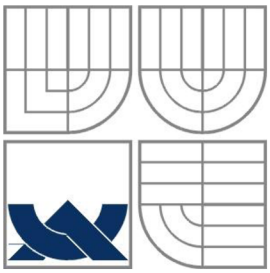
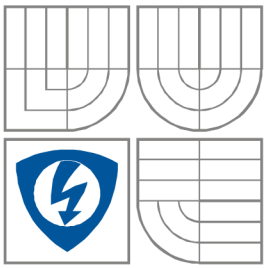


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

VYUŽITÍ TEORIE GRAFŮ PRO NÁVRH A OPTIMALIZACI ARCHITEKTUR DATOVÝCH SÍTÍ

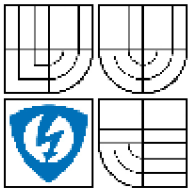
USE OF GRAPH THEORY FOR THE DESIGN AND OPTIMIZATION DATA NETWORK ARCHITECTURE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Adam Římský
AUTHOR

VEDOUCÍ PRÁCE doc. Ing. Vít Novotný, Ph.D.
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Adam Římský	***ID:***	78339
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Využití teorie grafů pro návrh a optimalizaci architektur datových sítí

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou teorie grafů, zaměřte se na řešení problému komunikačních sítí, a to její výstavby a optimalizace. Prostudujte přepojovací algoritmy vycházející z teorie grafů a používané v současných datových sítích. Na příkladě jednoduché síťové struktury ukažte použití teorie grafů při jejím návrhu na základě znalosti datových toků, ceny a kapacity spojů. Simulací zjistěte vliv výpadku spoje či uzlu na funkčnost navržené sítě.

DOPORUČENÁ LITERATURA:

[1] FRONČEK, D. Úvod do teorie grafů. Skriptum Slezská univerzita Opava, ISBN 80-7248-044-8, ČR, 1999

[2] WHITE,R., SLICE,D., RETANA,A. Optimal Routing Design. Cisco Press, ISBN 1-58705-187-7, USA, 2005

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: doc. Ing. Vít Novotný, Ph.D.

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

UPOZORNĚNÍ:

Abstrakt

Tato diplomová práce se zabývá využitím teorie grafů v oblasti datových sítí a to konkrétně v jejich návrhu a optimalizaci. Úvod práce tvoří obecný popis teorie grafů. tzn. základní pojmy používané k popisu grafů, dělení grafů atd. Další část je věnována popisu algoritmů, které vzešly z teorie grafů, jako například hledání nejkratší cesty. Poté se věnuji využití dříve uvedených algoritmů ve směrovacích protokolech, které se dnes používají. Předposlední část obsahuje teorii front a závěr obsahuje praktické ukázky užití teorie grafů při návrhu a optimalizaci datové sítě v prostředí programu Matlab.

Abstract

This master's thesis deals with graph theory and utilization of this theory for design and optimization of data network structures. Introduction chapter describes graph theory in general view, i.e. fundamental terms used for graph description, graph distinguishing, etc. Next part describes graph algorithms, for example a shortest path finding. After this I write about actual routing protocols where the graph algorithms are used. Last but one part deals with queuing theory and final part describes practical presentation of using graph theory for design and optimization of data network structure in Matlab programme environment.

Klíčová slova

Teorie grafů, graf, orientovaný, neorientovaný, OSPF, Dijkstra, minimální kostra, tok

Keywords

Graph theory, graph, directed, undirected, OSPF, Dijkstra, minimum spanning tree, flow

Bibliografická citace

ŘÍMSKÝ, A. *Využití teorie grafů pro návrh a optimalizaci architektur datových sítí.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 84 s. Vedoucí diplomové práce doc. Ing. Vít Novotný, Ph.D.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Využití teorie grafů pro návrh a optimalizaci architektur datových sítí 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 této diplomové práce 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 …… Podpis autora ……..

Poděkování

Děkuji vedoucímu diplomové práce doc. Ing. Vítu Novotnému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne …… Podpis autora ……..

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM SYMBOLŮ A ZKRATEK

b	bit
B	byte
b/s	bit/s
bps	bity/s
E	Množina hran
G	Graf
Gb/s	Giga bit/s
Gbps	Giga bit/s
m	metr
ms	milisekunda
Mb/s	Mega bit/s
Mbps	Mega bit/s
OSPF	Open Shortest Path First
RIP	Routing Information Protocol
u	Vrchol u
v	Vrchol v
V	Množina vrcholů
VLSM	proměnná délka masky sítě
w	váha

1 Úvod

Teorie grafů je relativně nová matematická disciplína, která je součástí diskrétní matematiky. Prvním představitelem této teorie byl švýcarský matematik Leonhard Euler. Ten se v roce 1736, ve své práci „Solutio problemas ad geometriam situs pertinentis“, zabýval řešením problému mostů města Kaliningradu. Euler řešil nejen tuto úlohu, ale také obecně vyřešil, které grafy lze nakreslit jedním tahem, tak vznikl pojem Eulerův tah. S Eulerovým jménem je také spojena úloha jezdce, která v grafové interpretaci vede k nalezení hamiltonovské kružnice v grafu.

Na graf můžeme pohlížet z různých perspektiv. V té jednodušší podobě je to pouhé zobrazení funkční závislosti dvou proměnných, čímž se ovšem v této práci zabývat nebudeme. V té komplexnější podobě budeme na graf pohlížet jako na diskrétní strukturu tvořenou vrcholy, jenž jsou propojeny hranami. Touto komplexnější podobou grafu se zabývá a zkoumá ji jeden z oborů matematiky nazvaný teorie grafů.

S takovouto podobou grafů se setkáváme v mnoha oborech, například v chemii se jí využívá k modelování vazeb mezi molekulami, při výzkumu umělé inteligence pomocí ní modelujeme umělé neuronové sítě atd. V té snáze představitelné podobě se s ní setkáváme při projektování komunikací a elektrifikaci území, kde vrcholy tvoří jednotlivá města a vesnice a hrany přestavují právě komunikace nebo elektrické vodiče. V životě se s touto podobou grafů setkáváme zcela běžně. Každá personální struktura firmy může být popsána takovým to grafem. Vždy, když chceme navštívit několik míst ležících v různé vzdálenosti, plánujeme trasu tak, aby byla výsledná ujetá vzdálenost například co nejkratší, tím využíváme některých algoritmů z teorie grafů zcela intuitivně.

Tato práce se bude zabývat strukturou datových sítí, konkrétně návrhem a optimalizací, což by se zjednodušeně dalo přirovnat k výše zmíněné dopravní problematice, nicméně křižovatky musíme zaměnit za směrovače a silnice za datové spoje. Proto je snahou, aby se veškeré zde uvedené poznatky vztahovaly a byly aplikovatelné právě na tuto problematiku, byť jsou mnohdy na tolik obecné, že přímá souvislost nemusí být na první pohled patrná.

Pokud jde o samotné členění této práce, tak po tomto úvodu následuje pasáž, ve které popíši základní pojmy používané v teorii grafů. V další kapitole se hodlám zaměřit na popis některých algoritmů, které jsou v teorii grafů využity pro operace s grafy a které jsou aplikovatelné na problematiku datových sítí. Po popisu těchto algoritmů následuje část, kde zmiňuji využití některých z nich v dnes používaných směrovacích protokolech, OSPF a RIP, a také se zde zmíním o struktuře sítě jako takové. Následující kapitola by se měla zabývat problematikou teorie front, což s teorií grafů sice přímo nesouvisí, ale pro problematiku optimalizace přenosů v sítích je to přínosné. Předposlední kapitola by měla obsahovat praktické ukázky využití teorie grafů, které jsou zpracovány v prostředí programu Matlab. Závěrečná kapitola shrnuje zjištěné poznatky a určitá doporučení, jak problematiku teorie grafů dále využít.

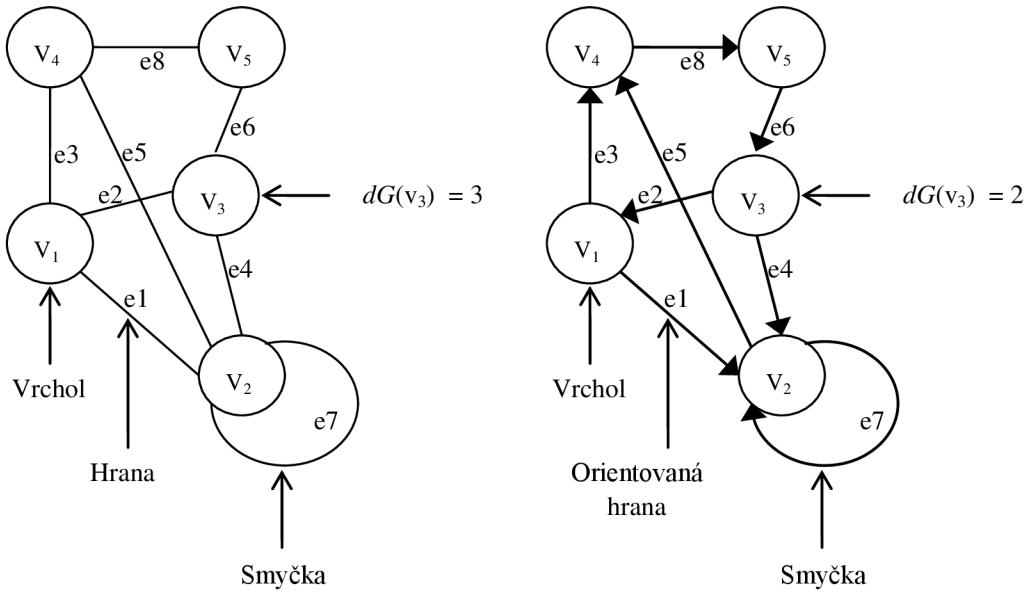
2 ZÁKLADNÍ POJMY TEORIE GRAFŮ

V této kapitole se budeme zabývat obecnými pojmy s nimiž pracuje teorie grafů. Konkrétně si zde uvedeme obecnou definici grafu a poté další používané termíny. Pokud by měl někdo hlubší zájem o studium dané problematiky, nechť využije literaturu [8], [11] a [16].

2.1 Definice grafu

Dle [6] můžeme uvést, že graf je uspořádaná dvojice $G = (V, E)$, kde V je množina vrcholů a E je množina hran. Množina vrcholů V je konečná a neprázdná. Vrcholy jsou propojeny hranami. O hraně e, jenž náleží do množiny E můžeme prohlásit, že je to neuspořádaná, nebo uspořádaná, v případě orientovaných grafů, dvojice vrcholů, tedy $e = \{u, v\} \text{pro} u, v \in V$. Vrcholy u, v, spojené hranou e, se stávají sousedy. Pokud dojde k tomu, že se všechny vrcholy grafu stanou sousedy, je graf úplný. Je možné, že nastane případ, kdy je počáteční i cílový vrchol stejný, v tom případě mluvíme o smyčce. Dalším termínem, o kterém mluvíme při souvislosti s vrcholy a hranami, je stupeň vrcholu. Stupněm vrcholu u v grafu G rozumíme počet hran vycházejících z u. Stupeň vrcholu u v grafu G značíme $dG(v)$. Nejvyšší stupeň v grafu G je značen $\Delta(G)$ a nejnižší $\delta(G)$. Součet všech stupňů v grafu je sudý. Takto popsaný graf je zobrazen na Obr. 2.1.

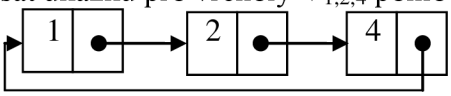
S vykreslením hran souvisí první větší rozdělení grafů, se kterým se v tomto textu setkáme. Hrany mohou být neorientované, v tom případě jde o symetrický vztah mezi uzly u a v. Nastávají ovšem situace, kdy potřebujeme hranu orientovat. Tato situace nastává například při řešení problematiky toků v síti, o které se zmíníme v kapitole 2.2.10. V takovém případě mluvíme o orientovaném grafu, který je, opět dle [8], definován jako uspořádaná dvojice $G = (V, E), \text{kde} E \subseteq V \times V$. U orientovaného grafu musíme brát na zřetel, že hrana d, která má počátek v uzlu u a končí v uzlu v, není totožná s hranou, jenž je orientována opačně.



Obr. 2.1: Neorientovaný (vlevo) a orientovaný (pravý) graf

Na závěr této podkapitoly si řekneme, jakými způsoby může být graf reprezentován. Způsoby jsou následující:

- Definicí, viz [8], dle které můžeme k Obr. 2.1 napsat:
  $G = (\{1,2,3,4,5\},\{[1,2],[1,4],[2,2],[2,4],[3,1],[3,2],[4,5],[5,3]\})$
- Diagramem, viz Obr. 2.1
- Maticí - sousednosti $\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$, vzdáleností, $\begin{pmatrix} 0 & d_1 & 0 & d_3 & 0 \\ 0 & d_7 & 0 & d_5 & 0 \\ d_2 & d_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d_8 \\ 0 & 0 & d_6 & 0 & 0 \end{pmatrix}$
- Různými datovými strukturami - například seznamem. Pro případ neorientovaného grafu, na Obr. 2.1, je možné zapsat ukázku pro vrcholy $V_{1,2,4}$ pomocí jednosměrného kruhového seznamu takto:



2.2 Další pojmy teorie grafů

Po uvedení základní definice grafu, zde budou popsány další termíny používané v teorii grafů.

2.2.1 Smíšený graf

Smíšený graf je kombinací orientovaného a neorientovaného grafu a vznikne tehdy, když nejsme schopni u všech vrcholů určit, zda jsou počáteční nebo koncové.

2.2.2 Převod orientovaného grafu na neorientovaný a opačně

Orientované a neorientované grafy jsou díky své konstrukci využívány pro odlišné typy úloh a reprezentaci různých problémů. Z toho také vyplývá, že práce s nimi je odlišná. Nastávají ovšem případy, kdy je potřebné převést orientovaný graf na neorientovaný nebo opačně. K tomu se využívá následujících procedur

- Orientace grafu
- Syntetizace grafu

V prvním případě jde o postup, kdy neorientovaný graf přetváříme na orientovaný. Toto může být provedeno pomocí dvou metod

- Symetrická orientace - tedy jde o postup, kdy ke každé neorientované hraně e spojující vrcholy u a v přiřadíme další hranu e_i, a poté pro každou z nich zvolíme opačnou orientaci. V případě smyček je postup odlišný, tyto nezdvojujeme, pouze každé smyčce určíme její směr. Jak je z tohoto popisu patrné, rapidně se nám tímto zvýší počet hran.
- Náhodná orientace - jedná se o postup, kdy je orientace každé neorientované hrany zvolena náhodně nebo podle předem určených pravidel. O náhodných grafech je možno získat další informace v [14].

Druhý zmíněný proces, tedy syntetizace grafu, se zabývá opačným problémem, tedy přetváří orientovaný graf na neorientovaný. V tomto případě je postup v pravdě triviální, jde totiž o pouhé odstranění smyslu orientace každé hrany.

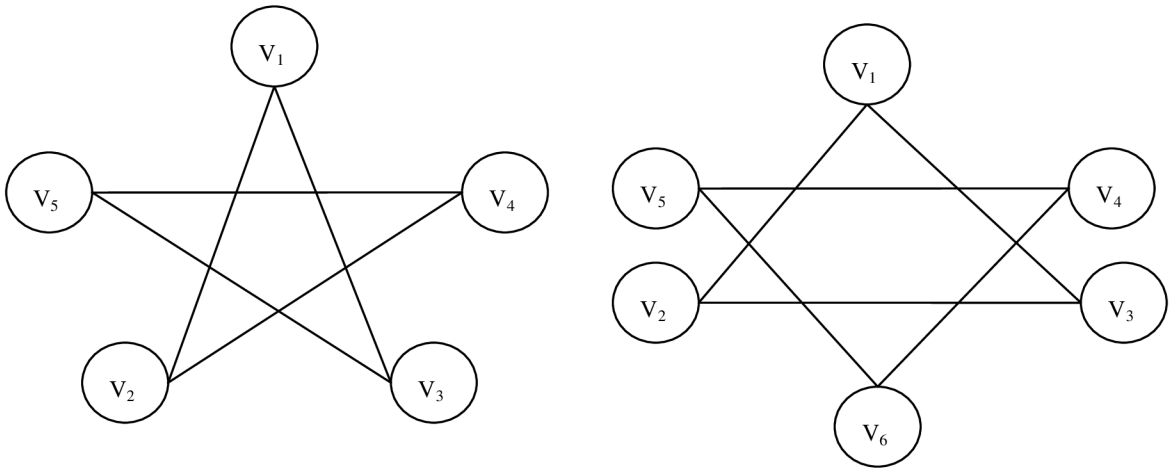
Dle právě zmíněných postupů pro převod mezi orientovaným a neorientovaným grafem je patrné, že proces syntetizace je výrazně jednodušší. Proto se také orientovaný graf považuje za základní graf a neorientovaný graf za odvozený z orientovaného. K tomuto rozdělení vedou i další skutečnosti jako například to, že při zadávání grafu vždy zmíníme odkud a kam se má hrana vést, byť na orientaci nakonec záležet nemusí.

2.2.3 Cesta

Než budeme moci napsat samotnou definici toho, co to je **cesta**, budeme si muset definovat další pojem, který k této definici budeme potřebovat. Jde o tzv. **sled**. Dle [16] můžeme napsat, že sled je posloupnost u_0, e_1, u_1, e_2, u_2, …, e_k, u_k, pokud každá hrana e_i této posloupnosti spojuje vrcholy u_{i-1}, u_i. Pokud se ve sledu, ať už orientovaném nebo ne, nevyskytuje opakovaně žádná hrana, nazýváme ho **tahem**. Pokud se ve sledu neopakuje žádný vrchol, nazýváme ho **cestou**, opět orientovanou nebo neorientovanou. V dalších kapitolách se budeme zabývat algoritmy pro nalezení různých typů cest.

2.2.4 Souvislost grafu

S výše definovaným pojmem cesta souvisí další pojem teorie grafů a to souvislost grafu. Dle [8] a [16] můžeme napsat, že je graf G souvislý tehdy, když jsou každé dva vrcholy grafu propojené cestou.



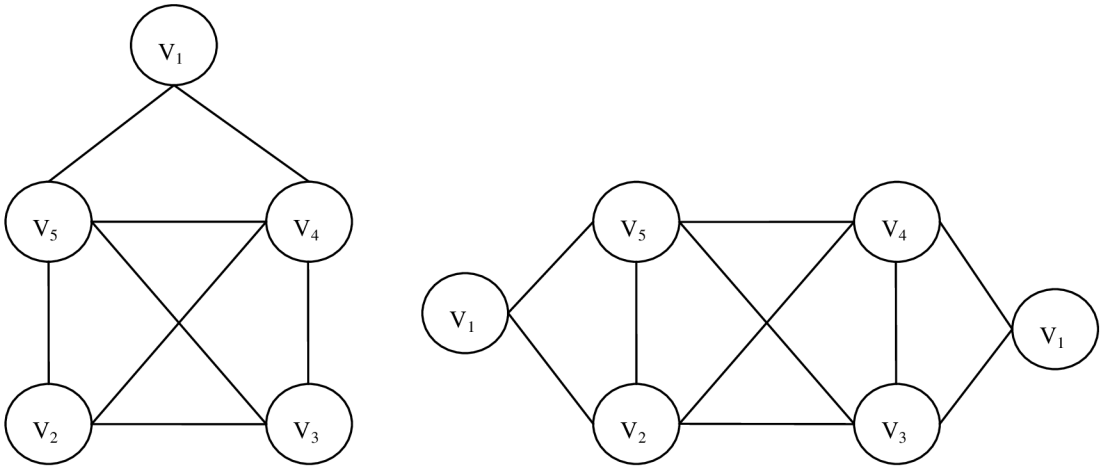
Obr. 2.2: Souvislý (vlevo) a nesouvislý (vpravo) graf dle [16]

2.2.5 Eulerův tah

Ve spojitosti se souvislostí grafů můžeme zmínit další, v teorie grafů využívaný pojem, a to Eulerův tah. Dle [8], [13], [16] můžeme napsat, že je to tah, který obsahuje všechny hrany grafu právě jednou.

Eulerovi tahy mohou být uzavřené a otevřené. Dle Eulerových vět platí, že souvislý graf obsahuje uzavřený Eulerův tah tehdy, když jsou všechny jeho vrcholy sudého stupně. Otevřený Eulerův tah se v souvislém grafu vyskytuje za předpokladu, že právě dva vrcholy tohoto grafu jsou lichého stupně a všechny ostatní sudého.

Tohoto bylo Eulerem využito například k řešení úlohy sedmi mostů města Kaliningradu. Bylo tak prokázáno, že úloha nemá řešení, viz literatura [8].

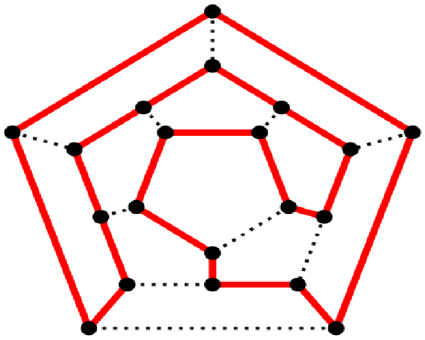


Obr. 2.3: Otevřený (vlevo) a uzavřený (pravý) Eulerův tah

2.2.6 Hamiltonovské cesty

Pro definici hamiltonovské cesty můžeme opět využít definici v [16], která říká, že je to taková cesta v grafu *G*, která obsahuje každý vrchol grafu právě jednou. Výjimku tvoří počáteční vrchol, který je zároveň cílovým. V teorii grafů existují úlohy, které hledají nejkratší hamiltonovskou cestu. Již teď si ale může říct, co lze o takovéto cestě prohlásit.

- Tvoří kostru grafu
- Pokud v zadaném grafu *G* odstraníme cestu, která nejkratší hamiltonovské cestě nenáleží, tato se nezmění.

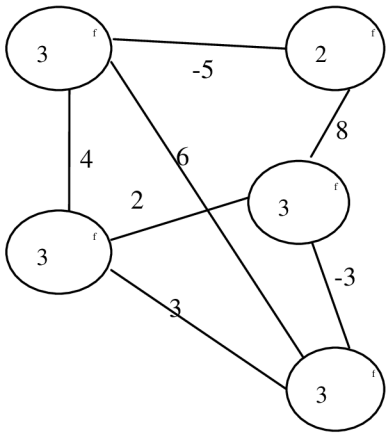


Obr. 2.4: Ukázka hamiltonovské cesty, dle [6]

Praktické využití hamiltonovské cesty je například při řešení všeobecně známe úlohy obchodního cestujícího.

2.2.7 Ohodnocení grafu

Až doposud jsme mluvili o grafech, kde se všechny hrany považovaly za rovnocenné a vrcholy se dělily pouze na zdrojové a cílové. Mnoho aplikací ke svému správnému chodu ovšem potřebuje hrany a vrcholy odlišit, a proto zavádíme ohodnocení grafu. Jedná se vlastně o to, že hranám a vrcholům přiřadíme hodnoty, většinou číselné, které je specifikují. U hran toto ohodnocení může představovat například jejich délku u uzlů počet hran, které do nich vstupují. Takové to hodnoty se většinou nazývají váhy jednotlivých prvků. Značíme je w a mohou být jak kladné, pak $w>0$ a řekneme, že graf je kladně ohodnocen nebo také $w<0$ a graf je záporně ohodnocen. Při ohodnocení grafu se nám samozřejmě také změní zadání grafu, kdy graf $G=(V,E,w)$. Každý prvek může být samozřejmě ohodnocen více hodnotami, jako doplňujícími podmínkami pro případy, kdyby se u některých prvků základní hodnoty rovnaly. Graf může být ohodnocen jednak pouze hranově, pak také pouze vrcholově a samozřejmě i kombinovaně.



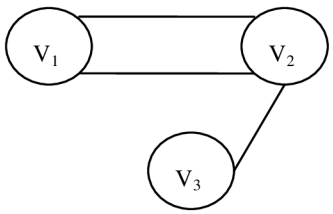
Obr. 2.5: Ukázka kombinovaně ohodnoceného grafu

2.2.8 Možné typy grafů

Grafy nabývají různých podob a tvarů často se stává, že z jednoho grafu lze odvodit několik dalších, ať už podgrafů nebo plnohodnotných nových grafů. V této části vycházím především z [13].

