Introducció a la API DirectPlay

Jordi Montaña Vázquez

Carlos Macías Sevilla

Idees bàsiques

- Llibreria per al desenvolupament d'aplicacions que utilitzin protocols client/servidor o p2p (peer-to-peer).
- Fa una abstracció del tipus de xarxa, configuració de la mateixa… => D'això s'encarrega la API.
- L'usuari només es preocupa de fer que els seus programes "s'entenguin" (sincronització de missatges…).
- Orientat sobretot al desenvolupament de videojocs *multiplayer*.

Funcionalitats que implementa (I)

- Creació i gestió de **sessions** client/server o p2p. (Una sessió és una entitat que agrupa usuaris connectats).
- Gestió de **usuaris** i **grups** dintre d'una sessió.
- Gestió de **missatges** entre usuaris d'una mateixa sessió.

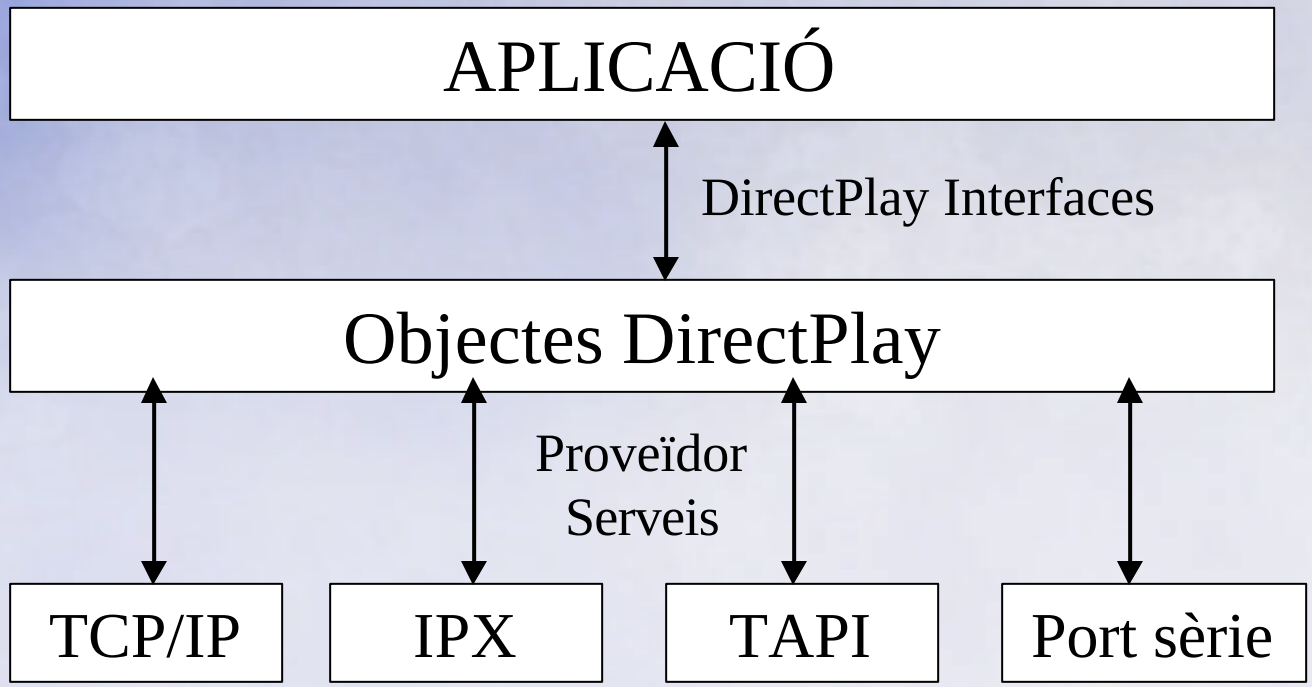
Funcionalitats que implementa (II)

- Permetre a les aplicacions interactuar amb “lobbies”(un lobby és una mena de servidor de chat on els jugadors es connecten per buscar partides a les quals unirse, contrincants, etc.).
- Fins i tot permetre comunicació entre usuaris per **veu**!! (al més pur estil CounterStrike).
- Ús opcional de **threads** per part de la llibreria.

- Protocols suportats

- TCP / IP
- Internet Packet Exchange (IPX)
- Conexió módem a módem
- Comunicació via port sèrie.

- Arquitectura



Protocol de transport DirectPlay (I)

- Objectiu: facil·litar l'enviament de dades des de una aplicació origen a una destí sense preocupar-se del que hi ha entremig.
- Enviament **seqüencial / no seqüencial** de missatges. Seqüencial => El destí rep els missatges en el mateix ordre que han estat enviats.
- Possibilitat d'establir 3 **prioritats** als missatges:
  - Alta (són els que entren primer a la cua de sortida)
  - Mitjana
  - Baixa

Protocol de transport DirectPlay (II)

- **Fragmentació** de missatges i posterior reensamblat. (Si el tamany d'un missatge supera la capacitat d'una xarxa concreta, DirectPlay el retalla i l'enganxa automàticament).
- Control de **congestió**. El flux de sortida de dades es acotat superiorment per el que pot rebre el destí. Això evita que poguem "floodejar" l'aplicació destí amb més missatges dels que pot rebre.

Protocol de transport DirectPlay (III)

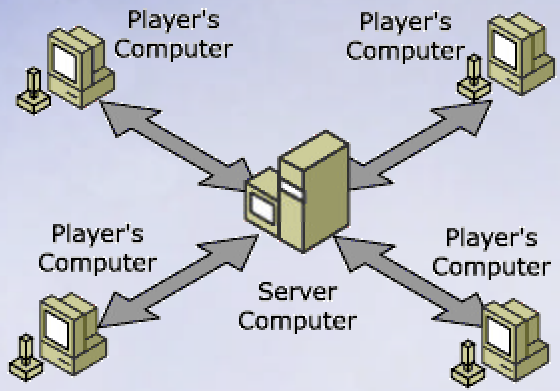
- Possibilitat d'afegir **timeouts**.Quan un missatge ha superat el seu timeout, és eliminat de la cua (independentment de si s'ha enviat o no) per permetre el pas a nous missatges.

• Creació i gestió de sessions

- Una sessió és una instància d'un joc multiplayer concret.
- Una sessió agrupa jugadors simultàniament que executen un client.
- Un usuari pot tenir més d'un jugador (que ha de gestionar l'aplicació mitjantçant a crides de la API DirectPlay).

