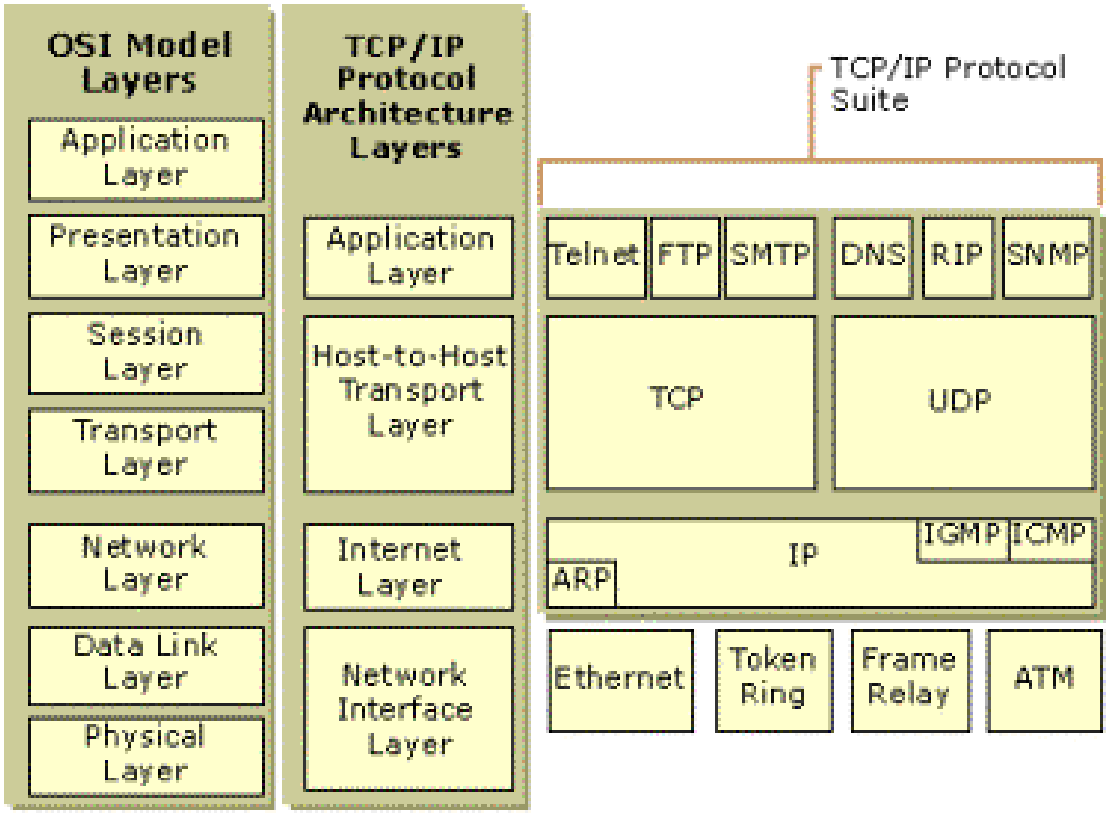
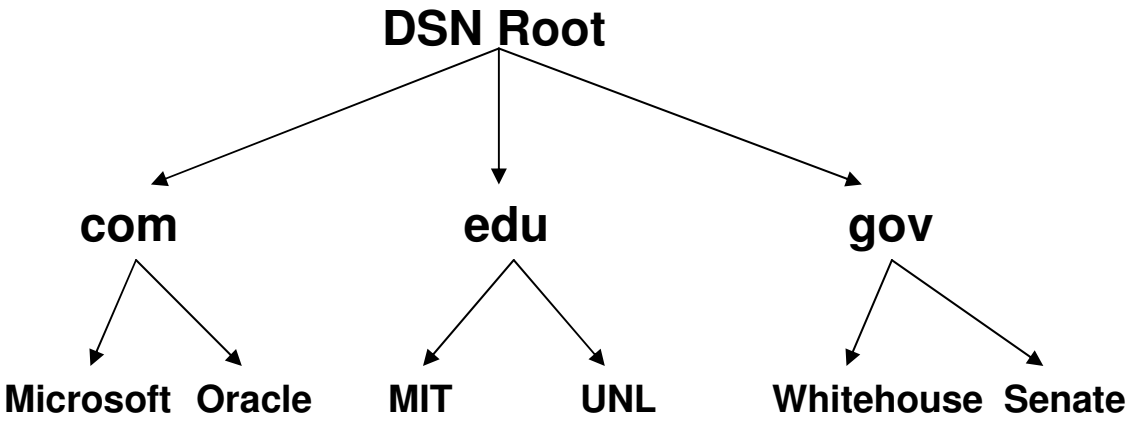
# Програмиране в среда на Internet



