SPŠ Elektrotechnická, K. Adlera 5, Bratislava

Klient pre textový chat

Konzultant: Ing. Vladimír Klimovský

Ivan Erben, 4.E

Školský rok: 2000/2001

Poďakovanie

Chcel by som sa poďakovať môjmu konzultantovi, Ing. Vladimírovi Klimovskému za pomoc pri realizácii projektu, Ralfovi Brownovi za jeho Interrupt list, Kos van der Houtovi za informácie, ako zisťovať adresy sietí.

Obsah

I. Úvod

Keď prišiel čas výberu témy na praktickú maturitnú skúšku, spomenul som si na svoje pokusy o prenos dát pomocou protokolu IPX a snahu vytvoriť veľmi jednoduchý chatovací program. Chatovací program umožňuje komunikáciu po sieti prostredníctvom textových správ. To, čo sa mi vtedy nepodarilo realizovať, som sa rozhodol spraviť na praktickú maturitu.

Mojím cieľom bolo vytvoriť program, ktorý by pracoval aj v sieti s viacerými segmentami, obsahoval zoznam užívateľov, ktorí ho majú spustený a pracoval aj na starších počítačoch s OS MS-DOS.

Dúfam, že používatelia zažijú veľa zábavy pri chatovaní.

II. Metodika práce

1. Teória o protokole IPX

IPX protokol je nespájaným typom protokolu, tzv. datagramový protokol. Znamená to, že na prenos údajov netreba vytvárať žiadne virtuálne "spojenia". Pracuje na sieťovej vrstve, údaje sa prenášajú ako pakety. Každý paket obsahuje hlavičku a správu (dáta ktoré chceme preniesť). V hlavičke sa nachádzajú informácie nevyhnutné pre prenos paketu zo zdroja do cieľa.

1.1. Hlavička IPX paketu

Maximálna veľkosť paketu je 576 bajtov, z toho hlavičku tvorí prvých 30 bajtov. Zvyšnú časť môžu tvoriť užívateľské dáta.

Tab. 1 – štruktúra IPX hlavičky

Pozícia	Pole	Typ	Popis
0	kontrolný súčet	WORD	nepoužíva sa, používa sa hardwarový
2	dĺžka	WORD	veľkosť paketu
4	riadenie prenosu	BYTE	obsahuje informácie pre smerovanie paketu
5	typ paketu	BYTE	identifikuje službu spojenú s paketom, malo by byť 0 alebo 4
6	adresa cieľa	ADRESA	adresa stanice, ktorej bude paket doručený
18	adresa zdroja	ADRESA	adresa stanice, ktorá paket odoslala

ADRESA je pole veľkosti 12 bajtov a má nasledovnú štruktúru:

Tab. 2 – štruktúra IPX adresy

Pozícia	Pole	Typ	Popis
0	adresa siete	BYTE[4]	4 bajtová adresa určujúca segment siete
4	adresa uzla	BYTE[6]	MAC adresa zariadenia
10	socket	WORD	číslo soketu

V programe (v unite ipxsmall.pas) je definované:

```
netAddr  = array[1..4] of byte;    { adresa siete }
nodeAddr = array[1..6] of byte;    { adresa uzla v sieti}

netAddress = record
                Net    : netAddr;   { adresa siete}
                Node   : nodeAddr;  { adresa uzla }
                Socket : word;      { číslo soketu}
             end;

IPXheader = record
              check  : word;           { checksum }
              length : word;           { dĺžka v bajtoch }
              tc     : byte;           { riadenie prenosu }
              pType  : byte;           { typ paketu }
              dest   : netAddress;     { adresa ciela }
              src    : netAddress;     { adresa zdroja }
            end;
```

1.2. Event Control Block

Rozhranie medzi IPX a aplikáciou tvorí Event Control Block (ECB). ECB obsahuje informácie, ktoré potrebuje IPX na odoslanie a prijatie paketu, prispôsobené potrebám aplikácie.

