

Aplikace typu klient server s výstupem na WWW

Client server application with web output

Bc. Richard Vytásek



ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Richard VYTÁSEK**
Osobní číslo: **A09495**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Aplikace typu klient server s výstupem na WWW**

Zásady pro vypracování:

1. Vytvořte síťovou aplikaci, která bude založena na architektuře klient-server.
2. Přenášená data po síti uložte do databáze.
3. Data z databáze zobrazte on-line na webu.
4. Umožněte archivaci dat na serveru a pozdější zobrazení přes WWW.
5. Programování C/C++ (sokety, vláknové programování) – případně využití jiných programovacích jazyků pro specifické úlohy.
6. Vytvořte finální sestavení z jednotlivých částí aplikace a navrhnete další funkce.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. DREPPER, Ulrich. Linux Threads Programming: Linux Concurrency And Performance. [s.l.] : [s.n.], 2007. 400 s. ISBN 0131487264.
2. GAY, Warren. Linux Socket Programming by Example. [s.l.] : [s.n.], 2000. 576 s. ISBN 0789722410.
3. GILMORE, W. Jason. Velká kniha PHP a MySQL 5 : Kompendium znalostí pro začátečníky i profesionály. [s.l.] : Zoner Press, 2007. 864 s. ISBN 8086815536.
4. PRATA, Stephen. Mistrovství v C++. [s.l.] : COMPUTER PRESS, 2007. 1120 s. ISBN 8025117499.
5. Bradford, N.; Buttlar, D. & Farrell, J. P.: Pthreads Programming A POSIX Standard for Better Multiprocessing. O'Reilly, 1996.
6. MATLAB Inc.. MATLAB C and Fortran API reference. The Mathworks Inc., Natick, USA, 2008.
7. Microsoft Corporation. Windows Sockets 2.[online] Available from: [http://msdn.microsoft.com/en us/library/ms740673\(VS.85\).aspx](http://msdn.microsoft.com/en us/library/ms740673(VS.85).aspx).

Vedoucí diplomové práce:

doc. Ing. Martin Sysel, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

24. února 2011

Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce je vytvoření finálního sestavení aplikace SimWebLink, která je složena z klientské části, serverové části a webového rozhraní pro vzdálený přístup. Aplikace slouží ke vzdálenému ovládání úloh v programu Simulink a presentaci naměřených hodnot ve webovém rozhraní.

V textu jsou popsány použité technologie, struktura aplikace a postup její instalace a konfigurace.

Klíčová slova: Simulink, S-function, platforma .NET, socket, WinSock

ABSTRACT

Goal of this work is to create final assembly of the SimWebLink application, which consist of client part, server part and web interface for remote access. The application itself serves for remote control of Simulink software tasks and allows presentation of measured values in web interface.

This text describes used technologies, structure of the application and the process of its installation and configuration.

Keywords: Simulink, S-function, .NET platform, sockets, WinSock

Rád bych tímto poděkoval vedoucímu mé práce panu doc. Ing. Martinu Syslovi, Ph.D. za jeho čas a cenné připomínky k vypracování této diplomové práce.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST	10
1 POUŽITÉ TECHNOLOGIE.....	11
1.1 MATLAB	11
1.2 SIMULINK	11
1.2.1 S-Funkce.....	12
1.3 PLATFORMA . NET	12
1.3.1 Jazyk C#	12
1.3.2 Common Language Infrastructure.....	12
1.4 WEBOVÉ TECHNOLOGIE.....	14
1.4.1 Php.....	14
1.4.2 Mysql.....	14
1.5 JAVASCRIPT.....	15
1.5.1 Ajax	15
2 PROGRAMOVÁNÍ SÍŤOVÝCH APLIKACÍ	16
2.1 MODEL TCP/IP	16
2.1.1 Spojová vrstva.....	16
2.1.2 Síťová vrstva	17
2.1.3 Transportní vrstva	17
2.1.4 Aplikační vrstva	17
2.2 IP PROTOKOL	17
2.3 TCP PROTOKOL	18
2.4 UDP PROTOKOL	19
2.5 SOCKETY V OPERAČNÍCH SYSTÉMECH WINDOWS	19
2.5.1 Blokuující a neblokuující síťové operace.....	20
2.6 PROGRAMOVÁNÍ SOCKETŮ V PROSTŘEDÍ .NET	20
2.6.1 Třída TcpClient	21
2.6.2 Třída TcpListener	21
2.6.3 Využití datových proudů při komunikaci přes Tcp.....	22
2.6.4 Třída UdpClient	22
II PRAKTICKÁ ČÁST	23
3 APLIKACE SIMWEBLINK.....	24
3.1 STRUKTURA PROGRAMU	24
3.1.1 S- function netout4	24
3.1.2 SimWebLinkClient – klientská aplikace.....	25
3.1.3 SimWebLinkServer - serverová aplikace.....	25
3.1.4 SimWebLinkWeb – webové rozhraní	26
3.1.5 Databáze s úlohami	26

3.2	SCHÉMA KOMUNIKACE	27
3.3	KOMUNIKACE JEDNOTLIVÝCH ČÁSTÍ	28
3.3.1	SimWebLinkClient a MATLAB Engine.....	28
3.3.2	S-funkce netout4 a SimWebLinkClient	29
3.4	TYPICKÝ PRŮBĚH UŽITÍ.....	30
4	SIMWEBLINKCLIENT – PROGRAMÁTORSKÁ ČÁST	31
4.1	STRUKTURA APLIKACE	31
4.1.1	Třída Program	31
4.1.2	Třída MatlabEng	32
4.1.3	Třída ServerConn	33
4.1.4	Třída ClientAcceptor.....	35
4.1.5	Třída handleClient.....	37
4.2	VYBRANÉ KLÍČOVÉ ČÁSTI PROGRAMU	38
4.2.1	Parsování XML souboru	38
4.2.2	Příjem klientů	39
5	SIMWEBLINKCLIENT-ZPROVOZNĚNÍ.....	40
5.1	S-FUNKCE NETOUT4	40
5.1.1	Nastavení nástroje Mex	40
5.1.2	Kompilace	40
5.1.3	Vložení S-funkce do schématu.....	42
5.2	SIMWEBLINKCLIENT	45
5.3	SIMWEBLINKCLIENT JAKO SLUŽBA.....	46
6	ZPROVOZNĚNÍ DATABÁZE ÚLOH	48
7	DALŠÍ MOŽNÝ VÝVOJ APLIKACE.....	49
	ZÁVĚR	50
	ZÁVĚR V ANGLIČTINĚ.....	51
	SEZNAM POUŽITÉ LITERATURY.....	52
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	54
	SEZNAM OBRÁZKŮ	55
	SEZNAM TABULEK.....	56
	SEZNAM PŘÍLOH.....	57

ÚVOD

Cílem této práce je tvorba síťové aplikace SimWebLink, založené na architektuře klient-server. Data přenášená po síti jsou ukládána do databáze, odkud k nim následně přistupuje webové rozhraní. Aplikace SimWebLink umožňuje vzdálené ovládání schémat v programu Simulink a následné zobrazení výsledků on-line v grafické podobě na webu.

Simulink je nadstavba programu MATLABu pro simulaci a modelování dynamických systémů. Přestože Simulink má své vlastní grafické rozhraní, neumožňuje vzdálené ovládání. Tvorba vzdáleného způsobu kontroly byla hlavní motivací pro vznik této práce.

Mezi výhody vzdáleného ovládání patří možnost ovládat úlohy z jakéhokoliv počítače připojeného do sítě, ovládání úloh na několika počítačích z jednoho místa. Další nespornou výhodou je schopnost aplikace archivovat naměřená data na serveru, kde jsou později snadno přístupné. Aplikace umožňuje ke každé úloze nastavit hesla pro dvě různé úrovně oprávnění - prohlížení a ovládání.

Tato práce částečně navazuje na práci mého kolegy, Aleše Holíka, který vytvořil serverovou část aplikace. Cílem této práce byla tvorba klientské části, vytvoření finálního sestavení z jednotlivých částí aplikace a odstranění problémů, které nastaly.

I. TEORETICKÁ ČÁST

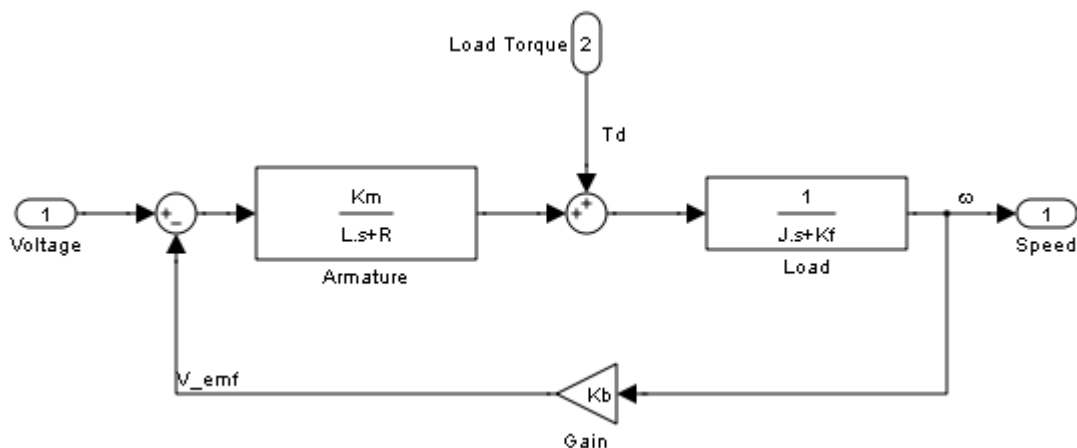
1 POUŽITÉ TECHNOLOGIE

1.1 MATLAB

MATLAB je programové prostředí sloužící k vědeckotechnickým výpočtům, počítačovým simulacím, modelování, měření a zpracování signálů. Název programu vznikl zkrácením slov Matrix Laboratory, což napovídá že se jedná o maticově orientovaný systém. To znamená, že základním objektem je matice a systém podporuje všemožné operace s nimi. Tyto operace spolu s dalšími funkcemi jsou nazývány M-funkce. Součástí prostředí jsou rozsáhlé knihovny vnějších M-funkcí, takzvané toolboxy. Jednotlivé toolboxy je možno zvolit při instalaci. [1][2]

1.2 Simulink

Simulink je nadstavba MATLABu sloužící k simulaci a modelování dynamických systémů. K tomu využívá algoritmy MATLABu pro numerické řešení nelineárních diferenciálních rovnic. Umožňuje uživateli rychle a snadno vytvářet modely dynamických soustav ve formě blokových schémat a rovnic. Výhodou je jeho otevřenost umožňující vytváření nových prvků. [1][2]



Obr. 1: Příklad schématu vytvořeného v Simulinku

1.2.1 S-Funkce

Mimo vestavěných funkcí umožňuje MATLAB i rozšíření o funkce a bloky napsané v jiných programovacích jazycích, typicky v jazyku C. Výhodou tohoto řešení je rychlejší běh díky kompilaci zdrojového kódu a přístup k prostředkům a rozhraním, ke kterým by jinak nebylo možné se z MATLABu dostat. Ke kompilaci zdrojového kódu slouží příkaz mex. Součástí MATLABu je i sada vzorových příkladů a šablon, díky kterým si je možno vcelku rychle osvojit vývoj těchto rozšíření. [1][2]

1.3 Platforma .NET

.NET je soubor softwarových technologií a knihoven vytvořený s cílem zjednodušit tvorbu programových aplikací. Prostředí obsahuje širokou řadu knihoven pro zjednodušení běžných úkonů. Platforma .NET nevyžaduje použití konkrétního programovacího jazyka. Výsledný kód v jakémkoli jazyku se při kompilaci vždy přeloží do mezijazyka Common Intermediate Language. Nejpoužívanějším programovacím jazykem ve spojení s touto platformou je jazyk C#. [4]

