Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

Raport

La disciplina: Pogramarea în rețea

Lucrarea de laborator nr.7

Tema: “Protocolul UDP”

A efectuat: Gubenco A.

Studentul grupei TI-142

A verificat: Donos E.

Chișinău 2017

Obiective

Întelegerea mecanismului de comunicare în rețea.

Link la repozitoriu: https://github.com/GubencoAndrei/PRLaboratoare

Sarcina

Elaborarea unei aplicații client-server: ce ar utiliza Protocolul UDP pentru comunicare.

Executarea lucrării

User Datagram Protocol (sau UDP, în traducere liberă din engleză Protocolul Datagramelor Utilizator) este un protocol de comunicație pentru calculatoare ce aparține nivelului Transport (nivelul 4) al modelului standard OSI.

Împreună cu Internet Protocol (IP), acesta face posibilă livrarea mesajelor într-o rețea. Spre deosebire de protocolul TCP, UDP constituie modul de comunicație fără conexiune. Este similar cu sistemul poștal, în sensul că pachetele de informații (corespondența) sunt trimise în general fără confirmare de primire, în speranța că ele vor ajunge, fără a exista o legătură efectivă între expeditor și destinatar.Practic, UDP este un protocol ce nu oferă siguranța sosirii datelor la destinație (nu dispune de mecanisme de confirmare); totodată nu dispune nici de mecanisme de verificare a ordinii de sosire a datagramelor sau a datagramelor duplicate. UDP dispune, totusi, în formatul datagramelor, de sume de control pentru verificarea integrității datelor sau de informații privind numărul portului pentru adresarea diferitelor funcții la sursa/destinație.

Caracteristicile de baza ale UDP îl fac util pentru diferite aplicații.

- orientat către tranzacții - util în aplicații simple de tip întrebare-răspuns cum ar fi DNS.
- este simplu foarte util în aplicații de configurări, precum DHCP sau TFTP (Trivial FTP).
- lipsa întârzierilor de retransmisie îl pretează pentru aplicații în timp real ca VoIP, jocuri online.
- lucrează excelent în medii de comunicații unidirecționale precum furnizarea de informații broadcast, în servicii de descoperire (discovery services), sau în partajarea de informații către alte noduri (RIP).

Pentru implementarea funcționalității programului nostru am folosit 3 interfețe:

- IConnection
- IComand
- IGenesisUDP

Obiectele IConnection dețin informații despre fiecare conexiune la o gazdă la distanță, indiferent dacă acea gazdă este un server sau un client.

ICommand deține informații despre un singur pachet de comandă, inclusiv câmpurile opcode și date.

IGenesisUDP este clasa reală de comunicații care controlează toate funcționalitățile.

Inclusiv cu sursa sunt doua proiecte " Laborator7-UDP Chat Client " si " Laborator7-UDP Chat Server ". Acestea sunt o pereche de proiecte care implementează un sistem de chat similar IRC. Folosirea acestor proiecte ca punct de referință ar trebui să contribuie la înțelegerea clasei Genesis.

Să ne uităm la GenesisChatServer - acest proiect arată cum Genesis poate acționa ca un server pentru a servi alte instanțe ale Genesis (clienții).

În primul rând, trebuie să declarăm și să instanțiăm obiectul Genesis în aplicația gazdă (Figura 1).

```
private GenesisCore.IGenesisUDP m_UDP;

...

m_UDP = GenesisCore.InterfaceFactory.CreateGenesisUDP("ChatServer");
```

Figura 1 – Instanțiarea obiectului Genesis

Metoda GetLocalAddresses este utilizată pentru a returna o listă de adrese IP locale pe mașina curentă - și este utilizată pentru a popula o casetă combo în aplicația server de chat (Figura 2).

```
string[] addresses = m_UDP.GetLocalAddresses( );
...
```

Figura 2 – Metoda GetLocalAddresses

Pentru a face față evenimentelor de comunicare din aplicație, acestea trebuie să fie în relație conform codului din Figura 3.

```
m_UDP.OnListenStateChanged += new ListenHandler(m_UDP_OnListenStateChanged);
m_UDP.OnConnectionAuth += new ConnectionAuthHandler(m_UDP_OnConnectionAuth);
m_UDP.OnCommandReceived += new IncomingCommandHandler(m_UDP_OnCommandReceived);
m_UDP.OnConnectionStateChanged += new
   ConnectionStateChangeHandler(m_UDP_OnConnectionStateChanged);
```