Tab. 3 – štruktúra Event Control Block (ECB)

Pozícia	Pole	Typ	Popis
0	spojovacia adresa	DWORD	pre vnútorné potreby IPX
4	adresa ESR	DWORD	adresa Event Service Routine (ESR), ktorá sa vykoná po prijatí/odoslaní paketu
8	príznak použitia	BYTE	obsahuje nenulovú hodnotu ak IPX pracuje s ECB
9	ukončovací kód	BYTE	po spracovaní udalosti oznamuje úspech alebo číslo chyby
10	socket	WORD	číslo soketu
12	pracovný priestor IPX	BYTE[4]	pre vnútorné potreby IPX
16	pracovný priekor ovládača	BYTE[12]	pre vnútorné potreby IPX
28	adresa najbližšieho uzla	BYTE[6]	obsahuje adresu posledného uzla, ktorým paket prešiel
34	počet fragmentov	WORD	počet bufferov z ktorých pozostáva paket
36	adresa fragmentu č. 1	DWORD	obsahuje adresu fragmentu č. 1
40	veľkosť fragmentu č. 1	WORD	obsahuje veľkosť fragmentu č. 1

Ak teda máme zadefinované horeuvedené štruktúry, môžeme po ich inicializovaní volať jednotlivé funkcie IPX drivera, ako napr. IPX send paket. Samotné volanie funkcií je veľmi jednoduché, pozri príklad.

```
procedure IPXsendPacket(var E : ECBtype);
var
 regs : registers;

begin
 regs.bx:=$0003;     { číslo funkcie ipx drivera, 3 - odošli paket }
 regs.es:=seg(E);    { adresa platného-inicializovaného }
 regs.SI:=ofs(E);    { Event Control Blocku }
 IPXCall(regs);      { volanie ipx drivera }
end;
```

2. RIP Protokol

Smerovače používajú RIP protokol na vzájomné informovanie sa o tom, aké siete sú prístupné. Každý smerovač si vytvára a udržiava dynamickú databázu na medzisieťové smerovanie. Táto databáza je vlastne sieť z pohľadu smerovača. Cez smerovač sú prístupné len tie segmenty siete, ktoré sa nachádzajú v jeho databáze.

2. 1. Štruktúra RIP paketu

RIP paket je rozšírením štandardtného IPX paketu, takým spôsobom, že v oblasti pre údaje sa nachádza nasledovná štruktúra (tabuľka 4).

Tab. 4 – štruktúra RIP paketu

Záznam	Pole	Veľkosť
	Operácia	BYTE[2]
1	Adresa siete	BYTE[4]
	Počet uzlov	BYTE[2]
	Oneskorenie	BYTE[2]
.	.	.
.	.	.
.	.	.
n	Adresa siete	BYTE[4]
	Počet uzlov	BYTE[2]
	Oneskorenie	BYTE[2]

Pole operácia určuje, či sa jedná o požiadavku (hodnota 1), alebo odpoveď (hodnota 2). Nasleduje jedna alebo viac skupín (n) záznamov, pričom každý obsahuje adresu segmentu siete, počet uzlov, ktorými paket prejde, aby dosiahol túto sieť a čas aký to potrvá. Čas je udávaný v "tikoch", pričom jeden "tik" znamená 1/18 sekundy.
Ak sa jedná o požiadavku, používa sa iba pole adresa siete v zázname 1, ostatné polia musia byť vynulované.

2. 2. Služby RIP protokolu

RIP protokol umožňuje nasledovnú výmenu informácií:

