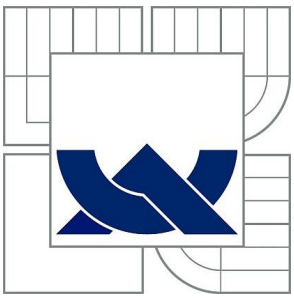
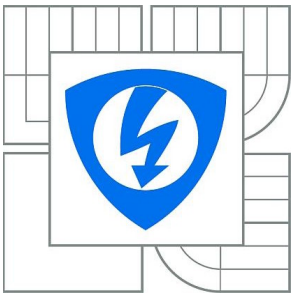


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SYSTÉMY DÁLKOVÉHO MĚŘENÍ V ENERGETICE

SYSTEMS FOR REMOTE MEASUREMENT IN POWER ENGINEERING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

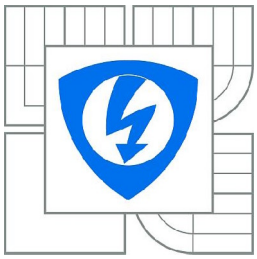
AUTOR PRÁCE — Bc. LUKÁŠ HUDEC

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JIŘÍ MIŠUREC, CSc.

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Lukáš Hudec ***ID:*** 98253
Ročník: 2 ***Akademický rok:*** 2010/2011

NÁZEV TÉMATU:

Systémy dálkového měření v energetice

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou sběru dat z velkého počtu měřicích zařízení umístěných na rozsáhlém území. Zaměřte se především na oblast moderních technologií v energetice typu smart metering a smart grids. Zaměřte se na hierarchický systém sběru dat s koncovými zdroji dat, sběrnými uzly a centrálním uzlem. Rozeberte různé typy předzpracování dat sběrnými uzly či předřazených koncentrátorech. Matematicky popište problematiku přenosu informace od zdroje k centrálnímu uzlu v závislosti na počtu koncových uzlů a organizaci stromu (počtu a propojení sběrných uzlů). Proveďte počítačové simulace ukazující na problémové části komunikace smart technologií.

DOPORUČENÁ LITERATURA:

[1] NOVOTNÝ, V. Integrované sítě. Skriptum VUT Brno, ISBN 80-214-2254-8, ČR, 2002
[2] FUJIMOTO,R.M. Network Simulation (Synthesis Lectures on Communication Networks). Morgan and Claypool Publishers, ISBN-13: 978-1598291100, USA, 2006

Termín zadání: 7.2.2011 ***Termín odevzdání:*** 26.5.2011

Vedoucí práce: doc. Ing. Jiří Mišurec, CSc.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ABSTRAKT

Práce se zabývá problematikou měření a řízení v energetice. Popisuje úvod do oblasti dálkových odečtů, řízení a popisuje současnou situaci v oboru moderních technologií Smart metering a Smart grids. Je zde rozebrána problematika sběrných sítí a shromažďování dat od velkého počtu měřidel na rozsáhlém území. Pro účely přenosu dat jsou popsány technologie GPRS, PLC, DSL,…

Dále jsou v práci uvedeny možnosti zefektivnění komunikace mezi měřidly a sběrnou centrálou. K tomuto je využita oblast hierarchické agregace. Pomocí algoritmu k-means je navržen program pro výpočet počtu koncentrátorů a jejich umístění ve skupině měřidel. Vytvořený program je napsán v programovacím jazyce Java. Obsahuje grafické rozhraní a znázorňuje, jak výpočet probíhá.

Pro ověření výsledků z optimalizačního programu je sestaven simulační model v nástroji OPNET Modeler. Ověřené výsledky jsou popsány v závěru práce a lze z nich odvodit, že použitím optimalizačního programu dochází k zefektivnění komunikace mezi měřidly a sběrnou centrálou.

ABSTRACT

The work deals with the measurement and management in power. Provides an introduction to the problems of remote meter reading, management, and describes the current situation in the field of modern technologies Smart metering and Smart grids. It also analyzed the issue of collection of networks and data collection from a large number of meters over a wide area. For the purpose of data transmission are described GPRS, PLC, DSL, ...

Further, there are given options to streamline communication. This area is used hierarchical aggregation. Using k-means algorithm is a program designed to calculate the number of concentrators and their location in the group of meters. The finished program is written in Java. It has a graphical interface and shows how the calculation is conducted.

To verify the results of the optimization program is given simulation model in OPNET Modeler tool. Audited results are described in the conclusion and can deduce that using the optimization program is to streamline communications.

KLÍČOVÁ SLOVA

HDO, AMR, AMI, AMM, chytré sítě, teorie grafů, graf, strom, hierarchická agregace, sběrná síť, koncentrace dat, sběr dat v energetice, Dijkstrův algoritmus, k-means, optimalizace sběru dat

KEYWORDS

Smart metering, Smart grid, MRC, AMR, AMI, AMM, graph theory, graph, tree, hierarchical aggregation, collection networks, data concentration, energy, Dijkstra's algorithm, k-means, optimization of data collection

.....

HUDEC, L. *Systémy dálkového měření v energetice*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 65 s. Vedoucí diplomové práce doc. Ing. Jiří Mišurec, CSc..

.....

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma Systémy dálkového měření v energetice jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne: 17.05.2011 .....

podpis autora

.....

PODĚKOVÁNÍ

Děkuji vedoucímu práce doc. Ing. Jiřímu Mišurcovi, CSc. za odborné vedení, cenné rady a konzultace při zpracování diplomové práce.

V Brně dne: 19.05.2011 .....

podpis autora

.....

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V současné době se neustále zvyšují požadavky na množství a kvalitu energií, které lidstvo vyžaduje. S rostoucí spotřebou je nutné zajistit její rychlou, bezporuchovou a stálou distribuci k zákazníkovi. Stále se zvyšující nároky nebude možné do budoucna uspokojit se stávajícím systémem distribuce energií a právě proto jsou vyvíjeny a testovány nejrůznější platformy pro snadnější a „chytřejší“ rozdělení a šíření energií.

V této práci budou uvedeny nejnovější poznatky z oblastí „smart“, tedy chytrých, technologií. Budou zde rozebrány jednotlivé druhy a pojmy jako jsou Smart metering, Smart grids, atd. Jejich nasazení je nejpatrnější v oblasti distribuce elektrické energie, ale mohou být použity i pro měření a regulaci jiných veličin, například plynu, vody, tepla. Práce se zabývá především implementací v oblasti měření a regulace elektrické energie. Základní pojmy z této oblasti jsou popsány v úvodu práce.

V části práce pojednávající o pilotních projektech jsou uvedeny průběžné výsledky, kterých je v současné době dosaženo. Jsou zde popsány projekty velkých firem. Například jde o firmy EON, ČEZ a další.

V kapitole Architektura chytrých systémů je popsán hierarchický systém sběru a přenosu dat. Je zde uvedeno uspořádání jednotlivých „chytrých“ prvků a jejich vzájemné propojení například pomocí: GPRS, GSM, PLC, GPRS, atd.

Dále jsou pak uvedeny možnosti popisu sběrných sítí z matematického hlediska. Je uveden postup hledání optimální cesty v sítích typu MESH. Podstatnou část tvoří popis hierarchické agregace a její uplatnění ve sběrných sítích.

Vlastní implementace hierarchické agregace ve sběrné síti společně s návrhem programu pro tyto výpočty je uvedena v kapitole 4. Sestavený program dokáže vypočítat optimální počet koncentrátorů pro danou skupinu měřidel a dokáže určit jejich optimální polohu v dané sběrné síti. Je zde také rozebrána funkce programu a způsob jeho ovládání.

Závěrem práce je ověření činnosti optimalizačního algoritmu v simulačním prostředí OPNET Modeler. Je zde naznačen postup jak provádět simulaci a jakých výsledků bylo dosaženo.

Rozbor a specifikace zadání

Po dohodě s vedoucím práce je zadání zpracováno v následujícím rozsahu:

- Rozbor základních pojmů z oblasti chytrého měření (Smart metering, Smart grids, systémy AMR, AMM, AMI). Jejich možnosti, vývoj a vliv při nasazení ve skutečných podmínkách.
- Specifikace průběžných výsledků nasazení chytrých technologií v pilotních projektech energetických firem.
- Obecná architektura systémů pro sběr dat, Smart metering.
- Krátký popis přenosových technologií vhodných pro nasazení do sběrných sítí. Rozbor pojmů GSM, GPRS, Internet, PLC, RF.
- Naznačení problematiky matematického modelu sběrných sítí. Možnosti popisu sběrné sítě z matematického pohledu.
- Návrh algoritmu pro optimalizaci rozsáhlých sběrných sítí v závislosti na umístění sběrných koncentrátorů a měřidel.
- V simulačním nástroji naznačit a ověřit, zda se projeví vlastnosti algoritmu v síti po provedení optimalizace.

1 Úvod do problematiky chytrého měření

1.1 Co je Smart metering

Smart metering je systém pro dálkovou obousměrnou komunikaci mezi měřidlem a datovou centrálou. Pod pojmem měřidlo si můžeme představit domovní měřič elektrické energie (měřič spotřeby), měřič plynu, vody a třeba i měřič spotřeby tepla.

Na druhé straně komunikace je datová centrála, která slouží pro sběr změřených dat. Datová centrála sesbíraná data uchovává a vytváří statistiky odběru, technických ztrát ale hlavně podklady pro fakturaci.

1.2 Základní pojmy

Pro lepší orientaci v problematice budou rozebrány základní pojmy z oblasti měření a dálkového ovládání v energetice. Pro názornost je uvedena Tab. 1 s přehledem jednotlivých technologií a směrem jejich komunikace.

Tab. 1: Přehled technologií

<table>
<tr><td rowspan="2">Směr komunikace</td><td>Zákazník
↓
Dodavatel</td><td>Elektro-mechanické a elektronické elektroměry (manuální odečet)</td><td>AMR elektroměry (vzdálený odečet)</td><td rowspan="2">AMM elektroměry (vzdálený odečet a ovládání)</td><td rowspan="2">Smart metering (vzdálený odečet, ovládání a automatické řízení)</td></tr>
<tr><td>Dodavatel
↓
Zákazník</td><td colspan="2">Systém hromadného dálkového ovládání HDO</td></tr>
</table>

1.2.1 Systém AMR (Automatic Meter Reading)

Systém AMR (Automatic Meter Reading) slouží pro odečet dat o spotřebě energie u zákazníka. Data jsou odesílány do datové centrály, kde se zpracovávají. Jde v podstatě pouze o dálkové automatické odečty. Výhodou těchto systémů je především zajištění efektivních odečtů, při kterých není potřeba, aby fyzická osoba navštěvovala každé odběrné místo a prováděla odečet manuálně.

1.2.2 Systém AMM (Automatic Meter Management)

Dalším pojmem, se kterým se u chytrého měření setkáme je AMM (Automatic Meter Management, Advanced Metering Management). Takto označené systémy jsou charakteristické obousměrnou komunikací, čímž se rozšíří možnosti AMR systému o další funkce, jako například řízení tarifu, dálkové odpojení odběrného místa (pro ochranu před neplatiči), nebo třeba dálkové nastavení maximálního vstupního příkonu (tzv.: Demand-Side Management). Tento systém by měl být schopen nahradit hojně využívaný systém hromadného dálkového ovládání (HDO), který je v současné době používán na území České republiky.

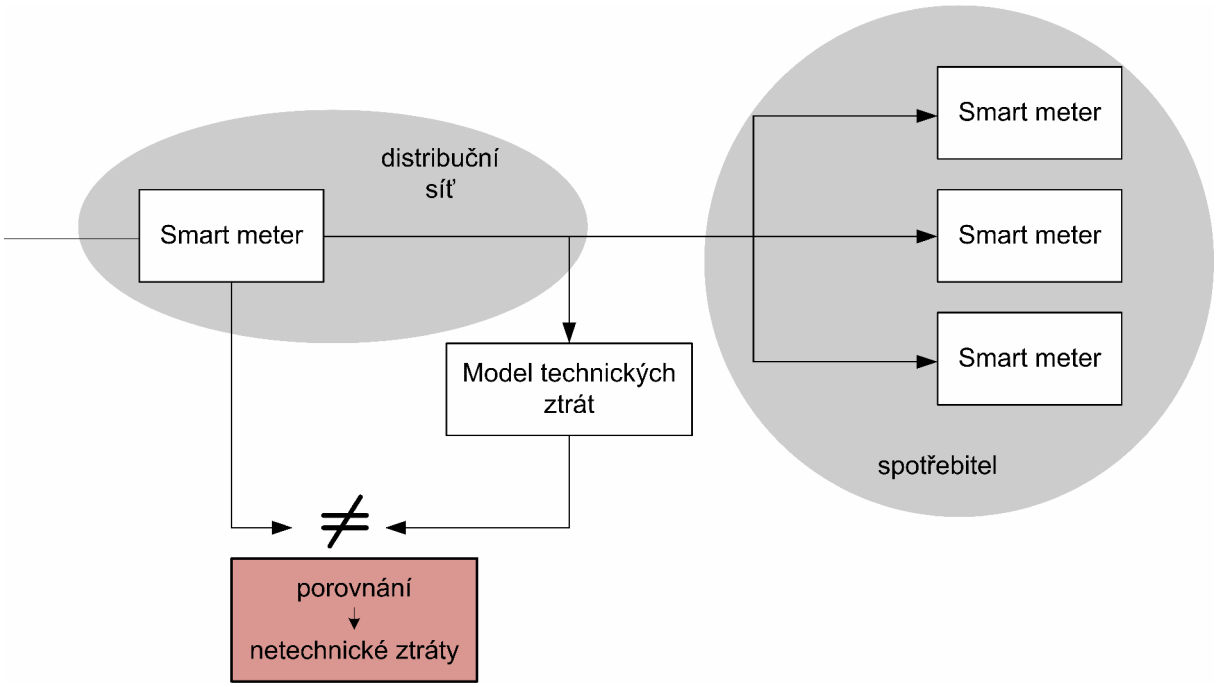
1.2.3 Systém AMI (Advanced Metering Infrastructure)

V blízké budoucnosti se předpokládá zavedení systému AMI (Advanced Metering Infrastructure) do praxe. AMI předpokládá rozšíření AMM o funkce týkající se ovládání konkrétních spotřebičů u zákazníka. Při komunikaci jsou tedy kladeny vysoké požadavky na přenos velkých objemů dat s co nejmenším zpožděním a téměř v reálném čase.

1.2.4 Systém Smart metering

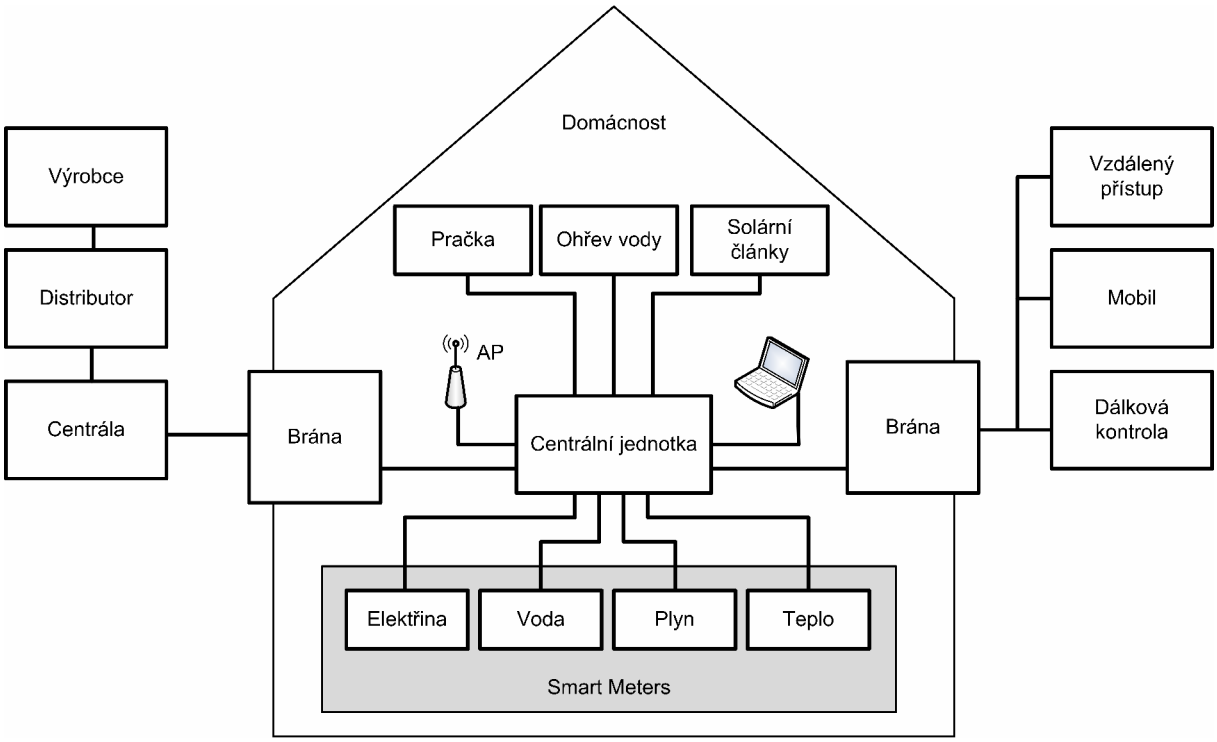
Pojem Smart metering uceluje vývojové fáze systémů AMR, AMM a AMI. Tento koncept umožňuje především vyhodnocování odečtených dat a následné řízení na základě těchto dat. Pokud bude systém rozšířen na větší území, dá se předpokládat, že plně nahradí současně používaný systém hromadného dálkového ovládání (zkratka HDO).

Další výhodou je, že systém umožňuje častější odečítání spotřeby koncových uživatelů (téměř v reálném čase). Tím se dají vytvořit přesné grafy vytížení sítě, ale co je ještě důležitější, je možné odhalit zdroje netechnických ztrát (černé odběratele). To vyžaduje, aby měřidla nebyla instalována pouze u koncových zákazníků, ale i na přenosové trase. Jednoduchým výpočtem z naměřených dat lze zjistit velikost a případně i přibližné umístění netechnických ztrát. Princip zachycuje Obr. 1.



Obr. 1: Určení netechnických ztrát pomocí chytrého měření

Do budoucna se předpokládá, že při nasazení chytrých měřidel bude mít zákazník možnost nejen zjistit informace o vlastní spotřebě, ale také o aktuálním tarifu a aktuální ceně za jednotku energie. Právě tímto zapojením zákazníka do energetické struktury dojde k rozložení zátěže. Energetický dodavatel bude schopen podle aktuální situace v síti nastavit tarif (a cenu) a zákazníci si sami podle ceny rozhodnou, zda zapnou spotřebiče nyní anebo se spuštěním vyčkají na nižší tarif (tedy i nižší cenu) a tím pádem ušetří. Se zavedením systému se počítá v následujících deseti až patnácti letech. Uplatnění bude ovlivněno legislativou a směrnicemi Evropské unie. V současné době se technologie testuje v pilotních projektech několika firem. Projekty jsou popsány v kapitole 1.3.
Jak bude technologie Smart metering s podporou AMI v budoucnu pracovat ukazuje Obr. 2.



Obr. 2: Budoucí pohled na Smart metering s podporou AMI

1.2.5 Systém Smart grids

Posledním pojmem pro úplnost je koncept Smart grids (chytré sítě). Jak zaznělo na konferenci Smart metering: ČEZ – Futuremotion (Jiří Feist) [1] definice této platformy zní: *„Smart grids neboli "inteligentní sítě" jsou spolehlivé, automatizované a efektivně řízené distribuční sítě 21. století. Principem je interaktivní obousměrná komunikace mezi výrobními zdroji a zákazníky o aktuálních možnostech výroby a spotřeby energie.“*

Základem je technologie Smart metering, která je ovšem rozšířena o měřidla ve vlastní síti. Díky tomu je možné monitorovat a analyzovat aktuální stav sítě. To je vhodné především v energetice, kde může docházet ke kolísání napětí vlivem aktuální zátěže sítě. Díky technologii Smart grids by mělo být možné takto vzniklé zátěžové špičky detekovat a dynamicky měnit tarify přímo u spotřebitelů. Tím by docházelo k rozložení zátěže v síti a odlehčení sítě. Základní charakteristika chytrých sítí:

- automatizace a monitorování distribuční soustavy, různé způsoby výroby energií
- bilanční vyrovnání mezi spotřebovanou a vyrobenou energií, integrace zákazníků

Výhody zavádění technologie Smart grids se dají zkoumat z různých pohledů. Základní pohled ukazuje Tab. 2 [1], ve které jsou uvedeny tři zúčastněné strany. První je dodavatel, následuje spotřebitel a posledním aspektem je globální pohled společnosti.

Tab. 2: Výhody Smart grids

Distributor	Spotřebitel	Společnost
▪ úspora nákladů ▪ prodloužení životnosti distribuční sítě ▪ lepší vytíženost distribuční sítě ▪ snadná lokace poruch ▪ informace o ztrátách	▪ zvýšení kvality dodávky elektrické energie ▪ omezení výpadků ▪ informovanost o aktuální ceně ▪ motivace k úsporám	▪ pozitivní vliv na životní prostředí ▪ zavádění nových technologií

1.2.6 Systém HDO (Hromadné Dálkové Ovládání)

V současné době je v České a Slovenské republice hojně využíván systém hromadného dálkového ovládání (zkratka HDO). Poprvé se začal využívat na počátku 20. století. Jde o systém založený na jednosměrné komunikaci směrem od dodavatele elektrické energie ke spotřebiteli. V podstatě se jedná o dálkovou změnu tarifu (známý též jako noční a denní proud) a s tím spojené připojení nebo odpojení určitých spotřebičů. Připojovanými spotřebiči bývají nejčastěji systémy na topení a ohřev užitkové vody. Důvodem zavedení takovéhoto dálkového ovládání je předejít výkonovým špičkám v energetické síti. Elektřinu totiž není možné skladovat, a tedy vždy musí existovat rovnováha mezi výrobou a spotřebou.

Během dne se mění diagram zatížení sítě, který má každý den podobný průběh. Úkolem HDO je připojit k síti výkonové spotřebiče v dobu, kdy je elektřiny nadbytek a tím zrovnoměrnit zatížení sítě [2]. Díky takovému řízení není nutné posilovat výrobní a přenosové kapacity a také není potřeba odstavovat výrobny elektřiny v noci, kdy je spotřeba elektrické energie nižší.

