PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Program pro podporu výuky na SŠ

2011 Petr Jelínek

Anotace

Názornost a přehlednost výuky je velmi důležitá vzhledem ke komplikovanější problematice učiva, kde je potřeba správná představa pro studenty. Tato práce je tvořena formou výukového programu pro předmět Počítačové systémy. Program je rozdělen do tří částí, které představují výukový proces. První část výuková seznamuje studenty s pojmy IP adresa, maska, routování, atd. Naopak část praktická se zabývá názornou ukázkou postupů. Třetí část testovací slouží k otestování praktických dovedností studenta.

Na tomto místě bych chtěl poděkovat Mgr. Jiřímu Zacpalovi, Ph.D. za vedení při zpracování této bakalářské práce. Dále děkuji svým rodičům a přátelům za morální podporu při studiu.

Obsah

Seznam obrázků

1. Zadání bakalářské práce

Úkolem této bakalářské práce bylo vytvoření výukového programu pro podporu výuky na střední škole. Podmínkou bylo vytvořit program, který by nějaká konkrétní škola využila. Proto jsem kontaktoval vyučujícího informatiky na mnou absolvované střední škole a vznikl nápad tématu, který se týká počítačové sítě, routování a nastavování jednotlivých PC v síti.

Projekt je vytvářen pro Obchodní akademii v Mohelnici, kde bude využit při výuce v předmětu Počítačové systémy. V předmětu se vyučuje látka z oblasti informatiky pro střední školy. Program pokrývá výuku probíranou začátkem čtvrtého ročníku, kde se studenti učí základům počítačové sítě a nastavovat směrování. Pro praktickou výuku a následné testování se využívá virtuálních počítačů, u kterých jsou přístupny pouze konzole linuxu. Tento postup je sice velice praktický z důvodu reálného nastavování a chování sítě, ale studentům dle mého názoru nenabízí přehlednost. Z vlastní zkušenosti mohu říct, že zejména studentky při testu pouze píší do konzole příkazy, které mají z hlavy naučené a nemají představu o tom, co příkaz způsobí.

Tento projekt se zaměřuje na přehlednost počítačové sítě, kterou simuluje. Neobsahuje nastavování počítačů a směrování pomocí příkazů, ale nastavení se zadává prostřednictvím formulářů a celou síť vidí přehledně na obrazovce. Student se tak nemusí starat o správnost příkazu, ale řeší samotnou funkčnost a snadněji tak pochopí, jak síť pracuje.

Vyučujícím tohoto předmětu bylo požadováno, aby program uměl testovat studenty. Tedy aby nastavil test, který se studentům spustí, a oni jej vypracují. S tím souvisí i příprava na test, kterou program musí nabízet.

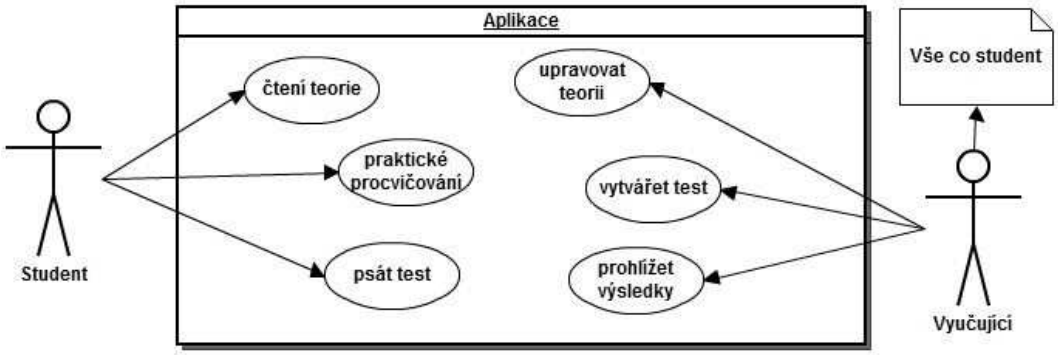
Podobnou problematikou se zabývá společnost CISCO, jejíž produkty jsou ve svém zaměření propracovanější, ale finančně nákladnější. Tento na míru vytvořený projekt by měl pro výukové potřeby daného předmětu dostačovat.

2. Popis projektu

Mým hlavním cílem bylo vytvořit zajímavé uživatelské prostředí, které by mělo jednoduché a intuitivní ovládání. To jsem vyřešil pomocí univerzální nástrojové lišty, která mění svůj obsah podle potřeby. Další cíl byl, aby problematika počítačové sítě byla lehce pochopitelná a orientace v nastavení simulace sítě byla snadná a srozumitelná. Díky animaci příkazu "ping" lze snadno odhalit, kde je chyba v nastavení a tak byl i tento cíl splněn.

2.1. Rychlý přehled funkcí

Program se dělí do tří částí, podle způsobu použití.



Obrázek 1. Use case diagram funkcí programu.

2.1.1. Teorie

První a po stránce funkčnosti nejmenší část programu je teorie. Teoretické základy jsou potřebné k plnému využití praktické části programu. Obsahuje tedy základní teoretické termíny a názorné vysvětlení s obrázky.

2.1.2. Praxe

Praxe tvoří základní část programu. Student má možnost vytvářet počítačovou síť na dané ploše, nastavovat jednotlivé prvky sítě, zkoušet komunikaci s názornou vizualizací a prohlížet uložené ukázkové sítě.

2.1.3. Test

V testovací části nemůžeme na rozdíl od praktické budovat či přesouvat prvky sítě. Povoleno je pouze nastavovat prvky sítě tak, aby bylo splněno zadání. Test pracuje na principu Klient-Server, kde server tvoří část programu pro vyučujícího.

2.1.4. Administrace

V této části je vyučujícímu umožněno přidávat nebo odebírat články z teoretické části, vytvářet a ukládat ukázkové sítě pro praktickou část a hlavně vytvářet testy a prohlížet výsledky. Vyučující má kontrolu nad tím, kdo test vypracovává, protože se mu zobrazují uživatelská jména připojených studentů.

2.2. Použití projektu

Program by měl být využit pro výuku na střední škole Obchodní akademie v Mohelnici na oboru Informatika v ekonomice. Po otestování se škola rozhodne zda opravdu program nastoupí do ostrého provozu již tento rok nebo bude muset podstoupit případné úpravy.

2.3. Přínos

Aplikace přináší jednoduchý pohled na problematiku počítačové sítě.Je utvořena pro studenty, kteří požadují pouze základní znalosti na úrovni střední školy. Nabízí efektivní ukázku, jak zjednodušeně síť pracuje.

2.4. Plán do budoucna

Do budoucna mám v plánu aplikaci postupně rozšiřovat a tím vylepšit názornost výuky na dané škole. Chtěl bych rozšířit stavbu sítě na rozdílné síťové prvky jako jsou switche, různé druhy propojovacích prvků a tím i výpočet teoretické přenosové rychlosti atd.

3. Teoretický úvod

V této kapitole se seznámíme s teoretickými základy, které jsou potřebné pro používání programu. Dále se pak seznámíme s použitými technologiemi.

3.1. Úvod do počítačových sítí

Počítačovou síť definujeme jako propojení počítačů pomocí propojovacích prvků, které slouží pro výměnu informací prostřednictvím protokolů. Největší takovou sítí je globální celosvětová síť internet. Používá protokol TCP/IP, který se využívá i v této práci pro komunikaci studentské testovací části aplikace s části pro vyučujícího.

Pomocí TCP komunikuje aplikace prostřednictvím spojení, přes která se přenáší data. Doručení paketů je díky kontrolování a případnému přeposílání spolehlivé. TCP se využívá například u webových stránek, kde všechna data potřebujeme pro plné načtení.

Druhý využívaný protokol je UDP. Říká se o něm, že je nespolehlivý, protože na rozdíl od TCP přenáší datagramy bez kontroly doručení. UDP se používá tam, kde nepotřebujeme každý datagram nebo je zbytečné ztrácet čas s požadavkem na přeposlání (např. online hry, videostream...).

3.1.1. Síťový prvek

Síťové prvky se dělí na aktivní a pasivní. Aktivní prvky jednoduše definujeme jako zařízení zapojené do sítě, které nějakým způsobem ovlivňují přenášené signály (zesiluje nebo upravuje). V této aplikaci je pouze jeden druh aktivních prvků a to je samotné PC. Mezi další aktivní prvky patří:

- **Repeater** - zesiluje signál, který klesá s nárůstem vzdálenosti.
- **Switch** - také zesiluje signál, umožňuje zapojit více kabelů a přeposílá data na porty, pro které jsou určena.
- **Router** - routeru je věnována kapitola níže.

Pasivními prvky označujeme datové rozvaděče, které přenášejí data.

- **Koaxiální kabel** - nejstarší typ.
- **Kroucená dvojlinka** - po budovách nejrozšířenější.
- **Optický kabel** - data se přenášejí světelnými impulsy. Využívá se pro přenos na velké vzdálenosti.

3.1.2. IP adresa

IP adresa není nic jiného, než adresa rozhraní počítače (síťové karty). Skládá se ze 4 čísel, můžeme je nazvat "okty", kde každé z nich je tvořeno 8bity. Dohromady má tedy adresa k dispozici 32 bitů. Nejčastěji se zapisuje v desítkové soustavě a jednotlivá čísla se oddělují tečkou (např. 192.168.1.1). 0 na pozici adresy počítače označuje adresu sítě a hodnota 255 určuje adresu hromadného packetu pro tuto síť.

```
IP binárně:              11000000.10101000.00000001.00000001
IP dekadicky:                 192.      168.       1.        1
Adresa sítě:                  192.      168.       1.        0
Hromadný paket:               192.      168.       1.      255
```

Pomocí IP se adresují počítače v síti (nebo jiné síťově prvky). Je důležité, aby v jedné síti byly vždy jedinečné adresy a to je současným problémem. U verze IPv4 dochází k vyčerpání IP adres a problém se řeší přechodem na verzi IPv6.