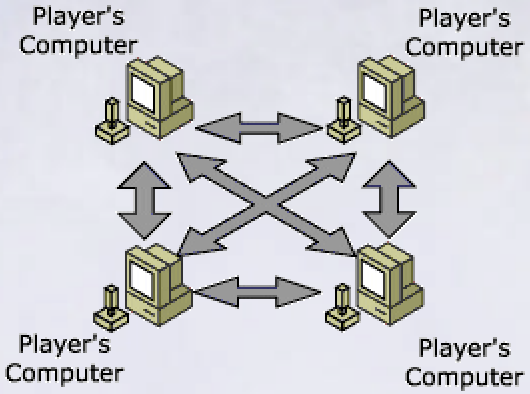
• Sessions: 2 models

- Client / Server (C/S)



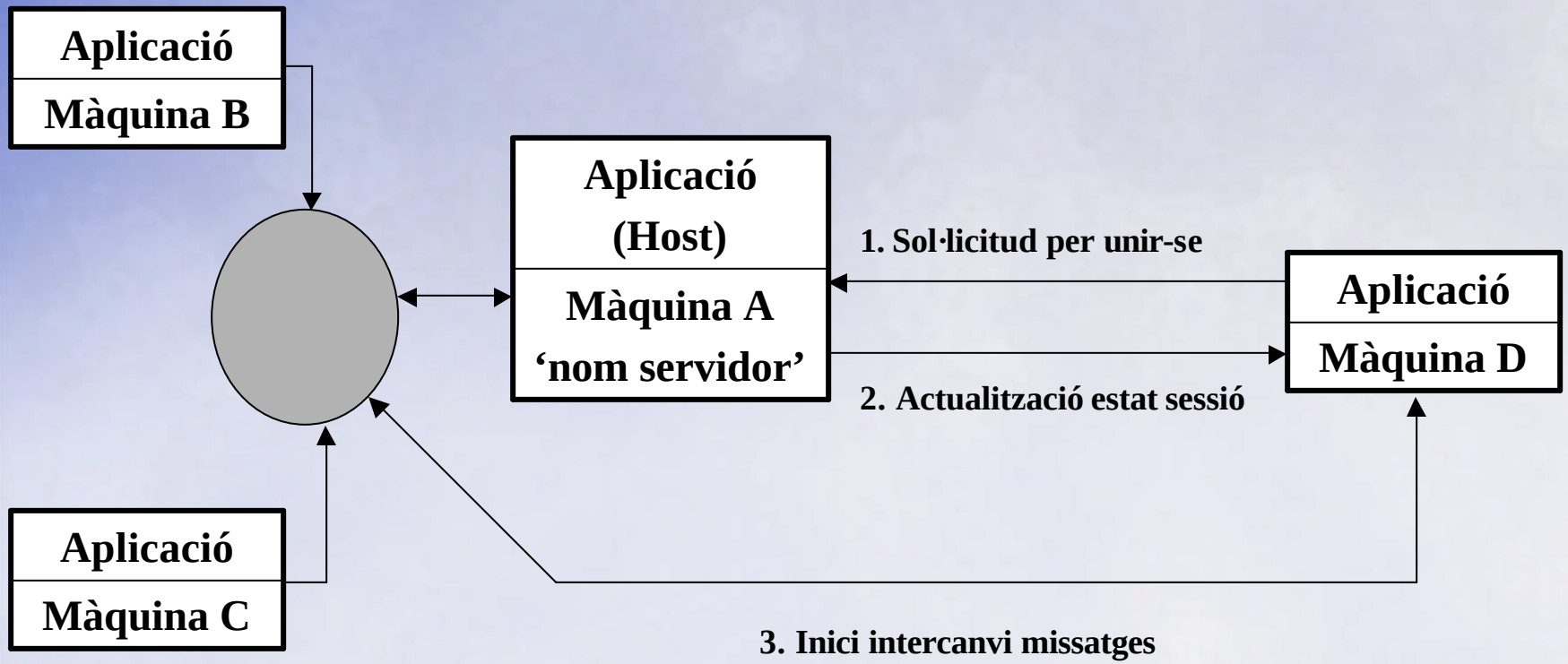
(orientat a connexió)

- Peer-to-peer (p2p)



(no orientat a connexió)

