České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Komunikace se sítí LonWorks

Praha, 2007 **Autor: Radek Láska**

České vysoké učení technické v Praze - Fakulta elektrotechnická

Katedra řídicí techniky ***Školní rok:*** 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Radek Láska

Obor: Technická kybernetika

Název tématu: Komunikace se sítí LonWorks

Zásady pro vypracování:

1. Nastudujte způsob komunikace v síti LonWorks.
2. Prostudujte protokol LonTalk.
3. Vytvořte SW modul, který bude přistupovat do sítě bez využití LNS serveru.
4. Vytvořte OPC server.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí diplomové práce: Ing. Tomáš Fencl

Termín zadání diplomové práce: zimní semestr 2005/2006 (změna zadání: 18.12.2006)

Termín odevzdání diplomové práce: květen 2007

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

v zast.

prof. Ing. Zbyněk Škvor, CSc.
děkan

V Praze dne 01.02.2007

Abstrakt

V této práci se zabývám vytvořením zařízení, které umožní ovládat a monitorovat zařízení zapojená v síti LonWorks.

Jádro práce spočívá ve vytvoření softwaru, který přistupuje do sítě LonWorks bez využití LNS Serveru a dokáže komunikovat na úrovni síťových proměnných. Vytvořený software používá pro přístup do sítě LonWorks síťové rozhraní s ovladačem, který nabízí rozhraní OpenLDV API nebo MIP API. Výhodou tohoto řešení je výrazně menší finanční náročnost oproti řešení, které používá LNS Server.

Software je založen na vlastní implementaci prezentační a aplikační vrstvy protokolu LonTalk (ANSI/EIA 709.1-B), rozboru těchto vrstev je věnována podstatná část teoretické části této práce. Součástí implementace je také koncept dynamických proměnných a interpretace síťových proměnných dle standardů sdružení LonMark.

Tento software byl v této práci dále využit pro vytvoření OPC serveru, který poskytuje přístup k síťovým proměnným zařízení zapojených v síti LonWorks. Funkční profil OPC serveru (seznam síťových proměnných, se kterými pracuje) lze měnit pomocí konfiguračního nástroje pro správu sítě LonWorks (např. LonMaker).

Abstract

This thesis deals with a design and implementation of a device for monitoring and controlling of devices connected by the LonWorks network.

Main part of this work is concerned with the implementation of software which enables the communication with other devices in the LonWorks network without using the LNS Server. This software uses a network interface equipped with a driver which contains the OpenLDV API or the MIP API. An advantage of this solution is that is cheaper than solution based on the LNS Server.

The software is based on the implementation of presentation and application layers of the LonTalk protocol. These layers are discussed in the theoretical part of this thesis. The implementation also includes the concept of dynamic variables and the interpretation of network variables according to LonMark standards.

This software is also used in the OPC server developed in this work. The OPC server provides access to network variables of devices in the LonWorks network. Function profile of the OPC server can be modified by some network management tool (eg. LonMaker).

Poděkování

Na tomto místě bych rád poděkoval svým rodičům a příbuzným za vytrvalou podporu při mých studiích. Také bych chtěl poděkovat vedoucímu mé diplomové práce Ing. Tomášovi Fenclovi za cenné připomínky a přátelský přístup. Můj dík patří také Ing. Pavlovi Burgetovi za pomoc při práci s modelem domovní automatizace a Ing. Janu Kelbelovi za rady při vytváření OPC serveru.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne: 22.5.2007

Radek Lášku

podpis

Echelon, LON, LONWORKS, LNS, LONMARK, LonMaker, LonTalk, NodeBuilder a ShortStack jsou registrované ochranné známky společnosti Echelon Corporation. OpenLDV je ochranná známka společnosti Echelon Corporation.

Obsah

1 Úvod **1**

2 Technologie LonWorks **3**

2.1 Koncepty LonWorks 4

2.1.1 Protokol LonTalk 4

2.1.1.1 ISO/OSI model 5

2.1.1.2 Přenosová rychlost 6

2.1.1.3 Metoda přístupu k médiu 6

2.1.1.4 Adresování 7

2.1.1.5 Služby zasílání zpráv 9

2.1.1.6 Autentizace 10

2.1.2 Neuron Chip 10

2.1.3 Síťové proměnné 11

2.1.3.1 Typy proměnných 11

2.1.4 Konfigurační parametry 12

2.1.5 Interoperabilita 13

2.1.5.1 Funkční bloky 14

2.1.5.2 Funkční profily 14

2.1.5.3 Profily zařízení 14

2.1.5.4 XIF soubor 15

2.1.6 Instalace sítě LonWorks 16

2.2 Přístupy ke sběrnici LonWorks 17

2.2.1 Embedded zařízení 18

2.2.1.1 Neuron Chip (hosted nodes) 18

2.2.1.2 Host-based nodes, MIP 18

2.2.1.3 Short Stack Development Kit 18

2.2.1.4 Procesor LC3020 19
2.2.2 Přístup z PC 20
2.2.2.1 LNS Server 20
2.2.2.2 OpenLDV 21
2.2.2.3 ORION Stack 22
2.2.3 Výběr řešení pro přístup do sítě LonWorks 22
2.2.3.1 Cenová rozvaha vybraného řešení 23

3 Aplikační a prezentační vrstva protokolu LonTalk 25
3.1 Služby prezentační a aplikační vrstvy 25
3.2 Aplikační rámec 26
3.3 Selektor síťové proměnné 27
3.4 Konfigurační data protokolu LonTalk 27
3.4.1 Konfigurační data v komunikačním koprocesoru 28
3.4.2 Konfigurační data v aplikačním procesoru 29
3.5 Odesílání a příjem síťových proměnných 30
3.6 Dynamické proměnné 31
3.7 Konfigurační zprávy 31
3.7.1 Rozšířená množina zpráv 33
3.8 Algoritmy pro správu sítě 34
3.8.1 Algoritmus instalace zařízení do sítě 35
3.8.1.1 Poznámky k algoritmu instalace zařízení 36
3.8.2 Algoritmus přidání dynamické proměnné 39
3.8.3 Algoritmus odebrání dynamické proměnné 39
3.9 Propojení síťových proměnných 1:N 40
3.9.1 Aliasing 40
3.9.2 Propojování pomocí adresy typu skupina 41

4 Implementace aplikační vrstvy 43
4.1 Vlastnosti aplikace 43
4.2 Architektura aplikace 44
4.2.1 Architektura Rozhraní SP 46
4.3 OpenLDV API 48
4.3.1 Stručný popis OpenLDV API 48
4.3.2 Použití OpenLDV API 49

4.3.2.1 Připojení DLL knihovny 49
4.3.2.2 Otevření spojení se síťovým rozhraním 50
4.3.2.3 Konfigurace síťového rozhraní 50
4.3.3 Rozdíly mezi rozhraním MIP API a OpenLDV API 52
4.4 Zpracování zpráv 53
4.4.1 Data třídy `COpenLDVStack` 54
4.4.2 Zpracování vstupních síťových proměnných 55
4.4.3 Zpracování konfiguračních zpráv 56
4.5 Interpretace síťových proměnných 56
4.5.1 LonMark Resource File API 57
4.5.2 Části síťové proměnné 58
4.5.3 Implementace interpretace síťových proměnných 59
4.6 Přenos dat metodou LW-FTP 61
4.7 Testovací aplikace OpenLDVStackDemo 62
4.8 Vytvoření XIF souboru 64

5 OPC server **66**
5.1 OPC ToolWorX 66
5.2 Architektura aplikace LonOPC 67
5.3 Aplikace LonOPCClient 68
5.3.1 Protokol TCP serveru 70
5.4 Testování aplikace LonOPC 72
5.4.1 Sestava zařízení pro testování 72
5.4.2 Testování LonOPC s OPC klienty 73
5.4.2.1 Testování s OPC Toolboxem Matlabu 74

6 Závěr **77**

Literatura **81**

A Formát zprávy - PDU **I**

B Přidání dynamické síťové proměnné v aplikaci LonMaker **II**

C Odebrání dynamické síťové proměnné v aplikaci LonMaker **IV**

D Deklarace tříd použitých v Rozhraní SP **VI**

E XIF soubor **VIII**

F Instalace a konfigurace OPC serveru LonOPC **XI**

F.1 Potřebný software XI

F.2 Instalace serveru LonOPC XII

F.2.1 Postup instalace serveru LonOPC XII

F.2.1.1 Rozbalení instalačního balíku XII

F.2.1.2 Základní konfigurace XII

F.2.1.3 Význam položek konfiguračního souboru XII

F.2.1.4 Registrace OPC serveru XIII

F.2.1.5 Spuštění serveru XIV

F.2.1.6 Konfigurace serveru v aplikaci LonMaker XIV

F.2.1.7 Konfigurace proměnných LonOPC XV

G Obsah přiloženého CD **XVI**

G.1 Struktura a obsah adresářů XVI

Seznam obrázků

2.1 Příklad logického rozdělení sítě LonWorks. 8
2.2 Funkční profil Node Object. 15
2.3 Příklad profilu zařízení. 16
2.4 Příklad konfigurace sítě LonWorks v aplikaci LonMaker. 17
2.5 Architektura host-based zařízení. 19
2.6 Architektura zařízení s využitím Short Stack Development Kit. 20
2.7 Architektura OpenLDV. 21

3.1 Aplikační rámec. 27
3.2 Konfigurační data protokolu LonTalk. 28
3.3 Aplikační rámec s ECS zprávou. 33
3.4 Algoritmus instalace zařízení do sítě. 36
3.5 Algoritmus přidání dynamické proměnné do zařízení. 39
3.6 Algoritmus odebrání dynamické proměnné ze zařízení. 40
3.7 Propojení SP 1:2 s využitím aliasingu. 41
3.8 Propojení SP 1:2 s využitím adresy typu skupina. 42

4.1 Architektura aplikace. 45
4.2 Architektura Rozhraní SP. 47
4.3 Formát OpenLDV API zprávy. 49
4.4 Konfigurace síťového rozhraní. 51
4.5 Vztahy mezi třídami. 53
4.6 Třídy zděděné od třídy `CVarPart`. 60
4.7 Aplikace OpenLDVStackDemo - hlavní okno. 63
4.8 Aplikace OpenLDVStackDemo - editace síťové proměnné. 64

5.1 Architektura OPC serveru LonOPC. 68
5.2 Aplikace LonOPCClient - hlavní okno. 69

5.3 Protokol komunikace s TCP serverem aplikace LonOPC. 71
5.4 Model domovní automatizace. 73
5.5 Přístup OPC serveru LonOPC do sítě LonWorks v modelu domovní automatizace. 73
5.6 Funkční blok LonOPC při testování. 74
5.7 Osvětlení v závislosti na výkonu svítidla. 75
5.8 Osvětlení v modelu domovní automatizace během 3 dní. 75
5.9 Teploty v modelu domovní automatizace během 3 dní. 76

A.1 Zpráva protokolu LonTalk (PDU - Protocol Data Unit). I

B.1 Výběr objektu *vstupní proměnná* a přetažení nad objekt funkčního bloku. II
B.2 Vytvoření proměnné. II
B.3 Určení zdroje proměnné, který určí SNVT typ vytvářené proměnné. . . . III
B.4 Vybrání zdrojové proměnné, ke které bude nově vytvořená proměnná připojena. III
B.5 V bloku funkčního profilu je nově vytvořená proměnná. III

C.1 Vybrání proměnné, kterou chceme odebrat. IV
C.2 Kontextové menu - vybereme Delete. V
C.3 Potvrzení odebrání. V

F.1 Jména síťových rozhraní. XI
F.2 Struktura adresářů. XII

Seznam tabulek

2.1 Nejpoužívanější fyzické vrstvy v síti LonWorks. 4
2.2 Služby v ISO/OSI modelu. 6
2.3 Způsoby adresování. 8
2.4 Doplňující informace k položkám SP SNVT_switch. 12
2.5 Cenová rozvaha nákladů na vývojové prostředky. 23

3.1 Nejpoužívanější konfigurační zprávy. 32
3.2 Přehled nejpoužívanějších aplikačních příkazů instalační zprávy. 34
3.3 Přehled zdrojů používaných v instalační zprávě. 34

4.1 Přehled síťových rozhraní s OpenLDV API nebo MIP API. 46
4.2 Metody třídy `COpenLDVStack` zpracovávající konfigurační zprávy. 56
4.3 Přehled typů částí síťové proměnné. 58
4.4 Parametry typů částí SP. 59

5.1 Příkazy TCP protokolu. 71

Seznam výpisů programu

2.1 Příklad definice síťové proměnné SNVT_switch. 12
3.1 Deklarace struktury verze 2 pro popis profilu zařízení. 38
4.1 Položky třídy `COpenLDVStack` pro uchování dat protokolu LonTalk. . . . 54
D.1 Deklarace třídy `CAddressTable` VI
D.2 Deklarace třídy `CDomainTable` VII
E.1 XIF soubor pro aplikaci OpenLDVStackDemo a server LonOPC. VIII
F.1 Konfigurační soubor LonOPC XIII
F.2 Obsah souboru se záznamy o běhu serveru. XIV

Seznam použitých zkratek

APDU	–	Application Protocol Data Unit - aplikační rámec, část zprávy protokolu LonTalk, viz 3.2.
ATL	–	Active Template Library je knihovna pro podporu programování technologie COM, viz [23].
Část SP	–	Část síťové proměnné, viz 4.5.2.
ECS zprávy	–	Extended Network Management Command Set, rozšířená množina zpráv definovaná v protokolu LonTalk 709.1-B, viz kap. 3.7.1.
KP	–	Konfigurační parametry, viz kap. 2.1.4.
LW-FTP	–	LonWorks File Transfer Protocol, metoda pro přenos většího množství dat, podrobněji je popsána v kap. 4.6.
MFC	–	Microsoft Foundation Classes je sada tříd, které zapouzdřují Windows API.
MIP	–	Microprocessor Interface Program, firmware pro NeuronChip, při použití tohoto firmwaru lze NeuronChip použít jako komunikační koprocesor, s kterým lze komunikovat pomocí MIP API.
NV	–	Network Variable - síťová proměnná.
OPC	–	OLE for Process Control je otevřený standard pro meziprocesní komunikaci v operačním systému Windows, který je založen na technologii COM.
PDU	–	Protocol Data Unit - zpráva používaná v protokolu LonTalk.
Rozhraní SP	–	Rozhraní síťových proměnných, viz kap. 4.
SP	–	Síťová proměnná.
SPID	–	Standard Program ID, identifikátor profilu zařízení.
XIF soubor	–	Soubor obsahující popis profilu zařízení, viz kap. 2.1.5.4.

Kapitola 1

Úvod

V současnosti jsou na moderní budovy kladeny stále vyšší nároky jak z hlediska komfortu a bezpečnosti uživatelů, tak z hlediska ekonomiky provozu budov. K zajištění těchto požadavků významně přispívá použití elektronických systémů pro automatizaci budov (v anglické literatuře označovány jako BAS - Building Automation Systems nebo building technologies). Tyto systémy řídí vytápění a ventilaci, zajišťují detekci požárů, bezpečnost budov, sledují a optimalizují spotřebu energií v budově. Každý takovýto systém se skládá z mnoha (v reálných instalacích až ze stovek) koncových zařízení, především pak z čidel, akčních členů a regulátorů.

Všechna tato zařízení si musí mezi sebou vyměňovat informace - musí mezi sebou komunikovat. Prostředky pro zajištění této komunikace obvykle nazýváme sběrnice nebo síť. Existuje řada sběrnicových technologií, které jsou v automatizaci budov používány. Mezi nejpoužívanější patří systémy využívající protokoly BACnet (Building Automation and Control Networking Protocol), LonTalk (součást technologie LonWorks) a EIB (European Installation Bus). Všechny tyto protokoly jsou otevřené standardy, což znamená, že protokol je zveřejněn a výrobci jej mohou implementovat do svých zařízení.

LonTalk je protokol, který je součástí celé technologie LonWorks, kterou vyvinula (a některé části si nechala patentovat, viz 2.1.1) firma Echelon z USA. Technologie LonWorks zahrnuje zejména samotný protokol LonTalk, speciální mikroprocesor Neuron Chip, ve kterém je protokol LonTalk implementován a funkční profily zařízení, které spravuje sdružení LonMark.

V této práci se zabývám realizací programu (aplikace), která pracuje v síti LonWorks a slouží pro monitorování a řízení většího množství (více jak 32) zařízení zapojených v síti LonWorks. Existuje několik možností, jak takovou aplikaci realizovat, tyto možnosti jsou podrobněji popsány v kapitole 2.2. Já jsem pro vývoj zvolil řešení, které je založené na takovém hardwaru pro přístup ke sběrnici LonWorks, ve kterém je výrob-

cem implementována první (fyzická) až pátá (relační) vrstva protokolu LonTalk (viz tab. 2.2) dle ISO/OSI modelu. Výhodou tohoto řešení je, že potřebný hardware je na trhu k dispozici od několika výrobců, např. od firmy Echelon (http://www.echelon.com/), Loytec (http://www.loytec.com/), Gesytec (http://www.gesytec.com/), ovladače jsou často k dispozici pro různé operační systémy (Windows XP, Windows CE, Linux). Určitou nevýhodou tohoto řešení je nutnost implementovat šestou (prezentační) a sedmou (aplikační) vrstvu protokolu LonTalk a interpretaci síťových proměnných[1] dle standardů sdružení LonMark.

V současnosti neexistuje open source implementace protokolu LonTalk, kterou by bylo možné využít. K dispozici je referenční implementace celého protokolu LonTalk v jazyce C, kterou je možné stáhnout z webových stránkách firmy Echelon, je ovšem nutné potvrdit souhlas s licencí, která dovoluje tuto referenční implementaci použít pouze pro účely studia protokolu, nelze zdrojové kódy využít pro vývoj komerčních zařízení.

Implementace aplikační a prezentační vrstvy protokolu LonTalk byla podstatná část mé práce, teoretickým poznatkům o těchto vrstvách protokolu LonTalk a úzce souvisejícím tématům (typy síťových proměnných, funkční profily zařízení, instalace zařízení do sítě LonWorks) jsou věnovány kapitoly 2 a 3. Popis realizace implementace prezentační a aplikační vrstvy protokolu LonTalk je v kapitole 4. V kapitole 5 je stručný popis realizace OPC serveru, který je také součástí této práce a který jsem vytvořil s využitím nástroje OPC ToolWorx firmy ICONICS.

[1] O interpretaci síťových proměnných viz dále v kap. 4.5.

Kapitola 2

Technologie LonWorks

Technologie LonWorks byla vyvinuta a uvedena na trh firmou Echelon. Technologie byla od začátku navrhována pro obecné použití v automatizaci, dle [22] bylo záměrem navrhnout systém, který pokryje 80 % průmyslových aplikací. Pod pojmem technologie LonWorks se skrývá nejen vlastní komunikační protokol LonTalk a síť LonWorks realizovaná různými fyzickými vrstvami, ale také speciální procesor Neuron Chip (viz 2.1.2), výrazná podpora interoperability (podrobněji v 2.1.5), vývojové nástroje firmy Echelon a podpora komunikace přes internet (specifikováno protokolem EIA-852).

Hlavní využití našla síť LonWorks v oblasti automatizace budov, ale je využívána i v jiných oblastech. Jmenujme například použití ve vlakových soupravách (např. v metru v New Yorku a Helsinkách, viz [4]). Další využití je při sběru informací při měření dodávek elektrické energie, při kterém se často využívá přenosu dat po rozvodné síti.

LonWorks klade velký důraz na podporu decentralizovanosti celého systému. Všechna zařízení zapojená do sítě vystupují na stejné úrovni, jsou si rovnocenná, žádné zařízení nelze označit jako master či slave. Komunikace mezi zařízeními je založena na protokolu LonTalk, který ovšem nespecifikuje fyzickou vrstvu (dle ISO/OSI modelu). Možné fyzické vrstvy jsou specifikovány sdružením LonMark v dokumentu *LonMark Layer 1-6 Interoperability Guidelines* ([20]). Přehled nejpoužívanějších fyzických vrstev a jejich hlavních vlastností je uveden v tab. 2.1. To, že protokol LonTalk nespecifikuje fyzickou vrstvu, je výhoda, protože lze vždy použít fyzickou vrstvu vhodnou pro danou část technologie. Na druhé straně kvůli tomuto přístupu protokol LonTalk nemůže zaručovat žádné časové parametry doručení zpráv a proto je nevhodný pro aplikace, kde je doručení zpráv v definovaném čase důležité.

Označení	Přenosové médium	Přenosová rychlost	Další poznámky
TP/FT-10	kroucená dvoulinka	78 kbs^{-1}	free-topology[1]
TP/XF-1250	kroucená dvoulinka	1250 kbs^{-1}	sběrnicová topologie
PL-20	rozvod síťového napětí 230 V	5 kbs^{-1}	specifikováno ANSI/EIA/CEA 709.2-A
IP-852	UDP protokol	nedefinována[2]	specifikováno CEA-852

Tabulka 2.1: Nejpoužívanější fyzické vrstvy v síti LonWorks.

2.1 Koncepty LonWorks

2.1.1 Protokol LonTalk

LonTalk byl původně proprietárním protokolem firmy Echelon, ale v roce 1999 byl přijat jako otevřený standard ANSI/EIA 709. Nyní je LonTalk také součástí evropského standardu EN14908, v oblasti železničních vozidel je standardizován jako IEEE 1473-L, organizace SEMI (Semiconductor Equipment and Materials International) LonTalk začlenila do svého standardu SEMI E54 Sensor/Actuator Network Standard, v Čínské lidové republice je LonTalk standardizován jako GB/Z20177.

Nejdříve byla jediná dostupná implementace protokolu LonTalk v procesoru Neuron Chip (viz kap. 2.1.2). Později vyvinula vlastní implementaci protokolu LonTalk firma Loytec do svého procesoru L-Chip (LC7093), který se již ale nevyrábí. L-Chip byl nahrazen výkonnějším procesorem LC3020, který je založen na 32-bit RISC architektuře a který má implementovánu podporu kromě protokolu LonTalk také pro protokol ANSI/EIA 852 a BACnet (viz [3]). V současnosti téměř všechna zařízení pracující v síti LonWorks využívají Neuron Chip nebo L-Chip (příp. zmíněný LC3020). Na webových stránkách firmy Echelon je k dispozici ke stažení referenční implementace protokolu LonTalk v jazyce C pro procesor Motorola MC68360, implementace nezahrnuje rozšířenou množinu konfigu-

[1] Topologie free-topology znamená, že lze kombinovat topologie: sběrnice, hvězda a strom.

[2] Přenosovou rychlost nelze přesně určit, protože závisí na mnoha faktorech, je dána konkrétní implementací protokolu CEA-852 a aktuálním stavem přenosové cesty mezi koncovými body, je třeba vzít úvahu, že cesta může být tvořena mnoha technologiemi, např. 10/100 Mbit ethernet, GPRS, Wi-Fi.

račních zpráv (o těchto zprávách viz odstavec 3.7.1). Tuto implementaci lze dle licence použít pouze pro účely studia protokolu, nelze ji použít pro vytvoření vlastní komerční implementace. LonTalk je sice otevřený standard, ale je také zatížen patenty firmy Echelon, nelze jej tedy implementovat zcela volně. Patentován je například algoritmus pro přístup k médiu *predictive p-persistent CSMA* (Carrier Sense, Multiple Access), viz 2.1.1.3. V případě zájmu o vlastní komerční implementaci je třeba souhlasit se smlouvou Protocol Patent License Agreement ([2]), ve které Echelon uděluje práva používat patentované technologie v prodávaných zařízeních, tato práva jsou udělována zdarma. Smlouva také řeší přidělování Neuron ID, což je celosvětově jedinečný 48-bitový identifikátor každého zařízení v síti LonWorks a firma Echelon přidělování rozsahů těchto identifikátorů spravuje (a také za to vybírá poplatek[1]).

2.1.1.1 ISO/OSI model

Protokol LonTalk definuje kromě fyzické vrstvy všechny vrstvy dle OSI (Open System Interconnection) modelu[2]. Tímto se odlišuje od jiných síťových protokolů, které často nevyužívají všechny vrstvy ISO/OSI modelu nebo je nemají jednoznačně oddělené, mezi takovéto protokoly patří např. CAN (Controller Area Network), EIB (European Installation Bus), PROFIBUS (Process Field Bus), BACnet (Building Automation and Control Networks) či ethernet. Každá vrstva v modelu nabízí služby, které může používat nadřazená vrstva. Stručný popis těchto služeb v protokolu LonTalk je uveden v tab. 2.2, zde je také uvedeno jaký procesor v Neuron Chipu danou vrstvu zpracovává.

[1] V roce 2007 byl poplatek 0,15 USD za jedno Neuron ID.

[2] Referenční OSI model je abstraktní model navržený pro popis síťových komunikačních protokolů, je od roku 1984 přijat jako ISO 7498 a proto je také většinou označován jako ISO/OSI model.

OSI vrstva		Služby vrstvy v protokolu LonTalk	Procesor v Neuron Chipu
7	aplikační	interpretace síťových proměnných dle SNVT (viz 2.1.3.1), přenos dat metodou LW-FTP (viz 4.6)	3
6	prezentační	příjem a odesílání síťových proměnných, explicitních zpráv a cizích rámců (viz 3); zpracování konfiguračních zpráv (viz 3.7); aliasing síťových proměnných (viz 3.9.1)	2
5	relační	služba dotaz/odpověď (transakce) (viz 2.1.1.5); autentizace (viz 2.1.1.6)	2
4	transportní	služba s potvrzováním; služba bez potvrzování; služba bez potvrzování s opakováním (viz 2.1.1.5)	2
3	síťová	adresování (unicast, multicast, broadcast) (viz 2.1.1.4)	2
2	linková	doplnění/kontrola CRC; přístup k médiu predictive p-persistent CSMA (viz 2.1.1.3); předcházení kolizím	1

Tabulka 2.2: Služby v ISO/OSI modelu.

2.1.1.2 Přenosová rychlost

Protokol LonTalk nijak přenosovou rychlost nespecifikuje, je tudíž dána zejména přenosovým médiem (viz tab. 2.1) a případně výkonem zařízení, na kterém je LonTalk zpracováván. Neuron Chip poskytuje rychlost až 1,25 Mbit.s^{-1}, LC3020 od firmy Loytec nabízí až 2,5 Mbit.s^{-1}.

2.1.1.3 Metoda přístupu k médiu

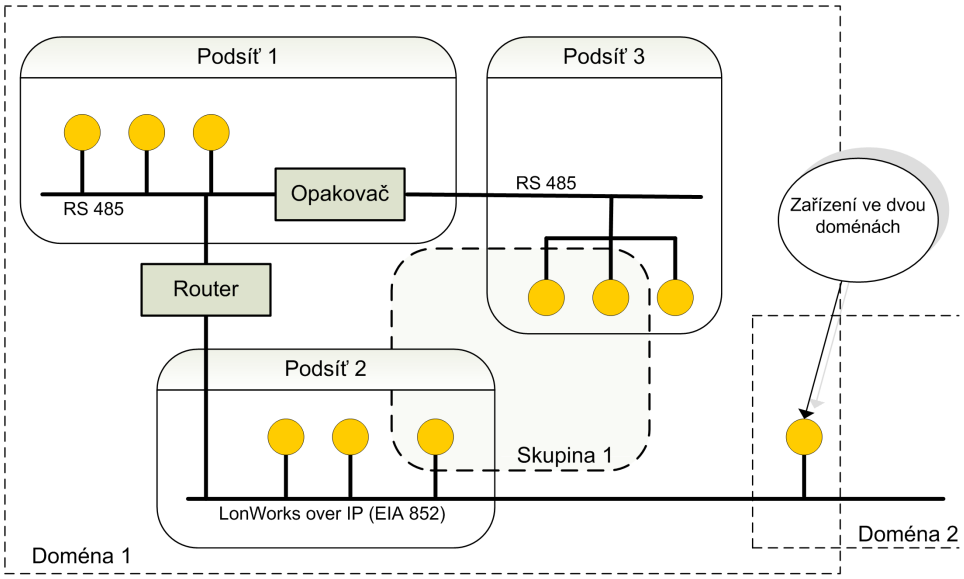
LonTalk využívá tzv. *predictive p-persistent CSMA* metodu pro přístup k médiu. Metoda je odvozena od metody CSMA, kterou používá také např. ethernet. Pokud chce zařízení vysílat na sběrnici, vyčkává, dokud nedetekuje klid na sběrnici (v této fázi žádné jiné zařízení na sběrnici nevysílá), poté začne vysílat a při vysílání zároveň odposlouchává sběrnici. Pokud při vysílání zprávy začne zároveň vysílat jiné zařízení, je tato kolize detekována, vysílání je ukončeno a po náhodně dlouhém čase z pevného intervalu je vysílání opakováno. *Predictive p-persistent CSMA* metoda se liší v tom, že interval, ze kterého je vybírána délka času, po kterém je neúspěšné vysílání opakováno, se mění v čase v závislosti na vytížení sítě. Podrobné vysvětlení jak se interval odhaduje v protokolu LonTalk lze nalézt v [1], [22] a porovnání s metodou CSMA lze nalézt v [15].

2.1.1.4 Adresování

Protokol LonTalk nabízí několik typů adresování. Každému zařízení je při výrobě přiděleno celosvětové unikátní 48-bitové identifikační číslo, tzv. Neuron ID. Tento identifikátor je neměnný a představuje fyzickou adresu zařízení. Fyzická adresa se ovšem pro adresování používá jen velmi zřídka (např. při instalaci). Výhodnější je používat adresy logické, které lze zařízením přiřazovat pomocí konfiguračních nástrojů (např. LonMaker). Protokol LonTalk nabízí pět způsobů logického adresování, lze adresovat jedno zařízení nebo skupinu zařízení najednou, adresování má hierarchickou strukturu. Na obr. 2.1 je příklad logického rozdělení sítě, zde jsou vysvětleny použité pojmy používané při adresování v síti LonWorks:

- *Doména* (domain) je logická skupina zařízení nejvyšší úrovně. Komunikace v síti LonWorks může probíhat pouze mezi zařízeními, která jsou ve stejné doméně. Identifikátor domény může být dlouhý 0, 1, 3 nebo 6 bytů.
- *Podsíť* (subnet) je logická skupina zařízení, která může obsahovat až 127 zařízení. V jedné doméně může být až 255 podsítí, tedy celkem může v doméně být maximálně 32 385 zařízení (tj. 255 podsítí krát 127 zařízení).
- *Uzel* (node) je identifikátor zařízení v rámci jedné podsítě.
- *Skupina* (group) slouží pro adresování více zařízení najednou. Takto adresovaná zařízení nemusí být v jedné podsíti (pouze musí být v jedné doméně). Každé zařízení smí být členem maximálně 15 skupin. Adresování pomocí skupin je velmi užitečné pro koncept síťových proměnných. V rámci jedné domény může být definováno až 255 skupin.
- *ID člena skupiny* je jednoznačný identifikátor zařízení v rámci jedné skupiny.

Tabulka 2.3 shrnuje možné typy adresování, typy 0a až 2b využívají logické adresy, způsob 3 využívá fyzickou adresu (Neuron ID). Každá zpráva zaslaná protokolem LonTalk (dále bude označována PDU - Protocol Data Unit) obsahuje adresu odesilatele (zdrojová adresa) a adresu příjemce (cílová adresa).



Obrázek 2.1: Příklad logického rozdělení sítě LonWorks.