1.3.1 Jazyk C#

Jazyk C# patří mezi vysokoúrovňové objektově orientované programovací jazyky. Byl vyvinut společností Microsoft společně s platformou .NET Framework. Jeho Syntaxe byla inspirována jazyky C++ a Java. Jazyk je schválen jako standart dle komisí ECMA (ECMA-334) a ISO (ISO/IEC 23270). Možnosti využití tohoto jazyka jsou velmi široké, lze jej využít k tvorbě klasických desktopových aplikací, webových aplikací a služeb, ale i softwaru pro mobilní aplikace. Při návrhu tohoto jazyka byl kladen důraz na zohledňování struktury Common Language Infrastructure (CLI), se kterou je používán.[4][13]

1.3.2 Common Language Infrastructure

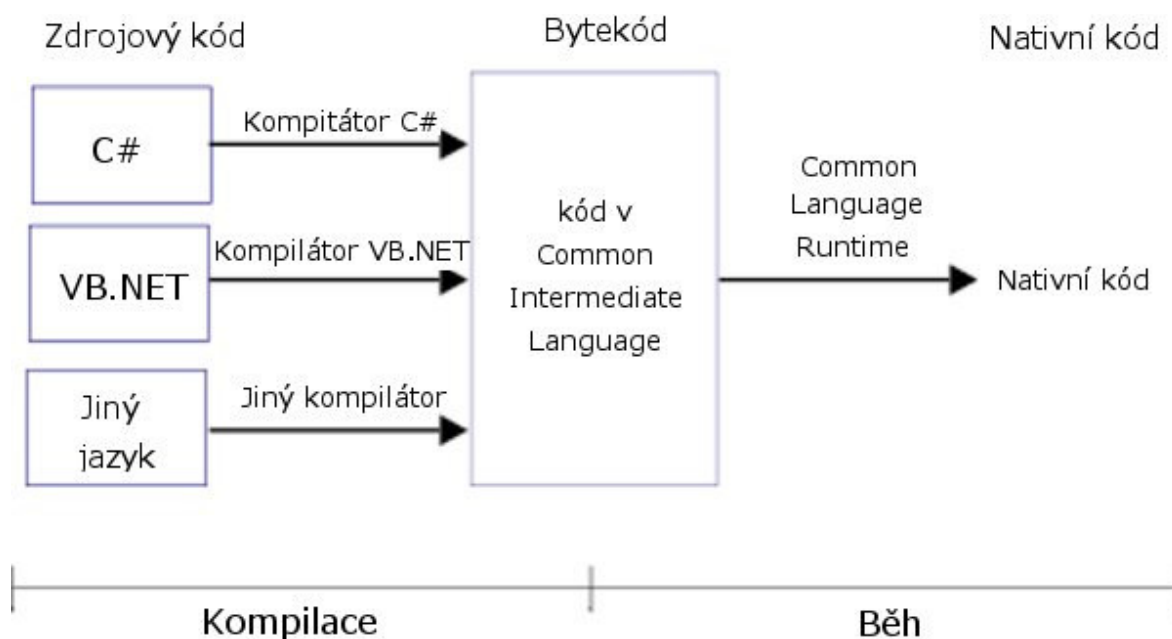
Common Language infrastructure je otevřená specifikace, na jejímž vývoji se podílela řada firem v čele se společností Microsoft. Slouží k popisu vlastností spustitelného kódu a prostředí pro jeho běh (anglicky Runtime Environment), jež formují jádro .NET frameworku od Microsoftu a open source implementací Mono a Portable.NET . Cílem specifikace je definovat prostředí umožňující použití více vysokoúrovňových programovacích jazyků na různých výpočetních platformách bez potřeby jejich úpravy pro

specifické architektury. Tato specifikace byla přijata jako standart roku 2001 organizací ECMA a roku 2003 organizací ISO. [14]

Specifikace mimo jiné popisuje základní čtyři hlediska:

- **The Common Type System (CTS)** — sada datových typů a operací, které jsou společné pro všechny programovací jazyky vyhovující CTS.
- **Metadata** — umožňuje snadno pracovat s kódem napsaném ve vývojáři neznámém programovacím jazyku, tím že informace o vnitřní struktuře programu popisuje jazykově nezávislým způsobem
- **Common Language Specification (CLS)** — Obsahuje základní pravidla, jež musí splňovat každý jazyk vyhovující CLS. Tím je umožněna vzájemná spolupráce s ostatními jazyky.
- **Virtual Execution System (VES)** — slouží k zavádění a spuštění programů, přičemž používá metadata k zajištění spolupráce mezi odděleně vytvořených kusů kódu za běhu programu.

Zdrojové kódy v jednotlivých jazycích jsou nejprve zkompileovány do formátu Common Intermediate Language. Tento bytekód je poté na různých platformách spouštěn v prostředí zvaném Common Language Runtime, které funguje jako kompilátor typu Just-in-time, tedy překlad kódu za běhu programu. Tímto je dosaženo platformní nezávislosti. Celý proces je znázorněn na obrázku č. 2.



Obr. 2: Kompilace do Common Intermediate Language a běh v Common Language Runtime

1.4 Webové technologie

1.4.1 Php

PHP je skriptovací jazyk vyvíjený od roku 1994 a původně navržený pro vytváření dynamických webových stránek. Postupem času přibýlo i rozhraní příkazové řádky, které může být použito i v klasických aplikacích. PHP je vydáván jako volně šiřitelný software s otevřeným zdrojovým kódem. Vývoj zajišťuje organizace PHP Group. Mezi jeho výhody lze zařadit platformní nezávislost a rozšířenost kvalitní dokumentace. Nevýhodou je naopak poměrně nekonzistentní vývoj, kdy není zaručena kompatibilita aplikací mezi jednotlivými verzemi.[7]

1.4.2 Mysql

MySQL je databázový server založený na jazyce SQL. Je rovněž dostupný jako open source, tedy volně šiřitelný software. Výhodou je také podpora všech hlavních platform, relativní jednoduchost, solidní výkon a kompatibilita s jinými systémy. MySQL se prosadilo jako universální řešení používané v naprosté většině internetových projektů. Vývoj je pod taktovkou společnosti Oracle.[3]

1.5 Javascript

Javascript je skriptovací jazyk používaný na mnoha webových stránkách. Na rozdíl od php neběží na webovém serveru, ale přímo u klienta v prohlížeči, čímž umožňuje bezprostředně reagovat na různé události. Tento jazyk vychází z mnoha programovacích jazyků, důraz při jeho vývoji byl kladen na jednoduchost a srozumitelnost i pro méně pokročilé programátory. I přes svůj název nijak nesouvisí s jazykem Java. Javascriptové kódy se na webu objevují už od roku 1995, kdy byla jeho podpora oficiálně implementována do prohlížeče Netscape Navigátor. [12]

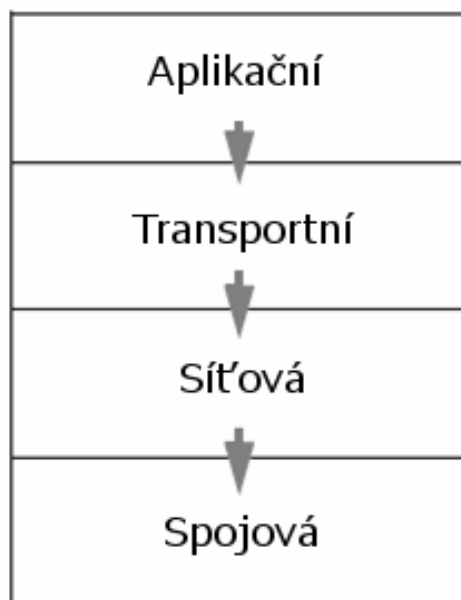
1.5.1 Ajax

AJAX je označení pro technologie vývoje webových aplikací, které jsou interaktivní, tedy dokážou změnit svůj obsah bez znovunačtení. Název je zkratkou Asynchronous Javascript and XML. Jak název napovídá, k vývoji aplikací je využito Javascriptu a rozhraní XMLHttpRequest. Javascript se zde využívá pro zobrazení a dynamickou změnu prezentovaných informací. Rozhraní XMLHttpRequest umožňuje komunikaci mezi klientským prohlížečem a serverem. [12]

2 PROGRAMOVÁNÍ SÍŤOVÝCH APLIKACÍ

2.1 Model TCP/IP

Síťový model TCP/IP slouží k popisu pravidel síťových propojení a jejich komunikace. Model je často přirovnáván k dalšímu síťovému modelu, ISO OSI, který byl původně zamýšlen jako návrhový. Oba síťové modely se skládají z několika vrstev, přičemž každá vrstva má definované jiné služby komunikuje pouze se sousedními vrstvami. Model TCP/IP obsahuje čtyři vrstvy: Spojovou, Síťovou, Transportní a Aplikační.[6] [11]



Obr. 3: Model TCP/IP

2.1.1 Spojová vrstva

Tato nejnižší vrstva se stará o ovládání dílčích přenosových cest spojení a zejména o přímé vysílání a příjem datových paketů. Vrstva byla navržena tak, aby byla nezávislá na použité přenosové technologii.

2.1.2 Síťová vrstva

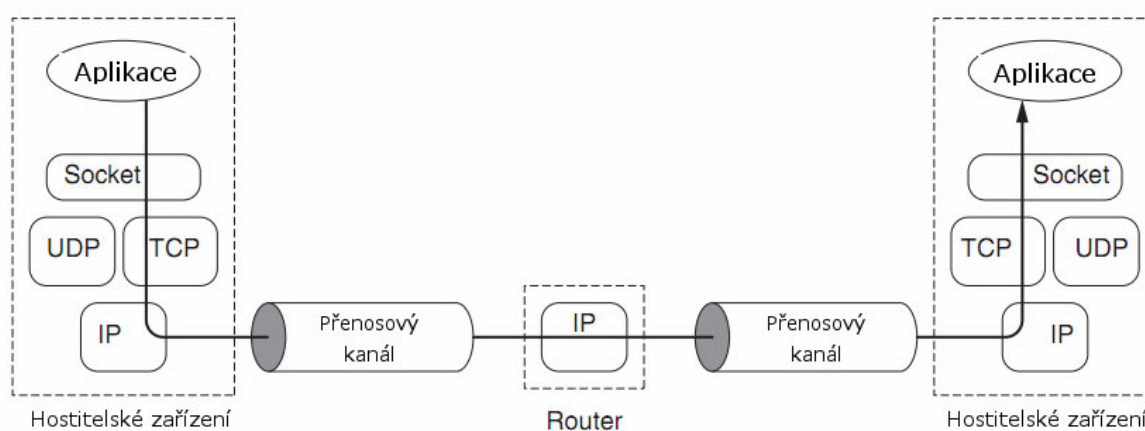
Tato vrstva má za úkol se postarat o to, aby byly jednotlivé pakety doručeny až ke svému konečnému příjemci. K tomuto účelu je využito prvků zajišťujících adresaci a směrování. Nejrozšířenějším protokolem této vrstvy je IP protokol.

2.1.3 Transportní vrstva

Cílem této vrstvy je především logické spojení koncových aplikací a zajištění přenosu dat mezi nimi. Nejčastěji používané protokoly této vrstvy jsou TCP (Transmission Control Protocol) a UDP (User Datagram Protocol).

2.1.4 Aplikační vrstva

Cílovými prvky tohoto spojení jsou jednotlivé aplikace používající ke komunikaci vlastní protokoly jako například http nebo FTP.



