UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

VOĽBA ŠÉFA V SIEŤACH S CHYBNÝMI LINKAMI

Bc. Ondrej Pašuth

2011

Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Voľba šéfa v sieťach s chybnými linkami

Bc. Ondrej Pašuth

diplomová práca

9.2.1 Informatika

Evidenčné číslo: 869032ab-2a95-457e-a86e-21fb38f127cb

Vedúci: doc. RNDr. Rastislav Královič PhD.

Bratislava, 2011

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Ondrej Pašuth
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Voľba šéfa v sieťach s chybnými linkami.

Cieľ: Navrhnúť algoritmus voľby šéfa v simple (resp. fractional) threshold modeli. Dosiaľ bol publikovaný výsledok o voľbe šéfa na kruhu. Cieľom práce je rozšíriť výsledky na iné topológie.

Vedúci: doc. RNDr. Rastislav Královič, PhD.
Katedra: FMFI.KI - Katedra informatiky

Dátum zadania: 02.02.2011

Dátum schválenia: 02.02.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
Vedúci

Dátum potvrdenia finálnej verzie práce, súhlas s jej odovzdaním (vrátane spôsobu sprístupnenia)

.....
Vedúci

Poďakovanie

Chcel by som na tomto mieste vyjadriť úprimnú vďaku vedúcemu mojej diplomovej práce doc. RNDr. Rastislavovi Královičovi PhD. za jeho ochotu, námahu a trpezlivosť, s ktorou sa mi venoval a vysvetlil mi náležitosti týkajúce sa mojej diplomovej práce. Tiež mu chcem poďakovať za výber zaujímavej témy, za zabezpečenie študijných materiálov a za to, že si na mňa vždy našiel čas. Na tomto mieste by som tiež chcel vyjadriť úprimnú vďaku všetkým vyučujúcim, ktorých som mal možnosť stretnúť akýmkoľvek spôsobom počas rokov svojich štúdií na Matfyze.

Abstrakt

Autor: Bc. Ondrej Pašuth
Názov diplomovej práce: Voľba šéfa v sieťach s chybnými linkami
Škola: Univerzita Komenského v Bratislave
Fakulta: Fakulta matematiky, fyziky a informatiky
Katedra: Katedra informatiky
Vedúci bakalárskej práce: doc. RNDr. Rastislav Královič PhD.

Vo svojej diplomovej práci som sa venoval modelovaniu dynamických chýb v synchrónnych sieťach (vrcholy rozposielajú správy len v čase tiknutia synchronizačných hodín) a tiež riešením niektorých problémov vo vytvorenom modeli. Zameral som sa najmä na broadcasting a voľbu šéfa, pričom som nadviazal na niektoré publikované výsledky a vytvoril algoritmus pre voľbu šéfa v ľubovoľnom hranovo k-súvislom grafe. Toto bol hlavný cieľ mojej diplomovej práce, pričom ani nebolo vopred jasné, či sa takýto algoritmus dá skonštruovať. Okrem detailnej analýzy tohto algoritmu som v práci tiež načrtol niektoré základné postupy a techniky pri tvorbe algoritmov pre niektoré problémy. Vytvorený algoritmus by sa dal použiť aj v reálnom svete, je však veľmi prísny na počty strácaných správ, pričom v reálnych sieťach je počet strát nižší.

KĽÚČOVÉ SLOVÁ: dynamické chyby, jednoduchý prahový model, voľba šéfa, synchronizácia, vlákno

Abstract

Author: Bc. Ondrej Pašuth
Caption: Leader election in the networks with faults
University: Comenius University in Bratislava
Faculty: Faculty of Mathematics, Physics and Informatics
Department: Department of Computer Science
Supervisor: doc. RNDr. Rastislav Královič PhD.

I was dealing with modeling of dynamic faults in synchronous networks (vertices send their messages only in the time of tick of synchronizing clocks) and also with solving of some problems in this model in my master thesis. I focused on broadcasting and leader election and I modified some published conclusions and created new algorithm for leader election in arbitrary k-connected graph. This was the main aim of my master thesis and even it was not clear if it is possible to do. Besides detailed analysis of this algorithm I mentioned some basic procedures and technics for creating algorithms for some problems in my thesis. The created algorithm could be used in real world, but it is very strict in the number of lost messages and there are less losses in the real networks.

KEY WORDS: dynamic faults, simple threshold model, leader election, synchronization, thread

Obsah

Úvod

Vo svojej diplomovej práci sa venujem modelovaniu dynamických chýb v synchronizovaných sieťach (vrcholy rozposielajú správy len v čase tiknutia synchronizačných hodín) a riešeniu problému voľby šéfa vo vytvorenom modeli. Modelovaniu dynamických chýb (môžu sa vyskytnúť kdekoľvek v sieti) sa venovalo a venuje množstvo prác, pričom výskumníci sa snažili namodelovať teoretický model čo najviac vystihujúci realitu, v ktorom sa dajú zrealizovať a dokázať aj isté netriviálne problémy.

Nadviazal som najmä na predchádzajúcu prácu vedeckého tímu, ktorého súčasťou bol aj môj školiteľ doc. RNDr. Rastislav Královič PhD [DKKS, DKP]. V spolupráci s ďalšími kolegami z vedeckej obce vytvoril abstraktný *jednoduchý prahový model*, ktorý modeluje straty správ v reálnych sieťach. Tento model je dosť prísny a takmer negarantuje hornú hranicu počtu stratených správ v jednom synchronizačnom tiku (istú garanciu však musí dávať, pretože v prípade nulovej garancie sa nedajú riešiť žiadne z uvažovaných problémov).