Typ	Formát cílové adresy	Formát zdrojové adresy	Typ cílové adresy	Příjemci
0a	doména, podsíť = 0	podsíť, uzel	broadcast	všechna zařízení v doméně
0b	doména, podsíť ≠ 0	podsíť, uzel	multicast	všechna zařízení v podsíti
1	doména, skupina	podsíť, uzel	multicast	všechna zařízení ve skupině
2a	doména, podsíť, uzel	podsíť, uzel	unicast	jedno zařízení v podsíti
2b	doména, podsíť, uzel	podsíť, uzel, skupina, ID člena skupiny	unicast	jedno zařízení v podsíti
3	Neuron ID	podsíť, uzel	unicast	jedno zařízení

Tabulka 2.3: Způsoby adresování.

Typ adresování 2b se používá pro potvrzování zprávy, která byla zaslána skupině (pokud tato zpráva byla zaslána službou vyžadující potvrzení). Tedy ta zařízení skupiny, která zprávu přijala, zasílají zprávu potvrzující příjem pomocí adresování typu 2b. Přesný formát jak je adresa uložena v PDU lze nalézt v [1] a je také znázorněn na obrázku A.1 v příloze.

Každé zařízení může být členem maximálně dvou domén a pro každou má přiřazenu jednu *základní adresu* ve tvaru *doména*, *podsíť*, *uzel*. Kromě těchto *základních adres* může být zařízení také přiřazeno dalších až 15 *doplňujících adres*[3]. *Doplňující adresy* již neobsahují identifikátor domény, obsahují pouze odkaz na jednu ze dvou *základních adres*, které identifikátor domény obsahují, viz také obr. 3.2.

2.1.1.5 Služby zasílání zpráv

Protokol LonTalk poskytuje čtyři různé služby pro zasílání zpráv, které jsou realizovány transportní a relační vrstvou dle ISO/OSI modelu.

- *Služba s potvrzováním* (Acknowledged service) je používána pro spolehlivé zasílání zpráv jednomu zařízení nebo skupině zařízení. Každé zařízení, které zprávu přijme, odešle potvrzení o příjmu zprávy. Pokud všichni příjemci nepotvrdí příjem zprávy, je zpráva vysílána znovu. Znovu vysílaná zpráva obsahuje seznam členů skupiny, kteří příjem nepotvrdili (a těm je zpráva fakticky určena, doručena je ale všem členům skupiny). Pokud je členů skupiny více než 16, již není součástí znovu vyslané zprávy obsah zprávy (APDU). Podrobnosti lze nalézt v kap. 4.7.3 v [22] nebo v kap. 10 v [1].

- *Služba bez potvrzování* (Unacknowledged service) je používána, jestliže je zasíláno velké množství dat nebo je potřeba dosáhnout velké přenosové rychlosti. Tato služba je používána například při přenosu konfiguračních parametrů do zařízení metodou LW-FTP (viz kap. 4.6).

- *Služba bez potvrzování s opakováním* (Unacknowledged repeated service) poskytuje stejné možnosti jako *služba bez potvrzování* a navíc lze nastavit počet opakovaní, kolikrát je zpráva odeslána. Tímto se dosahuje zvýšení spolehlivosti doručení zpráv zejména u zpráv, které jsou posílány většímu množství cílových zařízení.

- *Služba dotaz/odpověď* (Request/Response service) slouží pro získávání dat z cílového zařízení (nebo ze skupiny zařízení). Odpověď obsahuje požadovaná data. Zpráva může být zaslána jednomu zařízení nebo skupině zařízení. Tato služba je používána zejména při konfiguraci zařízení, tj. využívají ji zejména konfigurační zprávy (viz 3.7) a diagnostické zprávy aplikační vrstvy.

[3] Pokud zařízení podporuje rozšířenou množinu zpráv (viz 3.7.1), může doplňujících adres být až 65535.

2.1.1.6 Autentizace

Autentizace dovoluje příjemci ověřit identitu odesilatele. Pouze zprávy zasílané *službou s potvrzováním* nebo *službou dotaz/odpověď* mohu být autentizovány. Pro autentizaci se používá 48-bitový klíč a náhodně vygenerované číslo pro danou transakci. Klíč musí být uložen na serveru i u klienta a lze jej změnit pomocí konfiguračního nástroje, toto musí být ale prováděno velmi opatrně, protože nový klíč je zasílán nešifrovaný. Podrobný popis mechanismu lze nalézt v [22], [17] nebo v [1], kde je také uveden kódovací algoritmus.

2.1.2 Neuron Chip

Procesor Neuron Chip je důležitou součástí, doslova základním kamenem, technologie LonWorks. Tento procesor byl navržen firmou Echelon a jeho výroba byla svěřena firmám Toshiba, Motorola a Cypress Semiconductor. Neuron Chip a související vývojové nástroje byly navrženy především s ohledem na snadný a efektivní vývoj zařízení pracujících v síti LonWorks. Neuron Chip poskytuje kompletní implementaci protokolu LonTalk a 8-bitový procesor pro běh aplikace daného zařízení. Uvnitř je Neuron Chip tvořen třemi specializovanými 8-bitovými procesory, které jsou integrovány v jeden celek do jednoho pouzdra. 1. procesor zpracovává linkovou vrstvu protokolu LonTalk (dle ISO/OSI modelu) a přístup k médiu (MAC - Medium Access Control), 2. procesor zpracovává síťovou, transportní, relační, transakční a prezentační vrstvu protokolu, 3. procesor je určen pro zpracování aplikační vrstvy a především pro běh aplikace.

Pro implementaci aplikací pro Neuron Chip se používá speciální programovací jazyk Neuron C, který je odvozen od ANSI C. Jazyk Neuron C umožňuje programování řídicí aplikace s využitím zabudovaného plánovače úloh (multitasking scheduler). Neuron C také přímo podporuje událostmi řízené programování. Při každé nově přijaté hodnotě nějaké síťové proměnné se spustí procedura, ve které lze provést adekvátní obsluhu.

Neuron Chip lze provozovat ve dvou režimech. V prvním případě běží aplikace přímo v Neuron Chipu (jak bylo popsáno výše), popis vývoje tohoto typu zařízení lze nalézt v diplomových pracích [17] a [25]. V druhém případě slouží Neuron Chip pouze jako komunikační koprocesor a aplikace je vykonávána v jiném procesoru[4], takový procesor je poté označován jako *host processor* (já jej dále budu označovat jako *aplikační procesor*).

[4] Tímto procesorem může být také PC.

2.1.3 Síťové proměnné

Síťové proměnné (Network Variables, zkráceně NV) jsou datové objekty, které slouží pro sdílení dat v distribuované řídicí aplikaci. Síťové proměnné (dále budou pro zjednodušení označovány zkratkou SP) jsou nejčastěji používaný způsob výměny dat mezi zařízeními v síti LonWorks. SP představují u daného zařízení datové vstupy a výstupy. Maximální velikost jedné SP je 31 bytů. Propojení výstupních SP jednoho zařízení se vstupními SP jiných zařízení se provádí při konfiguraci sítě (to lze provádět i když síť ještě není fyzicky realizovaná, viz 2.1.6). Toto logické propojování SP je označováno jako *binding* a provádí se zpravidla pomocí nějakého konfiguračního nástroje[5].

Aplikace běžící v zařízení pouze aktualizuje výstupní SP a dále se již nestará o doručení hodnot SP k ostatním zařízením, toto už je úkolem protokolu LonTalk a záleží na aktuální konfiguraci, jakým zařízením a jak (pomocí jaké služby) budou data doručena. Obdobně u vstupních SP se aplikace nemusí starat o příchozí data, je-li přijata nová hodnota SP, je o tom aplikace informována. Např. v Neuron Chipu je spuštěna odpovídající obslužná funkce pro každou SP. U vstupních SP lze ale také využít mechanismu dotazování se na hodnoty vstupních proměnných (tzv. *polling*). Tohoto mechanismu lze využít jen tehdy, pokud dotazující se zařízení zná adresu dotazovaného zařízení, v obvyklé implementaci protokolu LonTalk se ale tyto adresy neukládají do konfiguračních struktur. Mechanismus proto používají zejména konfigurační nástroje, protože ty tyto adresy uloženy mají. Také jej mohou používat aplikace, které k datům konfiguračního nástroje mají přístup. Příkladem jsou aplikace, které využívají LNS Server[6], například *EasyLON OPC server L*[7] nebo OPC server *IPLONGATE* od firmy who Ingenieurgesellschaft mbH (http://www.who-ing.de).

2.1.3.1 Typy proměnných

Reprezentace dat SP je určena jejím typem. Typ SP určuje velikost proměnné a její konkrétní fyzikální význam. Sdružení LonMark spravuje definice tzv. *standardních typů* (SNVT - Standard Network Variable Type) a vydává jejich seznam nazývaný SNVT Master List. Tento seznam je dispozici nejen jako textový dokument (viz [10] nebo http://types.lonmark.org), ale také v podobě binárních souborů, které jsou volně ke stažení (viz [10]) a ke kterým sdružení LonMark také zveřejňuje API pro čtení a zápis

[5] Od roku 2005 je k dispozici také technologie ISI (Interoperable Self-Installation), pomocí které lze zařízení do sítě LonWorks přidávat a propojovat SP zcela bez konfiguračního nástroje, viz [6].

[6] LNS Server je produkt firmy Echelon, více o něm je v kap. 2.2.2.1.

[7] *EasyLON OPC server L* je produkt firmy Gesytec, která nabízí také OPC server *EasyLON OPC server M*, který ke své činnosti LNS Server nepotřebuje .

do souborů (podrobněji o tomto API je pojednáno v 4.5.3). Specifikace typu SP určuje uložení dat v proměnné (např. zda jde o celočíselné číslo, kolik bytů zabírá), fyzikální rozměr, jednotku, rozlišení a název. Každý typ má přidělen číselný index, je označován jako SNVT Index. Příklad definice typu jedné SP je ve výpisu 2.1, definice také specifikuje další parametry jednotlivých *částí SP* viz tab. 2.4.

```
typedef struct
{
  unsigned   value;
  signed     state;
} SNVT_switch;
```

Výpis 2.1: Příklad definice síťové proměnné SNVT_switch.

Položka	Parametr	Hodnota
value	typ dat	unsigned short
	minimum	0
	maximum	200
	škálovácí konstanty (A, B, C)	5, -1, 0
	výpočet hodnoty	$A.10^B.(data + C)$
	rozlišení	0,5
state	typ dat	signed short
	minimum	-1
	maximum	1
	škálovácí konstanty (A, B, C)	1, 0, 0
	výpočet hodnoty	$A.10^B.(data + C)$
	rozlišení	1

Tabulka 2.4: Doplňující informace k položkám SP SNVT_switch.

2.1.4 Konfigurační parametry

Konfigurační parametry (Configuration Properties) slouží k nastavení běhu zařízení. Konfigurační parametry (v dalším textu budou označovány zkratkou KP) se používají například pro nastavení časové periody, v jaké senzory odesílají naměřené hodnoty, pro nastavení offsetů, konstant PID regulátoru apod. Hlavní výhodou KP je, že kopie jejich hodnot je uložena v databázi konfiguračního nástroje a při výměně zařízení je lze opět jednoduše a rychle nahrát do nového zařízení, čímž zajistíme, že zařízení je nakonfigurováno

zcela stejně jako bylo zařízení měněné. KP také mají obdobně jako SP své standardní typy (SCPT - Standard Configuration Properties Type), které jsou spravovány rovněž sdružením LonMark. Přenos hodnot KP z konfiguračního nástroje do zařízení lze realizovat třemi různými způsoby (podrobnosti lze nalézt v [19] v kap. 2.7.3.2):

- Přenos KP jako síťové proměnné.
- Přenos s využitím příkazů pro přímý zápis do paměti zařízení.
- Přenos metodou LW-FTP pro přenos většího množství dat, podrobněji je tato metoda rozebrána v 4.6.

2.1.5 Interoperabilita

Pojmem *interoperabilita* zde budu označovat vlastnost systémů, která jim umožňuje pracovat jako jeden z prvků distribuované řídicí aplikace, přičemž každý takovýto prvek může být nahrazen prvkem jiného výrobce bez nutnosti provést změny v řídicí aplikaci.

Význam interoperability vzrůstá především v distribuovaných řídicích aplikacích, které obsahují mnoho (řádově stovky) spolupracujících zařízení, protože se v takovýchto instalacích obvykle využívají zařízení mnoha výrobců. K zajištění interoperability v takovém rozsahu jak bylo uvedeno výše je třeba zajistit nejen spolupráci mezi zařízeními na všech úrovních dle ISO/OSI modelu. Je také třeba zajistit:

- shodný *mechanismus přístupu* ke sdíleným datům (zahrnuje např. identifikaci dat, autentizaci přístupu),
- shodnou *definici dat* představující typ dat, velikost, uspořádání (např. zda jsou data big-endian nebo little-endian, kódování textů),
- shodný *význam dat*, což znamená stejnou logickou či fyzikální interpretaci dat,
- *shodnost chování aplikace*, zde se má na mysli funkční shodnost, tj. aplikace (zařízení) se zvenku musí projevovat stejně, vnitřní konkrétní implementace algoritmů může být libovolná,
- shodnost *komunikačního rozhraní zařízení* tj. datové vstupy a výstupy zařízení musí být stejné.

V technologii LonWorks je mechanismus přístupu k datům zajištěn autentizací (viz 2.1.1.6) a identifikace dat je zajištěna tzv. selektorem (podrobněji viz 3.3). Definici dat a význam dat zajišťují standardní typy síťových proměnných (SNVT) jak bylo vysvětleno v 2.1.3.1. Zbývá tedy zajistit definice rozhraní zařízení a shodnost aplikace, k tomuto slouží funkční bloky, funkční profily a profily zařízení.

2.1.5.1 Funkční bloky

Funkční bloky sdružují vstupní SP, výstupní SP a KP, které souvisejí s jednou aplikací zařízení (v jednom zařízení může být implementováno několik aplikací, např. senzor teploty vzduchu a senzor vlhkosti vzduchu). Funkční blok je častí profilu (rozhraní) celého zařízení, příklad takového rozhraní je na obr. 2.3.

2.1.5.2 Funkční profily

Funkční profily jsou šablony pro funkční bloky. Funkční profily specifikují pomocí SNVT povinné a volitelné vstupní a výstupní SP a povinné a volitelné KP. Jestliže nějaký funkční blok implementuje nějaký funkční profil, musí obsahovat všechny položky funkčního profilu, které jsou povinné. Funkční blok ale může také přidat své specifické SP a KP, které nejsou součástí funkčního profilu.

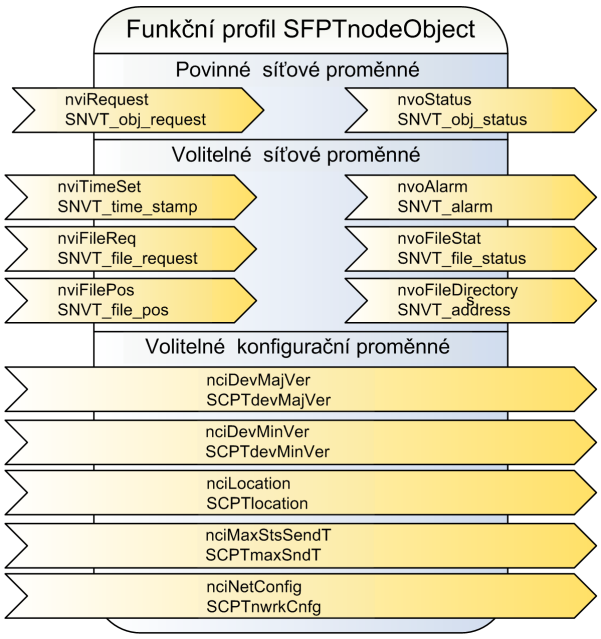
Sdružení LonMark specifikuje standardní funkční profily (SFPT - Standard Functional Profile Templates) pro různá zařízení (např. pro senzory teploty, vlhkosti, osvětlení atd.), seznam všech SFPT lze nalézt na webové stránce sdružení LonMark[8], každý standardní profil má přiděleno unikátní číslo (ID) nazývané *functional profile number* nebo *functional profile key*. Funkční profil je vždy definován pro jednu specifickou aplikaci, např. pro senzor přítomnosti osob v prostoru je specifikován funkční profil Occupancy controller. Příklad funkčního profilu s vyznačením povinných a volitelných SP a KP je na obrázku obr. 2.2.

2.1.5.3 Profily zařízení

Profil zařízení (Device Interface) popisuje zařízení jako celek, definuje:

- identifikátor profilu zařízení (tzv. Standard Program ID - SPID),
- jaké funkční profily zařízení implementuje,
- jaké další SP a KP (neobsažené ve funkčních profilech) implementuje,

[8] http://www.lonmark.org/technical_resources/guidelines/functional_profiles.shtml



Obrázek 2.2: Funkční profil Node Object.

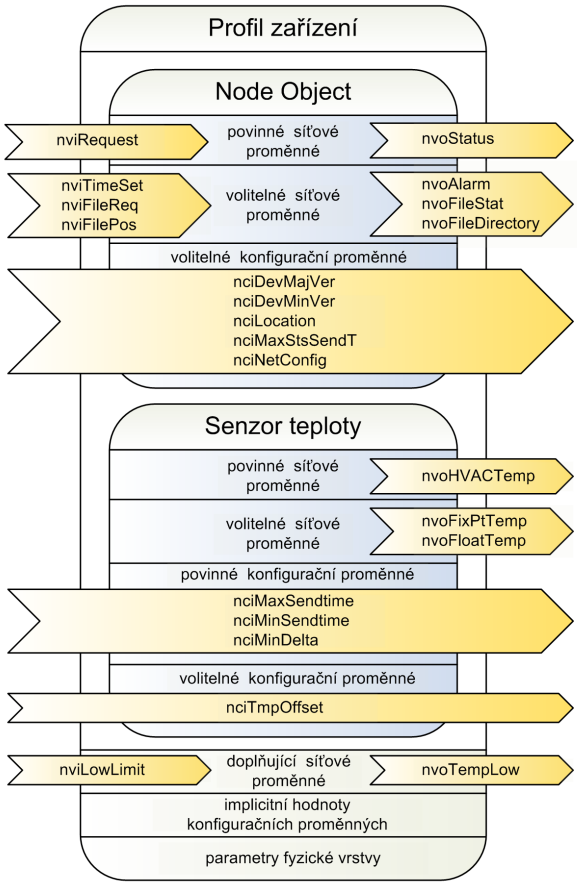
- implicitní hodnoty KP,
- vlastnosti implementace protokolu LonTalk (např. maximální možný použitelný počet SP, zda jsou podporovány dynamické síťové proměnné - o nich více v 3.6),
- parametry fyzické vrstvy.

Každé zařízení, které splňuje specifikace sdružení LonMark musí ve svém profilu obsahovat speciální funkční profil Node Object, který obsahuje SP, které používají konfigurační nástroje pro správu všech ostatních funkčních bloků v zařízení a pro zjišťování stavu zařízení. Přehled povinných a volitelných SP a KP pro funkční profil Node Object je na obr. 2.2.

Příklad rozhraní zařízení znázorněný na obr. 2.3 obsahuje povinný funkční blok Node Object, funkční blok dle profilu pro senzor teploty a další *doplňující síťové proměnné*, které nejsou členem žádného funkčního bloku.

2.1.5.4 XIF soubor

XIF soubor obsahuje popis profilu zařízení, formát souboru je specifikován sdružením LonMark, viz [21]. XIF soubor by měl být vždy dodáván výrobcem spolu se zařízením, protože jej používají konfigurační nástroje pro načtení informací o zařízení do své databáze. XIF soubor je textový soubor, lze jej tedy vytvořit i ručně, ale častěji je vytvářen



Obrázek 2.3: Příklad profilu zařízení.

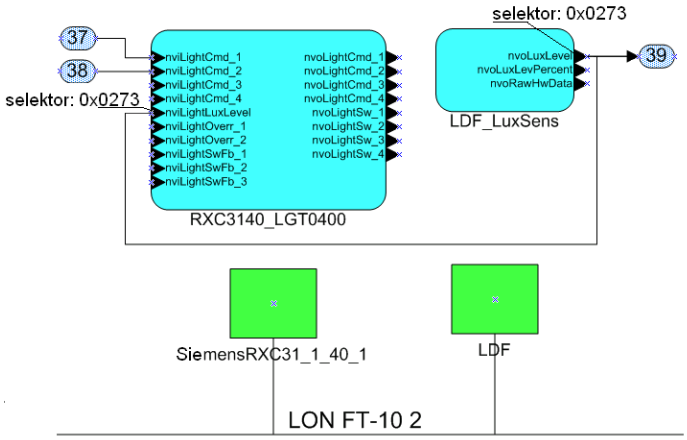
přímo vývojovými nástroji při programování aplikace pro procesor Neuron Chip, takovým nástrojem je např. Node Builder od firmy Echelon. Podrobněji je vytvoření XIF souboru pro mojí aplikaci popsáno v kap. 4.8.

2.1.6 Instalace sítě LonWorks

Existuje několik různých způsobů jak postupovat při instalaci sítě LonWorks, jednotlivé způsoby se liší flexibilitou, složitostí a vhodností pro různé aplikace dle velikosti instalované sítě. Dále se budu zabývat pouze způsobem instalace, který je označován jako *engineered system installation scenario*, česky jej budu nazývat *instalace s dopředným návrhem*, tento způsob instalace jsem používal při instalaci mé aplikace do sítě LonWorks. Další způsoby instalace jsou přehledně popsány v [26]. Instalace s dopředným návrhem se provádí ve dvou krocích:

1. Ve *fázi návrhu* je navržena celá struktura řídicí aplikace, tj. jsou zvolena jednotlivá zařízení a jsou vybrány způsoby fyzického propojení zařízení (fyzická vrstva dle ISO/OSI modelu). Provádí se propojování výstupních SP se vstupními SP jiných zařízení (tzv. binding). Lze také nastavit hodnoty konfiguračních parametrů pro jednotlivá zařízení. Všechny tyto konfigurační informace jsou ukládány do databáze konfiguračního nástroje, celý návrh lze tedy provádět bez fyzicky existující (instalované) sítě LonWorks, je zapotřebí pouze instalační nástroj a profily zařízení (XIF soubory). Na obr. 2.4 je příklad grafické reprezentace konfigurace sítě s dvěma zařízeními, která jsou propojena síťovou proměnnou typu SNVT_lux.

2. V druhé *realizační fázi* je konfigurační nástroj již připojen k fyzické síti LonWorks a konfigurace celé sítě je z databáze nahrávána do jednotlivých zařízení. Proces instalace (zapojení) zařízení do sítě LonWorks je označován jako *commision*, já budu dále používat termín *instalace zařízení do sítě*. Algoritmus instalace zařízení do sítě je podrobně popsán v 3.8.1. Při instalaci zařízení se také nahrávají do zařízení hodnoty KP a lze také nahrát aplikační program do zařízení.



Obrázek 2.4: Příklad konfigurace sítě LonWorks v aplikaci LonMaker.

2.2 Přístupy ke sběrnici LonWorks

Úkolem mé práce bylo vytvořit zařízení, které bude sloužit pro monitorování a řízení zařízení zapojených v síti LonWorks. Musel jsem se seznámit se způsoby, jakými lze v současné době realizovat přístup ke sběrnici LonWorks. V následujících odstavcích je uveden

rozbor jednotlivých možností, které se v současné době nabízejí a jejich nejdůležitější vlastnosti včetně nákladů na nezbytné vývojové prostředky. Nakonec je v odstavci 2.2.3 uvedeno jaké řešení jsem zvolil a z jakých důvodů.

2.2.1 Embedded zařízení

2.2.1.1 Neuron Chip (hosted nodes)

Většina zařízení pro LonWorks je založena na mikroprocesoru Neuron Chip (podrobněji je popsán v kap. 2.1.2). Neuron Chip obsahuje implementaci celého protokolu LonTalk. Uživatelská aplikace běží přímo v procesoru Neuron Chip a umožňuje pracovat maximálně s 62 síťovými proměnnými. K naprogramování procesoru Neuron Chip je potřeba vývojové prostředí NodeBuilder (cena cca 6595 USD u distributora EBV dle ceníku z prosince 2006).

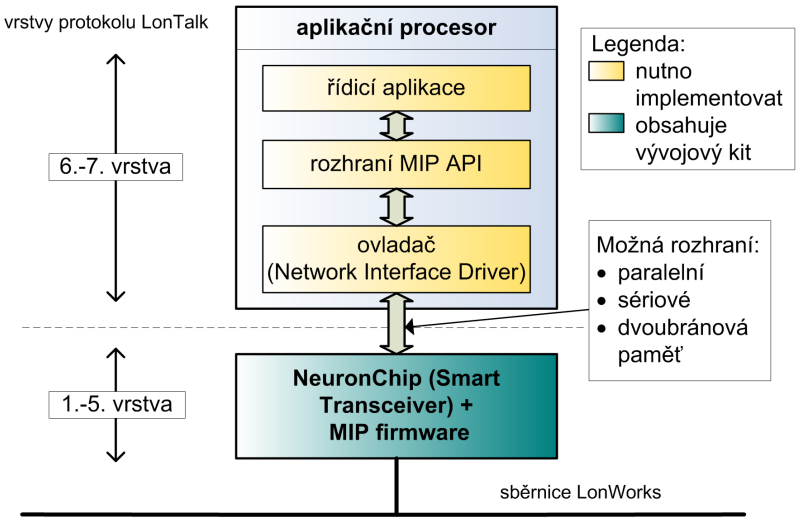
2.2.1.2 Host-based nodes, MIP

Tento způsob přístupu využívá Neuron Chip pouze jako komunikační koprocesor, je potřeba do něj nahrát speciální MIP (Microprocessor Interface Program) firmware, samotná uživatelská aplikace běží na jiném libovolném *aplikačním procesoru* (v anglické literatuře nazýván *host processor*). Tento přístup je vhodný pro aplikace, které není možné realizovat pomocí samotného procesoru Neuron Chip např. z důvodu náročnosti uživatelské aplikace na výpočetní výkon nebo na velikost paměti nebo na počet potřebných síťových proměnných (v této konfiguraci je možné použít až 4096 síťových proměnných).

Neuron Chip v tomto módu zajišťuje pouze 2. – 5. vrstvu protokolu LonTalk, vrstvy 6 a 7 je potřeba implementovat v aplikačním procesoru. Komunikace mezi procesorem Neuron Chip a aplikačním procesorem je specifikována zvláštním protokolem Network Interface Protocol (viz [12]), který musí být implementován v podobě ovladače (tzv. Network Interface Driver). Na úrovni fyzické vrstvy může komunikace probíhat po lince RS232 (označováno SLTA), pomocí paralelního rozhraní (MIP/P20, MIP/50) nebo pomocí dvoubránové paměti (MIP/DPS), viz obr. 2.5. K vývoji je třeba MIP Developer's Kit, který stojí cca 10 000 USD. Z hlediska softwaru je nutné v aplikačním procesoru implementovat Network Interface Driver a 6.–7. vrstvu protokolu LonTalk.

2.2.1.3 Short Stack Development Kit

Jedná se o další variantu, ve které Neuron Chip vystupuje v roli komunikačního koprocesoru a samotná aplikace běží na aplikačním procesoru. Short Stack Development

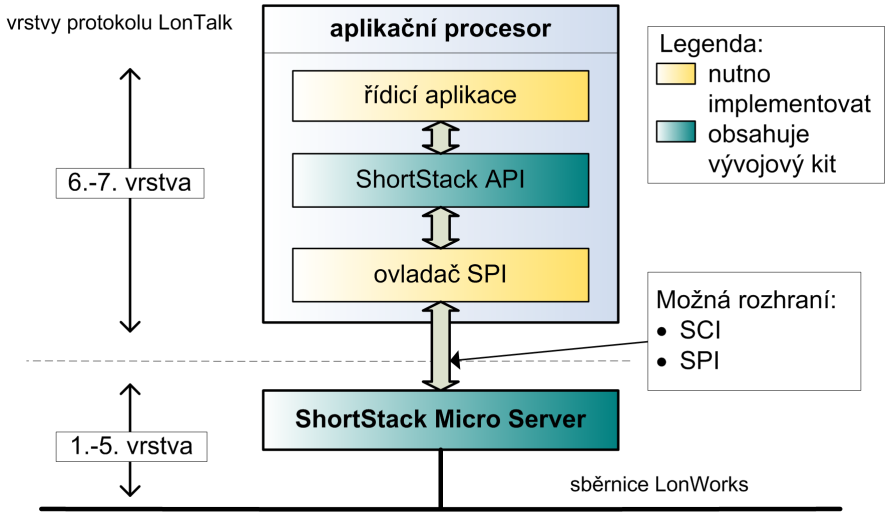


Obrázek 2.5: Architektura host-based zařízení.

Kit je zdarma poskytován firmou Echelon za podmínky použití procesoru Neuron Chip nebo Smart Transceiver ve vyvíjeném zařízení, kit obsahuje také speciální firmware pro *Neuron Chip* tzv. *Short Stack Micro Server*. V tomto případě je možné použít maximálně 62 síťových proměnných a nelze použít konceptu dynamických proměnných. Komunikace mezi procesorem Neuron Chip a aplikačním procesorem je realizována pomocí sériových rozhraní SCI nebo SPI. Z hlediska softwaru je nutné v aplikačním procesoru pouze naprogramovat ovladač *Serial Driver*, což je rozhraní, které v aplikačním procesoru poskytuje přístup k periferiím SCI nebo SPI pro vrstvu ShortStack API (ta je dodávána ve zdrojovém kódu jako součást Short Stack Development Kit). Architektura zařízení při použití Short Stack Development Kit je na obr. 2.6.

2.2.1.4 Procesor LC3020

Procesor LC3020 je procesor firmy Loytec, který má hardwarově implementovány vrstvy 1 a 2 dle ISO/OSI modelu protokolu LonTalk a je tak zřejmě jediným přímým konkurentem procesoru Neuron Chip. Pro zbylé vrstvy protokolu LonTalk, které je třeba implementovat v LC3020 lze využít *ORION Protocol Stack*. To je komerční implementace v ANSI C opět firmy Loytec, která je přenositelná na více operačních systému, neboť používá zvláštní vrstvu pro přístup k operačnímu systému OSSI - Operating System Services Interface. Může běžet na systémech Windows, RTEMS a Linux. ORION Protocol Stack podporuje 4096 síťových proměnných. Cena nástrojů pro vývoj zařízení s LC3020 je 19900 EUR.



Obrázek 2.6: Architektura zařízení s využitím Short Stack Development Kit.

2.2.2 Přístup z PC

Především pro konfiguraci a správu sítě LonWorks (např. pomocí nástroje LonMaker) je vhodné mít k síti přístup z PC. Hardwarově je toto většinou řešeno síťovými rozhraními pro PC, která jsou vyráběna několika výrobci (Např.: Echelon, Loytec, Gesytec) obvykle ve formě karty do PCI slotu nebo jako převodník na LPT či USB port. Z hlediska funkčního se tato rozhraní téměř neliší ani v závislosti na výrobci ani v závislosti na provedení (PCI, LPT, USB). Odlišují se především softwarem, který lze spolu s daným rozhraním použít, proto jsou následující odstavce členěny dle softwarového řešení.

2.2.2.1 LNS Server

LNS Server pochází z dílny firmy Echelon, ale lze jej použít i s rozhraními fy. Loytec. LNS Server je nazýván operačním systémem pro síť LonWorks, je to v podstatě server pro model server-klient. Aplikace jako např. *LonMaker* nebo OPC servery *EasyLON OPC server L*, *IPLONGATE* vystupují v roli klienta. Díky tomuto řešení klient-server je možné mít LNS Server spuštěn na jednom serverovém počítači a klienti (aplikace) mohou být spouštěny z počítačů připojených přes lokální ethernet či internet.