Obr. 4: Schéma komunikace přes TCP/IP

2.2 IP protokol

Protokol IP je protokolem síťové vrstvy modelu ISO/OSI. Jeho úkolem je dopravovat data na místo jejich určení, kterým je síťové zařízení. Každé zařízení je jednoznačně identifikováno IP adresou. Data se posílají v blocích označených jako datagramy. Každý z datagramů putuje sítí naprosto nezávisle a není zaručeno jejich doručení. Existují různé verze IP protokolu. V současné době je nejrozšířenější protokol verze 4, kde je IP adresa

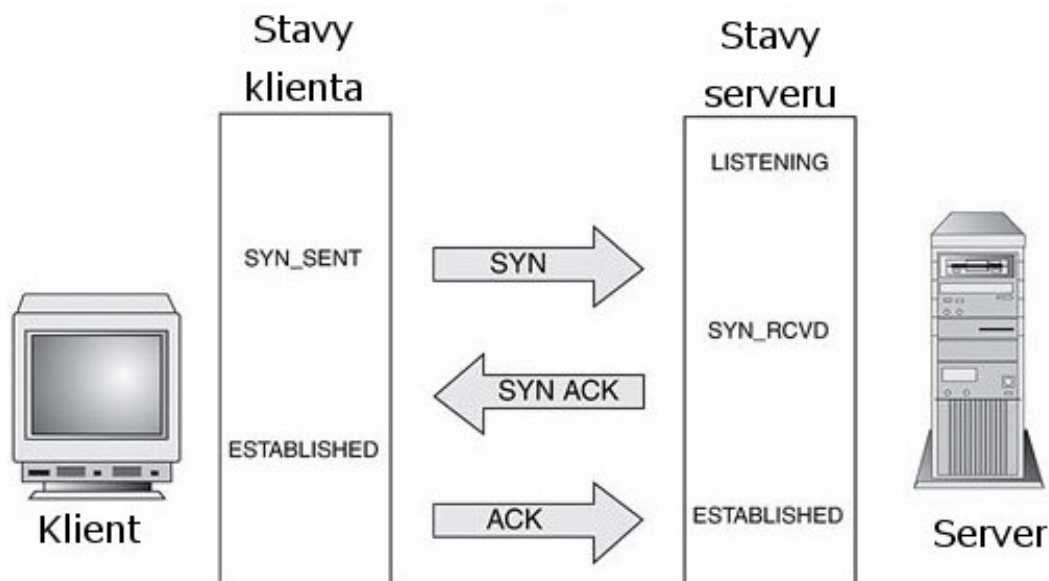
tvořena čtveřicí osmibitových čísel. Vzhledem k poměrně omezenému počtu možných kombinací došlo v nedávné době k vyčerpání posledních volných adres.[9] Z tohoto důvodu je tato verze postupně nahrazována novější, šestou verzí, jež má, mimo jiné, délku adresy 128 bitů a tím pádem umožňuje adresaci nesrovnatelně vyššího počtu zařízení. [11]

2.3 TCP protokol

Protokol TCP je protokolem transportní vrstvy modelu TCP/IP. Při použití TCP mezi sebou aplikace vytvoří spojení, přes které jsou následně přenášena data v balíčcích zvaných pakety. Postup vytvoření spojení je znázorněn na obr.č.5. Protokol garantuje spolehlivé doručení jednotlivých paketů v tom pořadí v jakém byly odeslány. Při ztrátě paketu po cestě je tento odeslán opakovaně až do úspěšného doručení. Správného pořadí je docíleno případným přeuspořádáním paketů. Jelikož transportní vrstva slouží ke komunikaci koncových aplikací, je třeba rozlišit jednotlivé aplikace v koncových zařízeních. K tomuto účelů slouží identifikátor zvaný port. Port tvoří šestnáctibitové kladné číslo. Každá hlavička TCP paketu obsahuje číslo zdrojového i cílového portu. Porty lze rozdělit do tří skupin:

- **známé porty** – porty v rozsahu 0 až 1023, jsou vyhrazené pro nejběžnější služby, na unixových systémech jsou pro jejich použití vyžadována práva superuživatele
- **registrované porty** – rozsah 1024 až 49151
- **dynamické a soukromé porty** – v rozsahu 49152 až 65535, vyhrazené pro dynamické přidělování, neměly by být žádné aplikaci pevně přiřazeny

Termín socket označuje uspořádanou dvojici sestávající se z IP adresy a portu, oddělených dvojtečkou, např. 127.0.0.1:80 . [11]



Obr. 5: Navázání TCP spojení, tzv. Three-Way Handshake

2.4 UDP protokol

Tento protokol představuje alternativu k TCP protokolu, na rozdíl od něj však nezaručuje spolehlivé doručení. To znamená, že odesílající strana se po odeslání zprávy o ni již nadále nestará a věnuje se dalším zprávám.

K rozlišení které aplikaci je zpráva určena slouží porty, jež fungují naprosto stejně jako u TCP protokolu. Za zmínku stojí, že čísla jednotlivých portů TCP a UDP jsou na sobě nezávislé, port se stejným číslem u jiných protokolů tedy mohou využívat různé aplikace.

Výhodou tohoto způsobu je jeho jednoduchost, hlavička obsahuje pouze zdrojový a cílový port, délku datové zprávy a kontrolní součet. Uplatnění tohoto protokolu je nejčastěji u datových přenosů kladoucích důraz na rychlost a tam, kde není vyžadována stoprocentní spolehlivost (např. VoIP, streamování multimedií). [11]

2.5 Sockety v operačních systémech Windows

Sockety ve Windows jsou implementovány pomocí rozhraní Winsock (název vzniknul zkrácením původního Windows Socket API). První verze tohoto rozhraní byla vydána roku 1992. V současné době je využívána druhá, zpětně kompatibilní verze. Winsock je standardní součástí programátorského rozhraní Win32 API.

Winsock vychází z rozhraní Berkley Sockets, internetového rozhraní unixového operačního systému BSD. Na rozdíl od ostatních implementací má Winsock drobné odlišnosti a rozšíření, není tedy plně kompatibilní. Programy běžící na odlišných systémech s odlišnými knihovnami spolu však mohou bez problémů komunikovat.[5]

2.5.1 Blokující a neblokující síťové operace

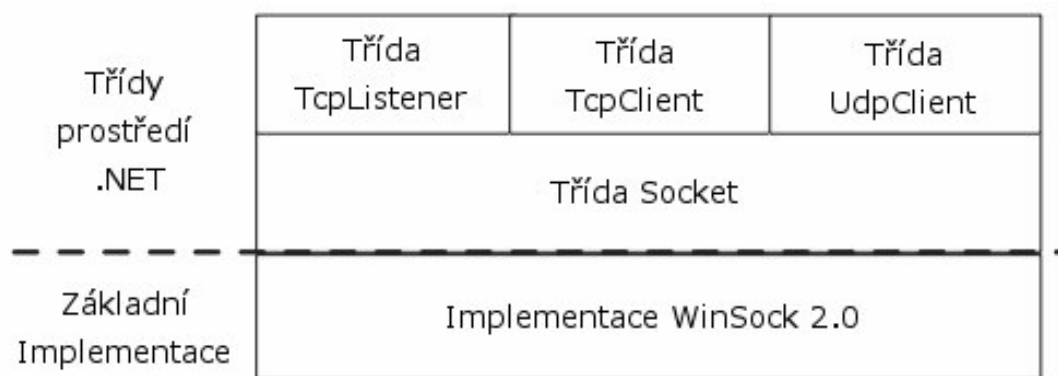
Práce se sockety ve WinSock API může probíhat ve dvou režimech, blokujícím a neblokujícím. Zásadní rozdíl mezi těmito režimy je způsob jakým se aplikace chová při provádění operací se socketem.

Při použití blokujícího spojení dojde k zablokování výkonu aktuálního vlákna až do okamžiku dokončení operace. Tento způsob spojení je často přirovnáván k telefonátu, kdy volající telefonista čeká na lince až do příjmu hovoru druhou stranou.

Na druhou stranu, neblokující spojení provede požadovanou operaci a již dále nečeká. Jde tedy o asynchronní komunikaci. Tento způsob spojení lze přirovnat ke komunikaci pomocí SMS zpráv, kdy po zadání pokynu na odeslání zprávy je možné ihned provádět další operace. [10][11]

2.6 Programování socketů v prostředí .NET

Jelikož platforma .NET klade důraz na objektový přístup, operace se sockety jsou zde implementovány několika třídami. Základní třídou pro práci se sockety je třída Sockets, která je universální a umožňuje využívat všechny vymoženosti rozhraní WinSock 2. Dále nám prostředí umožňuje využívat třídy TcpListener, TcpClient a UdpClient, které jsou na vyšší úrovni abstrakce, což sebou nese pozitiva v podobě zjednodušení pro programátora. Na druhou stranu ovšem abstrakce odebírá flexibilitu nižšího rozhraní. Vztah mezi jednotlivými třídami je pro názornost zobrazen na obrázku č. 6. [4]



Obr. 6: Vztahy mezi Socketovými třídami v prostředí .NET

2.6.1 Třída TcpClient

Třída TcpClient slouží k připojení a komunikaci se serverem, který pasivně čeká na připojení. Typicky provede každý klient tyto kroky:

1. Vytvoření instance klienta, přičemž cílová IP adresa a port mohou být předány jako parametr konstruktoru. Alternativní možností je použití konstruktoru bez argumentu a zavolání metody Connect()
2. Komunikace se serverem pomocí proudů, kdy každá připojená instance obsahuje NetworkStream
3. Ukončení spojení pomocí metody Close()

2.6.2 Třída TcpListener

Tato třída je určena k naslouchání na definovaném portu a očekávání připojení od klientů, vytváří tedy serverovou stranu připojení. Typicky každý server provádí tyto kroky:

1. Vytvoření instance TcpListener, do konstruktoru je zadána IP adresa rozhraní a samozřejmě také port na kterém má server naslouchat.
2. Spuštění naslouchání zavoláním metody Start()
3. Provádění následujícího cyklu:
 - a. Zavolání metody AcceptTcpClient(), jež přijme připojení od klienta. Metoda navrací novou instanci třídy TcpClient. V případě, že není žádný požadavek na připojení, zablokuje metoda běh aktuálního vlákna až do jeho přijmutí.

- b. Komunikace s klientem pomocí streamů
 - c. Uzavření spojení metodou Close()
4. Zastavení naslouchání pomocí metody Stop()

2.6.3 Využití datových proudů při komunikaci přes Tcp

Datové proudy (streamy) u jazyka C#, stejně jako u dalších vyšších programovacích jazyků, představují abstrakci k provádění základních operací pro různá vstupně/výstupní zařízení. Z programátorského hlediska tedy není rozdíl, zda zapisujeme data např. do souboru či na konzoli, rozdíl bude pouze v použitém streamu.

Při použití s protokolem TCP získáme konkrétní stream pomocí metody `GetStream()` zavolané nad instancí třídy `TcpClient`. K následnému zápisu do streamu používáme metodu `Write()` a ke čtení metodu `Read()`.

Ke správné funkci `NetworkStreamu` ještě potřebujeme vytvořit buffer, do kterého budeme ukládat odeslaná či přijatá data v binární podobě. Buffer vytvoříme jako pole proměnných typu `byte`. [4]

2.6.4 Třída `UdpClient`

Třída slouží pro komunikaci přes protokol UDP. Použití je velice podobné jako u `Tcp` klienta, hlavní rozdíl spočívá v nenavazování spojení pomocí metody `Connect()` a nepoužívání streamů. K odeslání datové zprávy slouží metoda `Send()`, k příjmu metoda `Receive()`. [4]

II. PRAKTICKÁ ČÁST

3 APLIKACE SIMWEBLINK

Program SimWebLink je určen ke vzdálenému ovládání a přenosu dat ze schémat aplikace Simulink. Naměřená data jsou přenášena na server, kde dojde k jejich uložení do databáze a v reálném čase se provede jejich vizualizace do grafické podoby. K těmto datům následně přistupuje webová aplikace, jež obstarává prezentaci dat uživateli.