Multigraf

Prvním typem grafu, který zde bude popsán je takzvaný multigraf. Ten vzniká tehdy, pokud graf G obsahuje násobné hrany s násobností alespoň ≥ 1 a zároveň se v něm nevyskytuje žádná smyčka. Násobné hrany vznikají tehdy, když jsou dva vrcholy propojeny více než jednou hranou.



Obr. 2.6: Multigraf

Z Obr. 2.6 je patrné, že multigraf může vzniknout například při procesu orientace grafu.

Hustota grafu

Pokud jde o počet hran, tak další kritérium, které je tímto počtem ovlivněno, je hustota grafu.

Za předpokladu, že $|E| << |V|^2$, tedy množství hran je lineární vůči množství vrcholů, můžeme o grafu říci, že je řídký.

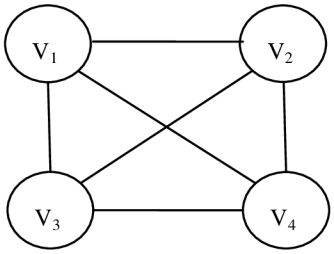
Pokud je naopak množství hran vysoké, tedy $|E| \sim |V|^2$, můžeme o grafu prohlásit, že je hustý.

Úplný orientovaný graf

Je to takový orientovaný graf, který je definován jako $G = (V, R)$, kde R značí množinu všech uspořádaných dvojic vrcholů z množiny V.

Úplný neorientovaný graf

Jedná se o takový neorientovaný graf, ve kterém jsou každé dva libovolné vrcholy propojeny hranou.



Obr. 2.7: Úplný neorientovaný graf

Diskrétní graf

Je to takový graf, pro který platí $G = (V, E)$ pro $E = \{\varnothing\}$. Z tohoto je tedy patrné, že se jedná o graf bez hran. Můžeme ho uvažovat jako nejmenší prvek množiny všech možných grafů vzhledem k pevně danému množství vrcholů. Tento typ grafu je dosti často uvažován na počátku některých grafových algoritmů, jako třeba Borůvkova pro hledání minimální kostry, tento si popíšeme v kapitole 3.2.2.

Podgraf

Jedná se o graf, který vznikne tím způsobem, že z původního grafu G použijeme k vytvoření nového grafu G' pouze některé jeho hrany a všechny vrcholy s těmito hranami incidentními. Matematicky toto můžeme zapsat následovně $G' = (V', E')$, kde $V' \subseteq V$ a $E' \subseteq E$. Musíme mít na paměti, že E' spojují pouze vrcholy z V'. Celkově tedy můžeme napsat, že $G' \subseteq G$.

V literatuře [8] se dále uvádí pojem **indukovaný podgraf**. Je to takový graf G', který obsahuje všechny hrany původního grafu G mezi vrcholy V' grafu G'.

Obr. 2.8: Graf G (vlevo) a jeho podgraf G' (vpravo)

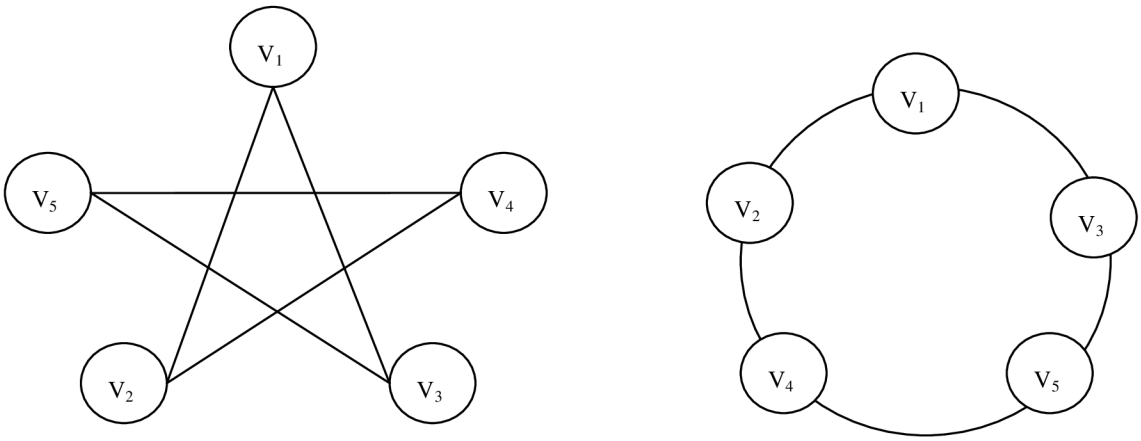
Obr. 2.9: Graf *G* (vlevo) a jeho indukovaný podgraf *G'* (vpravo)

Komponenta souvislosti

V souvislosti s podgrafy se setkáváme s pojmem komponenta souvislosti, která, dle [16], označuje podgraf *G'* grafu *G* takový, že je souvislý a maximální.

Isomorfismus

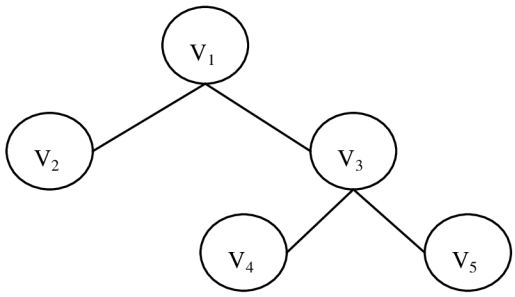
Dle [8] můžeme dva grafy *G* a *G'* považovat za ekvivalentní tehdy, pokud mají totožnou množinu vrcholů a hran, tedy *V'=V* a *H'=H*. Isomorfismus mezi grafy *G* a *G'* nastává tehdy, pokud mezi nimi existuje bijektivní zobrazení $f : V(G) \rightarrow V(G')$, pro které platí, že každá dvojice vrcholů $u, v \in V(G)$ je spojená hranou v *G* tehdy, jestliže je dvojice $f(u), f(v)$ spojená hranou v grafu G'. Isomorfní grafy mají dále tu vlastnost, že mají stejný počet vrcholů a hran



Obr. 2.10: Isomorfismus mezi grafy

Strom

Graf *G* můžeme nazvat stromem právě tehdy, pokud jsou jeho dva libovolné vrcholy propojeny jedinou cestou. O takovém grafu tedy můžeme mluvit jako o jednoduchém souvislém grafu bez kružnic. Přesto, že se tato struktura může jevit velice jednoduchá, našla si své uplatnění, mimo jiné, jako důležitá datová struktura v informatice.



Obr. 2.11: Ukázka stromu

Kostra grafu

Dle [8] mluvíme o kostře grafu *G* jako o jeho podgrafu *G'*, který je zároveň stromem, jenž obsahuje všechny vrcholy původního grafu *G*. Jedná se tedy o podgraf, kde jsou hranami spojeny všechny vrcholy původního grafu, ale zároveň se zde nevyskytuje žádná kružnice. Jeden graf *G* může mít i více než jednu kostru, dle Cayleyho formule má graf se třemi a více vrcholy až n^{n-2}, kde *n* je počet vrcholů, koster. V obecném neohodnoceném grafu jsou si všechny kostry rovny. V případě ohodnoceného grafu zavádíme pojem minimální kostra grafu, protože v tomto případě se budou jednotlivé kostry lišit součtem ohodnocení svých hran. V dalších kapitolách pak budou vysvětleny algoritmy pro výpočet minimální kostry grafu.

Obr. 2.12: Graf *G* a jedna z jeho koster

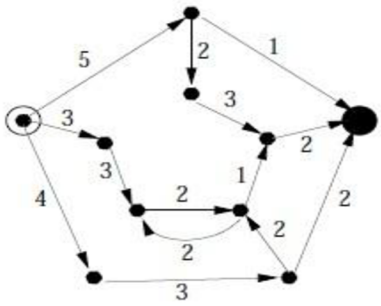
2.2.9 Vzdálenost a metrika v grafech

Definovat vzdálenost v grafu *G* můžeme dle [8] následujícím způsobem. Vzdálenost *dG*(*u, v*) dvou vrcholů *u, v* grafu *G* je dána délkou nejkratší cesty mezi *u* a *v* grafu *G*. Pokud cesta mezi *u* a *v* neexistuje, je vzdálenost $dG(u, v) = \infty$. V neohodnoceném grafu je vzdálenost reprezentována například počtem hran, z nichž se cesta skládá, v ohodnoceném grafu se využívá vah *w*. V neorientovaném grafu můžeme považovat vzdálenost *dG*(*u, v*) a *dG*(*v, u*) za totožnou. V další kapitole budou uvedeny algoritmy pro výpočet nejkratších cest mezi vrcholy grafu *G*.

Metrikou máme na mysli matici vah všech hran mezi propojenými vrcholy grafu *G*. Tedy například prvek *dG*(*u, v*) metriky *G* udává vzdálenost mezi vrcholy *u* a *v* grafu *G*.

2.2.10 Toky v síti

První, co si v souvislosti s toky v síti musíme definovat, je síť. K tomu můžeme využít [16], kde zjistíme, že síť je čtveřice $S=(G,z,s,c)$, kde G je orientovaný graf, z, s dva různé vrcholy grafu G takové, že do z nevstupují žádné hrany a z vrcholu s žádné hrany nevystupují, a c je funkce $c: H(G) \to R^{+}$, která každé orientované hraně h grafu G přiřazuje kladné reálné číslo $c(h)$. Vrchol z se nazývá *zdroj*, vrchol s *spotřebič* a číslo $c(h)$ *kapacita* (nebo *propustnost*) *hrany h*.



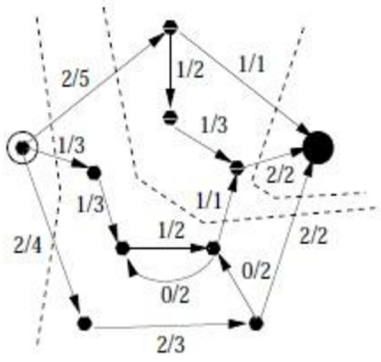
Obr. 2.13: Ukázka sítě dle [15], hrany ohodnoceny kapacitami

Poté, co jsme si definovali síť, můžeme definovat samotný tok v této síti. Podle [15] tokem v síti S rozumíme ohodnocení hran $f: E \to R$ takové, že součet hodnot toků vstupních hran u každého vrcholu v, kromě zdroje a spotřebiče, je stejný jako součet výstupních hran z téhož vrcholu, tj.

$$\sum_{e \in IN(v)} f(e) = \sum_{e \in OUT(v)} f(e) \tag{2.1}$$

Velikost toku pak lze podle této definice snadno vypočítat jako

$$|f| = \sum_{e \in IN(s)} f(e) - \sum_{e \in OUT(v)} f(e) \tag{2.2}$$



Obr. 2.14: Síť s hranami ohodnocenými toky v nich a jejich kapacitou, dle [15]

V praxi nás většinou nejvíce zajímá maximální tok v dané síti. Mohou nás ale také například zajímat nenasycené cesty, ve kterých by se dal tok zvýšit. Algoritmy určené pro řešení těchto úloh si popíšeme v kapitole 3.3.

2.2.11 Další úlohy teorie grafů

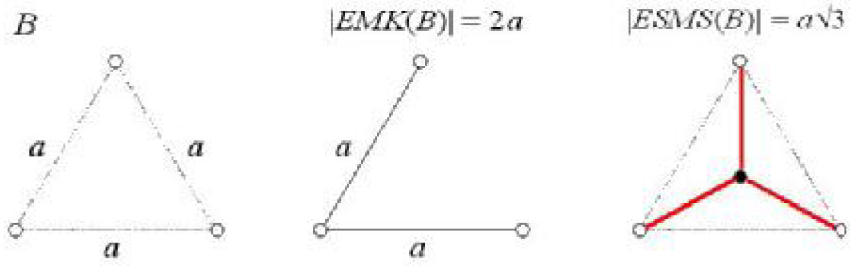
V této závěrečné pasáži ještě zmíníme, alespoň okrajově, další zajímavé úlohy, kterými se teorie grafů zabývá, ale v práci dále zmíněny nebudou.

Barvení grafů

Dle [16] obarvením grafu G (též obarvením vrcholů grafu) rozumíme ohodnocení vrcholů grafu $c:V(G\rightarrow B)$ hodnotami z množiny B (tzv. barvami) takové, že žádné dva vrcholy spojené hranou e nejsou ohodnoceny (obarveny) stejnou barvou. Zároveň platí věta, že pro každý rovinný graf G je $\chi(G)\leq 4$, tzn. že pro obarvení každého rovinného grafu si vystačíme s maximálně čtyřmi barvami.

Steinerovy problémy

Jde o vyhledání nejkratšího rovinného spojení mezi množinou dvojic bodů, tzv. terminálů, definovaných v euklidovské rovině. Výsledek této úlohy má tvar stromu nazvaného Euklidovský Steinerův minimální strom. Toto můžeme porovnat s kostrou grafu, ale na rozdíl od ní zde není vyžadováno, aby spojení vedlo pouze skrze zadané terminály, ale může vést i přes body, jenž s nimi nesouvisí. Těmto říkáme Steinerovy body.



Obr. 2.15: Porovnání minimální kostry a Steinerova minimálního stromu dle [16]

Jako další úlohy, kterými se teorie grafů zabývá, můžeme uvést například vykreslení Voroninových diagramů, hledání nejpočetnějšího párování, Delaunayho triangulace atd. Podrobněji se o této problematice zmiňuje literatura [8], [13], [16].

3 ALGORITMY ZALOŽENÉ NA TEORII GRAFŮ

V této kapitole se zaměříme na vysvětlení některých grafových algoritmů používaných pro řešení různých typů úloh. Jde většinou o úlohy optimalizační, ale najdou se i takové, které se zabývají návrhem. Je možné říci, že tyto algoritmy jsou uplatnitelné vždy, když je možné převést řešenou problematiku na grafovou strukturu. Toto u datových sítí možné je, a proto se v následující kapitole zaměříme právě na tuto problematiku.

Konkrétně se v této kapitole budeme soustředit na algoritmy hledání cest, minimální kostry grafů a nakonec na hledání maximálního toku v síti.

3.1 Algoritmy pro hledání cest

V případě těchto algoritmů rozeznáváme dle [2] tři typy úloh.

Prvním typem je **hledání nejkratších cest**. Tento typ můžeme dále, například dle [16], rozdělit na tři podskupiny, kterými jsou.

1. Hledání nejkratší cesty mezi dvěma zadanými vrcholy.
2. Hledání nejkratší cesty mezi zadaným vrcholem a všemi ostatními nebo naopak.
3. Hledání nejkratší cesty mezi dvěma libovolnými vrcholy grafu.

První dvě skupiny úloh lze řešit pomocí Dijkstrova a Bellman-Fordova algoritmu. Těchto je využito ve směrovacích protokolech popsaných v kapitole 4. Poslední skupina je řešitelná například pomocí Johnsonova algoritmu. Ve všech třech případech se jedná o optimalizační úlohy, které jsou při přenosu dat využity k jeho vyšší rychlosti a efektivnosti. Jednotlivé algoritmy si popíšeme v kapitolách 3.1.1 a 3.1.2.

Druhým typem úloh je tzv. **hledání nejspolehlivější cesty**. V případě úlohy tohoto typu se jedná o využiti Dijkstrova algoritmu pro nalezení nejkratší cesty mezi dvěma zadanými vrcholy. Rozdíl je zde ovšem v tom, že jednotlivé hrany nejsou ohodnoceny délkou, ale pravděpodobností úspěšného nebo neúspěšného průchodu skrze tuto hranu.

V případě ohodnocení hrany pravděpodobností neúspěchu pří průchodu touto hranou musíme, pro zachování funkčnosti Dijkstrova algoritmu, toto přepočítat na pravděpodobnost úspěšného průchodu. Dále je pak dle [2][1] postup následující. Pomocí logaritmické funkce tuto úlohu převedeme na úlohu hledání nejkratší cesty a výslednou spolehlivost dané cesty určíme ze vztahu

$$s(m(u,v)) = \prod_{h \in m(u,v)} p(h) \tag{3.1}$$

kde $p(h) \in \langle 0,1 \rangle$ je pravděpodobnost úspěšného průchodu hranou a $s(m(u,v))$ je spolehlivost cesty m. Nejspolehlivější cestu pak můžeme vyjádřit následovně.

$$s(m(u,v)) = ma\{s(m(u,v))\} \tag{3.2}$$

Tohoto je možné využít při optimalizaci přenosu dat a struktury sítě jako takové. Pokud určíme spojení, která jsou nestabilní, můžeme se jim pokusit při přenosu vyhnout nebo je upravit

strukturou sítě tak, aby tyto spoje byly zálohované a pokud nejsou nezbytně nutné pro fungování sítě tak je i odstranit.

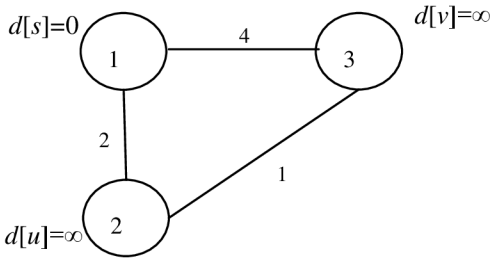
Poslední typ úloh se zabývá **hledáním cest s maximální kapacitou.** K řešení tohoto typu úloh se využívá například Ford-Fulkersonův algoritmus pracující s problematikou toků a řezů. Blíže si řešení těchto úloh předvedeme v části zabývající se hledáním maximálního toku v síti 3.3. V problematice sítí tohoto můžeme využít například k hledání cest s nejvyšší propustností.

Jednotlivé úlohy se dají kombinovat a ve spojení dále probíranými typy úloh zefektivňují návrh sítě a optimalizují provoz v nich.

3.1.1 Dijkstrův algoritmus

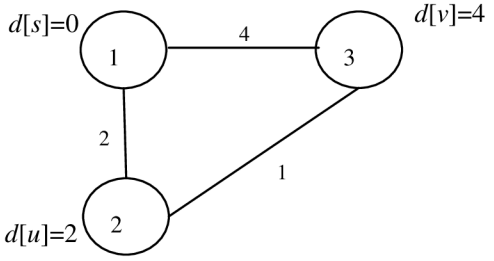
Základní podmínkou fungování Dijkstrova algoritmu je nezáporné ohodnocení hran grafu. Funkce algoritmu je založena na postupném zpřesňování odhadů vzdáleností z počátečního uzlu ke všem ostatním uzlům. Konkrétní postup určení vzdáleností popíši na základě literatury [16].

V prvním kroku určíme výchozí vrchol s. Tento bude mít vzdálenost $d[s] = 0$. Ostatní vrcholy grafu mají svou vzdálenost, $d[u]$ a $d[v]$, k vrcholu s rovnu ∞.



Obr. 3.1: První krok Dijkstrova algoritmu

Druhý krok spočívá v nastavení počátečních vzdáleností uzlů u a v, spojených s vrcholem s, vzhledem k ohodnocení hran w, jenž je spojují.

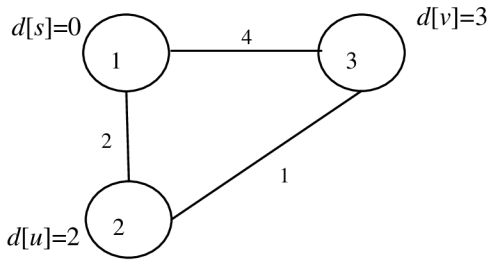


Obr. 3.2: Druhý krok Dijkstrova algoritmu

Ve třetím kroku přichází ke slovu tzv. relaxace hran, tedy proces upřesnění odhadu vzdálenosti vrcholu v vůči vrcholu s. K tomu ovšem potřebujeme z množiny vrcholů $V-s$ vybrat takový vrchol, vůči kterému toto zpřesnění provedeme. Tento je poté zařazen do množiny S vrcholů se známou konečnou hodnotou $d[v]$, která je na počátku prázdná. Protože algoritmus

používá hladovou strategii, vybere z množiny vrcholů $V-s$ do množiny S vrchol u s nejmenší hodnotou $d[u]$.

Pak bude platit následující. Pokud je součet $d[u]+w(u,v)<d[v]$, pak se stává $d[u]+w(u,v)$ novou hodnotou odhadu vzdálenosti $d[v]$.



Obr. 3.3: Třetí krok Dijkstrova algoritmu

Vzhledem k počtu vrcholů grafu jsme se dostali ke konci výpočtu. Můžeme tedy konstatovat, že nejkratší vzdálenost z vrcholu 1 do 2 je rovna 2 a z vrcholu 1 do 3 se rovná 3. Demonstrovali jsme tak jeho činnost pro vyhledání nejkratší cesty mezi zadaným a koncovým vrcholem a mezi zadaným vrcholem a ostatními vrcholy v grafu. V případě, že by vrcholů bylo v grafu více, proces by stejným způsobem pokračoval dále, dokud by nebyly nalezeny všechny nejkratší vzdálenosti. Algoritmus je využitelný jak pro orientovaný graf tak i pro neorientovaný, jako v tomto případě. Další optimalizace tohoto algoritmu je možná pomocí binární haldy, jak je uvedeno například v literatuře [8]. Algoritmus je konečný a optimální.

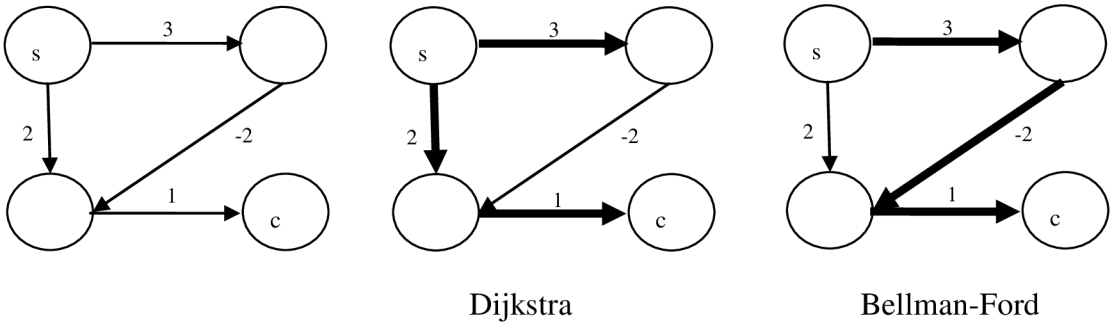
3.1.2 Bellman-Fordův algoritmus

Bellman-Fordův algoritmus je obdobou Dijkstrova algoritmu. Rozdíly mezi nimi jsou především v tom, že Bellman-Fordův algoritmus je schopen pracovat i se záporně ohodnocenými hranami, což samozřejmě pro datové sítě není, vzhledem ke vždy kladným hodnotám toků nebo vzdáleností, zásadní, ale je to jeho charakteristická vlastnost. Pracuje v orientovaném grafu a pro nalezení všech nejkratších cest potřebuje více času.

První dvě fáze jsou totožné s těmi pro Dijkstrův algoritmus. Ve třetí fázi proběhne n-1 iterací, kde n je počet vrcholů, ve kterých dojde k postupnému vylepšování odhadu hodnoty $d[v]$ dle pravidla, pokud je součet $d[u]+w(u,v)<d[v]$, pak se stává $d[u]+w(u,v)$ novou hodnotou odhadu vzdálenosti $d[v]$. Pro ověření, zda je tato podmínka splněna, se zavádí opravná funkce, tedy přiřazení $d[v]=\min\{d[u]+c(u,v)\}$, která má tyto vlastnosti.

- Nelze snížit hodnotu $d[v]$ pod hodnotu nejkratší cesty do v.
- Nastaví správnou hodnotu $d[v]$ tehdy, kdy u je předposledním vrcholem na cestě do v a hodnota $d[u]$ je správná.

Finálním krokem je pak kontrola, zda se v grafu nachází záporně ohodnocená smyčka. Tedy pokud pro některou hranu grafu platí, že $d[v]-d[u]>w$, pak řešení v takovém grafu neexistuje.



