COUDP

KNIHOVNA PROTOKOLU UDP

Příručka uživatele a programátora

***SofCon*®spol. s r.o.**
Střešovická 49
162 00 Praha 6
tel/fax: +420 220 180 454
E-mail: sofcon@sofcon.cz
www: http://www.sofcon.cz

Verze dokumentu 1.10

Informace v tomto dokumentu byly pečlivě zkontrolovány a SofCon věří, že jsou spolehlivé, přesto SofCon nenese odpovědnost za případné nepřesnosti nebo nesprávnosti zde uvedených informací.
SofCon negarantuje bezchybnost tohoto dokumentu ani programového vybavení, které je v tomto dokumentu popsané. Uživatel přebírá informace z tohoto dokumentu a odpovídající programové vybavení ve stavu, jak byly vytvořeny a sám je povinen provést validaci bezchybnosti produktu, který s použitím zde popsaného programového vybavení vytvořil.
SofCon si vyhrazuje právo změny obsahu tohoto dokumentu bez předchozího oznámení a nenese žádnou odpovědnost za důsledky, které z toho mohou vyplynout pro uživatele.

Datum vydání: 16.05.2003
Datum posledního uložení dokumentu: 16.05.2003
(Datum vydání a posledního uložení dokumentu musí být stejné)

Obsah :

1. O dokumentu

1.1. Revize dokumentu

Verze dokumentu	Verze SW	Autor	Datum vydání	Popis změn
1.00	1.XX	Čr		První vydání
1.10	1.XX	Tu	16.05.2003	Úprava dokumentu dle ISO9000

1.2. Účel dokumentu

Tento dokument slouží jako popis jednotky implementující protokol UDP.

1.3. Rozsah platnosti

Určen pro programátory a uživatele programového vybavení SofCon.

1.4. Související dokumenty

Pro čtení tohoto dokumentu je potřeba seznámit se s manuálem CoBase.

Popis formátu verze knihovny a souvisejících funkcí je popsán v manuálu LibVer.

2. Termíny a definice

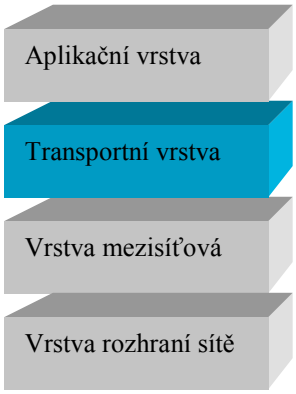
Používané termíny a definice jsou popsány v samostatném dokumentu Termíny a definice.

3. Úvod

3.1.1. Účel knihovny CoUDP

Knihovna **CoUDP** implementuje protokol UDP (User Datagram Protocol). Tj. jeden z transportních protokolů architektury TCP/IP zajišťující koncový přenos dat mezi dvěma stanicemi. Tento protokol je implementován pomocí zařízení TCoUdp. Z následujícího obrázku je zřejmé umístění transportní vrstvy v rámci TCP/IP architektury. Stručný popis architektury TCP/IP je uveden v dokumentaci ke knihovně CoBase.

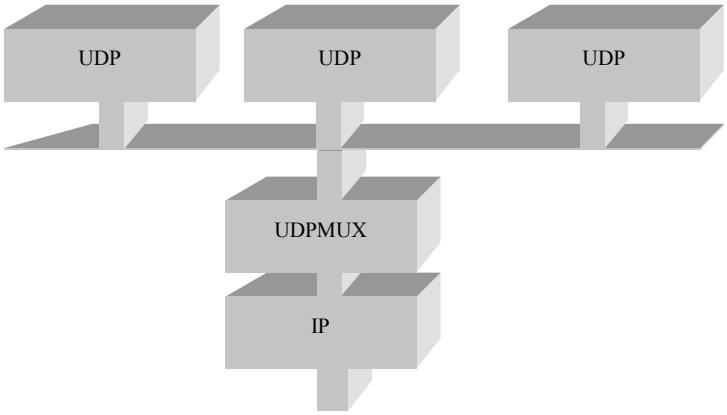
TCP/IP



UDP poskytuje nespolehlivou transportní službu pro ty aplikace, které nepožadují zabezpečení přenosu v takovém rozsahu, jako poskytuje např. protokol TCP. UDP neposkytuje spolehlivý koncový přenos dat, nemá fáze navazování a rušení spojení. UDP protokol však na rozdíl od TCP umožňuje vysílání na všeobecnou adresu (broadcast).