3.1 Struktura programu

SimWebLink je složen z následujících částí

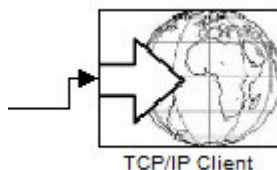
1. S-funkce netout4 – zajišťuje odesílání dat ze Simulinku
2. SimWebLinkClient – klientská aplikace
3. SimWebLinkServer - serverová aplikace
4. SimWebLinkWeb – webové rozhraní
5. Databáze - pro uložení dat

3.1.1 S- function netout4

Tento blok umožňuje odesílání dat ze Simulinku po síti. Pro použití bloku stačí nastavit jeho parametry a přivést na něj požadovaný signál. Bloku lze nastavit tyto parametry:

- IP adresa
- Port
- Perioda vzorkování

Blok byl naprogramován v jazyce C++. Mimo standardních knihoven využívá knihovnu simstruct.h, která je do systému nainstalována společně s MATLABem.



Obr. 7: Schematická značka bloku v Simulinku

3.1.2 SimWebLinkClient – klientská aplikace

SimWebLinkClient je aplikace běžící souběžně s MATLABem na počítači. Pracuje jako klient pro připojení k SimWebLinkServeru a zároveň jako server pro připojení bloků ze Simulinku. Aplikace byla vytvořena v jazyku C# na platformě .NET. Je zde využito více vláken, tak aby bylo možné obsloužit libovolné množství připojení ze Simulinku.

SimWebLinkClient má následující úkoly:

- Připojení k serveru
- Odeslání souboru s konfigurací úlohy
- Spuštění engine MATLABu
- Nastavení parametrů simulace
- Spuštění/zastavení simulace
- Příjem dat od klientských bloků a jejich odeslání serveru

Aplikace byla vytvořena ve dvou provedeních

- Přenosná aplikace – nevyžaduje práva administrátora a spouští simulace pod uživatelským účtem
- Služba systému Windows – umožňuje spouštění úlohy i bez přihlášení uživatele

3.1.3 SimWebLinkServer - serverová aplikace

Tento program, jak již název napovídá, běží na linuxovém serveru, kde zpracovává vstupní data přichozí od jednotlivých klientů. Z přijatých dat jsou generovány grafy průběhů simulací v obrazovém formátu PNG. Formát PNG je velice vhodný díky široké podpoře prohlížečů a také díky dobré kompresi, která není na úkor ztráty obrazové informace. Další výhodou je jeho nezatíženost patenty. Server je napsán jako vícevláknový, což znamená že

umožňuje připojení většího počtu klientů. Zdrojový kód je napsán v jazyce C++ a jsou zde využity následující knihovny:

- libGd – knihovna starající se o vykreslení obrázků
- libOpenThreads – knihovna pro práci s vlákny
- libmysqlpp – knihovna pro práci s databází MySQL
- b64 – knihovna pro kódování dat

Tato aplikace je dílem kolegy Aleše Holíka a byla vytvořena jak diplomová práce v roce 2009

3.1.4 SimWebLinkWeb – webové rozhraní

Webové rozhraní slouží k ovládání aplikace z kteréhokoli počítače připojeného do sítě. Umožňuje zadávat parametry simulace, spustit/zastavit simulaci. V tomto rozhraní jsou uživatelům prezentována naměřená data. Jednotlivé úlohy je možno archivovat do databáze pro jejich pozdější zobrazení. Při tvorbě bylo využito programovacích jazyků PHP a Javascript, spolu s jejich vzájemnou kombinací v technologii AJAX.

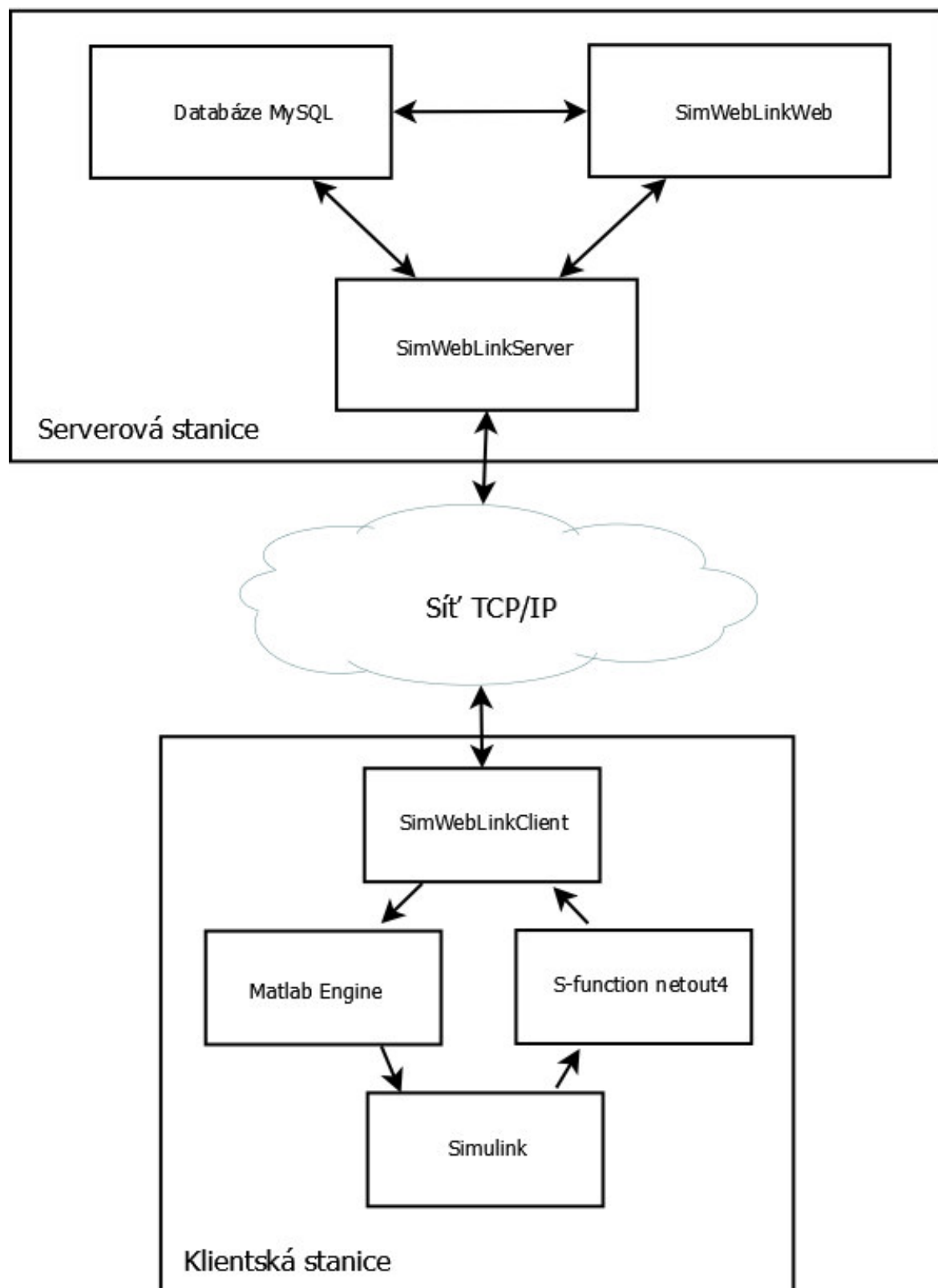
Toto rozhraní je rovněž dílem kolegy Aleše Holíka, byly na něm však provedeny některé úpravy.

3.1.5 Databáze s úlohami

Databáze slouží jako úložiště měřených i archivovaných úloh. Byla použita databáze MySQL, která je díky své jednoduchosti a otevřenému kódu ideální volbou. Do databáze je přístupováno z aplikace SimWebLinkServer a rozhraní SimWebLinkWeb.

Databáze byla rovněž navržena mým předchůdcem. Bohužel v práci mého kolegy byla chybně zdokumentována struktura jedné tabulky, takže při dodržení postupu v jeho práci nebyla aplikace plně funkční a bylo zapotřebí najít chybu. Z tohoto důvodu je v práci obsažen opravený postup instalace databáze.

3.2 Schéma komunikace



Obr. 8: Schéma komunikace jednotlivých částí aplikace

3.3 Komunikace jednotlivých částí

Komunikace mezi SimWebLinkClientem, SimWebLinkServerem byla dostatečně podrobně specifikována v práci mého předchůdce, proto se jí v této práci nebudu opakovaně věnovat. V této práci bude rozepsána specifikace mezi:

1. SimWebLinkClientem a MATLAB Enginem
2. S-funkcí netout4 a SimWebLinkClientem

Komunikaci mezi Simulinkem a S-funkcí není třeba rozepisovat protože se jedná o standardní propojení bloků s Simulinku.

3.3.1 SimWebLinkClient a MATLAB Engine

Ke komunikaci aplikace využívá knihovnu libeng.dll, které se do systému nainstaluje společně s MATLABem. Pro volání z programu psaného v platformě .NET pro její použití nejprve přidáme jmenný prostor **System.Runtime.InteropServices** pomocí direktivy using. Poté do programu importujeme nativní funkce knihovny pomocí DllImport. Mezi nejdůležitější funkce patří:

- engOpen – otevře novou instanci MATLAB enginu, která je vrácena jako návratová hodnota
- engEvalString – vykoná MATLAB příkaz, v našem případě nastavení parametrů, spuštění či zastavení simulace
- engClose – uzavře instanci MATLAB enginu

Importování funkcí:[16][17]

```
[DllImport("libeng.dll")]
static extern IntPtr engOpen(string startcmd);

[DllImport("libeng.dll")]
static extern IntPtr engEvalString(IntPtr engine, string Input);

[DllImport("libeng.dll")]
static extern IntPtr engClose(IntPtr engine);
```

3.3.2 S-funkce netout4 a SimWebLinkClient

Data jsou odesílána po síti pomocí spojení TCP na adresu a port zadané jako parametry do bloku v Simulinku. Jelikož data v sobě nesou označení o pořadí dat, bylo by teoreticky možné využít i protokolu UDP. Tato úprava aplikace by byla velmi jednoduchá. Protože se však u této aplikace nepředpokládá přenos velkého množství dat, nebyl by rozdíl v rychlosti příliš zásadní.

Data jsou odesílána v blocích po časových úsecích vzorkování. Formát bloku dat je následující:

ID_GRAFU:ID_SIGNÁLU:DATA_X:DATA_Y;;

Kde:

- **ID_GRAFU** představuje číselné označení výstupního bloku, tato informace je nezbytná z toho důvodu, aby bylo možno využít více výstupních bloků
- **ID_SIGNÁLU** udává číselné označení signálu. Více signálů do bloku vstupuje pokud v Simulinku použijeme funkční blok **MUX**, který slouží ke spojení více signálů
- **DATA_X** představuje údaj o hodnotě v souřadnici X, tedy časový odemžik odebrání vzorku
- **DATA_Y** udává hodnotu signálu v okamžiku odebrání vzorku

Jako oddělovač údajů byla zvolena dvojtečka která následuje za každým, i posledním, údajem. Každý blok dat musí být ukončen středníkem.

Po ukončení simulace pošle blok zprávu v následujícím tvaru:

Simulation Stop

Tím oznámí SimWebLinkClientovi, že již z daného spojení neprijdou žádná data.