IP adresa se skládá ze dvou základních částí:

- Adresa sítě
- Adresa počítače

IP adresa se dělí do několika tříd (obr. 2.) v závislosti na tom, kolik bytů adresuje síť a kolik pak počítač.

- Třída A - Používají ji hlavně nadnárodní společnosti. Dovoluje adresovat jen 126 sítí a v každé z nich 16 miliónů počítačů. Rozsah hodnot adres je 0.0.0.0 - 127.255.255.255.
- Třída B - Využívá pro adresování sítě první dva byty. U nás mají jen významné organizace. Adresuje 16 tisíc sítí a 65 tisíc počítačů.
- Třída C - Patří mezi nejpoužívanější třídy a je taky jediná možná v nastavení této aplikace. Adresuje 2 milióny sítí a v každé síti může být až 254 počítačů.

Pro vnitřní sítě jsou rezervovány tyto adresy:

- **Třída A -** 10.0.0.0 až 10.255.255.255
- **Třída B -** 172.16.0.0 až 172.31.0.0
- **Třída C -** 192.168.0.0 až 192.168.255.0

	1. byte	2. byte	3. byte	4. byte
Třída A	net	host	host	host
Třída B	net	net	host	host
Třída C	net	net	net	host

Adresa počítače
Adresa sítě

Obrázek 2. Základní rozdělení adres do tříd

3.1.3. Maska a tvorba podsítí

Maska sítě je číslo, které rozděluje síť na podsítě. Zapisuje se stejně jako IP adresa. V binárním zápisu je maska nepřetržitá řada jedniček končící nulou, která vyjadřuje číslo sítě. Řadu do počtu 32 bitů doplňují nuly vyjadřující rozsah prostoru pro adresu síťového rozhraní.

```
Maska dekadicky:      255.      255.    255.       0
Maska binárně:   11111111.11111111.11111111.00000000
```

Číslo masky je možno vyjádřit i zkrácenou formou pomocí tzv. CIDR notace, která vyjadřuje délku jedničkové řady.

```
IP adresa:      192.168.  1.  1
Maska:          255.255.255.  0
Zkrácený zápis: 192.168.  1.  1/24
```

Z čísla sítě a masky můžeme vypočítat rozsah IP adres pro danou síť. Potřebujeme zapsat masku a číslo sítě v binárním tvaru. Maska nám pomyslně rozdělí příklad uvedený níže na dva sloupce. Na jedničky, to znamená adresa fixní a na nuly, kde jsou všechna čísla v daném binárním rozsahu. Z rozsahu musíme vyloučit samé nuly a samé jedničky, protože se jedná o číslo sítě a adresu broadcastu.

```
Maska binárně:      11111111.11111111.11111111.1  0000000  (25 bitů)
Číslo sítě binárně: 11000000.10101000.00000001.0  0000000  (192.168.1.0)
1. IP adresa:       11000000.10101000.00000001.0  0000001  (192.168.1.1)
2. IP adresa:       11000000.10101000.00000001.0  0000010  (192.168.1.2)
...
Předposlední IP:    11000000.10101000.00000001.0  1111101  (192.168.1.125)
Poslední IP:        11000000.10101000.00000001.0  1111110  (192.168.1.126)
Broadcast:          11000000.10101000.00000001.0  1111111  (192.168.1.127)
```

Síť v příkladu poskytuje 126 IP adres a 2 podsítě. 192.168.1.128 je adresa druhé podsítě a tedy 192.168.1.129 je první použitelná IP druhé podsítě.

3.1.4. Výchozí brána

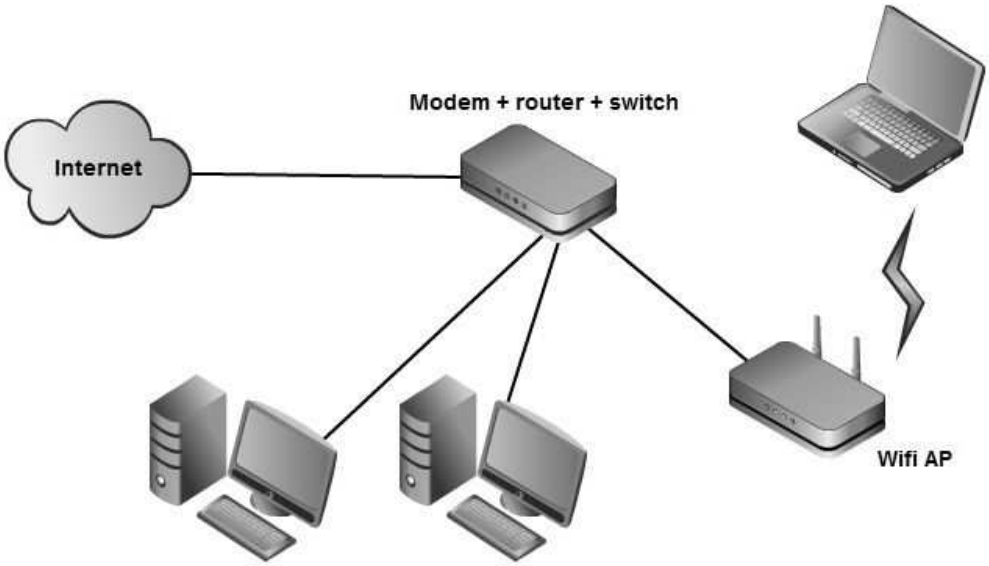
Výchozí brána je síťový prvek, na který se posílají pakety adresované mimo síť. Tedy na takové adresy, které nejsou z lokální sítě dostupné. Jednoduchým příkladem může být domácí modem připojený do telefonní linky. Ten je pro lokální počítače výchozí bránou.

Pro jednoduchý test odezvy slouží příkaz ping. Pokud při jeho spuštění dostane na vstupu lokální adresu, přímo se ji pokusí kontaktovat. Jestliže dostane adresu spadající do jiné sítě, kontaktuje výchozí bránu, je-li nastavena. Výchozí brána dále přebírá dotaz a zkontroluje svoji lokální dostupnost. Následně se řídí dle záznamů routovací tabulky, pokud nenalezne záznam, tak pošle paket na svou výchozí bránu.

3.1.5. Router a routování

Výchozí bránu můžeme také nazvat router. Router přeposílá pakety k jejich cíli. Spojuje dvě odlišné sítě a tím se liší od switche, který také přeposílá pakety, ale dělá tak pouze v rámci lokální sítě.

Router může mít podobu jakéhokoli počítače, který má dvě síťové karty a podporu pro síťování. I domácí modem je router, protože propojuje lokální síť s internetem. Na obr. 3. je router společně s modemem a switchem v jednom zařízení.



Obrázek 3. Ukázka zapojení routeru

Rozhodování, kam odkázat na cíl, se provádí na základě routovací tabulky. Je to seznam sítí a adres, kam dotaz přeposlat. Každý řádek tabulky tvoří jednu

směrovací informaci. Při zapojení většího počtu sítí vznikne z routeru pomyslná křižovatka.

Podle vzniku záznamu ve směrovací tabulce se směrování dělí na statické a dynamické.

3.1.6. Statické směrování

Statické směrování se používá v této aplikaci. Jedná se o ruční vyplnění jednotlivých tabulek routerů. Tyto údaje se v průběhu provozu nemění bez zásahu administrátora.

Statické směrování využívají koncové stanice nebo routery v malých lokálních sítích, protože záznamů není mnoho nebo není potřeba je často měnit.

3.1.7. Dynamické směrování

Při použití dynamického směrování si router sám průběžně zjišťuje stav sítě a reaguje na přidávání či odebírání sítí změnou záznamu v tabulce.

Hodí se zejména do velkých sítí, kde se záznamy často obměňují nebo je záznamu příliš mnoho na to, aby je člověk dokázal efektivně upravovat.

3.2. Použité technologie

Aplikace je psána v jazyce C# a testována na OS Windows 7 Professional s nainstalovaným rozhraním Microsoft .NET Framework 4. Pro grafické prostředí je využito subsystému WPF a na síťovou komunikaci .NET Remoting.

3.2.1. Jazyk C# a .NET Framework

Programovací jazyk C# často spojujeme s pojmem .NET Framework, protože metodika jazyka C# je základní metodika platformy .NET.

.NET je soubor technologií, který tvoří platformu. Není předepsáno, který programovací jazyk pro něj použít. Aplikace se vždy přeloží do mezijazyka Common Intermediate Language. Nejpoužívanější jazyk ve spojení s .NET jsou C#, Visual Basic .NET nebo Delphi.

Jazyk C# je poměrně nový a založený na objektově orientované programování a speciálně navržený pro použití s platformou .NET Framework. Podobá se jazyku C a Javě. Zvláštnost je rozlišování mezi hodnotovými a referenčními typy. Hodnotové typy jsou takové, kde proměnná uchovává data, ale u referenčních typů obsahuje proměnná adresu, na které jsou data dostupná. U C++ lze přirovnat referenční typ k přístupu k proměnné pomocí ukazatele.

Automatická správa paměti je také součástí jazyka. Modul této správy zajišťuje správu paměti v platformě .NET. Řeší hlavně uvolňování paměti.

Platforma .NET Framework umožňuje ošetřit výjimky stejně jako třeba v jazycích Java a C++. C#, navíc podporuje blok finally.