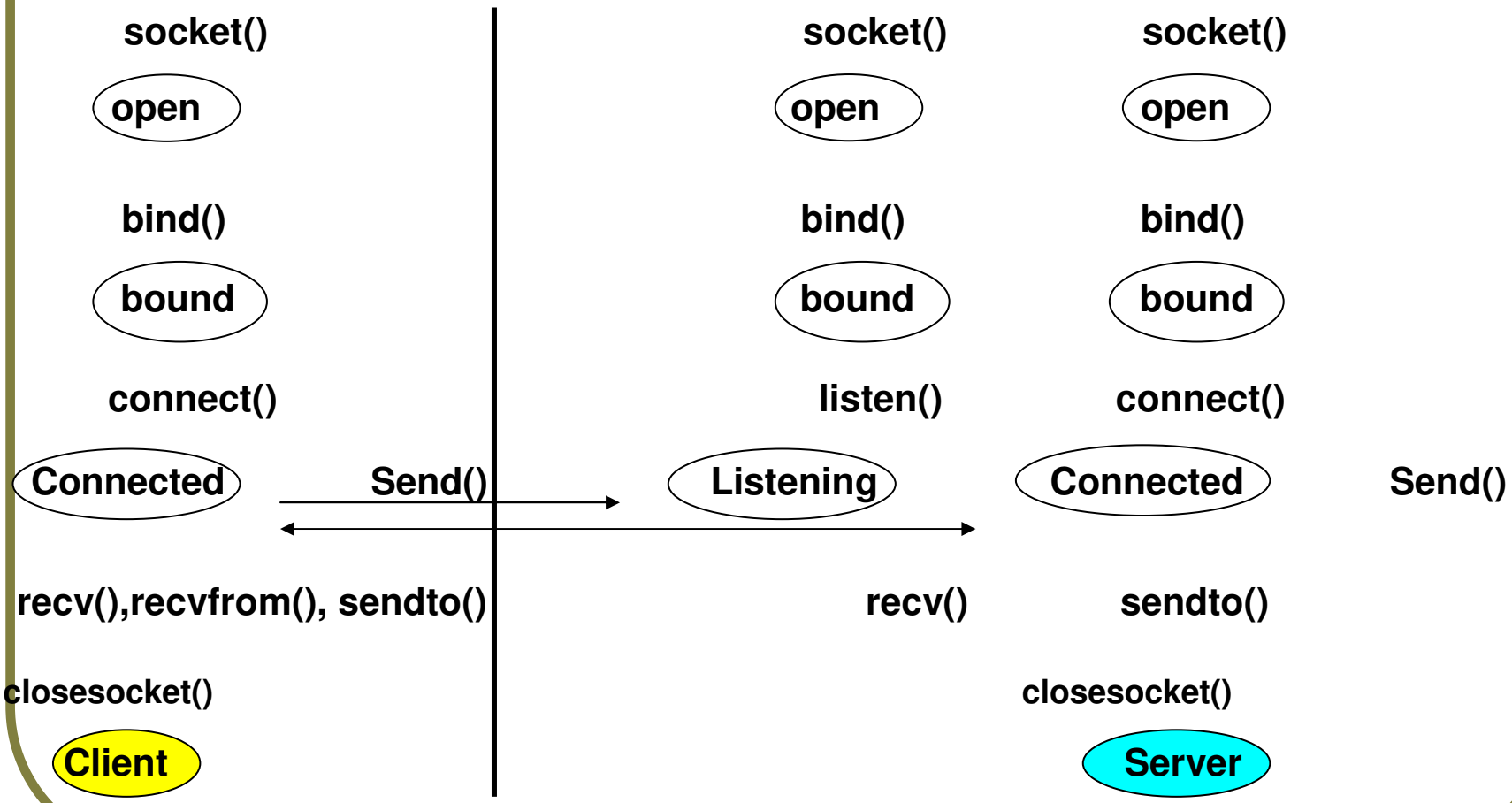
# TCP/IP protocol architecture

**Рутирането използва и DNS (Domain name service).**



# Обща схема на процеса на свързване

*(пример – куриер на DHL)*



## IP протокол. Ето проста IP дейтаграма
## connectionless, unreliable, responsible for addressing&routing packets

## (500 – 64000 байта)

| | | |
|---|---|---|
| **4 бита (д-на на хедър)** | | **16 бита- д-на дейтаграм** |
| | | |
| Fragments, TTL,protocol | | **16 бита – контр. сума** |
| **32 бита IP адрес на подател** | | |
| **32 бита IP адр. на получател** | | |
| **опции (ако има)** | | |
| **Данни** | | |