UDP protokol je implementován pomocí dvou zařízení:

- UDP zásuvka (zařízení UDP, TCoUdp)
- UDP multiplexer (zařízení UDPMUX, TCoUdpMux)



Nad síťovou vrstvou (IP protokol) je vždy jedno zařízení UDP multiplexer (zařízení UDPMUX). K zařízení UDPMUX je možné připojit libovolný počet UDP zásuvek (zařízení UDP), které jsou charakterizovány číslem portu. Aplikace vždy přistupuje pouze k zařízení UDP, nikdy ne k UDPMUX.

Protokol UDP je implementovaný pomocí třídy TCoDevice. V této dokumentaci jsou popsány pouze odlišnosti a speciality protokolu UDP (tedy zařízení UDP a UDPMUX). Ostatní naleznete v dokumentaci ke knihovně CoBase.

3.1.2. Protokol UDP

Dalo by se říct, že UDP protokol je v podstatě jednoduchou nadstavbou nad IP protokolem přidávající identifikaci procesů číslem portu.
UDP protokol je službou bez spojení, je tedy vhodný např. k transakčnímu zpracování (komunikace typu dotaz - odpověď), kdy by režie spojená s navazováním a závěrem spojení by byla příliš velká. Výhodou UDP protokolu je možnost posílání datagramů na všeobecnou adresu pomocí oběžníku (broadcast).

UDP datagram má následující formát datagramu a se vkládá do datové části IP datagramu:

0	16
Zdrojový port	Cílový port
Délka dat	Kontrolní součet
Přenášená data	

Zdrojový port — Zdrojový port identifikuje proces (aplikační protokol) ve zdrojové stanici.

Cílový port — Cílový port identifikuje proces (aplikační protokol) na cílové stanici.

Délka dat	Délka UDP datagramu (tj. součet délky dat a záhlaví)
Kontrolní součet	Obsahuje kontrolní součet záhlaví a dat UDP datagramu.
Přenášená data	Délka přenášených dat je omezena na 504 bajtů.

3.1.3. Porty

Rozhraní mezi transportní a aplikační vrstvou, tj. identifikace aplikačního protokolu, který bude poskytovanou transportní službu používat se označuje **číslem portu**. Port je abstrakce, kterou využívají protokoly transportní vrstvy k rozlišení konkrétního cílového aplikačního procesu běžícího na daném počítači.

Typy portů:

- **Známé porty** (well known) se pohybují v rozsahu 0 až 1023 a jsou protokoly pevně dané. Konkrétní hodnoty jsou uvedeny v RFC-1700.
- **Registrované porty** v intervalu 1024 až 49151 jsou užívány pro běžné procesy a aplikace.
- **Dynamické** nebo **soukromé porty** se pohybují v rozmezí 49151 až 65535

Při výběru čísla portu UDP zásuvky v rozsahu 0 až 1023, příp. 1024 až 49151 je vhodné se podívat do normy RFC-1700.

4. Konstanty a jednoduché typy

4.1. Konstanty

4.1.1. Návratové kódy CST_UDP_xxx

Současná verze knihovny CoUDP nedefinuje žádné vlastní návratové kódy CST_UDP_.

4.1.2. Identifikátory tříd zařízení DEV_CLASS_xxx

Třída TCoUdp (protokol UDP) resp. TCoUdpMux (multiplexer protokolu UDP) má přidělen identifikátor **DEV_CLASS_UDP** resp. **DEV_CLASS_UDPMUX**. Tyto identifikátory určují zařízení v řetězci při volání funkcí **CoIoctl**, **CoGetOption**, **CoSetOption** apod. Zaregistrované jména tříd pro použití s funkcí **CoCreateDevice** jsou **UDP** a **UDPMUX**.

4.1.3. Identifikátory řídicích operací IOCTL_UDP_xxx

Současná verze knihovny CoUDP nedefinuje žádné vlastní identifikátory IOCTL_UDP_xxx.

4.1.4. Identifikátory parametrů zařízení COPT_UDP_xxx

Identifikátory s prefixem **COPT_** specifikují parametry zařízení ve funkci **CoGetOption** příp. **CoSetOption**. Knihovna CoUDP definuje několik nových konstant s prefixem COPT_UDP_:

COPT_UDP_LPORT

Implicitní UDP port, na kterém bude UDP zařízení přijímat příchozí datagramy a kterým budou označeny datagramy odchozí. Tato hodnota koresponduje s parametrem LPORT v konfiguračním textovém řetězci. Parametr lze měnit v libovolném okamžiku, změna se však projeví až v okamžiku volání metody CoBind (pokud místo adresy uvedeme hodnotu **nil**)

Formát dat: Word Implicitní hodnota: 5000

COPT_UDP_RADDR

Implicitní adresa protistanice. Přestože UDP protokol nenavazuje spojení implementuje metodu **CoConnect** a **CoDisconnect** (viz. kapitola 6.2). Parametr lze měnit v libovolném okamžiku, změna se však projeví až v okamžiku volání metody CoConnect (pokud na místě parametru AAddress uvedeme místo konkrétní adresy uvedeme hodnotu **nil**).