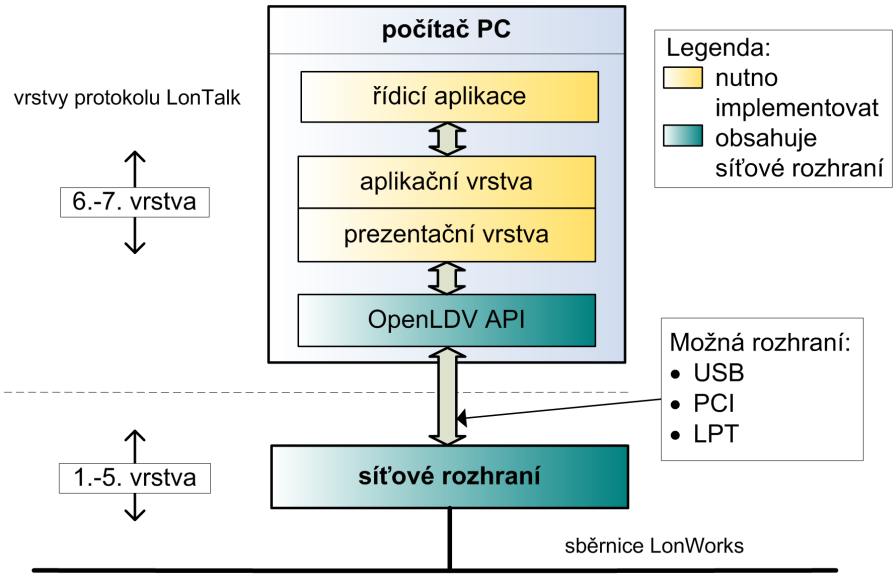
LNS Server poskytuje klientům (aplikacím) objektový přístup k síti LonWorks pomocí technologií COM a ActiveX (je tedy závislý na operačním systému Windows). Nabízí objekty, pomocí kterých lze efektivně instalovat, spravovat a monitorovat celou síť, lze např. provádět instalaci nových zařízení do sítě (včetně routerů), provádět propojování zařízení

síťovými proměnnými (binding), vytváření podsítí a kanálů. LNS Server využívá pro ukládání všech dat o aktuální konfiguraci sítě vlastní databázi (LNS global database), která obsahuje v podstatě kopii fyzické sítě (kopii konfigurací uložených ve všech zařízeních sítě).

Pro vývoj klientských aplikací je třeba *The LNS Application Developer's Kit*, který stojí 2745 USD. Lze ovšem také klientskou aplikaci vyvíjet na platformě Java, k dispozici je balík *LNS HMI Developer's Kit for the Java Platform*, který je zdarma.

2.2.2.2 OpenLDV

OpenLDV je API pro aplikace, které chtějí k síťovým rozhraním firmy Echelon[9] přistupovat na nízké úrovni, z hlediska ISO/OSI modelu je to přístup k 5. vrstvě (obr. 2.7).



Obrázek 2.7: Architektura OpenLDV.

Pomocí tohoto API lze přijímat a odesílat většinu zpráv LonTalk protokolu, některé zprávy obslouží přímo síťové rozhraní a dále je nepropouští, je to většina konfiguračních zpráv (viz 4.3.2.3). Síťové rozhraní zde vystupuje ve stejné roli jako komunikační koprocesor v MIP architektuře.

Rozhraní OpenLDV API je téměř shodné s MIP API. Lze tedy říci, že implementaci 6. a 7. vrstvy protokolu LonTalk pro OpenLDV API lze použít také pro aplikaci založené na MIP API, podrobnosti jsou uvedeny v kap. 4.3.3.

[9] Přehled síťových rozhraní je v tab. 4.1.

2.2.2.3 ORION Stack

K rozhraním firmy Loytec je dodávána knihovna `orion.dll`, která obsahuje implementaci ORION API tj. API pro přístup k ORION Stacku, připomeňme, že ORION Stack pro mikroprocesory není volně k dispozici a je nutné jej zakoupit. Použití ORION Stacku usnadňuje velké množství funkcí, které ORION API poskytuje. Hlavní rozdíl mezi MIP aplikacemi a aplikací založené na ORION Stacku spočívá v tom, že ORION Stack obsahuje implementaci 6. a 7. vrstvy protokolu LonTalk, zatímco OpenLDV toto neobsahuje. Shrnuto, ORION Stack nabízí na PC nejlepší přístup k síťovému rozhraní na nízké úrovni.

Ještě poznamenejme, že ORION Stack je dodáván k rozhraním firmy Loytec, ale lze k těmto rozhraním přistupovat také stejně jako k OpenLDV (MIP) pomocí čtyř základních funkcí `ldv_open`, `ldv_close`, `ldv_read`, `ldv_write`.

2.2.3 Výběr řešení pro přístup do sítě LonWorks

Pro vytvoření zařízení, které může komunikovat s ostatními zařízeními v síti LonWorks je důležité, aby takové zařízení mohlo zpracovávat (přijímat a odesílat) dostatečně velké množství síťových proměnných. Tuto podmínku splňuje:

- zařízení s architekturou *Host-based nodes* (viz odstavec 2.2.1.2),
- aplikace pro PC používající *LNS Server* (viz odstavec 2.2.2.1),
- aplikace na PC využívající rozhraní OpenLDV API, resp. MIP API (viz odstavec 2.2.2.2),
- aplikace na PC využívající ORION Stack (viz odstavec 2.2.2.3).

Pro realizaci mého zařízení jsem vybral řešení založené na počítači PC s operačním systémem Windows a přístupu do sítě LonWorks pomocí síťového rozhraní s ovladačem, který poskytuje OpenLDV API nebo MIP API. Toto řešení jsem vybral především kvůli minimální závislosti na softwaru či hardwaru nějaké konkrétní firmy, jak by tomu bylo v případě použití LNS Serveru nebo ORION Stacku. Rozhraní OpenLDV je téměř shodné s rozhraním MIP, které ke svým síťovým rozhraním pro přístup k síti LonWorks dodávají i jiní výrobci (např. Loytec, Gesytec) (více o přechodu mezi rozhraními OpenLDV a MIP lze nalézt v kap. 4.3.3). Navíc ovladače s rozhraním MIP dodávají někteří výrobci i pro jiné operační systémy (např. Linux, Windows CE).

Také jsem pro vývoj mé aplikace nemusel zakupovat žádné vývojové nástroje, jak by tomu bylo např. v případě použití LNS Serveru nebo architektury Host-based nodes. Výhodou rovněž je, že pro běh aplikace založené na této architektuře je zapotřebí pouze software dodávaný výrobcem spolu s hardwarem. Toto je rozdíl oproti aplikacím, které jsou založeny na LNS Serveru, protože ten je třeba distribuovat s aplikacemi. Při takto zvolené architektuře bylo nutné implementovat prezentační a aplikační vrstvu protokolu LonTalk, proto je rozboru těchto vrstev protokolu věnována kapitola 3.

2.2.3.1 Cenová rozvaha vybraného řešení

Výše nákladů na vývojové prostředky pro vývoj zařízení (dále jen *náklady*) je důležitým faktorem při rozhodování o koncepci zařízení. Proto je v tomto odstavci uvedeno porovnání nákladů pro čtyři varianty zmíněné výše, náklady jsou shrnuty v tab. 2.5. Ceny jsou z ceníku distributora EBV (www.ebv.com) z prosince 2006, kromě ceny *MIP Developer's Kits*, která byla zjištěna u pražského zastoupení distributora EBV v květnu 2007.

	LNS Server	Host-based nodes	ORION Stack	OpenLDV (MIP) API
LNS Turbo edition application development kit	2745 USD	×	×	×
LNS Turbo edition redistribution kit	2745 USD	×	×	×
MIP Developer's Kits	×	10000 USD	×	×
Součet	5490 USD	10000 USD	0 USD	0 USD

Tabulka 2.5: Cenová rozvaha nákladů na vývojové prostředky.

Bližší informace k jednotlivým položkám tabulky 2.5:

- *LNS Turbo edition application development kit* je nezbytný pro vývoj aplikace, která využívá LNS Server.
- *LNS Turbo edition redistribution kit* je licence nutná pro distribuci aplikací používající LNS Server třetím stranám.
- *MIP Developer's Kits* obsahuje speciální firmware pro Neuron Chip (podrobnosti viz v 2.2.1.2).

- Pro vývoj aplikace využívající *ORION Stack* nebo *OpenLDV API* na PC jsou třeba pouze DLL knihovny, které jsou distribuovány se síťovými rozhraními.

Je vidět, že z hlediska nákladů na vývojové prostředky je výhodné řešení používající *ORION Stack* nebo *OpenLDV*, ovšem nevýhodou při použití *ORION Stacku* je, že jej lze provozovat pouze s hardwarem firmy Loytec. Výhody použití *OpenLDV* byly zmíněny v předchozím odstavci. Je třeba ještě vzít v úvahu, že vývoj aplikace používající OpenLDV (MIP) je složitější o implementaci prezentační a aplikační vrstvy protokolu LonTalk.

Pokud by vyvíjené zařízení mělo být certifikováno sdružením LonMark, musel by výrobce být členem tohoto sdružení na úrovni *Partner*, toto členství stojí 5000 USD ročně[10]. Ovšem lze předpokládat, že výrobce většího množství modulů pro síť LonWorks (co do počtu prodaných kusů) je již členem sdružení LonMark.

[10] Dle informací z www stránek sdružení LonMark (www.lonmark.com).

Kapitola 3

Aplikační a prezentační vrstva protokolu LonTalk

V této kapitole je uveden podrobnější popis prezentační a aplikační vrstvy protokolu LonTalk. Důraz je kladen na ty oblasti, které bylo třeba implementovat v zařízení vyvíjeném v rámci této práce tj. v zařízení, které je založeno na OpenDLV API nebo MIP API s využitím *rozšířené množiny konfiguračních zpráv* (Extended Network Management Command Set - ECS, dále je budu označovat zkráceně *ECS zprávy*, podrobněji viz 3.7.1). Tato rozšířená množina byla specifikována ve verzi 709.1-B protokolu LonTalk a je nutná pro realizaci dynamických proměnných.

Jsou zde popsány služby, které tyto vrstvy poskytují nadřazeným vrstvám a jaké funkce sami vykonávají (např. správu konfiguračních dat). Je zde také popsán koncept dynamických proměnných, který jsem ve své aplikaci využil. Je zde také uvedeno jak probíhají (z hlediska aplikační vrstvy) některé operace při správě sítě LonWorks, konkrétně instalace zařízení do sítě, přidání a odebrání dynamické síťové proměnné.

3.1 Služby prezentační a aplikační vrstvy

Prezentační a aplikační vrstva nabízejí 6 služeb:

1. *Odesílání a příjem síťových proměnných* je služba, pomocí které může aplikace odesílat aktualizované hodnoty svých výstupních SP. Služba zajišťuje odeslání odpovídající zprávy všem příjemcům dle aktuální konfigurace, která je dána propojením síťových proměnných s ostatními zařízeními.

2. Služba pro *odesílání aplikačních zpráv* slouží aplikaci, pokud potřebuje komunikovat pomocí zpráv, které jsou specifické pouze pro danou aplikaci a které nejsou součástí protokolu LonTalk.

3. *Aliasing síťových proměnných* je služba, která umožňuje použití vícenásobných vstupních i výstupních SP. Tato služba je podrobně popsána v odstavci 3.9.1.

4. Služba *přenos cizích rámců* se používá pro přenos rámců jiného protokolu, např. BACnet – tzv. tunelování protokolu.

5. *Konfigurační zprávy* slouží pro aktualizaci a vyčítání konfiguračních dat. Toto je obvykle prováděno nástrojem pro správu sítě LonWorks, např. aplikací LonMaker. Obsluha konfiguračních zpráv je jádrem činnosti aplikační a prezentační vrstvy a je popsána v 3.7.

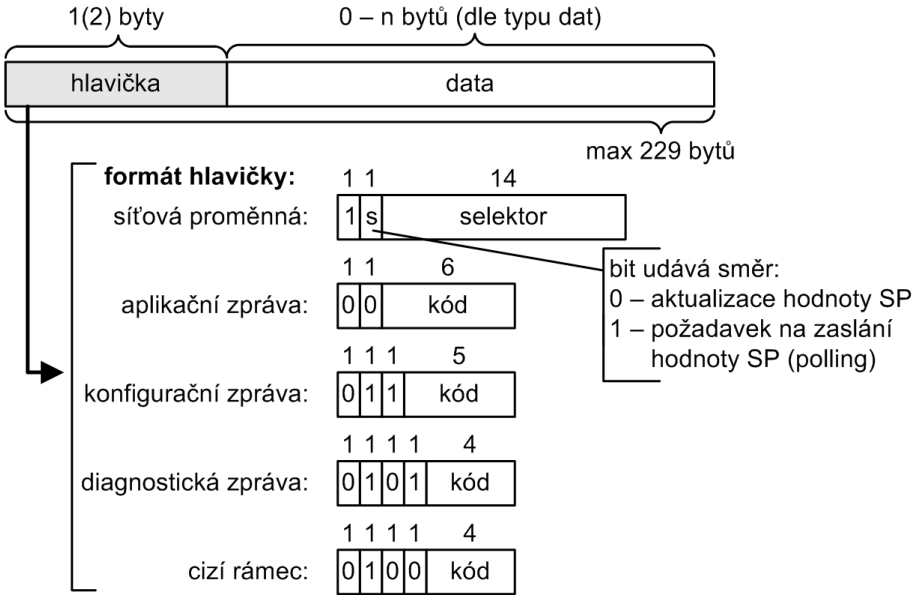
6. *Diagnostické zprávy* slouží pro zjištění stavu zařízení.

Nadřazená vrstva, tj. samotná řídicí aplikace, používá pouze službu pro odesílání a příjem SP a případně ještě službu pro odesílání aplikačních zpráv. Ostatní služby pracují uvnitř aplikační vrstvy, můžeme říci, že pracují na pozadí, aplikace o jejich činnosti není nijak informována. Jak již bylo řečeno výše, aplikace pouze odesílá a přijímá data SP a způsob jakým jsou odesílány (pomocí jaké služby, jakým zařízením, zda bude použita autentizace atd.) je záležitostí protokolu LonTalk a je to určeno konfigurací, která je nastavována zpravidla konfiguračním nástrojem. Správa SP a těchto konfiguračních dat je hlavním úkolem aplikační vrstvy.

3.2 Aplikační rámec

Aplikační rámec - část zprávy, která je zpracovávána aplikační vrstvou (APDU - Application Protocol Data Unit) obsahuje hlavičku a data. Začlenění aplikačního rámce v rámci celé zprávy protokolu LonTalk je na obr. A.1 v příloze A. Na obr. 3.1 je znázorněna struktura aplikačního rámce a způsob, jakým se rozlišují jednotlivé typy rámce podle toho, jakou zprávu rámec obsahuje. Jednotlivé typy rámců korespondují se službami, které aplikační vrstva nabízí (kromě aliasingu síťových proměnných). Pokud rámec obsahuje data síťové proměnné, je v jeho hlavičce selektor (o něm viz více v 3.3) a velikost hlavičky je v tomto případě 2 byty. Ostatní typy aplikačního rámce obsahují hlavičku

o velikosti 1 byte. V tomto případě část hlavičky obsahuje *kód zprávy*, dle kterého jsou rozlišována další data v rámci.



Obrázek 3.1: Aplikační rámec.

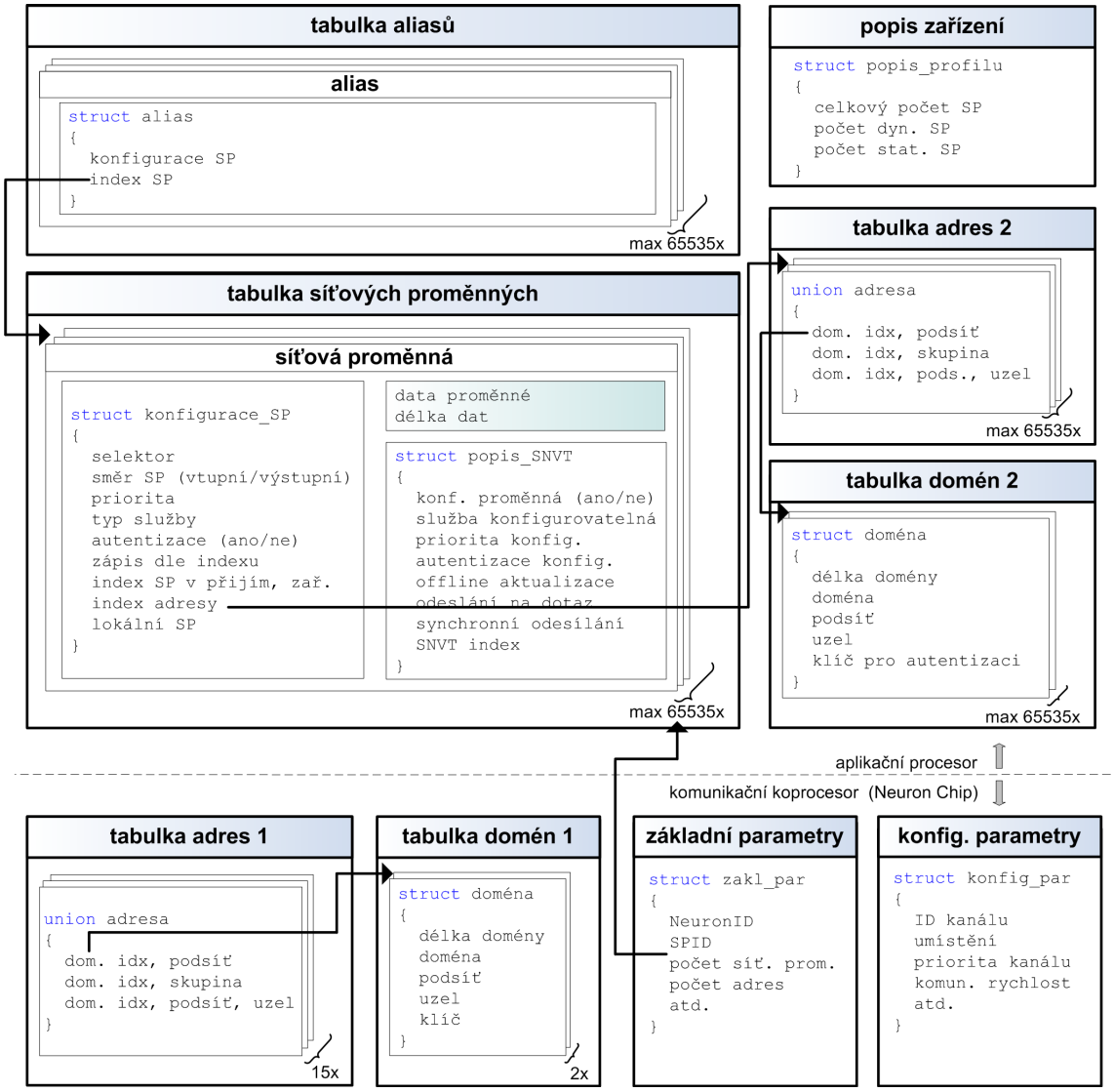
3.3 Selektor síťové proměnné

Selektor slouží jako identifikátor jednoho propojení SP, tedy výstupní SP jednoho zařízení má stejný selektor jako vstupní SP přijímajících zařízení (viz obr. 2.4). Každé propojení má v rámci celé sítě unikátní selektor. Selektor slouží pro identifikaci SP doručené do zařízení, SP v jednom zařízení tedy musí mít různý selektor. Selektor je součástí hlavičky aplikačního rámce (viz 3.2) a jeho hodnota je automaticky přiřazována konfiguračním nástrojem.

3.4 Konfigurační data protokolu LonTalk

Je-li zařízení tvořeno pouze procesorem Neuron Chip (jak je popsáno v 2.2.1.1), jsou všechna konfigurační data uložena přímo v něm (tato situace není na obr. 3.2 znázorněna). Dále budu popisovat pouze variantu, kdy je zařízení založeno na architektuře, ve které je Neuron Chip pouze v roli komunikačního koprocesoru, tj. na architektuře host-based

nodes nebo OpenLDV (viz 2.2.1.2 resp. 2.2.2.2) a využívá rozšířenou množinu konfiguračních zpráv (viz kap. 3.7.1). Tuto architekturu jsem použil pro svou aplikaci. V tomto případě jsou některá data uložena v procesoru Neuron Chip a některá v implementaci aplikační vrstvy v aplikačním procesoru jak je znázorněno na obr. 3.2. Popišme data dle toho, kde jsou uložena.



Obrázek 3.2: Konfigurační data protokolu LonTalk.

3.4.1 Konfigurační data v komunikačním koprocesoru

V komunikačním koprocesoru jsou uloženy:

- Informace týkající se *konfigurovatelných parametrů* (popis umístění zařízení, komunikační rychlost atd.), podrobnosti o těchto datech lze nalézt v dokumentaci výrobce procesoru Neuron Chip např. v [24] nebo v příloze B.6 v [1]. Tyto parametry jsou uchovávány v EEPROM paměti.

- Struktura *základních parametrů* obsahuje např. Neuron ID, SPID[1], počet položek v tabulce adres, počet síťových proměnných atd. Obsah této struktury nelze měnit konfiguračním nástrojem[2], proto je také někdy označována jako *read only structure.*

- *Tabulka domén 1* obsahuje dvě položky, protože zařízení může být členem maximálně dvou domén (viz 2.1.1.4). Položka obsahuje identifikátor domény, základní adresu a klíč pro autentizaci.

- Také je zde uložena *tabulka adres 1*, adresy z této tabulky se používají při příjmu zpráv. Přesněji řečeno, do této tabulky nahlíží síťová vrstva, když u přijaté zprávy rozhoduje, zda je tomuto zařízení určena. Tabulka může mít maximálně 15 adres a jednotlivé adresy (položky) mohou být různého typu (viz 2.1.1.4).

3.4.2 Konfigurační data v aplikačním procesoru

V aplikačním procesoru jsou uloženy zejména informace týkající se síťových proměnných:

- Položky *tabulky adres 2* jsou používány při odesílání hodnot výstupních SP. Tato tabulka se používá pouze u zařízení, která používají ECS zprávy[3] a může obsahovat až 65535 adres. Položky této tabulky jsou aktualizovány ECS zprávami.

- *Tabulka domén 2* se v aplikačním procesoru používá pouze, jsou-li využívány ECS zprávy. Tabulka může mít teoreticky až 65535 záznamů. Je třeba ale brát ohled na to, že tato tabulka se využívá pouze pro adresy při odesílání zpráv a případně při dotazování se na hodnotu SP.

- Ke každé proměnné se váže *konfigurace proměnné*, která obsahuje informace týkající se doručování SP (např. jakou službou se data odesílají, jakou mají prioritu atd.), na obr. 3.2 je konfigurace znázorněna strukturou `konfigurace_SP`[4].

[1] SPID - Standard Program ID, viz 2.1.5.3 na straně 14.
[2] Položky jako *počet adres 1* v tabulce adres mění Neuron Chip sám.
[3] Podrobněji o ECS zprávách viz kap. 3.7.1.
[4] Na obrázku jsou uvedeny pouze názvy položek, přesnou definici struktur v jazyce C lze nalézt v [1].

- U každé SP se také ukládá informace, která je dána jejím typem (o typech SP více v 2.1.3.1). Je zde také uložen SNVT Index. V obr. 3.2 jsou tyto data znázorněna strukturou `popis_SNVT`.
- *Hodnoty SP* lze považovat za konfigurační data, pouze pokud to jsou konfigurační parametry implementované pomocí síťových proměnných. Pak je třeba je také ukládat, jinak jsou hodnoty SP udržovány pouze v paměti RAM[5].
- *Popis zařízení* (Self-Identification Structures) obsahuje informace uložené v profilu zařízení. Tj. například maximální počet SP, aktuální počet SP, počet funkčních profilů a jejich ID atd. Tyto informace uložené v zařízení musí být v souladu s informacemi, které jsou uloženy v XIF souboru tohoto zařízení. Tyto informace jsou čteny resp. aktualizovány konfigurační zprávou *čtení paměti* resp. *zápis do paměti*, viz přehled zpráv v tab. 3.1.

3.5 Odesílání a příjem síťových proměnných

Je-li z rozhraní OpenLDV nebo MIP přijata zpráva, která je aktualizací některé SP, je úkolem aplikační vrstvy dle selektoru aktualizovat hodnotu SP a případně informovat nadřazené vrstvy (řídicí aplikaci) o změně hodnoty SP.

Při aktualizaci hodnoty výstupní SP aplikace aktualizuje hodnoty dané SP a aplikační vrstva protokolu LonTalk se stará (s využitím všech nižších vrstev) o odeslání hodnoty SP všem zařízením, ke kterým je SP připojena. K odeslání zprávy může aplikační procesor využít adresy uložené v komunikačním koprocesoru, kterému ve zprávě k odeslání předává pouze index do *tabulky adres 1* v komunikačním koprocesoru. Tento způsob je označován jako *implicitní adresování*. Druhý způsob je označován jako *explicitní adresování* a je používán spolu s ECS zprávami, aplikační procesor předává komunikačnímu procesoru zprávu včetně cílové adresy (hodnotu adresy získá z *tabulky adres 2*).

[5] RAM - Random Access Memory.

3.6 Dynamické proměnné

Koncept *dynamických proměnných* byl uveden ve verzi 709.1-B protokolu LonTalk spolu s ECS zprávami (viz kap. 3.7.1), které jsou potřeba pro funkci tohoto konceptu. Dynamické proměnné jsou SP, které lze do profilu zařízení přidávat nebo odebírat za běhu zařízení pomocí konfiguračního nástroje. Při klasickém přístupu je množina SP daného zařízení dána pevně jeho profilem, tyto SP jsou označovány jako *statické proměnné*.

Dynamické proměnné jsou vhodné zejména pro univerzální aplikace, jejichž rozhraní se přizpůsobuje speciálně vždy pro danou instalaci, mezi takovéto aplikace patří zejména aplikace sloužící pro monitorování ostatních zařízení. Dynamické proměnné jsou obvykle přidávány konfiguračním nástrojem do nového *virtuálního funkčního bloku*. Maximální počet dynamických proměnných pro jedno zařízení je dán konfigurací v XIF souboru. Teoretické maximum je 65535 SP. Konkrétní realizace (algoritmus) přidání a odebrání dynamické proměnné je popsán dále, viz kap. 3.8.2 a 3.8.3.

3.7 Konfigurační zprávy

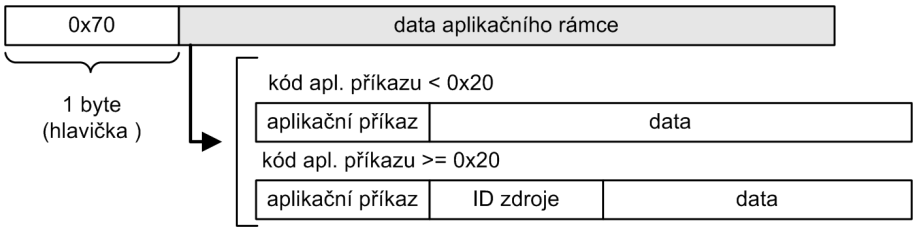
Konfigurační zprávy (Network Management Commands) slouží k aktualizaci konfiguračních dat, většinou jsou zasílány pomocí služby dotaz/odpověď. Zprávy jsou většinou požadavkem na inicializaci, čtení nebo aktualizaci nějaké položky v některé z konfiguračních tabulek, které jsou znázorněny na obr. 3.2. Konkrétní požadavek a struktura zasílaných dat se rozlišuje dle *kódu zprávy*, který je součástí hlavičky aplikačního rámce (viz 3.2) a případně ještě dle velikosti přijatých dat. Přesné datové struktury lze nalézt v [1] v příloze A. Konfigurační zprávy se používají zejména při instalaci zařízení do sítě a při propojování síťových proměnných. Přehled nejpoužívanějších konfiguračních zpráv a jejich stručný popis je v tab. 3.1, ve které je také uvedena odpovídající hlavička aplikačního rámce (její struktura viz obr. 3.1). V tab. 3.1 je také ve sloupci *Apl. proc.* uvedeno, zda je konfigurační zpráva předávána komunikačním koprocesorem aplikačnímu procesoru v případě MIP nebo OpenLDV architektury, je-li zpráva předávána, je ve sloupci A (Ano), jinak je ve sloupci uvedeno N (Ne). Vysvětlení tohoto mechanismu je v odstavci 4.3.2.3.

Zpráva	Apl. proc.	Hlavička (hexadecimálně)	Stručný popis
Identifikace zařízení	N	61	Slouží k zjištění NeuronID a Standard Program ID (SPID).
Aktualizace domény	N	63	Aktualizuje identifikátor jedné z domén zařízení. Lze použít také pro zneplatnění klíče pro autentizaci.
Nulování domény	N	64	Zneplatňuje identifikátor domény a nuluje klíč pro autentizaci.
Aktualizace klíče	N	65	Aktualizuje klíč pro autentizaci přičtením zaslané hodnoty k současnému klíči.
Aktualizace adresy	N	66	Aktualizuje položku tabulky adres (tabulka adres 1 na obr. 3.2.)
Čtení adresy	N	67	Slouží k čtení jedné položky (adresy) z tabulky adres (tabulka adres 1 na obr. 3.2.).
Čtení konfigurace SP	A	68	Umožňuje vyčíst konfiguraci SP nebo aliasu SP. Odpověď vrací obsah struktury `konfigurace_SP`.
Aktualizace adresy typu skupina	N	69	Slouží k aktualizaci adresy typu skupina v tab. adres 1. Aktualizovaná adresa je indexována číslem skupiny.
Čtení domény	N	6A	Zjišťuje identifikátor jedné z domén zařízení.
Aktualizace konfigurace SP	A	6B	Aktualizuje konfiguraci SP nebo aliasu SP, tj. aktualizuje obsah struktury `konfigurace_SP`.
Nastavení stavu zařízení	A	6C	Mění stav zařízení (popis stavů viz 3.8.1.1).
Čtení paměti	N	6D	Umožňuje čtení z paměti zařízení, požadavek obsahuje čtecí adresu, typ paměti a počet čtených bytů.
Zápis do paměti	N	6E	Slouží pro zápis do paměti zařízení.
Instalační zpráva	A	70	Je odeslána po stisku servisního tlačítka na zařízení a využívá se pro přenos ECS zpráv, viz kap. 3.7.1.
Čtení profilu zařízení	A	72	Odpověď vrací data dle verze struktury pro uložení dat profilu zařízení, viz kap. 3.8.1.1 Poznámka 3.
Čtení hodnoty SP	A	73	Zjišťuje aktuální hodnotu síťové proměnné dle jejího indexu v tabulce SP.

Tabulka 3.1: Nejpoužívanější konfigurační zprávy.

3.7.1 Rozšířená množina zpráv

Rozšířená množina zpráv (ECS zprávy - Extended Network Management Command Set) je v protokolu LonTalk k dispozici od verze 709.1-B. Tyto zprávy byly přidány kvůli podpoře dynamických proměnných (viz 3.6) a kvůli rozšíření velikosti tabulek adres a domén na velikost maximálně 65535 položek místo stávajících 15 položek v tab. adres a 2 položek v tab. domén. Rozšířená množina zpráv je přenášena *instalační zprávou* (viz tab. 3.1), která byla původně určena v předchozích verzích protokolu pouze k vyslání identifikace zařízení po stisku servisního tlačítka. Základní formát *instalační zprávy* zůstal zpětně kompatibilní, pouze byly přidány další možnosti formátu a obsahu zprávy a ty jsou nazývány právě ECS zprávami. Pokud zařízení podporuje ECS zprávy, provádí konfigurační nástroj konfiguraci zařízení právě pomocí těchto zpráv a nikoliv pomocí "klasických" zpráv, které jsou uvedeny v tab. 3.1. Zda zařízení ECS zprávy podporuje se konfigurační nástroj dozví z popisu v XIF souboru.