Vďaka uisteniu odoslania aspoň jednej správy v jednom synchronizačnom tiku v prípade, že v hranovo k-súvislom grafe reprezentujúcom danú sieť je v tomto tiku odoslaných aspoň k správ však vieme v tomto modeli vytvoriť algoritmy pre rozličné "všeobecné" problémy, ktoré sa rozoberajú a sú riešené v sieťach. Tieto problémy rozoberáme v teoretickej rovine, ich riešenia sa však môžu aplikovať aj v reálnych sieťach. Cieľom tejto diplomovej práce je najmä ponúknuť riešenia niektorých problémov.

V prvej fáze svojej práce som sa snažil hlbšie vniknúť do práce tohto tímu a do spôsobu vytvárania algoritmov pre daný model. V publikovaných článkoch sa títo akademickí a vedeckí pracovníci zamýšľajú najmä nad broadcastingom v sieti pre rozličné topológie (kruh, kompletný graf o n vrcholoch, hranovo

k-spojitý graf vo všeobecnosti) a zaoberajú sa aj voľbou šéfa v niektorých topológiách. S použitím niektorých ich výsledkov som sa v druhej fáze svojej práce snažil zamyslieť nad algoritmom voľby šéfa najmä pre hranovo k-súvislý graf. Toto bol hlavný cieľ mojej diplomovej práce, pričom vopred ani nebolo jasné, či sa takýto algoritmus dá skonštruovať. V prípade neúspechu bolo mojím cieľom zamerať sa na jednoduchšie topológie ako napr. torus alebo aj iné "umelé" topológie.

Postupnými malými krokmi som však v spolupráci so svojím školiteľom prišiel na výsledky, ktoré prezentujem v ďalších častiach tejto diplomovej práce. Vymyslených algoritmov nie je veľa, svojou komplexnosťou a riešením vznikajúcich, najmä "technických", problémov mi však dali zabrať. Na ďalších stranách sa snažím jasne prezentovať vymyslené postupy a dokázať ich korektnosť a časovú zložitosť.

Kapitola 1

Základné pojmy a definície

1.1 Distribuovaný systém s chybami

V bežných počítačových sieťach sa chyby vyskytujú bežne. Posielané packety sa častokrát pre rôzne dôvody nedoručia ku svojmu cieľu a ich nedoručenie adresátovi sa musí riešiť. V ľubovoľnom distribuovanom počítačovom systéme, kde sa jednotlivé uzly tohto systému dorozumievajú na základe správ, existuje istý predpoklad straty správ. Takáto strata je síce nepríjemná, ale nemôže byť neočakávaná. Či už môže nastať výpadok linky alebo uzla, či ďalšie problémy, strata správy musí byť obsiahnutá v scenári rôznych situácií a ich riešení.

Možný výskyt chýb v distribuovanom počítačovom systéme častokrát znemožňuje priamočiare riešenie požadovaného problému, niekedy sa riešená úloha v takomto systéme ani nedá vykonať. Napr. v *asynchrónnych sieťach*, v ktorých nemáme garantovaný čas chodu správy medzi dvoma susednými uzlami (nevieme zaručiť, v akom časovom okamihu príde správa ku svojmu adresátovi v prípade, že sa nestratila a raz bude doručená), už iba samotná možnosť straty správy na niektorých lokalizovaných linkách znamená nemožnosť riešenia takmer žiadnych netriviálnych úloh a problémov [FLP85]. Aj preto sa vo výskumnej i praktickej oblasti dáva dôraz a využíva sa najmä *synchrónny prenos správ*. To je taký, pri ktorom v sieti akoby existovali hodiny, ktoré tikajú. Na každý jeden tik môžu vrcholy v tomto systéme odoslať niektorým svojim susedom správu, ktorá sa doručí v nejakom "mikromomente". Do ďalšieho tiku hodín majú všetky vrcholy dostatok času na spracovanie získaných správ a vytvorenie nových, ktoré opäť

odošlú v ďalšom tiku hodín. Takúto synchrónnu štruktúru budeme používať aj v celej tejto práci.

Definícia 1.1.1. Synchrónna sieť (systém) je taká, v ktorej v jednom tiku abstraktných hodín dokážu všetci príjemcovia správy spracovať, vypočítať a v prípade potreby aj distribuovať novú informáciu (ktorá bude doručená adresátom v ďalšom tiku hodín).

Aj keď v reálnom živote nie je výskyt a množstvo stratených správ nijako obmedzovaný (niekedy sa doručia všetky správy, pri výpadku siete sa nedoručí ani jedna), pri snahe vytvoriť model reálneho sveta musíme zaručiť isté obmedzenia na stratu správ, v opačnom prípade nemôžeme garantovať prakticky nič (v najhorších prípadoch by sa totiž mohli strácať všetky správy dosť často). Ak budú tieto obmedzenia príliš silné, napriek dobrým "teoretickým" výsledkom nemôžeme garantovať podobnosť s reálnym svetom. Naopak, ak by boli obmedzenia príliš slabé, pri silnej podobnosti s reálnym svetom by sme nemohli zaručiť takmer žiadne relevantné výsledky. Dôraz sa preto dáva na isté rozumné obmedzenia (nejaké modely si ukážeme v ďalšej časti).