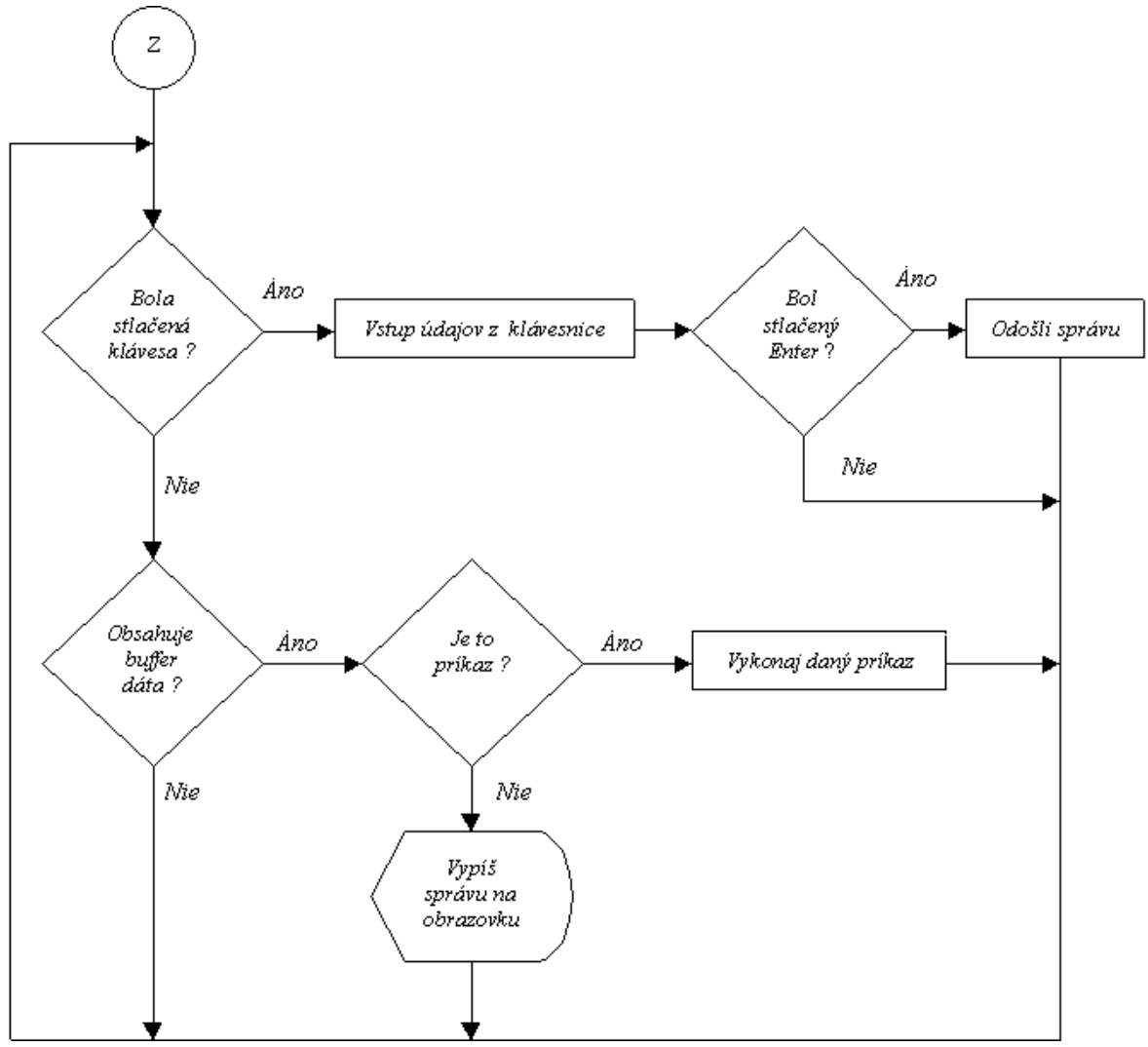
- Pracovné stanice si zistia najrýchlejšiu cestu do určitého segmentu siete vyslaním smerovacej požiadavky.
- Smerovače si upravujú svoje smerovacie tabuľky podľa odpovedí iných smerovačov.
- Smerovače odpovedajú na požiadavky od ostatných smerovačov a aj na požiadavky od pracovných staníc.
- Smerovače vysielajú svoje tabuľky každých 60 sekúnd, alebo keď zistia zmenu v konfigurácii siete.

III. Praktická realizácia

1. Princíp práce programu

Hlavná časť behu programu sa odohráva v hlavnom cykle, kde sa testuje, či nejaký z troch bufferov obsahuje prijaté dáta. Ak áno, treba tieto dáta spracovať. Zisťuje sa či je to príkaz alebo iba obyčajná správa. Správa sa vypíše na obrazovku, príkazy sa musia vykonať. Ak bola stlačená klávesa, načíta sa jej hodnota. Text, ktorý používateľ napísal sa odošle ako správa po stlačení klívesy Enter. Správy sa posielajú ako broadcast pakety do všetkých známych segmentov siete.

V hlavnom cykle sa zároveň pri štarte kontroluje, či zadaná prezývka už existuje a každú minútu sa vytvára nový zoznam užívateľov programu. Posledné dve menované veci sa vo vývojovom diagrame pre zjednodušenie nenachádzajú.



2. Časti programu, ich tvorba a problémy

Z dôvodu lepšej prehľadnosti a zrozumiteľnosti je program rozdelený na viac častí. Každá časť sa nachádza v samostatnom súbore – unite.

Program sa skladá z nasledovných častí:

- štruktúry a funkcie IPX – ipxsmall.pas
- zisťovanie segmentov sietí pomocou protokolu RIP – rip_unit.pas
- definície príkazov, vykreslenie užívateľského rozhrania – imain.pas
- samotný program – ichat.pas

2. 1. Unit Ipxsmall.pas

Zdrojový kód som získal z internetu, autor nie je známy. Upravil som ho pre potreby môjho programu, zmenil som volanie ipx drivera z volania prerušenia na novší spôsob použitím vstupného bodu. Volanie cez prerušenie je totiž iba na zachovanie kompatibility a od verzie Netwaru 2.0a by sa nemalo používať.

