as it does for peer-to-peer session's [5].

The designed client/server session requires two distinctly different applications [7]:

- The server application runs on a remote server. At a minimum, it serves as a central massaging hub and game host. The server must

receive and handle all incoming massages from the clients, and send appropriate massages back out. Any transfer of data from one client to another must be handled by the server application. The majority of the processing can be done on a separate computer (the server), therefore, there wasn't a need to rely on the power of the client's computer.

- The client application runs on each player's computer. The primary function of this application is to handle the user interface, and keep the player's game state synchronized with the server. A client\server session requires client to pass information to each other indirectly through the server. No automatic method exists for the server to pass information from one client to another, then the user should use the server to pass the information between the client's.

4- Design Steps:

1- At the beginning, the programmer should load the Microsoft DirectX version.9 to the computer.
2- Initiating the Client/Server session including:
   - The Server application.
   - The Client application.
3- Selecting a service provider for communication.
4- Selecting the Client/Server Host.
5- Connecting to the Client/Server Session.
6- Managing the Client/Server Session.
7- Handling the Client/Server messages.
8- Terminating the Client/Server Session.

4.1 The Server Application:

The server is responsible for updating clients about the Chat Program state , for example when player joins or leaves the session. For setting up the server, the programmer must do the following three tasks:

- Create a DirectPlay Server object
- Create the Server Address object
- Initiate Hosting

Client\Server games are often arranged through lobbies. The most straightforward way to launch the server is to implement it as a *lobbyable* application . This approach provides a way to launch the server, and supports communication between server and lobby during the course of the session. But the designing is through server application. The server directly launched, and then advertises itself as available and waits for clients to connect; first the programmer initialize the server side as shown in figure (4) and listed below:

- Create a DirectPlay Server object:

For creating a Microsoft DirectPlay Server object the programmer must pass the class identifier of server object (CLSID_DirectPlay8Server),

the identifier of the interface (IID_IdirectPlay8Server), and the address of pointer to the ***IDirectPlay8Server*** interface.

```
IDirectPlay8Server                    *g_pDPServer=NULL;
Hr=CoCreateInstance(CLSID_DirectPlay8Server, NULL,
                    CLSCTX_INPROC_SERVER,
                    IID_IDirectPlay8Server,
                    (LPVOID*)&g_pDPServer);
```

- **Initialize a DirectPlay Server object:**

After the programmer has created the DirectPlay Server object, he must initialize it by calling the ***IDirectPlay8Server::Initialize*** method. In the initialization, the programmer must pass the pointer to an Implementing a Callback Function in DirectPlay and DirectPlay Voice, *Direct Play Message Handler Server*, which handles messages received by the server.

```
Hr=g_pDPServer->Initialize(NULL, DirectPlayMessageHandlerServer,0)
```

4.2 The Client application:

The Client application is responsible for handling the user interface (UI) and processing messages from the server. Because the application is not lobby launched, the designed client application will receive neither the connection handle, nor the message. Therefore the programmer doesn't need to handle any lobbied messages. the programmer should only create and initialize a client object (CLSID_DirectPlay8Client). This object will be the primary means of communicating with DirectPlay and the sever as shown in figure (4).

4.3 Selecting a Service Provider for Communication:

The service provider is the network connection. Most games use either the Transmission Control Protocol/Internet Protocol (TCP/IP) or modem service provider, but Microsoft DirectPlay also provides support for serial and Internet work Packet Exchange (IPX) connections. They could determine which service provider to use, then the client object's ***IDirectPlay8Client::EnumSeviceProviders*** method is used to enumerate the available service providers. Then the programmer should have created the DirectPlay address object for the user (*a device address*). This address was used to identify the device with a number of DirectPlay methods as shown in figure (5).