Figura 3 – Relația evenimentelor

OnConnectionAuth este apelat atunci când un client a trimis informații de autorizare - acesta este locul în care clientul poate fi respins dacă, de exemplu, nu se acceptă acreditările de conectare. În exemplul serverului de chat, conexiunea este respinsă dacă porecla este prea scurtă sau dacă parola serverului este incorectă. Observați cum poate fi trimis un motiv de respingere clientului. Totul este controlat prin modificarea obiectului ConnectionAuthEventArgs trimis cu evenimentul. Comanda care conține informațiile de autorizare poate fi accesată de proprietatea AuthCommand a argumentelor evenimentului (Figura 4).

```
private void OnConnectionAuth(object o, ConnectionAuthEventArgs e)
{
    ...

    if(e.AuthCommand.Fields[1].Length < 3)
      {
        e.AllowConnection = false;
        e.DisallowReason = "Nickname too short.";
        return;
      }
      else if(e.AuthCommand.Fields[1].Length > 15)
      {
        e.AllowConnection = false;
        e.DisallowReason = "Nickname too long.";
        return;
        }

    ...
 }
```

Figura 4 – Metoda onConnectionAuth

Să ne uităm acum la partea clientului proiectului de chat, "GenesisChatClient". Acest lucru este similar cu aplicația serverului prin faptul că instanțiază o instanță Genesis și conectează diverse evenimente (Figura 5).

```
m_UDP.OnListenStateChanged += new ListenHandler(m_UDP_OnListenStateChanged);
m_UDP.OnLoginRequested += new SendLoginHandler(m_UDP_OnLoginRequested);
m_UDP.OnAuthFeedback += new AuthenticatedHandler(m_UDP_OnAuthFeedback);
m_UDP.OnConnectionStateChanged += new
  ConnectionStateChangeHandler(m_UDP_OnConnectionStateChanged);
m_UDP.OnCommandReceived += new IncomingCommandHandler(m_UDP_OnCommandReceived);
m_UDP.OnConnectionAuth += new ConnectionAuthHandler(m_UDP_OnConnectionAuth);
m_UDP.OnSocketError += new SocketErrorHandler(m_UDP_OnSocketError);
m_UDP.OnConnectionRequestTimedOut += new
  RequestTimedOutHandler(m_UDP_OnConnectionRequestTimedOut);
```

Figura 5 – Clientul Chat

OnLoginRequested este declanșat când serverul solicită datele de conectare de la client. Clientul trebuie să trimită un pachet de comandă înapoi la server cu opcode OPCODE_LOGINDETAILS și câmpurile de date corespunzătoare. În proba de chat, acest pachet conține porecla utilizatorului și parola serverului (Figura 6).

```
private void OnLoginRequested(object o, LoginSendEventArgs e)
{
    if(e.Connected)
    {
        e.ServerConnection.SendUnreliableCommand(0,
               GenesisConsts.OPCODE_LOGINDETAILS,
               new string[] {txtServerPW.Text, txtNickName.Text} );
        spState.Text = "Sending login details...";
    }
    else
    {
        spState.Text = "Connection rejected - " + e.Reason;
    }
```

Figura 6 – Metoda OnLoginRequested

OnConnectionAuth este declanșat atunci când o gazdă de la distanță încearcă să se conecteze la Genesis.Clientul de chat nu poate accepta conexiuni (deoarece acționează ca un client), deci o mică bucată de cod este introdusă aici pentru a respinge conexiunea și pentru a trimite un motiv de respingere înapoi. Dacă evenimentul nu a fost atins de aplicația client, conexiunea va fi în continuare respinsă (Figura 7).

```
private void m_UDP_OnConnectionAuth(object o, ConnectionAuthEventArgs e)
{
    //Clients don't accept connections.
    e.AllowConnection = false;
    e.DisallowReason = "Can't connect directly to a chat client";
}
```

Figura 7 - MetodaOnConnectionAuth

Un lucru important este modul în care conexiunile sunt stabilite de la client sunt arătate în Figura 8.

```
private void btnConnect_Click(object sender, System.EventArgs e)
{
    server_ip = txtServerIP.Text;
    m_UDP.RequestConnect(ref server_ip,
       Convert.ToInt32(txtServerPort.Text),
       out server_req_id);
    spState.Text = "Connecting...";
    btnConnect.Enabled = true;
}
```

Figura 8 – Conexiunile de stabilitate

Metoda RequestConnect se ocupă de inițierea conexiunii. Observați că parametrul IP al serverului este referitor; Acest lucru se datorează faptului că este posibil ca metoda să fie trecută printr-un nume de gazdă (mai degrabă decât o adresă IP), dar IP-ul va fi rezolvat, iar șirul actual va fi schimbat la adresa IP.Aplicația împachetează această funcție într-o metodă foarte ușor de utilizat. Două dintre metode se află în obiectul Connection, acestea sunt prezentate în Figura 9.

```
int SendUnreliableCommand(byte flags, string opcode, string[] fields);
int SendReliableCommand(byte flags, string opcode, string[] fields);
```