• Sessió p2p: Funcionament bàsic



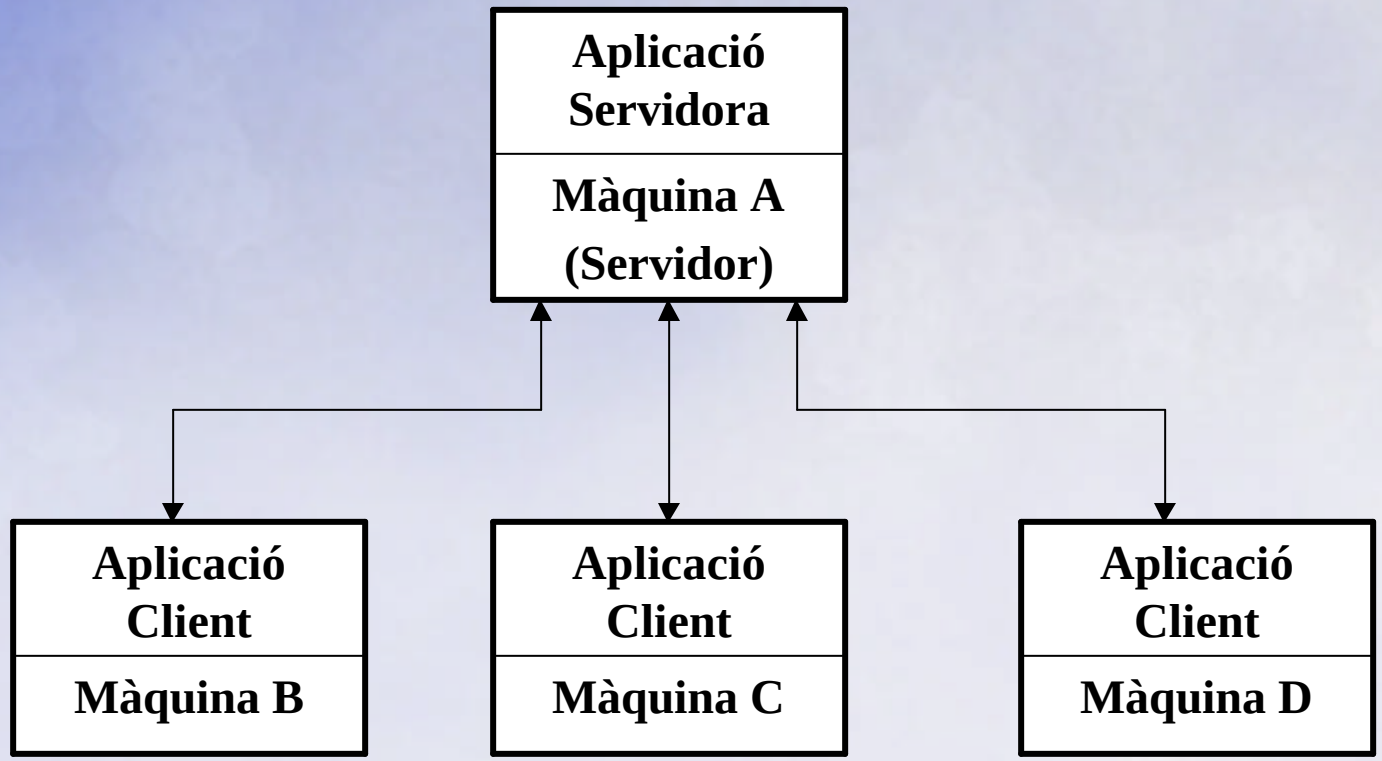
Sessió p2p: Avantatges

- Major simplicitat
  - Senzillament són un grup de usuaris que s'envien missatges entre ells.
  - Un únic codi per a tots els usuaris.
- Enviament directe de missatges
  - No existeix la figura del server => Augmenta la velocitat de transferència d'un usuari a un altre.

Sessió p2p: Inconvenients

- Baixa escalabilitat
  - Cada nou client incrementa **geomètricament** el tràfic de missatges
  - Inviable quan el nombre de clients esdevé > 20-30 aprox. (depén de la xarxa).
- Baix control de dades
  - Ja que gran part del missatges s'envien directament entre clients.

- Sessió C/S: Funcionament bàsic



• Sessió C/S: Avantatges

- Alta escalabilitat
  - Cada nou client incrementa **linealment** el tràfic de missatges
  - Aquesta escalabilitat es converteix en necessària quan el nombre de clients esdevé massiu.
- Control de dades
  - Al ser un model centralitzat, certs aspectes de la partida poden guardar-se i protegir-se en el server, evitant atacs a les dades en els clients.
- Reusabilitat i manteniment de codi
  - Gràcies a la independència de codi entre client i server, correcció de bugs o actualitzacions es realitzarant majoritàriament sobre el server, sense necessitat de proporcionar **n** actualitzacions als clients.

Sessió C/S: Inconvenients

- Manteniment del servei
  - S'ha de disposar d'una màquina amb potencia de càlcul i ample de banda necessaris per atendre als clients.
- Tolerància a fallades
  - En el cas de connexions massives de clients, el server ha de funcionar durant llargs períodes amb una taxa de caigudes pràcticament nul·la.
- Econòmicament més car

Crear una sessió C/S (Server)(I)

- 1: Crear l'objecte Server amb:
  - *CoCreateInstance(...IDirectPlay8Server.)*

- 2: L'aplicació server es comunica amb la API a través de crides a les seves interfícies. Per tal de que la API es pugui comunicar amb la nostra aplicació cal una funció callback que gestioni els missatges (una funció que s'executa cada cop que es rep un missatge).
  - Cridar a *IDirectPlay8Server::Initialize* per indicar quina será la funció que gestioni els missatges rebuts de DirectPlay.
  - A aquesta rutina li passem un punter a la rutina que volem que atengui als missatges.

Crear una sessió C/S (Server)(II)

- 3: Abans que els clients puguin conectar, cal establir els paràmetres del server amb:
  - Cridar a *IDirectPlay8Server::SetServerInfo*
- 4: Crear dispositiu d'adreces:
  - *CoCreateInstance(IDirectPlay8Address...)*
- 5: Assignar un SP i un port en el dispositiu creat
  - *IDirectPlay8Address->SetSP(&CLSID_DP8SP_TCPIP)*
- 6: Indicar a la API que la nostra aplicació està llesta per atendre connexions. Crear la sessió amb:
  - Cridar a *IDirectPlay8Server::Host*
  - *A partir d'aquest moment el server pot atendre a clients*

Unir-se a una sessió C/S (Client)(I)

- 1: Crear l'objecte Client amb:
  - *CoCreateInstance(IDirectPlay8Client...)*
- 2: Anàleg al pas 2 del server, necessitem indicar una funció callback.
  - Cridar a *IDirectPlay8Client::Initialize* per indicar quina serà la funció que gestioni els missatges rebuts de DirectPlay.
- 3: Creem dos dispositius d'adreces
  - *Un Device Address (adreça de la màquina on s'executa l'aplicació)*
  - *Un Host Address (adreça de la màquina remota)*
- 4: Opcionalment, si no se sap a quin host conectar, es poc recòrrer a:
  - *IDirectPlay8Client::EnumHosts* que busca aplicacions per conectar

Unir-se a una sessió C/S (Client)(II)

- 5: Assignar al Device Addres i el Host Address el mateix SP:
  - *IDirectPlay8Address::*SetSP(&guid), A guid li passem el identificador del Service Provider (SP).
- 6: Assignar al Host Address el nom del host i el port on conectar:
  - *IDirectPlay8Address::AddComponent* i passant un *AddComponent(DPNA_KEY_HOSTNAME, hostname, ...)* per indicar que li passem el nom del host
  - *IDirectPlay8Address::AddComponent* i passant un *AddComponent(DPNA_KEY_PORT, &PORT, ...)* per indicar que li passem el port.
- 7: Conectar amb el host:
  - *IDirectPlay8Client::Connect* especificant principalment una adreça device i una adreça host creades anteriorment.

• Més informació a...

- ***http://msdn.microsoft.com/library*** en la secció *Graphics and multimedia -> DirectX -> DirectPlay*
- ***Programación multimedia avanzada con DirectX***, Constantino Sánchez Ballesteros, Ed. Ra-Ma,

  ISBN: 84-7897-342-7

Apèndix

- La següent secció conté exemples de codi font senzills (i alguns no tant) que mostren la funcionalitat de les interfícies amb més nivell de detall que el descrit a les diapos.
- Principalment són “petites” proves que hem realitzat per experimentar amb la API.