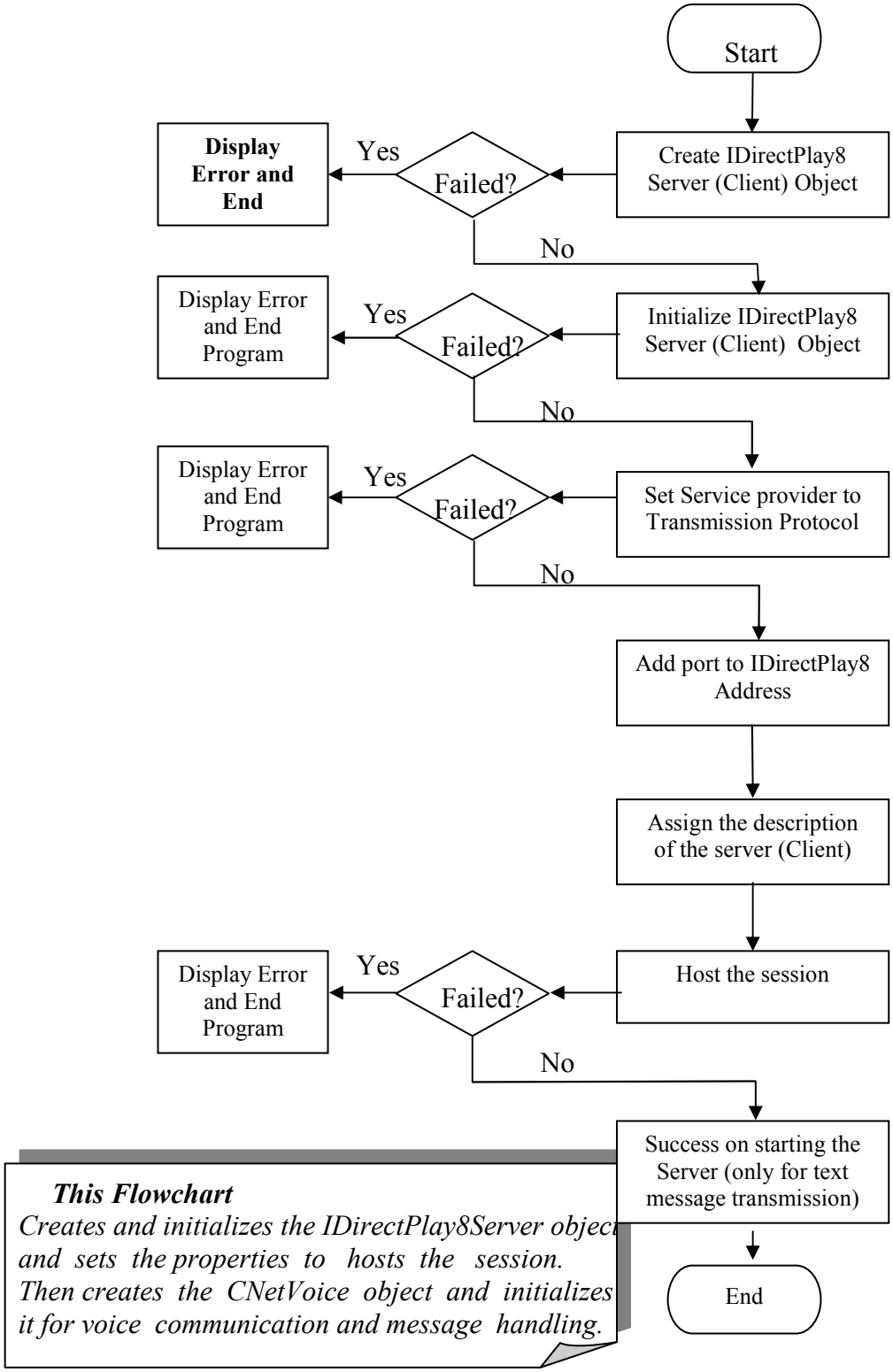