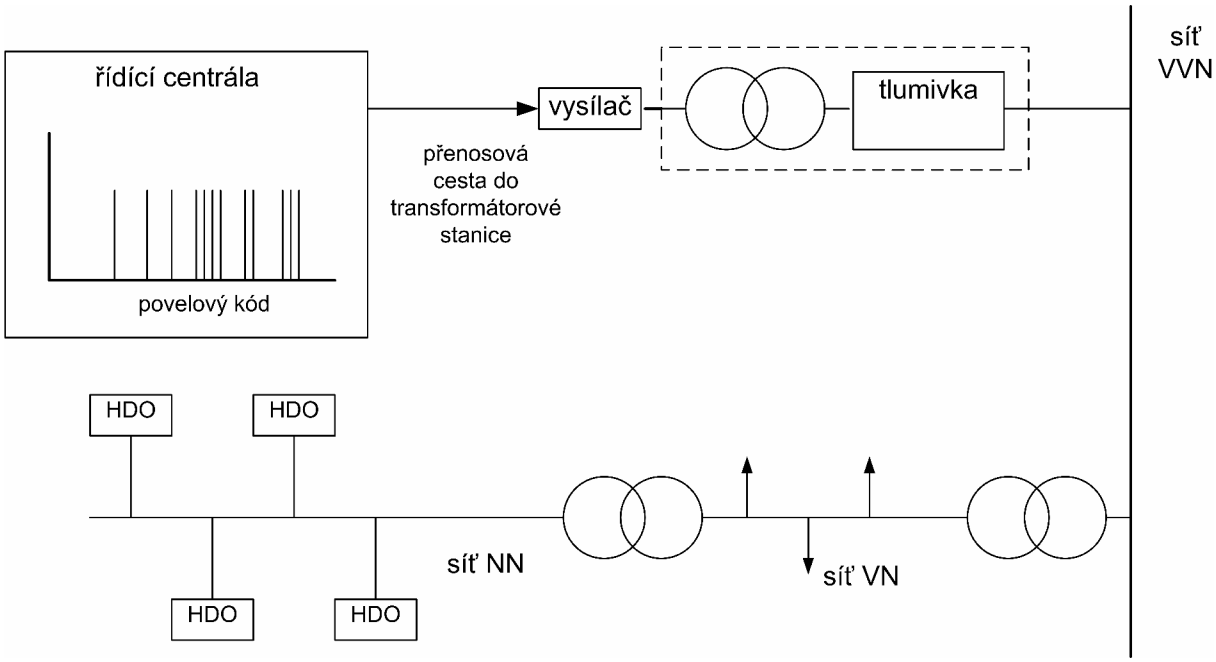
Jako ve veškeré technice je možné i u HDO pozorovat určitý vývoj. První typy ovládání byly instalovány přímo u odběratele. Zařízení nikterak nekomunikovala s okolím. Jednalo se v postatě o elektromechanické hodiny, které v předem nastavený čas automaticky spínaly určité spotřebiče. Hodiny ovládaly relé a toto relé spouštělo stykač, kterým se uváděl do chodu požadovaný spotřebič. Současně se na dvoutarifním elektroměru změnil tarif na nízký (tedy levnější). Nevýhody tohoto systému plynou z jeho statického provedení. Nebylo možné dynamicky měnit spínání hodin na dálku podle aktuální potřeby. Veškeré změny musel provádět pracovník dodavatele přímo u spotřebitele mechanickou změnou parametru spínacích hodin.

Následníkem elektromechanických hodin je právě systém HDO. Systém využívá silová vedení pro přenos informací směrem od dodavatele ke spotřebiteli. Dodavatel má tedy možnost dynamicky připojovat nebo odpojovat zařízení v určitých domácnostech a tím regulovat aktuální odběr a zatížení sítě.

Princip činnosti: Datová centrála u dodavatele elektrické energie vyšle řídicí signál. Na signál zareagují pouze přijímače, pro které je signál určen. Tyto přijímače změní tarif u elektroměru a připojí nebo odpojí ovládané spotřebiče.

Aby nedocházelo k rušení signálů různých dodavatelů elektrické energie, jsou jednotlivé kmitočty pro dálkové ovládání rozděleny jednotlivým distributorům podle lokálního umístění.

Přenos řídicích povelů je realizován impulzním kódem s kmitočtem řádově do jednotek kHz. Na území České republiky se převážně využívá frekvence 216,66 Hz. Další kmitočty, které se zde používají, jsou například: 183,33 Hz, 283,33 Hz, 760 Hz, 1 060 Hz [16]. Takto vytvořený kód je superponován na základní kmitočet energetické sítě, tedy 50 Hz. Vysílá se do každé fáze vysokého napětí a přes transformátory se dostává až do nízkonapěťových sítí 400 V nebo 230 V [2]. Princip přenosu řídicích informací HDO je ukázán na Obr. 3.



Obr. 3: Princip šíření řídicích signálů v HDO

1.3 Pilotní projekty

Jak už z hlediska vidiny budoucích výhod, zisku anebo jen z důvodu legislativních, dochází v současné době k realizaci několika pilotních projektů v tuzemských i zahraničních firmách. V této kapitole bude uvedeno několik ukázkových projektů s jejich charakteristikou a aktuálním vývojovým stavem.

1.3.1 EON

Rozsah pilotního projektu Smart metering ve společnosti EON ukazuje Tab. 3.

Tab. 3: Pilotní projekt EON

Obec	Instalované elektroměry	Datové koncentrátory	Převodové elektroměry
Křižanovice	67	1	1
Rousínov	1896	15	20
Ivanovice n/H	1186	9	14
Luleč	648	6	0
Vyškov	9	0	0
Celkem	3806	31	43

Pozn.: převodový elektroměr je elektroměr, který je zapojen přes měřicí transformátor. Poměr mezi primárními a sekundárními veličinami vyjadřuje převodový poměr. Převodový elektroměr může být cejchován v sekundárních jednotkách, anebo přímo v primárních hodnotách. [8]

1.3.2 Skupina ČEZ

Základní informace o pilotním projektu skupiny ČEZ [1]:
Projekt je založen na testování provozuschopnosti technologie Smart metering s podporou AMM.

Charakteristické vlastnosti:
Lokalita: Vrchlabí (2 600 měřidel), Pardubice (24 632 měřidel), Jeřmanice (11 335 měřidel)
Realizace projektu v období: 2010 – 2013
Cíle: Cílem je testování 40 tisíc měřidel, jejich vzájemné komunikace a bezpečnosti.

Druhým projektem je realizace „chytrého regionu“ v lokalitě Vrchlabí.
Účelem je testování technologie Smart metering na území středně velkého města.
Předpokládá se využití obnovitelných zdrojů energie (vítr, slunce,…) a integrace těchto zdrojů do testovaného regionu.
Charakteristické vlastnosti:
Realizace projektu v období: 2010 – 2015
Rozsah realizace:

- Městská kabelová VN síť 10 kV (cca 12 km)
- NN síť – kabelová i vrchní vedení (cca 66 km)
- Asi 4 900 odběrných míst

1.3.3 Skupina RWE

Spíše výhled do budoucnosti. Uvažuje i zapojení solárních kolektorů a do jisté míry i nabíjení elektromobilů. Jde o jediný projekt z těchto uvedených, který se dá po právu označit pojmem Chytrá síť.

Výsledky projektu se očekávají v polovině roku 2011.

1.3.4 Západoslovenská energetika

Jedná se o největší distribuční společnost na Slovensku (40% podíl na trhu), je členem skupiny EON. Do projektu inteligentního měření je zapojeno:

- 5000 velkoodběratelů, nasazeny elektroměry + GPRS (General Packet Radio System) modemy, denní přenos naměřených dat (15 minutové profily), měsíční odečty.
- 2200 podnikatelských subjektů, elektroměry + GPRS přenos
- 425 domácností
- 4500 elektroměrů + GPRS modemů pro centrální měření, denní přenos dat (15 minutové intervaly) měsíční odečty.

Pilotní projekty Smart metering s podporou AMM:

- s radiofrekvenční komunikací jsou instalovány na 365 přípojných míst (Leopoldov – 249, Stupava – 116). Pro komunikaci elektroměr – koncentrátor využita radiová komunikace. Přenos GPRS využit pro komunikaci koncentrátor – centrála.
- PLC komunikace je využita u 54 přípojných míst v lokalitě (Bratislava, Rusovce). PLC využívá pásmo A (9-95 kHz)

1.4 Problémy spojené se sběrem dat

Ze samotného principu fungování technologií chytrého měření vyplývá potřeba přenášet data mezi měřidlem instalovaným u zákazníka a datovou centrálou. Datová centrála zpracovává přijímaná data a alarmy od měřidel a odesílá řídicí data k měřidlům.

Problematika přenosu dat a architektura systému Smart metering je rozepsána v kapitole 2.

Pokud bychom se na datové přenosy dívali z pohledu jednotlivých účastníků, asi by nás problém s přenosem dat ani nenapadl. Každému je jasné, že měřidlo (např. systém AMM) odesílající data kupříkladu jednou za týden nezpůsobí velké požadavky na přenosové kapacity transportních tras. Ani pokud bychom sledovali výsledky pilotních projektů, zamyšlení nad

datovým provozem také není podstatné. Vždy se totiž jedná o instalaci několika (řádově tisíce) chytrých měřidel, což je v porovnání s budoucím využitím nesrovnatelně malý počet.

Pokud se ovšem podíváme na budoucí využití technologií v plném rozsahu a pravděpodobně i na rozsáhlých územích (například na celém území České republiky), můžeme se setkat s nepříjemným nárůstem datového provozu.

Data přenášená pomocí AMM systémů (tedy Chytrých měřidel) lze rozdělit na tři základní druhy:

Prvním typem jsou data nesoucí informace o spotřebě daného přípojného místa. Směr toku těchto dat je od zákazníka k centrále. Perioda odesílání těchto dat nemusí být příliš častá. Můžeme uvažovat 1x za měsíc.

Druhým typem jsou nečekané události, tzv. alarmy. Tento typ zpráv je odesílán nejčastěji ve směru od měřidla k datové centrále. Přenášené informace slouží pro oznámení nestandardního chování měřicích zařízení. Příkladem může být odeslání informací o neoprávněné manipulaci s ochranným krytem měřicího zařízení, systémová nebo jiná chyba. I když by se mohlo zdát, že odesílání tohoto typu dat nebude příliš časté, z výsledků pilotních projektů je zřejmé, že jedno takto připojené měřidlo generuje zprávu v průměru jednou za týden. V některých případech docházelo ke generování událostí v důsledku špatně provedeného ochranného kontaktu, který vlivem změn teploty neustále generoval falešné události.

Posledním, tedy třetím typem dat, jdou data ovládací. Systém chytrého měření předpokládá celkové nahrazení doposud hojně využívaného systému HDO (viz. Kap. 1.2.6). V budoucnu by mělo být možné ovládat spotřebiče přímo u spotřebitele. V současné době tuto funkci umožňuje HDO. Distributor by tak mohl přímo připojovat anebo odpojovat spotřebiče u spotřebitele v závislosti na aktuální výrobě a spotřebě elektrické energie. Další funkcí by měla být možnost informovat zákazníka o aktuální ceně za elektrickou energii.

Pokud bychom uvažovali 40 000 připojených měřidel, v číslech bychom potom mohli přemýšlet takto:

Tab. 4: Výpočet přenesených dat

Typ zprávy:	informace o spotřebě	alarmy a upozornění	dálkové ovládání
Perioda odesílání:	12x za rok	52x za rok	min. 2x za den
Objem dat:	10 kB	1 kB	1 kB
Data za rok:	4,8 GB	2,1 GB	orientačně 14,6 GB

Z Tab. 4 jsou již patrné objemy dat, které se pohybují v řádech GB, a to je stále počítáno jen se 40 tis. měřicími jednotkami. Do budoucna potom uvažujeme takto:

Počet domácností v České republice se pohybuje okolo 4 366 300 (prosinec 2010). Podle směrnice 2009/72/ES přílohy I, bodu 2 se požaduje, že do roku 2020 musí být inteligentními měřicími systémy vybaveno alespoň 80% spotřebitelů, za předpokladu pozitivně vyhodnocených projektů na zavádění inteligentních systémů. To by znamenalo, že počet připojených měřidel by se pohyboval okolo 3,5 milionu. A k tomu je nutné také připočítat firemní objekty. V takovém případě se datový tok značně zvýší. Navíc s rostoucí dobou roste i vyspělost chytrých systémů, které vyžadují stále větší objemy přenášených dat. Zákazníkovi se zpřístupňují další funkce, jako např. dálková správa a sledování aktuální spotřeby. Tím opět roste datový přenos. Navíc pro „pravé“ chytré sítě (Smart grids) a města (Smart cities) je nutné přenášet údaje o aktuální spotřebě téměř v reálném čase. Nestačí pouze jednou za měsíc odečíst spotřebu dané domácnosti. Stejně tak i řídicí informace je potřeba přenášet několikrát denně. Další nárůst datové komunikace se dá očekávat v případě aplikace chytrého měření do dalších oborů jako například plynárenství, vodárenství, atd.

Pro dílčí snížení datových toků můžeme data z části zpracovat v koncentrátorech. Jelikož ovšem nejsou přenášena redundantní (nadbytečná) data, předzpracování je možné jen částečné.

Například změřená data se musí přenést kompletní. Jisté předzpracování by bylo možné u dat typu událostí a alarmů, ale ani zde není sumarizace žádoucí hlavně z důvodu vnášeného zpoždění.

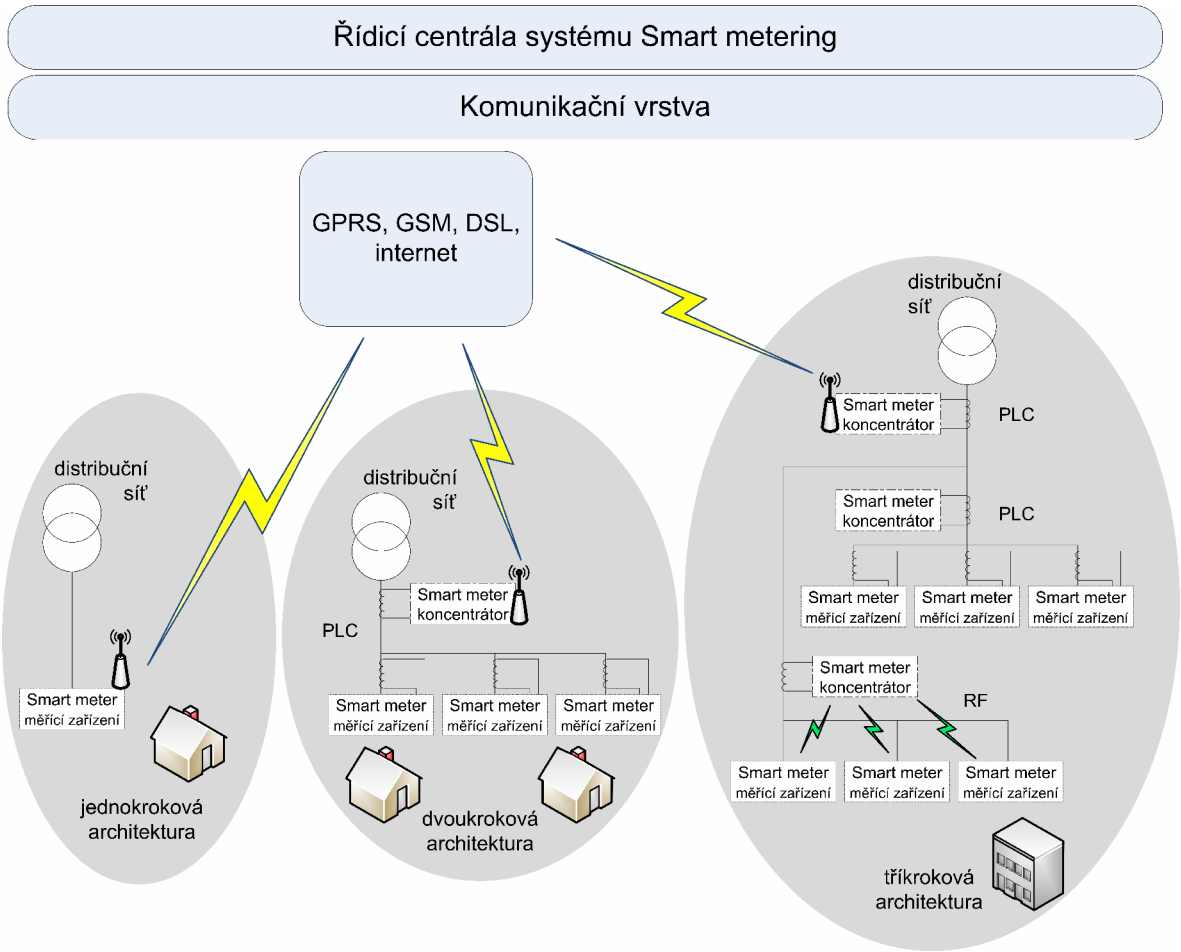
Poslední možností předzpracování jsou ovládací data. Nejefektivnější přenos by byl zajisté takový, který je podobný multicastu. Řídicí centrála by vygenerovala určitý povel směrem ke koncentrátoru a koncentrátor by sám zajistil rozdělení informace buď pro další koncentrátory anebo přímo pro koncová měřidla. Tímto by došlo k ušetření přenosových kapacit hlavně na dálkových (drahých) přenosech realizovaných nejčastěji pomocí GRPS.

2 Architektura systémů Smart metering

V kapitole bude rozebrán systém chytrého měření především z pohledu datových toků a architektury. Budou zde také uvedeny přenosové technologie, které je možné využít pro přenos dat.

2.1 Obecné uspořádání

Jak již bylo řečeno v kapitole 1.3 v současné době se testuje technologie chytrého měření v několika pilotních projektech. Obecně se dá říct, že ve všech případech je použita architektura uvedená na Obr. 4. V podstatě jde o jedno až tří stupňovou architekturu dálkového ovládání a měření [3].



Obr. 4: Architektura systému Smart metering

Z Obr. 4 je patrné, že jednotlivé architektury jsou vhodné pro různě rozsáhlé nasazení, pro různou koncentraci účastníků a také pro různou vzdálenost mezi účastníky.

Jednokroková architektura se nejvíce hodí pro případ osamělého domu, který je od ostatních chytrých měřidel vzdálen více než jednotky km. Smart meter je ovládán přímo, bez využití koncentrátorů, například prostřednictvím technologií GSM (Global System for Mobile Communications), GPRS (General Packet Radio System), DSL (Digital Subscriber Line). Tyto technologie jsou vhodné pro dálkový přenos dat do řídicí centrály. Směrem od uživatele jsou odesílána změřená data a směrem od řídicí centrály k uživateli jsou odesílány ovládací povely.

Dvoukroková architektura je vhodná pro nasazení do středně velkých oblastí. Technologie se vyznačuje použitím koncentrátorů, které shromažďují změřená a ovládací data. Koncentrátory mohou komunikovat s měřidly umístěnými u spotřebitelů pomocí technologie PLC (Power Line Comunication), popřípadě pomocí radiové komunikace RF (Radio Frequency). V dalším kroku, tedy od koncentrátoru do řídicí centrály, je využita některá z technologií GSM, GPRS, DSL,…

Poslední možností komunikace je třístupňová architektura. Ta je vhodná pro nasazení do oblastí s mnoha měřidly. Vyznačuje se použitím několika koncentrátorů. Jak i název napovídá, data se přeposílají ve třech krocích. Ve směru od uživatele k datové centrále jsou data v první fázi změřena pomocí chytrého měřidla. Dále, např. pomocí technologie RF, jsou data odeslána do druhého stupně, tedy do koncentrátoru. Následuje transport dat některou z technologií RF nebo PLC do dalšího kroku, tedy k následujícímu koncentrátoru. Ten je již vybaven rozhraním pro připojení některé z technologií GPRS, GSM, DSL,… Tyto technologie následně odesílají data na velkou vzdálenost do datové centrály.

Náklady na přenos měřených a řídicích dat vznikají především při použití technologií určených pro přenos dat na velkou vzdálenost (GPRS, GSM, DSL,…). Z tohoto důvodu se využívají koncentrátory, které mají za úkol shromáždit data od více uživatelů a následně je odeslat do centrály. V opačném směru do koncentrátoru přichází řídicí a informační data a ty jsou následně odesílány buď do dalšího koncentrátoru v hierarchii o úroveň níž anebo přímo do zařízení Smart meter. Z této teorie je jasné, že systémy s využitím jednokrokové architektury budou jednoduší a levnější na vybudování (není třeba použít koncentrátory), ale z hlediska nákladů na datové přenosy budou náklady vyšší. Druhým extrémem je tříkroková architektura, která je výhodná z hlediska úspor nákladů za datové přenosy, ale na druhou stranu jsou náklady na její vybudování mnohem vyšší.

2.2 Popis přenosových technologií

Existuje mnoho technologií, které je možné použít pro přenos informací řídicích a ovládacích mezi datovou centrálou a spotřebitelem. V této kapitole budou zmíněny a rozebrány především technologie, které jsou pro systémy Smart metering vhodné a v první řadě ty, které se v současné době používají v testovacích provozech a pilotních projektech zúčastněných firem.

Přehled technologií vychází z hierarchického modelu z Obr. 4. Přenosové technologie budou popsány v pořadí odpovídající směru komunikace od datové centrály ke koncovému uživateli (spotřebiteli).

2.2.1 GSM (Global System for Mobile communications)

Zkratka GSM (Global System for Mobile communications) označuje technologie pro mobilní komunikaci a poskytování telekomunikačních služeb. Vývoj začal v roce 1982.

Služby podporované GSM se neustále rozrůstají a zlepšují. Mezi ně patří např.: přenos hlasu, dat, faxu, přenos krátkých textových zpráv SMS (Short Message Service), připojení k internetu atd.

Další skupina doplňkových služeb rozšiřuje funkčnost a pohodní účastníka. Jsou to služby: identifikace volajícího, hlasová schránka, přidržení hovoru, atd. Zjednodušená struktura GSM sítě je na Obr. 5.

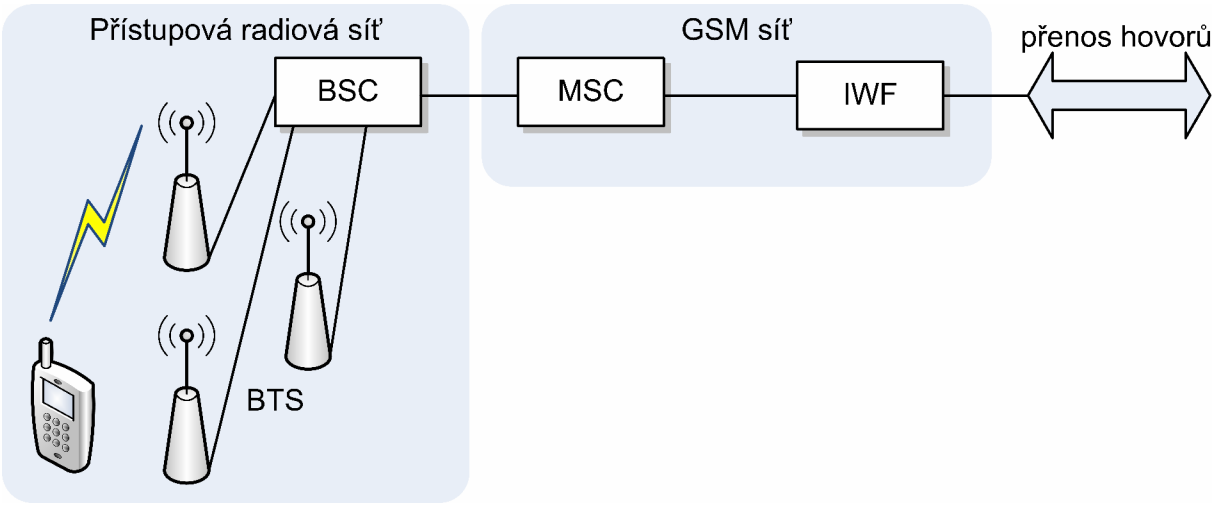
Vývoj systému GSM:

Podle vývoje systému GSM se rozlišují jednotlivé fáze [9]:

- GSM Phase 1: Kromě základních služeb jsou do systému přidány:
  - přesměrování hovorů

- blokování hovorů
- přidržení hovorů
- hlasová schránka

- GSM Phase 2:
  - konferenční hovory
  - přenos tarifních informací
  - identifikace volajícího
  - datové a faxové služby typu CSD (Circuit Switched Data)
- GSM Phase 2+: Jsou přidány další datové služby
  - HSCSD (High Speed Circuit Switched Data) – obvodově spojovaná technologie
  - GPRS (General Packet Radio System) – paketově spojová technologie (podrobně rozepsána v kapitole 2.2.2.)
  - EDGE (Enhanced Data Rates for GSM Evolution) – využívá osmi stavovou modulaci
- GSM Phase 3:
  - systém UMTS (Universal Mobile Telecommunication System)



Obr. 5: Architektura sítě GSM

Mobilní terminál komunikuje se základnovou stanicí BTS (Base Transceiver Station). Základnová stanice je řízena kontrolerem BSC (Base Station Controller). Hovor je dále předáván do mobilní ústředny MSC (Mobile Switching Centre) a v případě, že směřuje do jiné sítě je odeslán pomocí IWF (InterWorking Function) do sítě přenášející hovory.