3.2.2. WPF a XAML

Windows Presentation Foundation (WPF) je součástí .NET Framework od verze 3.0, který se nachází ve všech systémech Windows od Windows Vista a výše. Do Windows XP lze doinstalovat.

WPF slouží k vytvoření bohatého uživatelského rozhraní. Díky značkovacímu jazyku XAML je funkčnost oddělena od vzhledu aplikace. Grafika se vykresluje pomocí Direct3D knihoven a to umožňuje hardwarovou akceleraci pomocí grafické karty. Grafické objekty používané ve WPF jsou tvořené vektorovou grafikou. Spojením s animací tak dosáhneme plynulého přibližování bez rozmazání. WPF nabízí i 3D rendering a základní orientaci ve 3D prostoru.

Data Binding, který WPF podporuje, umožňuje načítat data z nějakého zdroje. Jsou tři typy:

- „one time“ stáhne data pouze poprvé.
- „one way“ data pouze čte.
- „two way“ aktualizuje se zdroj i cíl.

XAML je značkovací jazyk vycházející z XML, využívaný k popisu grafického rozhraní ve WPF a Silverlight. V .NET Frameworku se využívá od verze 3.0 stejně jako WPF.

Vše, co se napíše v XAML, lze přepsat jazykem C#. Hodí se zejména na vytváření a inicializaci objektů, protože zápis v XAML je oproti ekvivalentu v C# v takových případech jednodušší. V následujícím příkladu je inicializace jednoduchého textového bloku s textem ahoj a velikostí písma 12.

```
<TextBlock Name="mujblok" Text="ahoj" FontSize="12"></TextBlock>
```

Ekvivalent v C# by pak vypadal takto:

```
TextBlock mujblok = new TextBlock();
mujblok.Text = "ahoj";
mujblok.FontSize = 12;
```

Následující ukázka XAML zápisu ukazuje definice animací pro prvek na ovládací liště. Do místa tří teček pak stačí dát samotný obsah okna.

```
<Grid Margin="5,0,0,0">
  <Grid.RenderTransform>
      <ScaleTransform x:Name="MainScale" CenterX="25" CenterY="25"/>
  </Grid.RenderTransform>
  <Grid.Triggers>
      <EventTrigger RoutedEvent="MouseEnter">
          <BeginStoryboard>
              <Storyboard>
                  <DoubleAnimation Storyboard.TargetName="MainScale"
                                   Storyboard.TargetProperty="ScaleX"
                                   To="1.2" Duration="0:0:0.15"/>
                  <DoubleAnimation Storyboard.TargetName="MainScale"
                                   Storyboard.TargetProperty="ScaleY"
                                   To="1.2" Duration="0:0:0.15"/>
              </Storyboard>
          </BeginStoryboard>
      </EventTrigger>
      <EventTrigger RoutedEvent="MouseLeave">
          <BeginStoryboard>
              <Storyboard>
                  <DoubleAnimation Storyboard.TargetName="MainScale"
                                   Storyboard.TargetProperty="ScaleX"
                                   To="1" Duration="0:0:0.20"/>
                  <DoubleAnimation Storyboard.TargetName="MainScale"
                                   Storyboard.TargetProperty="ScaleY"
```

```
                                        To="1" Duration="0:0:0.20"/>
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
        <EventTrigger RoutedEvent="MouseLeftButtonDown">
            <BeginStoryboard>
                <Storyboard>
                    <DoubleAnimation Storyboard.TargetName="MainScale"
                                     Storyboard.TargetProperty="ScaleX"
                                     To="1" Duration="0:0:0"/>
                    <DoubleAnimation Storyboard.TargetName="MainScale"
                                     Storyboard.TargetProperty="ScaleY"
                                     To="1" Duration="0:0:0"/>
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
        <EventTrigger RoutedEvent="MouseLeftButtonUp">
            <BeginStoryboard>
                <Storyboard>
                    <DoubleAnimation Storyboard.TargetName="MainScale"
                                     Storyboard.TargetProperty="ScaleX"
                                     To="1.2" Duration="0:0:0"/>
                    <DoubleAnimation Storyboard.TargetName="MainScale"
                                     Storyboard.TargetProperty="ScaleY"
                                     To="1.2" Duration="0:0:0"/>
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </Grid.Triggers>
...
</Grid>
```

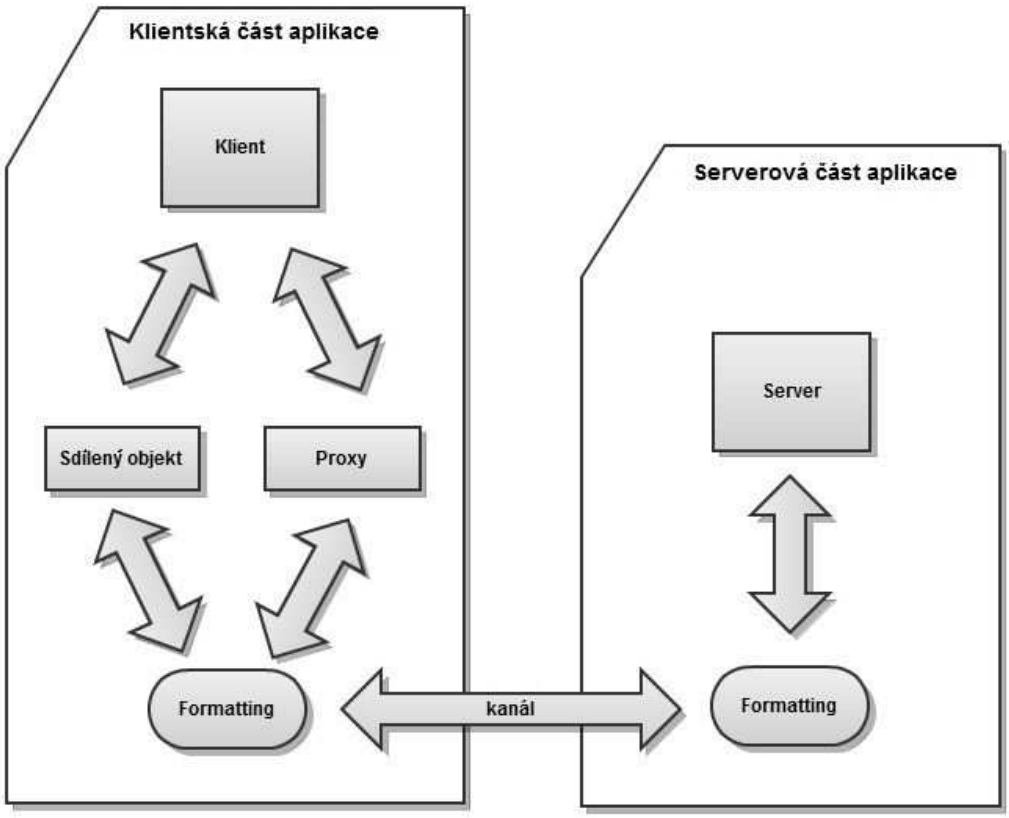
3.2.3. .NET Remoting

Máme více možností, jak docílit komunikace aplikací prostřednictvím sítě. Jednak můžeme udělat vlastní TCP server a klient nebo můžeme využít technologii .NET Remoting, kterou jsem pro komunikaci využil v této aplikaci.

Na první pohled vypadá možná trochu složitěji, ale po podrobnějším seznámení a prvních pokusech o zprovoznění se zdá být dobrým nástrojem. V .NET je HTTP a TCP kanál, prostřednictvím kterých probíhá komunikace mezi aplikacemi. TCP se využívá v rámci lokální sítě, kde mu nebrání firewally, naproti tomu HTTP poskytuje komunikace i přes firewall a je proto vhodný pro komunikaci v

rámci internetu, neposkytuje ale takovou výkonnost jako TCP.

Chceme-li využít technologii .NET Remoting, nespokojíme se pouze s klientem a serverem, ale je zapotřebí vytvořit tzv. sdílený objekt, který tvoří rozhraní pro přístupné metody serveru. Je to prostředník pro komunikaci. Pokud chceme komunikovat i pomocí událostí, budeme nuceni přidat i proxy objekt pro události. Klient se nemůže přímo zaregistrovat na událost severu, ale musí se registrovat přes sdíleného prostředníka a tím je proxy objekt. Událost se tak nevyvolává přímo ale přes proxy. Názorná ukázka použití obou těchto objektů je na obr. 4..



Obrázek 4. Schéma použití .NET Remoting.

4. Uživatelská příručka

4.1. Požadavky

Pro běh aplikace se vyžaduje operační systém Microsoft Windows XP a vyšší spolu s .NET Framework 4.0. Tento Framework ve Windows XP ve výchozím stavu není obsažen, a proto jej musíte doinstalovat pomocí přibaleného souboru „dotNetFx40_Full_x86_x64.exe“ nebo pomocí webových stránek Microsoftu.

4.2. Instalace

Instalace se provádí pouze rozbalením archivu „program.rar“ do požadované složky na disku. Před instalací je potřeba doinstalovat .NET Framework 4.0 (Pokud jej nemáte). Aplikace je navrhnuta pro spouštění přes síť. Je tedy ještě potřeba složku s programem nasdílet s oprávněním pro čtení (zejména nesmí být umožněno smazat soubor Pass ve složce Data). Sdílení je nutné pouze v případě, že chcete program využívat v plné míře.

4.3. Pracovní prostředí

Aplikace je rozdělena do tří částí, každá z nich představuje jednu část výukového procesu.

První část teoretická slouží pro seznámení studenta se základy potřebnými pro ovládání praktické části.