Obrázek 3.3: Aplikační rámec s ECS zprávou.

Struktura aplikačního rámce, který přenáší instalační zprávu obsahující ECS zprávy je na obr. 3.3. Každá zpráva obsahuje aplikační příkaz, přehled nejpoužívanějších příkazů a jejich stručný popis je v tab. 3.2, úplný popis lze nalézt v [1]. Zprávy s aplikačním kódem obsahují také identifikátor tzv. *zdroje*, na kterém má být operace specifikovaná aplikačním příkazem provedena. Formát a význam dat je potom dán dvojicí (aplikační příkaz, identifikátor zdroje). Zdroje jsou ve skutečnosti především konfigurační tabulky (viz kap. 3.4). Přehled zdrojů a jejich identifikátorů je v tab. 3.3, kde jsou uvedeny pouze ty zdroje, které ECS zprávy používají[6].

[6] Zdrojů je definována více (viz [1] kap. 13.7.16), ale některé nejsou zatím využívány a některé jsou vynechávány v instalační zprávě u aplikačních příkazů s aplikačním příkazem menším než 0x20.

Aplikační příkaz	Kód (hexadecimálně)	Stručný popis
Identifikace	00	Slouží k vizuální identifikaci zařízení, obvykle rozsvícením signalizační LED diody.
Definice SP	02	Přidá nebo modifikuje dynamickou SP.
Odebrání SP	03	Odebere dynamickou SP.
Aktualizace konf. SP	06	Aktualizuje konfiguraci SP (strukturu `popis_SNVT`).
Nastavení SP	08	Umožňuje nastavení hodnoty SP, používají pouze konfigurační nástroje.
Inicializace	20	Inicializuje položku v tab. dané identifikátorem zdroje.
Aktualizace	25	Aktualizuje položku v tab. dané identifikátorem zdroje.

Tabulka 3.2: Přehled nejpoužívanějších aplikačních příkazů instalační zprávy.

Název zdroje	Kód zdroje	Stručný popis
Zařízení	1	Zařízení jako celek.
Tabulka domén	2	Tabulka domén v aplikačním procesoru.
Tabulka adres	3	Tabulka adres v aplikačním procesoru.
Konfigurace SP	5	Konfigurace SP (statických i dynamických).
Tabulka aliasů	6	Tabulka aliasů v aplikačním procesoru.

Tabulka 3.3: Přehled zdrojů používaných v instalační zprávě.

3.8 Algoritmy pro správu sítě

Správa sítě představuje především instalaci sítě (podrobněji viz 2.1.6) a poté její údržbu, tj. změnu propojení SP mezi zařízeními, přidání nebo odebrání dynamických proměnných u zařízení, která toto podporují a přidání či odebrání celého zařízení ze sítě LonWorks. Veškeré změny v konfiguraci sítě jsou zpravidla prováděny konfiguračním nástrojem a většina těchto změn je v zařízení zpracovávána aplikační vrstvou.

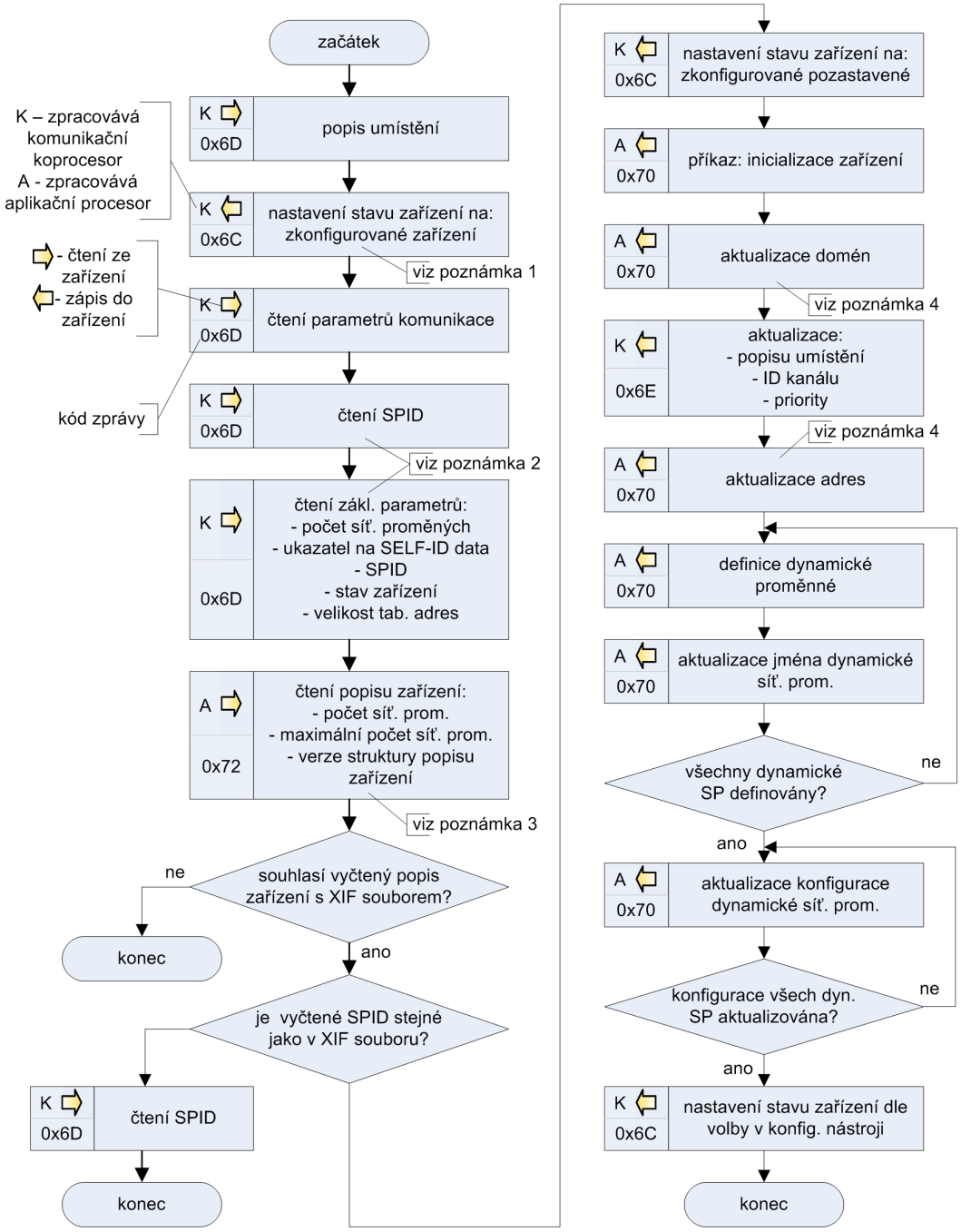
Algoritmy pro provádění změn v konfiguraci sítě používané konfiguračními nástroji nejsou součástí protokolu LonTalk. Protokolem není nijak specifikováno jaká konfigurační

data a v jakém pořadí mají být aktualizována, ani případně jaké informace mají být ověřeny, zda jsou v souladu s popisem zařízení v databázi konfiguračního nástroje (kam jsou importovány z XIF souborů). Protokol LonTalk pouze specifikuje, že po provedených změnách v konfiguraci má být zařízení uvedeno do stejného stavu v jakém bylo před provedením změn (viz kap. 13.5.1 v [1]) a že má být zařízení restartováno, pokud byly změněny parametry týkající se komunikace (např. změna komunikační rychlosti).

Dále popsané algoritmy vychází z mnoha pozorování instalace zařízení a měnění konfigurace zařízení vyvíjeného v této práci. Vše bylo prováděno pomocí nástroje LonMaker a komunikace byla sledována aplikace LPA - Loytec Protocol Analyzer. Aplikace LPA je volně ke stažení na webových stránkách firmy Loytec a byla pro mě při vývoji zařízení velmi užitečnou, takřka nepostradatelnou, pomůckou. Uváděné algoritmy vždy popisují činnosti, které provádí konfigurační nástroj.

3.8.1 Algoritmus instalace zařízení do sítě

V realizační fázi instalace sítě LonWorks jsou jednotlivá zařízení postupně instalována do sítě, tzn. konfigurace zařízení uložená v databázi konfiguračního nástroje je nahrávána do zařízení. Jsou aktualizovány hlavně tabulky adres a konfigurace proměnných, jsou-li použity dynamické proměnné, je do zařízení nahrávána také definice těchto proměnných. Také je kontrolována shodnost popisu profilu zařízení v XIF souboru (tj. popis, který je uložen v databázi konfiguračního nástroje) s informacemi, které jsou uloženy přímo v instalovaném zařízení. Dále popsaný algoritmus platí pro zařízení, které využívá rozšířenou množinu zpráv a nemá žádné konfigurační parametry. Na obr. 3.4 je znázorněn postup, který provádí konfigurační nástroj, když instaluje zařízení do sítě. V obrázku je také znázorněno, které zprávy jsou na straně zařízení zpracovávány komunikačním koprocesorem a které jsou zpracovávány aplikačním procesorem.



Obrázek 3.4: Algoritmus instalace zařízení do sítě.

3.8.1.1 Poznámky k algoritmu instalace zařízení

Poznámky se vztahují k jednotlivým krokům algoritmu, který je na obr. 3.4.

Poznámka 1:

Zařízení může být v jednom ze čtyř stavů, tento stav se ukládá do EEPROM[7] paměti komunikačního koprocesoru. Zařízení může být ve stavech:

- *Bez aplikace:* V tomto stavu je zařízení, pokud do něj ještě nebyla nahrána aplikace. V případě architektury *host-based nodes* nebo *OpenLDV* nemá tento stav praktický význam, protože aplikaci v aplikačním procesoru nelze přehrávat přes síť LonWorks jako u klasické koncepce (hosted nodes).
- *Nezkonfigurované:* V tomto stavu je již v zařízení nahrána aplikace, ale síťová konfigurace (tj. tabulky domén, adres a konfigurace síťových proměnných) ještě není v zařízení nahrána nebo právě probíhá její modifikace.
- *Zkonfigurované pozastavené:* Toto je stav, kdy aplikace i síťová konfigurace je nahrána, ale běh aplikace je pozastaven.
- *Zkonfigurované:* Tento stav označuje běžný provozní stav, aplikace i síťová konfigurace jsou nahrány a aplikace běží.

Poznámka 2:

Informace jsou vyčítány zprávou pro *čtení paměti* z oblasti *základní parametry*, která je v komunikačním koprocesoru (viz 3.4.1). Nejdříve je vyčítán pouze identifikátor profilu zařízení SPID (o něm viz 2.1.5.3) a poté je vyčítána větší oblast paměti, která obsahuje:

- počet SP (ten je 0, když Neuron Chip pracuje v režimu koprocesoru),
- ukazatel na data *popis zařízení*, ukazatel je roven 0xFFFF, když je Neuron Chip koprocesor,
- identifikátor profilu zařízení SPID,
- stav zařízení (viz poznámka 1),
- počet položek tabulky adres.

[7] EEPROM - Electrically Erasable Programmable Read Only Memory.

Poznámka 3:

Odpověď zařízení na dotaz zprávou *čtení profilu zařízení* je vytvořena zkopírováním požadovaného počtu bytů z datové struktury `popis_profilu`, ve které jsou informace o profilu zařízení uložené. Tato struktura existuje ve třech různých verzích a tato verze je uložena ve 4. bytu této struktury. Pokud zařízení používá rozšířenou množinu zpráv, musí používat strukturu verze 2 (verze jsou číslovány od 0). Bohužel struktura verze 2 není ve specifikaci protokolu LonTalk ([1]) správně a úplně popsána, což ztěžovalo implementaci odpovědi na požadavek čtení (konfiguračním nástrojem) této struktury z mého zařízení. Řešení jsem nalezl v bulletinu *Implementing Dynamic Network Variables* ([5]), kde je struktura pro popis profilu zařízení verze 2 uvedena spolu s dalšími doplňujícími informacemi. Deklaraci také uvádím ve výpisu 3.1.

```
typedef struct
{
  Word length;                  // offset 0x00
  Byte num_netvars_lo;          // offset 0x02
  Byte version;                 // offset 0x03
  Byte num_netvars_hi;          // offset 0x04
  Byte mtag_count;              // offset 0x05
  Word static_nv_count;         // offset 0x06
  Word current_nv_count;        // offset 0x08
  Word max_nv_in_use;           // offset 0x10
  Word alias_count;             // offset 0x12
  Word node_sd_text_length;     // offset 0x14
  Byte unused :6;               // offset 0x16
  Byte query_stats: 1;          // offset 0x17
  Byte binding_II: 1;           // offset 0x18
} Tsnvt_struct_v2;
```

Výpis 3.1: Deklarace struktury verze 2 pro popis profilu zařízení.

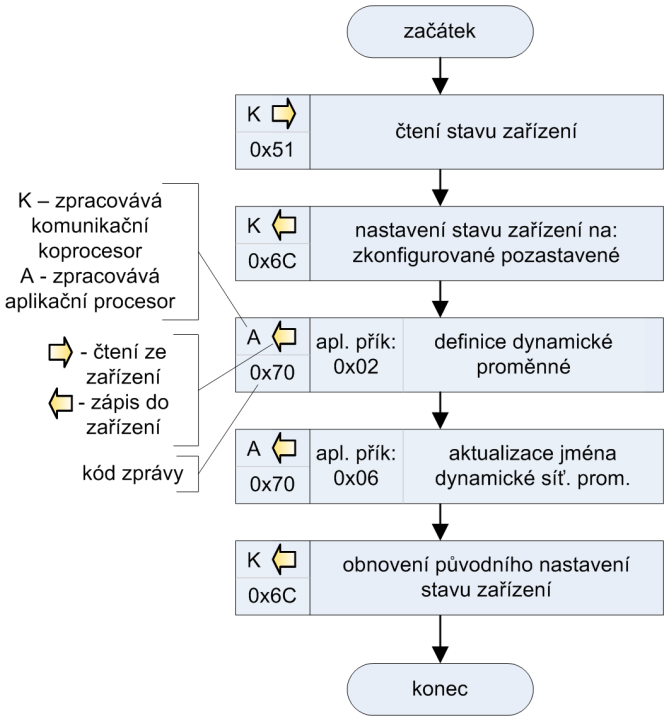
Poznámka 4:

Využívá-li zařízení rozšířené množiny zpráv, je tabulka domén a tabulka adres aktualizována jen pomocí těchto zpráv, nejsou využívány standardní zprávy (zprávy z tab. 2.1). Na toto je třeba brát ohled, protože tabulka adres v komunikačním koprocesoru (na obr. 3.2 označená jako *tabulka adres 1*) se využívá při příjmu zpráv ze sítě LonWorks a proto je na zodpovědnosti aplikační vrstvy v aplikačním procesoru, aby tuto tabulka také aktualizovala, více o tomto problému je uvedeno v 4.3.2.3.

3.8.2 Algoritmus přidání dynamické proměnné

U zařízení, která podporují dynamické proměnné, lze pomocí konfiguračního nástroje přidávat do profilu zařízení dynamické proměnné během instalace. Postup přidání dynamické proměnné do zařízení pomocí aplikace LonMaker je stručně popsán v příloze B.

Jednotlivé kroky, které konfigurační nástroj během přidávání dynamické proměnné provádí, jsou znázorněny na obr. 3.5. Nejdříve je instalační zprávou s aplikačním příkazem *definice SP* definována nová proměnná v tabulce SP na pozici dle daného indexu, který je součástí dat pro definici SP. Definicí SP je také zkonfigurována struktura `popis_SNVT` a některé položky struktury `konfigurace_SP`, zbývající položky (např. selektor) musí být inicializovány aplikační vrstvou. V dalším kroku je ještě aktualizováno jméno nově definované SP. Aplikační vrstva by také měla po přidání SP informovat nadřazenou vrstvu, tedy řídicí aplikaci.

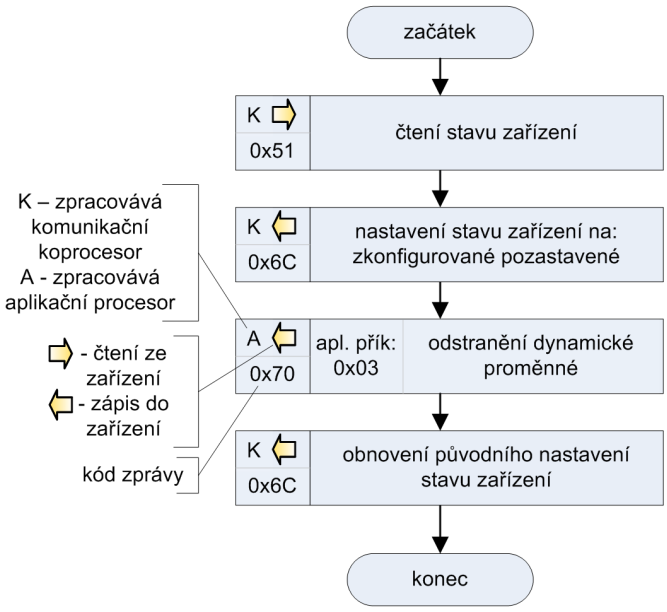


Obrázek 3.5: Algoritmus přidání dynamické proměnné do zařízení.

3.8.3 Algoritmus odebrání dynamické proměnné

Podobně jako lze dynamické proměnné do profilu zařízení přidávat, lze je také pomocí konfiguračního nástroje odebírat z profilu zařízení. Popis jak odebrat dynamickou proměnnou ze zařízení je uveden v příloze C. Na obr. 3.6 je uveden postup, kterým LonMaker

odebrání dynamické proměnné provádí. Pro odebrání proměnné je podstatná pouze jedna zpráva, která umožňuje odebrat více SP najednou. Zpráva obsahuje index SP, od kterého mají být SP odebrány a jejich počet.



Obrázek 3.6: Algoritmus odebrání dynamické proměnné ze zařízení.

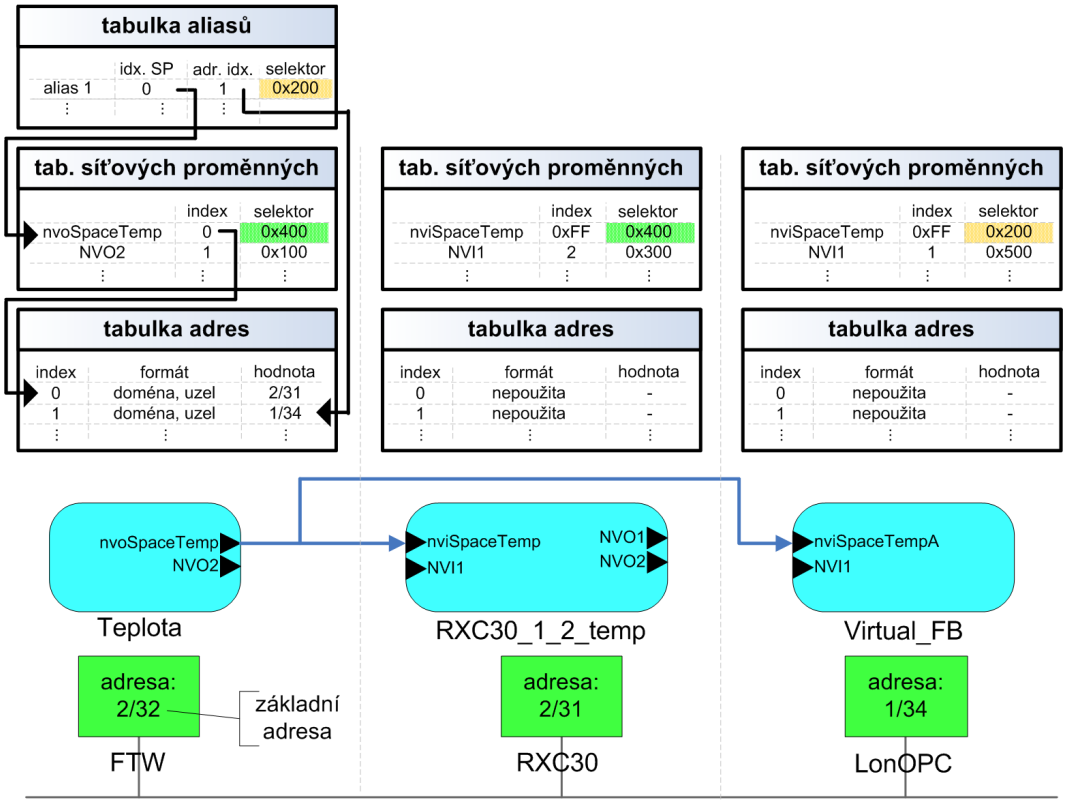
3.9 Propojení síťových proměnných 1:N

Někdy může nastat situace, kdy jsou síťové proměnné zapojeny ve vztahu 1:N. Tedy buď je výstupní proměnná jednoho zařízení spojena se vstupními proměnnými na více zařízeních nebo je spojena s několika různými vstupními proměnnými na jednom zařízení. Existují dva způsoby, jak jsou v těchto případech zasílány aktualizace hodnot síťových proměnných, oba jsou popsány v následujících odstavcích. Jaký způsob bude použit se volí při konfiguraci sítě.

3.9.1 Aliasing

Aliasing využívá tabulku *aliasů* uloženou v aplikačním procesoru, jejíž položky (aliasy) obsahují strukturu `konfigurace_SP` (stejnou jaká se používá pro klasické SP v tab. SP) a index na položku do tabulky SP, tedy index na některou SP (ta je nazývána *primární SP*). Aliasy jsou používány při odesílání aktualizace hodnoty SP. Nejdříve je aktuální hodnota SP odeslána na adresu, na kterou je uložen odkaz v tab. SP (ve struktuře

`konfigurace_SP`) a poté je prohledána tab. aliasů, zda se nějaký alias neodkazuje na právě zpracovávanou primární SP, pokud je nějaký takový alias nalezen, je aktuální hodnota odeslána také na adresu, která je uložena v aliasu (ve struktuře `konfigurace_SP`)[8]. Příklad takovéto konfigurace ilustruje obr. 3.7. Hodnota síťové proměnné nvoSpaceTemp je ze zařízení FTW nejdříve odeslána do zařízení RXC30, zpráva je tedy zaslána na adresu 2/31 (podsíť/uzel)[9] se selektorem 0x400. Poté je hodnota síťové proměnné nvoSpaceTemp také zaslána na adresu 1/34 (podsíť/uzel)[10] se selektorem 0x200. Aktualizace hodnoty je tedy v tomto případě přenášena pomocí dvou zpráv, což může zbytečně zvyšovat zátěž sběrnice.



Obrázek 3.7: Propojení SP 1:2 s využitím aliasingu.

3.9.2 Propojování pomocí adresy typu skupina

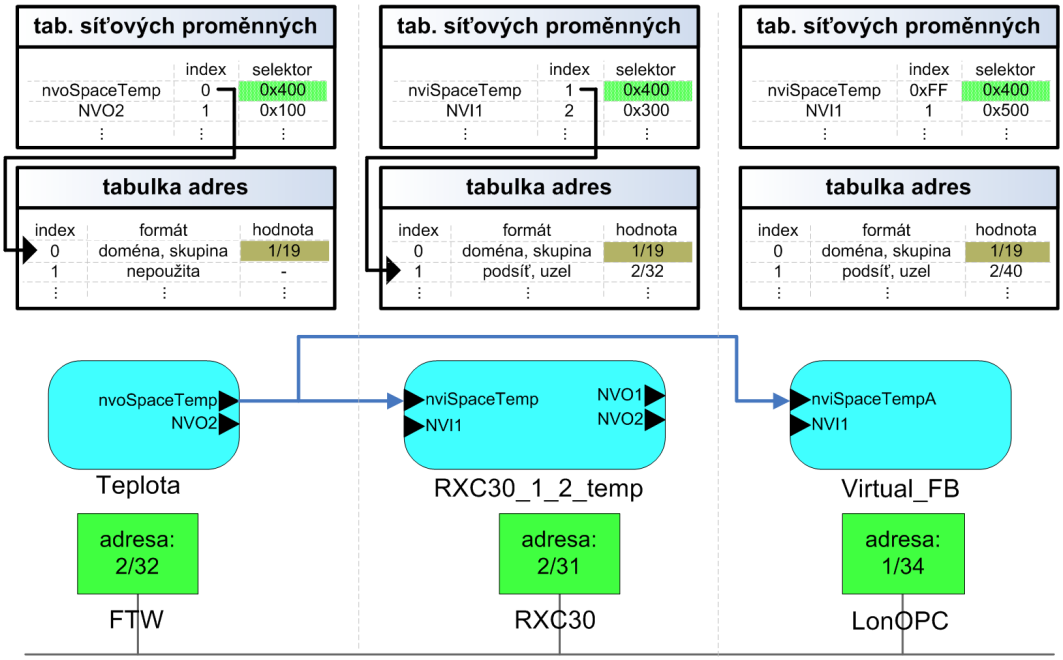
Pro odesílání hodnot SP, které jsou propojeny ve vztahu 1:N, lze s výhodou použít adresování typu doména/skupina. Tato varianta je graficky znázorněna na obr. 3.8. Zde

[8] Odeslání je samozřejmě provedeno také dle všech ostatních parametrů uložených ve struktuře `konfigurace_SP` v aliasu.
[9] Toto je základní adresa zařízení RXC30.
[10] Toto je základní adresa zařízení LonOPC.

zařízení FTW odesílá aktualizaci hodnoty výstupní proměnné nvoSpaceTemp se selektorem 0x400 na adresu skupiny 19, jejímiž členy jsou také zařízení RXC30 a LonOPC. Aktualizace je na obě přijímající zařízení odeslána jednou zprávou, což je výhodné z hlediska zatížení sběrnice.

Na tomto místě bych ještě zmínil jak probíhá příjem SP v zařízení. Nejdříve je rozpoznáno, zda je zpráva adresována zařízení. Toto je prováděno síťovou vrstvou a adresa přijaté zprávy je porovnávána nejen se základními adresami (mohou být dvě, viz 2.1.1.4), ale také s položkami tabulky adres (tabulka adres 1 u OpenLDV architektury). Tímto postupem je v uvedeném příkladě u zařízení RXC30 a LonOPC zachycena zpráva s adresou typu doména/skupina. Dalším krokem při příjmu je hledání dle selektoru v tab. SP, pokud je nalezena SP, která má stejný selektor jako v přijaté zprávě, je tato SP aktualizována.

Na obr. 3.8 stojí ještě za povšimnutí, že u vstupní SP u zařízení RXC30 je uložen odkaz do tabulky adres na adresu 2/32 (podsíť/uzel), která je adresou zařízení FTW. Jeli u vstupní SP definován odkaz do tab. adres, lze adresu využít pro dotazování se na hodnotu SP na této adrese. Jeli odkaz roven 0xFF[11], je to příznak toho, že zde není odkaz to tabulky adres.



Obrázek 3.8: Propojení SP 1:2 s využitím adresy typu skupina.

[11] V případě zařízení, které využívá rozšířenou množinu zpráv je index dvoubytový. Zde je tedy případně použita hodnota 0xFFFF.

Kapitola 4

Implementace aplikační vrstvy

Tato kapitola se zabývá implementací části aplikace pro monitorování a ovládání zařízení v síti LonWorks, kterou jsem vytvořil v rámci této práce. Je zde popsána část aplikace nazvaná *Rozhraní SP* (síťových proměnných), která zajišťuje spolupráci se síťovým rozhraním (OpenLDV nebo MIP[1]), zpracování aplikační vrstvy protokolu LonTalk a interpretaci síťových proměnných dle *standardních typů* specifikovaných sdružením LonMark.

Rozhraní SP jsem využil při implementaci OPC serveru (viz kap. 5). Pro účely vývoje a testování Rozhraní SP jsem vytvořil jednoduchou aplikaci s grafickým uživatelským rozhraním nazvanou OpenLDVStackDemo, která je popsána v odstavci 4.7.

Rozhraní SP je implementováno objektovým způsobem v jazyce C++ a pro překlad byl využíván překladač prostředí Visual Studio 2005.

4.1 Vlastnosti aplikace

Cílem bylo vytvořit aplikaci, pomocí které bude možné *monitorovat* a *ovládat* zařízení zapojená v síti LonWorks. Pojmem *monitorování* je myšleno vyčítání SP ze zařízení a pojmem *ovládání* je myšleno zápis hodnot SP do zařízení. Určujícím požadavkem také bylo vytvoření aplikace bez využití LNS Serveru (o něm viz kap. 2.2.2.1) především kvůli finanční náročnosti řešení, které by LNS Server používalo.

Dále bylo třeba, aby bylo možné konfigurovat, které SP kterých zařízení aplikace monitoruje, respektive ovládá. Proto bylo potřeba, aby bylo možné do množiny SP, které aplikace má jako své vstupy (resp. výstupy), přidávat SP, které budou mít stejný standardní typ, jako SP na zařízeních, která hodláme monitorovat (ovládat). Toto bylo realizováno

[1] Rozdíly při použití OpenLDV API a MIP API jsou popsány v kap. 4.3.3.

s využitím dynamických proměnných (viz 3.6), které lze přidávat do aplikace pomocí konfiguračního nástroje, tj. konfigurace aplikace je prováděna v podstatě prostřednictvím protokolu LonTalk.

Počet síťových proměnných, které aplikace umožní sledovat je až 65535. Je třeba ale vzít v úvahu, že tato hranice je teoretická a v praxi těžko využitelná. A to zejména kvůli přenosové rychlosti fyzické vrstvy, kterou používá síťové rozhraní využívané aplikací k připojení do sítě LonWorks. Vezměme v úvahu například situaci s parametry:

- síťové rozhraní je připojeno k fyzické vrstvě TP/FT-10 s přenosovou rychlostí $v_{pren} = 78\ kbs^{-1}$ (viz tab. 2.1),
- aktualizace hodnot SP chceme vyčítat s periodu $T = 5$ sekund,
- průměrná velikost dat jedné SP je $apl_ram_{prum} = 8$ bytů,
- průměrná velikost hlavičky zprávy $hlav_{prum} = 12$ bytů (viz [13] kap. 8),
- všechny aktualizace hodnot SP budou zasílány službou bez potvrzování[2],
- k dispozici máme 20% šířky přenosového pásma[3] ($k_{pasmo} = 0,2$).

Potom maximální počet SP, které bychom mohli takto přijímat je dán vztahem:

$$\begin{aligned} max_poc_SP &= \frac{v_{pren} \cdot T \cdot k_{pasmo}}{(hlav_{prum} + apl_ram_{prum}) \cdot 8} \\ max_poc_SP &= \frac{78 \cdot 2^{10} \cdot 5 \cdot 0,2}{(12 + 8) \cdot 8} \\ max_poc_SP &= 499,2 \doteq 500 \end{aligned} \tag{4.1}$$

Aplikace by tedy byla vzhledem k zadaným parametrům (předpokladům) schopna vyčítat maximálně 500 síťových proměnných.