Datový přenos je zde realizován především pomocí vytáčeného spojení HSCSD (High Speed Circuit Switched Data). Funkce HSCSD je taková, že pro přenos je alokován více než jeden timeslot po celou dobu spojení. Tím pádem je dosaženo vyšší přenosové rychlosti.

2.2.2 GPRS (General Packet Radio System)

Technologie GPRS je rozšířením sítě GSM o podporu rychlejšího přenosu dat a to až na teoretickou rychlost 171,2 kbit/s. Technologie je paketově orientována a pracuje na protokolu IP (Internet Protocol). 171,2 kbit/s je ovšem pouze teoretická přenosová rychlost, která by vyžadovala alokaci všech osmi timeslotů v komunikačním kanále. Ve skutečnosti se dosahuje rychlosti okolo 43 kbit/s. Tato rychlost je dána operátorem, který ve většině případů nedovolí obsadit více než tři timesloty současně. Jeden timeslot dokáže vytvořit přenosovou rychlost 14,4 kbit/s. Tím, že alokujeme současně tři timesloty vytvoříme přenos ve směru download

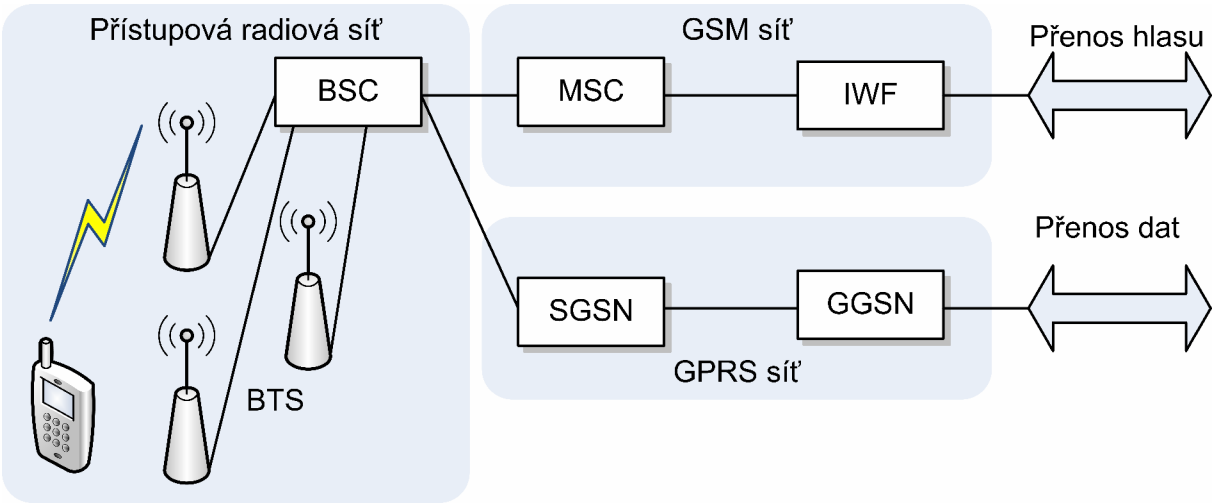
43,2 kbit/s. Současně je možné využívat i provoz upload s rychlostí 14,4 kbit/s. Poměr mezi rychlostmi uploadu a downloadu je dán použitým profilem, který udává kolik timeslotů bude použito pro upload a kolik pro download.

S dalším vývojem GPRS souvisí zavedení čtyř kódovacích schémat (Coding Scheme). Jak tyto kódovací schémata ovlivňují přenosovou rychlost ukazuje Tab. 5 [9].

Tab. 5: Přenosové rychlosti GPRS

kbit/s	Počet timeslotů							
Kódovací schéma	1	2	3	4	5	6	7	8
CS1	9,20	18,40	27,60	36,80	46,00	55,22	64,40	73,60
CS2	13,55	27,10	40,65	54,20	67,75	81,30	94,85	108,40
CS3	15,75	31,50	47,25	63,00	78,75	94,50	110,25	126,00
CS4	21,55	43,10	64,65	86,2	107,75	129,30	150,85	172,40

Se zaváděním GPRS do sítě GSM bylo nutné zvýšení počtu frekvencí v radiovém rozhraní BTS, posílení přenosových kapacit mezi BTS a PCU (Packet Control Unit) nahrazuje BSC, SSGN (Serving GPRS Support Node) a GGSN (Gateway GPRS Support Node). Poslední tři jmenované zajišťují směrování paketového provozu uvnitř sítě operátora. Architekturu GPRS a odlišnosti od GSM ukazuje Obr. 6.



Obr. 6: Architektura systému GPRS

Na rozdíl od GSM obsahuje GPRS samostatnou podsíť, která slouží pro přenos paketového provozu. SGSN má za úkol směrovat data uvnitř sítě daného operátora. Prvek GGSN je v síti vždy jeden a slouží pro posílání datového provozu do jiných sítí (například do sítě Internet).

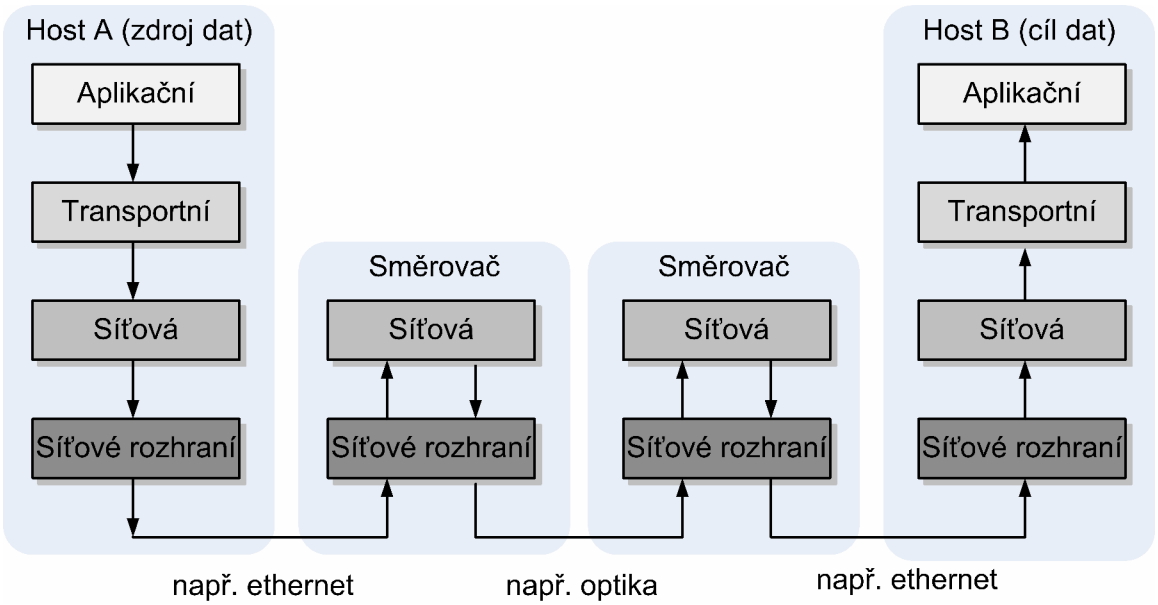
2.2.3 Internet

V současné době je stále více rozšířeno připojení domácností k internetu. Z hlediska ceny za přenos dat se Internet jeví jako levný, protože ve většině případů není nutné platit za přenesená data ani za čas připojení. Navíc mnoho domácností již má internetovou přípojku zřízenou.

Internet je celosvětová počítačová síť, ve které mezi sebou počítače komunikují pomocí protokolů označovaných TCP/IP (Transmission Control Protocol/Internet Protocol). Jde v podstatě o propojení jednotlivých podsítí dohromady.

Každá stanice v síti vyžaduje IP (Internet Protocol) adresu. Pomocí ní je stanice v síti identifikována. Provoz je směrován pomocí směrovačů (router).

Architektura TCP/IP je členěna do čtyř vrstev, které ukazuje levá část Obr. 7 [10].

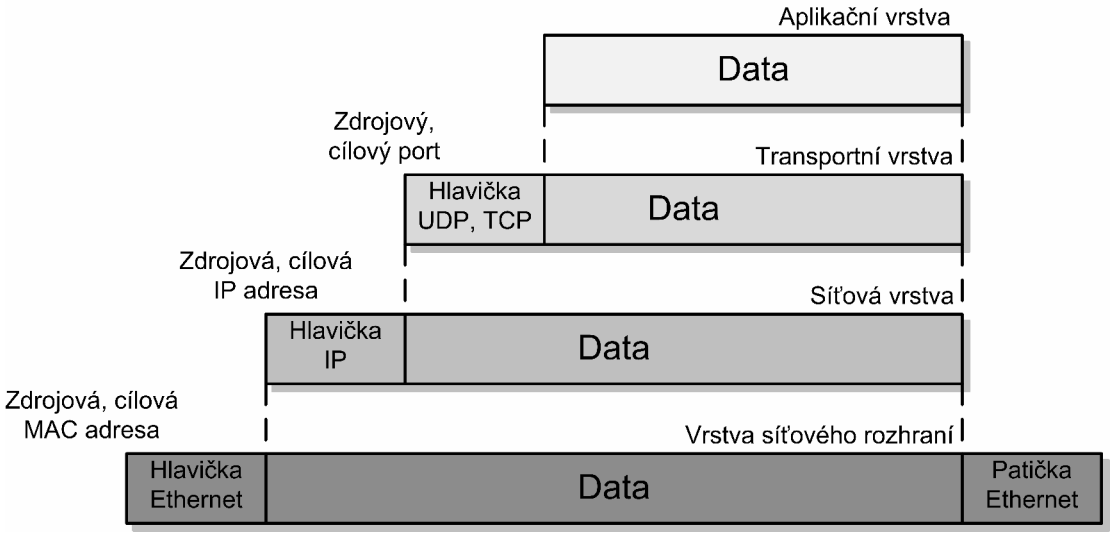


Obr. 7: Architektura TCP/IP

Na Obr. 7 je také vidět způsob přenosu informaci přes síť Internet od zdroje k cíli. Můžeme si to představit takto: Uživatel A odesílá data. Tato data jsou generována aplikační vrstvou a předána do vrstvy transportní, kde jsou opatřeny hlavičkou (obsahuje číslo portu) a datová jednotka se označuje jako datagram. Data jsou dále předána do síťové vrstvy. Zde je opět přidána hlavička, která obsahuje IP adresu pro směrování od zdroje k cíli. Poslední vrstvou je vrstva síťového rozhraní. Ta slouží pro přístup k fyzickému přenosovému mediu. Je specifikována pro každé fyzické médium a kromě hlavičky přidává i patičku.

Data dále putují do směrovače. Zde se provede „rozbalení“ paketu až k síťové hlavičce, která obsahuje IP adresu. Podle zjištěné IP adresy je paket směrován k cíli anebo do dalšího mezilehlého směrovače.

V cíli se opět provádí rozbalování datové jednotky, tentokrát v opačném pořadí. Proces zapouzdřování dat na jednotlivých vrstvách ukazuje Obr. 8.



Obr. 8: Zapouzdření dat v protokolu TCP/IP

2.2.4 Datové linky

Do mnoha firem a domácností jsou zavedeny telefonní linky. Ty nohou být s výhodou využity pro datovou komunikaci mezi měřicím zařízením a datovou centrálou dodavatele. Na počátku 90. let 20. století se začalo usilovně zkoumat, jak urychlit přenos po těchto zavedených telefonních linkách, které byly zatím využívány pouze pro přenos hovorů. Jako nejvhodnější se osvědčilo zavést na tyto telefonní linky technologii označenou zkratkou DSL (Digital Subscriber Line).

Technologie DSL využívá faktu, že pro přenos telefonních hovorů vystačí pásmo 3 - 3400 kHz. Metalická vedení jsou však schopna přenášet signály i v řádu MHz.

S tím, že vedení nejsou přímo uzpůsobena, aby pracovala na takto vysokých kmitočtech, jsou spojeny i jisté nevýhody. Nevýhodou je vzdálenost. Se zvyšující se vzdáleností značně klesá přenosová rychlost. Existuje několik druhů DSL technologií, které ukazuje Tab. 6 [11].

Tab. 6: Systémy DSL

Označení	download Mbit/s	upload Mbit/s	Použití
ADSL (Asymmetric DSL)	8	1	sdílení s tel. linkou
ADSL2+	25	4	novější verze ADSL
SDSL (Symmetric DSL)	2	2	symetrická linka
DSL (ISDN)	0,128	0,128	přenos hovoru, přístup na Internet
HDSL (High bit rate DSL)	2	2	propojení lokálních sítí
VDSL (Very high speed DSL)	36	36	přenos HDTV, multimédia

Pozn.: ISDN (Integrated Services Digital Network)

Rozlišujeme dva druhy přenosů. U symetrického je přenosová rychlost upload shodná s rychlostí download. Při nesymetrickém přenosu je zpravidla download vyšší než upload.

2.2.5 PLC (PowerLine Comunication)

Zkratka označuje způsob přenosu dat pomocí elektrické sítě. Nejedná se o technologii, která by se používala výhradně v systémech Smart metering. PLC je předurčeno k vytváření lokálních sítí a připojení koncových uživatelů do celosvětové sítě Internet. Podle způsobu používání a také podle přenosových rychlostí se PLC často označuje takto [4]:

- BPL (BroadBand over PowerLine)
- PLT (Power Line Telecom)
- PLN (Power Line Networking)
- PDSL (Power line Digital Subscriber Line)

Především technologie PLC nachází uplatnění v systémech chytrého měření elektrické energie. Pro přenos řídicích informací totiž využívají stávající elektrické vedení.

Problémem PLC je, že elektrická vedení nejsou pro přenos informací uzpůsobena. Z tohoto důvodu se při použití PLC vyskytuje značné rušení způsobené především spotřebiči zapojenými do sítě. Další nevýhodou je malý dosah způsobený velkým útlumem na vedení. Problémem je i absence standardu pro PLC, díky čemuž doposud tato technologie nepronikla příliš na trh. Nedostatek by měl řešit projekt OPERA (Open PLC European Research Alliance), který seskupuje sdružení zabývající se PLC. Patří sem:

- PLC Forum
- PowerLine Communication Association
- United Power Line Council
- PLC Utilities Alliance

Pro systémy Smart metering se využívá pouze úzkopásmové PLC. Přenosové rychlosti jsou v řádu jednotek až stovek kbit/s [7]. Úzkopásmové PLC je standardizované normou CENELEC EN 50065 a rozdělené na následující pásma Tab. 7:

Tab. 7: Kmitočtové rozdělení úzkopásmového PLC

Pásmo	Kmitočtový rozsah	Poznámka
	3 až 95 kHz	Vyhrazeno pro dodavatele elektrické energie
A	9 až 95 kHz	Vyhrazeno pro dodavatele, se souhlasem i pro odběratele
B	95 až 125 kHz	Vyhrazeno pro odběratele
C	125 až 140 kHz	Vyhrazeno pro odběratele – protokol ČSN EN 50065
D	140 až 148,5 kHz	

Dosah pro úzkopásmové PLC se pohybuje okolo 1500 m. Pokud je potřeba přenést data dále využívá se opakovačů.

2.2.6 RF (Radio Frequency)

Radiofrekvenční přenos se uplatňuje především v komunikaci mezi měřidlem u spotřebitele a datovým koncentrátorem. Jedná se o bezdrátovou komunikaci, ve většině případů v bezlicenčním pásmu 433 MHz (dosah 200 m) nebo 868 MHz. Radiofrekvenční technologie je vhodná u odečtů plynu, vody. Tedy tam, kde nejsou přímo k dispozici elektrické vodiče pro nasazení technologie PLC.

3 Teoretický rozbor

V praxi je důležité popsat problematiku sběru dat matematickými vztahy. V této kapitole bude rozebrán postup, jak si představit sběrnou síť z matematického hlediska. Například radiofrekvenční komunikace má charakter spojení typu MESH, kdežto technologie přenosu zpráv od koncového uživatele směrem k datové centrále má stromovou strukturu. Pro různé typy topologií je nutné najít správný matematický „vzor“. Popis struktur, které se nacházejí v systémech chytrého měření, je možný pomocí teorie grafů. V následující části bude tento postup popsán.

3.1 Úvod do teorie grafů

Jak již bylo zmíněno v úvodu, pro popis struktury je vhodné využít teorii grafů. V této části budou rozebrány základní prvky a pojmy z teorie grafů, které poslouží pro prvotní orientaci a následné pochopení provázání se sběrnou sítí.

3.1.1 Definice grafu

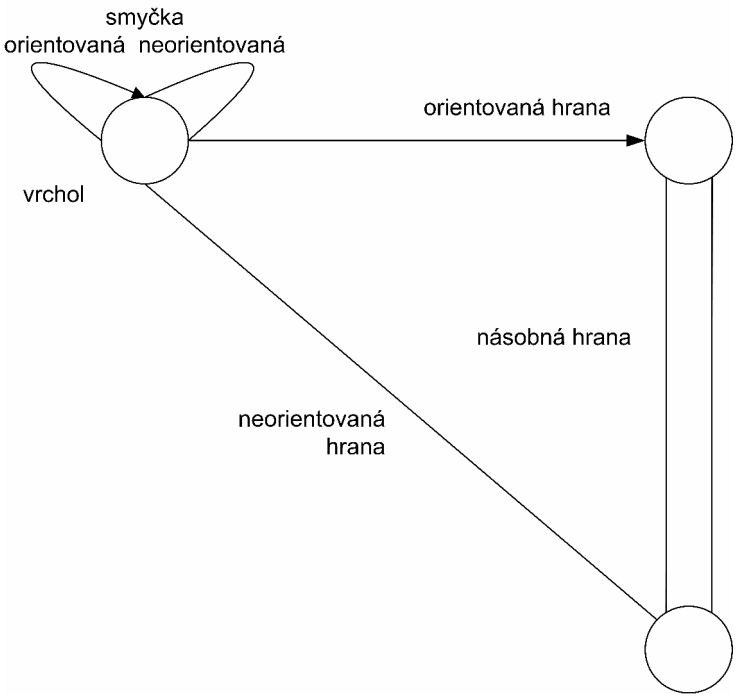
Pro základní orientaci je nejprve vhodné uvést obecný popis grafu. Popis není sice matematicky úplný, ale poslouží pro jednoduché pochopení toho, co si pod pojmem „graf“ vlastně představit.

Graf je složen z **vrcholů**, které jsou mezi sebou propojeny pomocí **hran**. Hrana může být orientovaná nebo neorientovaná a vždy spojuje vrcholy. **Orientovaná hrana** spojuje počáteční a koncový vrchol, kdežto u **neorientované hrany** chápeme spojení dvou vrcholů jako symetrické, a tedy počáteční a koncový uzel není rozlišen.

Orientovaný graf je složen pouze z orientovaných hran a **neorientovaný graf** obsahuje jen neorientované hrany. **Smíšený graf** je kombinací orientovaných a neorientovaných hran.

Posledním pojmem je smyčka. **Smyčka** spojuje jeden vrchol sám se sebou.

Jednotlivé pojmy ilustruje Obr. 9. Další podrobnosti lze najít v lit. [6].



Obr. 9: Smíšený graf

Nyní si uvedeme matematickou definici grafu.

Orientovaný graf, označíme písmenem *G*, je tvořen souborem vrcholů *V*, množinou hran označenou *E* a zobrazením $\varepsilon : E \rightarrow V^2$. Tento vztah se nazývá incidence a udává, že každé hraně $e \in E$ přiřazujeme uspořádanou dvojici vrcholů *(x, y)*. Matematická definice grafu je dána vztahem (1).

$$G = (V, E, \varepsilon) \tag{1}$$

Proměnou *x* se označuje **počáteční vrchol** a proměnnou *y* značíme **koncový vrchol**. Počáteční vrchol hrany *e* označujeme *Pv(e)* a koncový vrchol *Kv(e)*. Vrcholy můžeme také nazvat krajními vrcholy vzhledem k hraně *e*. Hrana *e* tedy spojuje body *x* a *y*. Vztah mezi vrcholy a hranou nazýváme incidentní a také naopak hrana *e* je incidentní s vrcholy *x* a *y*. Množina hran může být i prázdná!

Pokud budeme uvažovat počáteční bod hrany *Pv(e)* roven koncovému bodu hrany *e Kv(e)*, tedy *Pv(e) = Kv(e)*, pak hranu *e* nazýváme orientovanou **smyčkou**. Jinými slovy hrana *e* je incidentní pouze s jedním vrcholem. To znamená množina ε *(e)* je jednoprvková.

V grafu může nastat i situace, kdy k vrcholu nepřísluší žádná hrana (vrchol není incidentní s žádnou hranou). Takový vrchol nazýváme **izolovaným vrcholem**.

Jako **násobné** nebo také **rovnoběžné** označujeme takové hrany, u kterých jsou shodné počáteční a koncové uzly. Tedy $\varepsilon(e_1) = \varepsilon(e_2)$, nebo lze také napsat $Pv(e_1) = Pv(e_2)$ a $Kv(e_1) = Kv(e_2)$ [6].

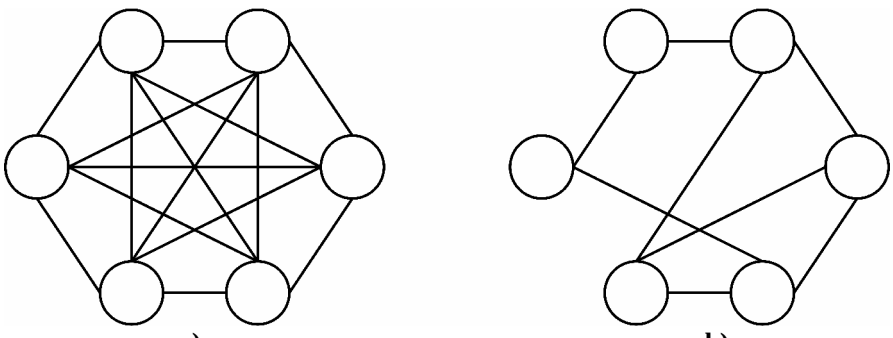
Neorientovaný graf slouží pro situace, kde nám nezáleží anebo nechceme rozlišovat mezi počátečními a koncovými vrcholy. Označení je shodné s (1) a pojmy jsou také velice podobné. Jediný rozdíl je v označení hran. Hrany se v tomto případě označují jako **neorientované**.

Pro úplnost je vhodné ještě uvést pojmy **prázdný graf** (graf, který neosahuje žádné vrcholy a tím pádem i žádné hrany) a **nekonečný graf** (tj. graf, který má nekonečně mnoho vrcholů).

3.2 Rozbor radiofrekvenční části

Při detailnějším pohledu na koncovou část systému Smart metering narazíme na radiofrekvenční přenos dat v oblasti zákazníka (pravá spodní část Obr. 4). RF technologie umožňuje spojení typu MESH. Výhodou této topologie je především redundance spojových tras a také možnost jednoduchého rozšiřování sítě.

Rozlišujeme dva typy sítí MESH. Prvním typem (Obr. 10a) je „plná“ MESH (full MESH), ve které je každý uzel přímo propojen s ostatními uzly. V takové síti existuje velké množství spojení a roste náročnost jak na vybudování, tak i na směrování. Z tohoto důvodu se častěji využívá síť typu částečný MESH (Obr. 10b), kde již nemají uzly pevně danou strukturu a propojení každý s každým neexistuje [5].



a) **b)**

Obr. 10: Topologie a) plná MESH, b) částečná MESH

V případě, že dojde k přerušení některé přenosové cesty, redundance zajistí nalezení jiné cesty a tím je dosaženo vyšší spolehlivosti. Díky existenci alternativních cest je možný i tzv. load balancing v překladu rozložení zátěže. Další výhodou je možnost libovolného rozšíření sítě. Jednotlivé uzly tvoří v podstatě další přístupové body, ke kterým je možné připojit další uzly a tím síť rozšiřovat.