• Codi font d'exemple d'un server

```
//includes de DirectPlay
#include <dplay8.h>
#include <dpaddr.h>
#include <objbase.h>
#include "DXUtil.h"

//port per defecte utilitzat per la sessio
#define PORT_DEFECTE 30000

int PORT = PORT_DEFECTE;
IDirectPlay8Server* pIServer = NULL;
IDirectPlay8Address* pIAddress = NULL;

//GUID de la aplicacio generat amb la utilitat GUIDGEN
// {7CA5F767-AB2D-4c65-BC24-77E7B14BAFEE}
GUID guidApp = { 0x7ca5f767, 0xab2d, 0x4c65, { 0xbc, 0x24, 0x77, 0xe7, 0xb1, 0x4b, 0xaf, 0xee } };

//funció callback que gestiona els events del server de directplay
HRESULT WINAPI DirectPlayMessageHandlerServer (PVOID pContextUsuari, DWORD dwTipusMissatge, PVOID
    pMissatge) {

    //context usuari es el context que retornará despres de la crida de callback
    //tipus missatge es omplert per direct play indicant el tipus d'event produit
    //pMissatge es un punter al contingut del missatge

    printf("callback\n");
```

```
switch(dwTipusMissatge) {

            case DPN_MSGID_INDICATE_CONNECT:
            {
                        //algun usuari ha conectat
                        printf("Un usuari ha solicitat connexió\n");

                        break;
            }

            case DPN_MSGID_ENUM_HOSTS_QUERY:
                        {
                        //algun usuari ha sol·licitat una enumeració de sessions
                        printf("Algun usuari ha sol·licitat una enumeració de sessions\n");
                        }

            case DPN_MSGID_NAT_RESOLVER_QUERY:
                        {
                                    printf("Missatge DPN_MSGID_NAT_RESOLVER_QUERY\n");
                                    break;
                        }
```

```
case DPN_MSGID_ADD_PLAYER_TO_GROUP:
                        {
                                    printf("Missatge DPN_MSGID_ADD_PLAYER_TO_GROUP\n");
                                    break;
                        }
            case DPN_MSGID_CREATE_PLAYER:
                        {
                                    //DPNMSG_CREATE_PLAYER* dades;
                                    //dades = (DPNMSG_CREATE_PLAYER) pMissatge;

                                    printf("Missatge DPN_MSGID_CREATE_PLAYER\n");
                                    break;
                        }

            case DPN_MSGID_DESTROY_PLAYER:
                        {
                                    printf("Missatge DPN_MSGID_DESTROY_PLAYER\n");
                                    break;
                        }
    }

    return DPN_OK;
}
```

```
int enumerarSPs(IDirectPlay8Server* pIServer, DPN_SERVICE_PROVIDER_INFO*& pInfoSPs) {

    //funcio que enumera el nombre de serveis disponibles i retorna la quantitat

    DPN_SERVICE_PROVIDER_INFO*     pdnSPInfoEnum;
    DWORD                       dwItems         = 0;
    DWORD                       dwSize          = 0;
    DWORD                                                       i;
    HRESULT                                                 hr;
    char                                                        strBuf[256]             =
    {0};

    // Determine the required buffer size
    hr = pIServer->EnumServiceProviders(NULL, NULL, NULL, &dwSize, &dwItems, 0);

    pInfoSPs = (DPN_SERVICE_PROVIDER_INFO*) new BYTE[dwSize];

    //Fill the buffer with service provider information
    hr = pIServer->EnumServiceProviders(NULL, NULL, pInfoSPs, &dwSize, &dwItems, 0 );

    pdnSPInfoEnum = pInfoSPs;
    for (i = 0; i < dwItems; i++)
    {
            //Mostra per pantalla cadascun dels proveidors de servei disponibles en la maquina
    actual
        DXUtil_ConvertWideStringToAnsiCch( strBuf, pdnSPInfoEnum->pwszName, 256 );
        printf("SP %ld: %s\n", i, strBuf);

            pdnSPInfoEnum++;
    }

    return dwItems;
}
```

```
HRESULT seleccionarSP(DPN_SERVICE_PROVIDER_INFO* pInfoSPs, IDirectPlay8Address* pAddr, int n) {
//funcio que selecciona el proveidor de servei num n de la llista de serveis pInfoSPs per a la
    adreça pAddr

    HRESULT hr = S_OK;
    DPN_SERVICE_PROVIDER_INFO* pdnSPInfoAux = pInfoSPs;
   for (int i = 0; i < n; i++)
   {
           //avança el punter fins al servei n-essim
           pdnSPInfoAux++;
   }

    //si s'ha triat TCP/IP -> establir un port per defecte
    hr = pAddr->AddComponent(DPNA_KEY_PORT, &PORT, sizeof(DWORD), DPNA_DATATYPE_DWORD);
           if(FAILED(hr)) {
                       printf("Error al triar el port\n");
                       exit(1);
           }

    //ara assignem a la adreça aquest servei
    hr = pAddr->SetSP(&(pdnSPInfoAux->guid));

    return hr;
}

HRESULT ConfigServerInfo(IDirectPlay8Server* pServer) {

    DPN_PLAYER_INFO playerInfo;

     // configurar descripció del server
   ZeroMemory(&playerInfo, sizeof(DPN_PLAYER_INFO));
   playerInfo.dwSize = sizeof(DPN_PLAYER_INFO);
   playerInfo.dwInfoFlags = DPNINFO_NAME;
   playerInfo.pwszName = L"Server";
```

```
    playerInfo.pvData = NULL;
    playerInfo.dwDataSize = NULL;
    playerInfo.dwPlayerFlags = 0;

     return pServer->SetServerInfo(&playerInfo, NULL, NULL, DPNSETSERVERINFO_SYNC);


}

HRESULT PrepararHosting(IDirectPlay8Server* pServer, IDirectPlay8Address* pAddr, char nomSessio[])
     {

     WCHAR buffer[256] = {0};
     DPN_APPLICATION_DESC           dpAppDesc;

      // configurar descripció de la nostra aplicacio