Obr. 3.4: Nelezení nejkratší cesty v záporně ohodnoceném grafu

3.1.3 Johnsnův algoritmus

Nyní se zaměříme na problém nalezení všech nejkratších cest mezi všemi páry vrcholů v grafu. Johnsnův algoritmu, který tento problém řeší, je založen na n-násobném (n je počet vrcholů) použití Dijkstrova algoritmu. Aby bylo možné použít Dijkstrův algoritmus, v případě grafů se záporným ohodnocením, musíme provést přehodnocení vstupního grafu, tedy výpočet hodnot w', které získáme transformací vstupního grafu Bellman-Fordovým algoritmem. Toto nové ohodnocení musí splňovat následující podmínky.

- Nejkratší cesta pro dva libovolné uzly u a v je shodná jak pro w, tak pro w'.
- Každá hrana grafu je po přehodnocení ohodnocena kladně.

Algoritmus provádí výpočet v následujících krocích.

- Do původního grafu G je přidán vrchol s, ze kterého je vedena orientovaná hrana ke všem ostatním vrcholům, ohodnocení těchto cest je rovno nule. Žádná z nejkratších mezi u a v neobsahuje uzel s.
- Díky prvnímu kroku můžeme každý vrchol původního grafu označit jako výchozí a spustit z něj Bellman-Fordův algoritmus. Ten proběhne tolikrát, kolik je počet vrcholů původního grafu. Na konci tohoto procesu ohodnotíme každý vrchol grafu hodnotou $b[u]$, nejlepší za všech vypočtených. Tímto zároveň proběhne kontrola, zda graf neobsahuje zápornou smyčku, tj. cesta by donekonečna procházela tímto záporně ohodnoceným cyklem a nejkratší spojení by neexistovalo. Pokud ano, proces výpočtu je ukončen.
- Nyní provedeme přehodnocení grafu dle vztahu $w'(u,v)=w(u,v)+b[u]-b[v]$, kde $b[u]$ je ohodnocení výchozího uzlu cesty $w(u,v)$ a $b[v]$ je ohodnocení koncového vrcholu této cesty.

- Poslední fáze je použití Dijkstrova algoritmu pro nalezení nejkratší cesty z každého vrcholu přehodnoceného grafu ke všem ostatním vrcholům a výsledné uložení nejlepší zjištěné hodnoty.

Graficky je tento postup zobrazen v literatuře [9].

Tento algoritmus je vhodný především pro řídké grafy, kde dosahuje lepších výsledků než další ze zástupců algoritmů pro hledání všech nejkratších cest Floyd-Warshallův algoritmus.

3.1.4 Časová náročnost algoritmů hledání nejkratších cest

Na závěr této části ještě porovnáme popsané algoritmy z hlediska jejich časové náročnosti.

Dijkstrův algoritmus

Jeho základní implementace má asymptotickou časovou složitost $O(|V|^2+|E|)$, *kde* $|V|$ je počet vrcholů a $|E|$ počet hran. Při použití binární haldy (stromová datová struktura) bude složitost $O((|E|+|V|)\log|V|)$.

Bellman-Fordův algoritmus

Složitost tohoto algoritmu je rovna $O(V*E)$.

Johnsonův algoritmus

Složitost tohoto algoritmu je rovna $O(V^2 \log V + VE)$.

3.2 Hledání minimální kostry grafu

V této kapitole probereme další typ úlohy, jenž je charakteristická pro teorii grafů, a která nám v případě struktur datových sítí poslouží k jejich počátečnímu návrhu, jenž je poté možné dále optimalizovat.

Co to je kostra grafu, je zmíněno v kapitole 2.2.8. Nyní uvedeme popis několika algoritmů, které jsou k řešení daného problému využívány. Konkrétně se jedná o Jarníkův, Borůvkův a Kruskalův algoritmus. Při jejich popisu budeme vycházet z literatury [11].

3.2.1 Jarníkův algoritmus

Jarníkův algoritmus, v zahraničí také někdy nazýván jako Primův algoritmus, dle jeho znovuobjevitele Američana Roberta Prima, byl poprvé popsán Čechem, profesorem Vojtěchem Jarníkem v roce 1930.

Tento algoritmus je založen na tom, že na počátku zvolíme výchozí vrchol v grafu, který spojíme s dalším vrcholem grafu takovým, který má k výchozímu vrcholu nejkratší vzdálenost. K tomuto podstromu je v každém kroku přidán jeden nový vrchol, z množiny vrcholů V grafu G, který má k dosud vytvořenému podstromu nejmenší vzdálenost ze všech vrcholů z dané množiny. Tyto po spojení musí stále tvořit jeden a tentýž podstrom. Protože se vzdálenosti již přidaných vrcholů k podstromu mohou při přidání nového vrcholu změnit musíme po každém kroku tyto hodnoty přepočítat. Proces končí po přidání všech vrcholů grafu k výchozímu podstromu.

Tím, že musíme určit, ve kterém vrcholu má algoritmus začít, nám vyvstává tzv. problém nejednoznačnosti minimální kostry. Tedy v případě, kdy začneme výpočet z jiného vrcholu a určíme další minimální kostru o stejné hodnotě, nemůže jasně říci, která má přednost, pokud nejsou specifikovány další podmínky.

3.2.2 Borůvkův algoritmus

Borůvkův algoritmus řeší problém nejednoznačnosti minimální kostry tím, že na počátku představuje každý vrchol grafu G samostatný podstrom, tedy tvoří množinu podstromů. Tyto podstromy jsou poté postupně spojovány hranami, které mají mezi jednotlivými podstromy, vrcholy, nejkratší délku, čímž se množina podstromů zmenšuje a vzniká jeden souvislý podstrom, který na konci procesu představuje jednoznačnou minimální kostru grafu G.

Tento algoritmus byl poprvé publikován Otakarem Borůvkou v roce 1926.

3.2.3 Kruskalův algoritmus

Kruskalův algoritmus, někdy také nazývaný Kruskalův hladový algoritmus, může pracovat dvěma způsoby.

Prvním je postupné přidávání hran s nejnižším ohodnocením do předem připraveného seznamu. Po každém přidání hrany se kontroluje, zda kostra neobsahuje kružnici, v tom případě hranu do seznamu nemůžeme přidat nebo zda jsou v takto vytvořené kostře grafu G obsaženy všechny vrcholy. Pokud takto vytvořená kostra obsahuje všechny vrcholy původního grafu, je vytvořena minimální kostra a algoritmus končí, v opačném případě je přidána další hrana.

Druhou možností je postup, kdy se nejprve naplní seznam všemi hranami grafu a tyto se setřídí dle velikosti. Poté postupně z tohoto seznamu odstraňujeme hrany s nejvyšším ohodnocením. V každém kroku musíme kontrolovat, zda i po odstranění hrany jsou v takto vytvořené kostře obsaženy všechny vrcholy původního grafu a zda nedošlo k rozpojení grafu. Hrany, které zůstanou na konci procesu, tvoří minimální kostru.

3.2.4 Časová náročnost algoritmů hledání minimální kostry

V případě Jarníkova a Borůvkova algoritmu můžeme jejich vhodnou implementací dosáhnout časové složitosti $O(V\log E)$. Pokud jde o Kruskalův algoritmus, zde je nejlepší možná hodnota $O(V\log E + V\alpha(E))$, kde α je tzv. Ackermannova funkce.

3.3 Hledání maximálního toku v síti

V poslední části třetí kapitoly je vysvětleno řešení problému hledání maximálního toku v síti. Základní informace jsou uvedeny v části 2.2.10. V problematice datových sítí tohoto můžeme využít pro hledání cesty s nejvyšší přenosovou kapacitou nebo naopak nejnižší, kterou je potřeba posílit. K řešení těchto typů úloh se využívá Ford-Fulkersonova algoritmu, který si dále dle [11] a [15] popíšeme.

3.3.1 Ford-Fulkersonův algoritmus

Nejprve si vypíšeme větu, dle které se chod Ford-Fulkersonova algoritmu řídí.

Věta Ford-Fulkerson: V každé síti je velikost maximálního toku rovna velikosti minimálního řezu.

K osvětlení smyslu této věty můžeme dle [15] napsat, že řezem v síti S rozumíme takovou množinu hran C, jež je podmnožinou množiny hran E, že po jejím odebrání nebude v grafu $G = (V, E \setminus C)$ žádná cesta ze zdroje ke spotřebiči. Ford-Fulkersonův algoritmus tedy hledá řez s minimální hodnotou a jeho tok.

Postup algoritmu je dle [15] následující.

- Všem hranám e patřícím do E nastavíme tok $f_{(e)}$ roven nule.
- Z vrcholu z, zdroj, prohledáváme síť do šířky a hledáme všechny vrcholy patřící do podmnožiny U množiny V, do kterých vede nenasycená cesta, tedy taková $f_{(e)} < w_{(e)}$.
- Dochází k opakování procedury, dokud vrchol s, spotřebič, patří do U takové, že vybereme nenasycenou cestu mezi z a s a zvýšíme tok f u všech jejích hran o minimální rezervu, tedy $w_{(e)} - f_{(e)}$. Po každém kroku obnovíme U.
- Algoritmus končí tehdy, kdy neexistuje nenasycená cesta mezi z a s. Z toho pak také plyne, že U neobsahuje s a pro všechny hrany e z U do zbytku je $f_{(e)} = w_{(e)}$, jinak bychom museli koncový vrchol e přidat k U.

Pokud chceme zajistit určité minimální průtoky v síti, můžeme je definovat do podmínek chodu algoritmu, ale musíme otestovat, zda jsou takové podmínky splnitelné.

3.3.2 Časová složitost Ford-Fulkersonova algoritmu

Celková časová složitost výše popsaného postupu skládajícího se z částí hledání cesty a zvyšování jejího toku je $O(|V|^2 * |E|)$.

4 SMĚROVÁNÍ S VYUŽITÍM TEORIE GRAFŮ

Tato kapitola bude obsahovat popis dvou, v současné době používaných, interních směrovacích protokolů, využívajících ke své činnosti algoritmů teorie grafů. Prvním popisovaným bude protokol OSPF, Open Shortest Path First, což je protokol typu Link State, což znamená, že tento protokol vytváří v paměti směrovačů kompletní mapu celé sítě, tento pojem bude dále upřesněn, Link State Database. Druhým v pořadí bude protokol RIP, tedy Routing Information Protocol, jenž patří do skupiny protokolů typu Distance Vector, mezi sousedícími směrovači dochází ke sdílení směrovacích tabulek a jediným hodnotícím kritériem volby cesty je počet skoků mezi směrovači v cestě k cíli.

4.1 OSPF, obecný popis

Na začátku je nutno uvést, že tento popis protokolu OSPF se zakládá na literatuře [5] a [12]. Dle těchto zdrojů tedy můžeme říci, že protokol OSPF je interním směrovacím protokolem a patří mezi zástupce protokolů typu Link State, který využívá Dijkstrova algoritmu pro hledání nejvýhodnějších, nejlevnějších, cest mezi jednotlivým směrovači.

Tento protokol má oproti dalším interním směrovacím protokolům značnou výhodu v tom, že je schopen pracovat v poměrně rozsáhlých sítích. Toho bylo dosaženo tím, že je síť rozdělena do menších oblastí. Při změně topologie v dané oblasti nemusí proběhnout přepočet nejvýhodnějších cest v celé síti. Proběhne pouze v dané oblasti a tato skrze své hraniční směrovače a páteřní oblast zasílá ostatním oblastem sumární informace s výsledky tohoto výpočtu.

Jednotlivé kroky protokolu OSPF při navazování komunikace a výměny směrovacích informací mezi směrovači můžeme popsat následovně.

- Směrovač skrze svá rozhraní, která jsou naprogramovaná pro funkci tohoto protokolu, vysílá tzv. Hello pakety. Pokud sousední, připojené, směrovače v patřičném čase odpoví, stávají se sousedy.
- Mezi některými sousedy vznikne bližší vazba, stanou se přilehlými. Kdy toto nastává, je popsáno v literatuře [12] kapitola 5 a částečně níže v textu.
- Přilehlé směrovače si předávají LSA pakety, které obsahují informace o síti, dané oblasti a pomocí kterých se ve směrovačích vytváří mapa sítě, přesněji řečeno dané oblasti. Tyto pakety si jednotlivé směrovače ukládají do paměti a přeposílají je dále.
- Jakmile je ve směrovačích vytvořena kompletní mapa celé sítě dané oblasti, přichází ke slovu Dijkstrův algoritmus pro výpočet nejkratší, zde nejlevnější, cesty. Pomocí tohoto protokolu si každý směrovač vypočítá patřičné cesty ke všem jemu známým směrovačům a tedy i k nim připojeným podsítím. Zároveň dojde k odstranění smyček v síti.
- Tyto vypočítané hodnoty slouží k naplnění směrovací tabulky.
- K aktualizacím dochází v případě změny topologie sítě a informuje o ní LSA paketem směrovač, jenž tuto změnu zjistil. Jak jsem ale psal výše, přepočet probíhá pouze v rámci dané oblasti.

4.1.1 Komunikace mezi směrovači v síti s OSPF protokolem

Komunikace probíhá vždy pouze mezi sousedními směrovači ležícími ve stejné oblasti. Data mezi nimi jsou přenášena pomocí IP protokolu, s identifikátorem protokolu 89. Při přenosu dat v síti Ethernet jsou využívány pouze multicastové adresy, což z komunikace vyloučí jednotlivé klientské stanice, je jim odepřen přístup do multicastové skupiny směrovačů.

4.1.2 Počáteční fáze komunikace, nalezení sousedního směrovače

Nalezení sousedního směrovače je první krok, který musí provést každý směrovač. Toto se děje pomocí tzv. Hello paketů, jenž jsou skrze rozhraní směrovačů, určených pro činnost OSPF protokolu, periodicky vysílány. Periodičnost je dána tzv. Hello intervalem, pro sítě LAN je to 10 sekund. Pokud směrovači nepřijde v Death intervalu, 4 ∗ Hello interval, odpověď, považuje cestu z tohoto rozhranní za nefunkční. Pro identifikaci jednotlivých směrovačů se využívá identifikátoru Router ID, který může mít hodnotu například nejvyšší IP adresy použité na kterémkoli jeho rozhraní. Hodnota Router ID nám také v pozdější fázi výměny topologických informací určuje, který směrovač bude zvolen jako Master. Spojení se považuje za navázané, když v poli Neighbor Hello paketu, jenž přišel jako odpověď, přibude k již známým Router ID identifikátorům hodnota směrovače, který se pokusil navázat komunikaci. Dále musí být splněna ta podmínka, že oba směrovače budou mít shodné Hello a Death intervaly a nachází se v téže oblasti.

4.1.3 Výměna topologických informací o síti

Mezi sousedními směrovači ležícími v té samé oblasti se vytváří bližší pouto a stávají se přilehlými. To jim umožňuje výměnu jejich databází s topologií sítě. Na konci procesu by tato databáze měla být stejná ve všech směrovačích dané oblasti.

Toto začíná přijetím náhodného sekvenčního čísla, které bude používáno pro další komunikaci. Toto číslo generují oba přilehlé směrovače Je obsaženo v Database Description paketech při jejich první výměně. Je zvoleno to, které bylo navrženo směrovačem zvoleným jako Master. Poté proběhne další výměna těchto paketů, nejprve ze strany Mastera, které jsou již značeny od zvoleného sekvenčního čísla. Oba směrovače si porovnají své záznamy s informacemi v příchozím paketu a pokud některý zjistí, že své informace potřebuje doplnit nebo obnovit proběhne následující proces

- Směrovač zašle paket Linsk State Request, kde o tyto informace žádá.
- Odpověď je v podobě paketu Link State Update.
- Tento je potvrzen zasláním Link State Acknowledgement paketu.

Pokud nedojde k potvrzení, je paket Update zaslán znovu. Tento paket je též zaslán v případě, kdy směrovač zjistí změnu v topologii sítě. O této pak informuje přilehlé směrovače a ty pak ostatní tak, aby se změny dostaly ke všem směrovačům v dané oblasti

S každou položkou v topologické databázi (LSA) je spojeno pole Age které je periodicky inkrementováno. Když jeho hodnota dosáhne Maxage, je LSA považováno za zastaralé a odstraněno z databáze. Proto jsou s periodou LSRefreshTime LSA rozesílány ke všem sousedům. Při přijetí LSA od sousedního routeru je pole Age vynulováno. Implicitní hodnota časovače MaxAge je v OSPF jedna hodina, pro časovač LSRefreshTime je to třicet minut.

4.1.4 Ohodnocení cest

Aby bylo možné z daných cest mezi směrovači vybrat tu nejlepší, musíme jednotlivé úseky ohodnotit. Na základě tohoto ohodnocení pak může být Dijkstrovým algoritmem vypočítána nejkratší, tedy v případě protokolu OSPF nejlevnější, cesta. V protokolu OSPF se totiž ohodnocení cesty nazývá cena, anglicky „cost“. Tato hodnota má rozsah 1 - 65535 a čím je nižší, tím je pochopitelně daná cesta výhodnější. Tato cena je odvozena na základě šířky pásma daného spoje dle vzorce

$$cena = \frac{1*10^8\ [b/s]}{šířka\ pásma\ \ [b/s]} \tag{4.1}$$

V případě spojů FastEthernet by se tato hodnota rovnala jedné, proto je možné v těchto případech konstantu $1*10^8$ zvýšit, ale není to podmínkou. Je také dále možné definovat dodatečné podmínky, které budou upřesňovat výběr cesty v případě rovnosti celkové ceny u vícero cest.

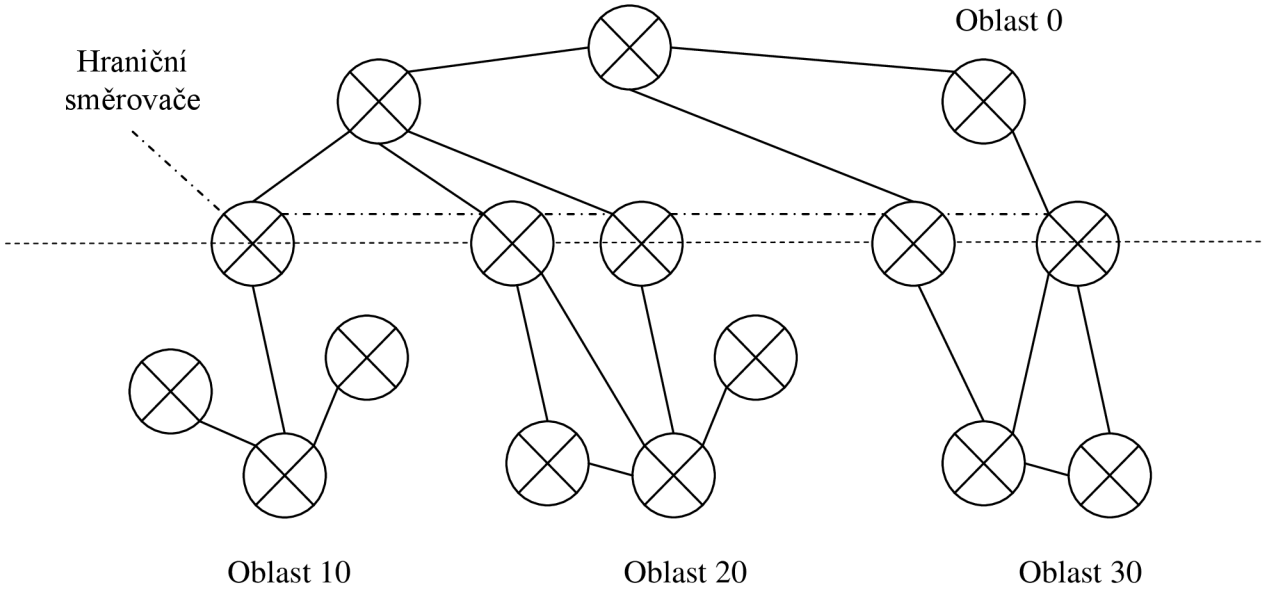
4.1.5 Naplnění směrovací tabulky

Poté, kdy je v paměti každého směrovače uložena kompletní mapa sítě dané oblasti, do které směrovač spadá, a jednotlivým cestám byla přiřazena patřičná cena, může začít výpočet nejlevnějších cest pomocí Dijkstrova algoritmu, popis výpočtu je popsán v části 3.1.1. Každý směrovač si vypočítá nejlevnější cestu ke všem známým uzlům v dané oblasti, do které spadá. Z těchto vypočítaných cest si ovšem uloží pouze hodnotu příštího skoku k nejbližšímu směrovači a celkovou cenu dané cesty. Pokud nastane situace, že je výsledná cena stejná pro několik cest, je na směrovači, kterou z nich využije, dle dodatečných podmínek, nebo zda jich použije více pro rozložení zátěže. Aby se předešlo situaci, kdy by se musel tento výpočet, který je pro směrovače náročný, často opakovat pro nestabilitu některých spojení, je jeho opakování omezeno časovým intervalem. Výsledky těchto výpočtů jsou v podobě sumárních informací šířeny do sousedních oblastí pomocí hraničních směrovačů.

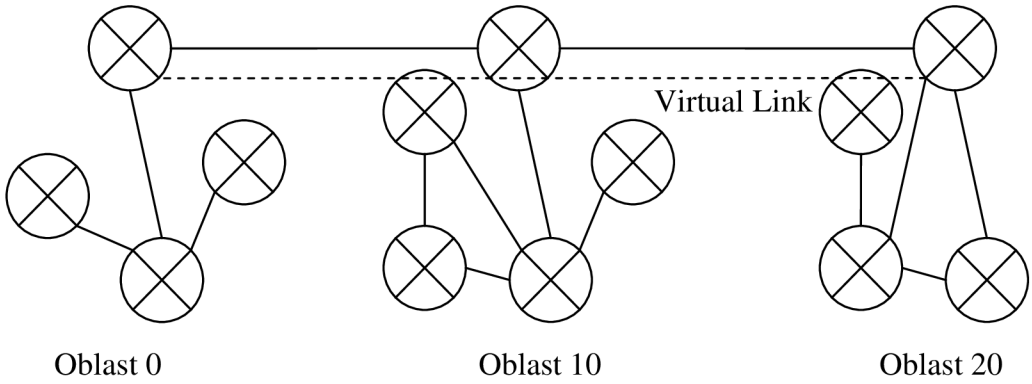
4.1.6 Hierarchie sítě a typy oblastí

Velkou výhodou, vůči ostatním interním směrovacím protokolům, protokolu OSPF je jeho schopnost pracovat v rozsáhlých sítích. Toho bylo dosaženo tím, že celá síť byla rozdělena na menší oblasti. Tím je omezena potřeba směrovačů na znalost kompletní topologie celé sítě pouze na oblast do které spadá. Z toho také vyplývá, že náročný výpočet nejlevnějších cest probíhá v daných oblastech samostatně, vzhledem ke změně topologie pouze v dotyčné oblasti. Výsledná data z těchto výpočtů se mezi oblastmi šíří pomocí tzv. hraničních směrovačů.

Jednotlivé oblasti jsou označeny 32 bitovým číslem, které může být zapsáno buď dekadicky nebo ve formátu IP adresy. Hierarchicky dominantní oblastí je tzv. páteřní oblast, značená jako oblast nula. Skrze tuto oblast musí mezi sebou komunikovat všechny ostatní oblasti, které jsou k ní připojeny hraničními směrovači, nebo virtuální linkou, a zároveň je v ní obsažen hraniční směrovač celého autonomního systému. V případě, kdy není možné fyzické propojení mezi páteřní oblastní a oblastí nižší úrovně, je možné použít tzv. virtuální spoj, který představuje logické spojení těchto oblastí. K využití této možnosti se ovšem vztahuje řada omezení, a proto je lépe se tomuto řešení vyhnout.



Obr. 4.1: Ukázka uspořádání oblastí v síti



Obr. 4.2: Využití virtuálního spoje

V zásadě rozlišujeme dva typy oblastí, tranzitní a stub.

Skrze tranzitní oblast pouze probíhá komunikace mezi oblastmi, nicméně provoz v nich nezačíná a nekončí, což je případ páteřní oblasti, popřípadě dalších takových oblastí, pokud obsahují hraniční směrovač autonomního systému.

Pokud jde o stub oblasti, tak v nich naopak pouze provoz začíná a končí. Jejich velkým přínosem pro rychlost a efektivitu směrování je omezení nutné znalosti topologie celé sítě směrovači. Tyto si vystačí se znalostí topologie dané oblasti, kam spadají.