Za cenu těchto výhod je ovšem v sítích typu MESH složité směrování, které má vliv na propustnost sítě a také na odezvu. Na jednotlivé přístupové body vyvstávají větší nároky z hlediska rozsáhlejších směrovacích tabulek a nutnost implementovat mechanizmus na ochranu sítě proti směrovacím smyčkám.

S radiofrekvenčním přenosem také souvisí nižší bezpečnost a spolehlivost. Bezpečnostní hledisko lze zaručit implementací nejrůznějších zabezpečovacích technik. Zajištění vyšší spolehlivosti je složitější. Obecně se dá říct, že radiofrekvenční přenos, tak jako každá bezdrátová technologie, je náchylný na přenosové prostředí. Tedy záleží na konkrétním umístění a aplikaci každého přístupového bodu. Dosah je ovlivněn stíněním, které je hlavně v hustě zastavěných oblastech značné. Z tohoto důvodu se zkracuje dosah radiofrekvenční technologie a v současné době se od jejího nasazení v oblasti chytrého měření spíše upouští.

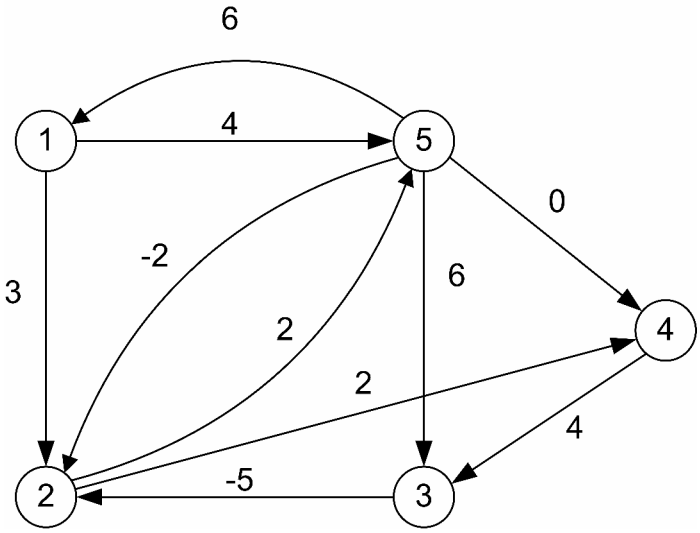
3.2.1 Hledání nejkratší cesty

Pokud se blíže podíváme na topologie typu MESH vidíme, že mezi zdrojem a cílem komunikace může existovat několik cest. Z matematického hlediska se topologie podobá grafu s různě orientovanými hranami. Z důvodu, že existuje velké množství cest, můžeme si zvolit cestu, kterou využijeme. Je jasné, že hledání nejkratší cesty nebude (především v rozsáhlých sítích) jednoduché. Abychom nemuseli hledat cesty ručně, využijeme výpočetní techniku a matematiku, která nám pomůže.

Jak bylo řečeno v úvodu této kapitoly, musíme náš problém (v tomto případě hledání nejkratší cesty) rozebrat z hlediska matematického a následně sestavit vhodný algoritmus. Ten nám pomůže s hledáním nejkratší cesty. Z tohoto důvodu si představíme naši topologickou síť typu MESH jako grafu.

Jednotlivé uzly sítě MESH (tedy radiofrekvenční měřidla) si znázorníme jako vrcholy grafu a přenosové cesty si zobrazíme jako hrany v grafu. Podle předpokladu nebude radiofrekvenční část tvořit úplnou MESH síť, půjde tedy o částečnou MESH.

Po překreslení topologie do grafu můžeme získat například Obr. 11, na kterém si ukážeme matematický postup výpočtu nekratší cesty.



Obr. 11: Graf k výpočtu nejkratší cesty

V grafu se nyní u jednotlivých cest objevuje nový pojem **ohodnocení cesty** nebo také cena. V našem případě je ohodnocením cesty oceněna vzdálenost mezi vrcholy. Tímto mechanizmem je možné preferovat jednu cestu před ostatními. To je vhodné, pokud máme například k dispozici více cest a některou z nich chceme využívat jen jako zálohu v případě výpadku preferované cesty. Například z reálného života si můžeme představit situaci, kdy potřebujeme najít nejkratší cestu z města 1 do města 3. Do grafu tedy vyneseme jednotlivá města (vrcholy) a propojíme je cestami (zakreslíme tedy hrany grafu). Každou cestu označíme počtem kilometrů a již nezbývá nic jiného, než spustit algoritmus na hledání nejkratší cesty. Pro příklad propojení měst je podivné ohodnocení hran zápornou délkou. Ale i toto má své opodstatnění. Pokud bychom například chtěli preferovat některou z cest, není nic jednoduššího, než ji ohodnotit zápornou délkou. A nyní již k vlastnímu výpočtu:

Na Obr. 11 je znázorněn orientovaný graf, a proto využijeme modifikovaný Dijkstrův algoritmus. Tento algoritmus je nejvhodnější, neboť konvertuje k výsledku nejrychleji. Algoritmus je založen na těchto principech:

- Princip trojúhelníkové nerovnosti [6]:

$$U(y) \leq U(x) + a(x, y) \tag{2}$$

U(x) je délka dosud nejkratší cesty z bodu *r* (počáteční vrchol – root) do bodu *x*. Pokud jsme žádnou cestu doposud nenašli, bude $U(x) = \infty$.

U(y) je délka cesty do nového bodu *y* z vrcholu *r*.

a(x, y) je délka hrany z bodu *x* do bodu *y*.

Jestliže trojúhelníková nerovnost neplatí, znamená to, že vzdálenost označená *U(y)* není nejkratší cestou z bodu *r* do bodu *y*. V tomto případě provedeme snížení hodnoty *U(y)* podle vzorce:

$$U(y) = U(x) + a(x, y) \tag{3}$$

Tím zajistíme, že v proměnné bude hodnota některé jiné (kratší) cesty z vrcholu *r* do vrcholu *y* [6].

- Druhým předpokladem je, že pokud hrana již splňovala trojúhelníkovou nerovnost, může k jejímu porušení dojít pouze za předpokladu, že došlo ke snížení trojúhelníkové nerovnosti v jejím počátečním vrcholu. Tedy pokud došlo ke snížení *U(x)* je nutné otestovat všechny hrany vycházející z tohoto vrcholu. Definujeme si tedy množinu vrcholů *M*, která bude obsahovat vrcholy, u nichž došlo ke snížení hodnoty *U*. Množina *M* je na počátku výpočtu prázdná, *M = 0*.
- Podle Dijkstrova algoritmu budeme z množiny *M* vybírat vždy vrchol, který má nejnižší hodnotu *U*. Tato volba je odůvodněna tím, že vrchol s nejnižší hodnotou *U* se s největší pravděpodobností již nebude měnit a tudíž nebude znovu zařazen do množiny *M*.
- Poslední proměnná bude udržovat informace pro sestavení cesty, kudy vede nejkratší cesta. Označíme ji např. *O*. Bude se měnit pokaždé, když dojde ke snížení hodnoty *U(y)* pro vrchol *y*. Do proměnné *O* tedy zaznamenáme hranu, která snížení způsobila.

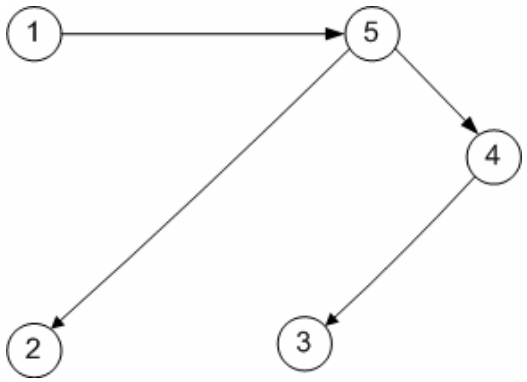
Takže podle příkladu z Obr. 11 budeme hledat nejkratší cestu z vrcholu 1 do ostatních vrcholů. Postup výpočtu je následující [6]:

		M{1}
x = 1:		M = 0
	U(2) = 3	M = {2}
	U(5) = 4	M = {2,5}
x = 2:		M = {5}
	U(4) = 5	M = {4,5}
	U(5) bez změny	M = {4,5}

x = 5:		M = {4}
	U (2) = 2	M = {2,4}
	U (3) = 10	M = {2,3,4}
	U (4) = 4	M = {2,3,4}
x = 2:		M = {3,4}
	U(4) bez změny	M = {3,4}
	U(5) bez změny	M = {3,4}
x = 4:		M = {3}
	U (3) = 8	M = {3}
x = 3:		M = 0
	U(2) bez změny	M = 0

Výsledek výpočtu:

vrchol x	1	2	3	4	5
vzdálenost U(x)	0	2	8	4	4
O(x)	-	(5,2)	(4,3)	(5,4)	(1,5)

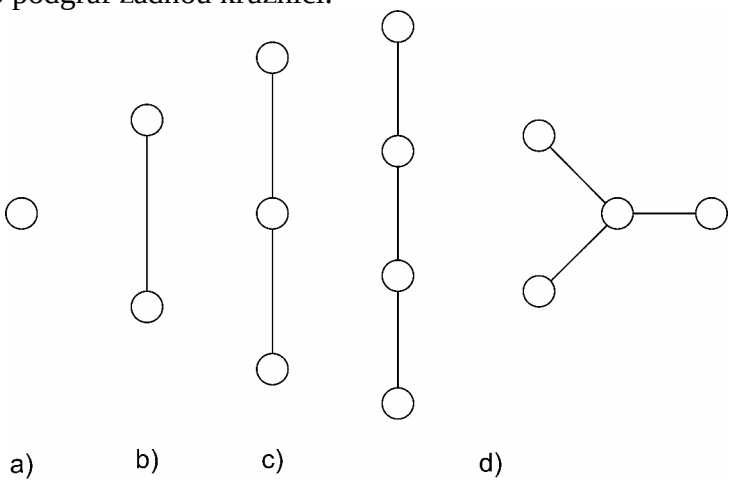


Obr. 12: Výsledek nejkratších vzdáleností

3.3 Stromová struktura

Pokud se podíváme na strukturu chytrého měření v celkovém měřítku, nejpravděpodobněji narazíme na stromovou architekturu. Pro její popis se jako nejvhodnější jeví pojem „strom“ z teorie grafů.

Strom je v teorii grafů definován takto: „Stromem se označuje konečný souvislý graf, který neobsahuje jako podgraf žádnou kružnici.“



Obr. 13: Různé struktury stromu

Obr. 13 ukazuje různé uspořádání stromu. Strom je popsán vztahem (4):

$$S = (V, E) \tag{4}$$

kde:
S označuje strom
V udává množinu vrcholů a
E je označení pro množinu hran [6].

Podle množství vrcholů rozlišujeme typy stromů:

- *V* = 1, ilustruje Obr. 13a), strom nemá žádnou hranu
- *V* = 2, Obr. 13b), strom obsahuje pouze jednu hranu
- *V* = 3, Obr. 13c)
- *V* = 4, Obr. 13d) vlevo – strom má dva uzly 1. stupně a dva uzly 2. stupně
  vpravo – strom obsahuje tři vrcholy 1. stupně a jeden 3. stupně

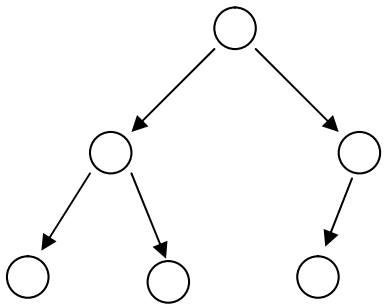
Obecně platí vztah [6]:

$$E = V - 1 \tag{5}$$

Vztah udává, že počet hran ve stromě je vždy o jednu menší než počet vrcholů.

Dalším pojmem je **kořenový strom**. V podstatě se jedná o orientovaný graf, v němž existuje význačný vrchol *r*, tzv. kořen (root). Do kořene nevede žádná hrana, do kteréhokoliv jiného vrcholu vede právě jedna hrana. Navíc jsou všechny vrcholy z kořene *r* dostupné.

Charakteristické uspořádání kořenového stromu ukazuje Obr. 14.

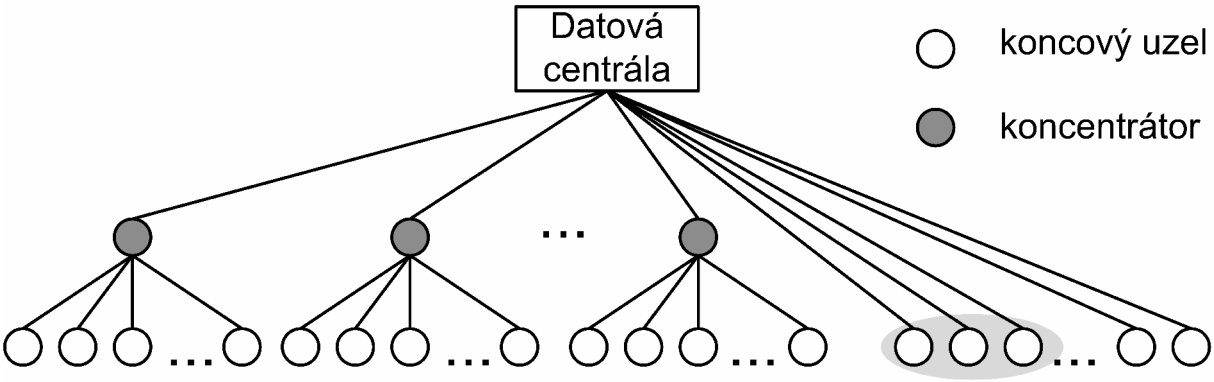


Obr. 14: Kořenový strom

Pro jednoduchost a lepší orientaci se často používá označování z reálného života. Pokud vede hrana z vrcholu *x* do vrcholu *y*, pak vrchol *x* označujeme „**otcem**“ (rodičem) vrcholu *y*. A stejně tak vrchol *y* označujeme „**potomkem**“ (synem) vrcholu *x*. Dva stejné vrcholy, které mají společného rodiče, se často označují jako „**bratři**“. Vrchol, který nemá žádného potomka označujeme jako „**list**“. Samotný kořen otce nemá.

Pro potřeby popisu stromu je také důležité znát jeho výšku. **Výšku stromu** určíme jako největší počet hran vedoucích z kořene stromu do jeho nejvzdálenějšího listu. Obdobně **výška** určitého **bodu** se určí jako maximální počet hran vedoucích z daného bodu do nejvzdálenějšího listu.

Sběrnou síť si představíme jako rozsáhlý kořenový strom s mnoha tisíci koncovými uzly (listy). Pokud budeme považovat tyto listy za měřidla instalované u spotřebitelů a budeme chtít odečítat změřené hodnoty z těchto měřidel, pak musíme určit intervaly, v jakých budou měřidla svá data odesílat. Tímto načasováním se vyhneme přetížením přenosových linek a také dojde k rozložení zátěže datové centrály. Zatížení nebude nárazové, bude rovnoměrně rozložené.



Obr. 15: Sběrná síť

Hledání optimální hodnoty není jednoduché. Je třeba uvážit, jaký počet připojených měřidel je vyhovující z hlediska přenosových kapacit spojů a také z ekonomického hlediska.

Způsob realizace odečtů také není jednoznačný. Odečty mohou být prováděny samostatně (na základě nastaveného algoritmu v měřidlu), nebo na vyžádání (podnět od datové centrály). Dalším typem může být odesílání dat podle potřeby (data se shromažďují a teprve když je jich dostatečné množství, tak se odešlou). Ve skutečném provozu se dá předpokládat využití těchto druhů v různých kombinacích.

V reálné situaci je možné předpokládat, že měřidla budou samostatně (podle potřeby) odesílat data do koncentrátoru. Koncentrátor pak může čekat na výzvu od datové centrály a jako odpověď odeslat sesbíraná data.

Jinou variantou je vytvoření „plovoucího okna“ (zvýrazněno šedě na Obr. 15), které by se předávalo z jednoho koncového zařízení na druhé. Jednotlivá zařízení by mohla odesílat data pouze v době, kdy je okno u nich. Tato varianta by měla uplatnění především (nejen) v částech sběrné sítě, kde by nebyly použity koncentrátory.

3.4 Hierarchická agregace

Výsledné nároky na přenosovou soustavu může do značné míry ovlivnit její struktura. A právě vhodnou volbou umístění koncentrátorů a jejich počtem lze ušetřit cennou přenosovou kapacitu a zároveň v optimalizované soustavě zrychlit čas odečtů.

Sběrná síť má charakter stromové topologie. Princip hierarchické agregace (HA) se opírá právě o stromovou architekturu a tak ji můžeme s výhodou nasadit do sběrné sítě.

3.4.1 Princip

Princip agregace spočívá v tom, že sběrnou síť rozšíříme o koncentrátory, které budou sloužit pro lokální sběr dat. Tyto data budou následně odesílat svému nadřazenému koncentrátoru. Tento proces bude pokračovat, dokud se data nedostanou až do kořene stromu, tedy do datové centrály u poskytovatele. Data se v každém koncentrátoru shromáždí a výsledkem hierarchické agregace by mělo být snížení potřebných přenosových kapacit. Díky tomu, že nebude potřeba tak velká šířka pásma, budeme moci realizovat přenosy dat častěji a tím lépe reagovat na aktuální požadavky.

3.4.2 Prvky hierarchické agregace

Hierarchická agregace rozšiřuje standardní sběrnou síť o prvek koncentrátoru. Pro lepší orientaci v problematice jsou uvedeny jednotlivé pojmy.

Cílový bod – jde o prvek, ke kterému se šíří jednotlivá data. V návaznosti na sběrné sítě se jedná o sběrnou centrálu.

Vysílač – prvek, který je zdrojem dat. Jeho úkolem je odeslat data do cílového bodu. Pro potřeby sběrných sítí se také označuje jako měřidlo.

Sumátor – jde o pojem, který se zavádí speciálně pro potřeby HA. V oblasti sběrných sítí se označuje jako koncentrátor.

3.4.3 Výška stromu

Pro efektivní návrh agregačního stromu musíme zjistit jeho optimální výšku. Definice je možné najít v kapitole 3.3. Výška agregačního stromu je závislá na mnoha faktorech. Hlavním kritériem je počet klientů připojených do sběrného stromu. Každý koncentrátor zvládne obsloužit pouze určitý počet měřidel, proto při narůstajícím počtu koncových stanic bude narůstat i počet potřebných koncentrátorů. Dalším faktorem ovlivňujícím počet koncentrátorů je délka přenášené zprávy a interval mezi jednotlivými přenosy. Posledním faktorem vstupujícím do výpočtu je šířka pásma mezi jednotlivými prvky. Čím bude šířka pásma větší, tím se přenos uskuteční rychleji a počet koncentrátorů bude nižší. Jejich počet v agregačním stromě určíme podle následujících vzorců [12]. Pro první vrstvu stromu platí:

$$KON_1 = \frac{NZ_1}{T_1 BW} \tag{6}$$

kde:
KON_1 - udává počet koncentrátorů v první vrstvě agregačního stromu [-]
N - počet připojených měřidel [-]
Z_1 - délka přenášené zprávy [b]
T_1 - perioda odesílání zpráv [s]
BW - šířka pásma přenosových cest [b/s]

Pro koncentrátory v druhé vrstvě pak platí vzorec:

$$KON_2 = \frac{KON_1 Z_2}{T_2 BW} \tag{7}$$

kde:
KON_2 - udává počet koncentrátorů v druhé vrstvě [-]
Z_2 - délka přenášené zprávy v druhé vrstvě [b]
T_2 - perioda odesílání zpráv v druhé vrstvě [s]

Obecně lze tedy odvodit vztah pro výpočet počtu koncentrátorů v určité vrstvě H agregačního stromu:

$$KON_H = \frac{N \sum_{n=1}^{H} Z_n}{BW^H \sum_{n=1}^{H} T_n} \tag{8}$$

Podle vzorce (8) můžeme tedy určit optimální počet koncentrátorů v jednotlivých vrstvách.

3.4.4 Hledání optimální polohy koncentrátorů

Pro určení optimálního umístění koncentrátorů na určitém území můžeme s výhodou použít shlukování *k – means*. Princip spočívá v tom, že v množině bodů

$X = \{x_1, x_2, ..., x_n\}$ dokáže určit vektory $\mu_1, \mu_2, ..., \mu_k$ (pro k < n), takové, že pro ně platí minimální střední kvadratická odchylka. Jinými slovy: algoritmus hledá určitý počet *k* vektorů, které pro danou skupinu dat *X* splňují nejmenší euklidovskou vzdálenost pro všechny body.

Metoda funguje tak, že ze skupiny měřidel, která jsou umístěna na určitém území, je možné vypočítat optimální polohu koncentrátoru. Vstupními hodnotami je množina měřidel a počet koncentrátorů, které chceme pro přenos použít. V prvním kroku náhodně (popřípadě pomocí vhodné heuristiky) zvolíme inicializační hodnotu umístění koncentrátorů, tedy μ_j, *pro j = 1,..., k.*

Následně se provede první krok výpočtu. Určí se minimum euklidovské vzdálenosti podle nejbližšího souseda pro data obsažená ve vektoru *X* tak, aby se klasifikovala do vektorů μ_j, *pro j = 1,..., k.* Postup je zapsán vztahem:

$$y_i = \min_{j=1,...,k} \left\| x_i - \mu_j \right\| \tag{9}$$

kde:

y_i – označuje minimum euklidovské vzdálenosti

Druhým krokem určíme střední hodnotu dat x_i a nové souřadnice vektorů μ_j. Budeme brát na zřetel pouze takové prvky ze skupiny *X*, které náleží konkrétnímu μ_j. Celý postup druhého kroku shrnuje vztah:

$$\mu_j = \frac{1}{N_j} \sum_{i \in \{i : y_i = j\}} x_i \tag{10}$$

kde:

N_j – obsahuje prvky z množiny *X* a udává počet vzorků, které náleží k určitému vektoru μ_j.

Celý algoritmus pouze opakuje kroky 1 a 2 do doby, kdy v každém kroku dochází ke změně alespoň jednoho prvku z množiny *X*. Také lit. [13].

4 Návrh optimalizačního algoritmu

Abychom mohli vyzkoušet, jestli daný algoritmus přinese nějaké vylepšení, co se komunikace týká, je nejprve nutná jeho implementace a následné otestování v simulačním nástroji. Pro testování bude zvolen některý z dostupných simulačních nástrojů. Jeho volba, postup simulace a simulační výstupy v podobě grafů budou rozebrány dále v následující kapitole 5. Tato kapitola popisuje možnosti pro implementaci algoritmu a zjišťuje, které řešení se jeví jako optimální. Obsahem kapitoly je i samotný návrh programu, popis jeho ovládání a postup, jak správně využít výstup z programu.

4.1 Obecný úvod

Pro implementaci algoritmu jsem zvolil programovací jazyk Java s grafickým prostředím. Vlastní implementace algoritmu bude tedy pro uživatele skryta. Uživatel pouze jednoduše zadá vstupní hodnoty proměnných, program provede výpočet a vrátí uživateli výsledky. Výhodou takového řešení je možnost zcela jednoduše doplnit do programu pomocné funkce jakými jsou například automatický výpočet počtu koncentrátorů, výpočet míry zefektivnění a v neposlední řadě je zde také grafické rozhraní. Tím je zajištěna přehlednost při ovládání programu a interpretace výsledků se stává snadnou a jednoduchou.