    ZeroMemory(&dpAppDesc, sizeof(DPN_APPLICATION_DESC));
    dpAppDesc.dwSize = sizeof(DPN_APPLICATION_DESC);
    dpAppDesc.dwFlags = DPNSESSION_CLIENT_SERVER | DPNSESSION_NODPNSVR;   // Flag describing the
     app
    dpAppDesc.guidApplication = guidApp;          // GUID for the application

    DXUtil_ConvertAnsiStringToWideCch(buffer,nomSessio,256);
     dpAppDesc.pwszSessionName = buffer;          // nom de la sessió


     // Host the application.
    return pServer->Host(&dpAppDesc,       // pdnAppDesc
                         &pAddr, 1,                        // prgpDeviceInfo, cDeviceInfo
                         NULL, NULL,            // pdpSecurity, pdpCredentials
                         NULL,                  // pvPlayerContext
                         0);                    // dwFlags


}
```

```
void AlliberarDirectPlay() {

    if(pIServer != NULL)
            pIServer->Close(DPNCLOSE_IMMEDIATE);

    if(threadPool != NULL)
            threadPool -> Close(0);
}

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int nRetCode = 0;
    int nSPs;
    int res = 0;

    // initialize MFC and print and error on failure
    if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(), 0))
    {
            // TODO: change error code to suit your needs
            cerr << _T("Fatal Error: MFC initialization failed") << endl;
            nRetCode = 1;
    }
    else
    {

            printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

            // Init COM so we can use CoCreateInstance => això extret d'un codi font d'exemple
    (tutorial 01)
            //no se ben bé perquè s'utilitza, però sense això no va i dóna seg. violation, o sigui
    q ho poso :D
            CoInitializeEx(NULL, COINIT_MULTITHREADED);

                        HRESULT hr;
```

```
    //crear objecte server
            hr = CoCreateInstance(CLSID_DirectPlay8Server,
                                                                    NULL,
                                                                    CLSCTX_INPROC_SERVER,
                                                                    IID_IDirectPlay8Server,
                                                                    (LPVOID*) &pIServer);

            if(FAILED(hr)) {
                        printf("No s'ha pogut crear objecte IDirectPlay8Server\n");
                        exit(1);
            }


            //inicialitzar
            hr = pIServer->Initialize(NULL, DirectPlayMessageHandlerServer, 0);

            if(FAILED(hr)) {
                        printf("No s'ha pogut inicialitzar objecte IDirectPlay8Server\n");
                        exit(1);
            }

//enumerarSPs
            DPN_SERVICE_PROVIDER_INFO* pInfoSPs = NULL;
            nSPs = enumerarSPs(pIServer, pInfoSPs);
            printf("S'han trobat %d proveidors de servei per aquesta maquina \n", nSPs);

            printf("Amb quin proveidor de servei vols que s'executi el server? (0, 1, 2,... nSPs -
    1)\n");

            while(res < '0' || res > (nSPs - 1 + '0'))
                        res = _getch();

            printf("Escollit proveidor de servei num %d\n", res - '0');
```

```
    //crear objecte d'adreces
        hr = CoCreateInstance(CLSID_DirectPlay8Address,
                                                                    NULL,
                                                                    CLSCTX_INPROC_SERVER,
                                                                    IID_IDirectPlay8Address,
                                                                    (LPVOID*) &pIAddress );


            if(FAILED(hr)) {
                        printf("No s'ha pogut crear objecte d'adreces\n");
                        exit(1);
            }

            //seleccionar el proveïdor de servei escollit (SP - Service Provider)
            hr = seleccionarSP(pInfoSPs, pIAddress, res - '0');

            if(FAILED(hr)) {
                        printf("No s'ha pogut seleccionar el dispositiu desitjat\n");
                        exit(1);
            }

            char nomsessio[256] = "Sessio de prova";

hr = ConfigServerInfo(pIServer);
            if(FAILED(hr)) {
                        printf("No s'ha pogut configurar la info del server\n");

                        exit(1);
            }

            printf("Preparant host...\n");

            hr = PrepararHosting(pIServer, pIAddress, nomsessio);
```

```
            if(FAILED(hr)) {
                        printf("No s'ha pogut hostatjar la sessio\n");
                        if(hr == DPNERR_DATATOOLARGE) printf("DPNERR_DATATOOLARGE\n");
                        if(hr == DPNERR_INVALIDPARAM) printf("DPNERR_INVALIDPARAM\n");
                        if(hr == DPNERR_DPNSVRNOTAVAILABLE) printf("DPNERR_DPNSVRNOTAVAILABLE\n");
                        if(hr == DPNERR_ALREADYINITIALIZED) printf("DPNERR_ALREADYINITIALIZED\n");

                        exit(1);
            }

            printf("Hosting sessio...\n Premeu 'q' per sortir...\n");

    //bucle on s'atenen missatges
            while(1) {

            }

            //alliberar objectes de directplay
            AlliberarDirectPlay();

            // Cleanup COM   (també extret del codi font del tutorial 01)
            CoUninitialize();

    }

    return nRetCode;
}
```