# Втори протокол е User Datagram Protocol – UDP:

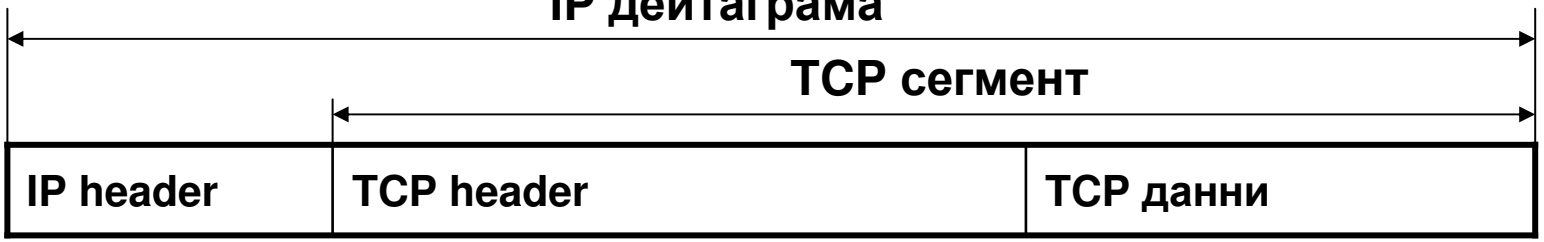
| | |
|---|---|
| 16 бита – номер порт на подател | 16 бита номер порт получател |
| 16 бита – обща д-на UDP | 16 бита – обща контр. сума |
| данни | |


За малки, неотговорни пакети данни. Не гарантира правилното получаване
Примери на използване: NetBIOS Simple Network  Management Protocol

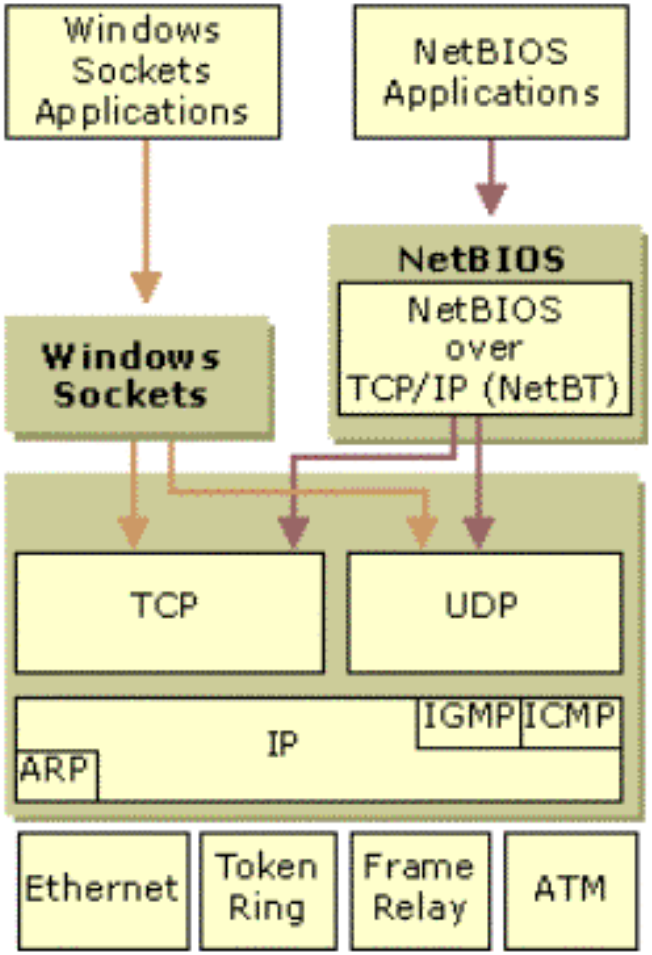
**TCP протокол:** възстановява последователността, reliable, connection-oriented, Сегментно-ориентиран. За всеки сегмент се връща ACK. Комуникацията е 3 стъпки, Hand shake.