Běžná oblast stub může být taková, ve které:

- Všechny směrovače jsou nakonfigurovány jako stub.
- Není skrze ní veden virtuální spoj.
- Neleží v ní hraniční směrovač autonomního systému.
- Není to páteřní oblast.
- Nejsou do ní propagovány externí cesty.

Dalším typem stub oblasti je **Totally stubby area**. Jde o rozšíření protokolu OSPF firmou CISCO. Směrovače v takovéto oblasti mají ve svých směrovacích tabulkách pouze cesty uvnitř dané oblasti a jedinou výchozí cestu z oblasti ven.

Posledním typem je **Not so stubby area**. Tedy taková oblast, která má sice vlastnosti velmi podobné běžné stub oblasti, je ale možné, aby v ní fungoval hraniční směrovač autonomního systému nezbytný pro fungování celé sítě.

4.2 Protokol RIP, obecný popis

Druhým popisovaný protokolem je RIP, Routing Information Protocol. Stejně jako OSPF patří mezi zástupce interních směrovacích protokolů. Na rozdíl od OSPF je však typu Distance Vector. Dalším rozdílem mezi oběma protokoly je jejich schopnost efektivně fungovat v rozsáhlých sítích s různými kapacitami přenosových spojů, která je v podání protokolu RIP slabá. Nerozděluje, tak jako OSPF, síť na oblasti a pracuje s fixní metrikou, což neumožňuje další dodatečné specifikace volby nejlepší cesty na základě aktuální situace. Pro tyto účely nebyl protokol RIP v době svého uvedení, druhá polovina 80. let 20. století, zamýšlen. Rozdíl mezi protokoly je také v tom, že RIP může být nastaven i na jednotlivých stanicích v síti, které budou mít vlastní směrovací tabulky. Některé nedostatky se podařilo částečně odstranit revizí tohoto protokolu nazvanou RIP v2 a tento se používá v menších sítích dodnes. Nespornou výhodou protokolu RIP je totiž jeho snadná implementace. Dále uvedený popis je pro původní verzi protokolu RIP s tím, že je na konci uveden popis změn ve verzi 2. Údaje jsou čerpány z literatury [7].

4.2.1 Komunikace mezi entitami v síti

Komunikace mezi jednotlivými entitami v síti probíhá prostřednictvím zpráv Request a Response. Tato data jsou přenášena prostřednictvím protokolu UDP, zdrojový i cílový port je shodně nastaven na hodnotu 520. Výměna dat probíhá pouze mezi propojenými entitami.

4.2.2 Metrika a výpočet nejkratších cest

V protokolu OSPF se za metriku považovala tzv. cena. Tato byla odvozena pro každý spoj na základě jeho šířky pásma. Nejlevnější cesta pak byla součtem těchto cen. V protokolu RIP se za metriku považuje počet skoků, hop-count, přes mezilehlé uzly k cílovému. Nejvhodnější cesta je pak ta s nejnižším počtem těchto skoků. To je také důvod, proč není tento protokol vhodný pro fungování v sítích s různou kapacitou přenosových spojů, tuto nezohledňuje a řídí se pouze počtem skoků.

Tato nejkratší cesta je vypočítána pomocí algoritmu Bellman-Ford popsaného v části 3.1.2. Proto, aby se zabránilo vzniku smyček v síti, je počet skoků omezen na hodnotu 15.

Hodnota 16 pak znamená, že daná síť je nedosažitelná. Toto omezení však také snižuje velikost sítě, ve které může být tento protokol používán.

4.2.3 Výměna směrovacích tabulek a časovače

Jak stojí v části 4.2.1, komunikace probíhá pomocí zpráv Request a Response. V první fázi nově připojený uzel vyšle skrze všechna svá rozhranní, na kterých má nakonfigurovaný protokol RIP, zprávu Request. Připojené uzly odpoví zprávou Response, která obsahuje jejich směrovací tabulky. Pokud uzel, který žádal o spojení zatím směrovací tabulku vytvořenou nemá, příjme za svou první příchozí. Další, které k němu dorazí, porovná se svou současnou a použije ty cesty, které mají nižší počet skoků k cíli. Pro upřesnění je ještě třeba uvést, že při příjmu směrovacích informací uzlem dochází ke zvýšení počtu skoků o jedna a toto je potřeba brát v potaz při aktualizací záznamů. Dále pak probíhá aktualizace směrovací tabulky v určitých časových intervalech daných časovači.

Časovače v protokolu RIP ovlivňují jednak dobu zaslání aktualizace směrovací tabulky daným uzlem, a také aktualizace směrovací tabulky jako takové. Aktualizace se zasílají pravidelně každých 30 s. Toto ovšem značným způsobem zatěžuje síť. Aktualizaci směrovací tabulky ovlivňují časovače timeout a garbage-collection timer.

První časovač je nastaven na hodnotu 180 s. Pokud během této doby neobdrží z některého svého rozhranní, kde byla dříve navázána komunikace, žádné informace, je tato cesta považována za neplatnou, ale stále zůstává uchována ve směrovací tabulce. Zároveň se spouští druhý výše zmíněný časovač garbage-collection timer. Tento je nastaven na hodnotu 120 s. Pokud během této doby nejsou informace pro danou cestu aktualizovány, je tato definitivně vymazána ze směrovací tabulky.

4.2.4 Nedostatky protokolu RIP a jejich řešení

S postupným rozšiřováním sítí se ukázal být jako hlavní nedostatek první verze protokolu RIP jeho neschopnost pracovat s různě velkými podsítěmi v rámci jedné třídy adres, chybí tedy podpora VLSM. Dále je první verze tohoto protokolu snadno napadnutelná, protože zde není zavedena žádná vzájemná autentizace uzlů, proto se k síti mohl připojit prakticky kdokoliv.

Druhá verze protokolu, RIP v2 vyřešila problém s různou velikostí podsítí zavedením přenášení informací o masce sítě. Také je zde autentizace uzlů pomocí, nejprve, hesel, které ale bylo možno lehce dešifrovat a později přibylo šifrování md5. Také je zde, pro omezení provozu v síti, zaveden přenos směrovacích tabulek pouze pomocí multicastové adresy, což ze směrování vynechává koncové stanice, ve verzi 1 jsou použity adresy unicastové. Také jsou zde zavedena opatření pro snížení zátěže sítě vlivem opakujících se přenosů aktualizací směrovacích tabulek tím, že jsou přenosy rozloženy v čase a neprobíhají všechny najednou.

4.3 Porovnání protokolů OSPF a RIP

Celkové srovnání obou popsaných protokolů provedeme formou tabulky a následného komentáře jednotlivých hodnot.

Tab. 4.1: Porovnání vlastností protokolů RIP a OSPF

	Protokol		
Vlastnost	**RIP v1**	**RIP v2**	**OSPF**
Typ	Distance Vector	Distance Vector	Link State
Konvergence	Pomalá	Pomalá	Rychlá
Vytížení sítě přenosem dat	Velké	Velké	Malé
Vytížení směrovače	Malé	Malá	Velké
Metrika	Počet skoků	Počet skoků	Cena
Zabezpečení	Žádné	Nízké	Dostatečné

Odlišnost v typu protokolu je v tom, že OSPF směrovač zná topologie celé sítě, tedy oblasti kam spadá, zatímco uzly v protokolu RIP si sice vyměňují celé své směrovací tabulky, ale kompletní topologii sítě neznají.

Konvergence je v případě protokolu OSPF mnohem rychlejší díky rozdělení sítě na menší oblasti, počet směrovačů, mezi kterými probíhá výměna informací, se tak rapidně sníží.

Vytížení sítě je v důsledku přenosu celých směrovacích tabulek, mnohdy od několika uzlů najednou, v podání protokolu RIP, mnohem větší.

Naopak, proces výpočtu nejlevnější cesty Dijkstrovým algoritmem zatíží směrovač více v případě protokolu OSPF.

Metrika je odlišná díky tomu, že protokol OSPF bere v potaz různou kapacitu přenosových spojů, a proto mu pro výběr nejlepší cesty nestačí prostý počet skoků k cíli.

První verze protokolu RIP nebyla nijak zabezpečená, a proto se do sítě mohl připojit kdokoli. Verze dva toto částečně řešila autentizací uzlů pomocí hesel, tato hesla byla ovšem snadno dešifrovatelná U verze dva je využito šifrování md5. OSPF tuto problematiku řešila nejprve autentizací pomocí hesel, poté se přešlo na šifrování md5.

5 ZÁKLADNÍ POPIS TEORIE FRONT

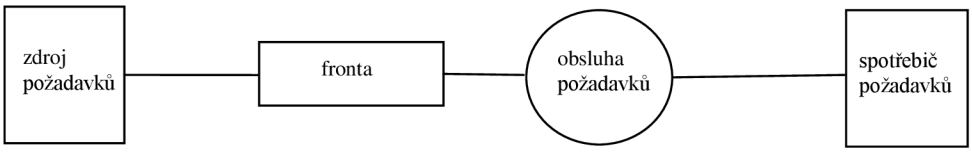
I přes to, že teorie front nepatří do oblasti teorie grafů, může se stát důležitou součásti v procesu optimalizace ať už datových sítí, jako v našem případě, tak i všech dalších struktur, ve kterých dochází k přesunům skrze uzly s omezenou kapacitou. V této kapitole uvedeme základní pojmy této problematiky a naznačíme jejich možné využití.

5.1 Definice teorie front

Teorie front se zabývá zkoumáním procesů, během kterých dochází k průchodu proudu, pro danou problematiku typických objektů skrze obslužná zařízení těchto objektů. V datových sítích to mohou být například směrovače nebo přepínače, jenž ovšem mají omezenou kapacitu množství těchto příchozích objektů, které mohou v jeden okamžik obsloužit. Vlivem těchto omezení se mohou začít tvořit fronty, jenž vedou ke zpoždění nebo dokonce odmítnutí obsluhy. Zařízení, jenž obsluhu vykonává se nazývá systém hromadné obsluhy.

5.2 Systém hromadné obsluhy

Dle literatury [3] a [18] si můžeme tento systém znázornit graficky následovně



Obr. 5.1: Systém hromadné obsluhy

Jednotlivé části Obr. 5.1 můžeme popsat následujícím způsobem

Vstupní proud

Je tvořen proudem objektů, zde si je můžeme definovat jako příchozí požadavky na obsluhu, které jsou generovány zdrojem požadavků. Tyto zdroje mohou být konečné, uzavřený systém, nebo nekonečné, otevřený systém.

Vstupní proud může být

- deterministický
- náhodný
- smíšený

Dále pak mohou být různé požadavky na obsluhu obslouženy s různou prioritou a přicházet jednotlivě nebo ve skupinách.

Fronta

Se skládá z požadavků na obsluhu čekajících na vyřízení. Fronty rozdělujeme na ty s konečnou délkou a nekonečnou délkou, pro velmi vysoký počet příchozích požadavků.

Dále pak dle jejich řádu, který určuje způsob, jakým se příchozí požadavek na vyřízení přesune z fronty do uzlu obsluhy.

- FIFO
- LIFO
- PRI - dle priority
- SIRO - náhodné pořadí

Také u jednotlivých front rozlišujeme jejich disciplínu, tedy fakt, jak dlouho budou čekat požadavky na obsluhu tehdy, když jsou všechny uzly obsluhy vytíženy.

- absolutně netrpělivá - příchozí požadavek ihned rezignuje na obsluhu.
- bez netrpělivosti - požadavek čeká tak dlouho, dokud není obsloužen.
- částečně netrpělivá - prvek ve frontě vyčká předem určenou dobu.

Uzel (stanice) obsluhy

To jest ta část, jenž se stará o obsluhu příchozího požadavku. Těchto uzlů může být v systému hromadné obsluhy hned několik. Propojení uzlů je buď sériové nebo paralelní. Další faktor, který u těchto uzlů sledujeme, je doba vyřízení příchozího požadavku. Ta může být deterministická nebo náhodná.

5.3 Modely systémů hromadné obsluhy

Je samozřejmě výhodnější vytvořit nejprve model systému hromadné obsluhy, ten otestovat a na jeho základě vytvořit skutečný systém, jenž poté nasadíme do provozu. K vytvoření takového to modelu můžeme přistoupit dvěma způsoby.

Analyticky

Tento způsob je použitelný pouze pro jednoduché systémy, ve kterých počítáme se standardním rozdělením náhodných veličin, většinou Poissonovým. Tento návrh probíhá tak, že sestavíme matematický model pomocí diferenciálních rovnic. Tento se poté analyticky řeší a získané vztahy použijeme pro vypočet charakteristik reálného systému. Jak je z tohoto popisu patrné, tento postup by byl pro komplexní systémy neefektivní.

Simulační

S využitím patřičného software simulujeme vybraný systém hromadné obsluhy. Výsledky se poté použijí pro určení charakteristik reálného systému

5.4 Efektivita systémů hromadné obsluhy

Pro kontrolu, zda systém hromadné obsluhy funguje dostatečně efektivně, můžeme využít několika hodnotících parametrů kterými dle literatury [3] to jsou.

- Průměrný počet požadavků v systému; průměrná délka fronty.
- Průměrná doba setrvání požadavku v systému; průměrná doba čekání ve frontě.
- Průměrný počet obsazených (volných) kanálů; koeficient využití (prostoje) kanálu.
- Pravděpodobnost, že všechny kanály jsou obsazené (pravděpodobnost čekání); pravděpodobnost, že všechny kanály jsou volné; pravděpodobnost, že je obsazených (volných) právě *k* kanálů.
- Pravděpodobnost, že v systému je *n* požadavků; pravděpodobnost, že počet požadavků v systému je větší (menší) než *n*.
- Pravděpodobnost, že doba setrvání požadavku v systému bude menší (větší) než daná hodnota; pravděpodobnost, že doba čekání bude menší (větší) než daná hodnota.

5.5 Využití systémů hromadné obsluhy

Jak je psáno již v úvodu, systémy hromadné obsluhy mají své uplatnění ve strukturách, ve kterých dochází k přesunům proudu objektů skrze uzly, jenž mají za úkol tyto objekty zpracovat, ale mají omezenou kapacitu. V datových sítích můžeme za takové uzly považovat směrovače a přepínače, jenž mají za úkol zpracovat proud do nich vcházejících paketů a na základě údajů v nich obsažených je co nejrychleji přesměrovat co možná nejefektivnější cestou k jejich příjemci nebo k dalšímu mezilehlému uzlu.

V případě, že by tohoto nebyly tyto uzly schopny, docházelo by k výpadkům v přenosu dat, která by musela být přenesena znovu, což by samozřejmě komunikaci zpomalilo a znemožnilo využití některých služeb, například těch v reálném čase. Je tedy důležité, aby tyto uzly byly zvoleny adekvátně k očekávanému provozu v síti.

Kapacita těchto uzlů může být považována za další hodnotící kritérium při výběru, například nejkratší cesty, pokud si bude vzdálenost vícero cest rovna. Další informace o této problematice nalezneme v literatuře [3] a [18].

6 PRAKTICKÉ UKÁZKY POUŽITÍ TEORIE GRAFŮ

Tato kapitola bude prezentovat praktické ukázky výše popsané teorie, kterou lze využít při návrhu a optimalizaci struktury datové sítě.

Cílem je vytvořit takový mechanismus, který na základě uživatelem definovaných parametrů vytvoří takový návrh sítě, který bude v maximální možné míře splňovat všechny definované podmínky. Zároveň je našim cílem to, aby výsledný návrh byl co nejúspornější, co se týče celkových nákladů na něj potřebných.

Pro názornost ukázky je využito menší sítě o nižším počtu uzlů, konkrétně osmi, nicméně daný postup, lze aplikovat i na rozsáhlejší struktury sítí.

Při návrhu budeme vycházet ze zadaných parametrů a požadavků na funkčnost sítě, kterými jsou:

1. Pevně dané, geograficky, rozmístění přepojovacích uzlů.
2. Je dána maximální možná chybovost spoje, $P_{zmax} = 1*10^{-4}$.
3. Je zadáno maximální možné zpoždění na libovolné trase v síti mezi koncovými uzly, $T_{max} = 10$ ms.
4. Uživatel zadá jednotlivé kapacity přístupových linek do přepojovacích uzlů.
5. Cena přepojovacího uzlu je složena ze zadaných hodnot nákladů na samotné přepojení datové jednotky (ve většině případů paketu) a nákladů na paměť potřebnou ke správné funkci přepojovacího uzlu (zamezení zahazovaní datových jednotek).
6. Cena spojů je dána zadanými náklady na jeden metr spoje. Uživatel definuje délku tohoto spoje. A nákladů na kapacitu daného spoje. Uživatel definuje kapacitu tohoto spoje.

Postup návrhu se bude skládat z jednotlivých dílčích návrhů, které budou určeny pro splnění jednotlivých požadavků a výsledného návrhu, který by měl být ideálním kompromisem mezi těmito dílčímu návrhy.

Jednotlivé návrhy budou vzájemně porovnávány z pohledu celkových nákladů a dalších parametrů aby bylo jasně patrné, jak výrazně se mohou lišit a jak ovlivňují celkový výsledek.

Dále pak budou provedeny testy výsledného návrhu, které budou demonstrovat situace, jakými mohou být například výpadek spoje nebo uzlu a to, jak tyto situace ovlivní funkčnost navržené sítě.

V této kapitole budu využito programu Matlab a jeho vestavěných funkcí.

6.1 Způsob zadávání vstupních parametrů návrhu

V této kapitolu je definováno, jakým způsobem bude potencionální uživatel zadávat vstupní parametry algoritmu vzhledem k jeho spuštění v prostředí programu Matlab.

6.1.1 Parametry spoje mezi dvěma přepojovacími uzly

Spoj mezi dvěma přepojovacími uzly definujeme jeho počátečním a koncovým uzlem pomocí řádkových vektorů. Dále musíme zadat délku jednotlivých spojů, v metrech, což bude sloupcový vektor. Stejně tak tomu bude i v případě hodnot kapacit jednotlivých spojů, jednotku

definujeme samostatně. Poledními údaji jsou hodnoty nákladů na jeden metr spoje a na jednotku kapacity v libovolné měně. Toto budou prostá desetinná čísla.

1. Počátek - pocatek = [1 1 1 2 2 3 3 3 4 4 5 5 6 7];
2. Konec - konec = [2 3 4 3 6 4 5 6 5 7 6 7 8 8];
3. Délka - delka = [2000; 1500; 2200; 1400; 800; 500; 900; 700; 1000; 1500; 1100; 1250; 1000; 2500];
4. Kapacita - BW = [500; 500; 500; 250; 250; 500; 500; 500; 100; 150; 100; 200; 100; 200];
5. Jednotka kapacity - jednotkaBW = 'Mbps';
6. Cena za 1 m - cenaza1m = 1;
7. Cena za 1 bit - cenaza1b = 0,0003;

U prvních dvou bodů si můžeme všimnout, že směr spoje je zadáván od zdroje k cíli, což by mohlo působit tak, že jde o orientovaný graf, tak jak je popsán v kap. 2.1. Faktem ale je, že data mohou ve spoji proudit obousměrně a jde jen o nutný postup vzhledem k Matlabu a jeho nástroji biograph, který umožňuje využití algoritmů teorie grafů implementovaných v Matlabu. Zároveň se nejedná o plné propojení mezi všemi uzly, full mesh, ale graf je souvislý, viz. kap. 2.2.4.

Délka a kapacita je pro každý spoj zadána samostatně. Co se týče jednotky kapacity, tak definována jako 'Bps', 'Mbps' nebo 'Gbps'.

Hodnoty nákladů na jeden metr délky a jeden bit kapacity definuje uživatel v závislosti na nabídce dodavatele. Zde je návrh omezen na jednotnou cenu za délku pro všechny spoje. Je samozřejmě možné, že se náklady budou u jednotlivých spojů lišit, ale v této verzi pracuje algoritmus s jednotnou cenou všech spojů.

6.1.2 Parametry přepojovacího uzlu

U přepojovacích uzlů nás především zajímá kapacita vstupních a výstupních spojů. Od této se totiž odvíjí množství datových jednotek, které bude muset uzel zpracovat a které budou muset být uloženy v jeho vstupní a výstupní paměti.

V případě této návrhové metody je maximální množství obsloužených požadavků přepojovacím uzlem vztaženo ke kapacitě jeho přístupové linky, to jest lince mezi ním a konečným uživatelem, jakožto k fixnímu spoji. Kapacitu těchto linek zadáme jako sloupcový vektor.

Potřebná velikost paměti uzlu se odvíjí od kapacity přístupové linky a celkové kapacity všech připojených spojů transportní sítě. Tyto pak v závislosti na směru toku dat tvoří vstupní a výstupní paměť uzlu.

Celkové náklady jsou součtem nákladů na možné množství obsloužených požadavků na obsluhu, jednotkovou cenu za přepojení jednoho požadavku zadává uživatel, a nákladů na potřebnou paměťovou kapacitu danou celkovým množstvím příchozích požadavků na obsluhu, neboli datových jednotek. Jednotkovou cenu opět definuje uživatel.

Jak je z popisu patrné, jedná o systém hromadné obsluhy, který je popsán v kapitole 5.3.

Je samozřejmě pravda, že velikost paměti se bude měnit nejen s počtem příchozích požadavků, ale také s jejich velikostí. V případě tohoto návrhu je ovšem paměť dimenzována na fixní velikost příchozího požadavku a to 1500 B, maximální velikost datové části rámce.

Jednotlivé vstupní parametry pro určení nákladů na přepojovací uzel jsou zadány následujícím způsobem:

1. Kapacita přístupových linek - BWpl = [500; 500; 500; 500; 250; 250; 250; 250];
2. Celková kapacita připojených spojů transportní sítě - BWcel je určena samostatným algoritmem, který bude dále popsán.
3. Cena za obsloužení příchozího požadavku - cenaza1pa = 0.25;
4. Cena za paměťový blok pro uložení příchozího požadavku - cenaza1pb = 0.3;

Dále pak samozřejmě definujeme jednotku kapacity, ale ta je stejná jako v předchozím případě popsaném v kapitole 6.1.1.

Pro správný chod všech dále popisovaných procedur jsou zapotřebí i další vstupní parametry, ale tyto budou popsány v příslušných částech textu.

Také je dobré zmínit, že všechny výše uvedené hodnoty vstupních parametrů jsou pouze demonstrační a mohou být zadány libovolně v rámci daných omezení.

6.2 Výpočet celkových nákladů návrhu

Jedním z prvních bodů plánování každého projektu je kalkulace celkových nákladů na něj potřebných. Jinak tomu není ani v případě výstavby datové sítě. Tím spíše, že je naší snahou celkové náklady minimalizovat, jak je uvedeno na počátku kapitoly 6, potřebujeme tuto hodnotu znát pro možné porovnání v rámci jednotlivých optimalizací.

Celkové náklady jsou rozděleny, pro lepší přehlednost, na náklady vynaložené na spoje, to jest samotné přenosové médium, a náklady na přepojovací uzly. Výsledná hodnota je pak dána jejich součtem.

6.2.1 Výpočet vynaložených nákladů na spoje

Náklady na spoje jsou v našem případě tvořeny součtem ceny za fyzickou délku spoje a cenou za kapacitu tomuto spoji náležícímu. Způsob zadávání patřičných parametrů je popsán v kapitole 6.1.1. Jak je uvedeno výše, nebereme v potaz rozdílné náklady spojů vzhledem například k terénu, kterými jsou vedeny atd.

Výpočet těchto nákladů je proveden vytvořeným programem zapsaným v m-file souboru nazvaném CenaLinky. Jeho struktura je následující:

1. Úvod tvoří nápověda a ošetření chybových vstupních stavů.
2. Je proveden výpočet ceny vzhledem k délce násobením prvků vektoru délky jednotkovou cenou dle následující rovnice (6.1):

$$\text{cenadelky} = \text{delka}.* \text{cenaza1m} \tag{6.1}$$

3. Je proveden výpočet ceny kapacity násobením prvků vektoru kapacity jednotkovou cenou dle následující rovnice (6.2), vztažené k jednotce kapacity Mbps:

$$cenakapacity = (BW * cenaza1b) * 1 * 10^6 \quad (6.2)$$

4. Je sečtena hodnota ceny za délku a kapacitu, která je uložena ve sloupcovém vektoru.

6.2.2 Výpočet vynaložených nákladů na přepojovací uzly

Jak je uvedeno výše, tak cena přepojovacího uzlu se skládá z ceny za obsloužení příchozího požadavku a ceny potřebné paměti. Bližší popis spolu s příkladem zadaných parametrů je uveden v kapitole 6.1.2.