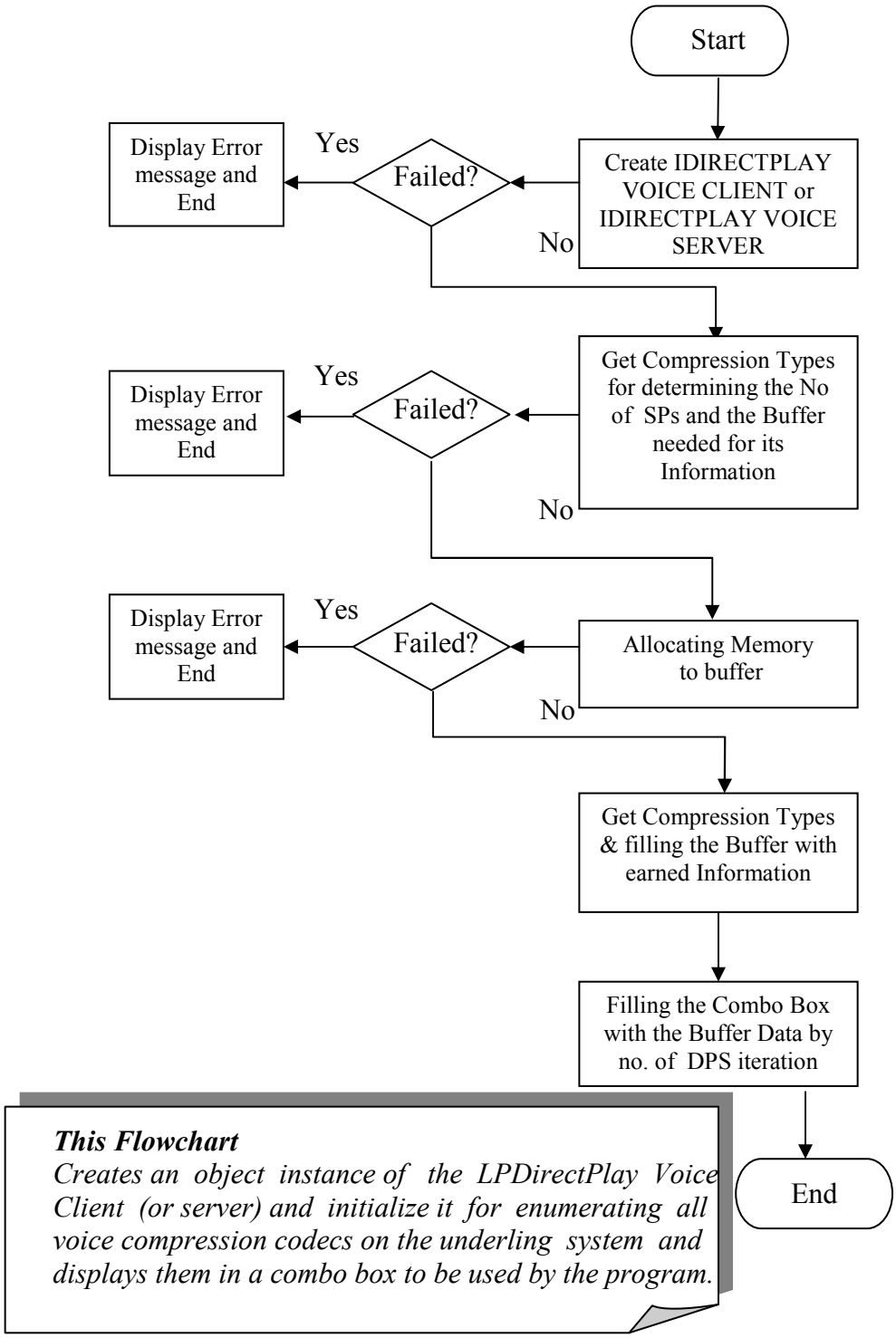