**TCP сегмент:**

| | |
|---|---|
| 16 bits номер порт на подател | 16 бита порт получател |
| 32 бита – пореден номер | |
| 32 бита - пореден номер за потвърждение | |
| 4 бита – д-на header | flags-SYN,ACK,FIN |
| 16 бита обща контр. сума | |
| опции (ако има) | |
| данни | |



**Windows Sockets Interface (API):**
**•Stream sockets: 2 посочни, надеждни, с възст. последователността – с TCP**
**• datagram socket: 2 посочни, с използване на UDP**



## WinSock API функции (44)  (кодът е в wsock32.dll от ядрото на windows)
## С тях се пише код както за клиент така и за сървър

* управление на разговор: преобразуване на формати, ..
* попълване база данни: информация за host, services, protocols (gethostbyaddr()…)
* за работа със sockets: bind(), connect(), listen(), recv(), send(), socket(), shutdown()
* за разширения към Windows: от последователен към message oriented…

# Работа с WinSock класове ( CAsincSocket, CSocket )

**Процедурата за работа със сокети е следната:**

**1. конструира се обект CSocket:**

```
CSocket sockSrve;                    или                    CSocket sockClient;
```

**2.създава се SOCKET handle ( ako е сървър се указва и порт):**

```
sockSrve.Create(nPort)               или        sockClient.Create();
```

**3. комуникация**

```
ako е клиент –                                  CAsyncSocket::Connect()
ako е сървър –                                  CAsyncSocket::Listen()
След получаване заявка – в сървъра              CAsyncSocket::Accept(.., * CSocket,  )
sockSrve.Listen()                               sockClient.Connect(strAddr, nPort)
CSocket sockRecv;
SockSrve.Accept(sockRecv);
```

**4. създава се CSocketFile обект и се асоциира с CSocket:**

```
CSocketFile file(&sockRecv);
```

**5. създава се CArchive обект и се асоциира с обекта CSocketFile:**

```
CArchive arIn(&file, CArchive::load);        //архив за вх операции;
CArchive arOut(&file, CArchive::store);      // архив за изходни операции.
```