2. 2. Unit Imain.pas

V tejto jednotke sa nachádzajú hlavne funkcie na vykresľovanie užívateľského rozhrania, sú tu zadefinované hlavné premenné a konštanty. Užívateľské rozhranie je vytvorené z ASCI znakov, ktoré sú vykreslené v cykloch (pozri grafickú prílohu).
Je tu definovaný základný typ Paket, ktorý obsahuje Event Control Block, IPX hlavičku a miesto pre dáta.

2. 3. Unit Rip_unit.pas

Táto jednotka obsahuje jedinú funkciu, Get_Networks. Pomocou tejto funkcie zistíme všetky dostupné segmenty siete. Obsahuje cyklus, v ktorom sa každú sekundu posiela požiadavka na zistenie všetkých dostupných sietí. Cyklus sa ukončí po prijatí odpovede, uplynutí 3 sekúnd, prípadne po stlačení klávesy ESC.

Po vytvorení tejto jednotky a jej zaradení do programu, som narazil na problém. Odoslané správy sa nezobrazovali na obrazovku (nič nebolo prijaté späť). Problém bol v tom, že zoznam segmentov sietí, ktoré nám smerovač vráti, neobsahuje "náš" segment siete. O tejto skutočnosti nie je v dokumentácii ani zmienka. Riešenie je však veľmi jednoduché, pretože adresu lokálneho segmentu poznáme (obsahuje ju premenná localAddr.net) a nie je problém ju do zoznamu pridať.

2. 4. Samotný program – ichat.pas

Program som tvoril tak, že najprv som vytvoril hlavný cyklus, v ktorom sa odohráva najväčšia časť behu programu. Na odoslanie správy sa používa socket 7777h, na prijímanie socket 6666h. Použiť dva sockety namiesto jedného som sa rozhodol z toho dôvodu, že pri

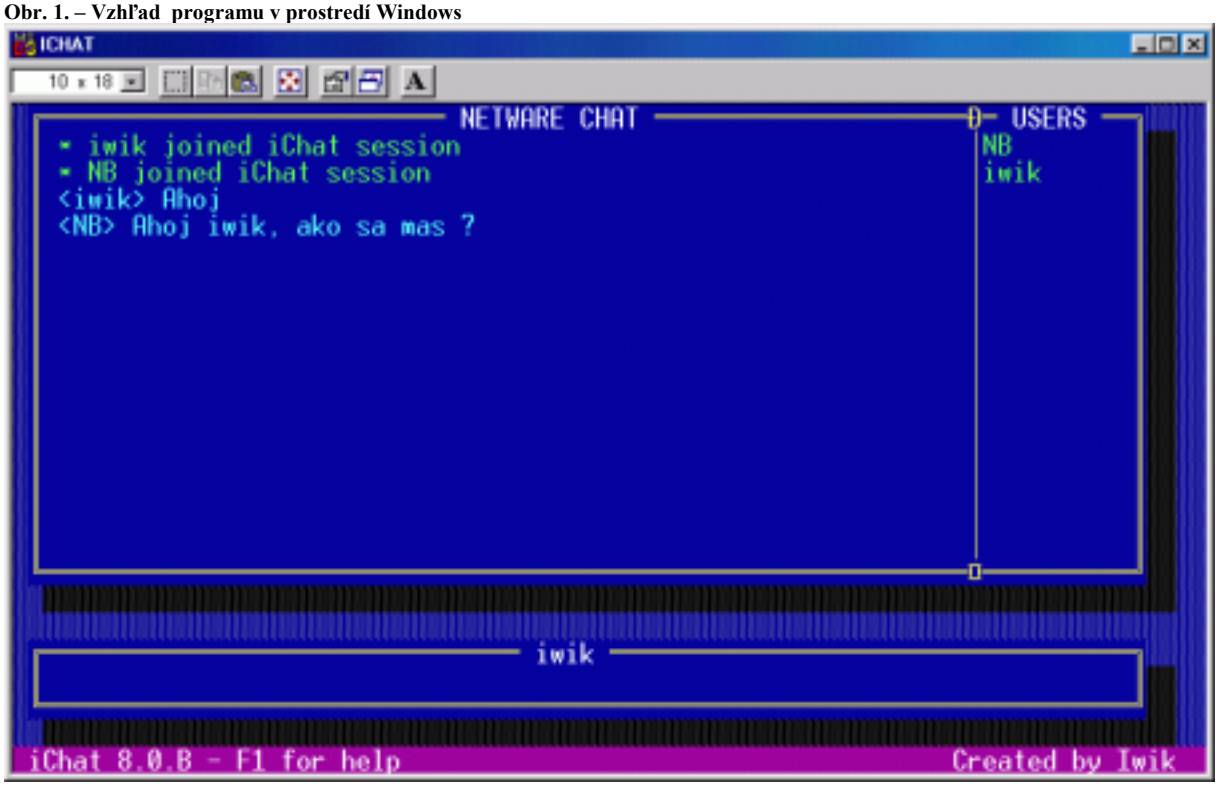
vytváraní zoznamu užívateľov sa vysiela požiadavka a následne sa prijíma rýchlo za sebou viac odpovedí. Pri použití jedného socketu sa mi stávalo, že niektoré odpovede neboli prijaté.