Figure 4: Complete General Flowchart of Setting up the Server/ Client



*SP: Service Provider

Figure 5 : Enumerating Voice Compression Codecs

4.4 Selecting the Client/Server Host:

In order to host the session, the address of the host device should be specified. This work can done by creating an **IDerctPlay8Address** object and calling the **IDerctPlay8Address::SetSP** method:

```
IDerctPlay8Address* g-pDeviceAddress=NULL;
……..
//Create our IDirectPlay8Adress Device Address
        hr=CoCreatInstance(CLSID_DirectPlay8Address,          NULL,
                           CLSCTX_INPROC_SERVER,
                           IID_IDirectPlay8Address,
                           (LPVOID*)&G_pDeviceAddress);

//Set the SP for our Device Address
       hr=g_pDeviceAddress->SetSP(&CLSID_DP8SP_TCPIP);
```

4.5 Connecting to the Client/Server Session:

All Clients must explicitly join the session by connecting them to the host. A connection establishes the client as a member of the session, and provides the host with the information which needs to communicate with the client. The host has the option of accepting or rejecting a connection request.

4.6 Managing the Client/Server Session:

After server and clients are created, initialized and the clients are connected to the server, they must handle the session and all messages (user messages or system messages) that are exchanged between the server and the clients.

As host, the server is responsible for managing the course of the session. The details depend on how the application is designed.

The server should have the capability of removing a player from the session by calling **IDirectPlay8Server::DestroyClient**:

```
HRESULT Destroy Client (
    Const DPNID dpnid Client,
    Const VOID *const pDestroylnfo,
    Const DWORD dwDestroylinfoSize,
    Const DWORD dwFlags
) ;
```

4.7 Handling Client\Server Messages:

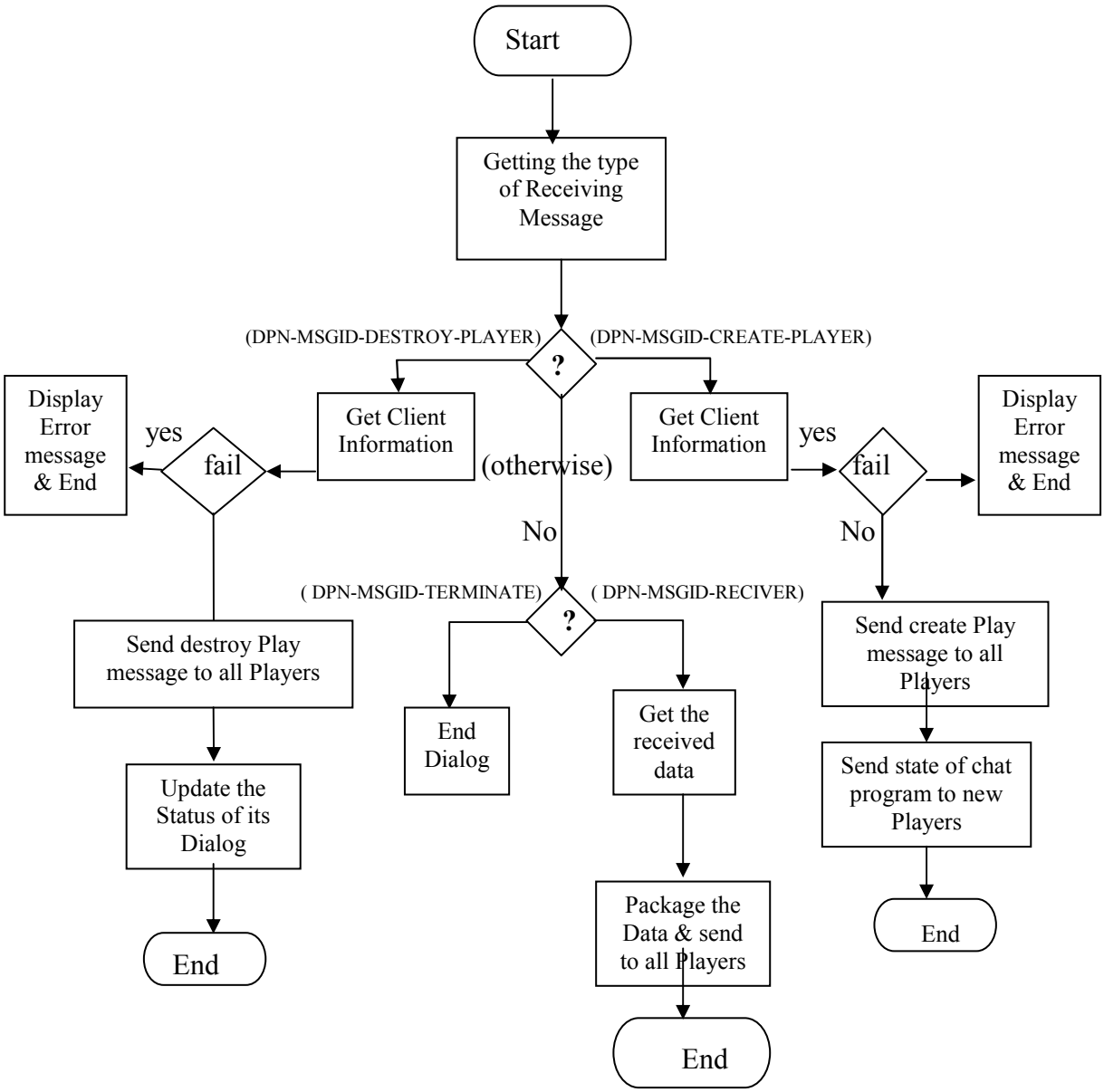
We must handle Microsoft DirectPlay messaging for the designed Client\Server session. First it is necessary to know that:

- A client does not receive messages that carry information about other players and no group-related messages because DirectPlay does not provide a way for a client to know about or to communicate with other clients.
- DirectPlay does not provide host-migration messaging because the server must be the host. A client cannot host the client/server session as shown in figure (6).

4.8 Terminating the Client/Server Session:

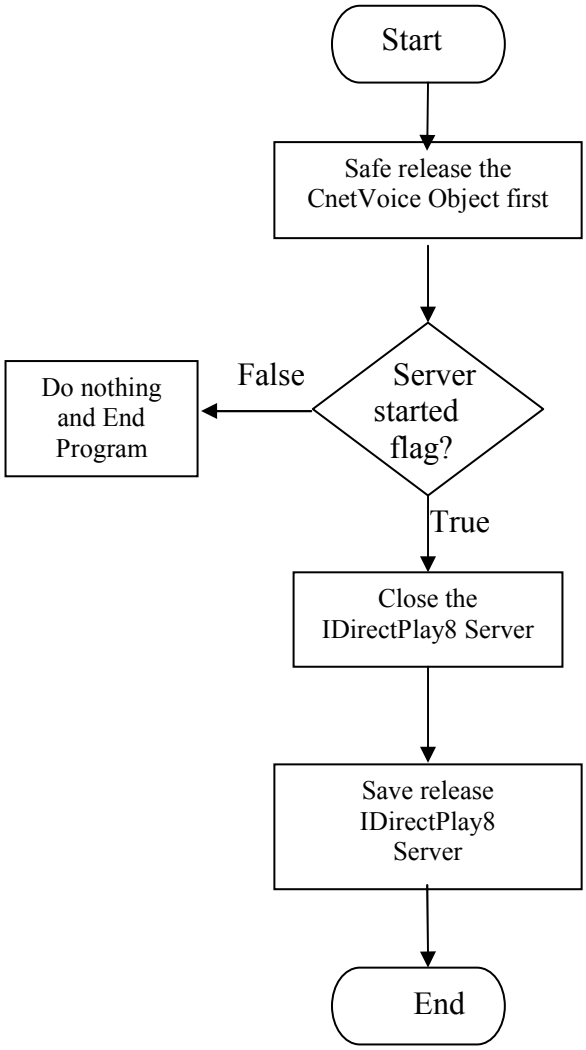
For terminating the client/server session, the server calls ***IDirectPlay8Server::Close.*** These methods terminate all connections and close the session. The client is notified of the session end by a **DPN_MSGID_TERMINATE_SESSION** message as shown in figure (7).

```
HRESULT Close (
Const DWORD dwFlags);
```