Treba už vopred objasniť, že táto práca sa bude zaoberať "deterministickým" modelovaním a riešením problémov. Existujú totiž aj pravdepodobnostné modely, pričom pravdepodobnosť chyby na niektorej linke je rovná nejakému $p < 1$. Hlavným problémom v takomto pravdepodobnostom modeli je vytvoriť algoritmus pre broadcasting s čo najlepším pomerom "časová zložitosť - pravdepodobnosť úspechu" [BDP97].

Chyby v sieťach môžu byť *statického*, ale aj *dynamického* charakteru. V prípade *statických (lokalizovaných) chýb* je už vopred dané, na ktorých linkách sa chyby môžu vyskytnúť, ostatné prepojenia medzi vrcholmi majú garanciu stopercentného prechodu správ [AGHK96]. Pri *dynamických chybách* nevieme dopredu určiť, kde sa vyskytne strata správy, pretože tá sa môže udiať na hociktorej hrane siete. Dynamické chyby, ktoré budeme používať v ďalšom pokračovaní tejto diplomovej práce, lepšie modelujú reálny svet sietí, kde linka môže vypadnúť prakticky medzi ľubovoľnými dvoma uzlami.

Definícia 1.1.2. Statické chyby sú lokalizované na určité linky v sieti, dynamické chyby sa môžu vyskytnúť kdekoľvek v uvažovanej sieti.

Z vyššieuvedeného môžeme zadefinovať pojem, ktorým sa budeme zaoberať v ďalších kapitolách tejto práce.

Definícia 1.1.3. Distribuovaný systém s chybami je distribuovaná synchrónna sieť s dynamickými chybami reprezentovaná grafom, pričom uzly siete sú vrcholy grafu a linky medzi uzlami sú hrany tohto grafu.

Hlavnými algoritmami, ktoré sa riešia v súvislosti s distribuovaným systémom s chybami, sú broadcasting a voľba šéfa. Prvá úloha je rozdistribuovať z jedného vrchola informáciu do celej siete. Druhý problém je vybrať z iniciátorov (vrcholy zobudené na začiatku alebo počas behu algoritmu) jedného, pričom on aj ostatné vrcholy si budú vedomí tejto jeho nadradenosti (väčšinou sa vyberá na základe ID jednotlivých vrcholov - šéfom sa stane iniciátor s najväčším ID).

Navrhnuté algoritmy musia úspešne skončiť pre ľubovoľný povolený výskyt chýb v priebehu vykonávania tej-ktorej úlohy v danom distribuovanom systéme s chybami. To znamená, že pri hocijakom strácaní správ v priebehu behu algoritmu, ktoré neporušuje nastolené obmedzenia, sa musí dosiahnuť požadované splnenie úlohy (v opačnom prípade je navrhnutý algoritmus zlý). Následne sa skúma časová zložitosť navrhnutých algoritmov, t. j. počet tikov hodín, ktorý potrebuje algoritmus na svoje úspešné zbehnutie v najhoršom možnom prípade. Keďže uvažujeme o najhoršom prípade, môžeme si celú situáciu predstaviť tak, že v sieti sa nachádza nejaký *inteligentný adversary (nepriateľ)*, ktorého snahou je bojovať proti nám a sťažovať nám plnenie predpísanej úlohy.

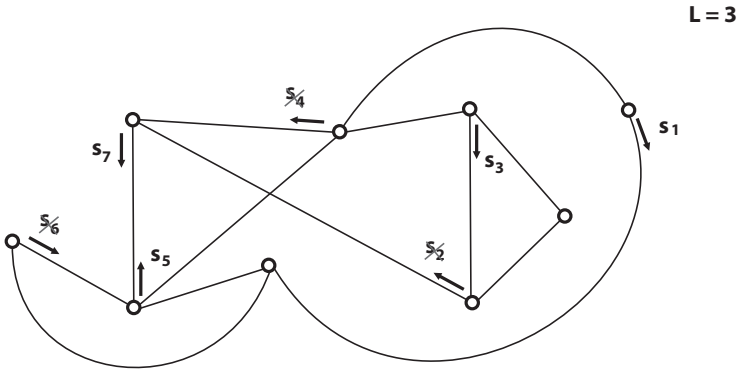
1.2 Modely distribuovaného systému s chybami

Na začiatok si pre istotu zapíšeme definíciu pojmu, ktorý budeme v ďalšej časti dosť využívať.

Definícia 1.2.1. Hranová súvislosť $c(G)$ grafu G udáva minimálny počet hrán, ktoré musíme odobrať z G, aby sa stal nesúvislým alebo 1-vrcholovým grafom (t.j. ak ľubovoľným odobratím menej ako k hrán nevznikne nesúvislý alebo 1-vrcholový graf, G je k-súvislý).

Prvá veľká skupina modelov distribuovaného systému s chybami sú tzv. *kumulatívne modely*. Tie dávajú nasledovné obmedzenie pre počet stratených správ v jednom tiku hodín: stratí sa maximálne L správ. V prípade problému broadcastingu (danému problému sa venuje článok [Dob04]) je za predpokladu $L < c(G)$, kde $c(G)$ je hranová súvisloť grafu reprezentujúceho daný distribuovaný systém

s chybami, riešenie veľmi jednoduché - stačí postupne počas určitého počtu krokov "zahlcovať" sieť správami obsahujúcimi pretláčanú informáciu.