Výpočet této ceny je proveden vytvořeným programem zapsaným v m-file souboru nazvaném CenaZaUzel. Jeho struktura je následující:

1. Úvod tvoří nápověda a ošetření chybových vstupních stavů.
2. Vzhledem k použité jednotce kapacity je proveden přepočet na základní jednotku a to jak u kapacity přístupové linky, tak i celkové kapacity připojených spojů transportní sítě.
3. Určujeme možné množství příchozích požadavků, velikost tohoto požadavku je 1500 B.
4. Vzhledem k možnému množství příchozích požadavků z přístupové linky uzlu určíme cenu za přepojení příchozího požadavku.
5. Sečteme možný počet příchozích požadavků z přístupové linky a transportní sítě a vynásobením tuto hodnotu jednotkovou cenou paměti, resp. ceny paměťového bloku pro uchování požadavku na obsluhu.
6. Vyjádříme celkové náklady na přepojovací uzel.

V kapitole 6.2.1 je uvedeno, že celková kapacita připojených spojů transportní sítě je určena samostatným programem. Ten je zapsán v souboru CelkovaBWdoUzlu.

Pro výpočet je využito matice sousednosti, popsané v kapitole 2.1, ve které byly hodnoty jedna změněny na hodnoty kapacit. Postup je následující:

1. Vytvoříme řídkou matici spojů ohodnocených jejich kapacitami.
2. Převedeme tuto matici na matici sousednosti.
3. V cyklu je proveden výpočet jednotlivých hodnot, které jsou uloženy ve sloupcovém vektoru.

6.2.3 Postup výpočtu celkových nákladů návrhu

Výpočet celkových nákladů návrhu je proveden programem zapsaným v soboru CelCenaSpoje.

Jeho struktura je následující:

1. Úvod tvoří nápověda a ošetření chybových vstupních stavů.
2. Zavoláme programy výpočtu ceny spojů a uzlů popsaných v kapitolách 6.2.1 a 6.2.2.
3. Vypočítáme sumy jednotlivých hodnot a sečteme je.

Tato celková hodnota nám bude následně sloužit pro porovnání vzhledem k celkovým nákladům po provedení jednotlivých optimalizací.

Co se týče nutnosti spouštění jednotlivých výpočtů, tak je můžeme spustit samostatně, ale v případě, že nás zajímá pouze celková cena, stačí spustit pouze tuto konkrétní úlohu. V obou případech je nutné provést výpočet celkové kapacity připojených spojů do uzlů z transportní sítě, tedy spustit program CelkovaBWdoUzlu.

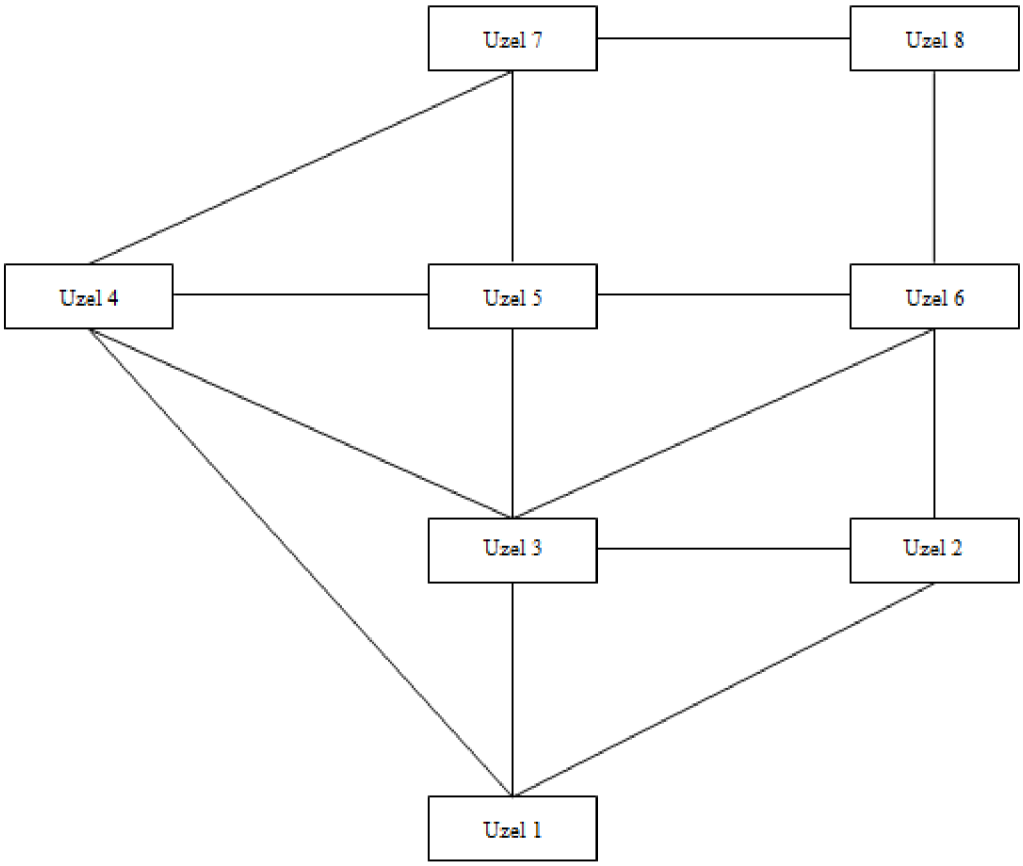
6.3 Optimalizace vzhledem celkovým nákladům

Jak je uvedeno v úvodu kapitoly 6, je naší snahou minimalizovat celkové náklady nutné na realizaci návrhu sítě. Jako první a nejsnazší možnost se jeví hrubé omezení nákladů a snaha vybudovat infrastrukturu tak, že pokryje všechny potřebné uzly s využitím těch nejméně nákladných spojů. Cílem této kapitoly je tedy demonstrace toho, jak takový návrh vytvořit, jaké jsou jeho náklady a jaké nevýhody takto striktní řešení přináší.

6.3.1 Nejméně nákladné řešení v podobě minimální kostry a výchozí stav

Pro nalezení naprostého minima nákladů je možné s úspěchem využít tzv. minimální kostru grafu, popis v kapitolách 2.2.8 a 3.2. Jednotlivé hrany grafu, tedy spoje v síti v našem případě, jsou ohodnoceny náklady, které byly zjištěny pomocí programů popsaných v kapitole 6.2.

Nejprve si předvedeme, jak vypadá výchozí stav návrhu, tedy plné propojení, prosím neplést si s tzv. full mesh sítí, kde je spoj mezi každým uzlem grafu.



Obr. 6.1: Výchozí stav návrhu

Jak je z Obr. 6.1 patrné, tak výchozí stav je značné neefektivní. Pro úplnost jsou dále uvedeny jeho konkrétní parametry a náklady.

Tab. 6.1: Parametry spojů výchozího návrhu

Počátek	1	1	1	2	2	3	3	3	4	4	5	5	6	7
Konec	2	3	4	3	6	4	5	6	5	7	6	7	8	8
Délka [km]	2	1,5	2,2	1,4	0,8	0,5	0,9	0,7	1	1,5	1,1	1,25	1	2,5
Kapacita [Mbps]	500	500	500	250	250	500	500	500	100	150	100	200	100	200

Obr. 6.1 zachycuje pouze spoje transportní sítě, které budou předmětem optimalizace neboť přístupové linky jsou fixně dané a není možno je např. odstranit. Je možné, že se výsledné grafy vygenerované programy díky použiti objektu biograph budou lišit vzhledově, pro zamezení křížení spojů, ale fyzická poloha uzlů je stále stejná. Co se týče parametrů přepojovacích uzlů, jsou následující.

Tab. 6.2: Parametry přepojovacích uzlů výchozího návrhu

Uzel	1	2	3	4	5	6	7	8
Kapacita přís t. linky [Mbps]	500	500	500	500	250	250	250	250
Kapacita linek trans. sítě do uzlů [Mbps]	1500	1000	2250	1250	900	950	550	300

Závěrečná tabulka obsahuje hodnoty nákladů a to jak celkových, tak podíl spojů a uzlů.

Tab. 6.3: Celkové náklady výchozího návrhu

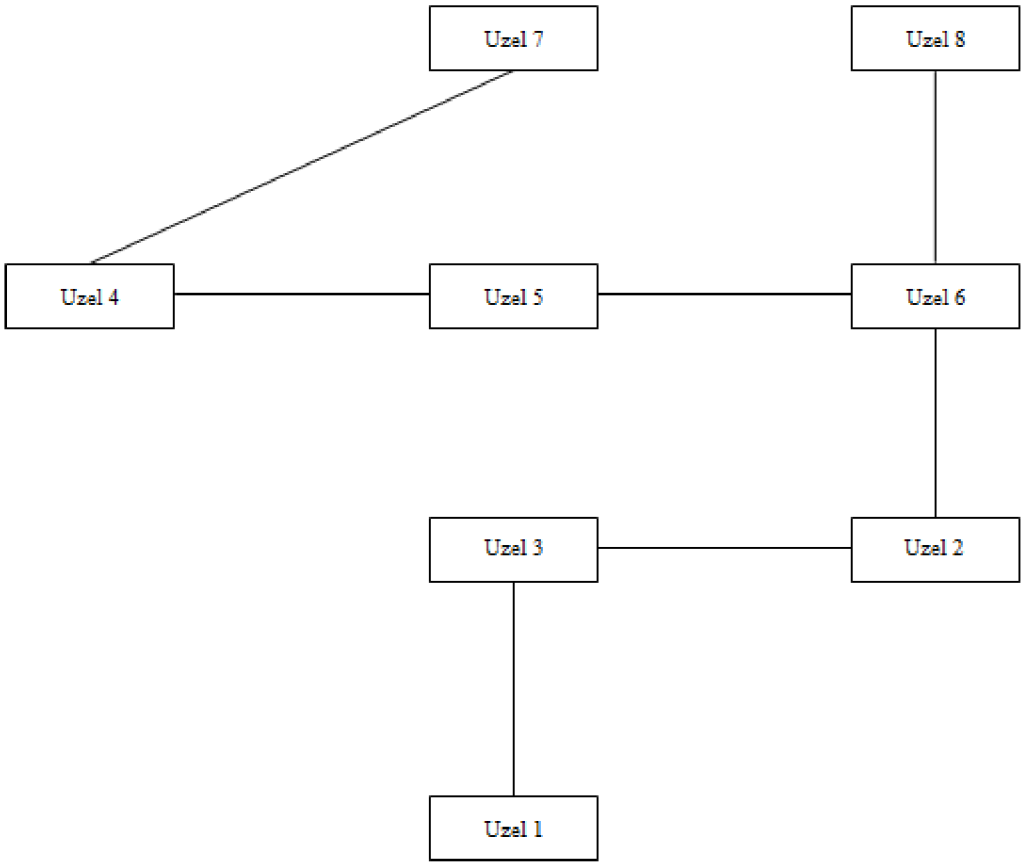
Náklady potřebné na přepojovací uzly [Kč]	355000
Náklady potřebné na spoje [Kč]	1323350
Celkové náklady [Kč]	1675350

Ještě jednou bude dobré uvést, že uvedené hodnoty jsou pouze demonstrační a budou se lišit v závislosti na konkrétních parametrech daného návrhu. I tak je ale možné téměř s jistotou říci, že nejnákladnější položkou bude vybudování fyzických spojů.

Nyní zpět k hledání minimálních nákladů prostřednictvím minimální kostry grafu. Tento úkon je součástí programu zapsaného v souboru nazvaném OptDleCeny, vlastně je to jeho počáteční fáze. Postup je následující:

1. Úvod tvoří nápověda a ošetření chybových vstupních stavů.
2. Je volána funkce CelCenaSpoje pro zjištění nákladů na jednotlivé spoje daných délkou a jejich kapacitou..
3. Je vytvořena řídká matice všech spojů ohodnocených náklady dle bodu 2.
4. Je využito vestavěné funkce Matlabu pro nalezení minimální kostry grafu na základě řídké matice z bodu 3. Pro funkčnost algoritmu nalezení minimální kostry, konkrétní název je graphminspantree a využívá Jarníkova algoritmu popsaného v kapitole 3.2.1, je potřeba převést vstupní matici na dolní trojúhelníkovou matici. Výsledkem je pak sloupcový vektor spojů minimální kostry grafu.
5. Z těchto dat je pak vykreslen graf.

Výsledná podoba struktury sítě dle minimální kostry je následující:



Obr. 6.2: Minimální kostra grafu na základě ceny

Jak je z Obr. 6.2 patrné, je využito minimální množství spojů a jsou použity ty nejméně nákladné. Konkrétně se jedná o tyto:

Tab. 6.4: Spoje použité v minimální kostře grafu dle ceny

Počátek	1	2	2	4	4	5	6
Konec	3	3	6	5	7	6	8
Délka [km]	1	1,4	0,8	1	1,5	1,1	1
Kapacita [Mbps]	500	250	250	100	150	100	100

Náklady na návrh minimální kostry dle ceny jsou následující:

Tab. 6.5: Náklady minimální kostry grafu dle ceny

Náklady minimální kostry [Kč]	653300
Z toho na uzly [Kč]	210000
Na spoje [Kč]	443300
Rozdíl proti vých. stavu [Kč]	-1022050
Procentuální úspora nákladů [%]	61

Jak je z Tab. 6.5 patné, pokles nákladů je skutečně drastický, jedná se o hodnotu 61 %. Zároveň nám ale Tab. 6.4 ukazuje, že jsou využity spoje s minimální kapacitou, což samozřejmě omezuje možné množství přenášených dat v síti, zvyšuje hodnotu zpoždění a omezuje na minimum možnost implementace aparátu jako je např. QoS. Dále můžeme z Obr. 6.2 vypozorovat, že síť je zbavena veškeré redundance, což je proti zásadám správného návrhu sítě.

Konkrétní hodnota zpoždění je uvedena v následující tabulce:

Tab. 6.6: Zpoždění v návrhu minimální kostry dle nákladů

Limit zpoždění [ms]	10
Maximální zpoždění [ms]	10,789
Průměrné zpoždění [ms]	5,673

Vidíme, z údajů v Tab. 6.6, že průměrné hodnoty zpoždění jsou na poměrně přijatelné úrovni, ale jsou zde i cesty které limit zpoždění překračují. Propustnost sítě je omezena úzkými místy na minimální hodnotu.

Postup výpočtu zpoždění bude uveden v kapitole 6.4.

6.3.2 Doplnění návrhu minimální kostry grafu dle ceny o redundanci

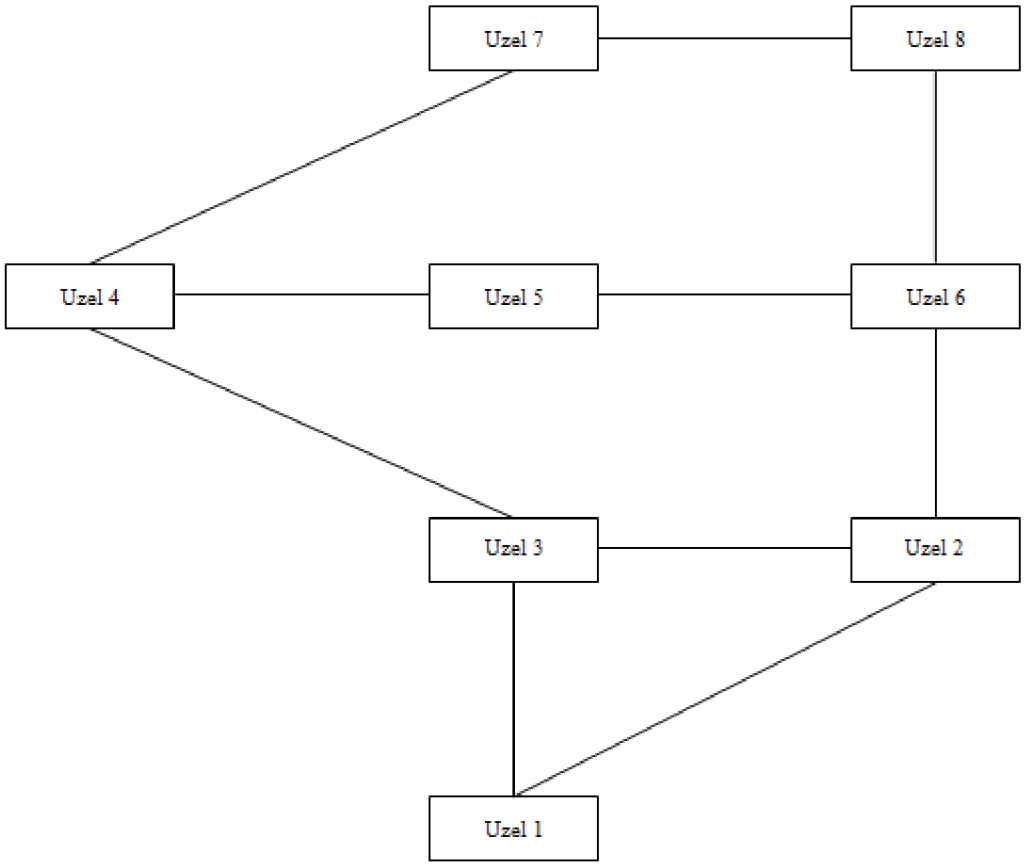
Další fází programu zapsaného v OptDleCeny je doplnění návrhu minimální kostry o redundanci. Toto je provedeno následujícím způsobem:

1. Je určen stupeň všech uzlů, které se vyskytují v návrhu minimální kostry dle ceny.
2. Hledáme uzly se stupněm menším než dva a jejich společné spoje. což je nejefektivnější cesta jak přidat chybějící spoje.
3. Pokud takové spoje existují, jsou dodány do návrhu.
4. Opět probíhá kontrola na stupeň uzlu.
5. V případě, že nebyly nalezeny společné spoje, musí být nalezeny nejvhodnější, nejlevnější, spoje pro dané uzly v matici všech spojů.
6. Toto probíhá v cyklu o dvou opakováních, v každé iteraci je provedena kontrola stupně uzlu, čímž je zajištěno, že žádný z uzlů nebude míst stupeň menší než dva.
7. Nalezené spoje jsou přidány do matice spojů návrhu.

Nyní by se mohlo zdát, že vzhledem ke stupni uzlů je již dodána vhodná redundance. Jak ale bylo během testů s rozdílnými parametry výchozího návrhu zjištěno, může nastat situace, že síť bude rozdělena na více, většinou dva, souvislých útvarů propojených pouze jedním spojem, jehož výpadek by znamenal ztrátu konektivity pro několik uzlů najednou. Proto závěrečná část procesu přidání redundantních spojů spočívá v tom, testujeme tuto možnost a hledáme řešení:

1. Z matice aktuálních spojů vybereme její pravou horní polovinu, ta obsahuje zpětné spoje, které zaručí redundantní souvislost sítě.
2. V této submatici hledáme v cyklu sloupce, které mají pouze nulové prvky.
3. Pokud je takový sloupce nalezen, hledám v odpovídající submatici zbylých dostupných spojů nejvhodnější variantu pro doplnění.
4. Vhodné spoje jsou doplněny do matice aktuálních spojů.

Tímto je zaručena nadbytečnost v síti při doplnění optimálních spojů. Výsledná podoba návrhu s redundancí vzhledem k minimálním nákladům je následující:



Obr. 6.3: Konečný návrh struktury sítě dle minimálních nákladů

Z Obr. 6.3 můžeme vypozorovat, že byly přidány tři nové spoje. Konkrétně se jedná o:

Tab. 6.7: Redundantní spoje návrhu dle minimálních nákladů

Počátek	1	3	7
Konec	2	4	8
Délka [km]	2	0,5	2,5
Kapacita [Mbps]	500	500	200

Spoje mezi uzly 1 a 2, 7 a 8 byly přidány jako společné spoje, zbylý spoj mezi uzly 3 a 4 byl přidán z důvodu zajištění souvislosti sítě při výpadku spoje mezi uzly 2 a 6, který by znamenal odříznutí uzlů 1, 2 a 3.

Výsledný návrh dle minimálních nákladů je pak tvořen těmito spoji:

Tab. 6.8: Spoje výsledného návrhu dle minimálních nákladů

Počátek	1	1	2	2	3	4	4	5	6	7
Konec	2	3	3	6	4	5	7	6	8	8
Délka [km]	2	1,5	1,4	0,8	0,5	1	1,5	1,1	1	2,5
Kapacita [Mbps]	500	500	250	250	500	100	150	100	100	200

Hodnoty zpoždění a propustnosti jsou v tomto případě tyto:

Tab. 6.9: Zpoždění v návrhu minimálních nákladů

Limit zpoždění [ms]	10
Maximální zpoždění [ms]	6,075
Průměrné zpoždění [ms]	4,313

Hodnoty v Tab. 6.9 ukazují, že předání redundantních spojů značným způsobem snížilo zpoždění v síti, porovnat jej můžeme s hodnotami v Tab. 6.6.

Tab. 6.10: Využití kapacit spojů a propustnost v návrhu dle minimálních nákladů

Počátek	1	2	2	3	4	4	5	6	7
Konec	3	3	6	4	5	7	6	8	8
Délka [km]	1,5	1,4	0,8	0,5	1	1,5	1,1	1	2,5
Kapacita [Mbps]	500	250	250	500	100	150	100	100	200
Využitá kapacita [Mbps]	250	150	100	650	300	600	400	600	800

Pokud jde o hodnoty využití kapacit spojů v návrhu a propustnost sítě, tak zde je situace o poznání horší, jak ukazují hodnoty v Tab. 6.10, a přenos většího množství dat by byl komplikovaný. Postup těchto výpočtů bude uveden v kapitole 6.5.

Konečné náklady tohoto návrhu jsou pak následující

Tab. 6.11: Konečné náklady návrhu dle minimálních nákladů

Náklady minimální kostry [Kč]	1078300
Z toho na uzly [Kč]	270000
Na spoje [Kč]	808300
Rozdíl proti vých. stavu [Kč]	-600050
Procentuální úspora nákladů [%]	35

Jak ukazuje Tab. 6.11, přidání pouhých tří spojů vedlo k nárůstu nákladu o 35 %, přičemž toto bylo nejefektivnější řešení. Za tímto nárůstem stojí nutnost přidání spojů o poměrně vysoké kapacitě, jiné nebyly uživatelem na počátku definovány.

Celkově můžeme tento návrh považovat za funkční, a v určitých ohledech splňující zadané požadavky, ale pro plné nasazené takovéhoto návrhu by byly potřeba další modifikace, především v oblasti propustnosti.

6.4 Optimalizace vzhledem ke zpoždění v síti

Jedním ze zásadních parametrů návrhu sítě je výsledná doba zpoždění při přenosu dat, především pak mezi koncovými uživateli. I v našem případě toto bereme na zřetel a mezi vstupními parametry uvedenými v úvodu kapitoly 6 figuruje požadavek na maximální dobu zpoždění v hodnotě 10 ms. Je samozřejmě pravda, že nejnižší zpoždění je v síti tehdy, když jsou komunikující uzly navzájem přímo spojeny, ale takovéto řešení je dosti neefektivní.

Tím se dostáváme k popisu další provedené optimalizace, a to vzhledem k minimalizaci zpoždění v síti. Optimalizace čistě z pohledu minimalizace zpoždění je užitečná tehdy, pokud v síti provozujeme služby jako například VoIP a další, které jsou na zpoždění citlivé a jeho neúměrná hodnota znamená neschopnost provozovat takovéto služby v patřičné kvalitě.

Program, který vytváří návrh vzhledem k minimalizaci zpoždění je zapsán v souboru OptDleZpozdeni.

6.4.1 Určení zdrojů zpoždění v síti a jejich hodnot

Jako první si musíme určit zdroje zpoždění v síti a jakou měrou k celkovému zpoždění přispívají. Zde práce vychází z údajů zjištěných v [1].

Zdroje zpoždění v síti můžeme rozdělit do dvou hlavních skupin.