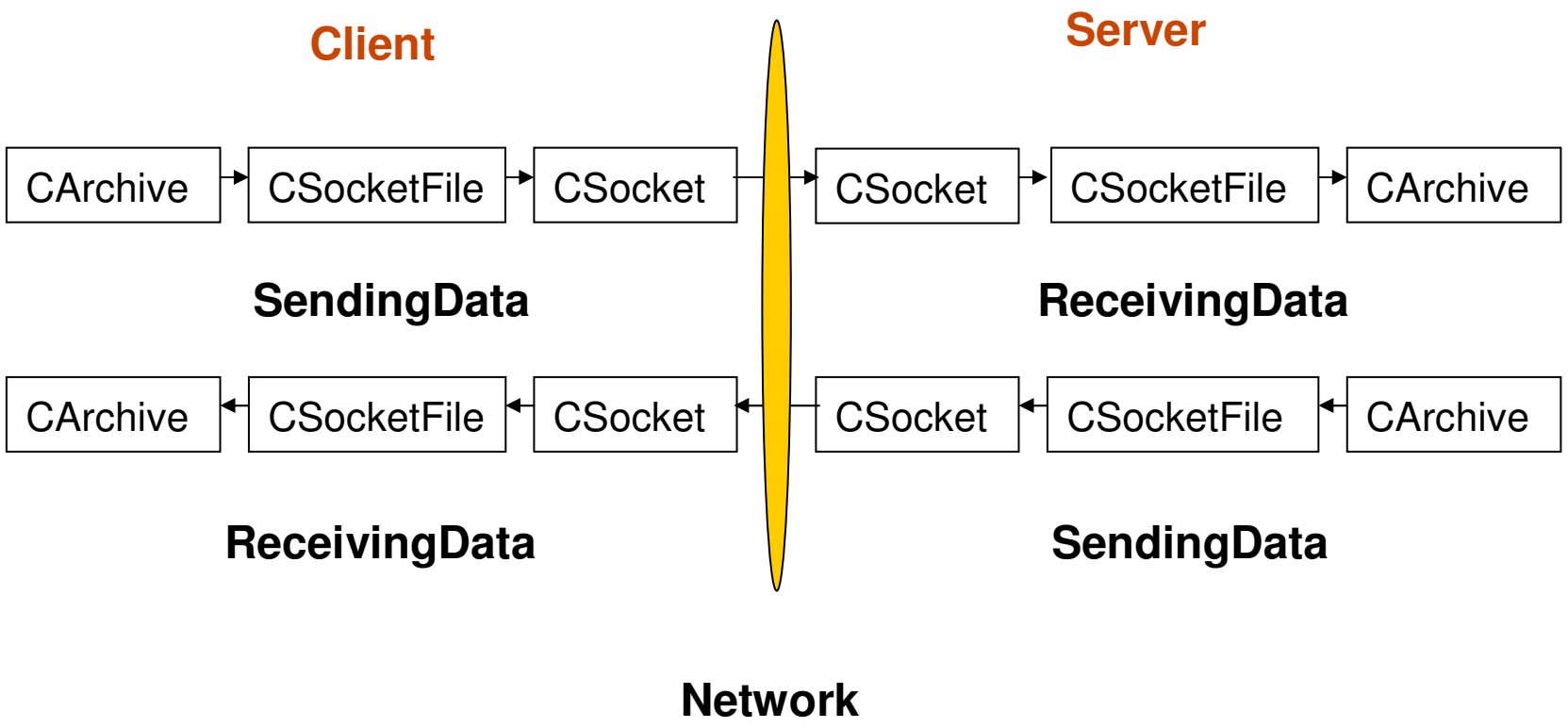
**6.използва се CArchive обектите за изпращане и получаване на данните:**

```
arIn >> dwValue                                          arIn >>dwValue
arOut << dwValue                                         arOut << dwValue
```

**7. накрая се разрушават обектите CArchive, CSocket , CSocketFile.**

CSocket обектът съществува в 2 състояния: асинхронно и синхронно.

## схема на използването на обектите при обмен:



## CAsyncSocket

Обхваща функционалността на WinSock API:

```
CAsyncSocket mysock;
mysock.Create();          }      създаване
```

Има над 20 ф-ии:

| | |
|---|---|
| Accept | създава връзка със сокета |
| Bind | асоциира локален адрес със сокета |
| Connect | прави връзка със съответстващия сокет |
| Receive | |
| Send | |
| overridable | |
| OnAccept | съобщава на прослушващ сокет, че може да accept чакаща заявка за връзка |
| OnReceive | notifies a listening socket че има чакаща данна за получаване с Receive |

## CSocket

## CSocketFile

```
CSocketFile(CSocket* pSocket, …);          //работи последователно, с прекъсвания
```

## WinInet (подходящ за многонишково програмиране)

**Това е API от по-високо ниво, но  работи само за HTTP,FTP, Gopher клиенти, не и за сървъри.**
**Работи синхронно и асинхронно.**
**Кодът е в WININET.DLL.**