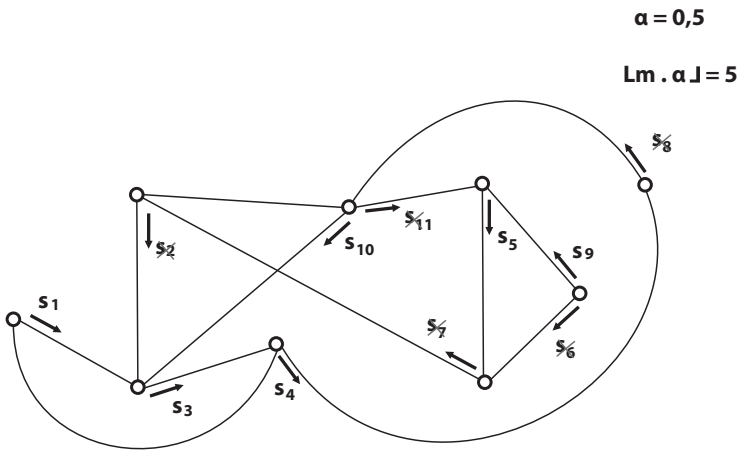
Obr. 1.1: Strata správ v prípade kumulatívneho modelu

Výhodou takýchto modelov je to, že sú "L-tolerantné", to znamená odolné voči $0, \ldots, L$ chybám za jednu časovú jednotku. Ich veľkou nevýhodou je však fakt, že neodrážajú skutočnosť reálnych sietí, v ktorých počet stratených správ dosť výrazne závisí od celkového počtu odoslaných správ v tom-ktorom časovom okamihu. Teoretické riešenia, v ktorých sa do siete rozposiela veľký počet správ, potom príliš neodrážajú skutočnosť a v reálnych sieťach častokrát nefungujú.

Narozdiel od kumulatívnych modelov sa v tzv. *podielových modeloch* (viac o nich v [KKR03]) počet stratených správ odvíja od celkového počtu odoslaných v nejakom tiku hodín. Počet chýb v distribuovanej sieti je v takomto modeli zhora ohraničený konštantným násobkom počtu odoslaných správ. Teda ak m_t je počet odoslaných správ v časovom okamihu t, potom sa v danom okamihu stratí najviac $\lfloor \alpha m_t \rfloor$ správ, pričom α je konštanta $0 \leq \alpha < 1$.

Výhodou takéhoto *podielového modelu* je to, že navrhnuté riešenia musia počítať so stratou v jednom časovom okamihu až do výšky daného konštantného násobku počtu odoslaných správ. Ich veľkou nevýhodou však je, že sa dajú "oklamať". Ak totiž bola v čase t odoslaná v celom grafe (sieti) iba jediná správa, tá sa s určitosťou aj doručí. Takýto model potom vedie opäť k určitým anomáliám medzi teóriou a reálnou situáciou v sieťach.

Ukázali sme dva možné modely distribuovanej siete s chybami, pričom sme opísali aj ich nevýhody a prečo celkom nevyhovujú reálnej situácii. Na jednej strane stálo prílišné zahlcovanie siete, kým na druhej bola možnosť "pretlačiť" informáciu vďaka malému počtu odoslaných správ - oba prípady sú extrémami, ktorých



Obr. 1.2: Strata správ v prípade podielového modelu

by sme sa radi zbavili, ďalší navrhnutý model by však zároveň mohol zahrňovať predchádzajúce modely ako svoj špeciálny prípad. Z predstavených prípadov si však môžeme zobrať dôležité ponaučenie - niekedy bude potrebné "nadbytočné (redundantné)" zahlcovanie siete správami, úplné bezmyšlienkovité posielanie správ všetkým vrcholom však tiež nie je vždy cestou k želanému k riešeniu.

1.3 Jednoduchý prahový model

Spojením výhod dvoch vyššieuvedených modelov dostaneme *podielový model s prahom* [DKKS]. V ďalších riadkoch si ho detailnejšie objasníme.

Definícia 1.3.1. Funkcia závislosti $F(m_t)$ popisuje maximálny možný počet stratených správ v tom-ktorom časovom okamihu v závislosti od počtu odoslaných správ m_t.

Funkcia $F(m_t)$ teda vymedzuje maximum, koľko správ môže *inteligentný nepriateľ* zahodiť v jednom tiku hodín v prípade distribuovanej siete s chybami. Samozrejme, ak mu to vyhovuje, nezahodí z posielaných správ ani jednu, prípadne iba jednu. V prípade kumulatívneho modelu sa $F(m_t) =$ konštante. V prípade podielového $F(m_t) = \lfloor \alpha m_t \rfloor$.

Definícia 1.3.2. Funkcia závislosti pre podielový model s prahom je

$$F(m_t) = max\{T - 1, \lfloor \alpha m_t \rfloor\},$$

pričom $T \leq c(G)$ je konštanta menšia alebo rovná hranovej súvislosti grafu a α je konštanta $0 \leq \alpha < 1$.

Čo hovorí definícia 1.3.2 podielového modelu s prahom? Majme pevne určené konštanty $T \leq c(G)$ a α, $0 \leq \alpha < 1$. Potom počas jedného časového okamihu môže nepriateľ odobrať v prípade odoslania menej ako T správ všetky tieto správy. Ak sa v grafe odošle aspoň T správ, nepriateľ môže zrušiť maximálne $\lfloor \alpha m_t \rfloor$ z nich.