Tou první je zpoždění způsobené šířením signálu skrze přenosové médium. Pokud bereme v potaz pouze pevná média, vynecháme bezdrátové technologie, tak je toto zpoždění v průměru 500 μs na 100 km délky média. Tato hodnota je přibližně stejná pro většinu dnes využívaných technologií.

Druhou skupinou zpoždění tvoří ta, která generují přepojovací uzly v síti. Jsou složeny především z doby potřebné pro obsluhu příchozího požadavku, přepojení datové jednotky, a zdoby nutné pro přijetí příchozího požadavku z příchozího spoje a jeho odeslání do odchozího spoje. Jedná se o tzv. serializační zpoždění. Pokud jde o dobu nutnou pro obsluhu příchozího požadavku, tak pro námi využívanou velikost tohoto požadavku, 1500 B, je to v průměru 1,4 ms v případě směrovače. Co se týče serializačního zpoždění, tak toto se odvíjí od kapacity spoje, ke kterému se vztahuje prováděná operace. Jeho velikost se určí dle následující rovnice (6.3):

$$\frac{datová\_jednotka\_[b]}{kapacita\_spoje\_[bps]} \tag{6.3}$$

Jak můžeme vyčíst z rovnice (6.3), tak v případě linek o nízké kapacitě je serializační zpoždění poměrné značné a velkou měrou přispívá k celkovému zpoždění.

Pokud se tedy jedná o celkové zpoždění spoje v transportní síti, tak příslušný program pracuje s následující rovnicí (6.4):

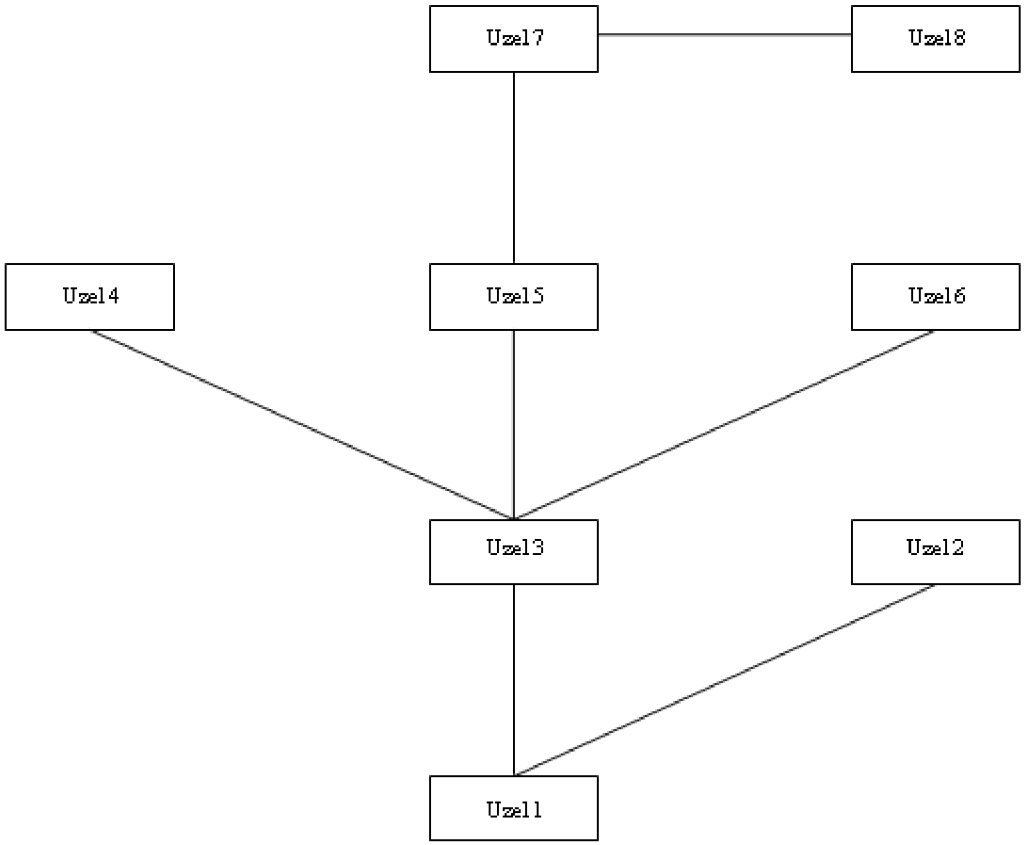
$$zpTransportSite = 2*serializacniZpozdeni + zpozdeniSpoje \tag{6.4}$$

V případě zpoždění v síti nesmíme opomenout zahrnout do výpočtů přístupové linky uzlů, které se na celkovém zpoždění také podílejí. Zpoždění dané obsluhou příchozího požadavku

přepojovacím uzlem započítáváme na základě toho, skrze kolik mezilehlých uzlů je daná cesta vedena. Postu je vysvětlen v následující kapitole.

6.4.2 Řešení v podobě minimální kostry dle zpoždění

Vzhledem k naší snaze o minimalizaci nákladů bude prvním uskutečněným řešením provedení návrhu minimální kostry grafu, tentokrát dle hodnot zpoždění. Vytvoříme tedy matici sousednosti, která tentokrát obsahuje hodnoty zpoždění v transportní síti, a z ní následně řídkou matici spojů. Další postup vytvoření minimální kostry grafu je pak totožný s tím, který je popsán v kapitole 6.3.1.



Obr. 6.4: Minimální kostra grafu na základě zpoždění

Na základě použitých spojů pro návrh určíme hodnotu zpoždění mezi všemi uzly sítě. K tomuto využijeme algoritmus hledání všech nejkratších cest v grafu, popis v kapitole 3.1.3. Jeho implementace v Matlabu je provedena algoritmem graphallshortestpaths. Výsledkem je matice hodnot zpoždění mezi všemi uzly sítě.

Nyní musíme do hodnot zpoždění zahrnout zpoždění přístupových linek.

1. V cyklu přičítáme ke každému řádku matice hodnot celkového zpoždění serializačního zpoždění jednotlivých přístupových linek uzlů. Nesmíme zapomenout ke každému řádku přičíst navíc znovu hodnotu výchozího serializačního zpoždění.
2. Prvky na hlavní diagonále matice celkového zpoždění změníme na nulu, to odpovídá zpoždění výchozího uzlu daného řádku k sobě samému.

Abychom nyní určily přesné zpoždění mezi koncovými účastníky, musíme matici zpoždění doplnit o zpoždění způsobené mezilehlými uzly v jednotlivých cestách.

Toto provedeme následovně:

1. Hledáme v cyklu nejkratší cesty od jednotlivých uzlů ke všem ostatním pomocí algoritmu graphshortestpath.
2. Pomocí jednoho z výstupních parametrů této funkce ukládáme do proměnné, stále v cyklu, konkrétní cesty složené z dílčích úseků ohodnocených zpožděním.
3. Zjišťujeme délku vektorů uložených v bodě 2.
4. Na základě délky vektoru přičteme k patřičné pozici matice všech zpoždění násobek hodnoty zpoždění generovaného mezilehlými uzly.
5. Hodnoty na hlavní diagonále matice zpoždění převedeme na 0.
6. Matice převedeme na horní trojúhelníkovou matici.

Výsledky návrhu minimální kostry grafu dle zpoždění jsou následující:

Tab. 6.12: Spoje použité v minimální kostře grafu dle zpoždění

Počátek	1	1	3	3	3	5	7
Konec	2	3	4	5	6	7	8
Délka [km]	2	1,5	0,5	0,9	0,7	1,25	2,5
Zpoždění [ms]	0,0580	0,0555	0,0505	0,0525	0,0515	0,1263	0,1325

Tab. 6.13: Parametry zpoždění všech cest návrhu minimální kostry dle zpoždění

Limit zpoždění [ms]	10
Maximální zpoždění [ms]	8,897
Průměrné zpoždění [ms]	5

Tab. 6.14: Náklady minimální kostry grafu dle zpoždění

Náklady minimální kostry [Kč]	1161850
Rozdíl proti vých. stavu [Kč]	- 516500
Procentuální úspora nákladů [%]	30

Hodnoty v Obr. 6.4, Tab. 6.12 a Tab. 6.14 můžeme porovnat s hodnotami uvedenými v kapitole 6.3.1. Už zběžným pohledem na tyto hodnoty zjistíme, že bylo využito spojů s podstatně vyšší kapacitou, což zásadním způsobem ovlivnilo výslednou cenu tohoto návrhu. Procentuální úspora nákladů je zde pouze 30 %, u minimální kostry dle nákladů to bylo 61 %.

Použité spoje ovšem umožňují to, že zpoždění již v případě minimální kostry je takové, aby splňovalo vstupní podmínky návrhu. Je zde samozřejmě možnost, že dojde k překočení hodnoty limitu zpoždění a v tom případě je nutné takové spoje identifikovat a pokusit se o nápravu.

Zjištění, o které se spoje se konkrétně jedná provedeme následujícím způsobem:

1. Vytvoříme si nulovou čtvercovou matici o rozměru strany dle počtu uzlů v síti.
2. Hledáme indexy hodnot v matici všech zpoždění, které překračují povolený limit.
3. Dle zjištěných indexů ukládáme hodnoty překračující limit do vytvořené nulové matice.
4. Tuto matice převedeme na řídkou matici, čímž získáme přesnou identifikaci spoje.

Postup v případě překročení hodnoty limitu zpoždění je uveden v kapitole 6.4.3 u návrhu s redundancí, kde je potřeba jej uplatnit vzhledem k jeho statusu optimálního návrhu vzhledem k dané veličině.

6.4.3 Doplnění návrhu minimální kostry grafu dle zpoždění o redundanci

Stejně jako v případě kapitoly 6.3 budeme nyní pokračovat v optimalizaci návrhu tím, že k minimální kostře grafu přidáme redundantní spoje. Postup tohoto procesu je totožný jako v kapitole 6.3.2. V tomto případě jej pouze drobně rozšíříme a přidáme i takový spoj, který bude redundantně spojovat horní a dolní polovinu uzlů, která sice již díky jednotlivým testům propojená je, ale díky tomuto spoje vznikne mezi nimi mnohem kratší cesta, což sníží hodnoty zpoždění. Je samozřejmě pravda, že pro minimalizaci zpoždění je nejvhodnější plné propojení sítě, které omezuje počet mezilehlých uzlů, ale takové řešení je neefektivní.

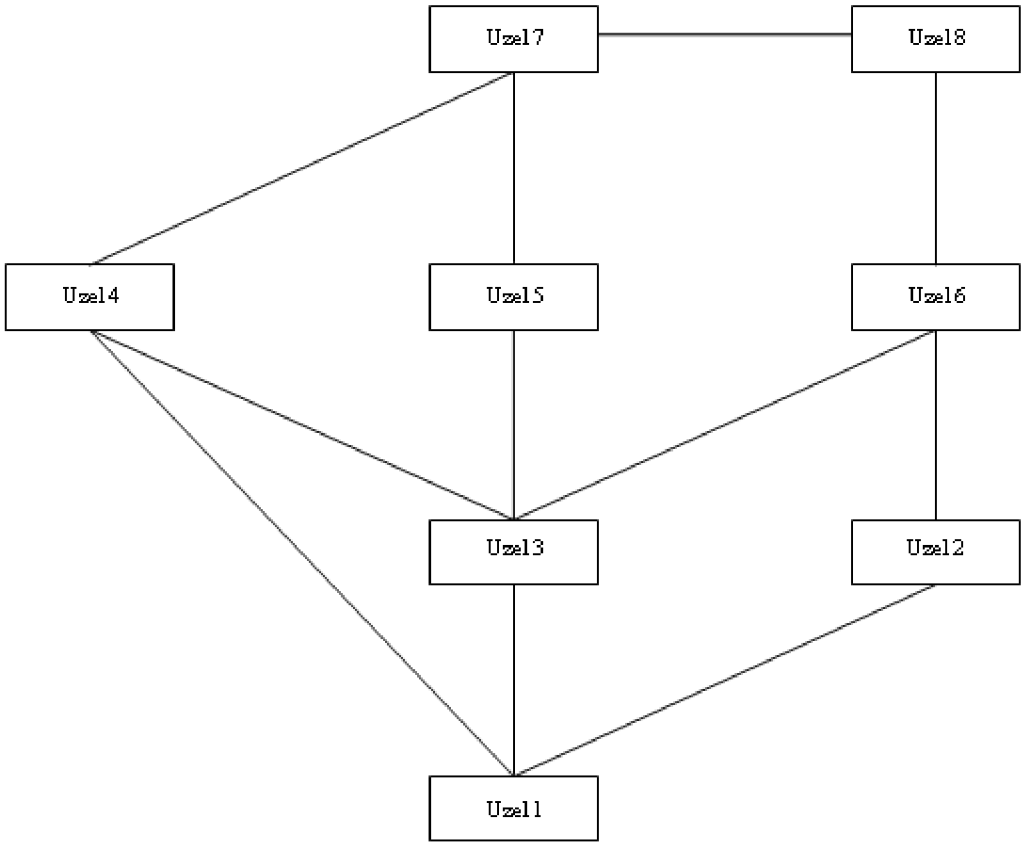
Opět si vytvoříme matici zpoždění mezi všemi uzly v síti pomocí algoritmu graphallshortestpaths a k těmto hodnotám zpoždění přičteme zpoždění způsobené přístupovými linkami a mezilehlými uzly jednotlivých cest. Postup opakujeme dle toho v úvodu kapitoly 6.4.2. Následně musíme tuto matici otestovat, zda neobsahuje hodnoty zpoždění překračující zadaný limit, postup na konci kapitoly 6.4.2.

Vzhledem k tomu, že se jedná o konečný návrh dle zpoždění musíme provést kontrolu toho, zda byly nalezeny cesty překračující limit zpoždění, v tuto chvíli se pravděpodobnost jejich výskytu značně snížila tím, že jsou v síti přidány nové cesty, a pokud byly nalezeny, musíme provést korekci návrhu následujícím způsobem:

1. V matici cest překračujících zpoždění hledáme jednotlivé indexy těchto hodnot (nenulové hodnoty).
2. Provedeme kontrolu matice indexů pomocí funkce isempty.
3. Pokud je matice indexů prázdná, přeskočíme proces následné optimalizace.
4. Pokud matice indexů není prázdná, provedeme následující.
   Určíme souřadnice x,y nenulových hodnot v matici zpožděných cest.
   Zjistíme počet souřadnic y, toto použijeme jednom z dalších kroků.
   Pomocí algoritmu graphshortestpath využívajícího Dijsktrova algoritmu, popis v kapitole 3.1.1, určíme všechny nejkratší cesty z výchozích souřadnic x.
   V dalším cyklu, jehož počet iterací odpovídá počtu nalezených souřadnic y, určujeme konkrétní cestu mezi souřadnicemi x a y a hodnotu zpoždění této cesty v jednotlivých úsecích.
   Určíme velikost matice jednotlivých dílčích zpoždění a na základě této velikost počítáme v cyklu rozdíl zpoždění jednotlivých úseků zpožděné cesty.
   Na základě vypočtených rozdílů jednotlivých úseků a nově zavedené proměnné přesah určujeme, zda jsou rozdíly mezi zpožděními jednotlivých úseků v rámci hodnot přesahu, např. do 5 %.
5. Pokud jsou rozdíly mezi úseky v rozsahu hodnoty přesahu, je výhodnější zvýšit kapacitu celé trasy. Kapacity jednotlivých úseků budou zvýšeny o percentuální podíl vzhledem k přesahu a jejich počtu.
6. Pokud jsou rozdíly větší než přesah, je výhodnější zvyšovat kapacitu o procenta přesahu v jednotlivých úsecích, které jsou vypsány.
7. Změnu kapacit provede uživatel změnou vstupních parametrů návrhu.

V případě námi zadaných parametrů, dle kap. 6 problém s překročením limitu zpoždění cest nenastal.

Výsledky návrhu s redundancí dle zpoždění jsou pak následující:



Obr. 6.5: Konečný návrh struktury sítě dle minimálního zpoždění

Tab. 6.15: Redundantní spoje návrhu dle minimálního zpoždění

Počátek	1	2	4	6
Konec	4	6	7	8
Délka [km]	2,2	0,8	1,5	1
Kapacita [Mbps]	500	250	150	100

Tab. 6.16: Spoje výsledného návrhu dle minimálního zpoždění

Počátek	1	1	1	2	3	3	3	4	5	6	7
Konec	2	3	4	6	4	5	6	7	7	8	8
Délka [km]	2	1,5	2,2	0,8	0,5	0,9	0,7	1,5	1,25	1	2,5
Zpoždění [ms]	0,058	0,056	0,059	0,100	0,051	0,053	0,052	0,168	0,127	0,245	0,133

Tab. 6.17: Parametry zpoždění všech cest návrhu s redundancí dle minimálního zpoždění

Limit zpoždění [ms]	10
Maximální zpoždění [ms]	6,024
Průměrné zpoždění [ms]	4,082

Tab. 6.18: Využití kapacit spojů a propustnost v návrhu dle minimálního zpožděné

Počátek	1	2	3	3	3	4	5	6	7
Konec	3	6	4	5	6	7	7	8	8
Délka [km]	1,5	0,8	0,5	0,9	0,7	1,5	1,25	1	2,5
Kapacita [Mbps]	500	250	500	500	500	150	200	100	200
Využitá kapacita [Mbps]	300	100	650	400	200	150	600	400	950

Tab. 6.19: Konečné náklady návrhu dle minimálního zpoždění

Náklady návrhu s redundancí [Kč]	1517350
Rozdíl proti min. kostře [Kč]	355500
Rozdíl proti vých. stavu [Kč]	- 161000
Procentuální úspora proti vých. stavu [%]	10

Pokud jde o porovnání nákladů celkových návrhů dle minimálního zpoždění, Tab. 6.19, a minimálních nákladů, Tab. 6.11, vidíme, že cena dle očekávání vzrostla a to o 40 % . Zajímavější je ale srovnání nárůstu nákladu v případě minimální kostry a návrhu s redundancí. V případě návrhu dle min. zpoždění nebyl nárůst tak dramatický, jednalo se o 30 %, v případě návrhu dle min. nákladů se jednalo o 65 %.

Z Tab. 6.13 a Tab. 6.17 vyčteme, že maximální a průměrná doba zpoždění po přidání redundantních spojů dle očekávání klesly. Nicméně vzhledem k už tak nízkým hodnotám zpoždění, je tento rozdíl minimální. V případě max. zpoždění jde o 1 % a průměrného zpoždění se jedná o hodnotu 6 %. V případě rozsáhlejších sítí, by tato hodnota dále klesla. Je to způsobeno hlavně tím, že návrh dle minimálních nákladů má dostatek přímých spojů. a proto se stírá výhoda vyšších kapacit některých spojů v návrhu dle minimálního zpoždění.

Dle očekávání došlo k navýšení využití kapacit spojů a propustnosti sítě, porovnání Tab. 6.18 a Tab. 6.10. Postup výpočtu a důvody, proč nejsou v tabulkách všechny spoje bude uveden v následující kapitole.

Celkově tedy můžeme takovýto způsob návrhu označit za plně funkční, a v některých případech nezbytný, ale značně neekonomický.

6.5 Optimalizace vzhledem k tokům v síti

Dalším z poměrně zásadních požadavků na provoz sítě je množství najednou přenesených dat, neboli propustnost sítě. V tomto případě se tedy snažíme dosáhnout maximálních toků v síti. Co to jsou toky v síti popisuje kapitole 2.2.10 a algoritmus pro jejich určení je popsán v kapitole 3.3.

V našem případě se zaměříme na transportní síť a snažíme se prostřednictvím progeamu zapsaného v souboru OptDleToku nejprve určit hodnoty toků v síti vzhledem k zadaným hodnotám kapacit jednotlivých spojů, zde se bude jednat, z důvodů které budou vysvětleny v kapitole 6.5.1, o cesty ke koncovým uzlům v síti. Dále budeme hledat spoj, který se v daných cestách nevyskytuje, čímž ho můžeme vyřadit a snížit náklady. V konečné fázi určíme poddimenzované toky a upravíme jejich kapacitu dle zadaného parametru.

6.5.1 Maximální toky v transportní síti dle zadaných parametrů kapacit

Jak je psáno v úvodu kap. 6.5, jako první se zaměříme na určení maximálních toků v transportní síti dle zadaných kapacit jednotlivých spojů v ní obsažených. Dle popisu v kapitole 2.2.10 je maximální tok vždy mezi dvěma uzly, z nichž je jeden zdroj a druhý spotřebič přenášené entity. Pro určení propustnosti sítě se jako nejvhodnější jeví hledání maximálních toků mezi všemi uzly současně. Jak se ale ukázalo, toto není technicky možné, neboť algoritmus zabudovaný v programu Matlab pro výpočet maximálních toků v síti,graphmaxflow, není schopen pracovat se zpětnými cestami, je schopen najít maximální tok mezi uzly 1 a 5, ale neurčí ho v opačném směru z uzlu 5 do 1. Nicméně se dá předpokládat, že obou případech bude maximální tok stejný.

Proto byl zvolen odlišný postup a to ten, že hledáme maximální tok k numericky poslednímu uzlu sítě, tedy z uzlu 1 do 8, 2 do 8, atd. Toto provádíme v jednom cyklu a hodnoty jednotlivých toků ve spojích daných cest k cíly sčítáme. Tím získáme přehled o tom, jakých maximálních toků dosáhneme ve spojích vzhledem jejich kapacitám, tedy toto platí pro spoje mezi dolní polovinou uzlů v síti, a jaké jsou možné maximální celkové toky a tedy i propustnost v síti vzhledem ke spojům mezi horní polovinou uzlů.

Před samotným postupem výpočtu a výsledky zde ještě jednou uvedeme kapacity jednotlivých spojů v transportní síti.

Tab. 6.20: Parametry spojů výchozího návrhu

Počátek	1	1	1	2	2	3	3	3	4	4	5	5	6	7
Konec	2	3	4	3	6	4	5	6	5	7	6	7	8	8
Délka [km]	2	1,5	2,2	1,4	0,8	0,5	0,9	0,7	1	1,5	1,1	1,25	1	2,5
Kapacita [Mbps]	500	500	500	250	250	500	500	500	100	150	100	200	100	200

Postup určení maximálních toků mezi danými uzly dle algoritmu zapsaného v OptDleToku je pak následující:

1. Vytvoříme si řídkou matici spojů ohodnocených jejich kapacitami dle Tab. 6.20.
2. Určíme si počet možných zdrojů v síti, což je vzhledem k výše popsanému problému se zpětnými cestami o jeden méně, než je celkový počat uzlů v síti.
3. Vytvoříme prázdnou matici budoucích toků ve spojích.
4. V cyklu určujeme pomocí algoritmu graphmaxflow hodnoty maximálních toků mezi zdrojovými uzly a spotřebičem.
5. Jednotlivé toky ve spojích tvořících cestu maximálního toku sčítáme do matice toků ve spojích vytvořenou v bodě 3.

Výsledné hodnoty jsou pak následující:

Tab. 6.21: Toky v jednotlivých spojích při výpočtu maximálních toků v síti

Počátek	1	1	2	2	3	3	3	4	4	5	5	6	7
Konec	3	4	3	6	4	5	6	5	7	6	7	8	8
Délka [km]	1,5	2,2	1,4	0,8	0,5	0,9	0,7	1	1,5	1,1	1,25	1	2,5
Kapacita [Mbps]	200	100	200	100	200	400	100	300	250	300	700	600	1150

Při porovnání Tab. 6.20 a Tab. 6.21 vidíme, jak úzká místa v transportní síti degradují využitelnou kapacitu jednotlivých spojů, především při srovnání dolní poloviny obou tabulek. Propustnost sítě, jenž můžeme očekávat, určíme vzhledem k hodnotě kapacity posledního spoje, skrze které jde veškerá komunikace.

Dále si můžeme všimnout, že v Tab. 6.21 chybí spoj mezi uzly 1 a 2. V jeho případě došlo k tomu, že nebyl využit v žádné z cest, a proto mohl být odstraněn. Určení tohoto spoje probíhá následovně:

1. Zjistíme maximální hodnotu v matici kapacity dle Tab. 6.21.
2. Zjistíme minimální hodnotu kapacity dle Tab. 6.20.
3. Podílem těchto dvou hodnot vynásobíme jednotlivé kapacity v Tab. 6.21 a tyto odečteme od patřičných hodnot v Tab. 6.20. Spoj, jenž se nezměnil se bude rovnat nule a tento můžeme odstranit.