Zoznam aktívnych užívateľov som realizoval ako StringCollection. Je to teda dynamický objekt, čo prináša niekoľko výhod, ako napr. to že všetky položky si "sám" usporiada podľa abecedy. Vždy po pripojení sa nového užívateľa do chatu alebo po uplynutí jednej minúty sa vyšle všetkým požiadavka na zistenie užívateľov - Urequest. Každý, kto ju prijme, vyšle opäť všetkým odpoveď Uadd so svojou prezývkou, ktorá znamená, že príjemca si má pridať prezývku do zoznamu. Ak sa prezývka v zozname už nachádza nepridáva sa do zoznamu, pretože StringColection dovoluje zakázať duplicitné položky.

IV. Prílohy

1. Grafická príloha

Obr. 1. – Vzhľad programu v prostredí Windows



Poznámka: "Chybné zobrazenie" dvoch znakov, na začiatku a konci rozdeľovacej čiary je spôsobené nadstavenou kódovou stránkou 852 (česko-slovenská). Pri zobrazení na celú obrazovku nenastáva.

2. Výpis programu

Z priestorových dôvodov tu uvádzam len výpis hlavného programu. Ostatné časti sú v elektronickej forme.

```
Program Ichat;

{$F+,R-}

Uses Dos,Crt,Ipxsmall,Imain,Rip_unit,Objects;

var
  done           : boolean; { urcuje ci sa skonci hlavny cyklus a tym aj program }
  k              : char;    { premenna na vstup z klavesnice }
  count          : byte;    { urcuje pocet sieti }
  Networks       : T_rip;   { zname segmenty siete }
  h,m,s,ms       : word;    { premenne na ulozenie casu }
  oldm,olds      : word;    { premenne na ulozenie casu }
  check_nick     : boolean; { ak true, program neopoveda na poziadavky o zoznam }
                     { uzitatelov-ked sa zistuje ci takato prezyvka uz existuje }

procedure SendLine(linex : string); { posle riadok }
begin
if count=0 then count:=1; { ak niesu zistene ziadne siete, tak }
send.data:=linex;         { budeme posielat vsetko do lokalnej }
for i:=1 to count do      { <= posleme to raz do kazdej siete }
 begin
   { nadstavime adresu cielovej podsiete siete zo zoznamu }
   send.ipx.dest.net:=Networks.rip_data[i].netnumber;
   send.ipx.dest.socket:=swap(RecvSocket);
   { zistime si pre tuto adresu najblizsi smerovac }
   if IPXGetLocalTarget(send.ipx.dest.net,send.ipx.dest.node,
     send.ipx.dest.socket,send.ecb.immedAddr)<>0
     then ShowLine('--IPX: No route to host.'); { chyba, neexistuje ziadna }
   repeat                                       { cesta do tejto podsiete }
   until ((send.ecb.inuse=0) and (send.ecb.complete=0));
   IPXsendPacket(send.ecb);
  end;
end;

{ spracuje riadok.. ak su tam dake riadiace symboly, vykona dane prikazy.. }
procedure ProcessLine(str : string);
var frmw, cmd, temp : string;
begin
  temp:=str;
  frmw:=Copy(str,2,Pos(' ',str)-3);
  Delete(str,1,Pos(' ',str));
  if str[1]='/' then { je to daky prikaz... }
  begin
    cmd:=UpWord(Copy(str,1,Pos(' ',str)-1));
    if ((cmd=Msg) or (cmd=Msg2)) and (Is4You(str)) then { je to sprava pre nas }
    begin
      Delete(str,1,Pos(' ',str));
      if Pos(' ',str)<>0 then
      begin
        Delete(str,1,Pos(' ',str));  { ostane nam IBA sprava... }
        TextColor(MsgFgC);
        TextBackground(MsgBgC);
        ShowLine('['+frmw+']: '+str);
        TextColor(NormFg);
        TextBackGround(NormBg);
      end;
    end;
```

```
    if (cmd=WhoIS) and Is4You(str) then { je to prikaz na zistenie kto si }
       SendLine('<whois> /M '+frmw+' '+nick+ ' is '+MyUserName);

    if (cmd=Urequest) then  { prijata poziadavka na zistenie uzivatelov }