V prípade $\alpha = 0$ dostávame *kumulatívny* model, pretože v definícii 1.3.2 nám zmizne časť závisiaca od počtu odoslaných správ. Ak nastavíme $T = 1$, dostávame *podielový* model, keďže maximálny počet stratených správ bude závisieť iba od celkového počtu odoslaných správ. V novom modeli sme teda splnili požiadavku na "obsiahnutie" predchádzajúcich dvoch predstavených modelov. Nastavením konštánt T a α dostávame širokú škálu ďalších "medziprípadov".

Dajme nepriateľovi maximálnu možnú silu, t. j. $T = c(G)$ a $\alpha = \lim_{\epsilon \to 0} (1 - \epsilon)$. Takto navrhnutý model nazývame *jednoduchý prahový model* a budeme s ním pracovať v celej tejto práci. Hovorí, že na doručenie aspoň jednej správy musí byť v jednom časovom okamihu odoslaných aspoň $c(G)$ správ, v opačnom prípade môže nepriateľ všetky odoslané správy zrušiť. Zároveň však má nepriateľ pri odoslaní dostatočného počtu správ algoritmom takú silu, že je povinný nechať doručiť svojmu adresátovi iba jedinú správu v sieti (môže, samozrejme, aj viac v prípade, že mu to bude "hrať do karát").

Definícia 1.3.3. Funkcia závislosti pre jednoduchý prahový model:

$$F(m_t) = \{c(G), \lfloor \alpha m_t \rfloor\},$$

pričom $c(G)$ je hranová súvislosť grafu G a $\alpha = \lim_{\epsilon \to 0} (1 - \epsilon)$.

Kapitola 2

Zaujímavé výsledky

V nasledujúcej kapitole uvádzam niektoré zaujímavé výsledky (publikované v [DKKS, DKP]) pre niektoré problémy a niektoré topológie v jednoduchom prahovom modeli. Ich preštudovaním a pochopením môže čitateľ získať lepší vhľad do tejto problematiky a tiež pochopiť princípy používané pri riešení problémov v danom modeli. V niektorých prípadoch podávam iba "suché výsledky" a niekedy sa viac pristavím pri samotnom algoritme alebo jeho zaujímavej časti. Táto kapitola však nechce byť nejakou dokazovacou, kde by sa mala ukázať korektnosť a tiež časová zložitosť jednotlivých algoritmov. Slúži ako základ pre ďalšie uvažovanie vo svete jednoduchého prahového modelu.

2.1 Broadcasting

2.1.1 Kruh

Kruh je hranovo 2-spojitý graf, takže v našom modeli $T = c(G) = 2$. Keď chceme doručiť aspoň jednu správu, musíme v danom časovom okamihu odoslať aspoň 2 správy. Algoritmus pre túto topológiu je veľmi zaujímavý z toho dôvodu, že jeho rozšírením a zovšeobecnením bol vytvorený algoritmus na pretláčanie správ v ľubovoľnom hranovo k-spojitom grafe. Nebudeme sa zaoberať prípadmi s alebo bez znalosti n, t. j. znalosti dĺžky kruhu, predstavíme si iba hlavný algoritmus a jeho myšlienku.

Popis algoritmu pre broadcasting na kruhu

Na začiatku máme v kruhu jeden informovaný vrchol, ktorý tento kruh predeľuje na *pravú* a *ľavú časť* (oni sa v kruhu niekde zbiehajú, nezáleží nám kde - čisto teoreticky to môže byť aj v samotnom informovanom vrchole, t.j. pravá alebo ľavá časť sa nachádza iba v danom vrchole). Každý vrchol nachádzajúci sa v jednej alebo druhej časti môže byť v stave *spiaci*, *aktívny* alebo *pasívny*. *Spiaci* vrchol je doteraz neinformovaný vrchol, *aktívny* vrchol už dostal z pravej alebo ľavej (nie však z oboch) strany rozposielanú informáciu, kým *pasívny* vrchol už dostal informáciu a z druhej strany potvrdenie o jej prijatí. Informácia sa na základe algoritmu 2.1.1 distribuuje do celej topológie, pričom sa postupne počas behu algoritmu "spasívňujú" jednotlivé vrcholy.

Ak bol vrchol informovaný z pravej strany, snaží sa pretláčať informáciu smerom ďalej, pričom ku zdroju svojej informácie (susedovi, od ktorého dostal danú správu) posiela naspäť potvrdenie informácie. Ak sa totiž takéto potvrdenie naozaj doručí svojmu adresátovi, ten už vie o informovanosti susedného vrchola a nemusí mu posielať ďalšie správy - spasívnie. Spasívňovanie je veľmi dôležité (inak by nepriateľ teoreticky mohol posielať správy stále po tej istej hrane, prípadne by mohli vzniknúť aj ďalšie problémy) a zaručuje progres v sieti. Postupne teda spasívňujeme všetky vrcholy. Treba však ešte poznamenať, že pasívnosť neznamená úplnú nečinnosť. Pasívny vrchol u totiž môže dostávať správy od svojho suseda s, ktorý netuší, že v minulosti už bola vrcholu u doručená jeho správa. Pasívny vrchol teda naďalej "odpovedá" na prijaté správy, nesnaží sa však informovať svojho suseda na druhej strane (keďže od neho už dostal potvrdenie o informovanosti).

Pre potreby nasledujúceho algoritmu 2.1.1 ešte dodám, že začiatočný informovaný vrchol je akoby rozdelený na dva "podvrcholy", pričom jeden sa nachádza na "pravej strane kruhu", zatiaľ čo druhý leží na "ľavej strane". Jeden teda posiela správy do jednej časti, druhý do druhej (navzájom si už správy neposielajú).