3.4 Typický průběh užití

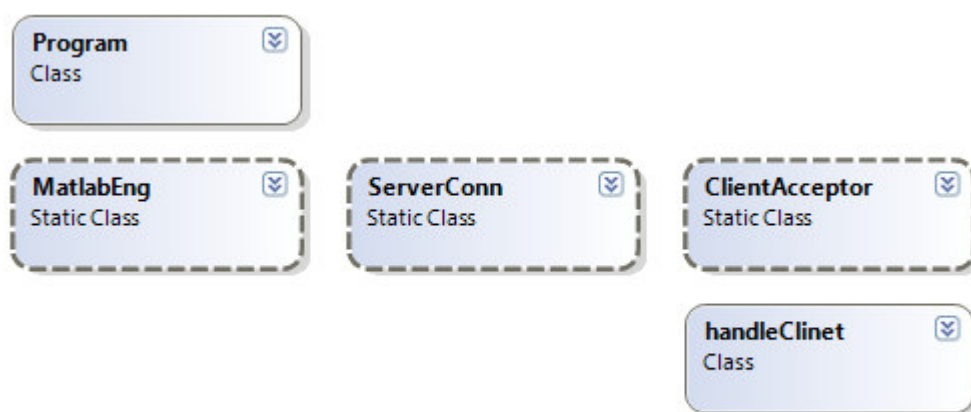
- Na serverové stanici je spuštěn SimWebLinkServer očekávající připojení od klientů
- Na klientské stanici dojde ke spuštění aplikace SimWebLinkClient
- SimWebLinkClient se pokusí připojit k serveru
- Po úspěšném připojení odešle klientská aplikace konfigurační xml soubor s parametry úlohy
- SimWebLinkServer předá soubor SimWebLinkWebu, který z něj vytvoří formulář umožňující nastavení jednotlivých parametrů
- Uživatel na webu nastaví parametry a stiskne tlačítko Start
- Z nastavených parametrů se na webu vygeneruje XML příkaz, který je následně pomocí SimWebLinkServeru doručen SimWebLinkClientu
- SimWebLinkClient přijme XML příkaz a zpracuje jej
- SimWebLinkClient inicializuje MATLAB Engine a načte požadované schéma
- SimWebLinkClient nastaví parametry simulace
- SimWebLinkClient začne naslouchat na definovaném portu
- Pokud XML příkaz obsahoval povel pro spuštění simulace, tak dojde ke spuštění
- V průběhu simulace dojde k odesílání dat přes S-funkce netout4, tyto data jsou následně přijaty SimWebLinkClientem
- SimWebLinkClient přijme data a odešle je SimWebLinkServeru
- Po ukončení simulace uzavře SimWebLinkClient MATLAB Engine
- SimWebLinkClient očekává další příkazy od serveru

4 SIMWEBLINKCLIENT – PROGRAMÁTORSKÁ ČÁST

Cílovou platformou této aplikace je operační systém Microsoft Windows. Při výběru programovacího jazyka byl zvolen jazyk C# a prostředí .NET, které obsahuje všechny potřebné knihovny. Výhodou je mimo jiné objektový přístup, podpora vyjímek, podrobná dokumentace a dobrá podpora ze strany společnosti Microsoft. Drobnou nevýhodou může být fakt, že aplikace vyžaduje na počítači nainstalované prostředí .NET Framework 2.0 a vyšší. Toto prostředí je však na novějších systémech Windows obsaženo již ve výchozím stavu po instalaci.

4.1 Struktura aplikace

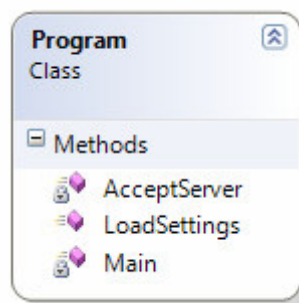
Program je rozdělen do pěti tříd, z nichž tři jsou statické. Rozdíl mezi normální a statickou třídou je ten, že u statické třídy nejsou vytvářeny instance. Výhodou tohoto řešení je, že v programu nemusíme zajistit předávání jednotlivých instancí mezi třídami.



Obr. 9: Struktura aplikace SimWebLinkClient

4.1.1 Třída Program

Toto je hlavní třída vytvořeného programu. Obsahuje funkci **Main()**, která je vstupním bodem aplikace. Prvním úkolem této funkce je načíst nastavení z konfiguračního souboru a předat hodnoty jednotlivých třídám. Toto bylo realizováno metodou **LoadSettings()**. Dalším úkolem funkce je navázání spojení se serverem, odeslat mu konfiguraci úlohy a poté od něj periodicky ve smyčce očekávat příkazy. Tyto úkoly implementuje metoda **AcceptServer()**.



Obr. 10: Struktura třídy Program

4.1.2 Třída MatlabEng

Tato třída zapouzdřuje všechny potřebné operace pro práci s MATLAB engine. Dále obsahuje metodu pro procházení přijatých XML příkazů

Třída má následující atributy:

- **engine** je ukazovatel na instanci MATLAB engine
- **isStarted** je proměnná typu bool udávající, zda simulace běží či je zastavena
- **path** je cesta k adresáři s uloženým schématem uložená jako řetězec
- **schema** je název schématu se kterým bude prováděna simulace

Metody **engClose**, **engEvalString**, **engOpen**, a **engSetVisible** jsou importovány z externí knihovny **libeng.dll**.

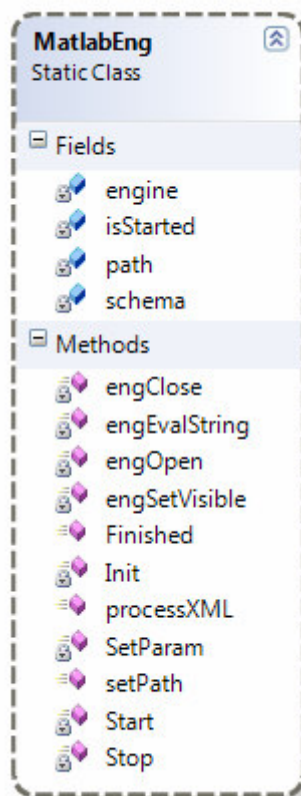
Metoda **Init** provede inicializaci MATLAB engine a dle parametrů **path** a **schema** načte schéma se simulací

Metoda **Finished** uzavírá MATLAB engine

Metoda **processXML** umožňuje parsování přijatých XML příkazů, z této metody jsou volány funkce **Start**, **Stop** a **SetParam**

Metody **Start** a **Stop** slouží k spuštění respektive zastavení simulace.

Metoda **SetParam** je učena k nastavení parametrů simulace. Vstupními parametry jsou název objektu jehož parametr chceme nastavit, název parametru a žádaná hodnota.



Obr. 11: Struktura třídy MatlabEng

4.1.3 Třída ServerConn

Tato třída slouží k zapouzdření komunikace se serverem.

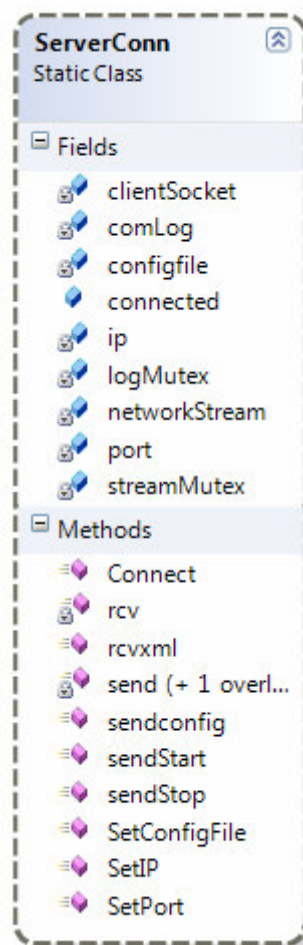
Třída má tyto atributy:

- **clientSocket** – instance třídy TcpClient
- **comLog** – slouží pro logování všech přijatých a odeslaných dat
- **configfile** – řetězec pro uložení obsahu konfiguračního souboru
- **connected** – udává zda je připojení aktivní
- **ip** – řetězec pro uložení IP adresy serveru
- **port** – proměnná typu integer pro uložení portu serveru
- **logMutex** a **streamMutex** – slouží k ošetření vícevláknového přístupu

- **networkStream** – proud pro komunikaci

Metody:

- **Connect** – provede připojení k SimWebLinkServeru a inicializaci **networkStreamu**
- **rcv** – slouží k přijmutí řetězce ze serveru
- **rcvxml** – filtruje přijaté řetězce a vše kromě XML příkazů zahazuje, u přijatého XML zároveň vyfiltruje oddělovací znaky
- **send** – odeslání řetězce serveru, připojení potřebných oddělovacích znaků
- **sendconfig** – odešle konfigurační XML soubor úlohy
- **sendStart** – odešle serveru zprávu signalizující spuštění simulace a zahájení odesílání dat z S-funkce netout4
- **sendStop** - odešle serveru zprávu signalizující zastavení simulace
- **SetConfigFile** – nastaví cestu ke konfiguračnímu XML souboru úlohy
- **SetIP** – nastaví IP adresu serveru
- **SetPort** – nastaví port serveru



Obr. 12: Struktura třídy ServerConn

4.1.4 Třída ClientAcceptor

Tato třída naslouchá na daném portu a očekává připojení klientů z S-funkce netout4. Po připojení každého klienta mu vytvoří instanci třídy **handleClient**. Poté se pokouší o přijetí dalšího klienta. Zároveň si třída udržuje počet připojených klientů, aby mohla při připojení prvního a posledního klienta vykonat následující operace:

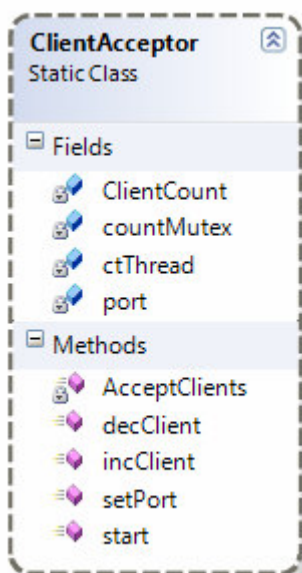
- Při prvním připojení klientu zavolá metodu **ServerConn.Start**, která signalizuje počátek odesílání dat
- Při odpojení posledního klienta zavolá metodu **ServerConn.Stop** signalizující serveru ukončení odesílání dat a také **MatlabEng.Finished** pro ukončení MATLAB enginu

Třída má tyto atributy:

- **ClientCount** – slouží pro uchování počtu aktuálně připojených klientů
- **countMutex** – slouží pro synchronizaci více vláken v metodách **decClient** a **incClient**
- **ctThread** – vlákno vykonávající metodu **AcceptClients**
- **port** – uchování čísla portu na kterém aplikace naslouchá

Metody třídy:

- **AcceptClients** - provádí příjem klientů a následné vytvoření instancí třídy **handleClient**
- **decClient** – sníží hodnotu proměnné **ClientCount** o jedna a vyhodnocuje zda došlo k odpojení posledního klienta
- **incClient** – zvýší proměnnou **ClientCount** o jedna
- **setPort**
- **start** – spuštění vlákna **ctThread**



Obr. 13: Struktura třídy *ClientAcceptor*

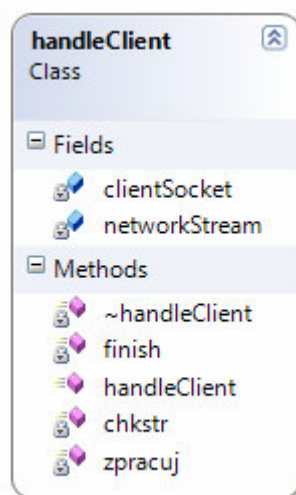
4.1.5 Třída `handleClient`

Třída má tyto atributy:

- **`clientSocket`** – instance připojeného socketu
- **`networkStream`** – datový proud pro příjem dat

Metody třídy:

- **`finish`** – obstarává ukončení vlákna
- **`handleClient`** – konstruktor třídy, provede inicializaci streamu a spuštění vlákna vykonávajícího funkci **`zpracuj`**
- **`chkstr`** – ověřuje platnost přijatých dat
- **`zpracuj`** – hlavní funkce třídy, zajišťuje příjem dat od klienta a jejich odeslání třídou **`ServerConn`**



Obr. 14: Struktura třídy `handleClient`

4.2 Vybrané klíčové části programu

4.2.1 Parsování XML souboru

Parsování XML souboru bylo implementováno pomocí instance **XmlDocument**, vestavěné třídy prostředí .NET. Dokument s XML strukturou je procházen od tagu na nejvyšší úrovni (tzv. kořenového) a postupně se vnořuje do nižších úrovní. K procházení je použit objektový přístup, jednotlivé tagy jsou instance **XmlNode**. Ke každému tagu existují metody, jež z něj dokáží vybrat všechny tagy na nižší úrovni a výsledek vrátit jako pole tagů. Toto pole můžeme dále jednoduše procházet pomocí cyklu **foreach**. Vnitřní obsah každé **XmlNode** můžeme získat ve formě řetězců pomocí metody **InnerText**. Toho je využito u tagů nejnižší úrovně. Název uzlu **XmlNode** získáme metodou **Name**. Atributy můžeme získat pomocí několika metod :

`uzel.Attributes.GetNamedItem(nazev_atributu).Value.`

Díky objektovému přístupu nezáleží na pořadí uzlů, důležité je však zachovat správné vnořování.