Figura 9 – Metodele Request

Aceste două metode permit trimiterea unui singur pachet de comandă către gazda de la distanță asociată cu obiectul IConnection. Câmpurile opcode și câmpurile simple pot fi orice date pe care aplicația gazdă le recunoaște, valori valide pentru drapele sunt indicate în Figura 10. (de la Constants.cs):

```
public static byte FLAGS_NONE = 0;
public static byte FLAGS_CONNECTIONLESS = 1;
public static byte FLAGS_ENCRYPTED = 2;
public static byte FLAGS_COMPOUNDPIECE = 4;
public static byte FLAGS_COMPOUNDEND = 8;
public static byte FLAGS_RELIABLE = 16;
public static byte FLAGS_SEQUENCED = 32;
```

Figura 10 – Recunoașterea datelor de intrare

Dintre steagurile disponibile în fișierul Constants.cs, numai unul ar trebui să fie folosit manual în apelurile metodice, și anume FLAGS_SEQUENCED, care aplică o secvențiere pachetelor nesigure. Celelalte steaguri sunt setate automat de Genesis și nu trebuie modificate de aplicația gazdă.Este posibil să transmiteți un pachet de comandă către mai multe gazde de la distanță folosind următoarele metode în interfața IGenesisUDP (Figura 11).

```
int SendUnreliableCommandToAll(BroadcastFilter filter,
                  byte flags, string opcode, string[] fields);
int SendReliableCommandToAll(BroadcastFilter filter,
                  byte flags, string opcode, string[] fields);
```

Figura 11 – Metodele interfeței IGenesisUDP

Aceste i-au steaguri de filtru difuzate, care sunt definite în Constants.cs (Figura 12).

```
[Flags]
public enum BroadcastFilter : int
{
    None     = 0,  //Filter out everything
    Servers  = 1,  //Send to servers we are connected to.
    Clients  = 2,  //Send to clients connected to us.
    All      = Servers | Clients,
                  //Send to both servers
                  //and clients (every connection).
    AuthedOnly = 4,//Only send to authed clients or servers we are authed with.
}
```