Algoritmus 2.1.1. *Pretláčací algoritmus pre broadcasting na kruhu:*

1. *Každý aktívny vrchol pošle informáciu svojmu potenciálne neinformovanému susedovi (od ktorého ešte nedostal "potvrdenie").*
2. *Každý aktívny vrchol z pravej strany (informáciu dostal od svojho ľavého suseda) pošle informáciu svojmu potenciálne neinformovanému susedovi.*

Každý vrchol z ľavej strany, ktorý dostal v kroku (1) správu, odpovie na ňu poslaním potvrdenia.

3. *Rovnako ako v časti (2), len pravé a ľavé strany sa navzájom vymenia.*

4. *Každý vrchol, ktorý prijal informačnú správu (nie potvrďenie)v krokoch (1)-(3), odpovie na ňu poslaním potvrdenia.*

Jedno trvanie pretláčacieho algoritmu 2.1.1 sa volá *fáza* celého programu pre broadcasting na kruhu. V jednej takejto fáze sa doručí aspoň jedno potvrdenie:

- v kroku (1) sa doručí aspoň jedna informačná správa
- v kroku (2) alebo (3) sa v prípade nedoručenia potvrdenia doručí informačná správa na druhej strane kruhu ako tomu bolo v kroku (1)
- v kroku (4) sa už posielajú iba potvrdenia, pričom tie putujú z oboch strán (sú minimálne dve), teda aspoň jedno sa doručí

Vďaka doručovaniu potvrdení, spasívňovaniu jednotlivých vrcholov a ďalšiemu šíreniu informačných správ vieme po $(n-1)$ fázach zaručiť, že o informácii iniciačného vrchola vie už každý jeden vrchol v danom kruhu. Na základe toho môžeme vysloviť nasledujúcu vetu.

Veta 2.1.1. *V jednoduchom prahovom modeli existuje algoritmus pre broadcasting v kruhu, ktorý pracuje v časovej zložitosti* $4\,(n-1)$.

2.1.2 Kompletné grafy

V kompletných grafoch je každý vrchol spojený s hociktorým iným v danom grafe, teda môžeme povedať, že graf je hranovo $(n-1)$- spojitý. V každom časovom okamihu teda potrebujeme poslať aspoň $n-1$ správ, v opačnom prípade môže nepriateľ odstrániť všetky odosielané správy.

V prípade kompletného grafu v jednoduchom prahovom modeli nie je jedno, či poznáme topológiu siete (každý vrchol vie, ktorý port vedie ku ktorému vrcholu) alebo každý vrchol pozná iba svoj lokálny svet (vie odlíšiť svoje porty, to je všetko). V prípade znalosti topológie existuje algoritmus pre broadcasting, ktorého časová zložitosť je $O\left(n^2\right)$. Tento algoritmus však pre ďalšiu časť tejto

práce nie je ničím extra zaujímavý, preto ho tu neuvádzam. Prípad, v ktorom vrcholy nepoznajú celkovú topológiu siete, je zaujímavejší. Pozrieme sa na niektoré jeho zaujímavé aspekty.

Známa a neznáma oblasť

Podobný koncept "spoznávania nového sveta (novej oblasti)" používam aj v ďalších častiach pri prezentovaní svojich výsledkov, preto je veľmi vhodné sa s ním čiastočne oboznámiť už tu. Žiadny z vrcholov nevie na začiatku rozoznať svojich susedov, iba svoje porty (vie si ich napr. očíslovať, nepozná však informáciu, ku komu vedú). Ako však putuje v určitej forme predstavenej v ďalšej časti informácia do nových a nových vrcholov, spolu s ňou putuje v správach aj informácia o *už známom svete* vo forme množiny trojíc (u, p, v). Táto trojica znamená to, že z vrcholu u sa dá prejsť do susedného vrchola v cez port p. Pri posielaní správy neznámemu vrcholu pošle vrchol u v správe okrem iného aj informáciu obsahujúcu svoje meno (napr. ID) a číslo portu, cez ktorý správa prechádza. Prijímateľ tejto nie úplnej informácie k nej pridá svoj identifikačný údaj a vznikne nová informácia, teda *známy svet* sa rozšíri. Pri každom prechode správy sa zväčšuje informácia toho-ktorého vrchola (pretože so správou prichádza aj informácia o doterajšom známom svete, prípadne aj samotný prechod správy cez doteraz neznámy port môže znamenať rozšírenie tejto znalosti).

Objaviteľské a pomocné správy

Nasledujúci koncept je veľmi zaujímavý, preto ho uvádzam v tejto práci. Ide o to, že informácia postupne prechádza do nových a nových vrcholov, zväčšuje sa známy svet a počet neinformovaných vrcholov klesá. V každom časovom okamihu (tiku hodín) však musí byť odoslaných aspoň $n - 1$ správ, inak sa nemusí doručiť ani jedna z nich. Predstavme si situáciu, že vrchol u zo známeho sveta chce posunúť informáciu niekomu z "neznámeho sveta" (nezáleží na tom komu). Ako to spraviť, keď "neinformovaných vrcholov" je už menej ako $(n - 1)$? Preto existujú okrem tzv. *objaviteľských správ* (tie do ešte neznámych vrcholov) aj *pomocné správy*. Ich úlohou je docieliť to, aby nejaký vyvolený "pomocník" (alebo aj pomocníci) dopomohli vrcholu u v odosielaní objaviteľských správ. Cieľom takejto pomoci je to, aby po nejakom čase (závisí od počtu ešte neznámych vrcholov) posielali všetci, vrchol u aj jeho pomocníci, iba objaviteľské správy do neznámeho sveta, pričom týchto správ je aspoň $(n - 1)$, teda máme garanciu doručenia aspoň jednej novoobjaviteľskej správy.