Právě tyto výše uvedené doplňkové funkce, které svou činností zjednodušují návrh sběrné sítě mne vedly k tomu, sestavit vlastní implementaci algoritmu do samotného programu napsaného v jazyce Java. Existují určitě i jiné složitější možnosti. Příkladem může být implementace v jazyce C/C++ nebo využít například editor stavového automatu v simulačním nástroji OPNET Modeler. OPNET je však licencovaný nástroj a jeho primární funkcí je provádět simulace v oblasti sítí. Dalším problémem by bylo v OPNETu implementovat pomocné funkce a problémy by mohly nastat i při interpretaci výsledků.

Další možností je vytvořit jednoduchý skript například v matematickém programu Matlab. Do tohoto skriptu následně zadat vstupní hodnoty a nechat si vypsat výsledky. Nicméně i při nenáročném algoritmu by bylo velice pracné doplnit do takovéhoto skriptu všechny pomocné výpočty tak, aby byl stále přehledný a pro uživatele přívětivý.

4.1.1 Popis programovacího jazyka Java

Jak je všeobecně známo, Java je objektově orientovaný programovací jazyk (OOP), jehož vývoj je zajištěn firmou Sun Microsystems. Jde o program představený v roce 1995. Podle zdroje [15] je Java v současné době nejpoužívanější programovací jazyk. Jedná se o multiplatformní jazyk, což by mělo znamenat, že kód napsaný například pod operačním systémem Linux je možné spustit i pod operačním systémem Windows. Od poloviny roku 2007 je Java publikována jako open source produkt, což také do značné míry přispívá její popularitě.

Multiplatformnost Javy je ovšem vykoupena tím, že před každým spuštěním je nutné provést celkové přeložení zdrojového kódu. Tím pádem je spouštění aplikací pomalejší. Další nevýhodou je i vyšší paměťová náročnost spouštěných programů.

4.1.2 Vývojové prostředí pro jazyk Java

Je asi jasné, že programování v textových souborech by nebylo nejmoudřejší volbou a proto je vhodné se poohlédnout po vývojovém prostředí pro Javu. Jako ve většině případů existují jak placené, tak bezlicenční programy pro vývoj. Z první licencované skupiny jde například o nástroj jBuilder. Vhodnější ovšem je zaměřit se na programy poskytované zdarma, protože právě oblasti vývojových prostředí pro jazyk Java je jejich dostupnost a propracovanost na velmi vysoké úrovni. Těmito variantami jsou například Eclipse, což je

vývojové prostředí podporované firmou IBM. Obsahuje značné množství nástrojů pro vývoj a je hojně používané. Jeho hlavním nedostatkem je neexistující rozšíření, které by zjednodušilo návrh a správu grafického rozhraní. Tímto není míněno, že by se grafické rozhraní nedalo ve vývojovém prostředí Eclipse vytvořit. Je to pouze nadmíru složité. Musí se totiž vytvářet implicitně řadou příkazů a definic.

Tuto nevýhodu v podobě chybějícího modulu pro návrh grafického rozhraní odstraňuje použití vývojového prostředí NetBeans, který je ze značné části vyvíjen programátory z České republiky. Vývoj probíhá za podpory firmy Sun Microsystems, ta je zároveň i hlavním sponzorem. V polovině roku 2000 bylo vývojové prostředí NetBeans uvolněno pod open source licencí pro volné používání. Vytváření grafického rozhraní zde probíhá jednoduchým klikáním a přetahováním potřebných položek. Tuto vlastnost lze s výhodou využit ve vytvářeném programu, protože hlavní funkcí bude grafické znázornění polohy koncentrátorů a měřidel [17].

4.1.3 Předpoklady pro spuštění a správnou funkci programu

Pro spuštění programu je nutné mít nainstalované prostředí pro běh Javy. Anglický název je Java Runtime Environment, zkratka JRE. Je to nezbytný program pro spuštění nejen vytvořeného optimalizačního programu, ale je nutný pro spouštění veškerých Java aplikací, jako jsou například internetové hry, nejrůznější webové aplikace atd.

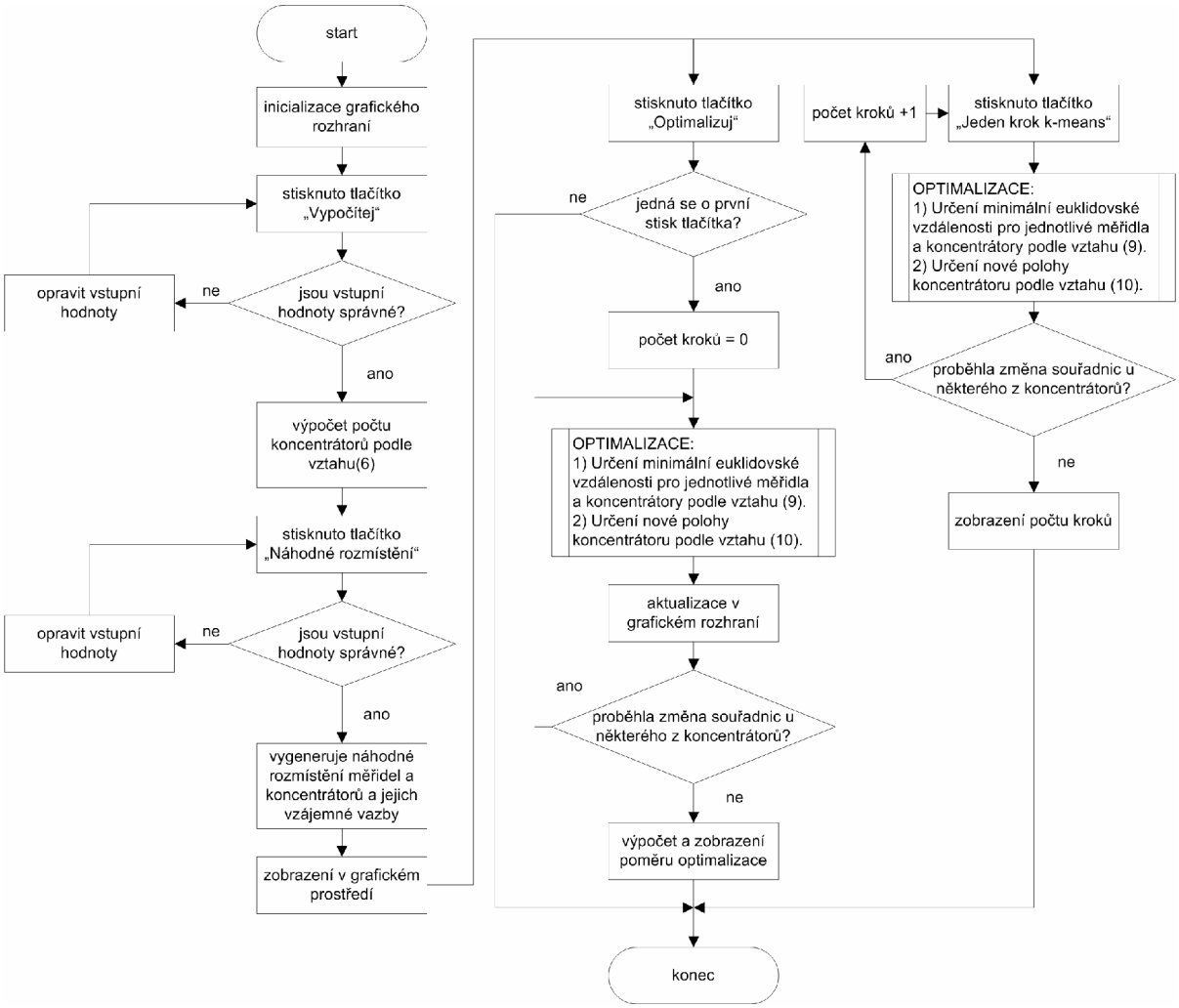
Existují verze pro různé platformy operačních systémů (OS). Běhové prostředí pro Javu lze samozřejmě stáhnout pro OS Windows, Linux, ale třeba i pro Solaris.

Pokud tedy na počítači, kde má být spuštěn optimalizační program běhové prostředí chybí, lze ho získat z odkazu:

http://www.java.com/en/download/index.jsp

4.2 Návrh optimalizačního programu

Funkci navrhovaného optimalizačního algoritmu lze odvodit z vývojového diagramu, který je uveden na Obr. 16, popis je uveden níže. Podrobnější informace o jednotlivých částech kódu a významu funkcí lze získat z dokumentace pro Javu v literatuře [17].



Obr. 16: Vývojový diagram funkcí optimalizačního programu

V prvním kroku programu dojde k inicializaci grafického rozhraní. Následuje stisk tlačítka **Vypočítej**. Program zkontroluje, zda jsou vstupní hodnoty zadány bezchybně, pokud je vše v pořádku dojde k výpočtu optimálního počtu koncentrátorů podle vztahu (6). Nejdůležitější část zdrojového kódu, který popisuje tuto činnost:

```
      ...                                   //načte zadané hodnoty z textových oblastí
int meridel = Integer.parseInt(jTextFieldPocetMeridel.getText());
int delkaZpravy = Integer.parseInt(jTextFieldDelkaZpravy.getText());
int perioda = Integer.parseInt(jTextFieldPeriodaZpravy.getText());
int sirkaPasma = Integer.parseInt(jTextFieldSirkaPasma.getText());

                                        //zkontroluje, zda nejsou vstupní hodnoty záporné
if (meridel <= 0 || delkaZpravy <= 0 || perioda <= 0 || sirkaPasma <= 0) {
                                        //pokud ano, vypíše chybu
      jLabelChyba.setText("Hodnoty musí být kladné a větší než nula!");
} else {                                //jinak spočítá optimální počet koncentrátorů
      kon = (meridel * delkaZpravy * 1000) / (perioda * sirkaPasma * 1000);
}
jTextFieldPocetKoncentratoru.setText(kon.toString());      //a uloží ho do textového pole
       ...
```

Následně, stiskem tlačítka **Náhodné rozmístění** dojde k vygenerování zadaného počtu měřidel a vypočítaného počtu koncentrátorů. Před vlastním generováním se provede kontrola vstupních zadaných dat a po vygenerování se vše zobrazí v grafické podobě v optimalizačním programu v podobě níže definovaných symbolů. Tuto funkci zajišťuje tento výňatek ze zdrojového kódu.

```
       ...
nahodnacisla(pocetM, pocetK);                       //generuje náhodná čísla pro souřadnice
meridlo = new Meridlo[pocetM];                      //nové pole objektů ze třídy měřidel
koncentrator = new Koncentrator[pocetK];
for (int i = 0; i < pocetM; i++) {                  //naplní objekty náhodnými souřadnicemi
       meridlo[i] = new Meridlo(i, xCislaM[i], yCislaM[i]);
}
       ...
```

Zobrazení grafických symbolů v optimalizačním programu se provede následujícím kódem:

```
       ...
for (int i = 0; i < jRadioMer.length; i++) {    //zobrazí se symbol na daných souřadnicích
       jRadioMer[i].setBounds(meridlo[i].getX(), meridlo[i].getY(), sirka, vyska);
}
       ...
```

Nyní může být stisknuto tlačítko **Optimalizuj** nebo **Jeden krok k-means**. Ve výsledku mají obě tlačítka stejnou funkci. Tlačítko **Jeden krok k-means** provádí optimalizaci po krocích. V podstatě je na něj nutné klikat tak dlouho, dokud nedojde k ustálení stavu a k nalezení optimální polohy. To je vhodné pro sledování polohy koncentrátorů, která se může měnit. Tlačítko **Optimalizuj** provede tento cyklus automaticky a vrátí pouze hodnotu s optimálním umístěním. Důležitá část zdrojového kódu:

```
       ...
do {
       kMeans(kroku);        //provádí metodu kMeans
       kroku++;              //po každém průchodu zvýší počet kroků
} while (zmena != 0);        //dokud dochází ke změnám
       ...
```

Jak je vidět, je volána metoda **kMeans**. Ta je základem celého programu a jde v podstatě o implementaci vztahů (9) a (10).

```
      ...                                   //hledá nejbližší koncentrátor k danému měřidlu
mezi[i] = (int) Math.sqrt(Math.pow(meridlo[j].getX() - koncentrator[i].getX(), 2) +
Math.pow(meridlo[j].getY() - koncentrator[i].getY(), 2));
       ...
jRadioMer[j].setBackground(koncentrator[i].getColor()); //měřidlo se označí příslušnou barvou
soucet[i][0] = soucet[i][0] + meridlo[j].getX();  //uloží pozice X měřidla pro koncentrátor i
soucet[i][2] ++;                                        //počet měřidel pro koncentrátor i
       ...                                              //testuje, zda došlo ke změně
if(koncentrator[i].getX() != (soucet[i][0] / soucet[i][2]) || koncentrator[i].getY() !=
(soucet[i][1] / soucet[i][2])){

koncentrator[i].setX(soucet[i][0] / soucet[i][2]);  //pokud ano, vypočítá nové souřadnice X
koncentrator[i].setY(soucet[i][1] / soucet[i][2]);  //a Y
zmena++;                                            //nastaví se příznak, že proběhla změna
       ...
```

Posledním zdrojovým kódem, který je zde uveden je třída definující strukturu měřidla a koncentrátoru. Obě struktury jsou podobné a slouží pro ukládání dat. Pro měřidlo je struktura následující:

```
       ...
public class Meridlo {              //datová struktura měřidla
    public int cislo;               //určuje pořadové číslo měřidla
    public int x;                   //souřadnice X
    public int y;                   //souřadnice Y
```

```
    public int barva;                //číselné označení barvy
    public Color color;              //konkrétní barevná hodnota
}
       ...
```

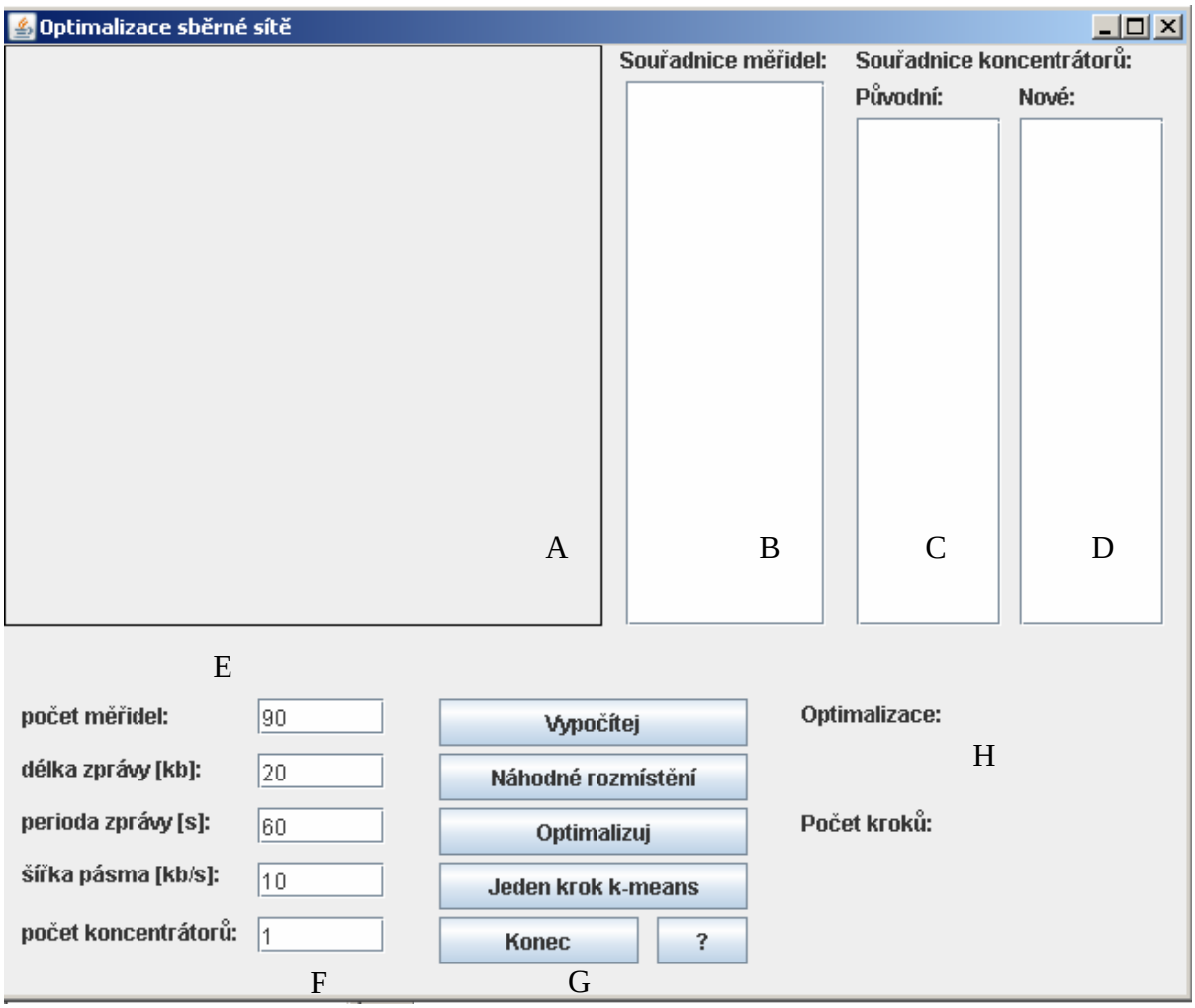
Kompletní zdrojový kód je přiložen v příloze A.

4.2.1 Popis a ovládání programu

Hlavní funkcí navrhovaného programu je hledání optimálního umístění koncentrátorů ve skupině měřidel. Doplňkovými funkcemi jsou výpočet počtu koncentrátorů, rozmístění měřidel a koncentrátorů v grafickém prostředí a také výpočet míry zefektivnění.

Program je uložen na přiloženém CD. Podrobnější informace lze získat v oddílu příloha B.

V prvním kroku po spuštění programu dojde k inicializaci grafického prostředí. Tento stav je zachycen na Obr. 17.



Obr. 17: Grafické rozhraní spuštěného programu pro optimalizaci

Program se dá rozdělit do sekcí, jejichž funkce je následující:

- sekce A: Tato sekce slouží pro grafické znázornění jednotlivých měřidel a koncentrátorů. Koncentrátory jsou po prvotním rozmístění označeny různobarevně, měřidla mají barvu šedou. Po provedení optimalizace se každé měřidlo označí takovou barvou, aby bylo jasné, ke kterému koncentrátoru patří.

- sekce B: Zde jsou zobrazeny náhodně vygenerované souřadnice měřidel. Výpis je ve formátu souřadnic *x*, *y*. Souřadnice lze jednoduše zkopírovat a přenést do libovolné jiné aplikace.
- sekce C: Po vygenerování náhodného rozmístění koncentrátorů jsou do tohoto pole umístěny souřadnice koncentrátorů.
- sekce D: Jsou zde také vypsány souřadnice koncentrátorů, jde o hodnoty po provedené optimalizaci. Tedy nové souřadnice, které vznikly výpočtem a použitím algoritmu pro optimalizaci sběrné sítě.
- sekce E: Prostor, kam se vypisují chybové stavy, špatné zadání a nestandardní chování programu.
- sekce F: Vstup programu. Je zde možné zadávat parametry, které slouží pro výpočet optimálního počtu koncentrátorů. Zobrazuje se zde také výsledek tohoto výpočtu, který lze i měnit.
- sekce G: Ovládací tlačítka programu.
- sekce H: Zde se zobrazí výsledné zoptimalizování sběrné sítě. Je zde zobrazen rozdíl mezi původní vzdáleností měřidel a nově získanou vzdáleností měřidel. V prvním řádku je absolutní rozdíl a ve druhém řádku se zobrazí procentuální zoptimalizování oproti původnímu stavu. Druhou položkou je počet kroků. Ta udává, kolikrát je nutné provést optimalizační cyklus, než je dosaženo nejlepšího umístění koncentrátorů.

Po spuštění se tedy zobrazí grafické rozhraní uvedené na Obr. 17. Stisknutím tlačítka **Vypočítej**, načte program aktuální zadaná data z polí **počet měřidel**, **délka zprávy**, **perioda zprávy** a **šířka pásma**. S těmito vstupními hodnotami se provede výpočet podle vztahu (6) a zjistí se optimální počet koncentrátorů.

Následným stiskem tlačítka **Náhodné rozmístění** dojde k vygenerování zadaného počtu měřidel a vypočítaného počtu koncentrátorů do grafického rozhraní. Koncentrátory jsou označeny symbolem a měřidla reprezentují  . Barevné odlišení znázorňuje, které měřidlo připadá k danému koncentrátoru.

Dále je možné stisknout tlačítko **Optimalizuj** nebo **Jeden krok k-means**. První jmenované tlačítko provede algoritmus pro určení optimální polohy koncentrátorů (k-means) tolikrát, dokud není nalezena optimální poloha všech koncentrátorů. Druhé jmenované tlačítko provede algoritmus k-means pouze jednou. Díky tomu je možné sledovat nejen, jak se mění umístění koncentrátorů, ale také to, že jednotlivá měřidla (na základě vzdálenosti) mění koncentrátory, ke kterým patří. Tato metoda je vhodná na ilustraci a pochopení funkce optimalizačního algoritmu. Po optimalizaci ještě dojde ke spočítání míry zefektivnění a zobrazení této hodnoty do příslušné sekce H. Také se zde objeví počet kroků, které jsou potřeba pro nalezení optimálního umístění.

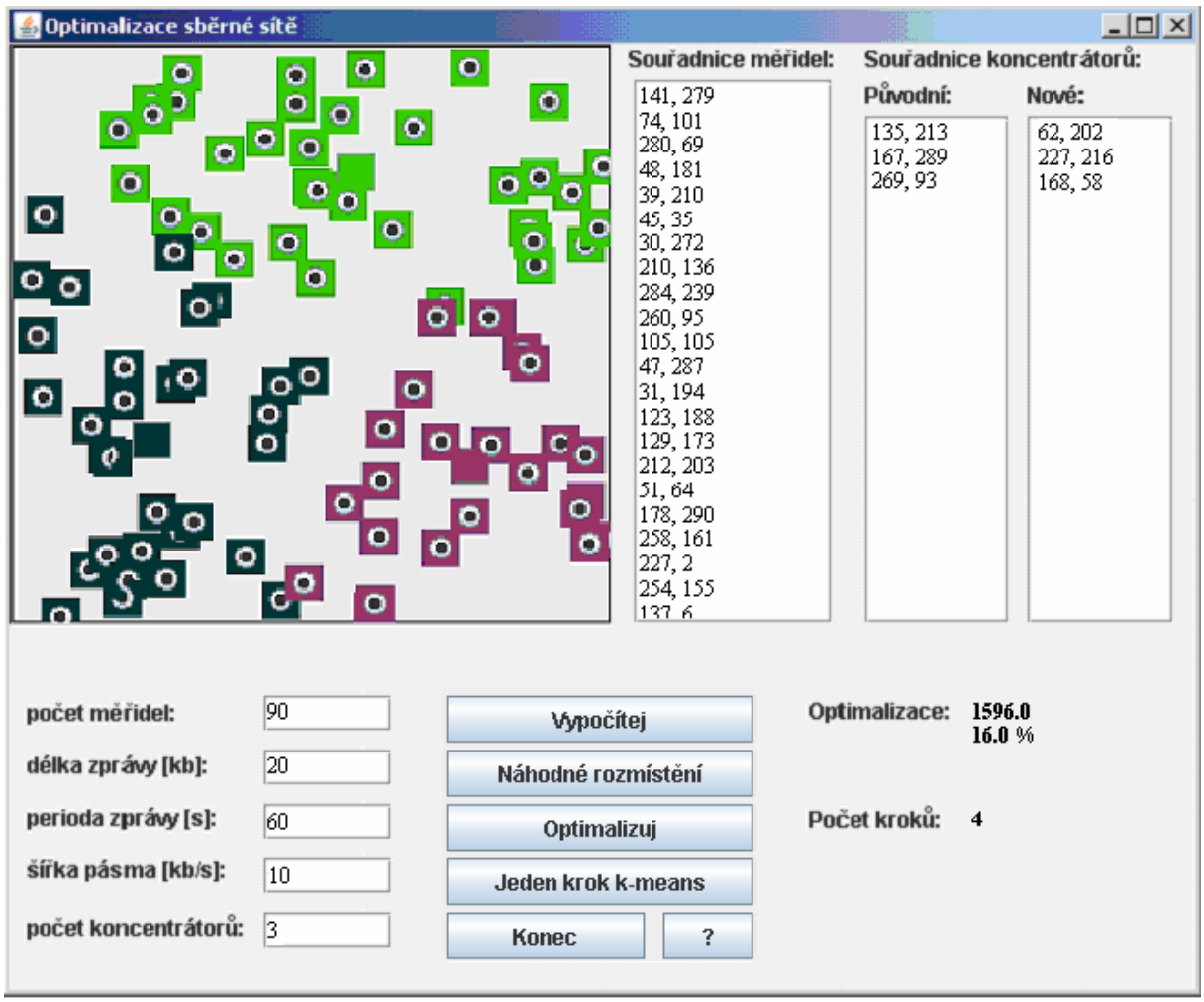
Posledním ovládacím tlačítkem je **Konec**. To slouží k ukončení programu a zavření okna.