Praktická část obsahuje jednoduchou simulaci počítačové sítě. Studenti si zde mohou vybudovat vlastní síť, nastavit jednotlivé uzly a otestovat komunikaci pomoci příkazu "ping". Tento příkaz je zpracován vizuální animací, která ukazuje tok dat. Díky těmto vlastnostem je problematika snáze pochopitelná.

Poslední část přístupná pro studenta je část testovací. Po přihlášení do testu dostane zadání a musí na počítačové síti zprovoznit komunikaci.

Do poslední části administrátorské smí pouze vyučující. Prostředí lehce umožňuje upravovat texty v teoretické části, vytvářet ukázkové sítě a sítě určené pro test. Samotný test také spouští spolu s procházením výsledků starších testů.

4.3.1. Hlavní okno

Okno, které se zobrazí hned po spuštění je malé a jednoduché. Nabízí přímo volbu mezi částmi aplikace, která se má spustit. Volba je zpracována pomocí čtvercových tlačítek opatřených ikonou. Vedle ikony křížku se nachází ikona otazníku pro spuštění nápovědy. Ukázka na obr. 5..

Obrázek 5. Hlavní okno.

4.3.2. Nástrojová lišta

Pro ovládání všech částí slouží stejná nástrojová lišta, ta mění svůj obsah podle aktuální potřeby a možností programu v daném stavu.

Obrázek 6. Nástrojová lišta.

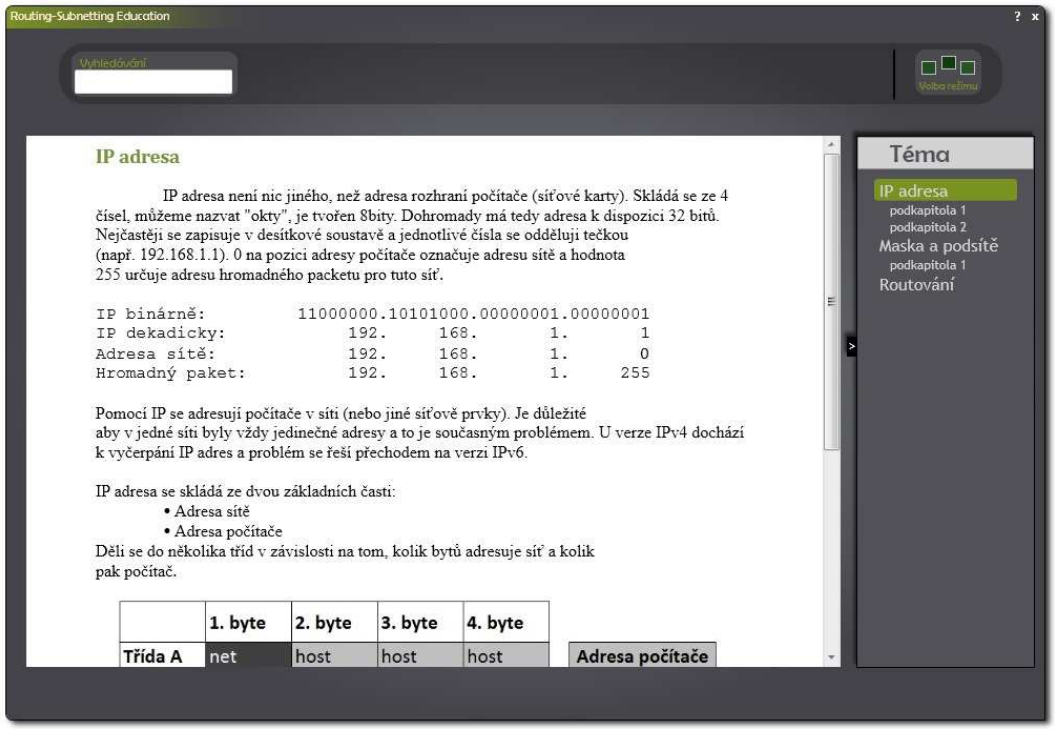
Na liště se úplně vpravo vyskytuje tlačítko pro volbu režimu. Po stisku vás odkáže do hlavního okna. Tento prvek je povolen vždy s jednou výjimkou. Při psaní testu je student povinen práci nejprve odevzdat a potom mu je umožněno opustit testovací část.

4.3.3. Teoretická část

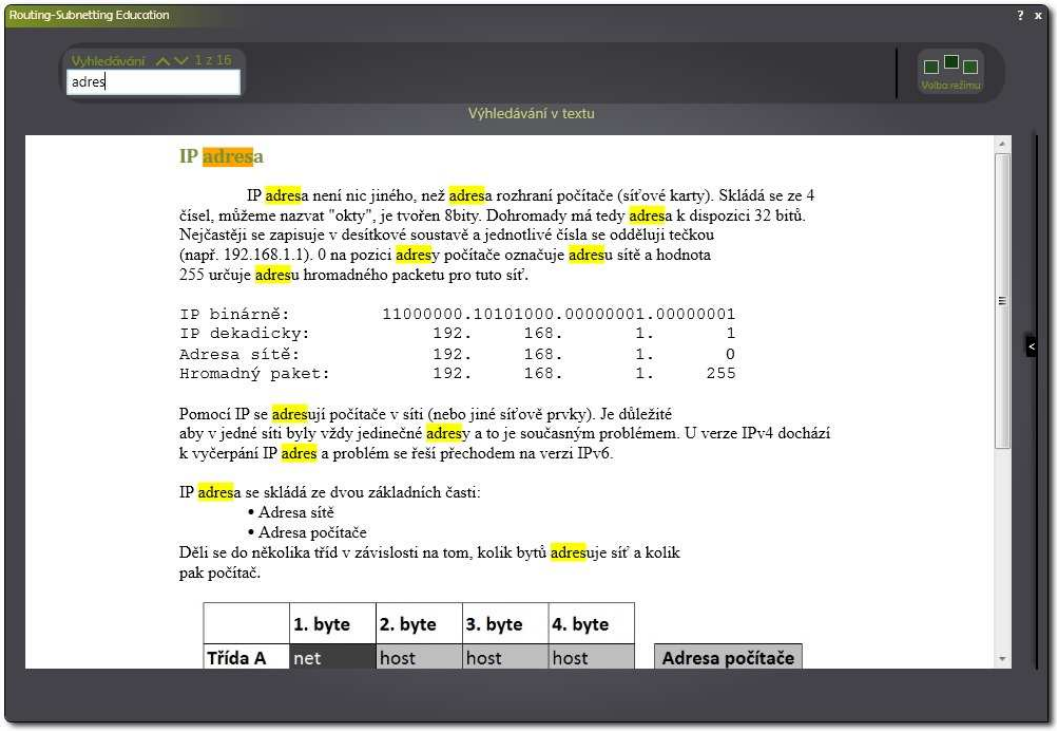
Část teoretická je nejjednoduší. Nabízí nejmíň možností a také proto je na nástrojové liště pouze jedna možnost, kterou je vyhledávání v textu.

Teoretická část je tvořena ovládací lištou nahoře. V levé části je box pro zobrazování textu, který si podle kapitoly vybereme v seznamu na pravé straně. Po kliknutí na ikonu s šipkou, která je součásti boxu, se box zasune za pravý okraj okna. Případně lze box zvětšovat a zmenšovat. (obr. 7.)

Vyhledávání v textu je interaktivní a při psaní se hned zobrazuje výsledek v textu. Nalezený řetězec se podbarví žlutou barvou a pouze první nalezený se podbarví oranžovou barvou. Po vyhledání panel zobrazí počet nalezených řetězců a nabídne možnost mezi nimi přeskakovat. Kvůli ulehčení pro algoritmus je vyhledávání aktivní až od dvou zadaných znaků.(obr. 8.)



Obrázek 7. Teoretická část.

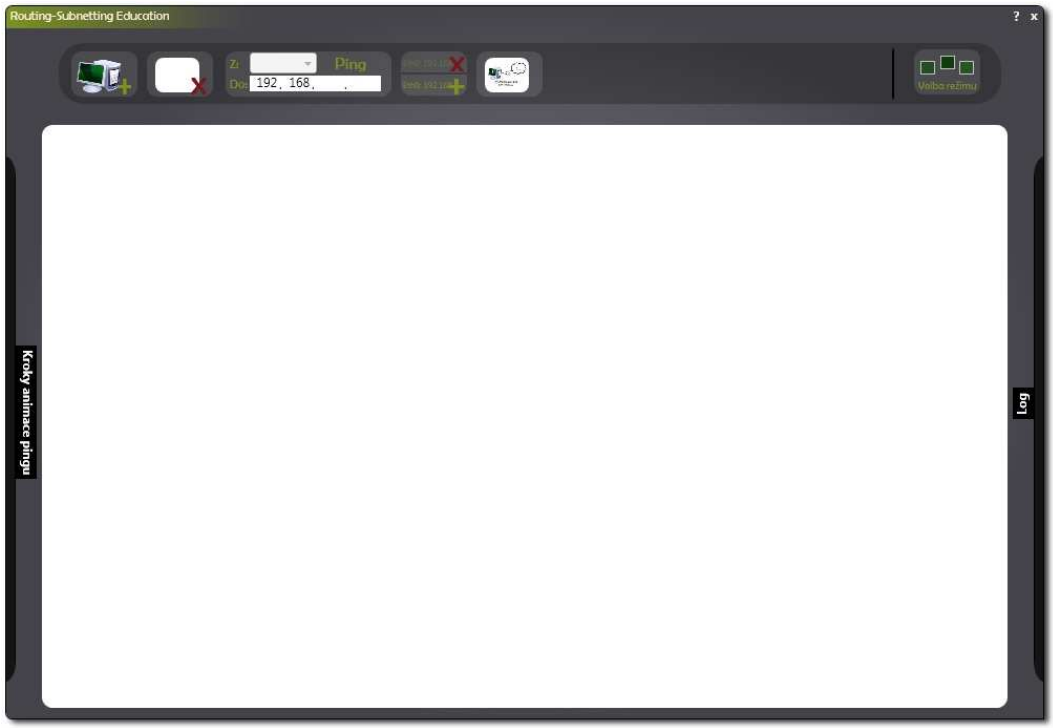


Obrázek 8. Vyhledávání v textu teorie.