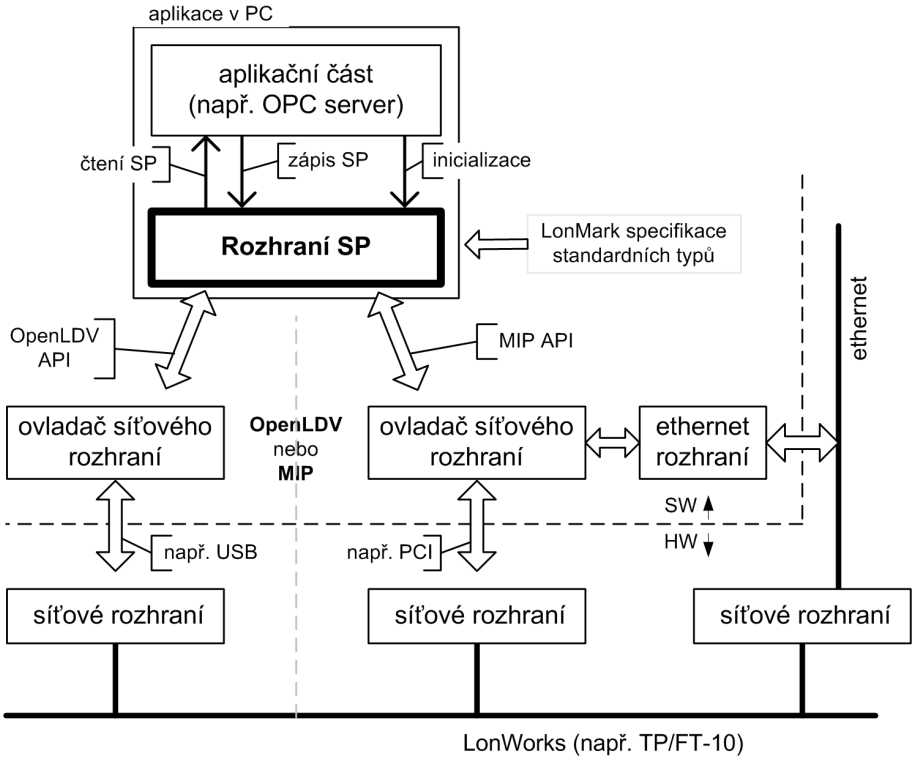
4.2 Architektura aplikace

Architektura celé aplikace včetně vazeb na hardware je znázorněna na obr. 4.1. Aplikace (program) je rozdělena na dvě části. První část (nižší vrstva) je zmiňované *Roz-*

[2] V případě použití služby s potvrzováním by bylo třeba ještě započítat velikost potvrzovací zprávy cca 10 bytů (přesná velikost zaleží na typu cílové adresy).

[3] Je třeba vzít v úvahu, že přenosový kanál mohou používat současně také ostatní zařízení a že při velkém vytížení bude docházet ke kolizím (i přesto, že je protokol LonTalk vybaven mechanismem predictive p-persistent CSMA viz 2.1.1.3).

hraní SP, jehož úkolem je zpracovávat protokol LonTalk (podrobněji viz 4.2.1) a poskytovat *aplikační části* aplikace (vyšší vrstvě) rozhraní pro čtení a zápis hodnot SP.



Obrázek 4.1: Architektura aplikace.

Rozhraní SP může využívat pro přístup k síti LonWorks hardware - tzv. síťové rozhraní, které poskytuje rozhraní OpenLDV API nebo MIP API, přehled takových síťových rozhraní je v tab. 4.1. Rozhraní SP jsem původně vyvíjel pouze pro OpenLDV API. Během práce došlo ale k poruše síťového rozhraní, které jsem používal (U10 od firmy Echelon). Protože jsem jiné rozhraní s OpenLDV API neměl k dispozici, upravil jsem Rozhraní SP tak, aby mohlo pracovat s MIP API, protože síťové rozhraní s tímto API jsem k dispozici měl. Aplikaci tedy nyní může být přeložena pomocí podmíněného překladu pro jedno či druhé rozhraní. Popis OpenLDV API je uveden v odstavci 4.3 a popis rozdílů a problémů při přechodu mezi OpenLDV API a MIP API je v odstavci 4.3.3.

Výrobce	Název	API	Připojení k PC	Fyzická vrstva[1]
Echelon	U10	OpenLDV	USB port.	TP/FT-10
Echelon	U20	OpenLDV	USB port.	PL-20
Echelon	PCLTA-21	OpenLDV	Karta do PCI slotu.	TP/FT-10, TP/XF-1250
Loytec	NIC709-USB	MIP	USB port.	TP/FT-10, RS-485, TP/XF-1250
Loytec	NIC709-PCI	MIP	Karta do PCI slotu.	TP/FT-10, RS-485, TP/XF-1250
Loytec	L-IP	MIP	Připojení přes ethernet.	TP/FT-10, TP/XF-1250
Gesytec	Easylon PCI-Bus Interface	MIP	Karta do PCI slotu.	TP/FT-10, TP/XF-1250
Gesytec	Easylon USB Interface	MIP	USB port.	TP/FT-10
Gesytec	Easylon Ethernet Interface	MIP	Připojení přes ethernet.	TP/FT-10

Tabulka 4.1: Přehled síťových rozhraní s OpenLDV API nebo MIP API.

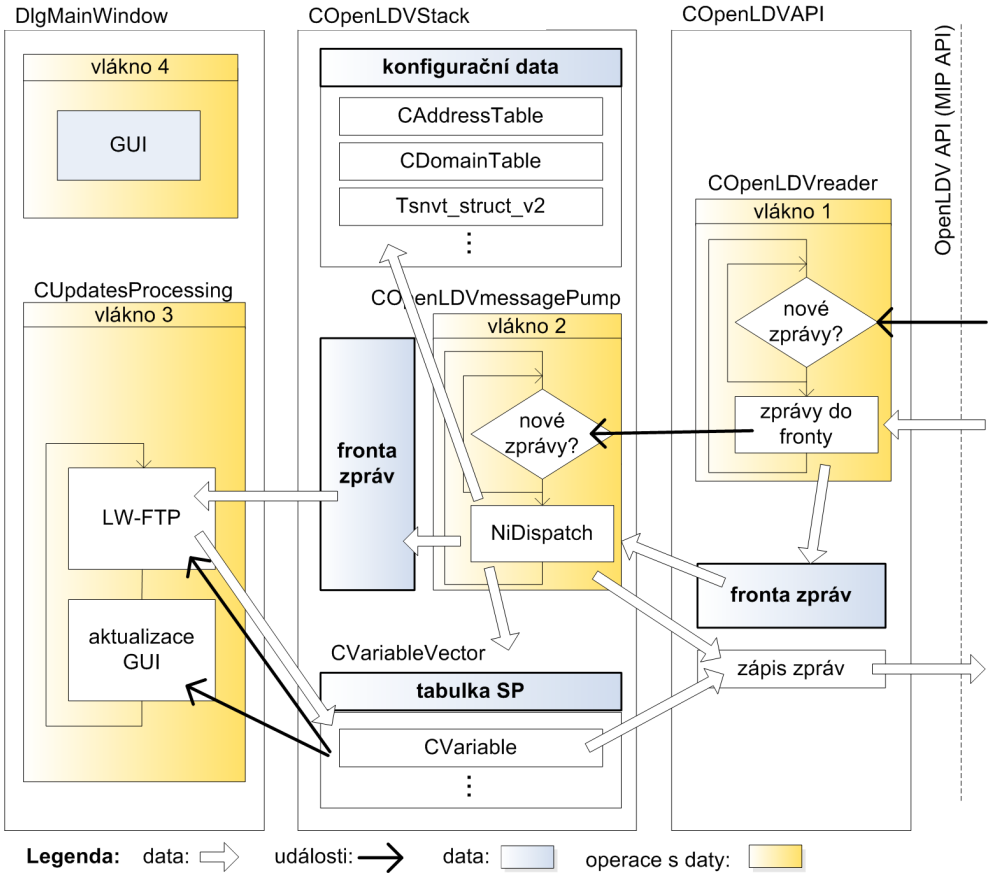
Jako *aplikační část* může být implementován např. nějaký podprogram, který monitorovaná data ukládá (archivuje) např. do databáze, nebo podprogram, který obsahuje řídicí algoritmy a který jako čidla a akční členy využívá zařízení, která komunikují pomocí sítě LonWorks. Já jsem v další části práce implementoval jako *aplikační část* OPC server, který poskytuje dostatečně obecné rozhraní pro další aplikace (alespoň na platformě operačního systému Windows), podrobněji viz kap. 5.

4.2.1 Architektura Rozhraní SP

Rozhraní SP lze rozdělit na tři logické části, každé části přísluší jedno vlákno, architektura je také zachycena na obr. 4.2. Komunikace mezi vlákny je založena na událostech a na datových frontách zpráv. Události slouží pro synchronizaci činnosti vláken a jsou

[1] Přehled jednotlivých fyzických vrstev a jejich vlastností viz tab. 2.1.

implementovány pomocí tzv. čekatelných objektů (viz heslo WaitForSingleObject v [23]). Při implementaci jsem použil některé třídy, které jsou jako příklad součástí instalačního balíku *OpenLDV Developer's Kit*, který je k dispozici ke stažení na webových stránkách firmy Echelon[4].



Obrázek 4.2: Architektura Rozhraní SP.

Třída `COpenLDVAPI` zapouzdřuje přístup k API ovladače (OpenLDV nebo MIP) síťového rozhraní a obsahuje vlákno (zapouzdřené ve třídě `COpenLDvReader`), které vyčítá přijaté zprávy do fronty zpráv. Třída `COpenLDVStack` obsahuje veškerá *konfigurační data aplikačního procesoru* (viz kap. 3.4.2) a zpracovává ve vláknu zapouzdřeném do objektu `COpenLDVMessagePump` veškeré přijaté zprávy, podrobněji viz dále v kap. 4.4. Třída `DlgMainWindow` je svázána s hlavním oknem aplikace OpenLDVStackDemo (viz 4.7) a obsahuje instanci třídy `CUpdatesProcessing`, jejíž vlákno aktualizuje (překresluje) okno aplikace, je-li to potřeba, a zpracovává hodnoty síťových proměnných, které se týkají přenosu dat pomocí LW-FTP (viz dále v 4.6).

[4] http://www.echelon.com/products/development/openldv/

4.3 OpenLDV API

4.3.1 Stručný popis OpenLDV API

Obecný popis možností a architektury OpenLDV byl již uveden v kap. 2.2.2.2, zde se zaměřím na stručný popis OpenLDV API samotného, detailní informace lze nalézt v [14]. OpenLDV API je dostupné v podobě dll knihovny `ldv32.dll`, která je součástí balíků *OpenLDV Developer's Kit*, *OpenLDV Installer* a také je dodávána spolu s ovladači k síťovým rozhraním firmy Echelon. Od února 2007 je také dostupná implementace pro Linux a Max OS X.

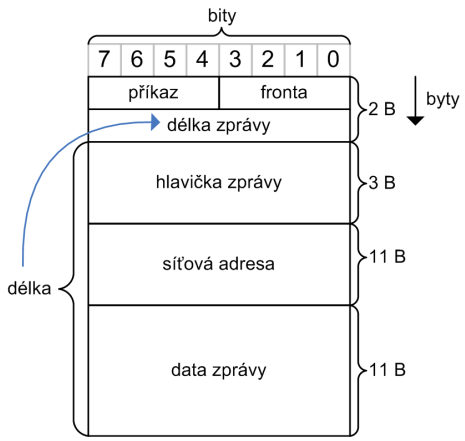
OpenLDV API je tvořeno šesti funkcemi (podrobnou dokumentaci lze nalézt v [14]):

- `ldv_get_version` slouží pro zjištění verze OpenLDV API,
- `ldv_open` slouží pro otevření spojení se síťovým rozhraním, funkce vrací identifikátor, který je odkazem na otevřenu instanci a používá se jako parametr dalších funkcí,
- `ldv_close` slouží k uzavření spojení se síťovým rozhraním,
- `ldv_read` slouží k vyčtení jedné příchozí zprávy ze síťového rozhraní,
- `ldv_write` slouží k odeslání zprávy do síťového rozhraní,
- `ldv_register_event` slouží k zaregistrování *objektu události* operačního systému Windows, pomocí kterého ovladač rozhraní informuje o přítomnosti příchozích zpráv ve frontě síťového rozhraní.

Formát (struktura) *OpenLDV API zprávy*, která je vracena funkcí `ldv_read`, respektive je předávána funkci `ldv_write` je na obr. 4.3 a ve zdrojovém kódu ji představuje typ (struktura) `ExpAppBuffer`.

Horní polovina prvního bytu zprávy označená *příkaz* určuje, zda je zpráva (odesílaná funkcí `ldv_write` resp. přijímaná funkcí `ldv_read`) určena k odeslání do (resp. přijmutí z) sítě LonWorks nebo do (resp. z) síťového rozhraní (komunikačního koprocesoru). Spodní polovina prvního bytu označená *fronta* určuje, zda jde o zprávu, která má být zařazena do fronty zpráv s vyšší prioritou. Byte *délka zprávy* určuje délku zprávy bez prvních dvou bytů, jak je naznačeno na obr. 4.3.

První byte je také používán vcelku (nedělený na poloviny) *příkazy pro ovládání síťového rozhraní* (tzv. *Immediate Commands*). Tyto příkazy slouží např. k nastavení stavu



Obrázek 4.3: Formát OpenLDV API zprávy.

síťového rozhraní (online/offline) nebo k odeslání standardní instalační zprávy[5] (simulace stisku servisního tlačítka).

Pole *hlavička zprávy* obsahuje informace, které popisují, jak má se zprávou síťové rozhraní pracovat, podrobný popis tohoto pole lze nalézt v [14]. Mezi nejdůležitější položky *hlavičky zprávy* patří: typ služby, jakou bude zpráva odeslána (resp. byla přijata), zda je použita autentizace a zda bude použito *implicitní* nebo *explicitní adresování*.

Pojem *implicitní adresování* označuje situaci, kdy jako síťová adresa, na kterou je zpráva odesílána je použita adresa z tabulky adres v síťovém rozhraní, tj. v komunikačním koprocesoru, v kap. 3.4 je tato tabulka označována jako *tabulka adres 1*. V tomto případě musí položka *tag* v poli *hlavička zprávy* obsahovat index do této tabulky.

Je-li použito *explicitní adresování*, je v poli *síťová adresa* obsažena přímo adresa, na kterou má být zpráva odeslána. Tato adresa je do pole *síťová adresa* doplněna aplikací (běžící v aplikačním procesoru), která volá funkci `ldv_write` a hodnota této adresy je nalezena v *tabulce adres 2* (značení dle kap. 3.4).

Část zprávy označená *data zprávy* obsahuje aplikační rámec protokolu LonTalk, jeho popis viz kap. 3.2.

4.3.2 Použití OpenLDV API

4.3.2.1 Připojení DLL knihovny

Jak již bylo uvedeno, OpenLDV API je implementováno v podobě DLL knihovny `ldv32.dll`. S touto knihovnou jsou také distribuovány další potřebné soubory: hlavičkový

[5] Viz tab. 3.1 v kap. 3.7.

soubor `ldv32.h`, který obsahuje deklarace funkcí a soubor `ldv32.lib`, který obsahuje reference potřebné k implicitnímu připojení dynamické knihovny. Knihovnu `ldv32.dll` lze tedy snadno připojit implicitně i explicitně[6] do programu psaném v jazyce C/C++[7], já jsem ji ve svém programu připojoval vždy implicitně.

4.3.2.2 Otevření spojení se síťovým rozhraním

Funkce `ldv_open` pro otevření spojení se síťovým rozhraním vyžaduje jako vstupní parametr identifikátor (název) síťového rozhraní. V dokumentaci [14] bohužel není uvedeno, kde lze zjistit hodnoty identifikátorů síťových rozhraní, která jsou na daném počítači instalována a lze se k nim tedy připojit. V příkladu, který je v *OpenLDV Developer's Kit*, jsou možné hodnoty identifikátorů zjišťovány ze systémového registru Windows z klíče `HKEY_LOCAL_MACHINE\\SOFTWARE\\LonWorks\\DeviceDrivers`, já jsem v mé aplikaci použil stejný postup.

4.3.2.3 Konfigurace síťového rozhraní

Každá aplikace potřebuje měnit konfigurační data síťového rozhraní, tj. konfigurační data komunikačního koprocesoru, která jsou popsána v kap. 3.4.1. Například je třeba nastavit SPID[8] tak, aby byl v souladu s SPID uloženým v popisu zařízení v XIF souboru. Toto se provádí zápisem (funkcí `ldv_write`) příslušných konfiguračních zpráv přímo do síťového rozhraní, tedy hodnota v poli *příkaz* v *OpenLDV API zprávě* musí specifikovat, že zpráva je určena rozhraní (není určena k odeslání do sítě), hodnota musí být 2 (viz [14], kap. Application Buffer Structure). Například pro nastavení zmíněného SPID se použije konfigurační zpráva *zápis do paměti*, ve které je třeba specifikovat, že se bude zapisovat do struktury `zakl_par` (viz 3.4.1) a adresa, na kterou se data mají zapsat. Je tedy třeba přesně znát formát konfiguračních struktur, který je popsán v [1] a v [24] (Appendix A).

Síťové rozhraní nepředává všechny přijaté zprávy nadřazené aplikaci (aplikačnímu procesoru), tzn. některé zprávy jsou zpracovány síťovým rozhraním a nelze je vůbec vyčíst pomocí funkce `ldv_read`. Síťové rozhraní předává pouze zprávy obsahující síťové proměnné, aplikační zprávy, cizí rámce a vybrané konfigurační zprávy, viz zprávy, které mají v tab. 3.1 na str. 32 ve sloupci *Apl. proc.* uvedeno A (Ano).

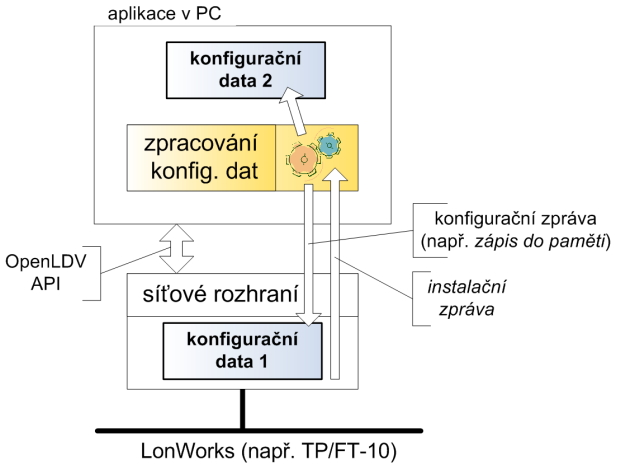
Pokud zařízení nepoužívá *rozšířenou množinu zpráv*, je situace jednoduchá, aplikace žádná konfigurační data (z hlediska protokolu LonTalk) neudržuje, všechna konfigurační

[6] Používám stejné názvosloví jaké je použito v [29].
[7] Knihovnu `ldv32.dll` lze také připojit samozřejmě i v jiných programovacích jazycích.
[8] Identifikátor programu, viz 2.1.5.3 na straně 14.

data jsou v síťovém rozhraní, které také obsluhuje konfigurační zprávy, které s těmito daty manipulují. V této situaci aplikace může používat pouze implicitní adresování.

Používá-li zařízení *rozšířenou množinu zpráv*, je situace složitější. Veškerá manipulace s konfiguračními daty je prováděna prostřednictvím *instalační zprávy* (jak je vysvětleno v 3.7.1), která je síťovým rozhraním přímo předávána ke zpracování aplikaci a proto síťové rozhraní neprovádí žádné operace se svými konfiguračními daty. Ovšem *tabulka adres 1* a *tabulka domén 1* je používána při příjmu zpráv pro rozhodnutí, zda je, či není zpráva určena zařízení, proto i tyto tabulky musí být aktualizovány. A tuto aktualizaci musí zajistit aplikace při obsluze aktualizace dat v *tabulce adres 2*. Situace je znázorněna na obr. 4.4, aplikace přijme požadavek na manipulaci s konfiguračními daty (např. přidání adresy do tabulky adres) prostřednictvím *konfigurační zprávy*, tento požadavek zpracuje (přidá adresu do *tabulky adres 2*) a zároveň vyšle zprávu *aktualizace adresy* do síťového rozhraní a tím zajistí přidání adresy také do *tabulky adres 1*.

Zde se může nyní zdát, že *tabulka adres 1* bude obsahovat kopii *tabulky adres 2*. Ovšem *tabulka adres 1* musí obsahovat jen adresy, které jsou používány pro příjem zpráv (např. aktualizace vstupních SP). Pro odesílání zpráv (např. aktualizace výstupních proměnných) není nutné *tabulku adres 1* použít, lze použít explicitní adresování a odesílat zprávy na adresy uložené pouze v *tabulce adres 2*. Takovýmto způsobem jsem to také řešil ve své aplikaci.



Obrázek 4.4: Konfigurace síťového rozhraní.

4.3.3 Rozdíly mezi rozhraním MIP API a OpenLDV API

Rozhraní MIP API nabízí ovladač (Network Interface Driver) u architektury nazývané host-based, která byla popsána v 2.2.1.2. MIP API je ale také nabízeno pro přístup k síťovým rozhraním jiných výrobců než firmy Echelon např. k rozhraním firem Loytec a Gesytec. Popis, jak lze použít MIP API u rozhraní firmy Loytec, lze nalézt v [7] v kap. Windows MIP/LDV Applications.

Z hlediska nabízených funkcí se MIP API a OpenLDV API liší pouze v tom, že MIP API nenabízí funkci `ldv_register_event`. Použití této funkce ovšem není nutné ani při práci s OpenLDV API, ačkoliv její použití zvětšuje efektivitu programu. Funkce `ldv_register_event` se používá pro zaregistrování čekatelného objektu, který umožňuje realizovat událostně spouštěné vyčítání zpráv ze síťového rozhraní. Bez použití funkce `ldv_register_event` lze periodicky volat funkci `ldv_read` a dle jejího návratového kódu detekovat, zda jsou v síťovém rozhraní nějaké zprávy k vyčtení.

Tímto způsobem jsem vyčítání realizoval já v třídě `COpenLDVapi` v případě, že pracuji s rozhraním MIP API. Třída `COpenLDVapi` může pracovat s MIP API nebo OpenLDV API, ale nemůže pracovat s oběma zároveň, volba je dána podmíněným překladem.

V této práci jsem používal OpenLDV API pro přístup k síťovému rozhraní U10 firmy Echelon a později jsem používal MIP API pro přístup k síťovému rozhraní L-IP[9] firmy Loytec. Při práci s těmito rozhraními jsem narazil na rozdílnost v jejich chování.

Pokud byl na síťové rozhraní U10 vyslán konfiguračním nástrojem LonMaker požadavek na přechod zařízení do stavu *zkonfigurované pozastavené* a síť. rozhraní do tohoto stavu přešlo, tak i nadále síť. rozhraní předávalo zprávy ke zpracování do aplikace. Pokud byl takovýto požadavek zaslán na síť. rozhraní L-IP a byl akceptován, rozhraní přestalo přeposílat zprávy do aplikace (nebylo možné je vyčíst funkcí `ldv_read`). Popis tohoto chování jsem v dokumentaci [7], [12] ani nikde jinde nenalezl.

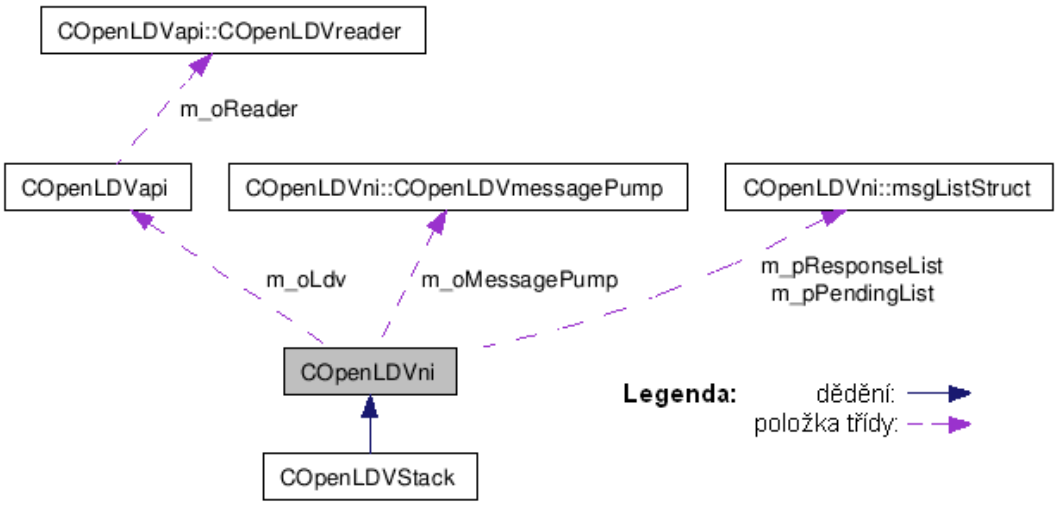
Toto chování se projevilo při instalaci mé aplikace OpenLDVStackDemo do sítě, když byla do sítě LonWorks připojena přes L-IP router. Během instalace zařízení do sítě (v mém případě tedy aplikace OpenLDVStackDemo) je konfiguračním nástrojem zasílán požadavek na přechod zařízení do stavu *zkonfigurované pozastavené* (viz popis algoritmu v kap. 3.8.1), poté co byl tento požadavek síťovým rozhraním zpracován, aplikace OpenLDVStackDemo již nemohla přijímat další konfigurační zprávy, tudíž na ně nemohla odpovídat a nebylo tak možné dokončit proces instalace aplikace do sítě.

[9] L-IP je router, který slouží k tunelování protokolu LonTalk skrz síť ethernet s využitím protokolu EIA-852.

Tento problém jsem vyřešil tak, že třída `COpenLDVapi` automaticky periodicky zasílá síťovému rozhraní požadavek na přepnutí do stavu *zkonfigurované* (popis možných stavů viz 3.8.1.1.). Požadavek je zasílán každé 2 s, tento poměrně dlouhý čas byl zvolen proto, že takto vynucená změna stavu síťového rozhraní se uplatní pouze při instalaci aplikace do sítě, kde je zdržení způsobené čekáním na přepnutí stavu akceptovatelné. Při běžném běhu aplikace se tento mechanismus nemusí uplatňovat, proto by bylo zbytečné zatěžovat komunikaci častým zasíláním požadavku na přepnutí stavu. Implementace tohoto mechanismu je provedena v metodě `LdvReaderThread` třídy `COpenLDVreader` a je opět vázána na podmíněný překlad.

4.4 Zpracování zpráv

Všechny zprávy vyčtené funkcí `ldv_read` ze síťového rozhraní jsou uloženy do fronty zpráv, která je členem[10] třídy `COpenLDVapi`, a je vyvolána událost, kterou je spouštěno zpracování zpráv ve vlákně zapouzdřeném do třídy `COpenLDVmessagePump`. Instance třídy `COpenLDVmessagePump` je členem třídy `COpenLDVStack`, která je potomkem třídy `COpenLDVni`. Vztahy mezi těmito třídami jsou také znázorněny na obr. 4.5, který byl ze zdrojových kódů vygenerován programem Doxygen[11].



Obrázek 4.5: Vztahy mezi třídami.

Třída `COpenLDVStack` je hlavní třída zajišťující zpracování aplikační vrstvy protokolu

[10] Položka `m_qUplink` typu `TMessageQueue`.
[11] Program lze zdarma získat na webové stránce http://www.doxygen.org.

LonTalk, spolu s třídami pro interpretaci SP (viz 4.5) je jádrem *Rozhraní SP*, její hlavní funkce jsou:

- zapouzdřuje všechna data potřebná pro aplikační vrstvu protokolu LonTalk, podrobněji o těchto datech viz následující odstavec 4.4.1,
- zpracovává přijaté aktualizace hodnot síťových proměnných, viz odstavec 4.4.2,
- zpracovává požadavky na manipulaci s konfiguračními daty, viz odstavec 4.4.3,
- zajišťuje ukládání konfiguračních dat do souborů na pevný disk počítače.

4.4.1 Data třídy `COpenLDVStack`

Položky třídy `COpenLDVStack`, které slouží pro uchování *konfiguračních dat* a *síťových proměnných* jsou uvedeny ve výpisu kódu 4.1, který obsahuje část deklarace třídy `COpenLDVStack`.

```
class COpenLDVStack : public COpenLDVni
{
  private:
      ⋮
    CVariableVector     Variables;
    Tsnvt_struct_v2     m_SnvtStruct;
    CUnprocMsgQueue     UnprocMsgQueue;
    CAddressTable*      pAddresses;
    CDomainTable*       pDomain;

    HANDLE              hEvent_unproc;
      ⋮
}
```

Výpis 4.1: Položky třídy `COpenLDVStack` pro uchování dat protokolu LonTalk.

Jednotlivé položky mají následující význam:

- Položka `Variables` je vektor obsahující síťové proměnné. Typ `CVariableVector` je deklarován jako:

  ```
  typedef vector<CVariable4Stack*>   CVariableVector;
  ```

Třída `CVariable4Stack` představuje jednu SP, obsahuje její data, její konfiguraci a umí její data reprezentovat dle jejího *standardního typu*, tato třída je podrobně popsána v odstavci 4.5 dále.

- Položka `m_SnvtStruct` obsahuje popis profilu zařízení, deklarace typu `Tsnvt_struct_v2` je uvedena ve výpisu kódu 3.1 na str. 38.

- Položka `UnprocMsgQueue` je fronta obsahující zprávy, které nebyly zpracovány ve třídě `COpenLDVStack`, tj. zprávy, které nebyly aktualizace hodnot SP ani konfigurační zprávy[12]. Nezpracované zprávy jsou ukládány do fronty `UnprocMsgQueue` a je vyhlášena událost, jejíž identifikátor (handle) je uložen v proměnné `hEvent_unproc`. Na tuto událost reaguje jiná část aplikace (třída `CUpdatesProcessing`, viz obr. 4.2), která může zprávu z fronty vyzvednout a zpracovat. Takovýto postup je v aplikaci OpenLDVStackDemo používán pro zpracování dat přenášených metodou LW-FTP (viz odstavec 4.6).

  Typ `CUnprocMsgQueue` je deklarován jako:

  ```
  typedef queue<ExpAppBuffer>   CUnprocMsgQueue;
  ```

  přičemž typ `ExpAppBuffer` je struktura pro *zprávu OpenLDV API*, která bylo popsána v 4.3.1.

- Položka `pAddresses` je ukazatelem na objekt typu `CAddressTable`, který je vytvářen dynamicky při inicializaci třídy `COpenLDVStack`, tj. při volání metody `Initialize`. Třída `CAddressTable` slouží jako *tabulka adres 2* a nabízí metody pro přidání, aktualizaci a odebrání adresy a metody pro uložení obsahu do souboru a načtení ze souboru. Deklarace třídy `CAddressTable` je uvedena ve výpisu kódu D.1 v příloze D.

- Ukazatel `pDomain` na třídu `CDomainTable` je odkazem na dynamicky vytvářený objekt během inicializace (v obsluze metody `Initialize`). Třída `CDomainTable` slouží k uchování *tabulky domén 2* a také nabízí metody pro přidání, aktualizaci a odebrání domény a metody pro uložení a načtení obsahu do (z) souboru, deklarace třídy `CDomainTable` je uvedena ve výpisu kódu D.2 v příloze D.

4.4.2 Zpracování vstupních síťových proměnných

Z vlákna pro zpracování příchozích zpráv (z vlákna v třídě `COpenLDVmessagePump`) je pro každou přijatou zprávu volána metoda `Dispatch`, která provádí obsluhu dle typu

[12] Typy zpráv jsou popsány v kap. 3.2 Aplikační rámec.

zprávy. Jedná-li se o zprávu obsahující aktualizaci vstupní proměnné, což je detekováno dle prvních 2 až 4 bitů aplikačního rámce (viz kap. 3.2), je volána funkce `HandleNvUpdate`, která zajistí další zpracování zprávy.