Načtení XmlDocumentu z řetězce xmltext obsahujícího celý XML příkaz:

```
XmlDocument xml = new XmlDocument();  
xml.LoadXml(xmltext);  
XmlElement root = xml.DocumentElement;
```

Získání parametrů a jejich předání metodě **SetParam**:

```
//ziskani nazvu schematu  
schema = type.Attributes.GetNamedItem("id").Value;  
  
//ziskani nazvu a hodnoty promenne  
if (cast.Name == "variable") variable = cast.InnerText;  
else if (cast.Name == "value") value = cast.InnerText;  
  
//ziskani nazvu objektu  
objekt = schema;  
if (block.Attributes.Count>0)  
objekt+="/" + block.Attributes.GetNamedItem("id").Value;  
  
//nastaveni parametru ve schematu  
SetParam(objekt, variable, value);
```

4.2.2 Příjem klientů

O příjem klientů je postaráno ve třídě **ClientAcceptor**. Základním pilířem je instance **TcpListener**, která zajišťuje naslouchání na daném portu. Nejdůležitější metodou tohoto objektu je **AcceptTcpClient()**. Tato metoda po svém zavolání zahájí čekání na připojení z druhého konce. Po úspěšném připojení je vytvořena nová instance třídy **handleClient**.

Inicializace třídy TcpListener:

```
myListener = new TcpListener(IPAddress.Any, port);  
myListener.Start();
```

Nekonečná smyčka pro příjem klientů:

```
while (true)  
{  
    clientSocket = myListener.AcceptTcpClient();  
    incClient();  
    handleClient client = new handleClient(clientSocket);  
}
```

5 SIMWEBLINKCLIENT-ZPROVOZNĚNÍ

5.1 S-Funkce netout4

Jelikož S-funkce, stejně jako schémata programu Simulink, nejsou kompatibilní mezi různými verzemi programu MATLAB, je nutno před použitím provést její kompilaci. Ta se provádí přímo v prostředí MATLABu, a to pomocí příkazu **MEX**.

5.1.1 Nastavení nástroje Mex

Před prvním použitím tohoto příkazu je třeba nástroj mex nakonfigurovat pro použití s externím kompilátorem. To se provede příkazem

Mex -setup

Následně po potvrzení volby klávesou Y se v příkazovém okně zobrazí seznam všech kompilátorů, které MATLAB našel. Bohužel MATLAB nalezne pouze ty kompilátory, které byly vydány před použitou verzí MATLABu. Například MATLAB verze R2009b tedy nenajde kompilátor obsažený v instalaci Visual Studio 2010. Vybraný kompilátor zvolíme zadáním čísla zobrazeného ve výpisu před názvem kompilátoru. Následně spustíme celý proces potvrzením klávesou Y. V tomto procesu dojde mimo jiné k nahrazení souboru **mexopts.bat** jedním z předpřipravených. Soubor **mexopts.bat** se nachází v následujícím umístění:

(Domovská složka uživatele)\AppData\Roaming\MathWorks\MATLAB\(\verze MATLABu)\

5.1.2 Kompilace

Po úspěšném nastavení je možné přistoupit k samotné kompilaci souboru. Nejprve se pomocí příkazu **cd (cesta ke složce)** přesuneme do složky, ve které máme připravený soubor se zdrojovým kódem (v našem případě netout4.cpp). Následně kompilaci spustíme zadáním:

mex -O netout4.cpp WS2_32.lib

kde:

- parametr **-O** značí že žádáme optimalizaci kódu
- **netout4.cpp** je název zdrojového souboru

- **WS2_32.lib** udává název knihovny která je využívána, v našem případě odkazuje na knihovnu zapouzdřující API WinSock 2

Pokud kompilace proběhne úspěšně, vytvoří se ve složce se zdrojovým kódem nový soubor:

netout4.mexw32

V některých případech ovšem může dojít k problému a nástroj vypíše následující chybovou hlášku:

netout4.cpp(14) : fatal error C1083: Cannot open include file: 'simstruc.h': No such file or directory

Nástroj nám hlásí, že nemůže nalézt potřebný hlavičkový soubor, . Postup řešení je následující:

1. Otevřeme k editaci soubor **mexopts.bat**, jehož umístění je uvedeno na konci předchozí kapitoly
2. v souboru nalezneme řádek začínající **set INCLUDE**, který se nachází v sekci **General parameters**
3. Na konec tohoto řádku doplníme následující údaj:

%MATLAB%\simulink\include;

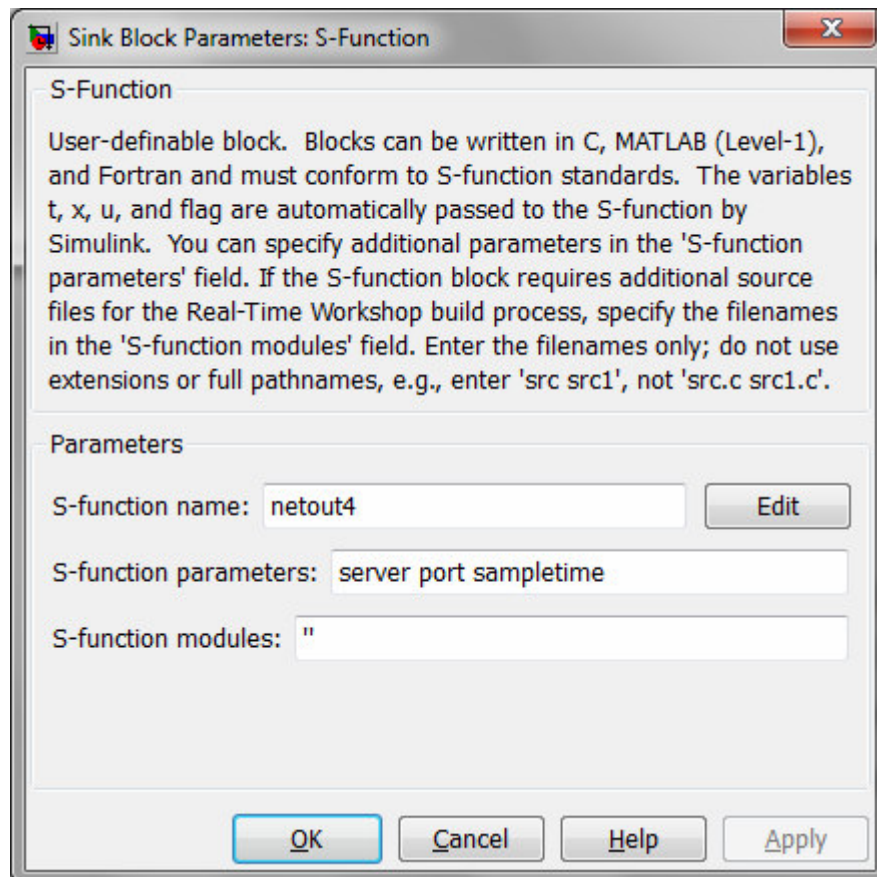
4. Tím jsme nástroji pro kompilaci sdělili, aby při kompilaci procházel zadané umístění, ve kterém je potřebný hlavičkový soubor umístěn
5. Soubor uložíme a pokusíme se o kompilaci znovu
6. Krok nastavení kompilátoru již neopakujeme, neboť by došlo k přepsání souboru **mexopts.bat** zpět do výchozího stavu

Postup kompilace byl vyzkoušen ve 32 bitových verzích MATLAB R2009b a MATLAB R2010b v kombinaci s Microsoft Visual Studio 2008 resp. Microsoft Visual Studio 2010.

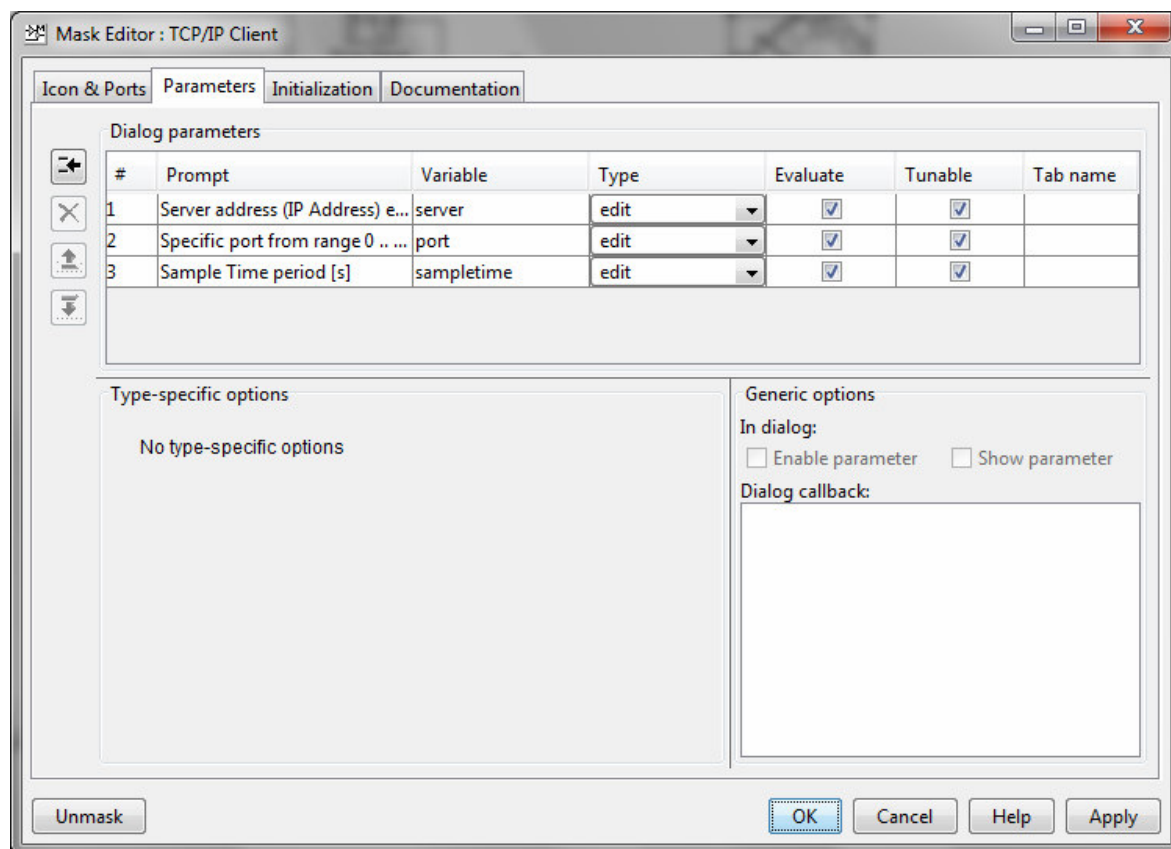
5.1.3 Vložení S-funkce do schématu

Pro vložení S-funkce do schématu otevřeme **Simulink Library Browser**, zvolíme knihovnu **Simulink**, dále kategorii **User Defined Functions**. Zde vybereme blok s názvem **S-function** a vložíme jej do schématu. Nakonec je třeba blok přiřadit naší S-funkci:

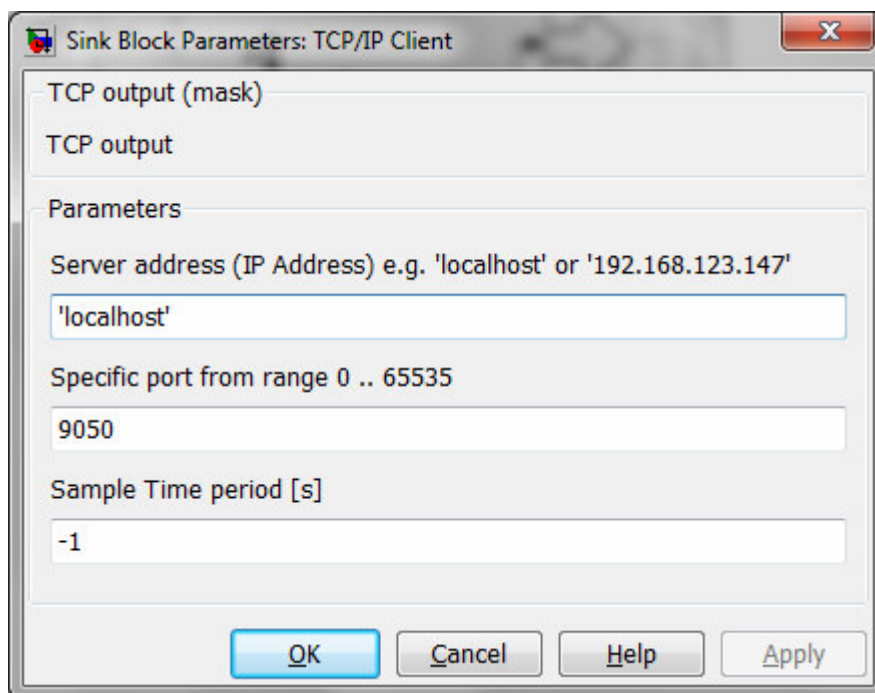
1. Předpokládáme, že soubory **netout4.mexw32** a **netout4.cpp** jsou ve stejném adresáři jako naše schéma. Dále je v adresáři umístěn obrázkový soubor **netout.jpg**
2. Klikneme na blok pravým tlačítkem a vybereme **S-function parameters**
3. Vypneme **S-function name**, v našem případě **netout4**. Dostupnost souboru můžeme vyzkoušet tlačítkem **Edit**, které nám po stisku otevře zdrojový kód.
4. Dále je třeba vyplnit **S-function parameters**. Sem zadáme vstupní parametry S-funkce oddělené mezerou. Vypneme tedy: **server port** a **sampletime**
5. Uzavřeme dialog
6. Klikneme na blok pravým tlačítkem a z kontextové nabídky vybereme **Edit Mask**
7. Na první záložce **Icon & Ports** můžeme vyplnit ikonku našeho bloku tím, že do pole **Icon Drawing commands** vyplníme příkaz **image(imread('netout.jpg'))**. Tento krok nemá žádný vliv na funkčnost, můžeme jej tedy klidně přeskočit.
8. Další záložka **Parameters** je mnohem důležitější. Umožňuje nám přiřadit bloku pole k nastavení parametrů. Vložíme tři položky typu **edit** s hodnotami **Variable** stejnými jako parametry které požadujeme, tedy **server port** a **sampletime**. Ve sloupci **prompt** můžeme vložit textovou informaci, které se zobrazí před políčkem.
9. Uzavřeme dialog
10. Poklepáním na blok můžeme nastavit parametry, pro použití se SimWebLinkClientem použijeme adresu **localhost** a port dle nastavení (výchozí port je 9050)



Obr. 15: Dialog S-function parameters



Obr. 16: Dialog Edit Mask



Obr. 17: Nastavení parametrů bloku odesílajícího data

5.2 SimWebLinkClient

Aplikaci SimWebLinkClient není třeba instalovat, byla vytvořena jako přenosná aplikace. Je však důležité zajistit aby aplikace nebyla blokována firewallem. Před prvním spuštěním je aplikaci potřeba nakonfigurovat. K tomuto účelu slouží soubor **config.ini**, který se nachází ve stejném adresáři jako spustitelný soubor **SimWebLinkClient.exe**.

Každý řádek souboru má tento formát:

(Nazev parametru)=(hodnota parametru);(komentář)

Konfigurační soubor obsahuje následující položky:

- **serverip** – IP adresa serverové stanice
- **serverport** – port, ne kterém naslouchá SimWebLinkServer
- **listenport** – port na kterém naslouchá SimWebLinkClient a očekává připojení od S-funkce Netout4 ze Simulinku
- **schemepath** – umístění adresáře ve kterém se nachází schéma či schémata, která hodláme vzdáleně ovládat
- **configfile** – XML soubor s konfigurací dané úlohy

Struktura konfiguračního XML je následující:

- Celý soubor je uzavřen v tagu **SimWeblinkClient**, který udává původ souboru
- Název úlohy který bude zobrazen na webu je ohraničen tagem **label**
- **adminpass** a **readpass**, sloužící k nastavení hesla pro přístup k úloze na webovém rozhraní
- Parametry jednotlivých grafů uzavřené v tagu **block**, tyto parametry jsou podrobně popsány v práci mého předchůdce [8]
- Tag **Simulink**, sloužící k vygenerování ovládacího panelu na webu, s následující strukturou:

- Tagy **block** s parametrem **id**, jež udává název bloku v Simulinku, ke kterému se vnořené tagy vážou. Pokud není parametr **id** přítomen, obsahuje blok nastavení pro celé schéma. Příkladem položky pro celé schéma může být například délka simulace. Každý **block** obsahuje tyto tagy:
 - **tag name** – popis jež bude ve webovém rozhraní zobrazen před políčkem pro vložení hodnoty
 - **tag variable** – název parametru který odpovídá jeho názvu v Simulinku
 - **tag value** – výchozí hodnota daného nastavení

Ke spuštění aplikace je vyžadováno nainstalované prostředí .NET Framework 2.0 a vyšší. Toto prostředí je na novějších systémech Windows obsaženo již ve výchozím stavu.

5.3 SimWebLinkClient jako služba

K instalaci aplikace SimWebLinkClient jako služby slouží instalační **MSI** balíček vygenerovaný přímo z prostředí Visual Studio. Pro instalaci jsou z bezpečnostních důvodů vyžadována práva administrátora. Na novější systémech Windows je ve výchozím nastavení nutno potvrdit dialog vyvolaný službou UAC. Po úspěšném dokončení instalace je soubor se službou umístěn v adresáři:

(kořenový adresář systémového disku)/Program Files/SimWebLinkService

Samozřejmostí je možnost službu odinstalovat pomocí standardního nástroje Windows **Programy a funkce**, jež je umístěn v Ovládacích panelech.

Konfigurační soubor SimWebLinkClienta **config.ini** je při použití jako služba umístěn v lokaci:

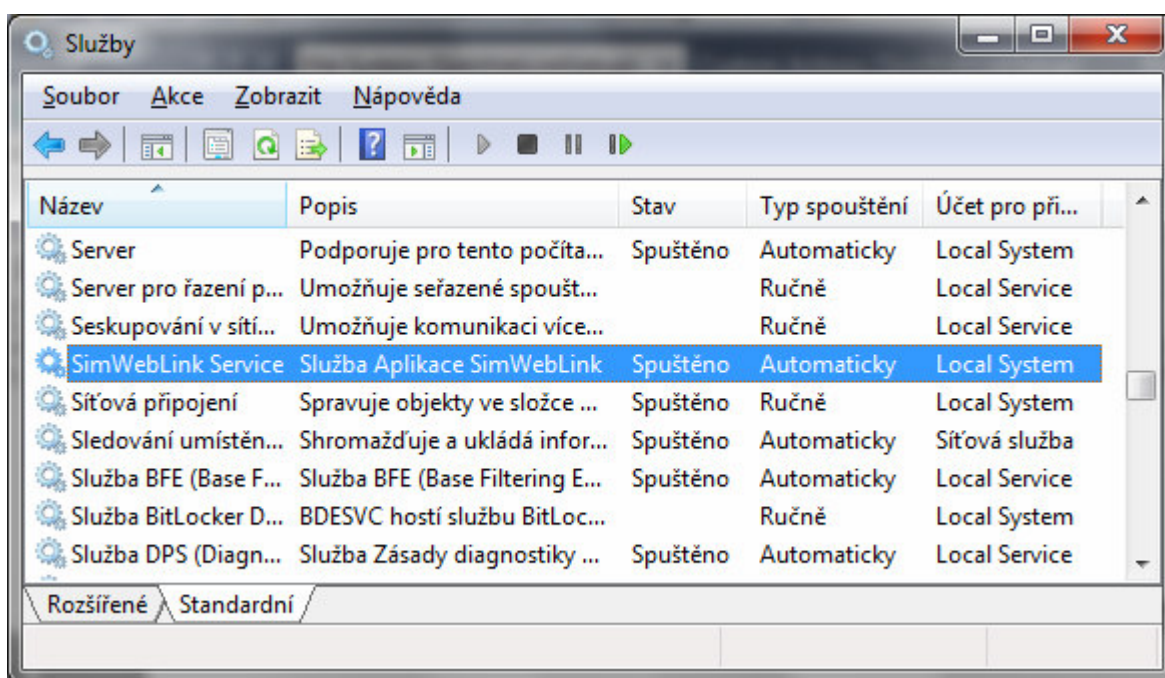
(kořenový adresář systémového disku)/SimWebLink/config.ini

Toto umístění bylo určeno z důvodu zabezpečení systému, editace konfiguračního souboru umístěného ve složce **Program files** vyžaduje speciální oprávnění.

Jelikož při použití služby nemáme možnost zobrazovat výstup na konzoly, byl tento výstup přesměrován do logovacího souboru **applog.txt** umístěného ve stejném adresáři jako výše

uvedený konfigurační soubor. Logovací soubor se automaticky maže při každém spuštění aplikace.

Služba je nainstalována s výchozím nastavením pro **typ spuštění Automaticky**, bude tedy spouštěna při každém spuštění počítače. Pro změnu tohoto nastavení je možné využít nástroj systém Windows pro nastavení služeb. Tento nástroj otevřeme příkazem **services.msc**. Je zde také možno zastavit již běžící službu. Alternativním způsobem je využití příkazu **net** v prostředí příkazové řádky.[15]



Obr. 18: Zobrazení služby v nástroji services.msc

6 ZPROVOZNĚNÍ DATABÁZE ÚLOH

Postup konfigurace databáze je podobně vysvětlen v práci mého předchůdce [8]. Bohužel v jeho dokumentaci se vyskytuje chyba, které má za následek nekorektní funkčnost SimWebLinkServeru. Chyba se projevovala nezobrazováním grafů ve webovém rozhraní. SimWebLinkServer v logovacím souboru nezobrazoval žádnou chybu, bylo tedy nutné najít chybu důkladnou analýzou zdrojového kódu.

Nakonec byla zjištěna nesrovnalost ve funkci ukládající vygenerovaný obrázek do databáze. Tato funkce se pokoušela do řádku tabulky uložit následující pole:

- ID_UD – udává číslo úlohy, ke které je daný graf přiřazen
- ID_GR – udává číslo grafu
- JMENO_OBR – slouží jako popis, tento údaj pochází z konfiguračního XML souboru úlohy
- DATA_OBR – obsahuje obrazová data grafu

Při porovnání s databází vytvořenou dle postupu mého kolegy však tabulka neobsahovala sloupec **JMENO_OBR**. Dodatečným vytvořením sloupce byl problém vyřešen.