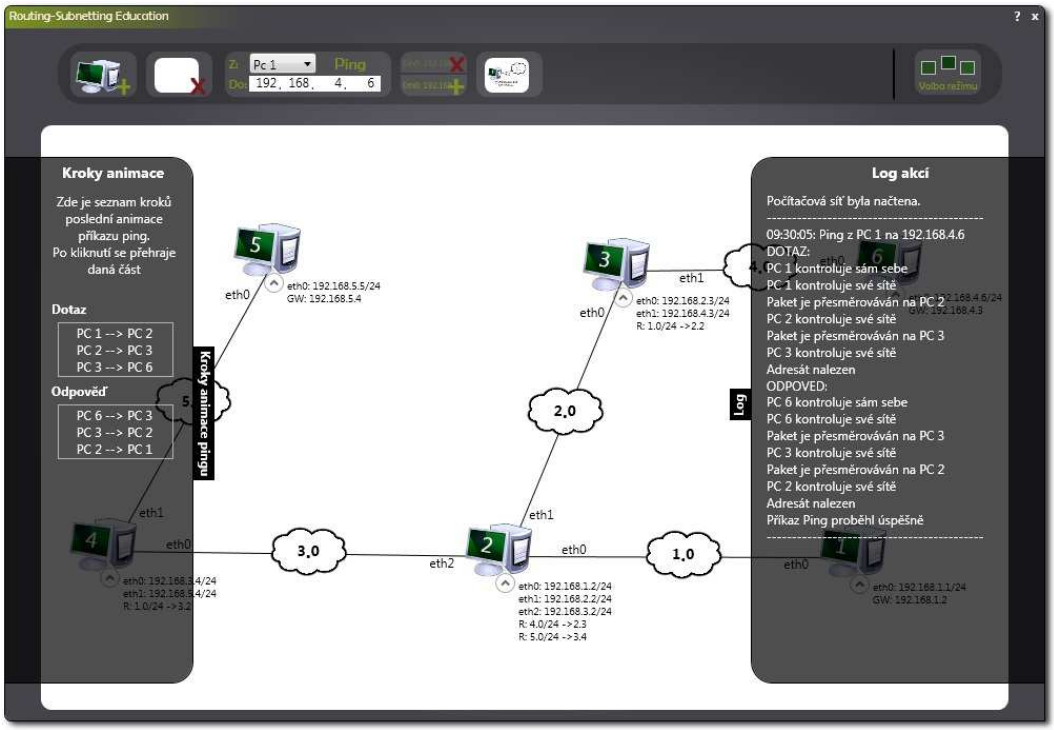
4.3.4. Praktická část

Praktická část tvoří v pořadí druhou část aplikace. Má nejvíce funkcí. Umožňuje vytvářet počítačovou síť, nastavovat jednotlivé počítače a testovat funkčnost komunikace uzlů.

Okno je tvořeno opět nástrojovou lištou v horní části. Zbytek okna zakrývá bílá plocha určená pro budování počítačové sítě. Po pravé a levé straně se nachází ještě textové ikony pro otevření výsuvného panelu. Pravý panel obsahuje logovací informace, zobrazuje změny v síťi a detailní výpis příkazu ping. Levý panel je aktivní pouze po spuštění příkazu ping, zobrazuje kroky animace, které lze postupně přehrát. (obr. 10.). Nástrojová lišta je tentokrát bohatší na vlastnosti. Pokud neberu v úvahu tlačítko pro změnu režimu, nabízí celkem 5 funkcí. První zleva slouží pro přidání prvků do sítě. Ikona zobrazuje PC a symbol +. Kliknutím se objeví na bílé ploše PC. Druhá možnost smaže veškerý obsah bílé plochy. Ikona je symbolizována bílou plochou a červeným křížkem. Další a největší tlačítko vyvolává vizualizaci příkazu ping. Tvoří jej seznam ID použitých počítačů v sítí a pole pro zadaní cílové adresy. Po kliknutí do libovolné plochy pozadí tlačítka se vyvolá akce. Podrobnosti vyvolané animací najdete na stránce 26. Další v pořadí je jediné tlačítko, které má dvě příbuzné funkce. Dělí se na spodní a horní část. Horní část skryje popisky s nastavením u počítačů v sítí. Naproti tomu spodní část je zobrazí. Poslední tlačítko lišty vyvolá dialogové okno obsahující uložené ukázkové sítě. Bez nutnosti vymýšlet vlastní síť tak můžete zkoušet nastavení na již vytvořené.

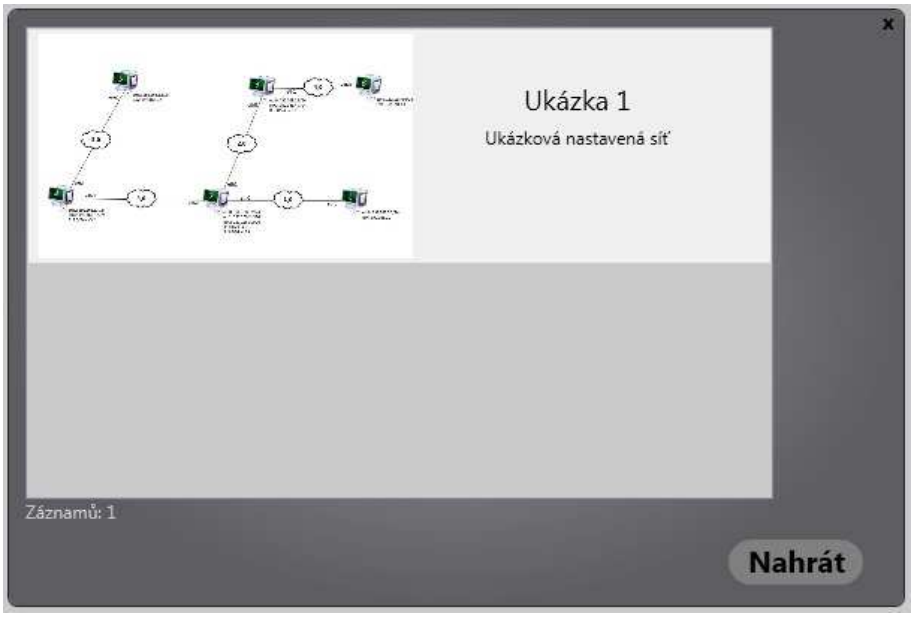


Obrázek 9. Praktická část



Obrázek 10. Praktická část - otevřené panely.

Dialogové okno zobrazuje veškeré sítě dostupné pro studenty (obr. 11.). Samotný seznam je tvořen obrázkem sítě a k názvu je připojen i popis. Pravým klikem myši na obrázek jej lze zvětšit a můžete tak vidět větší detaily. Pro navrácení zpět stačí kliknout levým tlačítkem myši. Při kliknutí na vybranou síť v seznamu jej nahrajete stiskem tlačítka "Nahrát". Předchozí síť se tak nahradí sítí vybranou.



Obrázek 11. Dialogové okno pro načtení sítě.

Uzel tvoří ikona počítače. Na monitoru ikony se zobrazuje ID uzlu. Toto ID je třeba vědět při používání příkazu ping, neboť značí počítač, ze kterého má algoritmus provádět příkaz. Pod ikonou je kulaté tlačítko pro zobrazení nebo skrytí popisu nastavení počítače. Pokud není počítač zapojen, je tato volba nedostupná. Při najetí kurzorem na ikonu se vpravo nahoře od ní zobrazí křížek. Ten slouží pro odstranění PC z plochy. Pro přesun ikony, stačí stlačit levé tlačítko a přetáhnout na požadované místo. Vše, co je potřeba se automaticky překreslí.

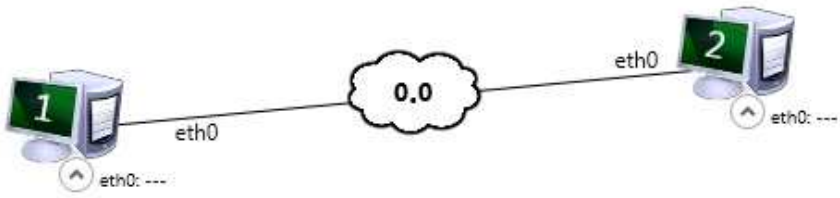
Obrázek 12. Ikona počítače reprezentující uzel v síti.

Samotné propojování počítačů se dělá pomocí pravého tlačítka myši. Režim

propojování se tedy aktivuje prvním pravým klikem na některou z ikon na ploše. Po přejetí na plochu se zobrazuje z místa kliku do aktuálního místa tenká čára. Tato čára se při najetí nad některou z ikon počítače ukotví a stane se tlustší. To značí možnost vytvořit spoj pravým klikem myši. Pokud kliknete při vytváření spoje do bílého plátna, režim propojování se tím vypne.

Obrázek 13. Režim propojování.