Funkce `HandleNvUpdate` nejdříve zjistí dle selektoru v přijaté zprávě index odpovídající SP v *tabulce SP* (ve vektoru `CVariableVector`). Pokud žádná SP nemá odpovídající selektor, je zpracování ukončeno. Aktualizace hodnoty dané SP je provedena voláním metody `SetValue` objektu SP (objekt typu `CVariable4Stack`). Metoda `SetValue` také vyvolá událost, která může být SP přiřazena, např. aplikace OpenLDVStackDemo na základě této události aktualizuje hodnotu SP v grafickém uživatelském rozhraní aplikace.

4.4.3 Zpracování konfiguračních zpráv

Pokud je metodou `Dispatch` třídy `COpenLDVStack` detekováno dle prvních 2 až 4 bitů aplikačního rámce, že příchozí zpráva je *konfigurační zprávou*, je volána specifická metoda pro obsluhu dle *kódu* v hlavičce aplikačního rámce.

Příslušná metoda pak provádí požadovanou operaci s *konfiguračními daty*, tzn. pracuje s datovými položkami (či objekty), které byly popsány v předchozí kap. 4.4.1. Přehled metod a zpráv, které metody zpracovávají je v tab. 4.2. Protože většina konfiguračních zpráv je zasílána službou *dotaz/odpověď*, je po provedení požadované operace také případně vytvořena odpověď a odeslána zpět do síťového rozhraní, odpověď je odeslána na adresu, ze které přišel požadavek.

Název metody	Zpráva zpracovávaná metodou	Kód zprávy (hexadecimálně)
`HandleSetNodeMode`	Nastavení stavu zařízení	6C
`HandleWink`	Instalační zpráva	70
`HandleQuerySI`	Čtení profilu zařízení	72
`HandleNvFetch`	Čtení hodnoty SP	73

Tabulka 4.2: Metody třídy `COpenLDVStack` zpracovávající konfigurační zprávy.

4.5 Interpretace síťových proměnných

Ke každé síťové proměnné se vztahuje její typ, který jednoznačně určuje interpretaci dat SP, jak bylo již vysvětleno v kap. 2.1.3.1. Aplikace musí rozumět struktuře dat,

aby s hodnotou SP mohla pracovat, tj. aby hodnotu SP mohla plnit (a pak odesílat do sítě LonWorks) a nebo interpretovat hodnotu SP přijatou ze sítě. Pokud má aplikace množinu svých SP (vstupních i výstupních) pevně danou a neměnnou, je možné pro interpretaci struktury dat deklarovat datové typy přímo v programovacím jazyce pro každý typ proměnné a pomocí těchto datových typů k jednotlivým částem hodnoty SP přistupovat. Tento přístup je použit u zařízení založených na samotném procesoru Neuron Chip (tzv. hosted nodes) a u zařízení založených na architektuře Short Stack Development Kit.

Pro zařízení, která využívají konceptu dynamických proměnných, je ale přístup založený na deklaracích datových typů pro jednotlivé typy SP nevhodný, protože v době tvorby softwaru ještě není známo, jakých typů proměnné budou. Pro tento typ zařízení je lepší strukturu dat SP dynamicky načítat z externího zdroje, např. ze souborů uložených na pevném disku počítače. A pro tyto účely sdružení LonMark zveřejňuje definice *standardních typů* v podobě binárních souborů, ke kterým také zveřejňuje API pro práci s těmito soubory, tzv. *LonMark Resource File API*.

4.5.1 LonMark Resource File API

LonMark Resource File API (dále jen zkráceně *LonMark API*) je k dispozici volně ke stažení na webových stránkách sdružení LonMark (http://www.lonmark.com) v rámci balíčku *Resource Files*. Tento balíček obsahuje:

- binární soubor `standard.typ` s popisem typů SP,
- binární soubor `standard.enu`, který obsahuje anglické texty k popisům typů SP,
- knihovnu `lcadrf32.dll` pro čtení a zápis do binárních souborů s popisem standardních typů,
- knihovnu `ldrf32r.dll`, která umožňuje pouze čtení z binárních souborů s popisem standardních typů,
- hlavičkové soubory `lcadrf.h` ke knihovně `ldrf32r.dll` a `lcadrfw.h` ke knihovně `lcadrf32.dll` a soubory `lcadrf32.lib` a `ldrf32r.lib`,
- zdrojový kód v jazyce ANSI C s implementací čtení z binárních souborů s popisem standardních typů,
- dokumentaci [18].

Samotné LonMark API je popsané pouze v dokumentaci [18], což je ale pouze seznam funkcí (je jich desítky) a popis jejich parametrů. Jejich použití (např. pořadí volání) zde nijak zdokumentováno není, takže jejich použití vyžadovalo částečně experimentální přístup. Parametry funkcí jsou často typů, které nejsou deklarovány v hlavičkových souborech `lcadrf.h` resp. `lcadrfw.h`, potřebné deklarace lze ale nalézt v hlavičkovém souboru `lcadrfi.h`, který je součástí zdrojového kódu pro čtení binárních souborů s popisem standardních typů.

4.5.2 Části síťové proměnné

Struktura dat SP může být pestrá, většinou je tvořena několika *částmi* (dále budou označovány jako *části SP*), příklad jednoduché struktury dat skládající se ze dvou *částí SP* je také uveden v kap. 2.1.3.1. V tab. 4.3 jsou uvedeny typy, jakých můžou *části SP* být a také je zde vždy název třídy, pomocí jaké jsem daný typ implementoval v Rozhraní SP.

Typ části	Třída zapouzdřující typ	Poznámka
celé číslo	CBaseType	Lze použít znaménkové i neznaménkové číslo o velikost 1, 2 a 4 byty.
reálné číslo	CFloatType	Reprezentace dle ANSI/IEEE 754, velikost 4 byty.
výčet	CEnumType	Velikost 1 byte se znaménkem.
bitové pole	CBitFieldType	Velikost dána počtem bitů, viz parametr *velikost* v tab. 4.4.

Tabulka 4.3: Přehled typů částí síťové proměnné.

Ke každému typu *části SP* se váží další parametry, které zpřesňují interpretaci této *části SP*, tyto parametry mají pevnou strukturu (jejich počet a význam) pro každý typ *části SP*. Konkrétní hodnoty těchto parametrů jsou specifické vždy pro daný *standardní typ* síťové proměnné (SNVT[13]) a lze je načíst z binárních souborů s popisem standardních typů. V tab. 4.4 je uveden přehled parametrů jednotlivých typů *částí SP*.

[13] SNVT - Standard Network Variable Type, viz také kap. 2.1.3.1.

Typ části	Parametr	Význam parametrů
celé číslo	minimum	Minimální platná hodnota před oškálováním.
	maximum	Maximální platná hodnota před oškálováním.
	škálovácí konstanta A	Hodnota po oškálování je pak rovna $A \cdot 10^B \cdot ((\text{hodnota části SP}) + C)$.
	škálovácí konstanta B	
	škálovácí konstanta C	
reálné číslo	minimum	Minimální platná hodnota.
	maximum	Maximální platná hodnota.
výčet	minimum	Minimální platná hodnota.
	maximum	Maximální platná hodnota.
	pole textů	Texty popisující jednotlivé položky výčtu.
bitové pole	velikost	Počet bitů, které zabírá.
	ofset	Ofset, od kterého hodnota v rámci bytu začíná.
	typ čísla	Příznak, zda je číslo znaménkové nebo neznaménkové.
	minimum	Minimální platná hodnota před oškálováním.
	maximum	Maximální platná hodnota před oškálováním.
	škálovácí konstanta A	Hodnota po oškálování je pak rovna $A \cdot 10^B \cdot ((\text{hodnota části SP}) + C)$.
	škálovácí konstanta B	
	škálovácí konstanta C	

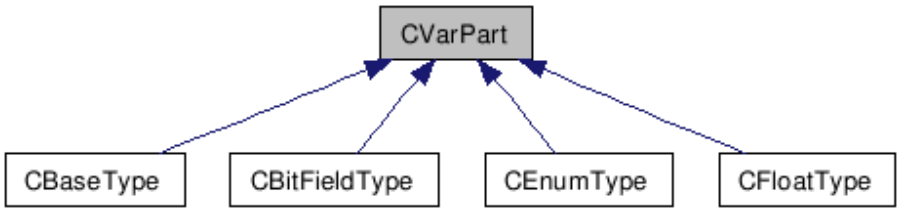
Tabulka 4.4: Parametry typů částí SP.

4.5.3 Implementace interpretace síťových proměnných

Při implementaci Rozhraní SP jsem pro práci se síťovou proměnnou vytvořil třídu `CVariable`, jejímž hlavním úkolem je zapouzdřit samotná data SP a také umět tato data interpretovat dle standardního typu SP.

Pro jednotlivé typy *částí SP* jsem implementoval třídy uvedené v tab. 4.3. Úkolem těchto tříd je umět interpretovat aktuální hodnotu *části SP* dle parametrů příslušejících danému typu *části SP* a poskytovat metody pro jednoduché načtení a aktualizaci hodnot *části SP* a metody pro získání textové reprezentace hodnoty *části SP* a pro získání jména *části SP*. Jelikož mají tyto třídy téměř stejné rozhraní, jsou zděděny (viz obr. 4.6) od společného předka, od třídy `CVarPart`. Třída `CVarPart` také deklaruje virtuální metody, jejichž implementace je závislá na typu *části SP*, tedy implementace je specifická pro každou odvozenou třídu. Takto implementované metody umožňují využít polymorfismu

(založeném na pozdní vazbě - viz [27]), což je v aplikaci OpenLDVStackDemo využíváno např. při voláni metody `ToString` pro získání textové reprezentace aktuální hodnoty SP.



Obrázek 4.6: Třídy zděděné od třídy `CVarPart`.

Při inicializaci objektu typu `CVariable` voláním metody `Initialize`, jejímž parametrem je také SNVT Index, je pomocí funkcí z LonMark Resource File API načtena struktura a popis síťové proměnné. Při načítání struktury je pro každou *část SP* vytvořen objekt odpovídající třídy (viz tab. 4.3) a odkazy na tyto objekty jsou uloženy ve třídě `CVariable` do privátní položky `VarPartVector`.

Pomocí metody `GetVarPart(Word index)` třídy `CVariable`, jejíž parametr index je pořadové číslo *části SP* (tj. index prvku ve vektoru `VarPartVector`), lze získat odkaz na příslušný objekt reprezentující *část SP*. Na tomto objektu lze již snadno volat metody:

- `GetName` pro zjištění názvu *části SP*,
- `GetValue` pro získání aktuální hodnoty *části SP*, typ návratové hodnoty této metody záleží na typu *části SP* (tedy na objektu jaké třídy je metoda volána),
- `SetValueToBuff` pro nastavení nové hodnoty *části SP* v bufferu hodnoty SP, odkud je možné po provedení změn ve všech *částech SP* přenést hodnotu z bufferu do skutečné hodnoty SP (toto se provede voláním metody `UpdateFromBuffer` třídy `CVariable4Stack`, čímž se provede odeslání do sítě LonWorks),
- `ToString` pro získání textové reprezentace dané *části SP*,
- `GetType` pro zjištění typu *části SP*.

Od třídy `CVariable` je ještě zděděna třída `CVariable4Stack`, která byla již zmíněna výše, tato třída navíc zapouzdřuje konfigurační data síťové proměnné (selektor, typ služby jakou je SP zasílána atd.).

4.6 Přenos dat metodou LW-FTP

Konfigurační parametry mohou být implementovány jako *konfigurační síťové proměnné* nebo jako *konfigurační soubory*[14]. Konfigurační síťové proměnné jsou v podstatě klasické síťové proměnné, jen jejich hodnoty obsahují nastavení (parametry) aplikace (řídícího algoritmu), proto se také jejich hodnoty mění jen zřídka. Hlavní nevýhoda je, že *konfigurační síťové proměnné* mohou mít maximální velikost stejnou jako klasické síťové proměnné, tj. maximálně 31 bytů.

Konfigurační soubory jsou větší bloky dat, které obsahují více konfiguračních parametrů najednou. Výhoda *konfiguračních souborů* je, že jejich velikost je prakticky omezena pouze zdroji na konkrétním zařízení a použitou metodou přenosu *konfiguračních souborů*. Data *konfiguračních souborů* lze přenášet pomocí *konfiguračních zpráv* pro čtení a zápis do paměti (zprávy s kódy 6Dh, 6Eh) nebo lze použít metodu LW-FTP (LonWorks File Transfer Protocol). V případě použití *konfiguračních zpráv* je maximální velikost *konfiguračního souboru* 65 kB, v případě použití metody LW-FTP je teoretická velikost *konfiguračního souboru* až 4 GB.

Je třeba vzít v úvahu také, že je-li aplikace založena na OpenLDV, tak aplikačnímu procesoru nejsou předávány *konfigurační zprávy* pro čtení a zápis do paměti, viz tab. 3.1, kde je označeno, které zprávy nejsou aplikačnímu procesoru předávány. Protože nejsou tyto zprávy předávány do aplikačního procesoru, lze u zařízení založených na OpenLDV použít pro přenos *konfiguračního souboru* pouze metodu LW-FTP. Toto je také důvod, proč jsem v Rozhraní SP implementoval metodu LW-FTP.

LW-FTP je metoda pro přenos většího množství dat pomocí protokolu LonTalk a lze ji použít obecně pro jakákoliv data, neslouží jen pro přenos *konfiguračních souborů*. Metoda LW-FTP je specifikována v [11]. LW-FTP je založeno na použití třech řídicích síťových proměnných[15], které slouží k řízení přenosu dat. Data samotná jsou přenášena pomocí aplikačních zpráv s hlavičkou 3Eh (typy hlaviček viz kap. 3.2). Podprogram zajišťující obsluhu LW-FTP pracuje na základě hodnot této trojice řídicích SP.

V Rozhraní SP jsem obsluhu LW-FTP implementoval ve třídě `CLONFileTransfer`, která reaguje na změny řídicích proměnných a zpracovává příchozí aplikační zprávy s daty. Obsluha změny některé řídicí proměnné je prováděna metodou `Process`, která je volána z vlákna `CUpdatesProcessing` na základě události informující o změně hodnoty řídicí SP, toto je naznačeno také na obr. 4.2. Obsluha příchozích dat v aplikační zprávě je prováděna

[14] Podrobné informace týkající se způsobů implementace konfiguračních parametrů lze nalézt v [19].
[15] Přesnou specifikaci těchto SP lze nalézt v [11], [10].

metodou `ProcessMsg`, která je volána z vlákna `CUpdatesProcessing` na základě události informující o přítomnosti nezpracovaných zpráv ve frontě[16].

Ve třídě `CLONFileTransfer` je implementováno zpracování příjmu souboru, není zde implementováno odesílání souboru. V Rozhraní SP je třída `CLONFileTransfer` použita pro příjem konfiguračního souboru z konfiguračního nástroje.

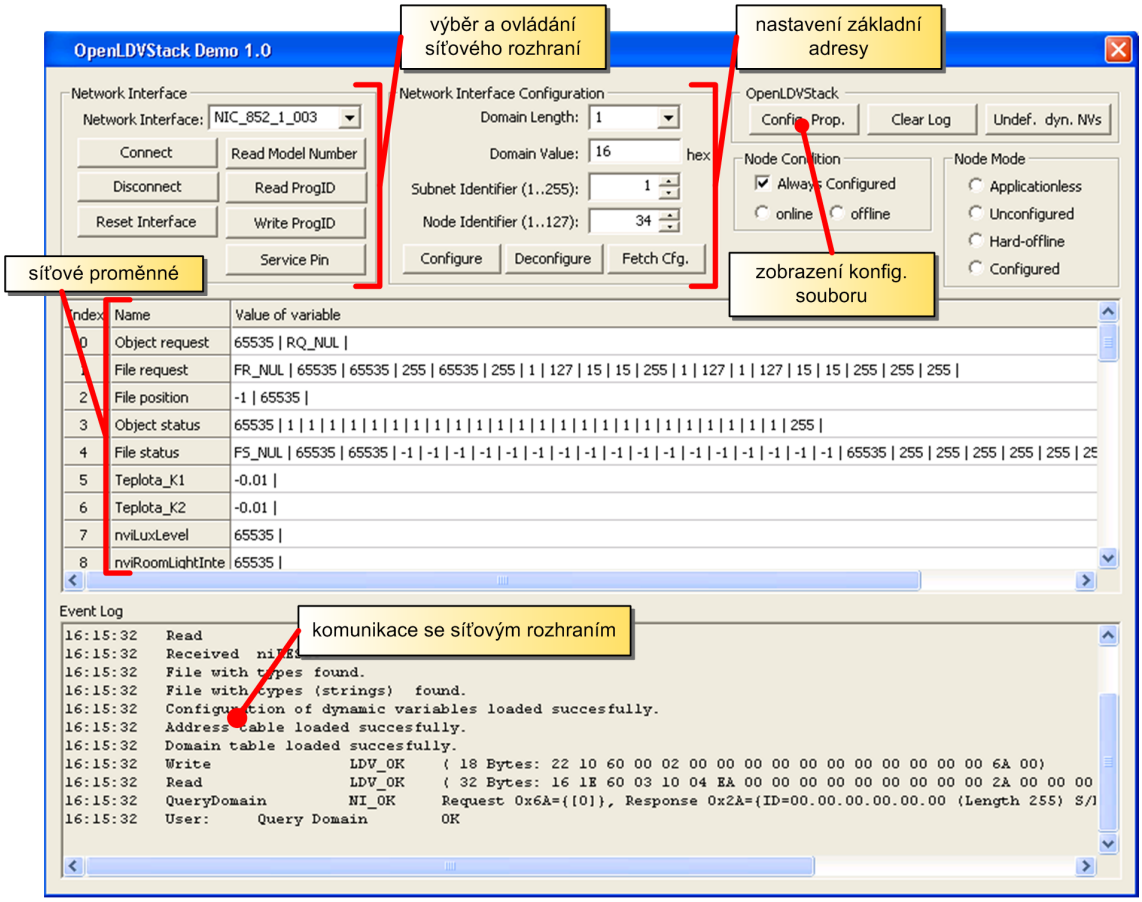
4.7 Testovací aplikace OpenLDVStackDemo

Aplikace OpenLDVStackDemo s jednoduchým uživatelským rozhraním vznikla pro účely vývoje a testování Rozhraní SP, je implementována v jazyce C++ s využitím knihovny MFC[17]. Aplikace umožňuje ovládat síťové rozhraní (nastavovat jeho *základní adresu*, nastavovat SPID), zobrazuje síťové proměnné a umožňuje je editovat.

Na obr. 4.7 je hlavní okno aplikace, na kterém se nachází:

- tlačítka pro výběr síťového rozhraní a připojení k němu,
- editační pole a tlačítka pro nastavení *základní adresy*,
- tlačítko pro zobrazení okna s obsahem *konfiguračního souboru*,
- přepínače pro nastavení stavu zařízení,
- přepínač *Always configured* zapínající/vypínající automatické periodické přepínání síťového rozhraní do stavu *zkonfigurované* (více o tomto mechanismu viz kap. 4.3.3),
- tabulka s přehledem síťových proměnných a jejich aktuální hodnotou,
- pole se záznamem komunikace se síťovým rozhraním a dalších událostí (např. změny hodnot SP).

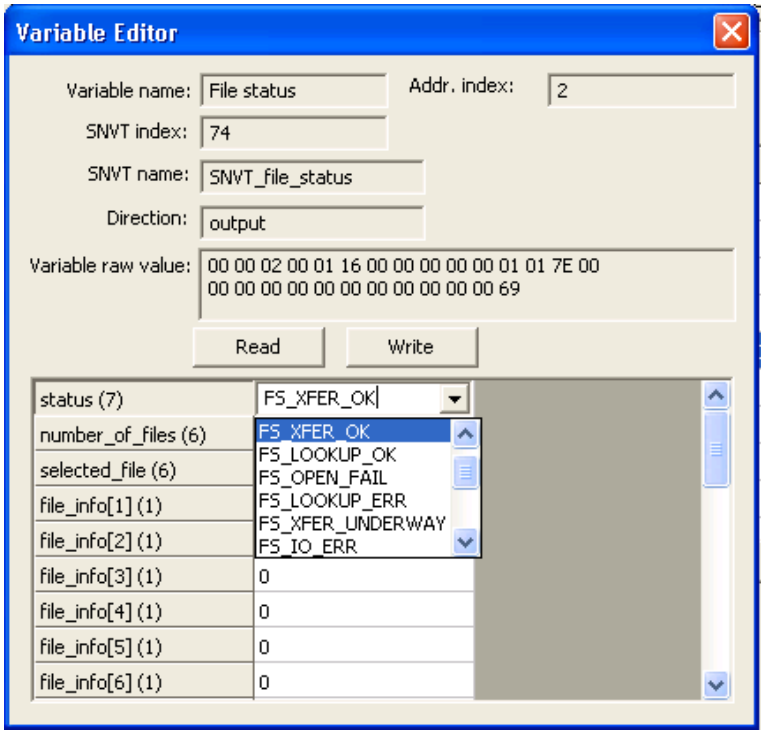
[16] To jsou zprávy, které nezpracovala třída `COpenLDVStack`.
[17] Microsoft Foundation Classes je sada tříd, která zapouzdřuje Windows API.



Obrázek 4.7: Aplikace OpenLDVStackDemo - hlavní okno.

Poklepáním na název proměnné v tabulce síťových proměnných lze otevřít okno pro prohlížení a editaci hodnoty SP. Ukázka tohoto okna je na obr. 4.8, okno obsahuje informace o síťové proměnné (její název, SNVT typ atd.) a ve spodní části okna je vidět struktura SP, jsou zde jednotlivé *části SP*. Lze editovat každou *část SP* pokud jde o výstupní SP. Na obr. 4.8 je zachycena editace *části SP* `status`, která je typu výčet[18], jsou zde vidět názvy jednotlivých položek výčtu. Informace o struktuře SP jsou poskytovány třídou `CVariable`, o které je více uvedeno v kap. 4.5.3.

[18] Přehled typů *částí SP*, viz kap. 4.5.2.



Obrázek 4.8: Aplikace OpenLDVStackDemo - editace síťové proměnné.

4.8 Vytvoření XIF souboru

XIF soubor obsahuje popis profilu zařízení, jak již bylo uvedeno v kap. 2.1.5.4. XIF soubor je potřebný především pro konfigurační nástroj pro správu sítě, který si z něj načte všechny potřebné informace o zařízení.

I pro aplikaci vytvářenou v rámci této práce bylo nutné XIF soubor vytvořit. XIF soubor má textovou podobu a jeho struktura je dobře zdokumentována v [21]. Lze jej tedy vytvořit ručně, bylo by to ale časově náročné. Já jsem při vytváření XIF souboru použil aplikaci NodeBuilder, která je na Katedře řídicí techniky ČVUT-FEL k dispozici. Pomocí průvodce v aplikaci NodeBuilder jsem vytvořil profil zařízení, který obsahoval pouze speciální funkční profil *Node Object* (o tomto profilu viz kap. 2.1.5.2) a SP potřebné pro realizaci přenosu dat metodou LW-FTP. Takto vytvořený profil jsem již dále upravil ručně v textovém editoru. Výsledný XIF soubor je v příloze E. Tento XIF soubor jsem použil pro aplikaci OpenLDVStackDemo i pro OPC server (nazvaný LonOPC, viz kap. 5).

V XIF souboru je také na 5. řádku (viz příloha E) uloženo SPID[19], jehož část ob-

[19] Identifikátor programu, viz 2.1.5.3 na straně 14.

sahuje identifikátor výrobce[20], jehož hodnotu výrobcům přiděluje sdružení LonMark, přesněji jej přiděluje členům sdružení LonMark. Lze ovšem získat dočasný identifikátor, který je přidělen zdarma po vyplnění formuláře na webových stránkách sdružení LonMark (http://www.lonmark.com). Takto jsem i já získal identifikátor výrobce, který jsem použil v SPID vyvíjené aplikace.

Při ruční editaci XIF souboru lze použít program `xif32bin.exe` pro kontrolu syntaktické správnosti připravovaného XIF souboru. Program `xif32bin.exe` původně slouží pro konverzi souboru `*.xif` do binární podoby (`*.xfb`), kterou používají konfigurační nástroje (např. LonMaker), se kterými je program `xif32bin.exe` distribuován.

[20] Přesnou strukturu SPID lze nalézt v [19] v kap. 2.3.

Kapitola 5

OPC server

V této kapitole je popsán OPC[1] server nazvaný LonOPC, který slouží pro monitorování a ovládání zařízení zapojených v síti LonWorks. LonOPC byl vytvořen s využitím Rozhraní SP, které bylo popsáno v předchozí kapitole a s použitím nástroje OPC ToolWorX od firmy ICONICS, který slouží pro vytvoření OPC rozhraní. Vlastnosti aplikace LonOPC jsou dány použitím zmíněných komponent. LonOPC přistupuje do sítě pomocí síťového rozhraní, které poskytuje MIP API[2], nepotřebuje tedy pro svou činnost LNS Server jako většina ostatních OPC serverů pracujících se zařízeními v síti LonWorks (např. IPLONGATE od firmy who Ingenieurgesellschaft mbH, NLOPC firmy Loytec atd.).

Množinu síťových proměnných, se kterými LonOPC pracuje je možné měnit pomocí konfiguračního nástroje pro správu sítě LonWorks, využívá se zde konceptu dynamických proměnných. Podrobnější popis instalace serveru LonOPC, jeho konfigurace do sítě LonWorks a konfigurace množiny síťových proměnných je uvedena v příloze F.

LonOPC implementuje standard *Data Access Custom Interface Standard* organizace *OPC Foundation*. V dalších odstavcích je popsána implementace aplikace LonOPC a také jsou zde uvedeny výsledky používání aplikace LonOPC v modelu domovní automatizace na Katedře řídicí techniky ČVUT-FEL.

5.1 OPC ToolWorX

Mým hlavním úkolem nebylo vytvořit vlastní implementaci OPC serveru a proto jsem pro vytvoření OPC serveru LonOPC použil nástroj OPC ToolWorX od firmy ICONICS,

[1] OPC - OLE for Process Control je otevřený standard pro meziprocesní komunikaci v operačním systému Windows, který je založen na technologii COM.

[2] O tomto API viz podrobněji v kap. 4.3.3 a 2.2.1.2.

který je na katedře řídicí techniky k dispozici.

OPC ToolWorX je sada objektů (naprogramovaných s využitím ATL knihovny[3]), které usnadňují vytvoření OPC serveru. Součástí nástroje je také průvodce (wizard) pro prostředí Microsoft Visual Studio .NET 2003, který slouží k vytvoření kostry celého serveru na základě požadavků na vlastnosti serveru, např. zda má být server vytvořen jako klasický program nebo zda to bude služba operačního systému Windows. Podrobnosti o nástroji OPC ToolWorX lze nalézt v [16]. Po vytvoření kostry serveru pomocí průvodce je třeba implementovat:

- vytváření struktury dat, které OPC server poskytuje, tj. vytvoření OPC datových položek (Items) a OPC skupin (Groups), do kterých jsou OPC položky zařazeny,
- propojení OPC položek se skutečnými daty, tj. v případě aplikace LonOPC se jedná o propojení položek s částmi síťových proměnných (OPC skupiny odpovídají síťovým proměnným).

5.2 Architektura aplikace LonOPC

Na obr. 5.1 je znázorněna vnitřní architektura aplikace LonOPC a vazby na další související software. Kostru aplikace jsem vytvořil pomocí průvodce nástroje OPC ToolWorX a postupně jsem do projektu přidával další potřebné moduly. Vytvořením základu aplikace pomocí průvodce z OPC ToolWorX byla také získána možnost spouštět aplikaci jako standardní program nebo jako službu operačního systému Windows. Na druhé straně nevýhodou bylo, že do takto vytvořené aplikace nebylo možné přidat grafické uživatelské rozhraní (aplikace je totiž interně založena na objektu `CATLExeModuleT`, další podrobnosti lze nalézt v [23]).

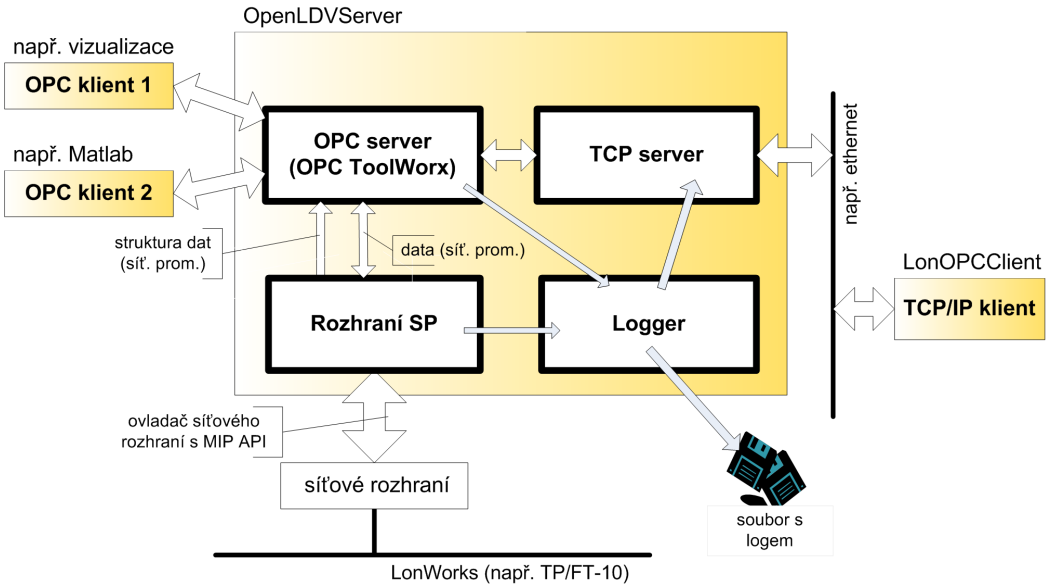
Uvnitř je aplikace LonOPC členěna na čtyři hlavní moduly:

- Modul *OPC server* je realizován třídami, které poskytuje OPC ToolWorX a zajišťuje obsluhu požadavků OPC klientů.
- Modul *Rozhraní SP* zajišťuje obsluhu protokolu LonTalk, komunikuje se síťovým rozhraním. Pro modul *OPC server* poskytuje možnost vyčíst strukturu dat, tj. možnost zjistit jaké *síťové proměnné* jsou aktuálně používány, jaké mají *části SP* a jaké mají *aktuální hodnoty*.

[3] ATL - Active Template Library je knihovna pro podporu programování technologie COM, viz [23].

- Úkolem modulu *TCP server* je nabízet komunikační rozhraní založené na TCP[4] protokolu, pomocí kterého je možné ovládat síťové rozhraní (např. simulovat stisk servisního tlačítka), vyčítat aktuální seznam SP a jejich hodnoty, monitorovat běh celé aplikace LonOPC. Tento modul byl implementován zejména jako meziprocesní komunikace pro aplikaci LonOPCClient, která je také zachycena na obr. 5.1 a která slouží pro monitorování a ovládání aplikace LonOPC. Více o aplikaci LonOPCClient je v kap. 5.3, kde je také popsán protokol pro komunikaci s tímto TCP serverem.

- Modul *Logger* slouží pro zaznamenávání informací o běhu celé aplikace. *Logger* informace ukládá do souboru na pevný a disk a pokud je připojen nějaký klient k TCP serveru, pak zasílá informace tomuto klientovi (přesněji všem připojeným klientům).



Obrázek 5.1: Architektura OPC serveru LonOPC.