**Предимства на WinInet:**
**1. кеширане**
**2. Прекарва обмена през proxy сървър.**
**3. буфрира се вход/изхода**
**4. удобен API (окрупнени ф-ии)**
**5. изгражда автоматично headers, незабелязано пренасочва повикванията.**

## Обвивка на WinInet в MFC класове

**CInternetSession**
**Конструкторът изработва session handle към FTP, HTTP, FILE.**
**Тогава може да се вика OpenURL()…..**
**CStdioFile *OpenURL(pstrURL, незад. параметри);**

**CHttpConnection (наследил CInternetConnection)**
**- след конструиране на CInternetSession**
**- с member ф-цията му – GetHttpConnection(), връща connection handle**
**- Връзката е активна само за времето на предаване на файл.**
**CFtpConnection, CGopherConnection**

**CInternetFile (наследник на CFile →CStdioFile →CInternetFile →CHTTPFile)**
**CHTTPFile* httpfile = (CHTTPFile*) internetSession.OpenURL(strurl);**
**//прави връзка и създава обект**
**CHttpFile**

**CFtpFileFind (за работа с директории и файлове в FTP сървър)**
**CInternetException**

# Нотификации за прогреса на Internet сесия (работят в блокиращ режим (нишки) и асинхронен (при message based)

**За да се работи с тях се конструира наследник, напр:**

***class CCallBackInternetSession: public CInternetSession***
***{..public: CCallBackInternetSession(…){EnableStatusCallback(..)}***
***protected: OnStatusCallback(..)};***

**като:**
**1. в конструктора се вика *CInternetSession::EnableStatusCallback(),* която разрешава обратни повиквания при промяна статуса.**
**2. предефинира се member функцията на CInternetSession, която се вика при променен статус**
***OnStatusCallback(… dwInternalStatus,…)*** // в нея може да се обнови интерф.

**Ето код на примерна кл. програма, осъществяваща заявка Get към Internet сървър. Кодът се поставя в отделна нишка (работи блокиращо).**

**За да стане това името на нишковата проц. да се постави в:**
***AfxBeginThread(ClientWinInetThreadProc,….);***
**на главната нишка на прил.**

```
CString g_strServerName = “Localhost”;
UINT ClientWinInetThreadProc( LPVOID pParam)
{CInternetSession session;                CHttpConnection* pConnection = NULL;
CHttpFile* pFile1=NULL;     char* buffer=new char[MAXBUFF];
UINT nBytesRead=0;
try{
   pConnection = session.GetHttpConnection( g_strServerName, 80); //TCP/IP port
   pFile1 = pConnection →OpenRequest( 1, “/”);   // GET операция
   pFile1→SendRequest();
 nBytesRead = pFile1→Read(buffer, MAXBUF – 1);
buffer[nBytesRead] = ‘\0’;
char temp[10];
if(pFile1→Read(temp, 10) != 0)            //повторен опит за четене за установяване
                    // на проблем, както и за кеширане на прочетената информация
AfxMessageBox(“Файлът надхвърля размера – файлът не е кеширан”);}
AfxMessageBox(buffer);
   }

catch( CInternetException* e)
{// обработка на изключението
e→delete;
}....
```

# Асинхронни псевдонимни (moniker) файлове

***За сваляне на големи обеми инф. от Internet в асинхронен режим, без блокиране на основната нишка:***

**- използвате Windows DLL – URLMON, който работи с WinInet и IE. Там е силния клас CAsyncMonikerFile.**

***- moniker е COM обект, създаващ псевдоним на името (URL) на истинския обект – все едно е файл с асинхронен достъп. В него е реализиран интерфейс IMoniker, чиито ф-ии асоциират четенето с памет за съхраняване на данните и интерфейс IBindStatusCallback, чиито ф-ии пък се активират по време на четене на данни от URL.***