Samozřejmě je vždy nutné při používání programu zadávat takové hodnoty, které jsou smysluplné a logické. Pokud je zadáno nadměrné množství měřidel (řádově stovky tisíc a víc), dojde ke značnému nárůstu výpočetní doby, v krajním případě až k vyčerpání paměti. Tato doba je závislá na výkonnosti počítače, na kterém je program spouštěn. Ke stejné situaci dojde i po zadání nesmyslného zadání ostatních parametrů.

4.3 Výstup programu

Výstupem programu jsou souřadnice měřidel a koncentrátorů. Jsou zde k dispozici souřadnice před optimalizací a po optimalizaci. Dalším prvkem je grafické rozmístění a barevné rozlišení toho, která měřidla náleží k danému koncentrátoru. Dále se zobrazuje

optimalizační poměr a počet kroků, které je nutno provést pro nalezení rovnovážného optimalizovaného stavu. Konkrétní výstup z programu je na Obr. 18. Výstupní hodnoty z tohoto obrázku jsou zároveň použity jako vstupní hodnoty pro simulaci, které se věnuje následující kapitola.



Obr. 18: Výstup z programu s konkrétními výsledky

5 Simulace v programu OPNET Modeler

Pro ověření funkčnosti a efektivity algoritmu uvedeného v předchozích kapitolách je nutné provést jeho odzkoušení v simulačním programu. Jako nejvýkonnější a nejlépe propracovaný program se jeví právě OPNET Modeler. Nabízí simulování, které je závislé na přenosových charakteristikách linek, dokáže poskytnout jednoduché i složité grafické závislosti a je schopen simulovat v několika scénářích. Další jeho předností je uživatelsky přívětivé simulační prostředí. Konfigurace sítě je prováděna v grafickém prostředí, které je velice intuitivní a přehledné.

Existuje i mnoho jiných simulačních programů, které jsou oproti OPNET Modeleru poskytovány zdarma nebo pod nekomerční licenční politikou. Příkladem může být například simulační nástroj NS2 (Network Simulator) nebo třeba OMNeT++, který je poskytován pod licencí GNU (General Public License). Oproti této výhodě nejsou ve většině případů tyto programy zdaleka tak dobře propracovány z funkčního hlediska a práce s nimi je složitá a zdlouhavá. V software NS2 se například provádí veškerá konfigurace pomocí textových souborů. OMNeT++ sice obsahuje grafické rozhraní pro rozvržení prvků v síti, ale kompilace se provádí neprakticky přes příkazový řádek. Neposlední nevýhodou je i licencování pod GPL, vývoj takto licencovaných aplikací je zajištěn mnoha programátory. Každý, naprosto nezávislý, programátor například rozšíří funkčnost programu o další oblast. A právě díky tomu ne vždy všechny komponenty takto licencovaných produktů fungují se stoprocentní spolehlivostí. Z těchto důvodů byl zvolen OPNET Modeler, který je komerčně vyvíjen a provozován. Vykoupením nevýhody s licencí je ovšem vysoká spolehlivost a uživatelská přívětivost.

5.1 Krátce o OPNETu

V OPNETu je možné vytvořit rozsáhlou síť a ověřit její chování v simulačním prostředí. V programu lze nastavit doby simulací, vzdálenost jednotlivých komponent, atd.

Při simulaci v OPNETu lze sledovat průběhy datové komunikace, jejich délku a efektivnost využití přenosových linek. Vše jde jednoduše zobrazit v grafických závislostech.

OPNET se dá rozdělit podle návrhu sítě do tří úrovní.

První úrovní je **project editor**. Ten slouží pro sestavení simulované sítě. Využívá palety objektů, ze kterých volí potřebné komponenty sítě. Objekty se umístí do project editoru a propojují potřebnými linkami.

Další úrovní je **node editor**. Do něho se dostaneme rozkliknutím prvku v project editoru. Uvnitř je vidět vnitřní struktura síťových prvků, jejich funkce a vzájemné vztahy.

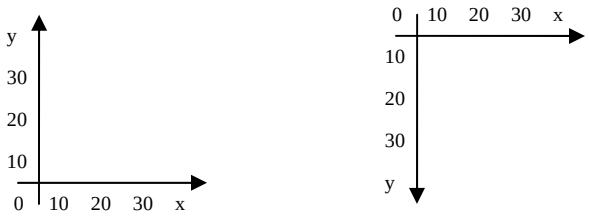
Posledním editorem je **process editor**. Ten je napsán v jazyce C/C++ a definuje pomocí konečného stavového automatu, jak se daný prvek chová.

Pro vytvoření podobných sítí s odlišnými konfiguracemi jsou využity tzv. **scénáře**, které jsou určeny pro simulaci různých architektur.

5.2 Simulace algoritmu v OPNETu

Výstupem programu uvedeného v kapitole 4 jsou souřadnice měřidel a koncentrátorů. Po spuštění programu dostaneme náhodné rozmístění měřidel a koncentrátorů. Po provedení optimalizační metody získáme nové souřadnice koncentrátorů a barevně rozlišená měřidla. Měřidlo, které je označeno stejnou barvou jako koncentrátor, náleží právě k tomuto koncentrátoru na základě optimalizačního algoritmu. Tyto výstupní hodnoty z výše uvedeného programu jsou vstupem pro simulaci v OPNETu. Jedinou nevýhodou je jiný souřadnicový systém v programovacím jazyce Java a v simulačním nástroji OPNET. Zatímco Java předpokládá počátek souřadnicového systému (tedy souřadnice 0,0) v levém horním

rohu pracovní plochy, OPNET Modeler obsahuje klasický souřadnicový systém, kdy jeho počátek (tedy souřadnice 0,0) je v levém spodním rohu. V jazyce Java tedy hodnota y-nových souřadnic roste směrem od shora dolů. Pro lepší orientaci je problém naznačen na Obr. 19. Tyto odlišnosti jsou sice matoucí, ale na výsledek programu nemají žádný vliv. Jen je třeba mít tento fakt na paměti při přenášení výsledného pospojování (barevného označení) do jiných prostředí.



Obr. 19: Souřadnicový systém a) v simulačním nástroji OPNET, b) v jazyce Java

5.2.1 Simulační model

Úkolem simulace je provést ověření funkčnosti optimalizačního algoritmu. Aby byla zajištěna srozumitelnost a názornost výsledků, je volen jednoduchý model. Model tedy není určen k simulování principů chytrého měření, ale jde o odzkoušení poznatků z teorie hierarchické agregace a optimalizačního algoritmu. Návrh srovnává výsledky simulace před použitím optimalizačního algoritmu a po jeho aplikování.

Sběrná síť se skládá ze tří prvků. Dříve v textu byly tyto prvky označovány jako měřidlo, koncentrátor a datová (sběrná) centrála. Protože OPNET neobsahuje takto označené a funkčně založené prvky, byly zvoleny komponenty s podobnou funkcí z nabídky OPNETu. OPNET slouží především pro simulování datových sítí a tak lze s výhodou použít prvky, které se jinak používají v datových sítích. Náhrada sice nebude naprosto přesná, ale pro odsimulování funkčnosti algoritmu z hierarchické agregace postačí.

Simulační model je tedy složen z měřidel, které v OPNETU reprezentují koncové stanice. Dále z koncentrátorů, které jsou reprezentovány přepínači a sběrnou centrálou. Vzájemné propojení je realizováno linkami, které mají přenosovou kapacitu 10 kbit/s. Jako zdroj dat byl zvolen FTP (File Transfer Protocol), který přenáší zprávy o velikosti 20 kbit. Počet komunikujících měřidel je zvolen na 90.

Po provedení výpočtu pomocí programu (viz. kap. 4), je stanoven počet koncentrátorů na 3. Z výstupu programu po provedení optimalizace získáme také vzájemné propojení jednotlivých koncentrátorů a měřidel.

Prvky jsou v modelu reprezentovány následujícími znaky:

měřidlo:

koncentrátor:

Aplication config:

sběrná centrála:

Profile config:

propojující linka:

koncentrátor – měřidlo:

koncentrátor – centrála:

Vlastní sestavování projektu začneme umístěním potřebného počtu měřidel (v OPNETu tedy uzlů anglicky: node) s označením **Sm_Int_wkstn**. Následně přidáme i koncentrátory (prvek je v OPNETu označen jako **3C_SSII_1100_3300_4s_ae52_e48_ge3**). Abychom si ulehčili práci a nemuseli přidávat jednotlivá měřidla jedno po druhém, je možné využít položku **Rapid Configuration** z nabídky **Topology -> Rapid Configuration**. Zde je možné zvolit styl topologie (v tomto případě **hvězda** (star)). Typ spojení mezi jednotlivými linkami zvolíme **ethetnet_adv**. Tento profil linky umožňuje nastavovat například propustnost, nebo i zpoždění charakteristické pro danou linku. Konfigurace přenosových kapacit a zpoždění se provádí opět v lokální nabídce (**Edit Attributes**) pro jednotlivé linky.

Za zmínku stojí také poměrně složité názvy jednotlivých prvků. Název každého z nich udává základní charakteristiku a vlastnosti pro daný prvek. Vezmeme-li si jako příklad označení **3C_SSII_1100_3300_4s_ae52_e48_ge3**, můžeme z něj odvodit: Prvek se skládá z 3Com SuperStack II 1100 a má dva Superstack II 3300. Obsahuje 52 ethernetových portů, které podporují auto-senging. Dále pak obsahuje 48 ethernetových portů a 3 ethernetové gigabitové porty [14].

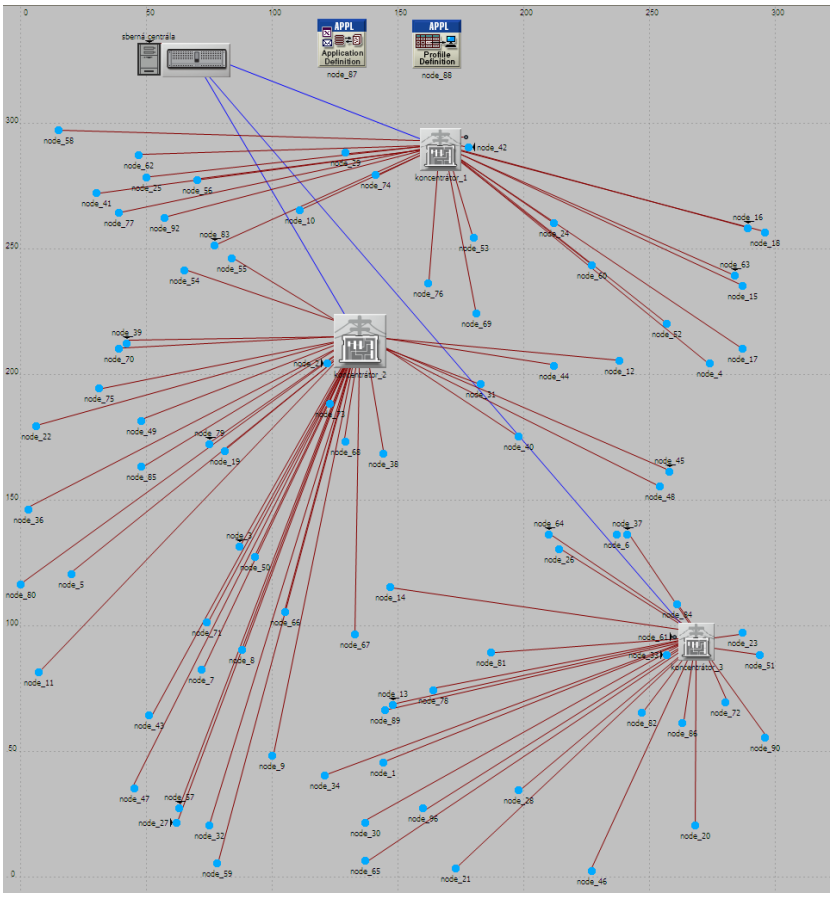
Dalším prvkem sítě bude sběrná centrála. Z pohledu OPNETu se bude jednat o klasický server s označením **Sm_Int_server**.

Pro nastavení datových přenosů mezi měřidly a datovou centrálou budeme využívat další dva prvky. Půjde o konfigurační objekt aplikací **Sm_Application_Config** a o objekt sloužící pro definici profilu daných aplikací **Sm_Profile_Config**. V aplikačním konfiguračním souboru nastavíme aplikace, které budeme v simulaci používat. V případě naší simulace postačí nastavit položku **Application Definitons** na hodnotu **Default**, což zajistí spuštění několika aplikací včetně FTP přenosu, který budeme využívat. A nyní již ke konfiguraci profilu aplikací v Profile Config. Zde konfigurujeme vlastnosti jednotlivých aplikací. Například jak často a v jakém čase se budou jednotlivé aplikace spouštět, jaká bude velikost přenášeného souboru a mnoho dalšího. Nastavíme potřebné parametry. V položce **Profile Configuration** v konfiguračním objektu profilů natavíme položku **Rows** na 1 a zvolíme parametry pro náš FTP přenos představující provoz ve sběrné síti. Nastavíme parametry spouštění a v položce aplikace (**Application**) nastavíme aplikace, které se v daném profilu budou uplatňovat. Nastavíme tedy opět **Row** na 1, jméno (**Name)** na **File Transfer** a spouštěcí čas (**Start Time)** nastavíme na konstantní (**constant**) s hodnotou například 100 vteřin.

Z hlediska datových přenosů je nastavení kompletní. Pro naši simulaci bude ještě potřeba nastavit přesné rozmístění jednotlivých koncentrátorů a měřidel. Z toho důvodu je v OPNETu velikost pracovní plochy zvolena na 300 x 300 km. Je to z proto, že program pro optimalizaci pracuje se stejným rozsahem souřadnic a tím pádem nenastává problém při přenosu výsledků z programu do simulačního prostředí.

Konfigurace souřadnic u jednotlivých koncentrátorů a měřidel je téměř totožná. Provádí se v položce **Edit Attributes**. Po zaškrtnutí položky **Advanced** se u objektu ukáže další možné nastavení, ve kterém lze navolit i souřadnice konkrétního umístění daného koncentrátoru nebo měřidla.

Výslednou strukturu simulované sítě ukazuje Obr. 20.



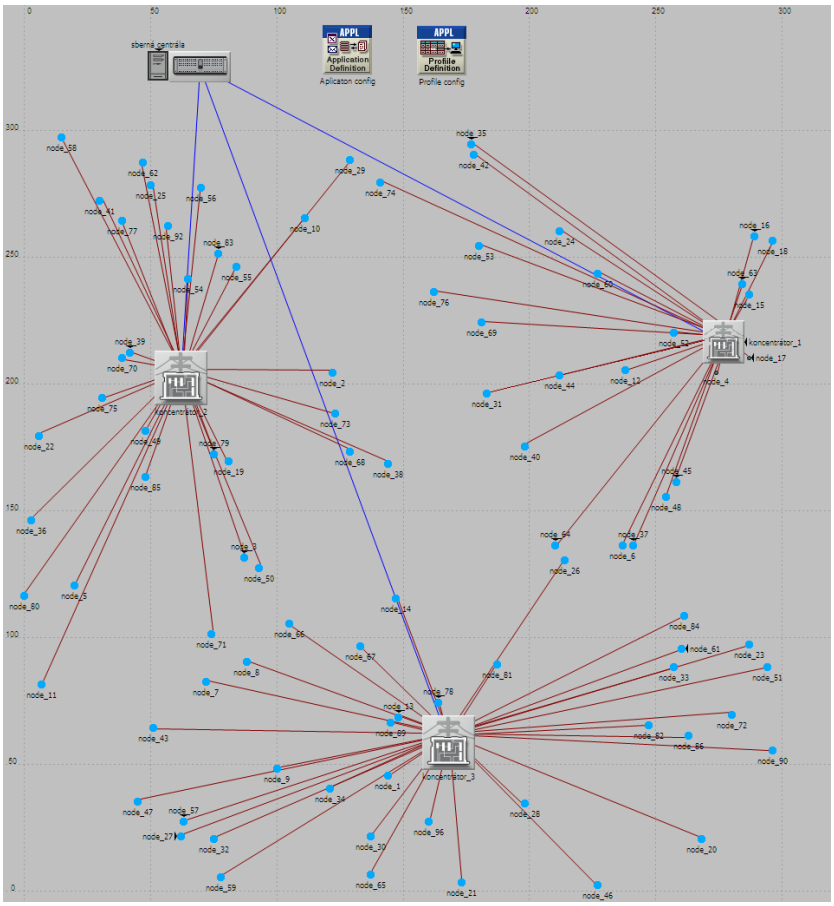
Obr. 20: Simulovaná síť bez optimalizace s náhodným pospojováním

Nedílnou součástí simulace je její výsledek. Před spuštěním simulačního procesu je nutné nastavit charakteristiky, které chceme sledovat. Celé nastavení se pro konkrétní prvek děje v kontextové nabídce pomocí položky **Choose Individual DES Statistics -> Node Statistics**. V kontextovém menu po kliknutí na pracovní plochu volbou položky **Choose Individual DES Statistics -> Global Statistics** volíme globální statistiky platné v celém projektu. V našem konkrétním případě jsme zvolili:

- zatížení datové centrály (bit)
- zatížení datové centrály (bit/s)
- přijatý provoz FTP (bajt/s)
- přeposílaný provoz na koncentrátorech (bit/s)
- využití linek (bit/s)

Následně již stačí spustit simulaci v nabídce **DES -> Configure/Run Discrete Event Simulation**, nastavit dobu simulace (v našem případě na 5 minut) a odstartovat stisknutím tlačítka **Run**.

Optimalizace se uskuteční přesunutím koncentrátorů do polohy, která odpovídá výstupu z programu na optimalizaci (je uveden v kap. 4). Přemístění koncentrátorů a jejich vzájemné propojení s měřidly je nutné provést podle výstupu z programu viz Obr. 18. Pro možnost jednoduchého porovnání je dobré zachovat předchozí model bez optimalizace a vytvořit nový do nového scénáře. To se provede v nabídce **Scenarios -> Duplicate Scenario**. Nyní budeme mít dva scénáře, jeden před optimalizací Obr. 20 a druhý po optimalizaci Obr. 21. Díky tomu bude snadné jejich vzájemné porovnání a zhodnocení. Částečné výsledky jsou uvedeny v kapitole 5.3.



Obr. 21: Simulovaná síť po optimalizaci

5.3 Výsledky simulace

Zobrazit výsledky simulace je možné v kontextové nabídce pracovní plochy OPNETu v položce **View Results**. Zde je také nastavení jednotlivých sledovaných statistik a konfigurace pro vykreslení grafů.

Simulované výsledky jsou zobrazeny na Obr. 22 až Obr. 26. Na Obr. 22 je znázorněn přijatý provoz FTP v celé síti. Modře je znázorněn průběh pro scénář bez optimalizace a červeně je průběh pro druhý scénář po optimalizaci. Je vidět, že došlo k zefektivnění. Osa *x* udává časový průběh simulace. Osa *y* přijatý provoz v bajtech/s.

Pokud z grafu odečteme hodnotu maxima modrého průběhu, získáme údaj 9 350 kB/s. Stejné maximum přečteme z grafu pro červený průběh, tedy 7 810 kB/s. Pokud z těchto hodnot vypočteme míru zefektivnění, dostaneme procentuální poměr 16,57 %. Tato míra zoptimalizování je dána přeuspořádáním koncentrátorů na základě výstupu ze sestaveného optimalizačního programu viz kapitola 4. Dále je z Obr. 18 vidět, že námi vypočítaná míra zefektivnění je téměř shodná s údajem zobrazeným v optimalizačním programu. Mírná odchylka může být způsobena nepřesným odečtem hodnot z grafu. Také je jasné, že simulační nástroj OPNET bere v úvahu i jiné proměnné (např. ztrátovost, zpoždění), než sestavený optimalizační program.

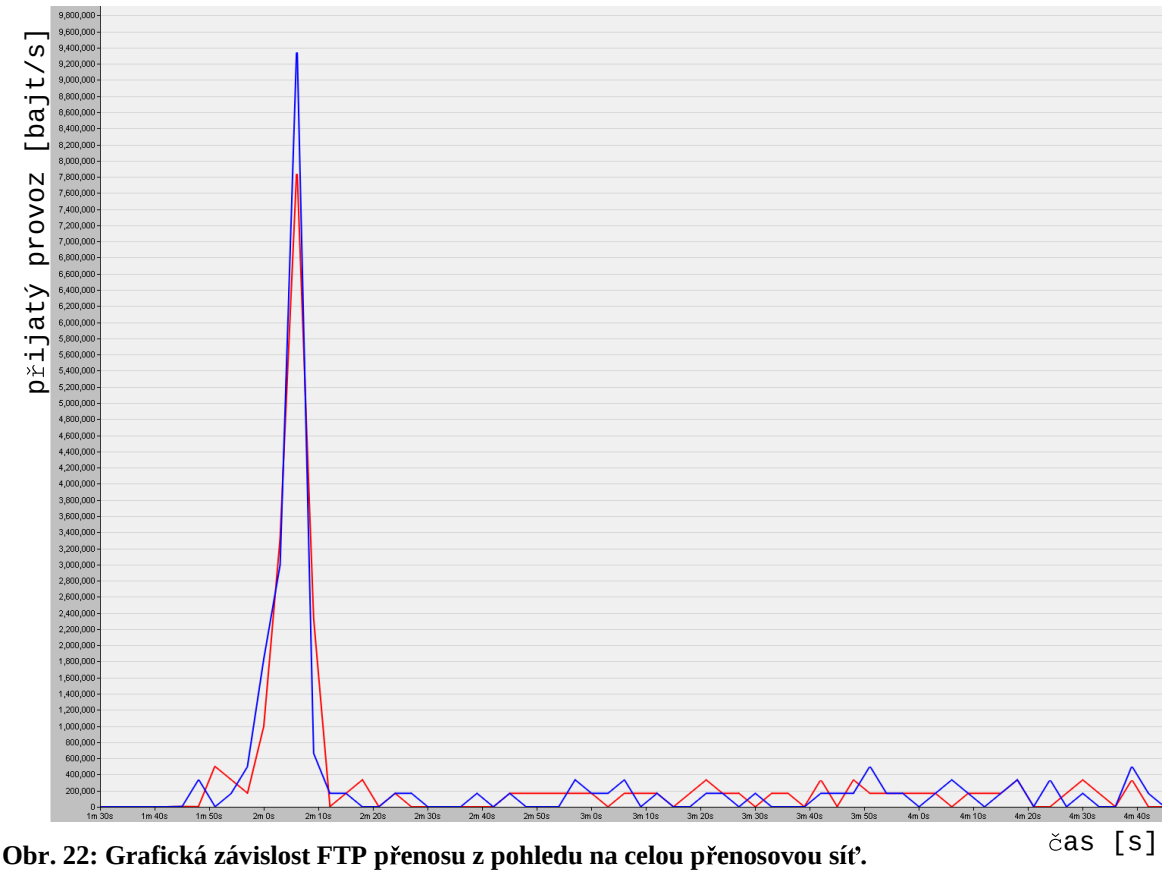
Další graf Obr. 23 ukazuje, jak se změní vytížení jednotlivých koncentrátorů po provedení optimalizace. Je zde vidět, jak se podle počtu připojených měřidel k danému koncentrátoru změní jeho odeslaný datový tok v bitech za sekundu. Na prvních dvou koncentrátorech se datový tok sníží, ke třetímu koncentrátoru je po optimalizaci připojeno více měřidel a proto na něm provoz vzroste.