• Codi font d'exemple d'un client

Molt codi es anàleg al server, només conté el més rellevant.

```
HRESULT Conectar() {

    HRESULT hr = S_OK;
    DPN_APPLICATION_DESC dpAppDesc;
    WCHAR   buffer[128] = {0};

    char hostname[128] = "127.0.0.1";


    //conectem al host 127.0.0.1,
    hr = pIHostAddress->AddComponent(DPNA_KEY_HOSTNAME, hostname, strlen(hostname) + 1 *
    sizeof(char), DPNA_DATATYPE_STRING_ANSI);
    if(FAILED(hr)) {
            printf("Error al establir la host address\n");
            exit(1);
    }

    //en el port per defecte
    hr = pIHostAddress->AddComponent(DPNA_KEY_PORT, &PORT, sizeof(int), DPNA_DATATYPE_DWORD);

    if(FAILED(hr)) {
            printf("Error al establir el port per a la host address\n");
            exit(1);
    }
```

```
ZeroMemory(&dpAppDesc, sizeof(DPN_APPLICATION_DESC));
    dpAppDesc.dwSize = sizeof(DPN_APPLICATION_DESC);
    dpAppDesc.guidApplication = guidApp;

            //conectar
            return pIClient->Connect(&dpAppDesc,       // pdnAppDesc
                                     pIHostAddress,      // pHostAddr
                                     pIDeviceAddress,  // pDeviceInfo
                                     NULL,              // pdnSecurity
                                     NULL,              // pdnCredentials
                                     NULL, 0,           // pvUserConnectData, Size
                                     NULL,              // pvAsyncContext
                                     NULL,              // pvAsyncHandle
                                     DPNCONNECT_SYNC);  // dwFlags
}

HRESULT ReInicialitzar() {

    if(pIClient != NULL)
            pIClient->Close(0);

    return pIClient->Initialize(NULL, DirectPlayMessageHandlerClient, 0);
}
```

```
bool tractarErrorConnect(HRESULT hr) {

    //retorna true si es capaç de recuperar-se del error

    if(hr == DPNERR_HOSTREJECTEDCONNECTION) printf("DPNERR_HOSTREJECTEDCONNECTION\n");
    if(hr == DPNERR_INVALIDAPPLICATION) printf("DPNERR_INVALIDAPPLICATION\n");
    if(hr == DPNERR_INVALIDDEVICEADDRESS) printf("DPNERR_INVALIDDEVICEADDRESS\n");
    if(hr == DPNERR_INVALIDFLAGS) printf("DPNERR_INVALIDFLAGS\n");
    if(hr == DPNERR_INVALIDHOSTADDRESS) printf("DPNERR_INVALIDHOSTADDRESS\n");
    if(hr == DPNERR_INVALIDINSTANCE) printf("DPNERR_INVALIDINSTANCE\n");
    if(hr == DPNERR_INVALIDINTERFACE) printf("DPNERR_INVALIDINTERFACE\n");
    if(hr == DPNERR_INVALIDPASSWORD) printf("DPNERR_INVALIDPASSWORD\n");
    if(hr == DPNERR_NOTHOST) printf("DPNERR_NOTHOST\n");
    if(hr == DPNERR_SESSIONFULL) printf("DPNERR_SESSIONFULL\n");
    if(hr == DPNERR_ALREADYCONNECTED) printf("DPNERR_ALREADYCONNECTED\n");
    if(hr == DPNERR_NOCONNECTION) printf("DPNERR_NOCONNECTION\n");

    if(hr == DPNERR_ALREADYINITIALIZED) {
            printf("Error: DPNERR_ALREADYINITIALIZED\n");
            printf("Reintentant...\n");
            if(FAILED(ReInicialitzar()))
                        return false;
            else
                        if (FAILED(Conectar()))
                        return true;
    }

    return false;
}
```

```
int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int nRetCode = 0;
    int nSPs;
    int res = 0;

    // initialize MFC and print and error on failure
    if (!AfxWinInit(::GetModuleHandle(NULL), NULL, ::GetCommandLine(), 0))
    {
            // TODO: change error code to suit your needs
            cerr << _T("Fatal Error: MFC initialization failed") << endl;
            nRetCode = 1;
    }
    else
    {

            printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n");

            // Init COM so we can use CoCreateInstance => això extret d'un codi font d'exemple
    (tutorial 01)
            //no se ben bé perquè s'utilitza, però sense això no va i dóna seg. violation, o sigui
    q ho poso :D
            CoInitializeEx(NULL, COINIT_MULTITHREADED);

            //crear objecte client
            HRESULT hr;
            hr = CoCreateInstance(CLSID_DirectPlay8Client,
                                                                    NULL,
                                                                    CLSCTX_INPROC_SERVER,
                                                                    IID_IDirectPlay8Client,
                                                                    (LPVOID*) &pIClient);

            if(FAILED(hr)) {
                        printf("No s'ha pogut crear objecte IDirectPlay8Client\n");
                        exit(1);
            }
```

```
//inicialitzar
            hr = pIClient->Initialize(NULL, DirectPlayMessageHandlerClient, 0);

            if(FAILED(hr)) {
                        printf("No s'ha pogut inicialitzar objecte IDirectPlay8Client\n");
                        exit(1);
            }

            //enumerarSPs
            DPN_SERVICE_PROVIDER_INFO* pInfoSPs = NULL;
            nSPs = enumerarSPs(pIClient, pInfoSPs);
            printf("S'han trobat %d proveidors de servei per aquesta maquina \n", nSPs);

            printf("Amb quin proveidor de servei vols que s'executi el server? (0, 1, 2,... nSPs -
    1)\n");

            while(res < '0' || res > (nSPs - 1 + '0'))
                        res = _getch();

            printf("Escollit proveidor de servei num %d\n", res - '0');

            //crear objecte d'adreces local
          hr = CoCreateInstance(CLSID_DirectPlay8Address,
                                                                        NULL,
                                                                        CLSCTX_INPROC_SERVER,
                                                                        IID_IDirectPlay8Address,
                                                                        (LPVOID*) &pIDeviceAddress
    );


            if(FAILED(hr)) {
                        printf("No s'ha pogut crear objecte d'adreces local\n");
                        exit(1);
            }
```

```
//crear objecte d'adreces remot
        hr = CoCreateInstance(CLSID_DirectPlay8Address,
                                                                NULL,
                                                                CLSCTX_INPROC_SERVER,
                                                                IID_IDirectPlay8Address,
                                                                (LPVOID*) &pIHostAddress );


           if(FAILED(hr)) {
                       printf("No s'ha pogut crear objecte d'adreces remot\n");
                       exit(1);
           }

           //seleccionar el proveïdor de servei escollit (SP - Service Provider)
           hr = seleccionarSP(pInfoSPs, pIDeviceAddress, res - '0');
           hr = seleccionarSP(pInfoSPs, pIHostAddress, res - '0');

           if(FAILED(hr)) {
                       printf("No s'ha pogut seleccionar el dispositiu desitjat\n");
                       exit(1);
           }

          printf("Conectant a 127.0.0.1:30000...\n");
           hr = Conectar();

           if(FAILED(hr)) {
                       if (tractarErrorConnect(hr));
                       else {
                                   AlliberarDirectPlay();
                                   printf("No s'ha pogut conectar.\n");
                                   exit(1);
                       }
           }

           printf("Conectat.\n");
```

```
//bucle on s'esperen clients fins que es premi 'q'
            while(res != 'q')
                        res = _getch();

            //alliberar objectes de directplay
            AlliberarDirectPlay();

            // Cleanup COM   (crida també extreta del codi font del tutorial 01)
            CoUninitialize();

    }

    return nRetCode;
}
```