Formát dat: TUdpAddress Implicitní hodnota: 0.0.0.0:5000

4.2. Typy

4.2.1. Třída TCoUdp

```
TCoUDP = object( TCoDevice );
```

Pokud použijeme knihovnu CoUDP (uvedeme ji v seznamu za klíčovým slovem **uses**), bude tato třída zaregistrovaná v globálním seznamu tříd zařízení pod jménem **UDP**. Toto jméno lze použít v případě vytváření třídy pomocí funkce CoCreateDevice. Číselný identifikátor třídy pro použití s metodami CoIoctl, CoGetOption, CoSetOption apod. je **DEV_CLASS_UDP**.

4.2.2. Třída TCoUdpMux

```
TCoUdpMux = object( TCoUdpMux );
```

Pokud použijeme knihovnu CoUDP (uvedeme ji v seznamu za klíčovým slovem **uses**), bude tato třída zaregistrovaná v globálním seznamu tříd zařízení pod jménem **UDPMUX**. Toto jméno lze použít v případě vytváření třídy funkcí **CoCreateDevice**. Číselný identifikátor třídy je **DEV_CLASS_UDPMUX**.

4.2.3. Struktura TUdpAddress

```
PUdpAddress = ^TUdpAddress;
TUdpAddress = packed record
  Size    : Byte;
  Address : TIpAddress;
  Port    : Word;
end;
```

Struktura **TUdpAddress** popisuje adresu na transportní vrstvě pro protokol UDP, která se použije při volání aplikačních metod třídy TCoUdp na místech, kde se vyskytuje typ **TCoAddress** (tj. v metodách CoSendBufferTo, CoRecvBufferFrom, CoBind a CoConnect).

Položka **Size** udává velikost struktury TUdpAddress. Před použitím struktury se vždy nezapomeňte ujistit, že položka **Size** obsahuje správnou hodnotu, tedy SizeOf(TUdpAddress). V položce **Address** je obsažena adresa uzlu, v položce **Port** je uloženo číslo Portu.

5. Textový konfigurační řetězec

Parametry zařízení TCoUDP lze nastavovat pomocí textového konfiguračního řetězce metodami **CoSetOptionString**.

V následujícím seznamu jsou uvedeny všechny povolené identifikátory parametrů:

LPORT Implicitní UDP port, který se použije v případě, že se zavolá metoda **CoBind** bez specifikované adresy. Parametr LPORT lze měnit v kterémkoli okamžiku, změna se však promítne až při volání metody **CoBind**.

Formát dat: číslo
Implicitní nastavení: 5000

RADDR Implicitní adresa cílové stanice v případě, že se zavolá metoda **CoConnect** bez specifikované adresy. Parametr RADDR lze měnit v kterémkoli okamžiku, změna se však promítne až při volání metody **CoConnect**.

Formát dat: text, IP adresa
Implicitní nastavení: 0.0.0.0

RPORT Implicitní port na cílové stanici v případě, že se zavolá metoda **CoConnect** bez specifikované adresy. Parametr RPORT lze měnit v kterémkoli okamžiku, změna se však promítne až při volání metody **CoConnect**.

Formát dat: číslo
Implicitní nastavení: 5000

Současná verze zařízení TCoUdpMux nemá žádné parametry a tudíž jí není potřeba konfigurovat pomocí textového konfiguračního řetězce.

Příklad:

```
var
  Sock : PCoDevice;

.
.

Sock^.CoSetOptionString( DEV_CLASS_UDP,  'LPORT=5001', nil );
```

6. Poznámky

6.1. Inicializace zařízení pomocí CoBind

Metoda **CoBind** provede inicializaci instance UDP protokolu a zaregistruje vyžádaný lokální port u zařízení nižší vrstvy (UDP multiplexer).

První parametr metody **CoBind**, tj. AAdress je ukazatel na lokální adresu (strukturu TUdpAddress). Protože, lokální IP adresa je jednoznačně specifikovaná již vrstvou protokolu IP, pouze položky Protocol a Size struktury TUdpAddress musí být vyplněny. Položku Address lze naplnit hodnotu IP_ADDR_ANY. Pokud místo ukazatele na strukturu TUdpAddress předáme hodnotu **nil**, pak se použije implicitní nastavení lokálního portu podle implicitního nastavení adresy protistanice (viz. COPT_UDP_LPORT nebo parametr konfiguračního textového řetězce LPORT).