Na záver ešte podotknem, že správy v tomto algoritme majú špeciálnu štruktúru, ktorú tu však nebudem rozoberať (aj v mojom algoritme majú správy špeciálnu štruktúru, tam už budem pri ich predstavovaní dôslednejší). Celý algoritmus funguje v časovej zložitosti $O\left(n^3\right)$.

2.1.3 Hranovo k-súvislé grafy

Jedným z vrcholov publikovaných výsledkov bol algoritmus pre voľbu šéfa pre ľubovoľný hranovo k-súvislý graf. Mnohé vedomosti som v modifikovanej forme použil aj pre vytvorený algoritmus pre voľbu šéfa v ľubovoľnom k-súvislom grafe a rozoberám ich pri hlbšej štúdii tohto algoritmu. Preto v tejto časti uvádzam iba dosiahnuté výsledky.

Veta 2.1.2. *Existuje algoritmus pre broadcasting v jednoduchom prahovom modeli, ktorý má pre ľubovoľný hranovo k-súvislý graf G so zmyslom pre orientáciu (vrcholy poznajú svojich susedov a "rozpoloženie celého grafu") časovú zložitosť $O(2^k nm)$, kde n je počet vrcholov daného grafu G a m je počet hrán tohto grafu.*

Dôsledok 2.1.1. *Existuje algoritmus pre broadcasting v jednoduchom prahovom modeli na orientovanej d-dimenzionálnej hyperkocke s časovou zložitosťou $O(n^2 \log n)$, kde $n = 2^d$ je počet vrcholov danej hyperkocky.*

Veta 2.1.3. *Existuje algoritmus pre broadcasting v jednoduchom prahovom modeli, ktorý pre ľubovoľný hranovo k-súvislý graf bez zmyslu pre orientáciu (vrcholy vedia rozlíšiť iba svoje porty) zbehne v čase $O(2^k m^2 n)$.*

2.2 Voľba šéfa

Pri voľbe šéfa môžeme uvažovať viacero možností. Okrem prípadov bez a so znalosťou topológie siete sa totiž v prípade voľby šéfa dá uvažovať o:

- simultánnom zobudení všetkých iniciátorov na začiatku algoritmu, pričom v priebehu algoritmu sa už samovoľne nemôže zobudiť žiadny z vrcholov siete
- postupnom samovoľnom budení iniciátorov (ktorí majú ambíciu stať sa šéfom) počas behu programu

Zaujímavou časťou pre nás je simultánne zobudenie sa všetkých iniciátorov na začiatku algoritmu. Postupné samovoľné budenie je technicky zaujímavé, no algoritmicky sa pri ňom nevyužívajú nejaké extra nové postupy, iba sa nejakým spôsobom upravia tie, ktoré sa využili pri simultánnom prebudení. Preto sa v tejto časti, ako aj v ďalších, zaoberám pri voľbe šéfa iba prípadmi simultánneho zobudenia sa všetkých iniciátorov na začiatku behu programu. Ostatné vrcholy (nie iniciátori) sú zobudené prijatím nejakej správy.

2.2.1 Kruh

Algoritmus riešiaci problém voľby šéfa na kruhu (bez ohľadu na to, či je známy alebo nie je počet vrcholov kruhu n) je založený na dvoch základných myšlienkach:

- na pretláčanie správ pre tento problém sa používa rovnaký algoritmus ako pri broadcastingu na kruhu (Pretláčací algoritmus pre broadcasting na kruhu 2.1.1)
- samotná voľba je založená na modifikovanej verzii Franklinovho algoritmu [Fra82] pre voľbu šéfa v kruhu v asynchrónnych sieťach

Najprv neformálne popíšem tento modifikovaný Franklinov algoritmus (pre asynchrónne siete), potom uvediem aj jeho formálnejšiu verziu. Na tomto príklade totiž dobre vidno, ako sa dá pre jednoduchý prahový model (aj pre distribuované systémy s chybami všeobecne) využiť riešenie nejakého problému pre asynchrónnu sieť.

Všetkých iniciátorov nazveme *kandidáti*, ostatné vrcholy sa nazývajú *porazené* a nie sú účastníkmi tohto algoritmu, iba "posúvajú správy" smerom po kruhu od prijímateľa ďalej. Každý kandidát v má v sebe okrem svojho ID aj číslo fázy, v ktorej sa momentálne nachádza (začínajúce od 0). V jednej fáze sa v snaží "uchytiť" aspoň jedného svojho suseda (kandidátskeho) v rovnakej fáze (pošle im *zachytávaciu správu*). Ak je úspešný (niektorý zo susedov mu pošle naspäť *víťaznú správu*), zvýši si o jedna číslo svojej fázy, v opačnom prípade "čaká", kým ho niekto "zajme".