This Flowchart
is used for handling the message of IdirectPlay&Server object in sum cases such as:
- *When a new player joins the session (connects to the server).*
- *When an existing player leaves the session (disconnects from the server).*
- *When the session is terminated.*
- *When any message is received from each player of the session.*

Figure 6: DirectPlay Message Handler



This Flowchart
Stops the server by deleting the CNetVoice and closing the IDirectPlay8Server object, then releasing it.

Figure 7 : Stopping the Server

5- Testing

First of all and before testing the designed program, DirectX-9 must be installed on the system.

The message box should be setting in the server program, so that when it is displayed, a new player can join the session. If the server is busy, it is suspended because the message box must be acknowledged by clicking the OK button. Then the user can talk to any of the players microphone. This data voice was sent to the server, not to the players in the session. After clicked the OK button of the message box, the data voice was sent to the players, some of data voice was lost, but it was very little compared to the whole of data voice that was sent to the server.

Another testing decline that the time elapsed to the server stop depends on the number of clients joined to the session. Because first of all the clients must be notified about the termination of the session and after receiving acknowledgment from them, the server closed all the objects.

When testing the quality of sound, found that it is very good, especially by using the best codec from ones available on the designed system.

The speed of the connections between the user and the server computer as found good due to the use of DirectPlay which gives a rapid developing without any hardware concerning, so the user can have a fast networking program.

The speed of the program was tested over two networks (using wireless) that have (300) meters distance between each other so it was very good. The synchronization is very well between the users that where speaking together.

In some situations that server may be busy with some of its internal processing or connection accepting; the whole data voice received by clients are saved in a queue then it will be sent to the clients after server being free.

6- Conclusions

A Client/Server program was implemented in C++ using strong object-oriented development technique.
The visual C++, MFC and API of the C++ itself can helps inr writing all the features explicitly.

It is a reality that DirectPlay is more useful for developing games for networks, since it shields the complexity of network connections and all

protocol and transport connections. Then the developer can develop the game programs more rapidly.
DirectPlay handles all network dependent tasks, also it has the most available messages handling, then it can be used to develop variety of network programs.

DirectPlay is an interface between the applications and the available protocols of the underling system, then it treats all network problems itself and makes relations between the other sides of connection. DirectPlay speaks with DirectPlay on another side, gets its messages, and gives them to the application as needed. The task handles all the messages before sending it to the client.

Another point is about using DirectX_SDK for developing the work. SDK adds a wizard to visual C++ that can build the backbone of the program and then the programmer can expand it.

Using 3D-voice feature of DirectPlayVoice is a good ability to create real games. It can simulate the real world space and give the players a sense that they are playing within a natural area rather than artificial 2D voice effects.

Another feature is guarantying the data, which is communicated between the players. A good quality of sounds and transmission of data is also good in this work. Furthermore, the players can select the best compression codec type for improving their voice transmission.

7- References

1- C++ Standards Committee. Boost library. http://www.boost.org/, 2004-01- 15.website.
2- Jesse Aronson. Dead reckoning: Latency hiding for networked games.http://www.gamasutra.com/, 2004-01-15. website
3- Microsoft DirectX SDK, January-2000.
4- MSDN (Microsoft Developer Network)-January 2000.
5- Microsoft DirectPlay http://www.microsoft.com/DirectX/, 2004-01-15. website
6- WWW.DirectX Experience.Com David Joffe's to Programming Games with DirecX – 1997.
7- WWW.DirectX Experience.Com_DirectPlay programming by Lar Madar -1999.
8- WWW.Microsoft.Com/DirectX.
9-WWW.Yaho.Com_Yaho Search *DirectX Books*_ inside DirectX by Bradley Bargain and peter Donnell.