Apèndix 2: Implementació des de C#

- La següent secció conté un codi desenvolupat per nosaltres d'un petit client de chat implementat en el recent llenguatge C#.

```
using System;
using Microsoft.DirectX.DirectPlay;



namespace PruebaDPlay
{
		/// <summary>
		/// Esta clase guarda la información del Host
		/// </summary>
		public class HostInfo
		{
				public Guid    GuidInstance;    // Guid de la sesión
				public Address HostAddress;     // Direccion del host de la sesión de directPlay
				public string  SessionName;     // Nombre de la sesión

				/// <summary>
				/// Reescribimos la igualdad
				/// </summary>
				public override bool Equals(object obj)
				{
						HostInfo node = (HostInfo) obj;
						return GuidInstance.Equals(node.GuidInstance);
				}

				/// <summary>
				/// Reescribimos el HashCode
				/// </summary>
				public override int GetHashCode()
				{
						return GuidInstance.GetHashCode();
				}
```

```
/// <summary>
                        /// Reescribimos la función ToString usada para escribir por pantalla el HostInfo
                        /// </summary>
                        public override string ToString()
                        {
                                string displayString = (SessionName != null) ? SessionName : "<unnamed>";
                                displayString += " (" + HostAddress.GetComponentString("hostname");
                                displayString += ":" + HostAddress.GetComponentInteger("port").ToString() + ")";

                                return displayString;
                        }
        }
}
```

```
using System;
using System.IO;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using Microsoft.DirectX;
using Microsoft.DirectX.DirectPlay;


namespace PruebaDPlay
{

        /// <summary>
        /// Tipo de conexión
        /// </summary>
        public enum ConnectionType { Disconnected, Hosting, Connected };



        /// <summary>
        /// Prueba de Peer2Peer con DirectPlay
        /// </summary>
        public class PruebaDPlay : IDisposable
        {

                #region Campos
//---------------------------------------------------------------------------
                // Identificador de la prueba en la red
                private static readonly Guid   m_AppGuid = new Guid("5e4ab2ee-6a50-4614-807e-c632807b5eb1");
```

```
                                    // Puerto por defecto
                                    public static readonly int     DefaultPort  = 2603;



                                    // DirectPlay
                                    private Peer              m_Peer = null;                  // Objeto Peer de DirectPlay
                                    private Address           m_LocalAddress = new Address();    // Dirección local

                                    // Variables globales del programa
                                    private Ventanita                                                 m_Form = null;
                                                                                                                        // Interfaz
                                    private Chat                                                                        m_Form2 = null;
                                                                                                                                          // Chat
                                    private ConnectionType      m_Connection = ConnectionType.Disconnected;     // Estado de conexión
                                    private string              m_SessionName = "Prueba DirectPlay";        // Nombre de la sesión (Host)
                                    private ArrayList           m_FoundSessions = new ArrayList();
                        // Lista de sesiones (Conectar)

//-------------------------------------------------------------------------------
                                    #endregion //fin de Campos

                                    #region Propiedades
                                    //-------------------------------------------------------------------------------

                                    public ConnectionType Connection { get{return m_Connection; } }
                                    public ArrayList      FoundSessions { get{ return m_FoundSessions; } }

                                    //-------------------------------------------------------------------------------
                                    #endregion // Fin de Propiedades

                                    #region Constructor
//-------------------------------------------------------------------------------
```

```
#region Constructor
//---------------------------------------------------------------------------
                    /// <summary>
                    /// Constructor
                    /// </summary>
                    public PruebaDPlay()
                    {
                                        // Inicializa el WinForm
                                        m_Form = new Ventanita(this);
                                        m_Form2 = new Chat(this);

                                        // Le pasa el puerto por defecto al WinForm
                                        m_Form.RemotePort = DefaultPort;
                                        m_Form.CrearPuerto = DefaultPort;

                                        // Inicializa el Peer de DirectPlay
                                        InitDirectPlay();

                                        // Set the service provider for our local address to TCP/IP
                                        m_LocalAddress.ServiceProvider = Address.ServiceProviderTcpIp;

                                        // Verify the computer supports our chosen service provider
                                        if (!ProveedorValidoQ(m_LocalAddress.ServiceProvider))
                                        {
                                                            MessageBox.Show(m_Form,
                                                                                "Necesitas tener instalado el protocolo TCP/IP
para usar esta prueba",
                                                                                "DirectPlay Tutorial",
```

```
MessageBoxButtons.OK, MessageBoxIcon.Error);

                                                m_Form.Dispose();
                                        }

                                }

//---------------------------------------------------------------------------
                                #endregion // fin de Constructor

                                #region Destructor
//---------------------------------------------------------------------------
                                /// <summary>
                                /// Destructor
                                /// </summary>
                                public void Dispose()
                                {
                                        if (m_Peer != null && !m_Peer.Disposed)
                                                m_Peer.Dispose();
                                }
//---------------------------------------------------------------------------
                                #endregion // fin de Destructor
```

```
                #region InitDirectPlay
//-------------------------------------------------------------------------------
                private void InitDirectPlay()
                {
                        // Reiniciamos recursos
                        if (m_Peer != null)
                                m_Peer.Dispose();

                        // Creamos un nuevo objeto Peer de DirectPlay
                        m_Peer = new Peer();

                        // Recogemos eventos de DirectPlay
                        m_Peer.FindHostResponse += new
FindHostResponseEventHandler(FindHostResponseHandler);
                        m_Peer.SessionTerminated += new
SessionTerminatedEventHandler(SessionTerminatedHandler);
                        m_Peer.Receive += new ReceiveEventHandler(ReceiveHandler);

                        m_Connection = ConnectionType.Disconnected;
                }
//-------------------------------------------------------------------------------
                #endregion //fin de InitDirectPlay
```

```
#region Eventos DirectPlay
                                        //-------------------------------------------------------------------------------------

                                        /// <summary>
                                        /// Proceso que manejará el evento "esperar respuesta del Host"
                                        /// </summary>
                                        public void FindHostResponseHandler(object sender, FindHostResponseEventArgs args)
                                        {
                                                HostInfo Node = new HostInfo();
                                                Node.GuidInstance  = args.Message.ApplicationDescription.GuidInstance;
                                                Node.HostAddress   = (Address) args.Message.AddressSender.Clone();
                                                Node.SessionName   = args.Message.ApplicationDescription.SessionName;

                                                // Si no lo teníamos lo añadimos
                                                if (!FoundSessions.Contains(Node))
                                                        FoundSessions.Add(Node);
                                        }

                                        /// <summary>
                                        /// Proceso que manejará el evento "sesión finalizada"
                                        /// </summary>
                                        public void SessionTerminatedHandler(object sender, SessionTerminatedEventArgs args)
                                        {
                                                MessageBox.Show(m_Form, "Conexión perdida o el Host ha cerrado sesión",
"Aviso");
                                                m_Form2.Close();
                                                Desconectar();
                                        }
```

```
/// <summary>
                                        /// Proceso que manejará la recepción de datos
                                        /// </summary>
                                        public void ReceiveHandler(object sender, ReceiveEventArgs args)
                                        {
                                                        if (m_Form2.listBox.Enabled)
                                                        {
                                                                        NetworkPacket packet = args.Message.ReceiveData;
                                                                        byte[] data = (byte[]) packet.Read(typeof(byte), packet.Length);
                                                                        m_Form2.listBox.Items.Add("-> " +
Encoding.Unicode.GetString(data));
                                                        }
                                        }