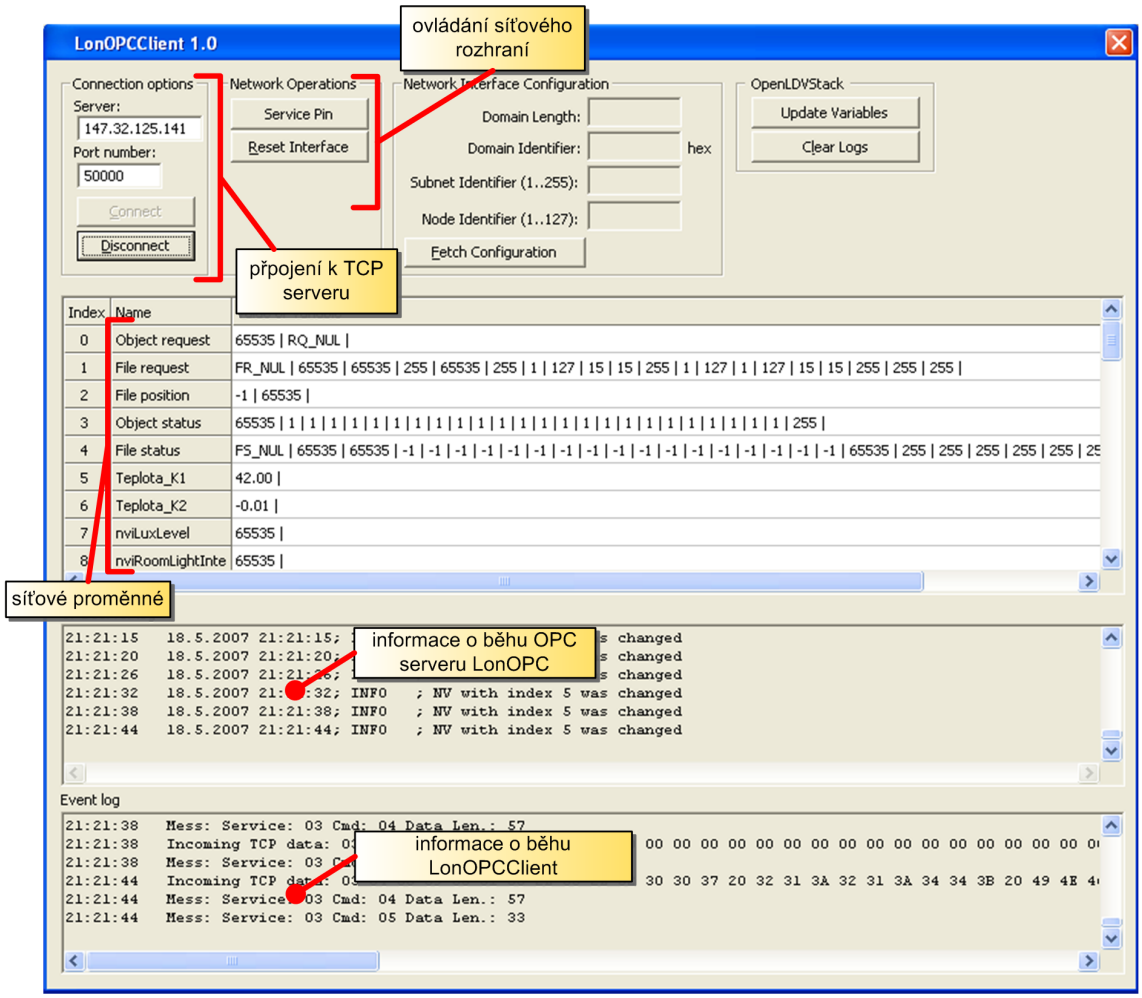
5.3 Aplikace LonOPCClient

Tato aplikace slouží pro monitorování běhu serveru LonOPC a k ovládání síťového rozhraní, které LonOPC aktuálně používá. Aplikace není určena k měnění hodnot SP v serveru LonOPC. Tato aplikace byla vyvinuta zejména proto, že do aplikace LonOPC

[4] TCP - Transmission Control Protocol.

nelze přidat uživatelské rozhraní, které je vhodné zejména při instalaci (začlenění) aplikace LonOPC do sítě LonWorks (popis instalace viz příloha F).

Uživatelské rozhraní (viz obr. 5.2) je podobné jako u aplikace OpenLDVStackDemo. Místo ovládacích prvků pro připojení k síťovému rozhraní tu jsou prvky sloužící pro zadání IP adresy a portu TCP serveru, ke kterému se budeme připojovat, tj. IP adresu počítače, na kterém je spuštěna aplikace LonOPC a port 50000.



Obrázek 5.2: Aplikace LonOPCClient - hlavní okno.

Nejdůležitější třídy, které zajišťují činnost aplikace LonOPCClient jsou `CTCPClient` a `CVirtualOpenLDVStack`. Třída `CVirtualOpenLDVStack` je obdobou třídy `COpenLDVStack` v Rozhraní SP, také zapouzdřuje síťové proměnné a konfigurační data. Navíc oproti třídě `COpenLDVStack` nabízí metody pro získání dat přes TCP klienta (ten je reprezentován třídou `CTCPClient`). Například kromě metody `GetVariableCount`, která vrací počet SP, nabízí třída `CVirtualOpenLDVStack` také metodu `GetVariableCountSrv`, která vyčte

počet aktuálně nakonfigurovaných SP v serveru LonOPC. Třída `CVirtualOpenLDVStack` narozdíl od třídy `COpenLDVStack` neobsahuje zpracování protokolu LonTalk. Komunikaci prostřednictvím TCP protokolu zajišťuje třída `CTCPClient`, která je odvozena od třídy `CSocketComm`, kterou jsem převzal z [9]. Protokol používaný mezi serverem LonOPC a klientem `CTCPClient` je popsán v následující kapitole.

5.3.1 Protokol TCP serveru

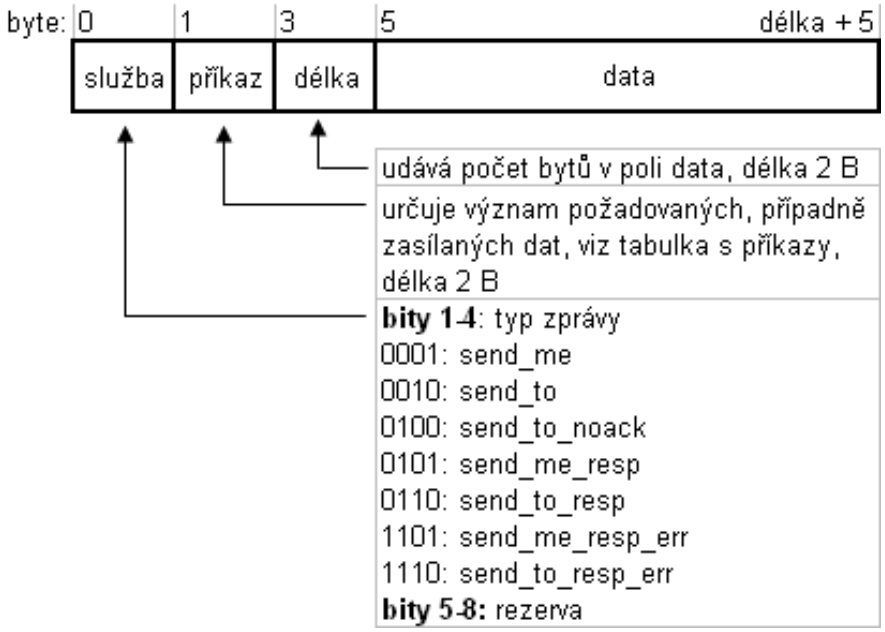
Pro komunikaci pomocí protokolu TCP mezi aplikací LonOPC a monitorovací aplikací LonOPCClient byl navržen jednoduchý komunikační protokol. Z hlediska této komunikace může LonOPC i LonOPCClient vystupovat v roli klienta i serveru. Protokol umožňuje použít tři typy komunikace, které byly pojmenovány podle jejich významu:

- Služba *send_me* slouží pro získání dat ze serveru, klient vysílá zprávu s příznakem *send_me* a odpovědí mu je zpráva s příznakem *send_me_resp* (tato zpráva obsahuje také požadovaná data). Nebo je odpovědí serveru zpráva s příznakem *send_me_err* (v tom případě zpráva obsahuje pouze identifikátor chyby).

- Služba *send_to* slouží pro zaslání dat klientem na server. Odpovědí serveru je zpráva s příznakem *send_to_resp*, pokud server data úspěšně přijal, případně je odpověď *send_to_err* pokud server zaslaná data neakceptuje.

- Služba *send_to_noack* slouží pro zaslání zprávy klientem na server. Server ovšem neposílá žádnou odpověď na tuto zprávu.

Struktura zprávy je znázorněna na obr. 5.3, pole *služba* obsahuje typ komunikace popsaný výše, pole *příkaz* určuje druh dat a význam dat (přehled příkazů je uveden v tab. 5.1) a pole *délka* obsahuje délku dat v poli *data*.



Obrázek 5.3: Protokol komunikace s TCP serverem aplikace LonOPC.

Kód příkazu	Symbolické jméno	Význam
1	NUM_VARS	Celkový počet proměnných.
2	SNVT	Typ proměnné, data v požadavku obsahují index proměnné (0 až NUM_VARS-1).
3	SERVICE_PIN	Požadavek na vyslání zprávy servisního pinu.
4	LOG_MSG	Textová zpráva.
5	VAR_VALUE	Hodnota proměnné, data v požadavku obsahují index proměnné (0 až NUM_VARS-1).
6	NUM_ADDR	Počet adres.
7	ADDR_VAL	Hodnota adresy, data v požadavku obsahují index proměnné (0 až NUM_VARS-1).
8	NV_CONF	Konfigurace jedné proměnné, data v požadavku obsahují index proměnné (0 až NUM_VARS-1).
9	DOMAIN_0	Adresa domény 0.
10	DOMAIN_1	Adresa domény 1.
11	VAR_NAME	Jméno proměnné, data v požadavku obsahují index proměnné (0 až NUM_VARS-1).

Tabulka 5.1: Příkazy TCP protokolu.

Protokol je jednoduchý, neobsahuje označování (číslování) zpráv. Proto je nutné požadavky na straně serveru i klienta vyřizovat pouze sekvenčně, což v případě delší doby obsluhy některého z požadavků je nevýhodné. Označování zpráv by bylo možné doplnit do pole označené *služba*, ve kterém byly pro tento účel rezervovány 4 bity (pole *rezerva* na obr. 5.3). Zpráva také neobsahuje žádný mechanismus zabezpečující konzistenci dat (např. CRC), zde se spoléhá na kvality služby TCP, případně by bylo možné doplnit CRC součet na konec zprávy. Protokol byl navržen takto jednoduše, protože pro účely monitorování činnosti OPC serveru to postačuje. Pokud by měl být tento způsob použit i pro jiné aplikace, např. pro vzdálené načítání dat SP, bylo by třeba doplnit zmíněné označování zpráv, případně doplnit nějaký mechanismus přístupových práv.

5.4 Testování aplikace LonOPC

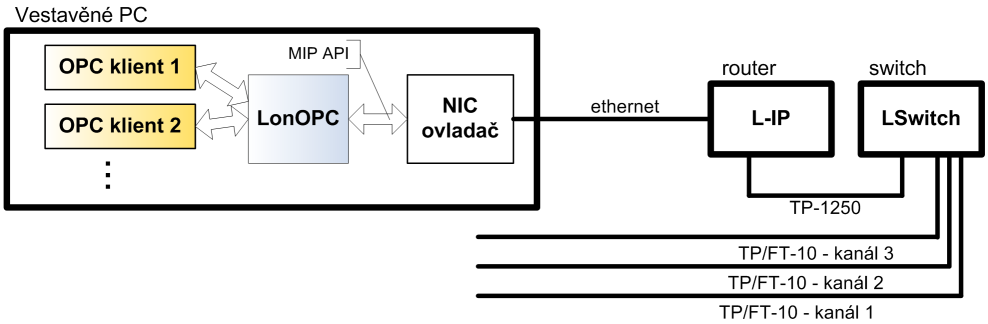
5.4.1 Sestava zařízení pro testování

Na Katedře řídicí techniky ČVUT-FEL je k dispozici model domovní automatizace[5] (viz obr. 5.4), který obsahuje zařízení, která jsou propojena sítí LonWorks.

V modelu je také vestavěno průmyslové PC, na kterém jsem prováděl vývoj a testování serveru LonOPC i aplikace OpenLDVStackDemo. Z tohoto PC je možné do sítě LonWorks instalované v modelu přistupovat přes zařízení L-IP router, který slouží k tunelování protokolu LonTalk přes ethernet dle protokolu EIA-852. Propojení PC s L-IP routerem a sítí LonWorks včetně softwaru na straně PC je schématicky zachyceno na obr. 5.5. LonOPC server tedy pro přístup do sítě LonWorks využíval MIP API, které je u ovladačů k zařízením Loytec poskytováno jako nadstavba nad standardní ovladač nazývaný NIC (Network Interface), detaily lze nalézt v [7] a [8].

[5] Podrobnosti o tomto modelu lze nalézt v diplomových pracích [28] a [25].

Obrázek 5.4: Model domovní automatizace.

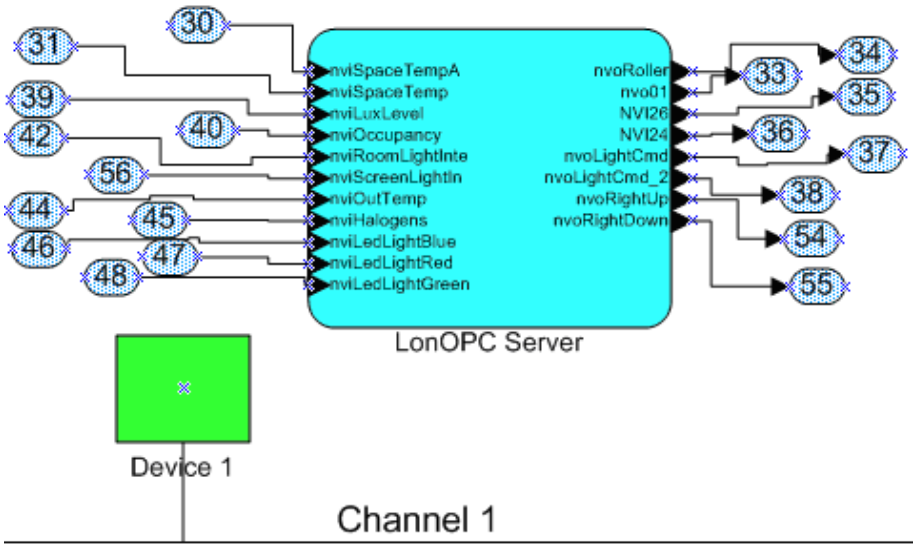


Obrázek 5.5: Přístup OPC serveru LonOPC do sítě LonWorks v modelu domovní automatizace.

5.4.2 Testování LonOPC s OPC klienty

Pro účely testování jsem aplikaci LonOPC nainstaloval dle popisu v příloze F a začlenil jsem ji do sítě pomocí konfiguračního nástroje LonMaker. Do virtuálního funkčního bloku *LonOPC server* zařízení LonOPC jsem přidal síťové proměnné (přidání dynamických SP viz příloha B) tak, abych mohl LonOPC propojit s ostatními zařízeními, která jsou v modelu připojena k síti LonWorks. Na obr. 5.6 je funkční blok *LonOPC server*, na

kterém jsou vidět všechny připojené vstupní i výstupní SP, celkem bylo připojeno 19 SP.



Obrázek 5.6: Funkční blok LonOPC při testování.

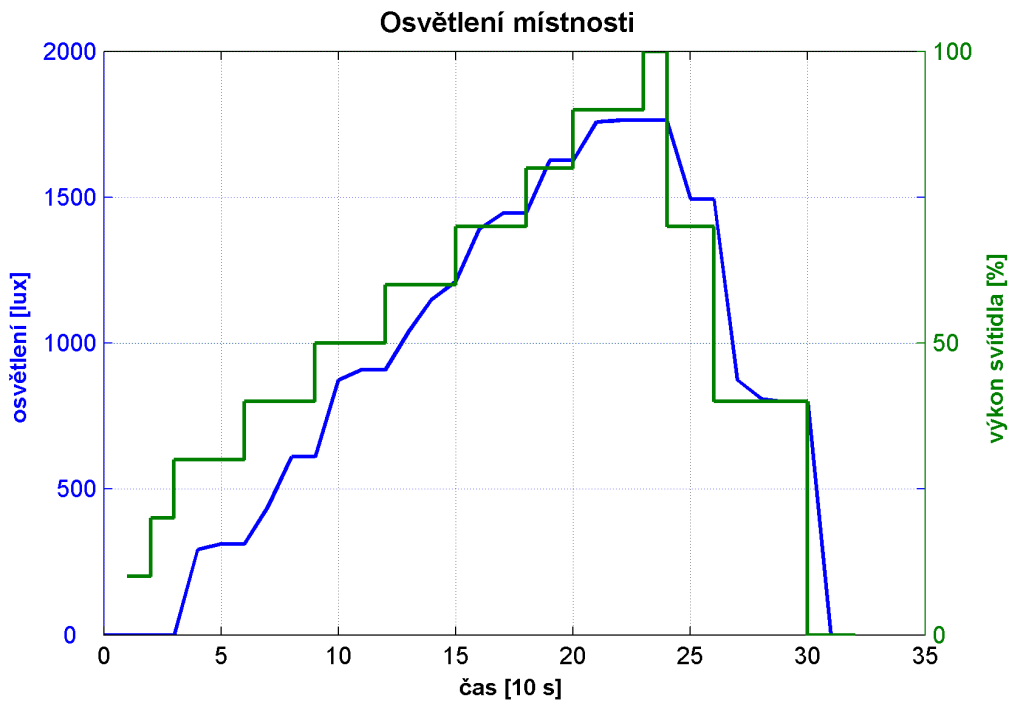
Funkci serveru LonOPC jsem testoval pomocí OPC klienta ICONICS OPC DataSpy[6], ze kterého jsem připojená zařízení ovládal (např. rozsvěcel světla, ovládal rolety, okna atd.) a vyčítal jsem data ze zařízení (např. teploty).

5.4.2.1 Testování s OPC Toolboxem Matlabu

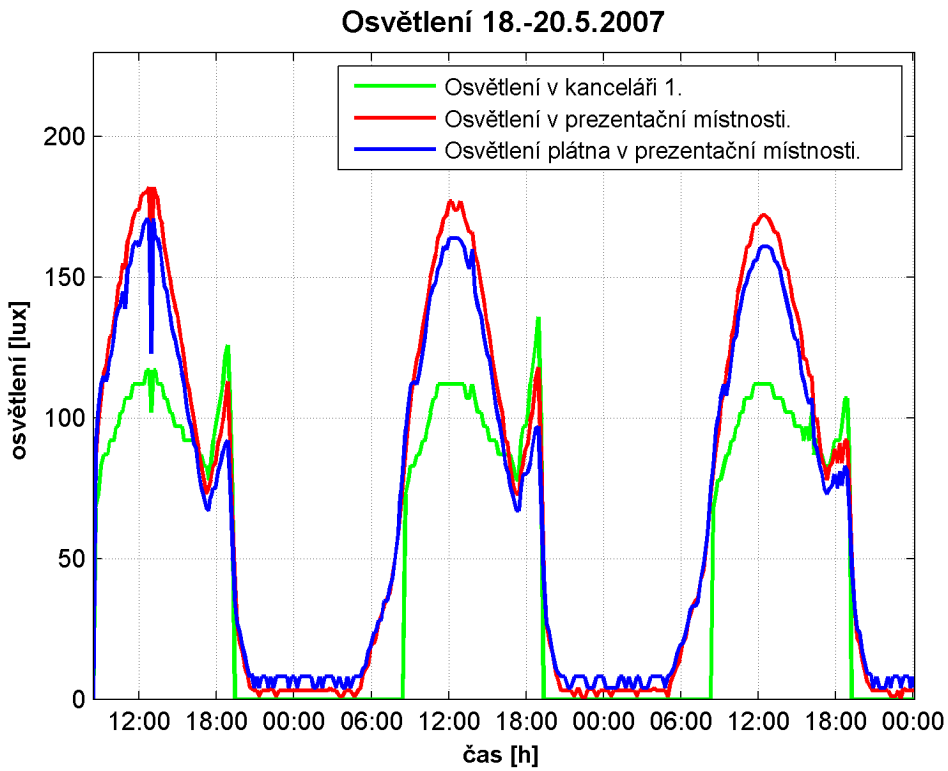
Další OPC klient, kterého jsem použil, byl OPC Toolbox z prostředí Matlab, který umožňuje zaznamenávat časové průběhy hodnot získaných z OPC serveru. Příklad takto naměřených dat je na obr. 5.7, na kterém je záznam měření osvětlení v jedné z místností modelu v závislosti na nastaveném výkonu svítidla. Výkon svítidla byl ovládán také přes LonOPC server.

Záznam z dlouhodobějšího provozu LonOPC serveru je na obrázcích 5.8 a 5.9. Měření bylo prováděno od rána 18. května do půlnoci 20. května 2007, během měření se nevyskytly žadné problémy. Na obr. 5.8 je zobrazen průběh přirozeného osvětlení v jednotlivých místnostech modelu. Na obr. 5.9 je průběh teplot vzduchu v místnostech a teplota vody v topení (během měření bylo topení vypnuto), je vidět že teplotní čidla mají výrazný ofset a bylo by vhodné je zkalibrovat. Také si lze všimnout, že teplota otopné vody má dopravní zpoždění 3 h vzhledem k teplotě vzduchu (v době kdy je topení vypnuté). Skokový průběh teploty *vnější stěny modelu* je způsoben nastavením histereze u čidla.

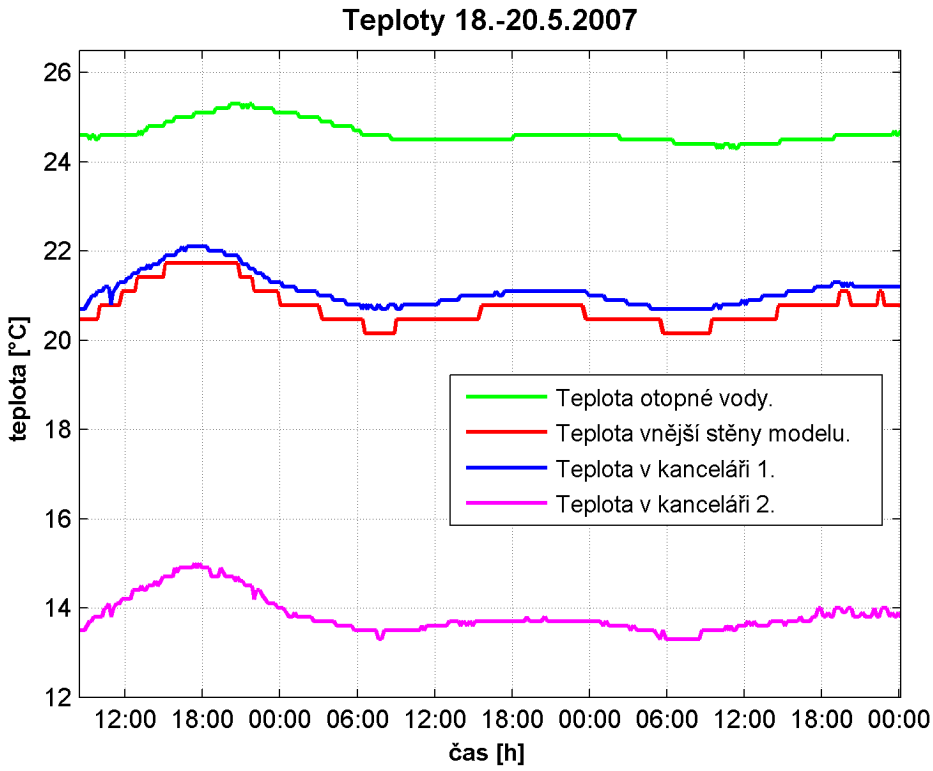
[6] Je volně ke stažení na www stránce firmy ICONICS (http://www.iconics.com/).



Obrázek 5.7: Osvětlení v závislosti na výkonu svítidla.



Obrázek 5.8: Osvětlení v modelu domovní automatizace během 3 dní.



Obrázek 5.9: Teploty v modelu domovní automatizace během 3 dní.

Kapitola 6

Závěr

V této práci jsem se seznámil s technologií LonWorks a zabýval jsem se realizací zařízení, které komunikuje s ostatními zařízeními v síti LonWorks pomocí protokolu LonTalk.

Provedl jsem rozbor možných řešení přístupu k síti LonWorks (viz kap. 2.2) a na základě tohoto rozboru jsem navrhl architekturu pro realizaci zařízení, které slouží pro monitorování a ovládání zařízení v síti LonWorks. Toto zařízení je realizováno jako aplikace běžící na počítači PC s operačním systémem Windows a pro přístup k síti LonWorks používá síťové rozhraní s ovladačem, který poskytuje MIP API nebo OpenLDV API.

Z takto zvolené architektury vyplynula potřeba implementace aplikační vrstvy protokolu LonTalk, protože implementace protokolu LonTalk s vhodnou licencí není k dispozici (viz 2.1.1). Implementace aplikační vrstvy protokolu LonTalk spolu s implementací interpretace síťových proměnných dle sdružení LonMark byla jádrem této práce. Proto jsem také teoretickému rozboru aplikační vrstvy protokolu LonTalk věnoval kap. 3 a popisu mé implementace je věnována kap. 4. Mnou provedená implementace (nazvaná Rozhraní SP - Rozhraní síťových proměnných) vychází ze specifikace protokolu LonTalk ANSI/EIA 709.1-B. Rozhraní SP obsahuje také koncept dynamických proměnných, který umožňuje dynamicky měnit funkční profil aplikace přidáním nebo odebráním síťových proměnných potřebných typů pomocí konfiguračního nástroje (např. pomocí aplikace LonMaker).

Rozhraní SP také obsahuje implementaci interpretace obsahu síťových proměnných dle standardů sdružení LonMark. Aplikace využívající Rozhraní SP je tedy kompatibilní s ostatními zařízeními jiných výrobců, která disponují rozhraním dle standardů sdružení LonMark. V rámci implementace Rozhraní SP jsem vytvořil aplikaci OpenLDVStackDemo (viz kap. 4.7), kterou lze použít pro čtení hodnot síťových proměnných ze sítě LonWorks a pro zasílání hodnot síťových proměnných do zařízení v síti LonWorks. Hodnoty síťových proměnných jsou v aplikaci OpenLDVStackDemo také interpretovány dle LonMark standardů, lze je tedy prohlížet a editovat po jednotlivých částech, které mají konkrétní

funkční význam (viz např. obr. 4.8 a 4.7).

V druhé části této práce jsem se zabýval implementací OPC serveru pro přístup k zařízením v síti LonWorks. OPC server nazvaný LonOPC jsem vytvořil s využitím nástroje OPC ToolWorX firmy ICONICS a s využitím zmíněného Rozhraní SP (popis architektury LonOPC viz kap. 5.2). Server LonOPC pro svou činnost nepotřebuje LNS server (dodávaný firmou Echelon). LonOPC potřebuje pouze síťové rozhraní s ovladačem, který nabízí MIP API nebo OpenLDV API, takové ovladače dodává ke svým síťovým rozhraním většina výrobců (viz tab. 4.1). Nezávislost na LNS Serveru je výhoda LonOPC serveru oproti většině dostupných OPC serveru pro LonWorks. Položky serveru LonOPC lze konfigurovat (přidávat a odebírat) pomocí nástroje pro konfiguraci sítě LonWorks (např. pomocí aplikace LonMaker), je zde využíván koncept dynamických proměnných. Jelikož server LonOPC nemá uživatelské rozhraní, tak jsem pro správu a monitorování běhu serveru LonOPC vytvořil aplikaci LonOPCClient (viz kap. 5.3). Tato aplikace slouží zejména pro spravování síťového rozhraní, které LonOPC server používá a pro monitorování běhu serveru. Aplikace LonOPCClient komunikuje s LonOPC serverem pomocí protokolu TCP/IP. LonOPC server byl úspěšně otestován na modelu domovní automatizace na Katedře řídicí techniky ČVUT-FEL.

Pro aplikaci OpenLDVStackDemo a server LonOPC jsem vytvořil XIF soubor, který popisuje rozhraní zařízení a je nezbytný pro konfigurační nástroj, kterým je daná aplikace začleňována do sítě LonWorks (popis vytvoření XIF souboru je v kap. 4.8).

Veškerý software vytvořený v rámci této práce jsem implementoval objektovým přístupem v jazyce C++. Ke zdrojovým kódům jsem vytvořil dokumentaci v podobě referenční příručky pomocí nástroje Doxygen (viz adresář Doc na přiloženém CD).

Vyvinutý software by bylo možné ještě zdokonalit, k OPC serveru LonOPC by bylo vhodné doplnit instalátor (průvodce) pro zjednodušení instalace a konfigurace serveru. Také by bylo možné u aplikace OpenLDVStackDemo vylepšit uživatelské rozhraní.

Věřím, že software a příslušná dokumentace, které jsem vytvořil v rámci této práce budou přínosné zejména pro toho, kdo bude implementovat zařízení komunikující prostřednictvím LonWorks. Práce může být také využita při výuce komunikace řídicích systémů pomocí technologie LonWorks na Katedře řídicí techniky ČVUT-FEL.

Literatura

[1] *Control Network Protocol Specification CEA-709.1-B.* Arlington: Consumer Electronics Association, 2002.

[2] *PROTOCOL PATENT LICENCE AGREEMENT* [online]. Echelon Corporation., March 2002 [cit. 2007-02-08]., ⟨http://www.echelon.com/developers/lonworks/protocol/protocolpat.pdf⟩.

[3] *User's Manual LC3k Controller Family.* LOYTEC electronics GmbH., July 2005 [cit. 2007-02-08]., ⟨http://www.loytec.com/assets/contents/files/LC3k_user_manual_1_3.zip⟩.

[4] *The IEEE 1473-L Train Network Protocol* [online]. Echelon Corporation., c1999 [cit. 2005-03-01]., ⟨http://www.tsd.org/ieee1473/⟩.

[5] Echelon. *Implementing Dynamic Network Variables* [online]. August 2000. [cit. 2007-03-15], ⟨http://www.echelon.com/support/documentation/docs/ImplementingDynamicNetworkVariables.pdf⟩.

[6] Echelon. *ISI Protocol Specification* [online]. Version 3. [cit. 2007-02-14], ⟨http://www.echelon.com/support/documentation/manuals/devtools/078-0300-01E.pdf ⟩.

[7] Loytec electronics GmbH. *NIC Developer's Manual for Loytec Network Interfaces.* LOYTEC electronics GmbH., Version 2.6 [cit. 2007-04-07]., ⟨http://www.loytec.com/assets/contents/pdf/NIC_Developers_Manual.pdf⟩.

[8] Loytec electronics GmbH. *NIC User Manual.* LOYTEC electronics GmbH., Version 2.6 [cit. 2007-04-23]., ⟨http://www.loytec.com/assets/contents/pdf/NIC_User_Manual.pdf⟩.

[9] Ernest Laurentin. *Multi-threaded Client/Server Socket class* [online]. Updated April 2004 [cit. 2007-04-18]., ⟨http://www.codeproject.com/internet/serversocket.asp⟩.

[10] Echelon. *LonMark SNVT Master List.* Version 13 Revision 00 [cit. 2007-02-15], ⟨https://cert.lonmark.org/installers/1300/LonMarkResourceFiles1300.exe⟩.

[11] Echelon. *File Transfer LonWorks Engineering Bulletin.* November 2006 [cit. 2007-04-12], ⟨http://www.echelon.com/Support/documentation/Bulletin/005-0025-01D.pdf⟩.