Třetí Obr. 24 znázorňuje zpoždění na lince mezi jedním z koncentrátorů a sběrnou centrálou. I když v jedné části charakteristiky zpoždění naroste, výsledné celkové zpoždění se jeví jako nižší.

Následující Obr. 25 popisuje také linku mezi koncentrátorem a datovou centrálou. Je na něm zachyceno využití dané linky. Hodnota využití je v procentech. Opět platí, že modrý průběh je charakteristika před optimalizací a červený průběh je charakteristika po optimalizaci.

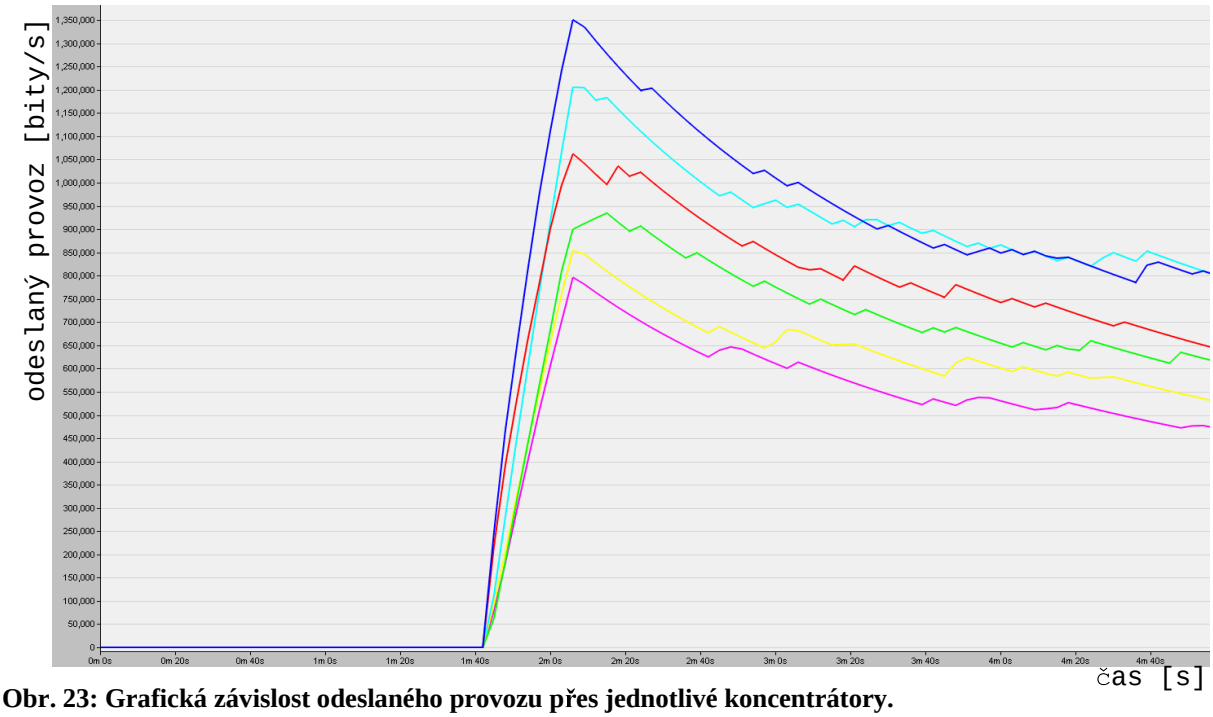
Posledním zkoumaným průběhem je charakteristické zatížení rozhraní datové (sběrné) centrály. Jak je vidět, zatížení po optimalizaci kleslo (červený průběh) oproti zatížení před optimalizací (modrý průběh). Tento stav zobrazuje Obr. 26.

Samozřejmě by bylo možné sledovat další různé charakteristiky. Možnosti OPNETu jsou v tomto ohledu velice široké. Uvedené grafy jsou nejvíce vypovídající o dané simulované síti a nejlépe zobrazují zefektivnění komunikace při použití optimalizačního programu.



Obr. 22: Grafická závislost FTP přenosu z pohledu na celou přenosovou síť.

Modře průběh v síti bez optimalizace. Červený průběh v optimalizované síti

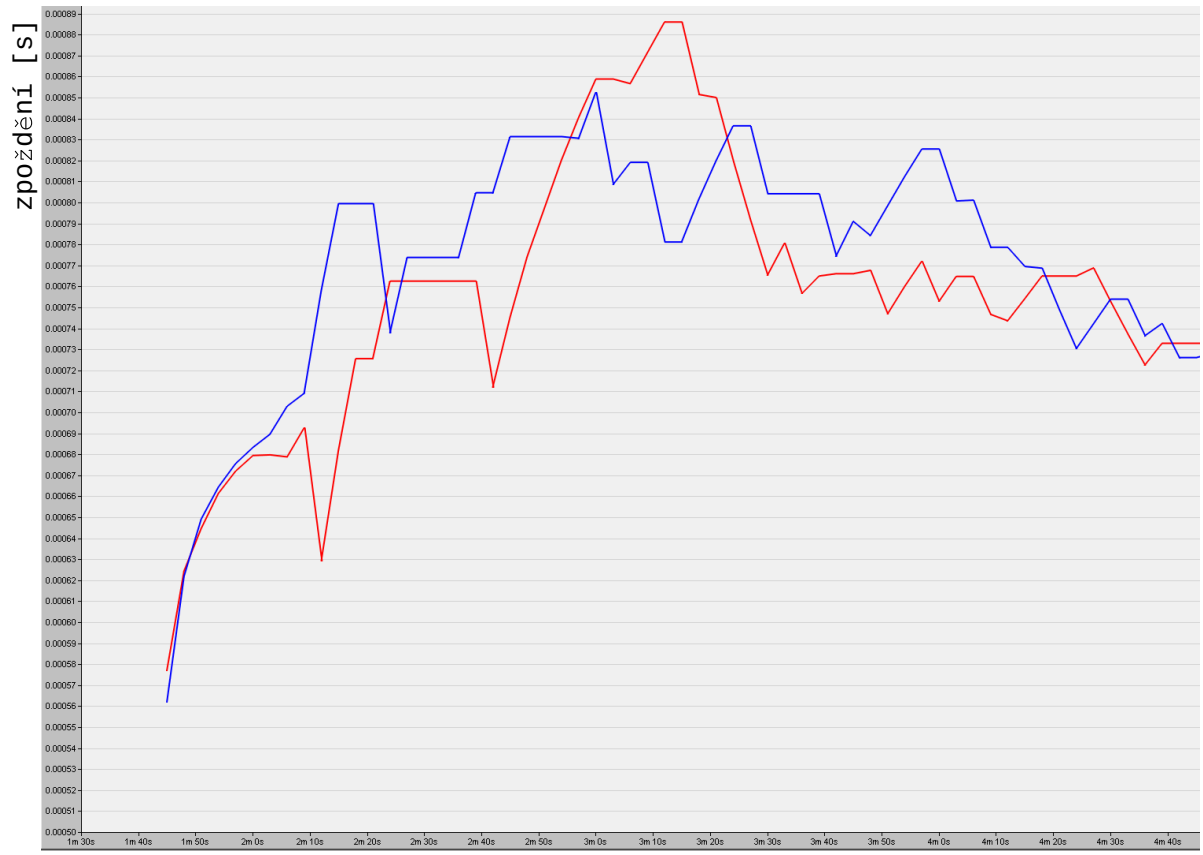


Obr. 23: Grafická závislost odeslaného provozu přes jednotlivé koncentrátory.

Koncentrátor 1: žlutá – bez optimalizace; fialová – po optimalizaci

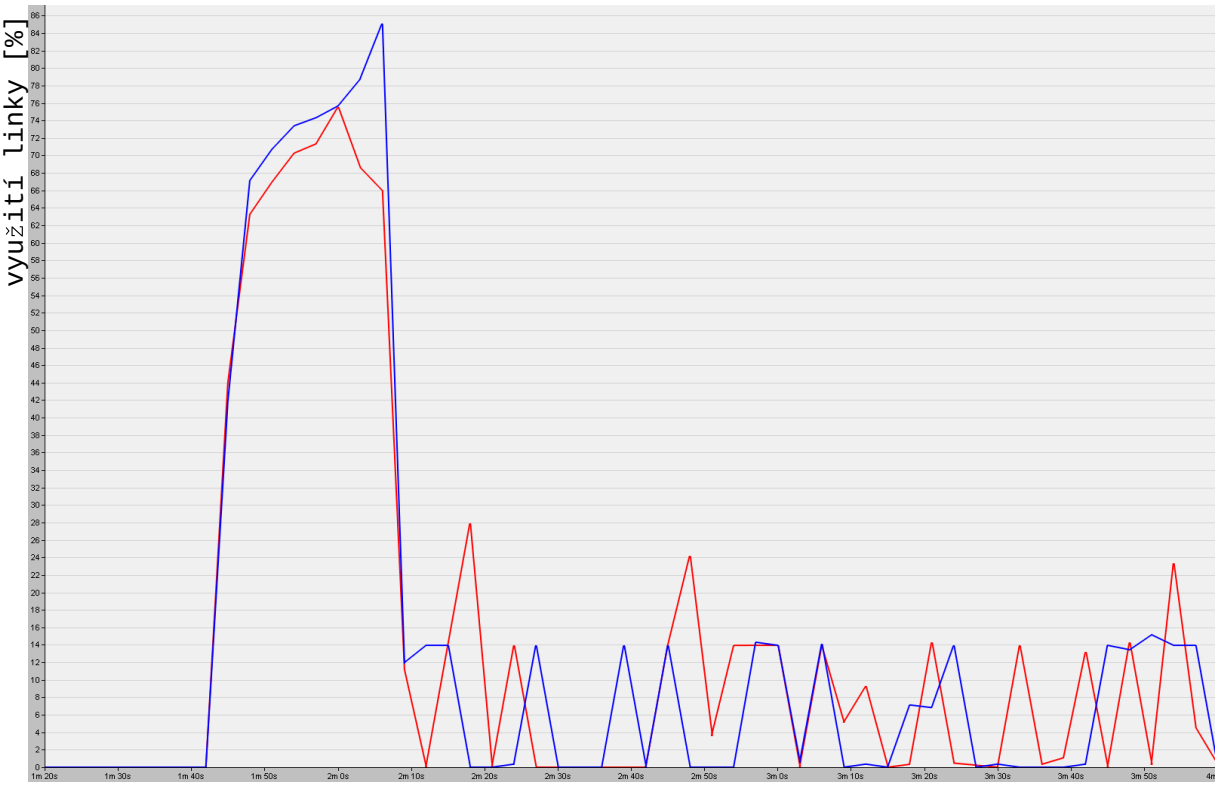
Koncentrátor 2: tmavě modrý – bez optimalizace; červený – po optimalizaci

Koncentrátor 3: zelená – bez optimalizace; světle modrá – po optimalizaci



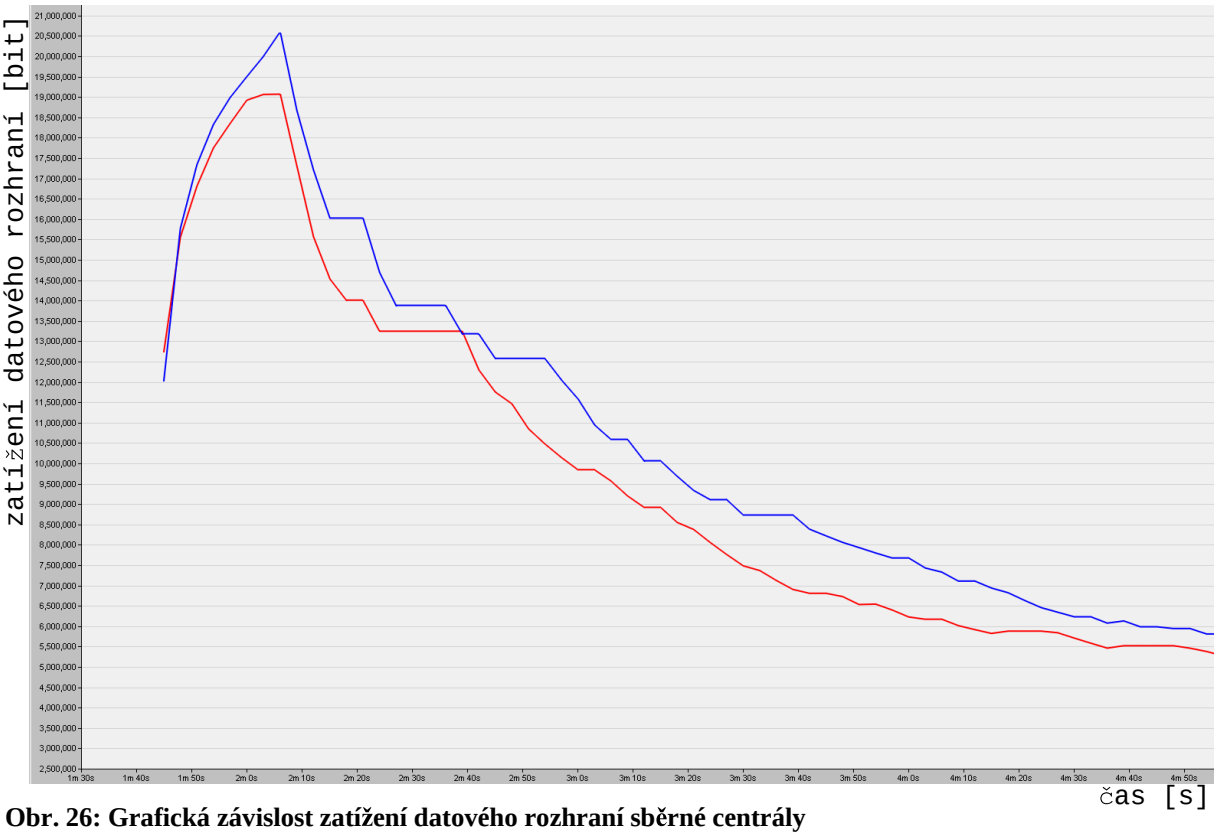
Obr. 24: Grafická závislost zpoždění na lince mezi koncentrátorem a sběrnou centrálou.

Zpoždění na lince v zapojení bez optimalizace (modrá) a po optimalizaci sítě (červená).



Obr. 25: Grafická závislost využití linky mezi koncentrátorem a sběrnou centrálou

Modrý průběh bez optimalizace. Červený po optimalizaci.



Obr. 26: Grafická závislost zatížení datového rozhraní sběrné centrály

v neoptimalizované síti (modře) a v optimalizované (červeně).

6 Závěr

V následujícím vývoji distribučních sítí bude téměř nutností zavádět systémy chytrého měření a chytrých sítí. Na jednu stranu je pravda, že doposud lidstvo podobné platformy vůbec nepotřebovalo a veškerá distribuce energií probíhala relativně spolehlivě bez velkých výpadků a poruch. Na druhou stranu se spotřeba energií neustále zvyšuje a rostou i ohlasy na ekologicky čistou energii vyráběnou z obnovitelných zdrojů. Ovšem právě z důvodu připojování solárních, větrných a podobných zdrojů energie do přenosové soustavy se vytváří napěťové špičky. Ty jsou ve většině případů problematické, protože elektřinu nelze skladovat.

Proto vývoj směřuje k nejrůznějším dynamickým elektrickým tarifům a motivaci zákazníka, aby přizpůsobil svoji spotřebu aktuální výrobě. Z toho vyplývá potřeba řídit distribuční síť téměř v reálném čase. Tím ovšem rostou požadavky na přenosové rychlosti komunikace a množství přenášených dat mezi distributorem a měřidlem u zákazníka.

Práce je zaměřena na problematiku sběru dat z velkého počtu měřicích zařízení umístěných na rozsáhlém území. Okruh je směrován na „chytré“ systémy měření, jejich vlastnosti a budoucí perspektivu. Jsou zde uvedeny základní pojmy z oblasti Smart metering, jako jsou AMR, AMM, AMI. Pro nahlédnutí do budoucnosti slouží kapitola týkající se Smart grids. V ní jsou uvedeny poznatky, jak by distribuční soustava mohla vypadat za několik let.

Současnou situaci charakterizují pilotní projekty velkých firem (EON, ČEZ, Západoslovenská energetika, atd.) a jejich průběžné výsledky.

Se sběrem dat z velkého počtu odběrných míst souvisí i přenášení informací. Pro přenos se v oblasti chytrého měření využívají technologie GSM, GPRS, Internet, PLC, atd. Krátká charakteristika těchto technologií a celkové uspořádání systému Smatr meter je popsáno v kapitole 2.

V následující části je uveden matematický pohled na systémy sběru dat. Pro popis se podle mého názoru hodí teorie grafů. Základní pojmy z této oblasti jsou uvedeny na začátku třetí kapitoly. Výsledná sběrná síť má charakter stromu. Úvaha, jak zefektivnit výměnu informací, aby nedocházelo zbytečně k přetěžování datových spojů, je naznačena v závěru třetí kapitoly. Ta popisuje hierarchickou agregaci, její možnosti a uplatnění ve sběrných sítích.

Získané poznatky o sběrných sítích a hierarchické agregaci jsou aplikovány do navrhovaného programu pro optimalizaci. Jeho funkcí je najít nejen optimální počet koncentrátorů v dané skupině měřidel, ale především jejich ideální polohu. Program je napsán v programovacím jazyce Java.

Pro ověření funkčnosti sestaveného programu je v závěru práce provedena simulace v nástroji OPNET Modeler. Základní informace o problematice modelování sítí jsou shrnuty v úvodu kapitoly 5. Je zde také zařazen krátký popis simulačních nástrojů. Nejvíce vyhovujícím se zdá být OPNET Modeler, který je pro testování míry zefektivnění názorný a jeho ovládání jednoduché.

Z výsledků získaných ze simulace je patrné, že po použití sestaveného programu, dojde k zefektivnění komunikace mezi měřidly a datovou centrálou. Tím pádem je zajištěna větší propustnost datových spojů a lze tedy obsloužit více měřidel, popřípadě provádět odečty a řízení sítě častěji.

Použité zdroje:

[1] KONFERENCE Smart metering. Říjen 2010, Praha

[2] DUCHEK, Pavel. Půlstoletí HDO. *3pól : Magazín plný pozitivní energie* [online]. 3.3.2010, 3, [cit. 2010-11-23]. Dostupný z WWW: <http://3pol.cz/892-pulstoleti-hdo>.

[3] KOZUBÍK, Libor Centrální systém AMM v prostředí utilitní společnosti. In *IBM Utility Seminar* [online]. IBM: IBM Cororation, 2008 [cit. 2010-11-23]. Dostupné z WWW:<http://www-05.ibm.com/sk/events/presentations/resources/Centralni_system_AMM_v_pro-stredi_utilitni_spolecnosti_Kozubik.ppt>.

[4] *Plc.cz* [online]. 2008 [cit. 2010-11-23]. Dostupné z WWW: <www.plc.cz>.

[5] HYNČICA, Ondřej. Bezdrátové sítě typu MESH. In ČAPEK, Josef. *AUTOMA: Časopis pro automatizační techniku* [online]. Brno: Automa, 2005 [cit. 2010-11-23]. Dostupné z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=30826>.

[6] DEMEL, Jiří. *Grafy a jejich aplikace*. 1. vydání. Praha: Akademie věd České republiky, 2002. 257 s. ISBN 80-200-0990-6, 1567.

[7] ZEMAN, Václav. *Power Line Communication*. *Přednášky*. Brno: VUT, 2010, 7, s. 3-34.

[8] *Technické požadavky* [online]. - [cit. 2010-12-05]. ELPRO - DELICIA, a.s. Dostupné z WWW: <http://www.elpro-delicia.cz/de_pr5_3.htm>.

[9] NOVOTNÝ, Vít. *Komunikační prostředky mobilních sítí*. Brno: VUT, 2006. 92 s.

[10] JEŘÁBEK, Jan. *Pokročilé komunikační techniky*. Brno: VUT, 2010. 247 s.

[11] ŠKORPIL, Vladislav; KAPOUN, Vladimír; GREGOŘICA, Miroslav. *Přístupové a transportní sítě*. Brno : VUT, 2004. 160 s.

[12] Müller, J.; Komosný, D.; Burget, R.; aj.: Advantage of Hierarchical Aggregation. 2008, international Journal of Computer Science and Network Security, 2008, roč. 2008, č. 8, s. 1-7. ISSN: 1738-7906.

[13] ŠOCHMAN, Jan. *Shlukování k-means* [online]. 7. prosince 2005 [cit. 2011-03-16]. Cvičení z RPZ. Dostupné z WWW: <http://cmp.felk.cvut.cz/cmp/courses/recognition/Labs/kmeans/kmeans.pdf>.

[14] MOLNÁR, Karol; ZEMAN, Otto; SKOŘEPA, Michal. *Moderní síťové technologie: Laboratorní cvičení* [online]. Brno: VUT v Brně, Fakulta elektrotechniky, Ústav telekomunikací, 2008 [cit. 2011-04-19]. Dostupné z WWW: <http://www.utko.feec.vutbr.cz/~molnar/mmos/MMOS_lab.pdf>.

[15] *TIOBE Programming Community Index for May 2011* [online]. 2011 [cit. 2011-05-06]. TIOBE sofrware. Dostupné z WWW: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

[16] SÝKORA, Tomáš. *Kvalita elektrické energie – Signál HDO : Přednáška z předmětu Distribuce elektrické energie* [online]. Praha : 12.4.2006 [cit. 2011-04-18]. České vysoké učení technické v Praze. Dostupné z WWW: <http://k315.feld.cvut.cz/download/dee/signal_HDO.pdf>.

[17] *JavaTM 2 Platform* [online]. Std. Ed. v1.4.2. 2010 [cit. 2011-05-1]. Manuál Java. Dostupné z WWW: <http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html>.