       begin
       if check_nick then exit;
       if (frmw=nick) then { poziadavka odomna, seba tam dam "napevno" }
         begin
           c^.insert(NewStr(nick));
           UpdateUserWindow;
         end;
       SendLine('<'+nick+'> '+Uadd+' ');
       end;

    if (cmd=Uadd) then {prida user-a do zoznamu }
       begin
       c^.insert(NewStr(frmw));
       UpdateUserWindow;
       end;

    if (cmd=UNew) then {prida user-a do zoznamu a vypise ze sa pripojil}
       begin
       c^.insert(NewStr(frmw));
       UpdateUserWindow;
       TextColor(SpcCol);
       ShowLine('* '+frmw+' joined iChat session ');
       end;

    if (cmd=Urem) then { odstrani usera zo zoznamu }
       begin
        if -1<>c^.indexOf(NewStr(frmw)) then c^.Free(NewStr(frmw));
        UpdateUserWindow;
        TextColor(SpcCol);
        ShowLine('* '+frmw+' left iChat session ');
       end;
   end
   else
   begin
     case temp[1] of
      '*': color:=SpcCol;
      '-': color:=NoticeCol;
      'ţ': color:=MeColor;
     else color:=NormFg;
     end;
     Textcolor(Color); { spravna farba podla typu spravy }
     Writeln(temp);
   end;
end;

function FiltLines(var str : string) : string;
var cmd : string;
begin
  FiltLines:='';
  cmd:=UpWord(Copy(str,1,Pos(' ',str)-1)); { mozno to je prikaz s parametrom }
  if cmd='' then cmd:=UpWord(str); { mozno bez parametra }

  if (cmd=Help) then
  begin
    str:='';  { nic nebudeme posielat prec... }
    ShowHelp; { ...ale zobrazi sa help }
  end;

  if (cmd=Me) or (cmd=Notice) then Delete(str,1,Pos(' ',str));
  if (cmd=Me) then FiltLines:='ţ ['+nick+'] '+str
  else
```

```
  if (cmd=Notice) then FiltLines:='-- '+str
  else

  if (str<>'') then
   FiltLines:='<'+nick+'> '+str;
end;

procedure ProcessPacket(var prijaty:packet);
begin
  Window(4,2,64,17);     { aktivuje hlavne okno }
  textBackground(Blue);
  textColor(NormFg);
  Gotoxy(1,y);           { nadstavi kurzor na spravny riadok }
  ProcessLine(prijaty.data);  { spracujeme riadok }
  y:=WhereY;             { ulozime poziciu kurzoru }
  Window(4,22,74,22);    { aktiny je teraz editovacie okno }
  prijaty.data:='';      { vyprazdime buffer }
  IPXListenForPacket(prijaty.ecb) { cakame na dalsi paket }
end;

procedure Main;
begin
  new(c,Init(10,30));  { vytvorime novu PstringCollection na zoznam uzivatelov }
  check_nick:=True;  { pri starte programu sa bude kontrolovat nick }
  ClrScr;
  InitScreen;
  WindowBar;
  y:=1;
  SendLine('<'+nick+'> '+URequest+' '); { zistujeme uzivatelov }
  line:='';
  done:=FALSE;
  GetTime(h,oldm,olds,ms);

  repeat
    repeat
    ImIdle;
    GetTime(h,m,s,ms);
    if oldm=59 then oldm:=0;
    if olds=59 then olds:=0;

    if (check_nick) and (olds+1=s) then
     begin
      for ai:=0 to c^.count-1 do
      begin
       a:=c^.at(ai);
       if a^=nick then begin done:=True; exit; end;
      end;
     check_nick:=False;
     olds:=s;
     SendLine('<'+nick+'> '+Unew+' ');
     end;

    if oldm+1=m then { po 1 minute zrus zoznam uzivatelov, vytvori sa novy }
     begin
       c^.FreeAll;
       SendLine('<'+nick+'> '+URequest+' ');
       oldm:=m;
     end;
    until done or KeyPressed or
         (receive1.data<>'') or (receive2.data<>'') or (receive3.data<>'');
    if KeyPressed then begin
       k:=ReadKey;
         case k of
          #13 : if (line<>'') then
                  begin
```

```
                    if check_nick then exit;