[12] Echelon. *LonWorks Microprocessor Interface Program (MIP) User's Guide* [online]. Revision 3 [cit. 2007-02-28]., ⟨http://www.echelon.com/support/documentation/manuals/devtools/078-0017-01C.pdf ⟩.

[13] Echelon. *ShortStack Users's Guide* [online]. Version 2, March 2002 [cit. 2007-03-28]., ⟨http://www.echelon.com/support/documentation/manuals/devtools/078-0189-01C.pdf ⟩.

[14] Echelon. *OpenLDV Programmer's Guide* [online]. Release 2.1 [cit. 2007-04-04]., ⟨http://www.echelon.com/support/documentation/manuals/devtools/078-0275-01C.pdf⟩.

[15] Hanzálek, Z.; Čapek, J. *Channel Backlog Estimation in LonWorks.* In Mahalik, N.P.(Ed.): Fieldbus Technology - Industrial Network Standards for Real-Time Distributed Control. Springer, 2003.

[16] ICONICS. *OPC ToolWorX Version 3.1.* Version 3.1, June 2004 [cit. 2007-04-15].

[17] Linhart, M. *Komunikační sběrnice LON (Local Operating Network).* Praha, 2004. Diplomová práce, ČVUT, Fakulta elektrotechnická, Katedra řídicí techniky.

[18] LonMark. *LonMark Resource File API Reference Guide.* Revision 3, June 2003 [cit. 2007-04-08], K dipozici na ⟨http://www.lonmark.com⟩.

[19] LonMark International. *LonMark Application Layer Interoperability Guidelines.* Version 3.4 September 2005 [cit. 2007-02-15], ⟨http://www.lonmark.com/technical_resources/guidelines/docs/LmApp34.pdf⟩.

[20] LonMark International. *LonMark Layer 1-6 Interoperability Guidelines.* Version 3.4 [cit. 2007-02-15], ⟨http://www.lonmark.com/technical_resources/guidelines/docs/LmPhy34.pdf⟩.

[21] LonMark International. *LonMark Device Interface File Reference Guide.* Revision 4.4 June 2003 [cit. 2007-02-23], ⟨http://www.lonmark.org/technical_resources/guidelines/docs/LmXif4401.pdf⟩.

[22] Loy, D.; Dietrich, D.; Scheinzer, H. J. *Open Control Networks.* Kluwer Academic Publishers, 2001.

[23] Microsoft. *MSDN* [online]. [cit. 2007-04-04], ⟨http://msdn.microsoft.com⟩.

[24] Motorola. *LonWorks Technology Device Data.* Rev. 5 [cit. 2007-03-07], ⟨http://www.toshiba.com/taec/components/Generic/DB_neuron.jsp⟩.

[25] Slezák, M. *Model domovní automatizace.* Praha, 2006. Diplomová práce, ČVUT, Fakulta elektrotechnická, Katedra řídicí techniky.

[26] Tiersch, F. *LonWorks Technology An Introduction.* Erfurt: LonTech Thüringen e. V., 2005.

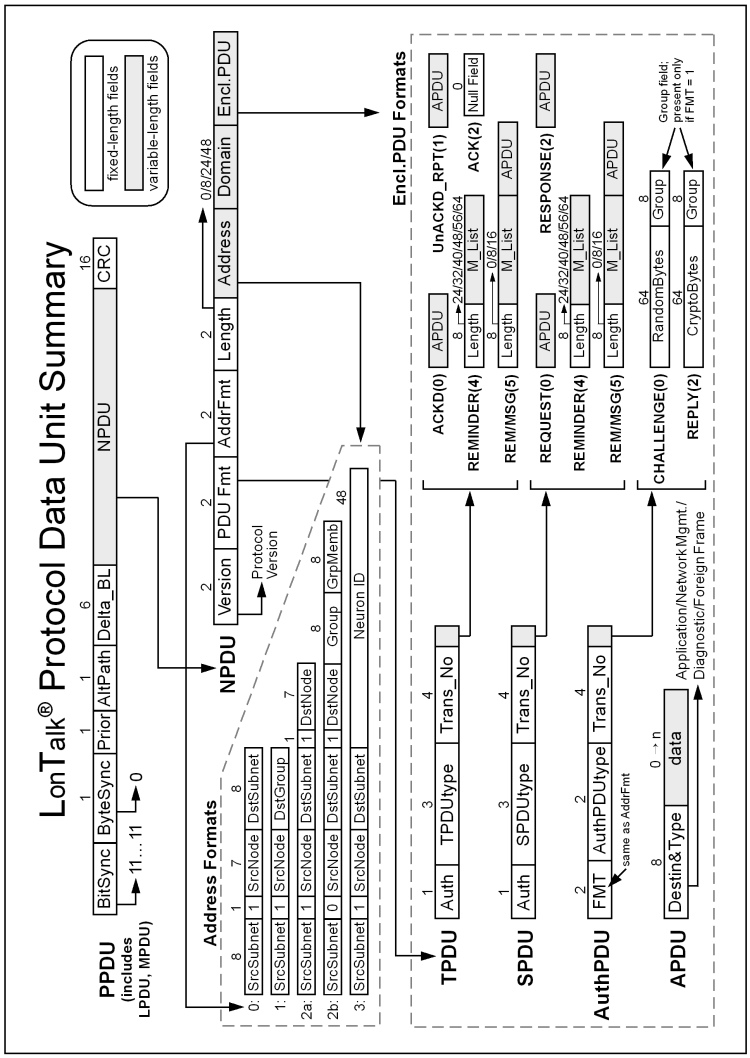
[27] Virius, M. *Jazyky C a C++, kompletní kapesní průvodce programátora.* Praha: Grada, 2006.

[28] Vozár, V. *Model automatizace budov.* Praha, 2005. Diplomová práce, ČVUT, Fakulta elektrotechnická, Katedra řídicí techniky.

[29] Šusta, R. *Programování pro řízení ve Windows.* Praha: Vydavatelství ČVUT-FEL, 1999.

Příloha A

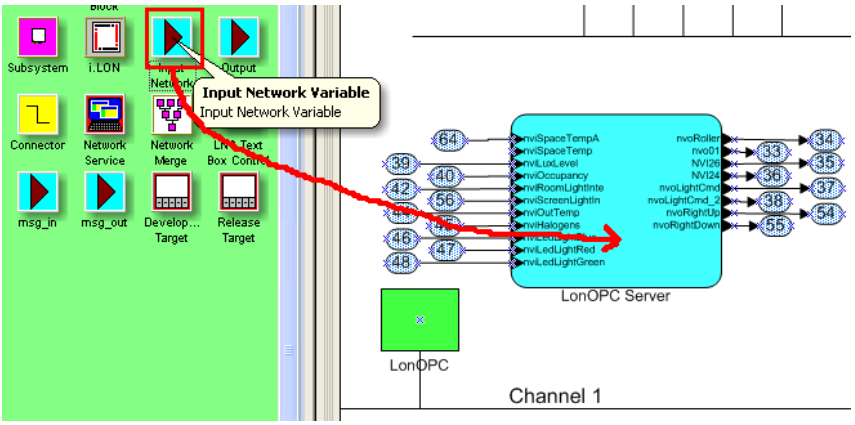
Formát zprávy - PDU



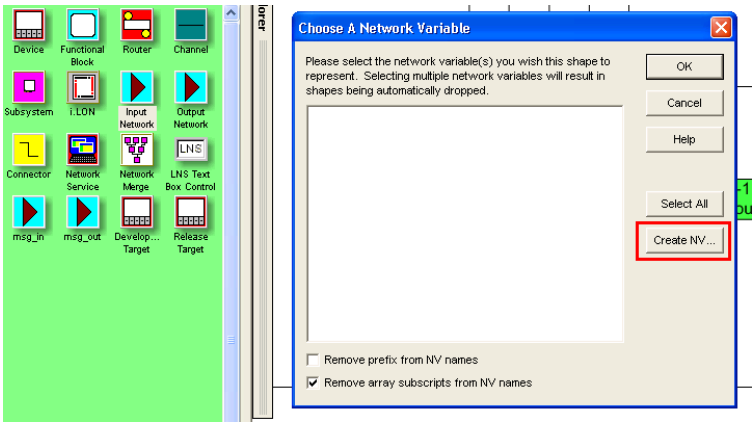
Obrázek A.1: Zpráva protokolu LonTalk (PDU - Protocol Data Unit).

Příloha B

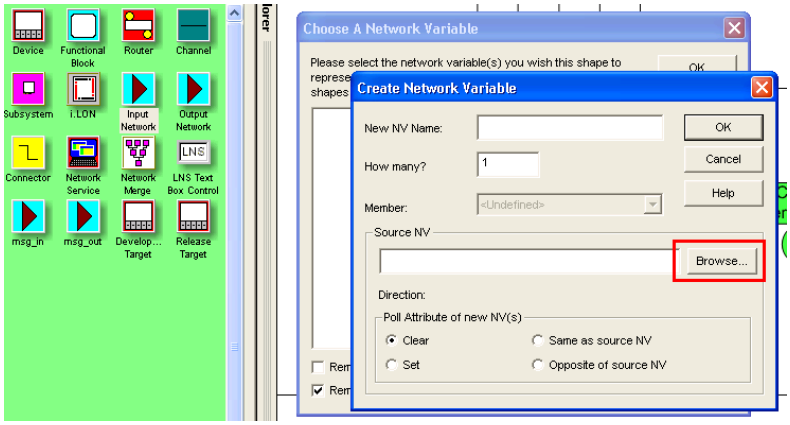
Přidání dynamické síťové proměnné v aplikaci LonMaker



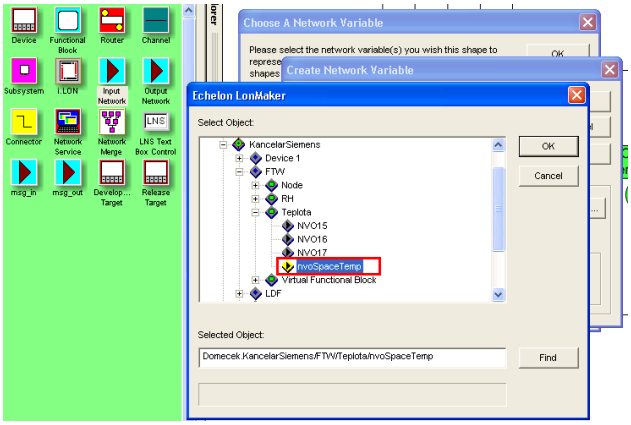
Obrázek B.1: Výběr objektu *vstupní proměnná* a přetažení nad objekt funkčního bloku.



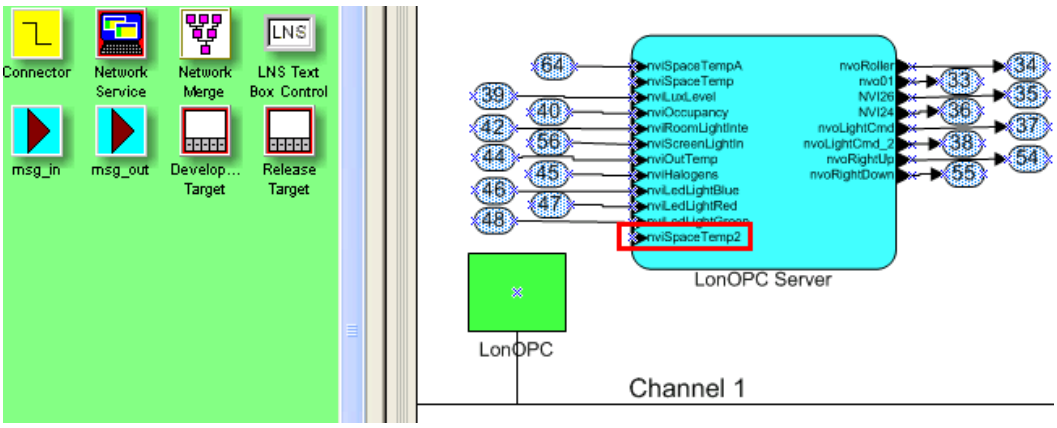
Obrázek B.2: Vytvoření proměnné.



Obrázek B.3: Určení zdroje proměnné, který určí SNVT typ vytvářené proměnné.



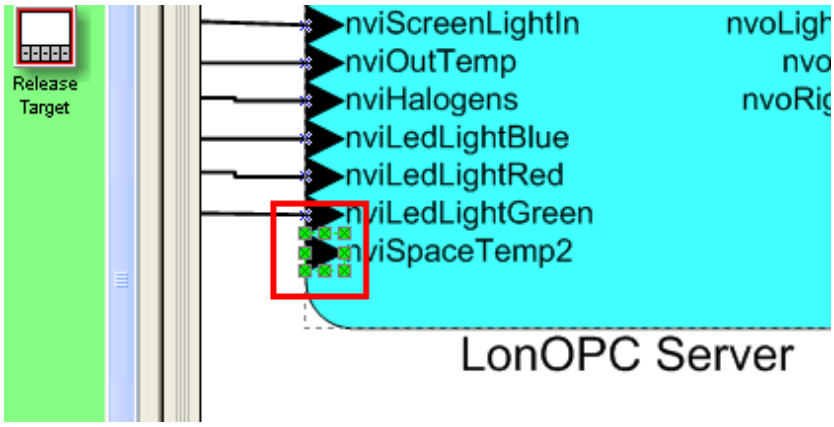
Obrázek B.4: Vybrání zdrojové proměnné, ke které bude nově vytvořená proměnná připojena.



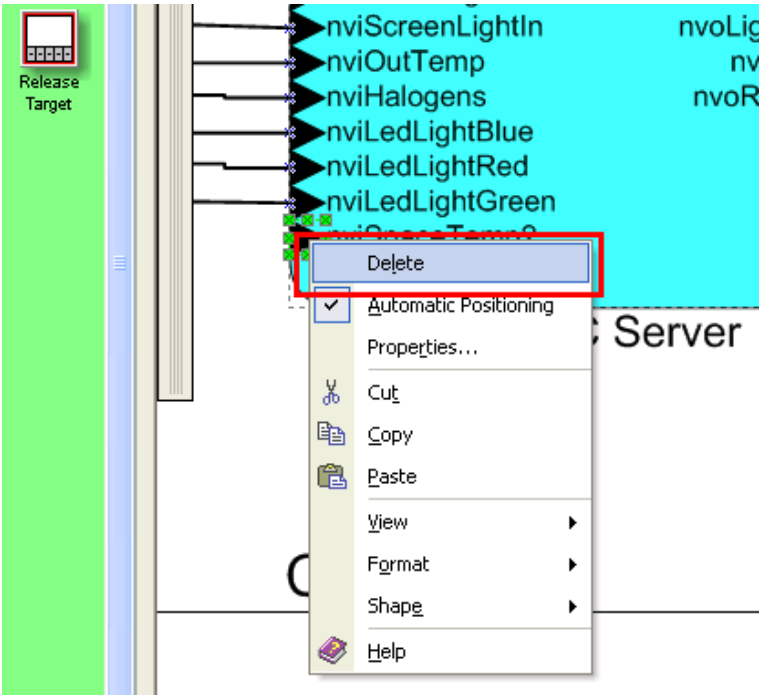
Obrázek B.5: V bloku funkčního profilu je nově vytvořená proměnná.

Příloha C

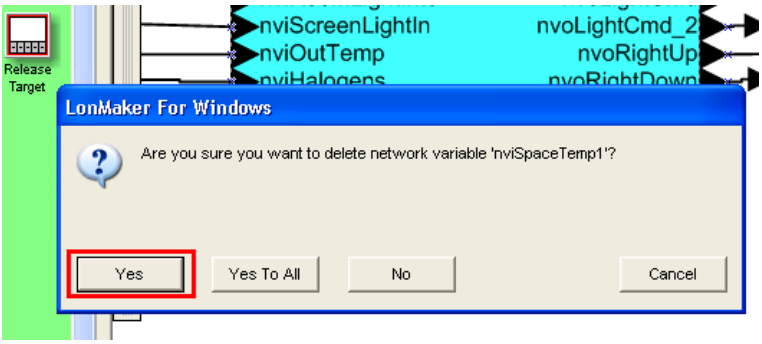
Odebrání dynamické síťové proměnné v aplikaci LonMaker



Obrázek C.1: Vybrání proměnné, kterou chceme odebrat.



Obrázek C.2: Kontextové menu - vybereme Delete.



Obrázek C.3: Potvrzení odebrání.

Příloha D

Deklarace tříd použitých v Rozhraní SP

```
class CAddressTable
{
  public:
    CAddressTable(Word number_of_entries);
    ~CAddressTable(void);

    bool    UpdateAddress(Word index, Taddress_struct_ext
       new_address);
    bool    GetAddress(Word index, Taddress_struct_ext*
       paddress);
    string  ToString(void);
    bool    Save(string file_name);
    bool    Load(string file_name);
  private:
    CAddressesVector AddressesVector;
};
```

Výpis D.1: Deklarace třídy `CAddressTable`

```
class CDomainTable
{
  public:
    CDomainTable(Word number_of_entries);
    ~CDomainTable(void);

    bool   UpdateDomain(Word index, Tdomain_struct new_domain);
    bool   GetDomain(Word index, Tdomain_struct* pdomain);
    bool   Load(string file_name);
    bool   Save(string file_name);
    string ToString(void);
  private:
    CDomainVector DomainVector;
};
```

Výpis D.2: Deklarace třídy `CDomainTable`

Příloha E

XIF soubor

```
1  File: LonOPC_50NV.XIF generated by Manual 0.0.0 Version 1.0,
       XIF Version 4.400
2  Copyright (C): none
3  Run on Mon Apr 09 12:00:00 2007
4
5  90:00:00:00:00:00:00:10
6  2 15 0 5 0 3 3 3 3 3 4 11 9 2 4 0 0 4 11 1 1 11 50 0 1 2 0 0
       0 0 0 0 2 15 0 0 0 0 0 1 0 0
7  14 7 13 13 28 554 0 15 5 3 174 4 10000000
8  1 7 1 0 4 4 4 15 200 0
9  78125 0 0 0 0 0 0 0 0 0 0 0
10 90 0 240 0 0 0 40 40 0 7 50 19 54 100 103
11 *
12 "&3.3@0NodeObject
13
14 VAR␣nviRequest␣0␣0␣0␣0
15 0␣1␣63␣0␣0␣1␣0␣1␣0␣1␣0␣0␣0
16 "@0|1
17 92 *   2
18 2 0 0 0 0
19 1 0 0 1 0
20 VAR nviFileReq 1 0 0 0
21 0 1 63 0 0 1 0 1 0 1 0 0 0
22 "@0|5
23 73␣*␣␣␣6
24 1␣0␣0␣1␣0
25 2␣0␣0␣0␣0
26 2␣0␣0␣0␣0
27 4␣0␣5␣0␣0
28 1␣0␣0␣0␣0
29 1␣0␣0␣0␣0
30 VAR␣nviFilePos␣2␣0␣0␣0
31 0␣1␣63␣0␣0␣1␣0␣1␣0␣1␣0␣0␣0
```

```
32 "@0|7
33 90 *    2
34 1 0 0 1 4
35 2 0 0 0 0
36 VAR nvoStatus 3 0 0 0
37 0 1 63 1 0 1 0 1 0 1 0 1 0
38 "@0|2
39 93␣*␣␣26
40 2␣0␣0␣0␣0
41 3␣0␣1␣0␣0
42 3␣1␣1␣0␣0
43 3␣2␣1␣0␣0
44 3␣3␣1␣0␣0
45 3␣4␣1␣0␣0
46 3␣5␣1␣0␣0
47 3␣6␣1␣0␣0
48 3␣7␣1␣0␣0
49 3␣0␣1␣0␣0
50 3␣1␣1␣0␣0
51 3␣2␣1␣0␣0
52 3␣3␣1␣0␣0
53 3␣4␣1␣0␣0
54 3␣5␣1␣0␣0
55 3␣6␣1␣0␣0
56 3␣7␣1␣0␣0
57 3␣0␣1␣0␣0
58 3␣1␣1␣0␣0
59 3␣2␣1␣0␣0
60 3␣3␣1␣0␣0
61 3␣4␣1␣0␣0
62 3␣5␣1␣0␣0
63 3␣6␣1␣0␣0
64 3␣7␣1␣0␣0
65 3␣0␣8␣0␣0
66 VAR␣nvoFileStat␣4␣0␣0␣0
67 0␣1␣63␣1␣0␣1␣0␣1␣0␣1␣0␣0␣0
68 "@0|6
69 74 *    4
70 1 0 0 1 0
71 2 0 0 0 0
72 2 0 0 0 0
73 4 0 22 0 0
74
75 FILE template 0 2
76 "1.1;
77 "1,0,0\x80,17,31;
78 "1,0,0\x80,165,1;
79 "1,0,0\x80,166,1;
```

```

FILE value 1 1
\xAA \xBB \x00 \x00 \x00 \x00 \x00 \x00
          \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00
          \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00
          \x00 \x00 \x00 \x00 \x00 \xEE \xFF
\x01 \x02
```

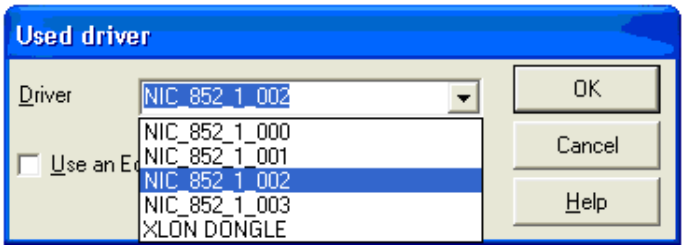
Výpis E.1: XIF soubor pro aplikaci OpenLDVStackDemo a server LonOPC.

Příloha F

Instalace a konfigurace OPC serveru LonOPC

F.1 Potřebný software

Je zapotřebí server LonOPC, nástroj pro konfiguraci sítě LonWorks, např. LonMaker. Je třeba mít nainstalované ovladače síťového rozhraní. V případě použití síťového rozhraní firmy Loytec je třeba mít nainstalován NIC Legacy Driver a ten nakonfigurovat pomocí aplikace LConfig. Dále potřebujeme vědět identifikátor (jméno) síťového rozhraní, pod kterým se lze k němu připojit. Tyto názvy jsou uloženy v registrech Windows (viz kap. 4.3.2). Jednoduše je lze zjistit např. pomocí utility NLUtil, která je v trial verzi volně dostupná na www stránkách firmy NEWRON SYSTEM (http://www.newron-system.com). Jména všech síťových rozhraní aktuálně dostupných v počítači zjistíme pomocí této utility výběrem v menu `File` → `New Site`, otevře se okno Used driver (viz obr. F.1), kde je v seznamu Driver výpis všech dostupných síťových rozhraní.



Obrázek F.1: Jména síťových rozhraní.

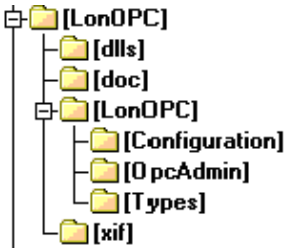
F.2 Instalace serveru LonOPC

LonOPC server se k síti LonWorks připojuje pomocí síťového rozhraní (nevyužívá LNS Server) a proto v síti LonWorks vystupuje ve stejné roli jako ostatní zařízení. Všechny proměnné, které chceme pomocí LonOPC sledovat musíme připojit jako vstupní proměnné LonOPC a všechny proměnné, pomocí kterých chceme ovládat zařízení v síti LonWokrs musíme připojit jak výstupní proměnné LonOPC. LonOPC podporuje koncept dynamických proměnných, takže proměnné, které LonOPC na svém vstupu či výstupu má k dispozici, můžeme měnit dle potřeby.

F.2.1 Postup instalace serveru LonOPC

F.2.1.1 Rozbalení instalačního balíku

Do libovolného adresáře rozbalíme soubor LonOPC.zip[1], získáme adresářovou strukturu jako na obr. F.2.



Obrázek F.2: Struktura adresářů.

DLL knihovny z adresáře dll zkopírujeme do systémového adresáře (obvykle c:\windows).

F.2.1.2 Základní konfigurace

Provedeme základní konfiguraci serveru. Konfigurace je uložena v konfiguračním souboru `lon_opc_conf.cnf`, který můžeme editovat jako textový soubor a ve kterém provedeme základní nastavení. Příkladový obsah souboru je ve výpisu F.1.

F.2.1.3 Význam položek konfiguračního souboru

Version: verze konfiguračního souboru, tuto položku neměníme.

Interface name: název síťového rozhraní, pomocí kterého bude OPC server přistupovat do sítě LonWorks, podrobnější popis jak tento název získat viz odstavec F.1.

[1] Na přiloženém CD je uložen v adresáři `Software\Bin`.

```
Version:1
Interface name:NIC_852_1_002
Address file:c:\radek\_final\LonOPC\Configuration\addresses.dat
Configuration file:c:\radek\_final\LonOPC\Configuration\domains.dat
Dynamic file:c:\radek\_final\LonOPC\Configuration\dynamic.dat
Types:c:\radek\_final\LonOPC\Types\STANDARD.TYP
Types enu:c:\radek\_final\LonOPC\Types\STANDARD.ENU
```

Výpis F.1: Konfigurační soubor LonOPC

Address file: cesta (musí být kompletní, nestačí relativní cesta) k souboru kam se ukládá tabulka adres.

Configuration file: cesta (musí být úplná, nestačí relativní cesta) k souboru kam se ukládají konfigurační informace.

Dynamic file: cesta (musí být kompletní, nestačí relativní cesta) k souboru kam se ukládají informace o nakonfigurovaných dynamických proměnných OPC serveru.

Types: cesta (musí být kompletní, nestačí relativní cesta) k souboru se specifikací typů proměnných dle sdružení LonMark (nejaktuálnější verzi tohoto souboru lze získat z www stránek sdružení LonMark).

Types enu: cesta (musí být kompletní, nestačí relativní cesta) k souboru s texty ke specifikacím typů proměnných dle sdružení LonMark (nejaktuálnější verzi tohoto souboru lze získat z www stránek sdružení LonMark).

Nejdůležitější položka je **Interface name**. U ostatních položek upravíme cesty dle adresáře, kam jsme vše rozbalili. Soubory z položek **Address file**, **Configuration file** a **Dynamic file** obsahují informace o přidaných dynamických síťových proměnných OPC serveru a o jejich zapojení s ostatními SP ostatních zařízení sítě LonWorks. Pokud chceme zálohovat "zapojení síťových proměnných", zazálohujeme právě tyto soubory.

F.2.1.4 Registrace OPC serveru

OPC server musíme registrovat v systému Windows, registrace v podstatě znamená zápis informace o existenci OPC serveru do registrů systému. Spuštěním dávkového souboru `reg_normal.bat` provedeme registraci OPC serveru LonOPC jako klasického spustitelného exe souboru. Spuštěním dávkového souboru `reg_service.bat` provedeme registraci OPC jako služby Windows. Odregistrování serveru můžeme provést spuštěním souboru `unreg.bat`. Pokud OPC server zaregistrujeme jako službu, je třeba také nastavit vlastnosti zabezpečení této služby, protože OPC server obsahuje podporu DCOM.

Toto provedeme v Ovládací panely → Nástroje pro správu → Služba Component Services, kde vybereme Služba komponent → Počítače → Tento počítač DCOM a službu OpenLDVServerDA and OpenLDVServerAE. Jedno z možných nastavení je nastavit přihlášení jako uživatel s dostatečnými oprávněními (lze provést na kartě Identita).

F.2.1.5 Spuštění serveru

Pokud jsme server zaregistrovali jako službu, spustíme jej ze správce služeb Windows (Ovládací panely → Nástroje pro správu → Služby). Pokud jsme jej zaregistrovali jako klasický exe soubor, nemusíme jej spouštět, bude spuštěn automaticky systémem v okamžiku, kdy jej nějaký OPC klient bude potřebovat.

Po spuštění zkontrolujeme soubor `_log.txt`, kam je zaznamenáván běh programu. Při úspěšném spuštění serveru vypadá výpis z logu jako ve výpisu F.2.

```
1  21.0.2007 0:7:00; INFO    ;
     ==========================================
2  21.0.2007 0:7:00; INFO    ; c:\radek\LonOPC\LonOPC\
3  21.0.2007 0:7:00; INFO    ; Logger start: OK.
4  21.0.2007 0:7:00; INFO    ; Configuration loading: OK.
5  21.0.2007 0:7:00; INFO    ; TCP server start: OK.
6  21.0.2007 0:7:04; INFO    ; Write                LDV_OK  ( 2
       Bytes: 60 00)
7  21.0.2007 0:7:04; INFO    ; Write                LDV_OK  ( 2
       Bytes: 50 00)
8  21.0.2007 0:7:04; INFO    ; Read         LDV_OK  ( 2 Bytes:
     50 00)
9  21.0.2007 0:7:04; INFO    ; Received     niRESET
10 21.0.2007 0:7:04; INFO    ; File with types found.
11 21.0.2007 0:7:04; INFO    ; File with types (strings) found.
12 21.0.2007 0:7:04; INFO    ; Configuration of dynamic
     variables loaded succesfully.
13 21.0.2007 0:7:04; INFO    ; Address table loaded succesfully.
14 21.0.2007 0:7:04; INFO    ; Domain table loaded succesfully.
15 21.0.2007 0:7:04; INFO    ; LonWorks server start: OK.
```

Výpis F.2: Obsah souboru se záznamy o běhu serveru.

F.2.1.6 Konfigurace serveru v aplikaci LonMaker

LonOPC server musíme do sítě LonWorks začlenit stejně jako jakákoliv jiná zařízení. Jako funkční profil použijeme soubor `LonOPC.XIF`, který je součástí instalačního balíku.

Při instalaci je třeba na instalovaném zařízení stisknout servisní tlačítko. Toto provedeme tak, že spustíme LonOPCClient, přihlásíme se k LonOPC na portu 50000 a poté stiskneme tlačítko Service Pin. Tím je instalace dokončena.

F.2.1.7 Konfigurace proměnných LonOPC

Síťové proměnné přidáváme (resp. odebíráme) do (resp. z) virtuálního profilu instalovaného zařízení LonOPC standardním způsobem (viz také přílohy B a C). Přidáním proměnné do virtuálního funkčního bloku je proměnná také přidána do OPC serveru LonOPC. Pokud LonOPC běží jako služba, je třeba službu restartovat, aby se v OPC serveru aktualizoval seznam proměnných.

Příloha G

Obsah přiloženého CD

K této práci je přiloženo CD, na kterém je uložen text této práce a software vyvinutý v rámci této práce. Jsou zde zdrojové kódy všech programů i spustitelné verze všech aplikací.

G.1 Struktura a obsah adresářů

- `/Software`
  - `/Bin`
    - `/LonOPC` - obsahuje OPC server LonOPC
    - `/LonOPCClient` - obsahuje aplikaci LonOPCClient
    - `/OpenLDVStackDemo` - obsahuje aplikaci OpenLDVStackDemo
  - `/Source`
    - `/LonOPC` - zdrojové kódy k LonOPC
    - `/LonOPCClient` - zdrojové kódy k LonOPCClient
    - `/OpenLDVStack` - zdrojové kódy k OpenLDVStackDemo
    - `/Doc`
      - `/OpenLDVStackDemo`
        - `/Html` - dokumentace ke zdrojovým kódům OpenLDVStackDemo
      - `/LonOPC`
        - `/Html` - dokumentace ke zdrojovým kódům LonOPC
      - `/LonOPCClient`
        - `/Html` - dokumentace ke zdrojovým kódům LonOPCClient
  - `/XIF` - obsahuje XIF soubor
- `/Text DP` - text této práce v PDF formátu