6.5.2 Určení spojů s poddimenzovanou kapacitou a jejich úprava

Z hodnot kapacit spojů dle Tab. 6.21 vidíme, že značná část z nich není schopná přenést takové množství dat, na které jsou dle zadání dimenzovány. Toto je způsobeno úzkými místy v síti, tedy spoji s nízkou kapacitou.

Pro odstranění tohoto neduhu nejprve určíme poddimenzované spoje, což můžeme provést například tak, že mezi sebou odečteme hodnoty kapacit v tabulkách Tab. 6.21 a Tab. 6.20, čímž získáme kladné hodnoty u spojů s překročenou kapacitou. Dle dříve popsaného postupu a důvodů je jasné, že se bude jednat především o spoje mezi horní polovinou uzlů, ale toto našim dalším záměrů nijak nebrání.

Tab. 6.22: Poddimenzované spoje v síti a hodnota poddimenzování

Počátek	4	4	5	5	6	7
Konec	5	7	6	7	8	8
Délka [km]	1	1,5	1,1	1,25	1	2,5
Poddimenzované o [Mbps]	200	100	200	500	500	950

Následně zvýšíme jejich kapacitu vzhledem k onomu poddimenzování o zvolenou hodnotu koeficientu úpravy toků, tento si definujeme ve vstupních parametrech.

Postup je následující:

1. Testujeme matici poddimenzovaných spojů. Pokud nastane situace, že se nevyskytne žádný, např. pokud mají všechny spoje shodnou kapacitu, celou tuto proceduru přeskočíme.
2. Pokud matice poddimenzovaných spojů není prázdná, zvýšíme u jednotlivých spojů v ní obsažených jejich původní kapacitu o hodnotu danou zvoleným koeficientem vzhledem k poddimenzování. V našem případě jsme si určili hodnotu koeficientu na 10 % z hodnoty poddimenzování.
3. Do proměnná UT, která tento postup identifikuje, si uložíme hodnotu 1.
4. Provedeme nový výpočet maximálních toků v síti po úpravě kapacit spojů.

Výsledkem tohoto procesu je pak zvýšení kapacit poddimenzovaných spojů o tyto hodnoty:

Tab. 6.23: Hodnota kapacit spojů po úpravě

Počátek	1	1	2	2	3	3	3	4	4	5	5	6	7
Konec	3	4	3	6	4	5	6	5	7	6	7	8	8
Délka [km]	1,5	2,2	1,4	0,8	0,5	0,9	0,7	1	1,5	1,1	1,25	1	2,5
Kapacita [Mbps]	500	500	250	250	500	500	500	120	160	120	250	150	295

Nyní provedeme opětovné určení maximálních toků v síti dle stejného postupu jako v kap. 6.5.1, samozřejmě ale s novými hodnotami kapacit spojů. Výsledek je pak následující:

Tab. 6.24: Toky v jednotlivých spojích při výpočtu maximálních toků v síti

Počátek	1	1	2	2	3	3	3	4	4	5	5	6	7
Konec	3	4	3	6	4	5	6	5	7	6	7	8	8
Délka [km]	1,5	2,2	1,4	0,8	0,5	0,9	0,7	1	1,5	1,1	1,25	1	2,5
Kapacita [Mbps]	480	265	450	250	495	900	280	660	630	660	1570	1440	2695

Hodnoty v Tab. 6.23 ukazují, že provedení výše popsané úpravy kapacit spojů způsobilo značené zvýšení celkové propustnosti transportní sítě, nárůst je bezmála o 134 %, a zároveň je daleko lépe využita kapacita většiny spojů. Dále nám hodnota kapacity u spoje mezi uzly 3 a 5 říká, že tento uzel bude značně vytížen a v případě možných potíží se na něj bude nutno zaměřit a posílit jej.

Na co si v tomto případě musíme dát pozor je vhodná volba hodnoty koeficientu úpravy toků. Pokud bychom zvolily příliš velkou hodnotu, bude opětovně docházet k poddimenzování neupravených spojů a značným způsobem porostou náklady na návrh.

Spoje s takto pozměněnou kapacitou můžeme dále využít i v případě finálního návrhu., Jak je totiž z Tab. 6.22 vidět, jde o spoje s nízkými náklady, které nyní mohou v síti pomoci zvýšit propustnost a snížit zpoždění při udržení nízkých hodnot nákladů na návrh.

6.5.3 Náklady na návrh optimalizovaný dle toků

Jako poslední krok celé optimalizace musíme určit výsledné náklady na daný návrh. Jak v případě úpravy kapacit spojů, tak i bez provedení této úpravy musíme určit novou matici spojů dle délky, protože jsme odstranili spoj či spoje, a také musíme přepočítat celkovou kapacitu vstupující do jednotlivých uzlů z transportní sítě. V obou případech je postup shodný, liší se jen použitými hodnotami, volba proběhne na základě hodnoty proměnné UT.

Postupujeme takto:

1. Vypočítáme celkovou hodnotu kapacit vstupujících do uzlů z transportní sítě. Toto je provedeno v cyklu.
2. Vytvoříme matici spojů s ohodnocením délkou spojů a v ní najdeme spoj, který byl v návrhu odstraněn a tento odstraníme i zde.
3. V případě, že nedošlo k úpravě kapacit spojů, opakujeme postup z bodu 2 i pro matici kapacit. Pokud došlu k úpravě kapacit, použijeme nově spočtené hodnoty, mezi kterými odstraněný spoj nefiguruje.
4. Všechny výsledky z bodů 1, 2 a 3 musíme převést na sloupcové vektory bez nul.
5. Zavoláme příslušné funkce pro výpočet cen uzlů, spojů a celkové ceny návrhu.

Tab. 6.25: Celkové náklady návrhu dle maximálních toků v síti bez úpravy kapacit spojů

Celkové náklady návrhu [Kč]	1501350
Rozdíl proti vých. stavu [Kč]	- 177000
Procentuální úspora proti vých. stavu [%]	11

Spočtené celkové náklady jsou pak v případě úpravy kapacit spojů následující:

Tab. 6.26: Celkové náklady návrhu dle maximálních toků v síti

Celkové náklady návrhu [Kč]	1587100
Rozdíl proti vých. stavu [Kč]	- 91250
Procentuální úspora proti vých. stavu [%]	5,44

Tab. 6.25 a Tab. 6.26 obsahují hodnoty celkových nákladů tohoto typu návrhu a vcelku jasně demonstrují, že úspora je v tomto případě nepatrná. Toto je daň, která musí být zaplacena za maximalizaci velikosti přenášených dat v síti. Úspory by mohly být výší v případě rozsáhlejších sítí, ale nikdy se nebudou moci rovnat těm, kterých bylo dosaženo v kapitolách 6.3 a 6.4.

6.6 Určení spolehlivosti spojů

Poslední operací, kterou budeme provádět nad výchozím návrhem bude určení spolehlivosti jednotlivých spojů v transportní síti a nalezení nejspolehlivějších cest mezi jednotlivými uzly této sítě. Tento proces nebude mít formu optimalizace jako v předchozích případech, ale bude spíše informativního charakteru.

Ano, v úvodu kap. 6 je uvedeno, že jedním z požadavků na návrh sítě je spolehlivost spojů $P_{zmax} = 1*10^{-4}$, ale tato hodnota je ovlivněna především referenční hodnotou, od které se odvíjí určení spolehlivost jednotlivých spojů. V našem případě se referenční hodnota vztahuje k 1 Gbps spoji a jemu určené spolehlivosti, která činí $1*10^{-5}$.

V takovémto případě je nutno pro dodržení hodnoty maximální chybovosti spoje použít alespoň 1 Mbps spoj.

Pro určování spolehlivosti spojů a hledání nejspolehlivějších cest je využito postupu pro hledání těchto typů cest popsaného v kapitole 3.1.

6.6.1 Určení procentuální spolehlivosti jednotlivých spojů

První fází programu pro určení spolehlivosti spojů a hledání nejspolehlivějších cest zapsaného v souboru OptDleChybovosti je určení procentuální spolehlivosti jednotlivých spojů.

Jak je uvedeno výše, vycházíme z předpokladu, že chybovost spoje o kapacitě 1 Gbps je $1*10^{-5}$. Postup pro určení chybovosti a z ní vycházející spolehlivosti spojů o jiné kapacitě je následující:

1. Vypočítáme si procentuální podíl chybovosti jednotlivých spojů vůči hodnotě $1*10^{-5}$.
2. Výpočet z bodu 1 musíme převést tak, aby odpovídal vyšší či nižší chybovosti oproti referenční hodnotě.
3. Výsledek z bodu 2 vynásobíme referenční hodnotou chybovosti.
4. Odečteme výsledek bodu 3 od maximální hodnoty pravděpodobnosti, tedy 1.
5. Vynásobíme výsledek bodu 4 hodnotou sto, čímž získáme procentuální vyjádření spolehlivosti jednotlivých spojů.

Zjištěná procentuální spolehlivost spojů je následující:

Tab. 6.27: Procentuální spolehlivost jednotlivých spojů, hodnoty od 3. desetinného místa

Počátek	1	1	1	2	2	3	3	3	4	4	5	5	6	7
Konec	2	3	4	3	6	4	5	6	5	7	6	7	8	8
Spolehlivost [%]	25	25	25	125	125	25	25	25	05	075	05	1	05	1

Dle hodnot v Tab. 6.27 můžeme říci, že všechny spoje mají spolehlivost vyšší než 99,99 %, což odpovídá vstupním požadavkům.

Toto byla spolehlivost jednotlivých spojů, nyní budeme hledat nejspolehlivější cesty.

6.6.2 Hledání nejspolehlivějších cest

Po určení spolehlivosti jednotlivých spojů můžeme pokračovat v postupu hledání nejspolehlivějších cest popsaného v kap. 3.1.

Konkrétní postup je následující:

1. Hodnoty spolehlivosti jednotlivých spojů, před vyjádřením v procentech, musíme logaritmovat logaritmem o základu 10.
2. Výsledky bodu 1 vyjádříme v absolutní hodnotě.
3. Vytvoříme řídkou matici spojů ohodnocených hodnotami bodu 2 a tuto převedeme na dolní trojúhelníkovou matici.
4. Pomocí dvou cyklů budeme hledat všechny nejspolehlivější cesty od jednotlivých uzlů ke všem ostatním, ano jedná se o využití algoritmu hledání nejkratší cesty pomocí algoritmu graphshortestpath. Výsledkem je výpis mezilehlých uzlů, které tvoří jednotlivé cesty.
5. Použijeme dolní trojúhelníkovou matici z bodu 3 jako vstupní parametr funkce pro hledání všech nejkratších, v našem případě nejspolehlivějších, cest v síti, tedy algoritmu graphallshortestpaths.
6. Výsledek bodu 5 převedeme na horní trojúhelníkovou matici, postačují nám hodnoty jednoho směru, opačný směr je totožný. Tímto získáme pravděpodobnost, s jakou může nastat v nalezených nejspolehlivějších cestách z bodu 4 chyba při přenosu dat.
7. Konečný výsledek bodu 6 vynásobíme hodnotou sto, čím získáme procentuální vyjádření dané pravděpodobnosti.

Pro ilustraci uvedeme výpis několika nalezených nejspolehlivějších cest mezi některými uzly v síti:

Tab. 6.28: Možná podoba nejspolehlivějších cest v síti

Počátek cesty	1	2	3
Konec cesty	8	7	7
Mezilehlé uzly	1; 3; 6; 8	2; 1; 4; 7	3; 5; 7

Na závěr uvedeme procentuální vyjádření pravděpodobnosti, s jakou nastane na jednotlivých nejspolehlivějších cestách v síti chyba při přenosu dat:

Tab. 6.29: Pravděpodobnost chyby při přenosu nejspolehlivější cestou v procentech

Počátek cesty	Konec cesty	Pravděpodobnost chyby [%]
1	2	0,003257330765707
1	3	0,002171526698136
2	3	0,003800242979709
1	4	0,003257330765707
2	4	0,006514661531414
3	4	0,003257330765707
1	5	0,005428857463843
2	5	0,007057573745416
3	5	0,003257330765707
4	5	0,004125993565877
1	6	0,005428857463843
2	6	0,003800242979709
3	6	0,003257330765707
4	6	0,006514661531414
5	6	0,004125993565877
1	7	0,007274740531381
2	7	0,010532071297088
3	7	0,007166157002657
4	7	0,004017409765674
5	7	0,003908826236950
6	7	0,008034819802827
1	8	0,009554851029720
2	8	0,007926236545586
3	8	0,007383324331584
4	8	0,007926236002624
5	8	0,007817652473900
6	8	0,004125993565877
7	8	0,003908826236950

Tímto bychom uzavřeli fázi dílčích návrhů a postupů a přejdeme na návrh celkový.

6.7 Vytvoření konečného návrhu vzhledem k optimalizacím

Již od počátku naše snažení směřoval k jediném cíli, tím je vytvoření kompletního návrhu sítě vzhledem ke všem vstupním požadavků definovaným na počátku kap. 6 při minimalizaci nutných nákladů. K tomuto samozřejmě využijeme dílčích výsledků předchozích operací.

Program pro vytvoření kompletního návrhu je zapsán v souboru VyslednyNavrh. Tento soubor si v dalším textu popíšeme a objasníme si úskalí a nedostatky tohoto typu návrhu a návrhové procedury.

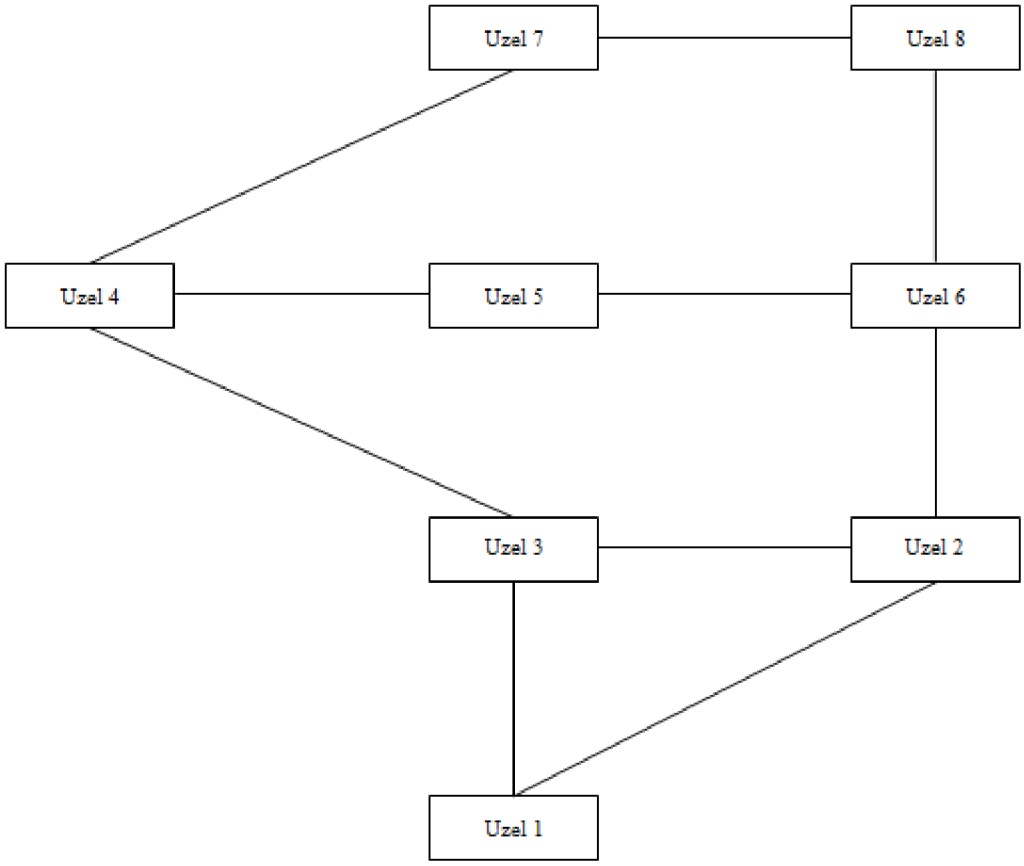
Postup výsledného návrhu je tedy následující:

1. Úvod je standardní a to nápověda a ošetřená chybných vstupů.
2. Ve druhém kroku voláme jednotlivé, dříve vytvořené, programy pro načtení jejich výstupních hodnot.
3. Nyní musíme vytvořit základní kostru výsledného návrhu. Protože se snažíme o minimalizaci nákladů., hledáme ve výstupních maticích spojů návrhů dle minimální ceny, zpoždění a maximálních toků shodu. Konkrétně se tedy jedná o shodu především s maticí spojů dle minimálních nákladů. V případě, že nalezneme shodný počet spojů jak s maticí spojů dle min. zpoždění, tak i s maticí spojů dle maximálních toků, upřednostňujeme ty dle minimálního zpoždění, což nám zaručí jak minimální zpoždění v síti, tak i dostatečnou propustnost sítě.
4. Vytvoříme rozdílovou matici spojů požitých pro základní návrh a výchozí maticí spojů dle minimálních nákladů.
5. Protože při určení základní kostry výsledného návrhu mohla nastat situace, že ne všechny uzly v síti budou alespoň stupně dva, v krajním případě je to nula, musíme provést kontrolu jednotlivých uzlů a případné chybějící spoje doplnit. Ony chybějící spoje použijeme z rozdílové matice vytvořené v bodě 5, pro minimalizaci nákladů. Konkrétní postup kontroly a přidání spojů je popsán v kapitole 6.3.2.

Tímto jsme vytvořili topologii výsledného návrhu, jejíž parametry budeme optimalizovat. V našem konkrétní případě byly použity následující spoje:

Tab. 6.30: Spoje konečného návrhu sítě

Počátek	1	1	2	2	3	4	4	5	6	7
Konec	2	3	3	6	4	5	7	6	8	8
Délka [km]	2	1,5	1,4	0,8	0,5	1	1,5	1,1	1	2,5
Kapacita [Mbps]	500	500	250	250	500	100	150	100	100	200



Obr. 6.6: Topologie konečného návrhu sítě

Tab. 6.30 ukazuje, že v našem případě se kompromisní spoje shodují s těmi, které byly využity v návrhu s redundancí dle minimálních nákladů. Toto se dalo dle nastavených pravidel očekávat, chceme minimalizovat náklady Tím pádem se můžeme odkázat na kapitolu 6.3, kde jsou popsány parametry tohoto návrhu.

Jak jsem ale psal výše, v síti se nachází mnoho úzkých míst, viz Tab. 6.22, jejichž vhodnou úpravou, která je popsána v kapitole 6.5.2, můžeme docílit poměrně zajímavých výsledků ve výsledném návrhu.

Déle se tedy budeme zabývat tímto alternativním řešením.

6.7.1 Toky v síti výsledného návrhu při úpravě kapacit spojů

Prvním ze sledovaných parametrů je hodnota kapacit jednotlivých spojů v tomto návrhu a také hodnoty maximálních toků v této síti což nám dále pomůže v určení propustnosti této sítě. Zároveň jsou toto důležité hodnoty z hlediska zpoždění.

Použijeme následující postup:

1. Z, v úvodní fázi, načtené matice všech spojů sítě ohodnocených uživatelem zadanými kapacitami vybereme ty spoje, které jsou použity v topologii konečného návrhu. Jejich ohodnocení dále změníme podle hodnot vypočítaných v kapitole 6.5.2 . Pokud pro některý spoj nemáme upravenou hodnotu kapacity, použijeme původní hodnotu.
2. Vytvoříme řídkou matici spojů z těch, které byly vybrány v bodě 1. Toto jsou hodnoty kapacit jednotlivých spojů v konečném návrhu při úpravě kapacit spojů.
3. Dle stejného postupu jako v kapitole 6.5.1 určíme hodnoty maximálních toků v síti tohoto konečného návrhu.

Výsledné hodnoty toků a kapacit jsou následující:

Tab. 6.31: Hodnota kapacit spojů konečného návrhu při úpravě kapacit spojů

Počátek	1	1	2	2	3	4	4	5	6	7
Konec	2	3	3	6	4	5	7	6	8	8
Délka [km]	2	1,5	1,4	0,8	0,5	1	1,5	1,1	1	2,5
Kapacita [Mbps]	500	500	250	250	500	120	160	120	150	295

Tab. 6.32: Toky ve spojích při výpočtu maximálních toků v síti konečného návrhu, ú. t.

Počátek	1	1	2	2	3	4	4	5	6	7
Konec	2	3	3	6	4	5	7	6	8	8
Délka [km]	2	1,5	1,4	0,8	0,5	1	1,5	1,1	1	2,5
Kapacita [Mbps]	30	280	160	180	720	360	640	480	810	935

Tab. 6.32 jasně ukazuje, jak se celková propustnost sítě snižuje při sníženém počtu spojů, v porovnání s hodnotami v Tab. 6.24. S tímto ale musíme při snižování nutných nákladů počítat.

Dále tyto údaje porovnáme s těmi v Tab. 6.10. Zde je patrný nárůst propustnosti a lepší využití kapacit jednotlivých spojů v síti, což je samozřejmě žádaný efekt.

Dále nám hodnota kapacity u spoje mezi uzly 3 a 4 říká, že tento uzel bude značně vytížen a v případě možných potíží se na něj bude nutno zaměřit a posílit jej.

6.7.2 Zpoždění v síti výsledného návrhu při úpravě kapacit spojů

Vzhledem ke změně kapacit spojů musíme provést nové výpočty zpoždění jednotlivých spojů.

Postup tohoto výpočtu se shoduje s tím, který je popsán v kapitole 6.4.2. Výsledná zpoždění jednotlivých spojů a cest v síti jsou následující:

Tab. 6.33: Zpoždění jednotlivých spojů ve výsledném návrhu při změně kapacit spojů

Počátek	1	1	2	2	3	4	4	5	6	7
Konec	2	3	3	6	4	5	7	6	8	8
Délka [km]	2	1,5	1,4	0,8	0,5	1	1,5	1,1	1	2,5
Zpoždění [ms]	0,058	0,056	0,103	0,100	0,051	0,205	2,958	0,206	0,165	0,094

Tab. 6.34: Parametry zpoždění všech cest konečného návrhu dle minimálního zpoždění

Limit zpoždění [ms]	10
Maximální zpoždění [ms]	5,995
Průměrné zpoždění [ms]	4,281

Hodnoty zpoždění po změně kapacit některých spojů opět porovnáme s těmi, které jsme zjistili vzhledem k původním kapacitám, tedy s těm v Tab. 6.9, ale také s těmi, které jsem zjistili po optimalizaci návrhu pro minimalizaci zpoždění, tedy s Tab. 6.17. Z tohoto srovnání vidíme, že po úpravě kapacit spojů se zpoždění snížilo, a to jak průměrné, tak hlavně maximální.

6.7.3 Spolehlivost spojů výsledného návrhu při úpravě kapacit spojů

Spolehlivost jednotlivých spojů je stejně jako zpoždění ovlivněno změnou kapacity spojů.

Stejně jako v případě zpoždění musíme při změně kapacit provést nové výpočty, což je případ i našeho návrhu. Postup těchto výpočtů je stejný jako v kapitolách 6.6.1 a 6.6.2.

Výsledky jsou v případě našeho návrhu následující:

Tab. 6.35: Procentuální spolehlivost jednotlivých spojů k. n., hodnoty od 3. desetinného místa

Počátek	1	1	2	2	3	4	4	5	6	7
Konec	2	3	3	6	4	5	7	6	8	8
Spolehlivost [%]	25	25	25	125	125	06	08	06	075	15

Hodnoty Tab. 6.35 ukazují, že je splněna podmínka maximální ztrátovosti v konečném návrhu. Následuje tabulka nejspolehlivějších cest v daném návrhu.