**За щастие всичко е капсулирано в CAsyncMonikerFile : CFile, което позволява да работите като с обикновен файл.**

- ***Вместо да се отвори дисков файл, ф-ята Open() изработва IMoniker и привързва четенето с място за съхранение (IStorage).***
- ***Това което остава да направите е 1. деклариране наследник на CAsyncMonikerFile; 2. да предефинирате поведението на call back ф-ии (OnProgress, OnDataAvailable), които се викат асинхронно от Windows при случване на съответните събития.*** 3. инстанция на горния клас да се подаде като параметър при отваряне на връзка с URL

# Контроли за IP адреси

изисква IE)

**Класът е CIPAddressCtrl**

***CIPAddressCtrl:: SetAddress();***
***CIPAddressCtr::GetAddress();***
***::ClearAddress();***
***::IsBlank();***
***::SetFieldFocus(); // фокусира в избрано поле на контрола.***
***::SetFieldRange(); // ограничава диапазона на въвеждани ч-ла в поле.***

**Пример на инициализиране на контрола с IP адрес, пазен в 4 променливи по части:**

```
CIPAddressCtrl m_wndIPAddress;
BYTE m_field1, m_field2, m_field3, m_field4;.....
BOOL CMyDialog::OnInitDialog()
{     CDialog::OnInitDialog();
m_wndIPAddress.SetAddress( m_field1, m_field2, m_field3, m_field4);
return TRUE;  }
```

***контролът подава нотификационни съобщения на родителя си:***
***EN_SETFOKUS;  EN_KILLFOKUS;  EN_CHANGE;  към WM_COMMAND***
***IPN_FIELDCHANGED при промяна стойност на поле  към WM_NOTIFY.***

# Създаване примерно Web приложение (чете 30 реда от HTML документ)

**1. създавате нов AppWizard проект (MFC базиран). SDI.**
**2. създавате ресурс на диалог за въвеждане на URL (задавате ID на диалога и на edit контрола в него).**
**3. създавате диалогов клас, свързан с току що създадения диалог**
**4. задавате име на променлива, свързана с edit контрола (CString).**
**5. модифицирате менюто с цел добавяне опция Make_Connection със собствен ID.**
**6. чрез ClassWizard асоциирате меню-опцията с ваша функция (напр. OnConnect()).**
**7. добавете следния код за тази функция:**

```
CURLDlg dialog(this);
dialog.m_url = http://www.vmei.acad.bg; //текст по подразбиране
int result = dialog.DoModal();       // визуализира
if (result == IDOK)
{
CString url = dialog.m_url;          // чете въведен URL адрес
CInternetSession internetSession;
try {
   CHttpFile* httpfile = (CHttpFile*) internetsession.OpenURL(url);
   httpFile→SetReadBufferSize(4096);
  for( int x = 0; x<30; ++x)
         httpFile→ReadString(m_webPageLines[x]);
  httpFile→Close();
}
catch (CInternetException* pException) {
        ……
}
internetsession.Close();
Invalidate();} // ще визуализира прочетения текст в OnDraw()
```

**към класа – изглед добавете инициализации:**

```
for( int x = 0; x<30; ++x)
m_webPageLines[x] = “”;
```

**към метода OnDraw() на изгледа добавете:**

```
for ( int x = 0; x<30; ++x)
 pDC→TextOut(…, m_webPageLines[x]);
```

**към заглавния (.h) файл на изгледадобавете декларация:**

```
 protected: CString m_webPageLines[30];
```

## EXTENSIONS & FILTERS
## Функционални разширения, насочени към програмиране на сървъри (MFC ISAPI classes)

* позволяват създаване на DLL, който става IIS server extension
* основни MFC класове:
  - CHttpServer — 1 в рамките на сървъра; за всеки клиент се вика member ф-ия HttpExtensionProc()
  - CHttpServerContext — обектът се създава от CHttpServer по 1 за всеки thread (всяка заявка); основна ф-ия ReadClient()
  - CHttpFilter — описва какво да се случи при опреелени събития:
    - authentication schema
    - криптиране
    - log
    - trafic analysis

    1 обект за целия модул
  - CHttpFilterContext — по 1 за всеки thread (всяка заявка)