                    line:=FiltLines(line);
                    if line<>'' then
                    SendLine(line);line:='';
                  end;
          #8  : if length(line)>0 then
                    line:=copy(line,1,length(line)-1);
          #0  : begin
                    k:=ReadKey;
                    if k=#59 then ShowHelp;
                end;
          #27 : done:=TRUE;
        else
          if length(line)<70 then
            line:=line+k
          else
            begin
              sound(1000);
              delay(100);
              noSound;
            end;
        end;
        TextColor(White);
        TextBackground(LightBlue);
        Window(4,22,74,22);
        GotoXY(1,1);ClrEol;Write(Line);
      end;

      if receive1.data<>'' then ProcessPacket(receive1);
      if receive2.data<>'' then ProcessPacket(receive2);
      if receive3.data<>'' then ProcessPacket(receive3);

  until done;
  { odchadzame a dame to vediet ostatnym }
  if not check_nick then SendLine('<'+nick+'> '+Urem+' ');
end;

var Result:Integer;

begin
  TextAttr:=7;
  ClrScr;

  if (UpWord(Paramstr(1))='-?') or (UpWord(Paramstr(1))='-HELP') or
     (UpWord(Paramstr(1))='/?') or (UpWord(Paramstr(1))='/HELP')
     then Usage;

  InitIPX;
  GetNick;
  Get_Networks(networks,count);

  { Najprv skusime sockety zavriet, ked su zavrete nic sa nestane.
   Je to z toho dôvodu ze vo windows niektori uzivatelia maju zvyk
   programy  pre DOS ukoncovat ako programy pre windows, kliknutim
   ikony zavriet program. Toto vsak nie je korektny sposob, DOS program
   je "surovo" ukonceny a v mojom pripade ostanu sokety na posielanie
   a primanie otvorene. To pri opatovnom spusteni programu sposobi chybu
   pri pokuse ovorit socket - chyba 255 soket je uz otvoreny }

  IPXcloseSocket(SendSocket);
  IPXcloseSocket(RecvSocket);

  Result:=IPXopenSocket(0,SendSocket) or IPXopenSocket(0,RecvSocket);
  if Result=0 then
    begin
```

```
      with send do
        InitSendPacket(ecb,ipx,sizeof(String),SendSocket);
      with receive1 do
        InitReceivePacket(ecb,ipx,sizeof(String),RecvSocket);

      with receive2 do
        InitReceivePacket(ecb,ipx,sizeof(String),RecvSocket);

      Main;
      delay(200);
      IPXcloseSocket(SendSocket);
      IPXcloseSocket(RecvSocket);
      dispose(c); { este uvolnit z pamate zoznam uzivatelov }
    end
    else
     begin
        clrscr;
        Writeln(Version);
        Write('Error Opening Socket ',Dec2Hex(SendSocket),'h or
              ',Dec2Hex(RecvSocket),'h  - ');
       case Result of
         $fe  : Writeln('socket table full');
         $ff  : Writeln('socket already open');
          else   Writeln('unknown error ',Dec2Hex(result),'h');
        end;
        Write(#13#10,'Press any key...');
        readkey;
        halt(3);
     end;

  TextColor(LightGray);
  TextBackground(Black);
  window(1,1,80,25);
  clrScr;
  Writeln(Version,' done. (c) 1999-2001 Iwik'#13#10);
  if not check_nick then
  begin
   Writeln('Visit http://www.iwik.sk for new version');
   Writeln('Comments, suggestions, etc. write to iwik@iwik.sk'#13#10);
  end
  else begin { prezyvka uz existuje, ukoncenie }
   Writeln(Version);
   TextColor(White);
   Writeln('Aborted: Nick name "'+nick+'" already in use!');
   TextColor(7);
   Writeln(#13#10'Press any key...'#07);
   delay(500);
   readkey;
  end;

end.
```