Tab. 6.36: Pravděpodobnost chyby při přenosu nejspolehlivější cestou v k. n. v procentech

Počátek cesty	Konec cesty	Pravděpodobnost chyby [%]
1	2	0, 003257330765707
1	3	0, 003257330765707
2	3	0, 003800242979709
1	4	0, 006514661531414
2	4	0, 007057573745416
3	4	0, 003257330765707
1	5	0, 010597221544633
2	5	0, 007882802992928
3	5	0, 007339890778926
4	5	0, 004082560013219
1	6	0, 007057573745416
2	6	0, 003800242979709
3	6	0, 007600485959419
4	6	0, 008165120026438
5	6	0, 004082560013219
1	7	0, 010510354569621
2	7	0, 011053266783624
3	7	0, 007253023803914
4	7	0, 003995693038207
5	7	0, 008078253051426
6	7	0, 007719928045985
1	8	0, 011074983511090
2	8	0, 007817652745383
3	8	0, 010955542084225
4	8	0, 007698211318518
5	8	0, 008099969778893
6	8	0, 004017409765674
7	8	0, 003702518280311

Po změně kapacit některých spojů došlo také ke zvýšení spolehlivosti cest v konečném návrhu, srovnání Tab. 6.29 a Tab. 6.36.

6.7.4 Náklady konečného návrhu

Závěrečná fáze návrhu výsledné sítě spočívá ve výpočtu celkových nákladů, které jsou zapotřebí na její vybudování.

V našem případě musíme do výpočtu nákladů zahrnout změnu kapacity spojů v síti. Jednotlivé postupy výpočtů cen uzlů a spojů jsou popsány v kapitolách 6.2.1 a 6.2.2.

Konečné náklady jsou následující:

Tab. 6.37: Celkové náklady konečného návrhu

Celkové náklady uzlů [Kč]	279750
Celkové náklady spojů [Kč]	866800
Celkové náklady [Kč]	1146550
Rozdíl proti výchozímu stavu [Kč]	-531800
Procentuální úspora nákladů proti výcho. stavu [%]	32

Hodnoty nákladů popsané v Tab. 6.37 ukazují, že výsledná úspora je, i po úpravě kapacit spojů, proti výchozímu stavu 32 %. Úspory návrhu bez úpravy kapacit spojů, viz. Tab. 6.11, byly 35% a můžeme tedy říci, že nedošlo k nijak dramatickému nárůstu nákladů. Docílili jsme tedy až třetinové úspory nákladů při splnění všech požadovaných parametrů. Zde je vidět jasný příklad toho, že tento proces má z pohledu úspory nákladů svůj smysl.

6.8 Testy výsledného návrhu

Po vytvoření konečného návrhu sítě je zapotřebí jej otestovat. Je několik možností jak test provést. Jednou z nich vytvoření modelu sítě v simulačním programu typu OPNET Modeler, který nám jakožto komplexní nástroj, umožní nasimulovat různé provozy v síti, výpadky spojů a uzlů nebo využití různých směrovacích protokolů. Toto je ale značně komplikované a vzhledem k poměrně statickým parametrům našeho návrhu až zbytečné.

Testy budeme provádět opět v programu Matlab. Budeme navozovat situace, kdy odstraníme spoj ze sítě, což je možné chápat jako jeho výpadek, a následně zjistíme parametry provozu v této síti. Dále budeme snižovat kapacity některých spojů, čímž se dá simulovat jejich přetížení, a poté opět určit parametry spojení.

V testu se zaměříme na efektivnější návrh se změněnými hodnotami kapacit některých spojů.

6.8.1 Simulace výpadku spoje

V případě tohoto testu, který simuluje poruchu spoje, z návrhu vyčleníme značně vytížený spoj mezi uzly 3 a 4.

Výpadkem spoje mezi uzly 3 a 4 dojde k tomu, že komunikace mezi uzly 1,2 a 3 a zbytkem uzlů v síti probíhá pouze skrze spoj z uzlu 2 do uzlu 6, jak to ukazuje Obr. 6.6, který není pro takovou zátěž dostatečně dimenzován a tím pádem je celková propustnost sítě značně snížena.

Výsledkem je samozřejmě také to, že zpoždění v síti v důsledku výpadku daného spoje vzroste a v případě některých cest může dojít k překročení limitu zpoždění, což se v našem případě nestalo. I na tomto je možno demonstrovat úspěšnost návrhu.

Tab. 6.38: Parametry zpoždění při testu výpadku spoje mezi uzly 3 a 4

Limit zpoždění [ms]	10
Maximální zpoždění [ms]	7,662
Průměrné zpoždění [ms]	4,812

V případě hodnot Tab. 6.38 je nevhodnější náprava rovnoměrné zvýšení kapacity celé trasy.

Důsledkem výpadku spoje v síti je také snížení spolehlivosti jednotlivých cest v síti, ale to je vzhledem k předchozím závěrům pochopitelné.

6.8.2 Simulace snížení kapacity některých spojů

V tomto testu, který simuluje přetížení spoje, snížíme kapacitu vybraných spojů a následně znovu zjišťujeme parametry sítě. Konkrétně snížíme na polovinu kapacitu spoje mezi uzly 3 a 4 a o čtvrtinu snížíme kapacitu spoje mezi uzly 2 a 6.

Tab. 6.39: Kapacity vybraných spojů pro potřeby testu jejich přetížení

Počátek	2	3
Konec	6	4
Původní kapacita [Mbps]	250	500
Snížená kapacita [Mbps]	187,5	250

Opět jsou sníženy kapacity spojů, které zajišťují komunikaci mezi uzly 1,2 a 3 a zbytkem uzlů.

Toto snížení kapacit vybraných spojů samozřejmě vede ke snížení propustnosti sítě a také ke snížení využití kapacit jednotlivých spojů. Toto demonstruje následující tabulka:

Tab. 6.40: Toky ve spojích při výpočtu maximálních toků v síti při testu přetížení

Počátek	1	1	2	2	3	4	4	5	6	7
Konec	2	3	3	6	4	5	7	6	8	8
Délka [km]	2	1,5	1,4	0,8	0,5	1	1,5	1,1	1	2,5
Kapacita [Mbps]	150	160	160	260	570	240	610	360	810	905

Hodnoty v Tab. 6.40 můžeme porovnat s těmi v Tab. 6.32. Vidíme, že došlo ke značnému navýšení provozu přes spoj z uzlu 2 do uzlu 6, na což tento není dostatečně dimenzován. K tomuto došlo tím, že byla snížena kapacita spoje mezi uzly 3 a 4, a proto algoritmus výpočtu maximálních toků rozhodl o tom, že bude provoz takto přesměrován.

Pokud jde o zpoždění v síti, tak zde není situace tak dramatická.

Tab. 6.41: Parametry zpoždění při testu výpadku přetížení spojů

Limit zpoždění [ms]	10
Maximální zpoždění [ms]	6,075
Průměrné zpoždění [ms]	4,313

Jak je z obou simulací patrné, větší vliv má výpadek nebo omezení spoje na hodnoty toků v síti než na zpoždění. V obou případech je ovšem nutný operativní zákrok, který bude podnícen včasným zjištěním poruchy v dohledovém centru.

6.9 Střet s podmínkami reálného návrhu

I přes to, že prezentované výsledky práce svědčí o tom, že využití teorie grafů pro návrh sítí je relevantní, práce samotná musela být z pochopitelných důvodů v mnoha ohledech zjednodušena

Proto by bylo dobré uvést, jaké parametry je třeba do reálného návrhu doplnit a s jakými úskalími se můžeme setkat. Tyto informace pak také poslouží k tomu, abychom definovali další rozvoj této návrhové metody.

1. Průzkum využití poskytovaných služeb a s tím souvisejících nároků na kapacitu spojů. V případě této práce uživatel kapacitu spojů definuje ve vstupních parametrech.
2. Možnosti umístění přepojovacích uzlů a dalšího vybavení. V našem případě máme toto umístění dáno pevně a od něj odvíjíme délku jednotlivých spojů.
3. Průzkum trasy, kterou povedeme kabeláž. S tímto souvisí možné úskalí vlastnických sporů, kdy možné komplikace s pronájmem tras a jeho cenou prodraží celkový návrh. Také může díky tomuto dojít k značným časovým prodlevám.
4. Určitým způsobem je v návrhu nutné počítat s budoucím rozvojem sítě a možností jejího rozšíření.
5. Náklady na infrastrukturu zahrnují krom nákladů na spoje a uzly, kdy v případě této práce uživatel definuje cenu za jeden metr spoje, bit kapacity, obsloužení příchozího požadavku (přepojení paketu) a cenu za jeden paměťový blok přepojovacího uzlu, také náklady na podpůrná zařízení jakou jsou opakovače a zesilovače signálu, záložní zdroje napájení, dohledová centra, atd. v případě spojů bude dále nutno rozlišovat jejich cenu dle trasy, kterou budou vedené.
6. Provoz v síti je třeba monitorovat a vyhodnocovat tudíž další náklady vznikají při těchto činnostech. Mezi parametry, které v síti monitorujeme, patří zpoždění, ztrátovost a dodržení QoS.

Na základě tohoto výpisu je patrné, že další směrování vývoje této návrhové metody by mělo vést k její větší variabilitě, která by dokázala pokrýt co největší procento výše popsaných parametrů.

Toto samozřejmě překračuje možný rámec této práce a v mnoha případech bude potřeba využít poznatků z jiných matematických a vědních oborů, ale pro nasazení v reálném prostředí budou tyto kroky potřebné.

Jedním z prostředků, které by bylo možno využít pro podporu této návrhové metody je využití genetických algoritmů, kterou mohou pomoci s výběrem nejvhodnějšího umístění přepojovacích prvků a další infrastruktury. O jejich využití píše následující kapitola 6.10.

6.10 Využití genetických algoritmů při návrhu struktury sítě

Následující text bude vycházet z [10], kde je možno získat o této problematice další informace.

Pro nalezení ideálního řešení návrhu sítě je zapotřebí brát v potaz mnoho faktorů, jak jsem uvedl výše, a jejich vhodná kombinace mnohdy překračuje lidské možnosti. Proto se společnosti snaží vyvinout software, který bude toto rozhodování provádět automaticky a nalezne nejvhodnější řešení.

Firma British Telecom vyvinula v 90, letech 20. století systém pro návrh metalických vedení nazvaný GenOSys. Tento systém využívá genetických algoritmů pro optimální nebo alespoň optimálnímu řešení nejbližší návrh sítě. V podání této společnosti se jedná výstavbu telekomunikačních sítí, a proto se výsledek ne zcela ztotožňuje s potřebami datových sítí, ale princip tohoto systému je možno uplatnit i v našem případě na datové sítě.

Vstupními parametry tohoto systému jsou buď stromový graf, popis v kapitole 2.2.8, nebo plně propojená síť, která reprezentuje všechna místa, kam je možno umístit technické vybavení sítě a kde se nachází koneční uživatelé. Systém pak hledá na základě těchto dat optimální řešení, které bude splňovat ekonomické a funkční předpoklady.

Systém GenOSys je kombinací aplikace genetických algoritmů, což je jádro systému, teorie grafů a teorie pravděpodobnosti.

Genetické algoritmy jsou použity pro samotný rozhodovací proces. Využívají pro svou činnost operací křížení a mutace.

Teorie grafů je zde pro výpočty nejkratších cest ve zadané síti a generování struktury grafu.

Teorie pravděpodobnosti je využito pro omezení prostoru, na kterém probíhají výpočty.

Celek pak tvoří hybridní model vyhledávání s výbornými výsledky. Model tohoto systému si můžeme představit jako hybridní systém, který je rozdělen na část využívající genetických algoritmů pro snížení časové náročnosti výpočtu a na část, ve které své výpočty provádí teorie grafů a pravděpodobnosti, které jsou využity v části genetických algoritmů. Jde tady o paralelní metodu hledání.

Systém GenOSys může pracovat ve třech režimech:

1. manuálním - vyžaduje časé zásahy operátora pro upřesnění některých údajů.
2. poloautomatickém - Omezuje nutnost vnějšího zásahu na minimum.
3. automatickém - provede kompletní návrh bez nutnosti zásahu operátora. Je snahou maximálně využít tento režim.

Hlavní výhody z použití tohoto systému jsou následující:

1. Jednotný systém návrhu dle stanovených pravidel.
2. Nižší nároky na tvůrce sítí.
3. Kratší čas potřebný pro návrh sítě, snížení nákladů na návrh a možnost rychlé změny návrhu.

Poslední vývoj směruje k využití tohoto systému i v jiných typech sítí a pokrytí až 1000 uzlů.

7 ZÁVĚR

Tato práce je rozdělena na dva větší tematické celky.

Tím prvním je teoretická část, která obsahuje popis jednotlivých pojmů užívaných v teorii grafů a pak také algoritmů, určených pro řešení úloh, kterými se teorie grafů zabývá. Toto jsou kapitoly 2 a 3. Následuje kapitola 4, která obsahuje popis dnes používaných směrovacích protokolů OSPF a RIP, které pro svou činnost využívají algoritmů popsaných v části 3.1. Závěrečná teoretická část se pak zabývá teorií front. Toto sice přímo nesouvisí s teorií grafů, ale je to poměrně důležité pro správné fungování přenosů dat v datových sítích.

Druhým tematickým celkem, kapitola 6, této práce je praktická ukázka využití, dříve popsané, teorie grafů pro návrh a optimalizaci struktury datové sítě. Konkrétně se jedná o soubor programů zapsaných v m-file souborech. Pro jejich tvorbu jsem využil programu Matlab a jeho vestavěných komponent. Cílem bylo vytvoření kompromisního návrhu, který by splňoval zadané podmínky při minimalizaci nutných nákladů Tato ukázka byla provedena na autorem smyšlené síti.

Nejprve je v kapitole 6 uveden postup, jakým bude daná praktická část koncipována a jsou zde vypsány jednotlivé požadavky, které má výsledný návrh sítě splňovat. Dále pak je zde popsán postup, jakým způsobem uživatel zadává obecné vstupní parametry, které jsou nutné pro činnost jednotlivých programů. V případě některých programů jsou potřeba i další vstupní proměnné, které jsou ovšem popsány v konkrétních kapitolách, které tyto programy a postupy popisují.

Na základě obecných parametrů, které uživatel zadal je možné provést výpočet celkových nákladů, pokud by došlo k realizaci tohoto návrhu bez úprav a je vygenerován graf, reflektující danou topologii sítě. Hodnota nákladů je vyčíslena v Tab. 6.3 a topologie je vyobrazena na Obr. 6.1.

První provedenou optimalizací, kapitola 6.3, je hrubé omezení nákladů. Postup je popsán v kapitole 6.3. Nejprve provedeme návrh pomocí minimální kostry grafu, spoje jsou ohodnoceny cenou, což je celkově nejméně nákladný návrh, náklady uvedeny v Tab. 6.5. Vzhledem ke zjevným nedostatkům takového návrhu, žádná redundance, nízká propustnost, vyšší zpoždění při přenosu atd. doplníme návrh minimální kostry a vhodné redundantní spoje. Náklady takového řešení jsou uvedeny v Tab. 6.11 a jeho topologie je vyobrazena na Obr. 6.3. Zároveň jsou v Tab. 6.9 a Tab. 6.10 výsledky měření zpoždění a propustnosti takovéhoto návrhu.

Stejným postupem je provedena další optimalizace, která spočívá v návrhu sítě s minimálním zpožděním, kapitola 6.4. Zde jsou spoje ohodnoceny svým zpožděním, které je dáno přenosem signálu médiem a serializačním zpožděním na vstupu do krajních uzlů spoje.

Následně musíme určit nejrychlejší cesty v síti, vzhledem k použitým spojům, a k nim přičíst hodnoty serializačních zpoždění přístupových linek uzlů a zpoždění dané zpracováním informací v jednotlivých mezilehlých uzlech. Toto provádíme zvlášť pro minimální kostru takovéhoto návrhu, výsledky v Tab. 6.13 a Tab. 6.14, a pro návrh s redundancí, Tab. 6.16, Tab. 6.17, Tab. 6.18 a Tab. 6.19.

Poslední dílčí optimalizací je maximalizace toků v dané síti.

Zde bohužel není možno určovat toky ve zpětných cestách, ale pouze dopředných, a proto jsme jako spotřebič v síti určili uzel 8. Hodnoty propustnosti pak určujeme ve spojích do tohoto uzlu. Výsledky v Tab. 6.21. V tomto případě se jedná o nejnákladnější návrh, což je dáno použitím většiny spojů, Tab. 6.25.

Zároveň jsme ale v síti identifikovali úzká místa, která brzdí provoz. Proto je alternativní výstup tohoto procesu návrh, který upravuje kapacitu těchto poddimenzovaných spojů, Tab. 6.23.

Jako doplněk je vytvořen program, který určuje spolehlivost jednotlivých spojů v síti. Tato spolehlivost je ovšem ovlivněna určením výchozí ztrátovosti, v našem případě pro 1 Gbps spoj je to $P_{Zmax} = 1*10^{-5}$, a tím pádem je pro dodržení požadavků na síť nutné použít spoje alespoň o kapacitě 1 Mbps. Hodnoty spolehlivosti spojů a cestou v Tab. 6.27 respektive Tab. 6.29.

Všechny tyto dílčí operace pracují v mezích, které zadal uživatel jako vstupní parametry, v případě nemožnosti je splnit, hledají možná řešení.

Výsledkem těchto dílčích návrhů je pak celkový návrh, který má splňovat všechny zadané podmínky a být co nejméně nákladný. Porovnáním spojů jednotlivých návrhů byla nakonec vytvořena struktura, která odpovídá návrhu dle minimálních nákladů, Obr. 6.6. Jak jsme ale zjistili dříve, v síti se nachází mnoho úzkých míst, které nepříznivě ovlivňují parametry výsledného návrhu, a proto jako optimální řešení požijeme v konečném návrhu upravené kapacity spojů, které zvýší propustnost sítě, sníží zpoždění a nezpůsobí neadekvátní nárůst celkových nákladů. Výsledky celkového návrhu jsou zaznamenány v Tab. 6.31 až Tab. 6.37.

Výsledný návrh testujeme na situaci, kdy dojde k výpadku při přetížení spojů. Z obou testů je patrné, že daný stav se nejvíce projeví na propustnosti sítě než na zpoždění v něm. Dílčí výsledky testů jsou v Tab. 6.38 až Tab. 6.41.

Předposlední kapitola srovnává podmínky a úskalí reálného procesu návrhu sítě s tím, jak je postupováno v této práci a dává určitá doporučení, kam by mě směrovat další vývoj této návrhové metody.

Poslední kapitola představuje jednu z možností budoucího směrování, a to použití genetických algoritmů

Co se týče závěrečných doporučení, kam by se měl směrovat další vývoj této metody, tak je to zcela určitě větší variabilita, díky které by bylo možno pokrýt co největší procento možností, jak síť navrhnout pro poskytování vybraných služeb s co nejvyšší efektivitou.

Jednou z možností jsou právě genetické algoritmy, které by na základě vstupních hodnot rozhodovaly samostatně o návrhu celé sítě, včetně vhodného rozmístění infrastruktury.

Dále je pak možnost, že by systém na základě informací o využiti služeb v rámci daného přepojovacího uzlu navrhnul propojení v četně kapacit potřebných spojů, zde by se ale muselo definovat jejich možné umístění a umístění infrastruktury, což by bylo možné například v případě sítí ve městech, kde lze tato místa předem poměrně snadno definovat. Nejlepší by samozřejmě byla varianta kombinace obou těchto postupů.

LITERATURA

[1] BALEJ, J. *Simulace zpoždění při přenosu dat mezi stanicemi v IP sítích.* VUT Brno, Semestrální projekt 2009.

[2] BRÁZDOVÁ, M. *Využití některých metod teorie grafů při řešení dopravních problémů.* [online]. 2007 http://pernerscontacts.upce.cz/05_2007/Brazdova.pdf

[3] DVOŘÁK, Jiří. *Systémy hromadné obsluhy*[online]. Brno : UAI VUT Brno. www.uai.fme.vutbr.cz/~jdvorak/vyuka/tsoa/PredO11.ppt

[4] FRONCEK, D. *Úvod do teorie grafu*. Skriptum Slezská univerzita Opava, ISBN 80-7248-044-8, CR, 1999

[5] GRYGÁREK, Jiří. Směrovací protokol OSPF [online]. Ostrava : VSB http://www.cs.vsb.cz/grygarek/SPS/

[6] *Hamiltonovský graf* [online] http://cs.wikipedia.org/wiki/Hamiltonovsk%C3%BD_graf

[7] HENDICK, Charles. RFC1058 - Routing Information Protocol [online]. New Jersey : Rutgers University

[8] HLINĚNÝ, Petr. *Teorie grafů* [online]. Brno : Fakulta informatiky MU. http://www.fi.muni.cz/~hlineny/Vyuka/GT/Grafy-text09.pdf

[9] Chang-Ju, L. *Johnson's algorithm* [online]. www.caveshadow.com/.../Chang-Ju%20Lin%20-%20Johnson's%20Algorithm.ppt

[10] Poon, K., Conway, A., Wardeop, G., Mellis, J. *Successful application of genetic algorithms to network design and planning*. BT Technol J Vol 18 No 4 October 2000.

[11] MAREŠ, Martin. *Krajinou grafových algoritmů.* ITI, 2007. 72 s. ISBN 978-80-239-9049-2

[12] MOY, John. *OSPF - Anatomy of an Internet Routing Protocol*. New Jersey : Pearson Education Corporate, 2000. 348 s. Sales Division. ISBN 0-201-63472-4.

[13] REINHARD, Diestel. *Graph theory* [online]. New York : Springer-Verlag, Heidelberg 2005 http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/

[14] SLANINA, František. *Teorie sítí : společný jazyk buňky a internetu* [online]. Praha : Fyzikální ústav AV ČR http://www.otevrena-veda.cz/ov/users/Image/default/C1Kurzy/NH2006pdf/6.pdf

[15] SLOVÁK, Jan. *Drsná matematika III — 11. přednáška - Toky v sítích* [online]. Fakulta informatiky MU Brno http://is.muni.cz/el/1433/podzim2006/MB103/um/mIII11.pdf?fakulta=1433;obdobi=3523;kod=MB103

[16] ŠEDA, Miloš. *Teorie grafů* [online]. Brno : UAI, FSI VUT Brno. http://www.uai.fme.vutbr.cz/~mseda/TG03_MS.pdf

[17] WHITE,R., SLICE,D., RETANA,A. Optimal Routing Design. Cisco Press, ISBN 1-58705-187-7, USA, 2005

[18] ŽIŽKA, Miroslav. *Teorie front* [online]. http://quercus.kin.tul.cz/~miroslav.zizka/multiedu/Teorie_front_1.pdf

PŘÍLOHY

Příloha A - Obsah CD

Složka mFiles obsahuje následující soubory:

1. CelCenaSpoje.m
2. CelkovaBWdoUzlu.m
3. CenaLinky.m
4. CenaZaUzel.m
5. OptDleChybovosti.m
6. OptDleCeny.m
7. OptDleToku.m
8. OptDleZpozdeni.m
9. Vstupy.m
10. VyslednyNavrh.m

Složka Práce obsahuje elektronickou verzi práce ve formátu .pdf.

Příloha B - Spuštění programů z mfile souborů

Spuštění programů je nutno provádět v programu Matlab s patřičnými toolboxy (Bioinformatics a další). Pro přístup k souborům je nutné nastavit v pracovním prostředí programu cestu k těmto souborům. To provedeme změnou cesty v okně Current Directory, zde také máme přístup k jednotlivým souborům.

Jako první je nutné definovat vstupy. Ty jsou zaznamenány ve Vstupy.m, tento stačí otevřít, běžný dvojklik myší na soubor v Current Directory, a stisknout F5. V tomto souborů můžeme vstupy také dle potřeby upravovat.

Dále je nutné pro chod dalších programů spustit CelkovaBWdoUzlu.m. Pomocí příkazu help a názvu souboru, zapsaného do Command Window, otevřeme nápovědu k programu. Zkopírujeme inicializační řádek **BWcel = CelkovaBWdoUzlu(pocatek, konec, BW, pocetuzlu)** a vložíme opět do Command Window, potvrdíme stiskem klávesy Enter. Tento řádek tedy definuje výstupní proměnnou, název souboru a vstupní parametry, výsledkem je výpočet a zobrazení patřičných hodnot.

Stejným způsobem poté můžeme spouštět všechny ostatní programy, tedy především jednotlivé optimalizace a celkový návrh. Pokud chceme mezi jednotlivými výpočty vymazat pracovní okno, použijeme příkaz clc.