Seznam zkratek

ADSL	Asymmetric Digital Subscriber Line
AMI	Advanced Metering Infrastructure
AMM	Automatic Meter Management, Advanced Metering Management
AMR	Automatic Meter Reading
BPL	BroadBand over PowerLine
BSC	Base Station Controller
BTS	Base Transceiver Station
CS	Coding Scheme
CSD	Circuit Switched Data
DLS	Digital Subscriber Line
EDGE	Enhanced Data rates for GSM Evolution
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GNU	General Public License
GPRS	General Packet Radio System
GSM	Global System for Mobile communications
HA	Hierarchické Agregace
HDO	Hromadného Dálkového Ovládání
HDSL	High bit rate Digital Subscriber Line
HSCSD	High Speed Circuit Switched Data
IP	Internet Protocol
ISDN	Integrated Services Digital Network
IWF	InterWorking Function
JRE	Java Runtime Environment
MSC	Mobile Switching Centre
NS2	Network Simulator
OOP	Objektově Orientovaný Programovací jazyk
OPERA	Open PLC European Research Alliance
OS	Operační Systém
PCU	Packet Control Unit
PDSL	Power line Digital Subscriber Line
PLC	Power Line Communication
PLN	Power Line Networking
PLT	Power Line Telecom
RF	Radio Frequency
SDSL	Symmetric Digital Subscriber Line
SMS	Short Message Service
SSGN	Serving GPRS Support Node
TCP/IP	Transmission Control Protocol/Internet Protocol
UMTS	Universal Mobile Telecommunication System
VDSL	Very high speed Digital Subscriber Line

Seznam veličin a symbolů

$a(x, y)$	délka hrany z vrcholu x do vrcholu y
BW	šířka pásma přenosových cest [b/s]
E	množina hran grafu
e	hrana
G	matematická struktura - graf
KON_1	udává počet koncentrátorů v první vrstvě agregačního stromu [-]
KON_2	udává počet koncentrátorů v druhé vrstvě [-]
$Kv(e)$	koncový vrchol hrany e
M	množina vrcholů u nichž došlo ke změně délky
N	počet připojených měřidel [-]
N_j	vzorky náležící k danému vektoru μ_j
O	množina hran, které způsobily snížení vzdálenosti
$Pv(e)$	počáteční vrchol hrany e
r	kořen stromu - root
S	označuje strom
T_1	perioda odesílání zpráv [s]
T_2	perioda odesílání zpráv v druhé vrstvě [s]
$U(x)$	délka doposud nejkratší cesty z vrcholu r do vrcholu x
$U(y)$	délka nové nejkratší cesty z vrcholu r do vrcholu y
V	množina vrcholů grafu
V	udává množinu vrcholů ve stromě
v	vrchol
x_i	množina bodů
y_i	minimum euklidovské vzdálenosti
Z_1	délka přenášené zprávy [b]
Z_2	délka přenášené zprávy v druhé vrstvě [b]
ε	zobrazení v grafu
μ_j	vektory

Seznam příloh

Příloha A:

Třída Main.java

```
public class Main {                          //hlavní třída
    public static void main(String[] args) {
        Gui gui = new Gui();                 //po spuštění zviditelní ...
        gui.setVisible(true);                //... grafické rozhraní
    }
}
```

Třída Gui.java

```
package optimalizacestromu;

import java.awt.Color;

public class Gui extends javax.swing.JFrame {
    int sirka = 20;                          //velikost symbolů měřidel a koncentrátorů
    int vyska = 20;
    int stisk = 0;                           //zda bylo stisknuto tlačítko Optimalizace stromu
    int klik = 0;                            //zda bylo stisknuto tlačítko Jeden krok k-means
    double delkaPred;                        //pro výpočet efektivity
    double delkaPo;
    Integer zmena = 0;                       //zda v posledním kroku proběhla změna
    Integer[] xCislaM;                       //pro vygenerování náhodných čísel pro souřadnice
    Integer[] yCislaM;                       //měřidel
    Integer[] xCislaK;                       //koncentrátorů
    Integer[] yCislaK;
    Meridlo[] meridlo;                       //pole proměnných s datovou strukturou Meridlo
    Koncentrator[] koncentrator;             //-"- pro koncentrátor
    javax.swing.JRadioButton jRadioMer[];    //pole objektů JRadioButton (pro zobrazení měřidel)
    javax.swing.JButton jButKon[];           //-"- pro koncentrátory

    public Gui() {                           //konstruktor
        initComponents();                    //inicializuje veškeré komponenty
    }

    public void kMeans(int kroku) {          //metoda s algoritmem k-means (vztahy (9) a (10)
        if (kroku == 0) {    //první průběh algoritmu k-means (pouze označí nejbližší měřidla)
            zmena = 1;
            delkaPred = 0;
            int[] mezi = new int[koncentrator.length];     //proměnná pro uložení vzdáleností

            for (int j = 0; j < meridlo.length; j++) {
                      //dočasná proměnná pro porovnání (inicializační hodnota musí být vysoká)
                int pom = 10000;

                                        //hledá nejbližší koncentrator k danému měřidlu
                for (int i = 0; i < koncentrator.length; i++) {
                    mezi[i] = (int) Math.sqrt(Math.pow(meridlo[j].getX() -
koncentrator[i].getX(), 2) + Math.pow(meridlo[j].getY() - koncentrator[i].getY(), 2));
                }
                for (int i = 0; i < mezi.length; i++) {
                    if (mezi[i] < pom) {
                        pom = mezi[i];             //porovná vzdálenosti a najde nejbližší
                    }
                }

                for (int i = 0; i < mezi.length; i++) {
                    if (pom == mezi[i]) {
                        meridlo[j].setBarva(i);    //přiřadí každému měřidlu barvu
                                                   //a uloží délku pro výpočet optimalizačního poměru
                        delkaPred = delkaPred + mezi[i];
                    }
                }
            }
            for (int j = 0; j < meridlo.length; j++) {
                for (int i = 0; i < koncentrator.length; i++) {
                    if (meridlo[j].getBarva() == i) {   //nastaví barvu dané grafické ikoně
                        jRadioMer[j].setBackground(koncentrator[i].getColor());
                    }
                }
```

```
            }

        } else {        //pokud prochází program druhým cyklem, provede se vlastní algoritmus
            int[] mezi = new int[koncentrator.length];          //mezi výsledek
                                        //pro uložení průběžných výsledků nových souřadnic
            int[][] soucet = new int[koncentrator.length][3];
            zmena = 0;
            delkaPo = 0;                    //pro výpočet míry zefektivnění

            for (int j = 0; j < meridlo.length; j++) {  //stejné jako v prvním cyklu
                int pom = 10000;

                                    //hledá nejbližší koncentrátor k danému měřidlu
                for (int i = 0; i < koncentrator.length; i++) {
                    mezi[i] = (int) Math.sqrt(Math.pow(meridlo[j].getX() -
koncentrator[i].getX(), 2) + Math.pow(meridlo[j].getY() - koncentrator[i].getY(), 2));
                }
                for (int i = 0; i < mezi.length; i++) {
                    if (mezi[i] < pom) {
                        pom = mezi[i];
                    }
                }

                for (int i = 0; i < mezi.length; i++) {
                    if (pom == mezi[i]) {
                        meridlo[j].setBarva(i);
                        delkaPo = delkaPo + mezi[i];
                    }
                }
            }
            for (int j = 0; j < meridlo.length; j++) {
                for (int i = 0; i < koncentrator.length; i++) {
                    if (meridlo[j].getBarva() == i) {
                                                    //měřidlo se označí příslušnou barvou
                        jRadioMer[j].setBackground(koncentrator[i].getColor());
                        soucet[i][0] = soucet[i][0] + meridlo[j].getX();   //uloží se pozice X
                                                        //měřidla pro daný koncentrátor i
                        soucet[i][1] = soucet[i][1] + meridlo[j].getY();        //pro pozici Y
                        soucet[i][2]++;         //počítá počet měřidel pro daný koncentrátor i
                    }
                }
            }
            for (int i = 0; i < koncentrator.length; i++) {
                if (soucet[i][2] == 0) //pro případ,že by vyšel počet koncentrátorů roven nule
                    soucet[i][2] = 1;
                }
            }
            for (int i = 0; i < koncentrator.length; i++) {     //testuje, zda došlo ke změně
                if (koncentrator[i].getX() != (soucet[i][0] / soucet[i][2]) ||
koncentrator[i].getY() != (soucet[i][1] / soucet[i][2])) {
                    koncentrator[i].setX(soucet[i][0] / soucet[i][2]);          //pokud ano,
                                                            //vypočítá nové souřadnice X
                    koncentrator[i].setY(soucet[i][1] / soucet[i][2]);          //a Y
                    zmena++;                        //nastaví se příznak, že proběhla změna
                }
                jButKon[i].setBounds(koncentrator[i].getX(), koncentrator[i].getY(), sirka,
vyska);
            }
            jTextAreaSourKonNove.setText("");
            for (int i = 0; i < koncentrator.length; i++) {       //vypíše souřadnice
                jTextAreaSourKonNove.append(koncentrator[i].getX() + ", " +
koncentrator[i].getY() + "\n");
            }
        }
    }
                                        //generátor náhodných čísel (pro souřadnice)
    private void nahodnacisla(int pocetM, int pocetK) {
        xCislaM = new Integer[pocetM];
        yCislaM = new Integer[pocetM];
        xCislaK = new Integer[pocetK];
        yCislaK = new Integer[pocetK];
        for (int i = 0; i < pocetM; i++) {
            xCislaM[i] = (int) ((Math.random() * 1000) % 300);  //omezený rozsah do 300
            yCislaM[i] = (int) ((Math.random() * 1000) % 300);
        }
        for (int i = 0; i < pocetK; i++) {
```

```
            xCislaK[i] = (int) ((Math.random() * 1000) % 300);
            yCislaK[i] = (int) ((Math.random() * 1000) % 300);
        }
    }

    @SuppressWarnings("unchecked")
        //grafické komponenty
               ...
        kompletní kód je uveden v příloze C
        ...

        //grafické komponenty

    private void jButtonNahodRozActionPerformed(java.awt.event.ActionEvent evt) {
        try {                                 //stisk tlačítka Náhodné rozmístění
            jTextAreaSourKonNove.setText("");
            jLabelChyba.setText("");
            jLabelOptimalAbs.setText("");   //Vymaže pole zobrazená v předchozím běhu programu
            jLabelOptimalProc.setText("");  //... pro případ, že není program spuštěn poprvé
            jLabelPocetKroku.setText("");
            stisk = 0;
            klik = 0;
            jLabelPocetKroku.setText("");
            jLabelChyba.setText("");
            jPanel1.removeAll();                    //odstraní grafické komponenty
                                                    //načte počet měřidel a koncentrátorů
            int pocetM = Integer.parseInt(jTextFieldPocetMeridel.getText());
            int pocetK = Integer.parseInt(jTextFieldPocetKoncentratoru.getText());

            if (pocetM <= 0 || pocetK <= 0) {                       //ošetření chybného zadání
                jLabelChyba.setText("Hodnoty musí být kladné a větší než nula!");
            } else {
                nahodnacisla(pocetM, pocetK);       //generuje náhodná čísla pro souřadnice
                meridlo = new Meridlo[pocetM];      //nové pole objektů ze třídy měřidel
                koncentrator = new Koncentrator[pocetK];

                for (int i = 0; i < pocetM; i++) {     //naplní objekty náhodnými souřadnicemi
                    meridlo[i] = new Meridlo(i, xCislaM[i], yCislaM[i]);
                }

                for (int i = 0; i < pocetK; i++) {         //-"- pro koncentrátory
                    koncentrator[i] = new Koncentrator(i, xCislaK[i], yCislaK[i]);
                }
                                                        //vytvoří instance grafických ikon
                jRadioMer = new javax.swing.JRadioButton[meridlo.length];
                jButKon = new javax.swing.JButton[koncentrator.length];

                                            //nastavení vlastností grafických ikon měřidel
                for (int i = 0; i < jRadioMer.length; i++) {
                    jRadioMer[i] = new javax.swing.JRadioButton();  //typ ikony
                    jPanel1.add(jRadioMer[i]);            //přidáno do zobrazovacího panelu
                    jRadioMer[i].setVisible(true);        //zviditelnění
                    jRadioMer[i].setSelected(true);       //pouze styl, jakým se má zobrazovat
                    jRadioMer[i].setBackground(Color.gray);     //barva komponenty
                }
                for (int i = 0; i < jButKon.length; i++) {      //to stejné pro koncentrátory
                    jButKon[i] = new javax.swing.JButton();
                    jPanel1.add(jButKon[i]);
                    jButKon[i].setVisible(true);
                                                                //generátor náhodných barev
                    Color color = new Color((int) (Math.random() * 1000) % 255, (int)
(Math.random() * 1000) % 255, (int) (Math.random() * 1000) % 255);
                    jButKon[i].setBackground(color);              //obarvení komponenty barvou
                    koncentrator[i].setColor(color);              //uložení dané barvy
                }
                                                    //zobrazí symbol na daných souřadnicích
                for (int i = 0; i < jRadioMer.length; i++) {
                    jRadioMer[i].setBounds(meridlo[i].getX(), meridlo[i].getY(), sirka, vyska);
                }

                for (int i = 0; i < koncentrator.length; i++) { //-"- pro koncentrátory
                    jButKon[i].setBounds(koncentrator[i].getX(), koncentrator[i].getY(), sirka,
vyska);
                }

                jTextAreaSouradniceMeridel.setText("");
```

```
                for (int i = 0; i < pocetM; i++) {              //výpis souřadnic měřidel
                    jTextAreaSouradniceMeridel.append(meridlo[i].getX() + ", " +
meridlo[i].getY() + "\n");
                }

                jTextAreaSourKonPuvodni.setText("");       //výpis souřadnic koncentrátorů
                for (int i = 0; i < koncentrator.length; i++) {
                    jTextAreaSourKonPuvodni.append(koncentrator[i].getX() + ", " +
koncentrator[i].getY() + "\n");
                }
                jPanel1.revalidate();       //pro uplatnění nových změn v grafickém prostředí
                jPanel1.repaint();
            }
        } catch (Exception e) {                                 //ošetření výjimek
            jLabelChyba.setText("Chybně zadané vstupní parametry!");
        }
    }

    private void jButtonOptimalizaceActionPerformed(java.awt.event.ActionEvent evt) {
        try {                               //stisk tlačítka Optimalizuj
            if (stisk == 0) {
                Integer kroku = 0;
                do {
                    kMeans(kroku);          //provádí metodu kMeans
                    kroku++;                //po každém průchodu zvýší počet kroků
                } while (zmena != 0);       //dokud dochází ke změnám
                stisk++;                    //vypíše počet kroků
                jLabelPocetKroku.setText(Integer.toString(kroku - 2));
                                            //spočítá absolutní míru optimalizace
                jLabelOptimalAbs.setText(Double.toString(delkaPred - delkaPo));
                jLabelOptimalProc.setText(Double.toString(Math.round((1 - (delkaPo /
delkaPred)) * 100)) + " %");                //relativní
            }
        } catch (Exception e) {             //ošetření nestandardních stavů
            jLabelChyba.setText("Nejprve je nutné provést výpočet a náhodné rozmístění!");
        }
    }

    private void jButtonVypocetActionPerformed(java.awt.event.ActionEvent evt) {
                                                //stisknuto tlačítko Vypočítej
        Integer kon = 0;
        try {
            jLabelPocetKroku.setText("");           //vymaže pole z předchozího běhu programu
            jLabelChyba.setText("");
            jLabelOptimalAbs.setText("");
            jLabelOptimalProc.setText("");
            jTextAreaSourKonNove.setText("");
            jTextAreaSourKonPuvodni.setText("");
                                                //načte zadané hodnoty z textových oblastí
            int meridel = Integer.parseInt(jTextFieldPocetMeridel.getText());
            int delkaZpravy = Integer.parseInt(jTextFieldDelkaZpravy.getText());
            int perioda = Integer.parseInt(jTextFieldPeriodaZpravy.getText());
            int sirkaPasma = Integer.parseInt(jTextFieldSirkaPasma.getText());

            if (meridel <= 0 || delkaZpravy <= 0 || perioda <= 0 || sirkaPasma <= 0) {
                                            //zkontroluje, zda nejsou vstupní hodnoty záporné
                jLabelChyba.setText("Hodnoty musí být kladné a větší než nula!");
                                                        //pokud ano, vypíše chybu
            } else {
                kon = (meridel * delkaZpravy * 1000) / (perioda * sirkaPasma * 1000);
                if (kon == 0) {
                    kon = 1;
                }                               //a uloží ho do textového pole
                jTextFieldPocetKoncentratoru.setText(kon.toString());
            }
        } catch (Exception e) {                 //ošetření výjimek
            jLabelChyba.setText("Chybně zadané vstupní parametry!");
        }
    }

    private void jButtonKrokActionPerformed(java.awt.event.ActionEvent evt) {
                                            //stisknuto tlačítko Jeden krok k-means
        try {
            kMeans(klik);                   //provede kMeans pokud:
            if (zmena != 0) {               //došlo v předchozím kroku ke změně
                klik++;
```

```
            }
            jLabelPocetKroku.setText(Integer.toString(klik - 1));
            if (klik != 1) {                                //výpis efektivity optimalizace
                jLabelOptimalAbs.setText(Double.toString(delkaPred - delkaPo));
jLabelOptimalProc.setText(Double.toString(Math.round((1 - (delkaPo / delkaPred)) * 100)) + "
%");
            }
        } catch (Exception e) {                 //ošetření výjimky
            jLabelChyba.setText("Nejprve je nutné provést výpočet a náhodné rozmístění!");
        }
    }

    private void jButtonKonecActionPerformed(java.awt.event.ActionEvent evt) {
                                                    //stisknuto tlačítko Konec
        System.exit(0);                             //Ukončení programu
    }

    private void jButtonInfoActionPerformed(java.awt.event.ActionEvent evt) {
                                                //stisknuto tlačítko ? - informace
       JOptionPane.showMessageDialog(null, "O programu\nAutor: Lukáš Hudec\nDiplomová práce
       2011: Systémy dálkového měření v energetice\n\nOvládání:\n1) Nastavit vstupní
       parametry.\n2) Stisknout tlačítko Vypočítej - vypočítá počet koncentrátorů.\n3)
       Stisknout tlačítko Náhodné rozmístění - náhodně rozmístí měřidla a koncentrátory.\n4)
       Stisknout tlačítko Optimalizuj nebo Jeden krok k-means - provede optimalizaci.\n5)
       Odečíst výsledky optimalizace.","O programu",JOptionPane.INFORMATION_MESSAGE);
    }

    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new Gui().setVisible(true);             //inicializace grafického rozhraní
            }
        });
    }
    // deklarace proměnných
    private javax.swing.JButton jButtonKonec;
    private javax.swing.JButton jButtonKrok;
    private javax.swing.JButton jButtonNahodRoz;
    private javax.swing.JButton jButtonOptimalizace;
    private javax.swing.JButton jButtonVypocet;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JLabel jLabelChyba;
    private javax.swing.JLabel jLabelOptimalAbs;
    private javax.swing.JLabel jLabelOptimalProc;
    private javax.swing.JLabel jLabelPocetKroku;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane5;
    private javax.swing.JScrollPane jScrollPane6;
    private javax.swing.JTextArea jTextAreaSourKonNove;
    private javax.swing.JTextArea jTextAreaSourKonPuvodni;
    private javax.swing.JTextArea jTextAreaSouradniceMeridel;
    private javax.swing.JTextField jTextFieldDelkaZpravy;
    private javax.swing.JTextField jTextFieldPeriodaZpravy;
    private javax.swing.JTextField jTextFieldPocetKoncentratoru;
    private javax.swing.JTextField jTextFieldPocetMeridel;
    private javax.swing.JTextField jTextFieldSirkaPasma;
}
```

Třída Meridlo.java

```
import java.awt.Color;              //importované knihovny

public class Meridlo {              //datová struktura měřidla

    public int cislo;               //určuje pořadové číslo měřidla
    public int x;                   //souřadnice X
    public int y;                   //souřadnice Y
    public int barva;               //číselné označení barvy
    public Color color;             //konkrétní barevná hodnota

    public Meridlo() {
    }

    public Meridlo(int cislo, int x, int y) {    //konstruktor
        this.cislo = cislo;
        this.x = x;
        this.y = y;
    }

    public void setColor(Color color) {     //metoda set pro nastavení barvy
        this.color = color;
    }

    public Color getColor() {               //metoda get pro přečtení barvy
        return color;
    }

    public int getCislo() {
        return cislo;
    }

    public int getBarva() {
        return barva;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void setCislo(int cislo) {
        this.cislo = cislo;
    }

    public void setBarva(int barva) {
        this.barva = barva;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }
}
```

Třída Koncentrator.java

```
import java.awt.Color;
                                                    //význam stejný jako ve třídě Meridlo
public class Koncentrator {

    public int cislo;
    public int x;
    public int y;
    public int barva;
    public Color color;

    public Koncentrator() {
    }

    public Koncentrator(int cislo, int x, int y) {
        this.cislo = cislo;
        this.x = x;
        this.y = y;
    }

    public void setColor(Color color) {
        this.color = color;
    }

    public Color getColor() {
        return color;
    }

    public int getCislo() {
        return cislo;
    }

    public int getBarva() {
        return barva;
    }

    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public void setCislo(int cislo) {
        this.cislo = cislo;
    }

    public void setBarva(int barva) {
        this.barva = barva;
    }

    public void setX(int x) {
        this.x = x;
    }

    public void setY(int y) {
        this.y = y;
    }
}
```

Příloha B:

Spustitelný program s optimalizačním algoritmem je uložen na přiloženém CD v souboru **Příloha B _ Optimalizační program**. Informace a podmínky spuštění jsou uvedeny níže.

O PROGRAMU
Autor: Lukáš Hudec
Diplomová práce 2011: Systémy dálkového měření v energetice

SPUŠTĚNÍ PROGRAMU
Pro spuštění programu je nutné mít nainstalované běhové prostředí pro JAVu, tzv.:
Java Runtime Environment (JRE). Je uložen na CD, nebo jej lze stáhnout z odkazu:

http://www.java.com/en/download/index.jsp

Po nainstalování lze spustit optimalizační program souborem Optimalizace.jar.

POSTUP OVLÁDÁNÍ
V prvním kroku je nutné vyplnit vstupní parametry pro optimalizaci.
Pro výpočet počtu koncentrátorů pro daný počet měřidel lze stisknout tlačítko Vypočítej.
Náhodné rozmístění měřidel a koncentrátorů se provede pomocí tlačítka Náhodné rozmístění.
Optimalizaci je možné provádět krok po kroku (tlačítkem Jeden krok k-means) nebo jednorázově (tlačítkem Optimalizuj).
Po optimalizaci je graficky znázorněn výsledek, který je též možné odvodit ze souřadnic.

Zadávané parametry musí být vždy voleny rozumně, aby nedocházelo ke značnému nárůstu počtu koncentrátorů a měřidel. Množství těchto prvků značně ovlivňuje výpočetní náročnost. Tím pádem může být doba potřebná k výpočtu značná.

Výstupem je grafické uspořádání měřidel a koncentrátorů. Jejich vzájemné vazby jsou odlišeny barvou. Dané měřidlo, které náleží konkrétnímu koncentrátoru je vyznačeno shodou barvou jako daný koncentrátor.

Příloha C:

Je uložena na přiloženém CD ve složce: **Příloha C _ Projekt NetBeans - Optimalizační algoritmus**. Jde o kompletní projekt s optimalizačním algoritmem uložený v programovacím prostředí NetBeans. Podrobné informace:

Verze produktu: NetBeans IDE 6.9.1 (Build 201011082200)
Java: 1.6.0_23; Java HotSpot(TM) Client VM 19.0-b09
Systém: Windows 7 verze 6.1 běžící na x86

Příloha D:

Je uložena na přiloženém CD ve složce **Příloha D _ Zdrojové kódy optimalizačního programu**. Jdou zde uvedeny kompletní zdrojové kódy ve formátu prostého textu. Lze je tedy číst a editovat kterýmkoli textovým editorem.

Složka obsahuje textové soubory, které mají stejný název jako je název třídy v daném projektu.

Gui.txt – Obsahuje celý program, včetně nastavení grafického rozhraní a komponent. Je zde obsažena metoda **kMeans**, která provádí vlastní optimalizaci. Dále se zde nachází generátor náhodných čísel, výpočet míry optimalizace a počtu kroků.

Main.txt – Třída, kterou se spustí celý program.

Meridlo.txt – Datová struktura, která uchovává informace o měřidlech. Je zde uložena jejich barva, souřadnice, atd.

Koncentrator.txt – Datová struktura definující vlastnosti koncentrátorů, které jsou podobné jako u měřidel.