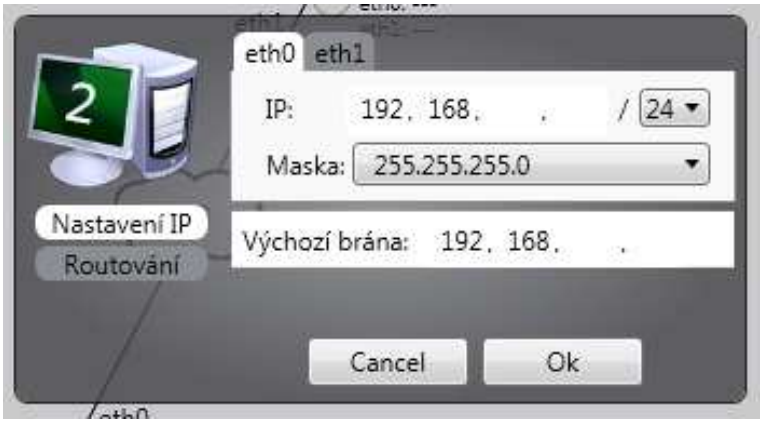
Po vytvoření spoje se objeví pár nových prvků. Jednak zůstane propojovací čára, u ikon počítače se zobrazí popisky, značící síťové zařízení, ze kterého spoj vychází a také se povolí a zobrazí popis nastavení. Uprostřed spoje je také ikona mráčku, ve které jsou 2 pole pro čísla. Ty reprezentují poslední 2 čísla adresy sítě. Celá tato ikona tak má pouze pomocný charakter. Student si do ní smí poznamenat, o jakou síť se jedná. Celá síť je tak přehlednější. To neplatí v testovací části, kde ikona mráčku tvoří součást zadání a není možno hodnoty měnit.



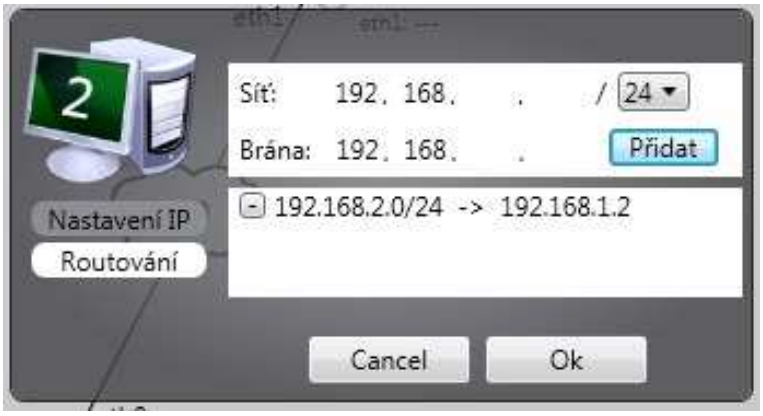
Obrázek 14. Dva propojené počítače.

Pro nastavení počítače slouží dialogové okno, které vyvoláte dvojklikem na ikonu počítače nebo klikem na popis nastavení. V levé části dialogového okna je zobrazena ikona nastavovaného počítače. Pod ní je přepínání režimu nastavování. První režim zobrazuje ve zbytku okna nastavení pro jednotlivé zařízení. Nahoře jsou záložky zařízení a pod ní příslušné nastavení. Nastavuje se IP a maska sítě. Pro masku sítě jsou dvě možnosti, jak ji nastavit. Jednak lze masku nastavit pomocí zkrácené formy za IP adresou nebo pomocí delší formy pod ní. Oba prvky jsou propojené a nedají se nastavit rozdílné hodnoty. Pod nastavením pro zařízení

se nachází nastavení výchozí brány. Pro usnadnění orientace v síti při nastavování lze s oknem tahem pohybovat po okně aplikace. V druhém režimu dialogové okno zobrazuje nastavení pro routování. Vrchní panel je určen pro přidávání hodnot do tabulky, která je pod ní. Záznamy z tabulky se odstraňují tlačítkem se znakem "mínus".

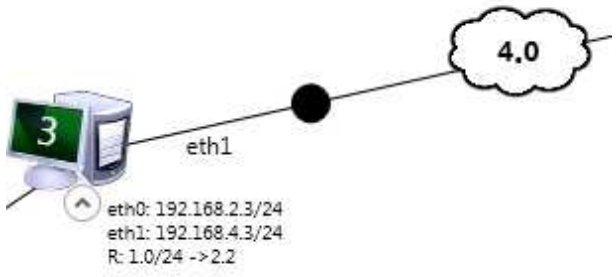


Obrázek 15. Dialogové okno pro nastavení počítače - rozhraní.



Obrázek 16. Dialogové okno pro nastavení počítače - routování.

Animace příkazu ping je tvořena kuličkou, která reprezentuje datový paket. Ten se pohybuje po síti a mění barvy v závislosti na tom, jakého je typu. Při vypuštění je barva černá. To značí dotaz a paket hledá adresáta. Při jeho nalezení se změní barva kuličky na oranžovou. Ta symbolizuje odpověď, která musí dorazit do místa startu příkazu. Při doražení se mění na barvu zelenou, což signalizuje nalezení cesty a tím tedy odpověď od hledaného počítače. Při dokončení animace se vždy zobrazí finální efekt. Ten spočívá v postupném zvětšování kuličky do ztracena. Efekt připomíná rozplynutí.



Obrázek 17. Zobrazení paketu, který hledá adresáta.

Obrázek 18. Finální efekt při úspěšném provedení příkazu.

Pokud je síť špatně nastavena, nebo prostě paket necestuje podle plánu, nebude finální efekt zelený, ale bude mít barvu podle fáze, ve které se právě nachází. Tak se lehce podle barvy zpozoruje, jestli je chyba při hledání cílové adresy nebo při hledání cesty zpět. Černý finální efekt tedy znamená nemožnost naleznout cílovou adresu a oranžový zase nemožnost doručení odpovědi do místa vypuštění.

4.3.5. Testovací část

Část testovací slouží pro přihlášení k testu, který automaticky po spuštění vyhledává (obr. 19.). Po připojení stačí jen čekat, až vyučující spustí test (obr. 20.). Spuštění testu se projeví otevřením části programu, podobné praktické části se zprávou o zadání v dialogovém okně. Nástrojová lišta nemá v tomto režimu žádnou funkci, která by modifikovala síť. Povoleno je pouze měnit nastavení.

Pro odevzdání práce slouží příslušné tlačítko na liště. Po odevzdání je test dokončen a uživatel smí program bezpečně zavřít.

Obrázek 19. Nalezení testu.

Obrázek 20. Připojení k testu.

4.4. Administrátorská část

Do administrátorské části se vstupuje pomocí skrytého tlačítka, které se zobrazí po najetí do levého spodního rohu hlavní nabídky. K přístupu musí uživatel znát heslo. Ve výchozím stavu je přihlašovací jméno „admin“ a heslo „admin123“. Pro změnu hesla stačí v administrátorské části kliknout na tlačítko „Změna hesla“ v pravém horním rohu vedle tlačítka pro nápovědu. Vyresetování hesla se provádí smazáním příslušného souboru (více v kapitole 5.1.).

V administrátorské části je nástrojová lišta rozdělena na dvě části oddělené svislou čarou. První část této lišty nabízí možné spustitelné režimy a část druhá je proměnlivá a nabízí aktuální nabídku.

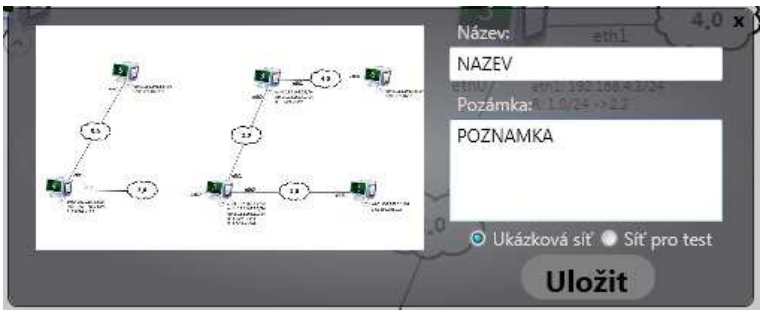
4.4.1. Správa teoretické části

První režim je správa textu. Vypadá stejně jako teoretická část, ale nástrojová lišta zde nabízí možnost odstraňovat kapitoly a možnost vyvolání dialogového okna pro přidání nové kapitoly neb podkapitoly. Volba, zda se bude jednat o kapitolu či podkapitolu, se zadává pomocí nabídky pod názvem kapitoly. Samotné přidávání textu je pouze navázání jména kapitoly na příslušný soubor formátu .rtf ve složce Docs/. Editace samotného textu se tak provádí pouhým editováním .rtf dokumentu.

Obrázek 21. Dialogové okno pro přidání textu.

4.4.2. Správa praktické části

Administrátorská praktická část se opět moc neliší od normální praktické části. Na liště přibylo tlačítko na vyvolání dialogového okna pro uložení síťě. Zde se rozlišuje, jestli má být síť určena pro studenty jako ukázková v praktické části, nebo pro administratora při tvorbě testu. Dále je pozměněno okno pro načítání sítě. To umožňuje přepínat mezi načítáním ukázkových sítí a testovacích sítí pomocí přepínače umístěného ve spodní části okna.