3. Užívateľská príručka

Program ichat je textový chatovací program pre LAN siete s protokolom IPX.
Pre používanie je nutnou podmienkou prítomnosť ipx drivera.

Voľba prezývky
Po štarte si program vyžiada Vašu prezývku, pod ktorou budete vystupovať.
Maximálna dĺžka prezývky je 10 znakov.
Prezývku môžete taktiež zadať ako parameter príkazového riadku, napr. ak chcem vystupovať pod prezývkou iwik spustím program takto:
ichat.exe iwik

Užívateľské rozhranie
Užívateľské rozhranie obsahuje dve samostatné okná. Hlavné okno slúži na zobrazovanie správ, spodné okno je editačné, sem píšete vašu správu alebo príkazy.
Hlavné okno má dve časti, v ľavej sa zobrazujú správy, pravá obsahuje zoznam aktívnych používateľov, t.j. všetkých, ktorí majú spustený ichat na svojom počítači.

Príkazy
Príkazy sa zadávajú do editačného okna, t.j. ako text. Každý príkaz sa zadáva na začiatku s lomítkom (/). Na veľkosti písmen nezáleží.

Zoznam príkazov
/m *komu správa* - privátna správa *správa* pre užívateľa *komu*, to isté ako /msg
/msg *komu správa* - privátna správa *správa* pre užívateľa *komu*, to isté ako /m
/whois *kto* - vypíše užívateľské meno (username) pre prezývku *kto*
/notice *poznámka* - vypíše text *poznámka* ako poznámku
/me *správa* - upraví text *správa* do tvaru [prezývka] *správa*

Poznámka: Príkaz /whois pracuje iba na sieťach typu Novell.

Príklad 1:
Chceme vedieť, kto používa prezývku iwik, tak napíšeme:
/whois iwik
zobrazí sa nám napr.
[whois] iwik is 0857.Ostud4.Ostud.Spse

Príklad 2:
Chceme poslať súkromnú správu prezývke iwik, tak napíšeme:
/m iwik Ten ichat je super vec :)

V. Diskusia

V tejto časti by som chcel poukázať, aké mne známe možnosti riešenia zadania som mal na výber.

Spôsoby, ako umožniť užívateľom posielať privátne správy sú dva. Prvý spôsob je poslať správu naozaj len tomu, komu je určená. To znamená paket poslať nie ako broadcast, ale presne s adresou používateľa, ktorému je určená. To si však vyžaduje dodatočnú komunikáciu ohľadom zistenia adresy daného používateľa. Jednoduchší, ale na druhej strane menej bezpečný spôsob je poslať privátnu správu všetkým (ako bežnú správu), a potom v programe kontrolovať podľa prezývky, či je správa určená danému používateľovi. Ak áno, tak sa správa zobrazí, ak nie, správa sa nevypisuje. Pre minimálne zaťažovanie siete som sa rozhodol pre druhý jednoduchší spôsob.

Na realizáciu zoznamu užívateľov som mal tiež dve možnosti. Vytvoriť klasické statické reťazcové pole, alebo ako objekt StringCollection. Použite StringCollection prináša niekoľko výhod:

- nemusíme dopredu poznať počet prvkov
- počet prvkov je limitovaný iba veľkosťou voľnej pamäte
- automatické triedenie prvkov podľa abecedy
- automatická kontrola duplikátnych položiek

Tieto možnosti veľmi zjednodušia realizáciu zoznamu, preto som sa jednoznačne rozhodol pre druhú možnosť.

VI. Záver

Mojou úlohou bolo vytvoriť program na výmenu textových správ, tzv. chat, ktorý by pracoval na počítačoch so systémom MS-DOS, využíval protokol IPX a pracoval v sieti s viacerými segmentami.

Na začiatku si používateľ zvolí svoju prezývku, ktorá musí byť unikátna, t.j. nemôžu existovať dvaja používatelia s rovnakou prezývkou. Každý používateľ má k dispozícii zoznam mien, ktoré sú v danej chvíli on-line. Zoznam som realizaval ako StringCollection, pretože to prináša viaceré výhody oproti klasickému zoznamu, realizovanému ako pole. Používateľ si môže zistiť, kto sa pod prezývkou "skrýva", pomocou príkazu *whois*. Program podporuje aj privátne správy, ktoré je možné poslať pomocou prikazu *msg*. Program bol vytvorený pre prácu v sietiach typu Novell, pracuje však na každej sieti s protokolom ipx s minimálnym obmedzením.
Pracuje spoľahlivo aj v prostredí Windows, dokonca aj pri prepojení počítačov pomocou Priameho pripojenia káblom.

Dúfam, že program si nájde veľa spokojných užívateľov.

VII. Zhrnutie

Mojou úlohou bolo vytvoriť program, ktorý by umožňoval výmenu textových správ (chat) na lokálnej sieti. Program obsahuje zoznam pripojených užívateľov a umožnuje tiež posielať súkromné správy. Program funguje aj na starších počítačoch, takže určite nájde uplatnenie napr. na školách.

My job was to create program which allows to exchange text messages on local network (chat). Program contains list of connected users and allows to send private messages too. Program works fine on older computers, so it surely finds its use for example at schools.

VIII. Bibliografia

[1] Novell: Assembly Developer's Guide, elektronická dokumentácia, Novell inc., 1999

[2] Novell: Assembly Transport Function Reference, elektronická dokumentácia, Novell inc., 1999

[3] Novell: Routing Information Protocol (RIP), elektronická dokumentácia, Novell inc., 1994

[4] Ralf Brown: Interrupt list, elektronická dokumentácia, Ralf Brown, 1989