                                        //-------------------------------------------------------------------------------------
                                        #endregion //fin de Eventos Directplay
```

```
/// <summary>
                    /// Conecta como Host
                    /// </summary>
                    public void HostSession()
                    {
                                m_Form.mensaje("Conectando...");

                                m_SessionName = m_Form.sesionbox.Text;

                                // Create una descripción de aplicación
                                ApplicationDescription AppDesc = new ApplicationDescription();
                                AppDesc.GuidApplication = m_AppGuid;
                                AppDesc.SessionName = m_SessionName;
                                AppDesc.Flags = SessionFlags.NoDpnServer;

                                // Configura el puerto por el que vamos a hacer de Host
                                m_LocalAddress.AddComponent("port", m_Form.CrearPuerto);

                                try
                                {
                                            // Intentamos hacer de host de una nueva sesión
                                            m_Peer.Host(AppDesc, m_LocalAddress);

                                            m_Connection = ConnectionType.Hosting;

                                            m_Form2.ShowDialog(m_Form);
                                            m_Form.mensaje("Hosting: " + m_SessionName +
                                                        " (" + m_Form.CrearPuerto.ToString() + ")");
```

```
}
                                        catch(Exception ex)
                                        {
                                                m_Form.mensaje(ex.Message);
                                        }
                                }

/// <summary>
                                /// Conectarse a una sesión
                                /// </summary>
                                public void ConnectToSession()
                                {

                                        m_Form.mensaje("Conectando...");

                                        // Coger el host deseado
                                        HostInfo SelectedHost = (HostInfo) m_Form.listado.SelectedItem;
                                        if (SelectedHost == null)
                                                return;

                                        // Crear la descripción de aplicación del Host
                                        ApplicationDescription appDesc = new ApplicationDescription();
                                        appDesc.GuidInstance = SelectedHost.GuidInstance;
```

```
                    // Intentamos conectarnos a la sesión DirectPlay elegida. Cuando estemos
                    // conectados, recibiremos mensajes y eventos de DirectPlay. Como pasamos el
                    // flag "Sync", la conexión se bloqueará hasta que se acabe el time out.
                    try
                    {
                              m_Peer.Connect(appDesc,
          // Descripción de la aplicación
                                        SelectedHost.HostAddress,
// Dirección de Host
                                        m_LocalAddress,
                    // Dirección local
                                        null,
                                        // Datos de usuario (ninguno)
                                        ConnectFlags.Sync);
                    // Flags

                              m_SessionName = SelectedHost.SessionName;
                              m_Connection = ConnectionType.Connected;
                              m_Form2.ShowDialog(m_Form);
                              m_Form.mensaje("Conectado: " + m_SessionName);

                    }
                    catch(Exception ex)
                    {
                              m_Form.mensaje(ex.Message);
                              return;
                    }
          }
```

```
}
                                        catch(Exception ex)
                                        {
                                                m_Form.mensaje(ex.Message);
                                        }
                                }

/// <summary>
                                /// Conectarse a una sesión
                                /// </summary>
                                public void ConnectToSession()
                                {

                                        m_Form.mensaje("Conectando...");

                                        // Coger el host deseado
                                        HostInfo SelectedHost = (HostInfo) m_Form.listado.SelectedItem;
                                        if (SelectedHost == null)
                                                return;

                                        // Crear la descripción de aplicación del Host
                                        ApplicationDescription appDesc = new ApplicationDescription();
                                        appDesc.GuidInstance = SelectedHost.GuidInstance;
```

```
public void Send()
{
        NetworkPacket packet = new NetworkPacket();
        packet.Write(Encoding.Unicode.GetBytes(m_Form2.textBox.Text));

        m_Peer.SendTo((int) PlayerID.AllPlayers,        //Enviar a toda la sesión
                packet,
                                                        // Datos enviados
                0,
                                                                //
Timeout (por defecto)
                SendFlags.Sync |
        // Flags
                SendFlags.NoLoopback);

        m_Form2.listBox.Items.Add("<- " + m_Form2.textBox.Text);
        m_Form2.textBox.Text = "";
}


/// <summary>
/// Desconecta la conexión
/// </summary>
public void Desconectar()
{
        InitDirectPlay();
        m_Form.mensaje("Desconectado");
}
```

```
/// <summary>
        /// Encuentra todas las sesiones con la dirección indicada
        /// </summary>
        /// <param name="hostname">Dirección IP o hostname</param>
        /// <param name="port">Puerto remoto</param>
        public void EnumerarSesiones(string hostname, int port)
        {
                Address HostAddress = new Address();
                HostAddress.ServiceProvider = Address.ServiceProviderTcpIp;

                // Si hay, añadimos el host
                if (hostname.Length > 0)
                        HostAddress.AddComponent("hostname", hostname);

                // Si tenemos el purto, lo añadimos
                if (port > 0)
                        HostAddress.AddComponent("port", port);

                ApplicationDescription AppDesc = new ApplicationDescription();
                AppDesc.GuidApplication = m_AppGuid;

                // Buscamos las direcciones que coincidan. Cuando encontramos alguna,
                // DirectPlay llama nuestro FindHostResponseHandler. Como pasamos el
                // flag "Sync", la conexión se bloqueará hasta que se acabe el time out.
```

```
try
                                                {
                                                        m_Peer.FindHosts(AppDesc,               // Descripcion de la
aplicación
                                                                HostAddress,
                        // Dirección del Host
                                                                m_LocalAddress,
                        // Dirección Local
                                                                null,
                                                // No pasamos datos
                                                                0,
                                                        // Numero de datos que pasamos (por defecto)
                                                                0,
                                                        // Intervalo de reintento (por defecto)
                                                                0,
                                                        // Timeout (por defecto)
                                                                FindHostsFlags.Sync);                   // Flags
                                                }
                                                catch(Exception ex)
                                                {
                                                        Console.WriteLine("Error:"+ex.Message);
                                                        return;
                                                }
                                        }
```

```
/// <summary>
                    /// Verica si el proveedor de servicio figura en la lista de proveedores
                    /// instalados
                    /// </summary>
                    /// <param name="proveedor">Guid del proveedor que hay que buscar</param>
                    /// <returns>cierto si el proveedor es valido</returns>
                    private bool ProveedorValidoQ(Guid proveedor)
                    {
                                        // Listar los proveedores mediante DirectPlay
                                        ServiceProviderInformation[] SPInfoArray = m_Peer.GetServiceProviders(true);

                                        // Por cada proveedor en la lista...
                                        foreach (ServiceProviderInformation info in SPInfoArray)
                                        {
                                                            // Comparar el proveedor de la lista con el parametro
                                                            if (info.Guid == proveedor)
                                                                                return true;
                                        }

                                        // No encontrado
                                        return false;
                    }
```

```
/// <summary>
        /// Enumera los proveedores por pantalla
        /// </summary>
        public void EnumerarProveedores()
        {
                ServiceProviderInformation[] SPInfoArray = null;

                try
                {
                        // Preguntamos a DirectPlay por la lista de proveedores
                        SPInfoArray = m_Peer.GetServiceProviders(true);
                }
                catch(Exception ex)
                {
                        Console.WriteLine(ex.Message);
                }

                //Escribimos la lista de proveedores por pantalla
                int i = 0;
                foreach(ServiceProviderInformation info in SPInfoArray)
                {
                        i++;
                        Console.Write(i);
                        Console.WriteLine(": "+info.Name);
                }
        }
```

```
        static void Main()
        {
            PruebaDPlay a = new PruebaDPlay();
            if (!a.m_Form.IsDisposed)
                Application.Run(a.m_Form);
            a.Desconectar();
            a.Dispose();
        }
    }
}
```