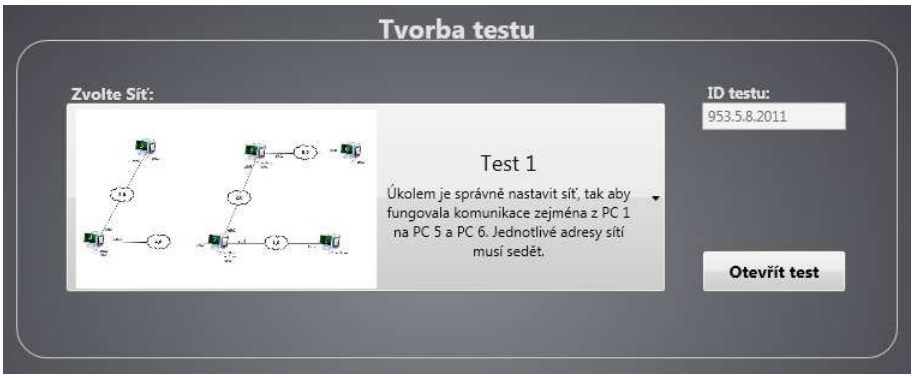


Obrázek 22. Dialogové okno pro ulození sítě.

4.4.3. Vytváření a správa testů

Poslední tlačítko první části lišty nabízí možnost tvorby a správy testu. Pokud není spušten žádný test, automaticky se po kliknutí zobrazí přehled vypracovaných testů. Jeliže test běží, zobrazí se okno se spuštěným testem. Mezi těmito volbami pak lze přepínat pomocí tlačítek v druhé části lišty.

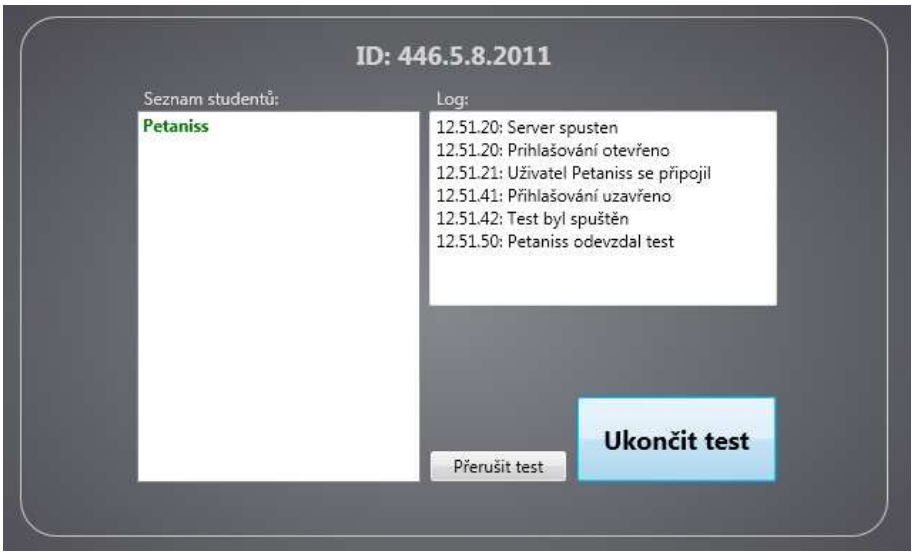
Pro tvorbu testu je nejprve nutné vybrat některou síť, která je určená pro test. Automaticky se vygeneruje ID testu složené z trojmístného vygenerovaného čísla a aktualního data. Pokračuje se tlačíkem "Otevřít test".



Obrázek 23. Tvorba testu.

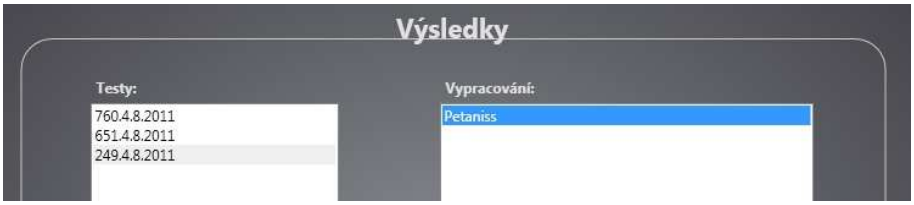
Následuje obrazovka s přehledem přihlášených studentů. V tomto okamžiku je spuštěno pouze přihlašování k testu, samotný test se odstartuje tlačítkem "Spustit

test". Přihlášení studenti jsou vypsáni v seznamu v levé části. Po odstartování se studentům zobrazí zadání a mohou pracovat. Vyučující má možnost libovolně překlikávat v režimech, které administrátorská sekce nabízí a spuštěný test přitom zůstane na pozadí a zobrazí se při návratu do dané sekce. Pokud jméno studenta v seznamu je zelenou barvu, znamená to, že dokončil a odevzdal svůj test. Pokud jsou tedy všichni v seznamu takto označeni, test je kompletně hotov a můžeme jej ukončit příslušným tlačítkem. Pokud test přerušíme dřív, uloží se pouze testy, které jsou odevzdané. Test je možné i nasilně přerušit. Při přerušení nedojde k žádnému uložení a test je úplně zrušen.



Obrázek 24. Ukázka dokončeného testu.

Pro prohlížení vypracovaných testů stačí kliknout v seznamu testů na jeho název. Poté se v seznamu napravo zobrazí jednotlivé vypracování označené jménem studenta. Tlačítkem "Zobrazit" se otevře vypracovaný test označeného studenta. Pro návrat zpět slouží šipka na nástrojové liště, která v této chvíli převezme místo po tlačítku na výběr režimu.



Obrázek 25. Okno s vypracovanými testy.

5. Programátorská příručka

Program je psán v jazyce C#.

5.1. Souborová a adresářová struktura programu

K programu patří celkem 3 složky.

- Data - složka obsahuje soubory obsahující heslo, nastavení teoretické části a vypracované testy.
- Docs - složka obsahuje .rtf dokumenty, které se zobrazí při vkládání nového textu do teoretické části a program se na ně dále odkazuje při otevření daného textu.
- UlozeneSite - obsahuje všechny uložené ukázkové a testovací sítě.

Ve složce Data jsou dále tyto soubory:

- Pass - Tento soubor je nositelem přihlašovacích informací. Pokud soubor neexistuje, je při přihlášení vytvořen znovu a tím se heslo vyresetuje na výchozí.
- docslist - V tomto souboru jsou uloženy vztahy mezi názvem kapitoly a .rtf dokumentem, který má otevřít. Pokud soubor neexistuje, je při přidání kapitoly vytvořen soubor nový.
- Testy - Soubor má v sobě uloženy všechny vypracované testy. Pokud neexistuje, je při uložení testu vytvořen nový.

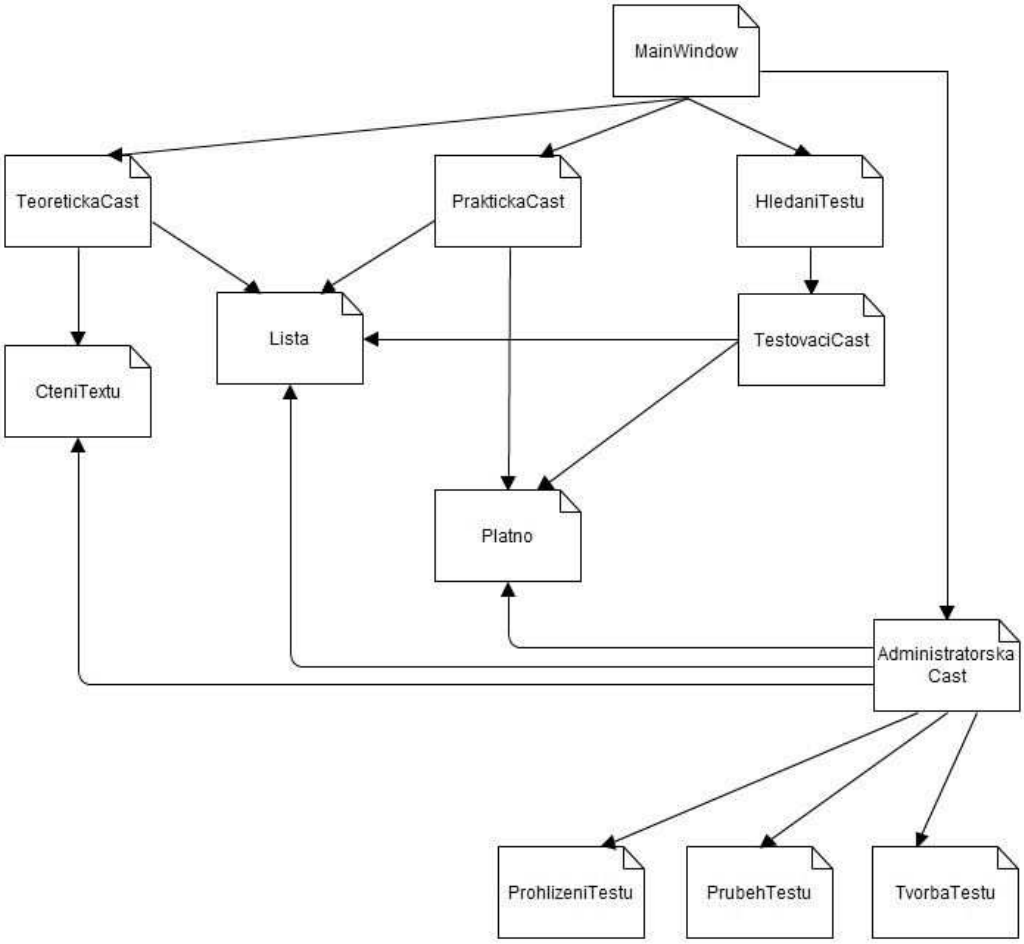
5.2. Základní návrh aplikace

Hlavní třída programu je MainWindow. Tato třída se stará o zobrazení hlavního okna, volbu režimu, změnu velikosti okna a přepínání režimu. Při spuštění zobrazí objekt „Selector“, který tvoří samotnou nabídku režimů. Při výběru režimu Selector předá zprávu třídě MainWindow, ta zruší Selector a zobrazí příslušný objekt. Program se tímto dělí na další části. (Podrobněji ukazuje diagram tříd na obr. 26.)