Figura 12 – Steagurile definite în Constants

Rezultate

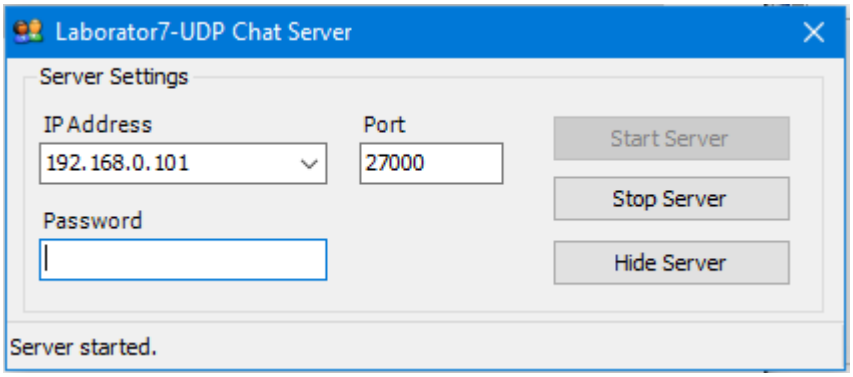

Figura 13 – UDP Chat Server

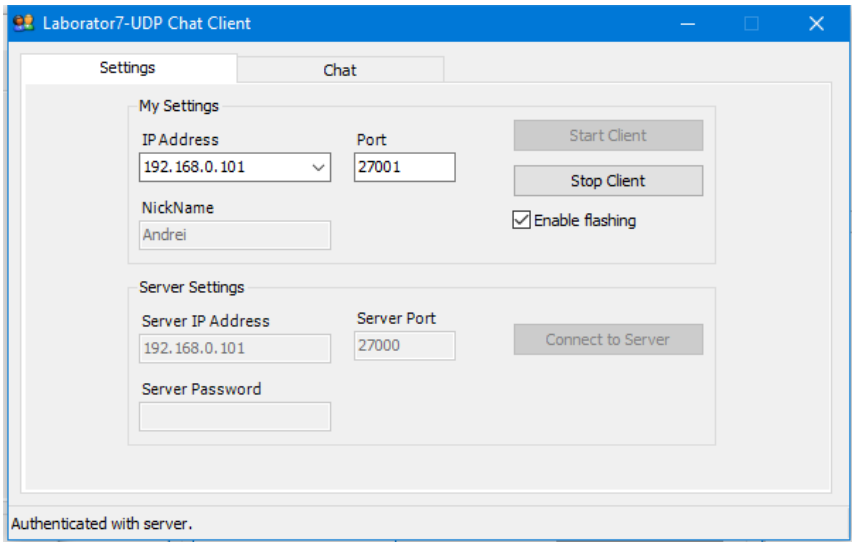

Figura 14 – UDP Chat Client 1

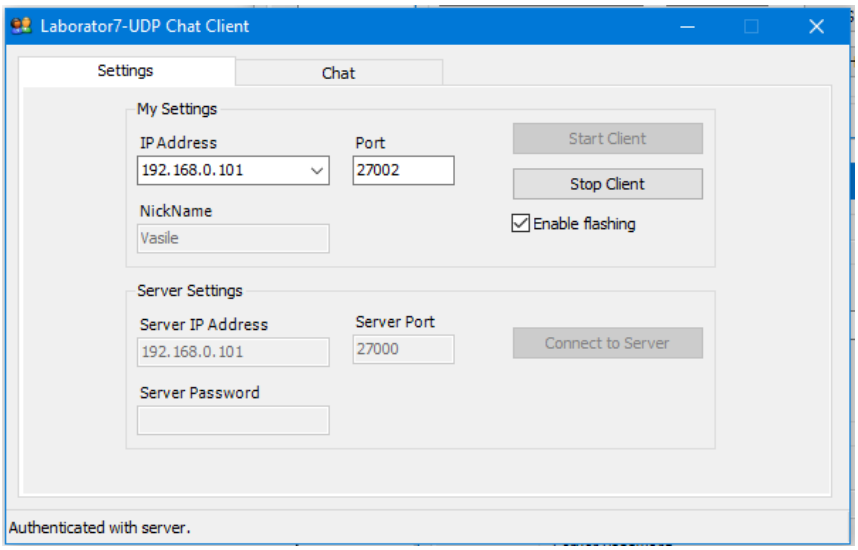

Figura 15 – UDP Chat Client 2

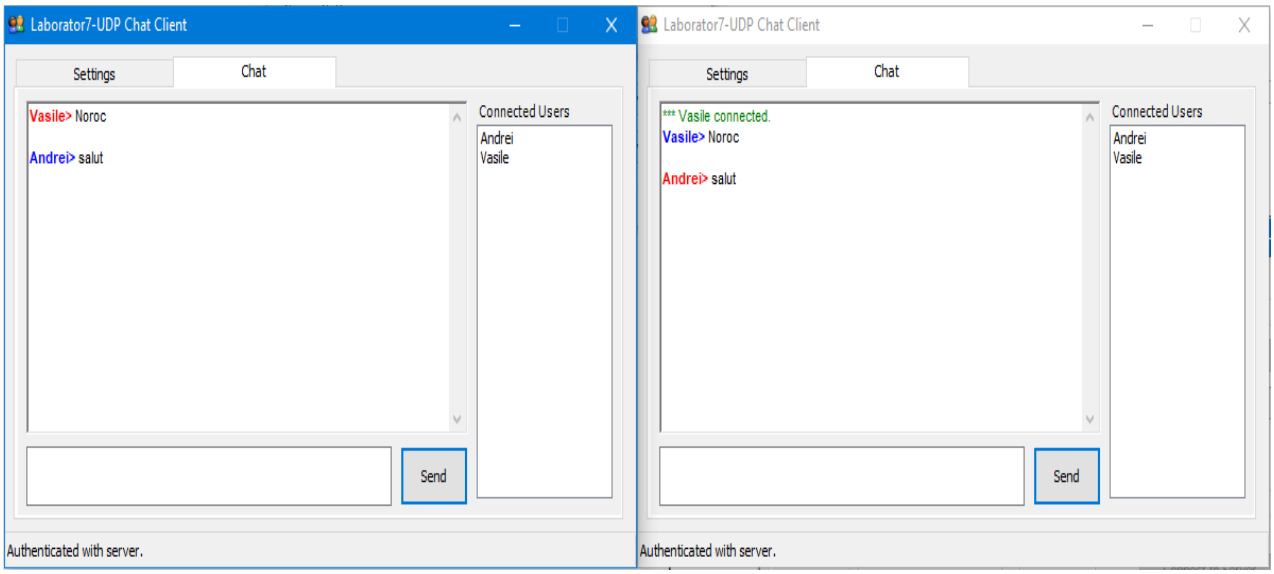

Figura 16 – Comunicarea între cliente

Concluzii

În urma realizării acestei lucrări de laborator am studiat noi abordări pentru comunicare în spațiul Web prin intermediul UDP. Pentru a realiza sarcina propusă am creat o aplicație client-server care reprezintă un chat simplu cu posibilitatea de conectare, deconectare și trimiterea mesajelor către toți utilzatorii chatului conectați la server.