6.2. Navazování spojení pomocí CoConnect

Protokol UDP neumožňuje navázat spojení mezi dvěma stanicemi. Přesto zařízení UDP implementuje metodu **CoConnect**. Metoda **CoConnect** neprovádí navazování spojení, pouze si uloží zadanou adresu protistanice. Uložená adresa protistanice se použije na filtrování příchozích datagramů (jsou přijaty pouze datagramy od stanice se kterou je „navázáno spojení“). Po zavolání metody **CoConnect** lze tedy používat metody **CoSendBuffer** bez určení adresy cílové stanice a **CoRecvBuffer**, kdy adresa zdrojové stanice je jednoznačná.

První parametr metody **CoConnect** je ukazatel na adresu protistanice (strukturu TUdpAddress). Struktura TUdpAddress musí mít před voláním metody **CoConnect** vyplněné všechny položky. Pokud místo ukazatele na strukturu předáme hodnotu nil, pak se použije implicitní nastavení adresy protistanice (viz. COPT_UDP_RADDR, nebo parametr konfiguračního textového řetězce RADDR, RPORT).

6.3. Posílání a příjem dat

K posílání a příjmu dat jsou určené metody **CoSendBufferTo** a **CoRecvBufferFrom** (příp. **CoSendBuffer** a **CoRecvBuffer** v případě „navázaného spojení“). K předávání adresy cílové příp. zdrojové adresy je určena struktura TUdpAddress. V případě metod pro odesílání a příjem dat jsou využity všechny položky struktury TUdpAddress.

UDP protokol umožňuje posílat data na všeobecnou adresu (broadcast). Všeobecná adresa je 255.255.255.255 (konstanta IP_ADDR_BROADCAST)

7. Příklad

Následující příklad ukazuje jak použít UDP zásuvku nad TCP/IP zásobníkem. Ukázkový program přijímá na portu 5000 libovolné datagramy
a odesílá je zpět zdrojovým stanicím.

```
uses
  Crt,
  CoBase,
  CoEth01,
  CoUDP,
  CoINet;

var
  Status   : TCoStatus;
  Sock     : PCoDevice;
  SrcAddr  : TUdpAddress;
  RecvBuff : array[0..575] of Byte;
  RecvLen  : Word;

begin
  { Vytvoření TCP/IP zasobniku }
  { Předpokládá desku IOETH01 na adrese $320 }
  Status := NetOpenStack( 'ETH01', 'IOBASE=$320',
            'IPADDR="192.168.1.200" , '' );

  if Status <> CST_SUCCESS then
  begin
    WriteLn( 'NetOpenStack() failed: ', CoStatusToStr( Status ) );
    Exit;
  end;

  { Vytvoření UDP zásuvky }
  Status := NetOpenSocket( 'UDP', 'LPORT=5000', '', Sock );

  if Status <> CST_SUCCESS then
  begin
    WriteLn( 'NetOpenSocket() failed: ', CoStatusToStr( Status ) );
    NetCloseStack( '' );
    Exit;
  end;

  { Inicializace UDP zásuvky, lokální adresa je daná konfiguračním
    řetězcem (port=5000) }
  Status := Sock^.CoBind( nil, COF_PROPAGATE, nil );

  if Status <> CST_SUCCESS then
  begin
    WriteLn( 'CoBind() failed: ', Sock^.CoStatusToStr( Status ) );
    NetCloseSocket( Sock );
    NetCloseStack( '' );
    Exit;
  end;

  repeat
    { Příjem datagramu, do bufferu RecvBuff }
    Status := Sock^.CoRecvBufferFrom( @RecvBuff, SizeOf( RecvBuff ),
                                      @RecvLen, 0, @SrcAddr, nil );
    case Status of

      CST_SUCCESS:
      begin
        { Odeslání přijatého datagram zpátky odesílateli }
```

```
        Status := Sock^.CoSendBufferTo( @RecvBuff, RecvLen, 0,
                                        @SrcAddr, nil );
        if Status <> CST_SUCCESS then
        begin
          WriteLn( 'CoSendBufferTo() failed: ',
            Sock^.CoStatusToStr( Status ) );
        end;
      end;

      CST_ERR_TIMEOUT:
      begin
        { Chybu Timeout nezobrazujeme jako ostatní chyby }
      end;

    else
      WriteLn( 'CoRecvBufferFrom() failed: ',
        Sock^.CoStatusToStr( Status ) );
    end;

  until Keypressed;

  { Uvolnění UDP zásuvky }
  Status := Sock^.CoUnbind( COF_PROPAGATE, nil );

  if Status <> CST_SUCCESS then
  begin
    WriteLn( 'CoUnbind() failed: ', Sock^.CoStatusToStr( Status ) );
  end;

  { Uzavření UDP zásuvky }
  NetCloseSocket( Sock );

  { Uzavreni TCP/IP zasobniku }
  NetCloseStack( '' );
end;
```