Následuje seznam objektů spuštěných z MainWindow. Každý z nich má na starosti zobrazit nástrojovou lištu a dál objekty potřebné pro danou část.

- TeoretickaCast - stará se o zobrazení objektu pro čtení textu (CteniTextu.xaml.cs) a komunikaci s lištou.
- PraktickaCast - má na starosti zobrazit objekt pro tvoření sítě (Platno.xaml.cs) a komunikaci s lištou.

- HledaniTestu - má na starosti najít testový server a po spuštění otevřít TestovaciCast.xaml.cs.
- AdministratorskaCast - řídí a spouští všechny objekty administrátorské části.



Obrázek 26. Diagram tříd.

5.3. Nástrojová lišta

Nástrojová lišta tvoří hlavní ovládací panel. Při inicializaci je lišta prázdná až na tlačítko pro volbu režimu. Pro každé tlačítko je příslušná metoda pro přidání na lištu (např. PridejPC(), PridejPingPanel() ...). Při zavolání metody se na lištu přidá prvek a přidá se událost reagující na kliknutí. Tyto události potom vyvolává přímo objet Lista. Objekt, který má na starosti komunikaci mezi lištou a daným objektem, zachytává tyto události a reaguje na ně. Lišta má celkem 20

různých událostí. Všechny prvky, které je možné přidat na lištu jsou v namespace „PrvkyListy“.

5.4. Počítačová síť

Velkou třídu, jejíž objekt se stará o vykreslování počítačové sítě a simulaci příkazu ping, je Platno.xaml.cs. Samotné algoritmy však nejsou přímo v této třídě, ale jsou pro přehlednost rozděleny do několika části. Všechny třídy, které nemají grafickou podobu, jsou obsaženy v knihovně Logistic. Tam také spadají třídy, které se starají o výpočet pozic prvků sítě(JadroVykreslovaci.cs) a třída, která je nosičem všech informací o síti(JadroData.cs). Pro metodu ping je pak třída JadroLogikaSite.cs.

Třída Platno se tak stará pouze o vykreslování prvků na panel Canvas a na základě událostí lišty vyvolává akce v Jádrech.

Následuje výpis hlavních metod třídy JadroVykreslovaci.cs:

- PridatPC - Pro přidání PC do sítě. Vytvoří instanci objetu PC(MyPC) a přes událost předá tuto instanci a souřadnice objetu Platno.
- ZmenitPolohu - Stará se o přesouvání objektů po panelu Canvas, mění pozice a přepočítává souřadnice všech objektů, co je třeba překreslit. O všech objektech informuje opět přes událost.
- pridatCaru - Aktivuje se při pravým kliku na ikonu PC v síti. Podle toho jestli je již aktivován režim kreslení čar, začne kreslit čáru nebo kreslení čar dokončí.
- PrekresliPripojeneCary, PrekresliSubsite, PrekresliLabely - Metody spojené s překreslováním. Volají se v těle metody ZmenitPolohu.

Třída JadroData obsahuje strukturu, která uchovává všechny potřebné informace o síti. Metody v této třídě do struktury data přidávají, čtou nebo odstraňují. Některé z metod pro čtení dat:

- VratZacinajiciCary - Metoda vrátí všechny čáry, které začínají v PC na vstupu.
- VratKonciciCary - Vrací naopak čáry, které končí v PC na vstupu.
- VratPcZId - vrátí instanci objektu MyPC na základě čísla ID, které si metoda vyhledá ve struktůře.

Některé z metod pro zápis dat:

- PridatPc - přidá PC do struktury.
- PridatStartCaru - přidá k danému PC záznam o čáře.

Některé z metod pro odebírání dat:

- SmazatVsechnyLine - Odstraní ve struktuře všechny výskyty čáry dané na vstupu.
- SmazatPc - Smaže záznam o daném PC.

5.5. Síťová komunikace

Pro síťovou komunikaci je v této aplikaci použit .NET Remoting. Serverová část je napsána v třídě Server.cs a klientská v Client.cs. Pro komunikaci potřebujeme ještě sdílený objekt, který musí být odvozen od MarshalByRefObject, aby bylo možné objekt předávat odkazem přes .NET Remoting. Důležité je, aby každý objekt, který je posílán skrz kanál, byl serializovatelný.

V této aplikaci tvoří sdílený objekt instance třídy Server.cs. Připojení klienti tak mohou přímo volat metody v serveru prostřednictvím interface (IServerObject), který je zvlášť v knihovně sdileno.dll. To proto, aby klientská část aplikace nepotřebovala zdrojové kódy serveru.

Výhoda tohoto řešení spočívá v aktualizacích, protože klient má přístup pouze k interface serveru, můžeme změnit obsah metod na serveru a přitom na klientské části budou tyto metody stále fungovat bez změny aplikace. Nanejvýš se pouze upraví knihovna obsahující interface k serveru.

Komunikace v druhém směru je řešena pomocí událostí. Nutné je opět použít prostředníka, protože zde platí to stejné jako v případě serveru. Klient nemůže serveru říct, kterou metodu zaregistrovat jako posluchače události, když tuto metodu server nezná. Pokud ale klient serveru přiřadí k této události metodu z proxy(EventProxy.cs) objektu, který dále vyvolá událost na kterou je už přihlášena metoda klienta, zprovozníme tak komunikaci v druhém směru.

Ukázka spuštění serveru:

```
Hashtable props = new Hashtable();
props["port"] = port; //port na kterém běží server
props["name"] = serverURI; //název (test.Rem)

//nastavení parametrů
BinaryServerFormatterSinkProvider serverProv =
    new BinaryServerFormatterSinkProvider();
serverProv.TypeFilterLevel =
    System.Runtime.Serialization.Formatters.TypeFilterLevel.Full;

//vytvoří instanci TCP kanálu
```

```
TcpChannel serverChannel = new TcpServerChannel(props, serverProv);

try
{
   // registrace kanálu
   ChannelServices.RegisterChannel(serverChannel, false);
   // začne sdílet instanci tohoto objektu na adrese serveru + /name
   internalRef = RemotingServices.Marshal(this, props["name"].ToString());
}
catch (Exception ex)
{
                //Chyba
}
```

Ukázka připojení klienta:

```
// Vytvoří instanci kanálu
TcpChannel tcpChannel = new TcpChannel();
// zaregistruje
ChannelServices.RegisterChannel(tcpChannel, false);

// přiřadí vzdálenou instaci objektu k remoteServer.
IServerObject  remoteServer =
(IServerObject)Activator.GetObject(typeof(IServerObject), serverURI);
```

K připojování lze přidat registraci různých událostí. Například v této aplikaci registrace události pro příjem objektu zprávy přes proxy vypadá takto:

```
eZpravaDelegate delegatZpravaRemote =
new eZpravaDelegate(eventProxy.LocallyHandleZpravaClientum);
remoteServer.eZpravaClientum += delegatZpravaRemote;
```

Závěr

Cílem této práce bylo vytvořit výukový program pro výuku na střední škole s části teoretickou, procvičovací a testovací. Procvičovací část má podobu simulace počítačové sítě. Část testovací tvoří test, který probíhá přes síť prostřednictvím .NET Remoting. Tuto technologii jsem se díky této práci naučil využívat. Stejně tak jsem si osvojil i základy grafického subsystému WPF. Cíl práce byl tedy splněn. Projekt by měl ulehčit výuku dané problematiky pomocí názorné výuky.

Conclusions

The goal of this project was to create education programe for teaching in high school with teorethical, excercising and testing part. Excercising part is made by simulation of computer network. Testing part is made by test, which is running on network via NET Remoting. I learned to use this technology thanks to this project. Also I learned basics of graphic subsystem WPF. Goal of the project was achieved. Project should make teaching less difficult with illustrative lessons.

Reference

[1] Christian Nagel, Bill Evjen, Jay Glynn, Karli Watson, Morgan Skinner, Allen Jones.

C# 2005 Programujeme profesionálně

Computer Press, a.s., Brno, 2006.

[2] Charles Petzold.

Mistrovství ve Windows resentation Foundation

Computer Press, a.s., Brno, 2008.

[3] John Sharp.

Microsoft Visual C# 2010: Krok za krokem

Computer Press, a.s., Brno, 2010.

[4] Ukázka .NET Remoting v C#.

`http://www.codeproject.com/kb/ip/net_remoting.aspx`

[5] Teorie k počítačovým síťím.

1. `http://cs.wikipedia.org/wiki/počítačová_síť`
2. `http://cs.wikipedia.org/wiki/ip_adresa`
3. `http://cs.wikipedia.org/wiki/maska_sítě`
4. `http://cs.wikipedia.org/wiki/routování`

A. Obsah přiloženého CD

`bin/`
: Obsahuje složku s programem, který je spustitelný přímo z CD. Při takovémto spuštění není možno ukládat. Administrátorská část aplikace tak pro plnou funkčnost musí běžet přímo z pevného disku. Dále obsahuje archiv „program.rar“ ve kterém je složka zabalena a instalátor pro .NET Framework 4.0.

`doc/`
: Dokumentace práce ve formátu PDF, PS a zdrojové soubory nutné k vytvoření dokumentace.

`src/`
: Kompletní zdrojové texty programu.

`readme.txt`
: Instrukce pro instalaci a spuštění programu, včetně požadavků pro jeho provoz.