Field	Type
<u>id</u>	int(11)
ID_UD	int(11)
ID_GR	int(11)
JMENO_OBR	varchar(50)
DATA_OBR	blob

Obr. 19: Správná struktura tabulky temp_obr

Příkaz pro vytvoření tabulky je uveden v příloze P I. Ostatní příkazy jsou v práci mého předchůdce [8]

7 DALŠÍ MOŽNÝ VÝVOJ APLIKACE

- Vytvoření nástroje pro grafickou editaci a vytváření XML konfiguračních souborů úloh. Tento nástroj by parsoval soubor s uloženým schématem Simulinku a dával uživateli na výběr z parametrů, které by bylo možné na webu nastavit. Dále by obsahoval formuláře pro nastavení základních informací o úloze a hesel. Poté by bylo možné nastavit parametry jednotlivých grafů. To by mohlo být navíc doplněno generovanými náhledy výsledných grafů.
- Šifrování hesel k úloze algoritmy SHA-1, případně MD5. Přidání této funkce je vhodné při implementaci předchozího bodu. Tato změna by vyžadovala zásah do SimWebLinkWebu. Úprava by byla velmi jednoduchá, jelikož jsou hashovací algoritmy součástí jazyka php.
- Další rozšíření funkcionality SimWebLinkWebu, například odeslání změny parametrů bez nutnosti úlohu zastavit a spouštět znova. Tato změna by měla význam při napojení Simulinku na reálnou soustavu, simulace probíhají ve většině případů velice rychle.

ZÁVĚR

Soubor aplikací SimWebLink slouží ke vzdálenému ovládání schémat v programu Simulink a následné presentaci naměřených hodnot ve webovém rozhraní. SimWebLink se skládá z několika částí. SimWebLinkServer je aplikace běžící na operačním systému Linux, jejímž úkolem je komunikace s připojenými klienty a vykreslování grafů z naměřených dat. SimWebLinkWeb je webové rozhraní, které zajišťuje komunikaci s uživatelem, tedy umožňuje zadávání parametrů simulace společně se zobrazením výstupu v grafické podobě. SimWebLinkClient zajišťuje ovládání Simulinkového schématu a odeslání dat serveru.

Teoretické část diplomové práce je rozdělena na dvě části. V první části jsou popsány technologie použité při vývoji aplikace. Druhá část se věnuje programování síťových aplikací na operačních systémech Microsoft Windows a platformě .NET.

V praktické části je popsána struktura aplikace SimWebLink a způsob komunikace jednotlivých jejích částí. Následující část obsahuje podrobný popis aplikace SimWebLinkClient, která byla v rámci této práce vytvořena. Jsou zde popsány všechny třídy z nichž je aplikace složena včetně jejich atributů a metod. Následuje podrobný návod na zprovoznění aplikace včetně zprovoznění jako služby. Práce je zakončena výhledem na další možné rozšiřování aplikace do budoucna.

ZÁVĚR V ANGLIČTINĚ

The SimWebLink application set serves for remote control of Simulink software schemas and for presentation of measured values in web interface. SimWebLink consists of several parts. SimWebLinkServer is application running on Linux operating system, whose task is to communicate with remote clients and to painting of graphs from received data. SimWebLinkWeb is a web interface, which provides communication with the user, thus allowing input of simulation parameters along with displaying output in graphical form. SimWebLinkClient ensures control of Simulink schema and sending of data to server.

Theoretical part of this work is divided into two parts. The first part is a description of technologies used during the development of this application. Second part deals with programming of network applications on Windows operating system family and the .NET platform.

In the practical part of this thesis, there is a description of application structure and communication between its parts. The following part is an description of SimWebLinkClient application, which was created during the work on this project. There is a description of all classes composing the application, including their attributes and methods. The next part is detailed guide to getting the application running, including running the app as service. The thesis is finished with outlook for further possible expansion of the application.

SEZNAM POUŽITÉ LITERATURY

- [1] MATLAB Inc.. MATLAB C and Fortran API reference. The Mathworks Inc., Natick, USA, 2008.
- [2] MATLAB Product documentation [online]. [cit.2011-04-15]. Dostupný z WWW: <<http://www.mathworks.com/help/>>
- [3] MySQL Reference Manual [online]. [cit.2011-04-15]. Dostupný z WWW: <<http://dev.mysql.com/doc/refman/5.5/en/index.html>>
- [4] Elsevier Inc.. TCP/IP Sockets in C# - Practical Guide for Programmers, 2004, ISBN: 0-12-466051-7
- [5] Microsoft Corporation.. Windows Sockets 2.[online] [cit. 2011-04-17] Dostupný z WWW: <<http://msdn.microsoft.com/en-us/library/ms740673%28v=vs.85%29.aspx>>
- [6] Microsoft Corporation.. The TCP/IP model. [online] [cit. 2011-04-15] <<http://technet.microsoft.com/en-us/library/cc786900%28WS.10%29.aspx>>
- [7] PHP: Hypertext Preprocessor [online]. [cit. 2011-04-16]. Dostupný z WWW: <<http://www.php.net/>>.
- [8] Bc. Aleš Holík. Serverová část aplikace SimWebLink , Diplomová práce UTB, 2009
- [9] Ministerstvo průmyslu a obchodu. Adresy Ipv4 jsou vyčerpány [online]. [cit. 2011-04-17]. Dostupný z WWW: <<http://www.mpo.cz/dokument84604.html>>
- [10] Blocking vs. Non-Blocking Sockets [online] [cit 2011-05-1] Dostupný z WWW: <http://www.developerfusion.com/article/28/introduction-to-tcpip/8/>
- [11] IBM Redbooks. TCP/IP Tutorial and Technical Overview, 2006, ISBN: 0-73-849468-2
- [12] Andrew Curioso. Ajax with PHP 5, , 2007, ISBN: 978-0-596-51403-7
- [13] Standard ECMA-334 ,C# Language Specification [online] [cit 2011-04-28] Dostupný z WWW: <<http://www.ecma-international.org/publications/standards/Ecma-334.htm>>

- [14] Common Language Infrastructure (CLI) [online] [cit 2011-04-28] Dostupný z WWW: <<http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf>>
- [15] C# Corner, Windows Services - Windows Service in C# [online] [cit 2011-04-29] <<http://www.c-sharpcorner.com/UploadFile/mirfan00/1753/>>
- [16] Writing C Functions in MATLAB (MEX-Files) [online] [cit 2011-04-30] <<http://cnx.org/content/m12348/latest/>>
- [17] Lab# : Matlab and Scilab access from C# [online] [cit 2011-04-30] <<http://code.google.com/p/labsharp/>>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	Application Programming Interface, rozhraní pro programování aplikací
UAC	User Account Control
XML	Extensible Markup Language
ECMA	European Computer Manufacturers Association
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
CLI	Common Language Infrastructure
php	Php hypertext processor
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
PNG	Portable Network Graphics
AJAX	Asynchronous JavaScript and XML

SEZNAM OBRÁZKŮ

Obr. 1: Příklad schématu vytvořeného v Simulinku	11
Obr. 2: Kompilace do Common Intermediate Language a běh v Common Language Runtime	14
Obr. 3: Model TCP/IP	16
Obr. 4: Schéma komunikace přes TCP/IP	17
Obr. 5: Navázání TCP spojení, tzv. Three-Way Handshake	19
Obr. 6: Vztahy mezi Socketovými třídami v prostředí .NET	21
Obr. 7: Schematická značka bloku v Simulinku	24
Obr. 8: Schéma komunikace jednotlivých částí aplikace	27
Obr. 9: Struktura aplikace SimWebLinkClient	31
Obr. 10: Struktura třídy Program	32
Obr. 11: Struktura třídy MatlabEng	33
Obr. 12: Struktura třídy ServerConn	35
Obr. 13: Struktura třídy ClientAcceptor	36
Obr. 14: Struktura třídy handleClient	37
Obr. 15: Dialog S-function parameters	43
Obr. 16: Dialog Edit Mask	44
Obr. 17: Nastavení parametrů bloku odesílajícího data	44
Obr. 18: Zobrazení služby v nástroji services.msc	47
Obr. 19: Správná struktura tabulky temp_obr	48

SEZNAM TABULEK

SEZNAM PŘÍLOH

Příloha P I: VYTVOŘENÍ TABULKY TEMP_OBR V MYSQL

Příloha P II: PŘÍKLAD KONFIGURAČNÍHO XML SOUBORU ÚLOHY

Příloha P III: KONFIGURAČNÍ SOUBOR SIMWEBLINKCLIENTA

PŘÍLOHA P I: PŘÍKAZ PRO VYTVOŘENÍ TABULKY TEMP_OBR

```
CREATE TABLE IF NOT EXISTS `SimWebLink`.`temp_obr` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `ID_UD` int(11) NOT NULL,  
  `ID_GR` int(11) NOT NULL,  
  `JMENO_OBR` varchar(50) NOT NULL,  
  `DATA_OBR` blob NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=MyISAM ;
```

PŘÍLOHA P II: PŘÍKLAD KONFIGURAČNÍHO XML SOUBORU ÚLOHY

```
<?xml version="1.0" encoding="utf-8"?>
<SimWeblinkClient>
  <label>Ukazkova uloha</label>
  <adminpass></adminpass>
  <readpass></readpass>
  <block id="0">
    <rozliseniX>400</rozliseniX>
    <rozliseniY>400</rozliseniY>
    <minX>0</minX>
    <maxX>50</maxX>
    <minY>-1</minY>
    <maxY>1</maxY>
    <dilekX>10</dilekX>
    <dilekY>1</dilekY>
    <popisekX>osa X</popisekX>
    <popisekY>osa Y</popisekY>
    <paint_lines parameter="yes">#ffcaff</paint_lines>
    <points_on_axes parameter="yes">#000000</points_on_axes>
    <paint_axes parameter="yes">#000000</paint_axes>
    <name>Prvni graf</name>
    <signal id="0">
      <name>jmeno1</name>
      <color>#00ccbb</color>
      <line>plna</line>
      <pointType></pointType>
    </signal>
    <signal id="1">
      <name>jmeno2</name>
      <color>#4759FF</color>
      <line>plna</line>
    </signal>
```

```

    <type>graf</type>
</block>
<block id="1">
    <rozliseniX>400</rozliseniX>
    <rozliseniY>400</rozliseniY>
    <minX>0</minX>
    <maxX>50</maxX>
    <minY>-1</minY>
    <maxY>1</maxY>
    <dilekX>10</dilekX>
    <dilekY>1</dilekY>
    <popisekX>osa X</popisekX>
    <popisekY>osa Y</popisekY>
    <paint_lines parameter="yes">#ffcaff</paint_lines>
    <points_on_axes parameter="yes">#000000</points_on_axes>
    <paint_axes parameter="yes">#000000</paint_axes>
    <name>Druhy graf</name>
    <signal id="0">
<name>jmeno1</name>
<color>#00ccbb</color>
<line>plna</line>
<pointType></pointType>
    </signal>
    <signal id="1">
<name>jmeno2</name>
<color>#cc0000</color>
<line>plna</line>
    </signal>
    <type>graf</type>
</block>

```

```
<Simulink id="netout_schema">
  <block id="netout">
    <name>Vzorkovaci perioda</name>
    <variable>sampletime</variable>
    <value>0.5</value>
  </block>
  <block id="Gain">
    <name>Zesileni</name>
    <variable>Gain</variable>
    <value>0.1</value>
  </block>
</Simulink>
</SimWeblinkClient>
```

PŘÍLOHA P III: KONFIGURAČNÍ SOUBOR SIMWEBLINKCLIENTA

serverip=192.168.1.10 ;ip adresa simweblink serveru

serverport= 9999 ;port simweblink serveru

listenport=9050 ;port na kterém aplikace naslouchá

schemepath=c:\SimWebLink\example\ ;cesta k místu uložení schématu

configfile=c:\SimWebLink\example.xml ;soubor s konfigurací úlohy