Treba poznamenať, že vrchol s najvyššou dvojicou [fáza, ID] vždy vyhráva, takže nemôže nastať deadlock. Vrchol v pritom môže zvýšiť svoju fázu iba v prípade,

že "uchytí" nejaký iný kandidátsky vrchol s rovnakou fázou a nižším ID (tento uchytený vrchol sa stáva porazeným a už vypadáva z "bojov" o celkové líderstvo v sieti - neposiela ďalej žiadne zachytávacie správy). Jeden kandidát s môže byť uchytený maximálne dvoma vrcholmi (tým po jeho ľavej i pravej strane - v prípade ďalších prechodov cez vrhol s musia mať prechádzajúce správy vyššiu fázu, pretože pri uchytení s si uchycujúci kandidáti zvýšili svoju fázu o hodnotu 1). Z toho vyplýva, že v ľubovoľnej fáze j môže do ďalšej fázy $j+1$ postúpiť maximálne 2/3 aktívnych kandidátov danej fázy j. Počet aktívnych kandidátov Q_j, ktorí sa dostali do fázy j, je teda najviac $k\,(2/3)^j$. Z toho vyplýva počet p postačujúcich fáz na voľbu šéfa je $p \leq 1 + \log_{3/2} k$. Na základe toho a tiež poznatku, že v ľubovoľnej fáze môže byť cez graf poslaných maximálne $3n$ správ (po hrane môže z oboch strán - sprava doľava aj zľava doprava - prejsť "informačná" správa s fázou j a tiež jedna "víťazná" správa pre fázu j) dostávame:

Tvrdenie 2.2.1. *Celkový počet poslaných správ v modifikovanom Franklinovom algoritme pre voľbu šéfa v asynchrónnej sieti je $O\,(n \log k)$, pričom n je počet vrcholov v kruhu a k je počet iniciátorov.*

Modifikovaný Franklinov algoritmus

Procedúra pre iniciátorov

faza:=0, stav:=kandidat
loop begin
pošli oboma smermi zachytávaciu správu obsahujúcu dvojicu [faza,ID]
počkaj, kým ti nie je doručená aspoň jedna víťazná správa
if stav=kandidat **then** faza++ **else** skonči
end loop

Spracovanie správy od kandidáta v kandidátskym vrcholom w

if dve *zachytávacie správy* od v sú vo w a platí $[\text{faza,ID}]_w \leq [\text{faza,ID}]_v$ **then**
w spoznal identitu lídra v a môže odštartovať finálnu fázu
end if
if správa došla do vrchola w s vyšším [faza, ID] **then**
sprava "sa stratí"
else if správa prišla do vrchola w s rovnakou fázou $faza$ a nižším ID **then**
stav_w:=porazeny, $[\text{faza, ID}]_w := [\text{faza, ID}]_v$
w odošle naspäť k odosielateľovi *víťaznú* správu

else
stav$_w$:=porazeny, [faza, ID]$_w$:= [faza, ID]$_v$
w iba pošle ďalej prijatú *informačnú* správu
end if

Spojením dvoch horezmieňovaných konceptov vznikol algoritmus pre voľbu šéfa pre kruh v jednoduchom prahovom modeli. Každý vrchol snažiaci sa odoslať na základe Franklinovho algoritmu správu (zachytávaciu alebo víťaznú) tak činí buď sprava doľava, alebo zľava doprava, pričom tieto správy (zachytávaciu aj víťaznú) považujeme v prípade "Pretláčacieho algoritmu pre broadcasting na kruhu 2.1.1" za informačné správy. Dostatočne dlho nechávame opakovať beh algoritmu 2.1.1 a v zodpovedajúcich časových okamihoch tohto algoritmu odosielame tieto informačné správy (či už vo forme zachytávacej alebo víťaznej správy). Prijímateľ akejkoľvek informačnej správy sa snaží na základe Franklinovho algoritmu o jej ďalšie šírenie, prípadne o poslanie odpovede (víťaznej správy), okrem toho však posiela v prípade potreby odosielateľovi aj potvrdenie v zmysle algoritmu 2.1.1 (v rámci niektorých krokov algoritmu 2.1.1). Po prijatí potvrdenia totiž odosielateľ vie, že už nemusí posielať danú informačnú správu, pretože jej príjemca ju už obdržal. Inými slovami, vrcholy snažiace sa odoslať zachytávaciu alebo víťaznú správu sú aktívne (nevedia, či ich adresát už dostal nimi odosielanú správu), kým tie, ktoré už obdržali potvrdenie, sú pasívne (vedia, že adresát už dostal odosielanú správu, preto ju nemusia ďalej posielať). Takto sa spasívňujú jednotlivé vrcholy (v každom behu algoritmu 2.1.1 aspoň jeden) a prebieha proces v rámci Franklinovho algoritmu.

Treba poznamenať, že v odosielaných správach je aj informácia o momentálnom kroku v rámci algoritmu 2.1.1, aby sa príjemca mohol pripojiť k vykonávaniu daného algoritmu v presnom kroku (1-4), v ktorom sa nachádzajú všetky ostatné vrcholy (inak by sme nemohli garantovať nasledujúcu vetu 2.2.1). Nasledujúca veta tiež hovorí o tom, "ako dlho" zhruba treba dokola vykonávať algoritmus 2.1.1 (dá sa to vyjadriť aj presne, to však teraz robiť nebudeme).

Veta 2.2.1. *Pre simultánne zobudenie k iniciátorov v kruhu existuje pre koncept jednoduchého prahového modelu algoritmus pre voľbu šéfa, ktorý pracuje v časovej zložitosti $O(n \log k)$.*

2.2.2 Kompletný graf

Výsledok pre voľbu šéfa na kompletnom n-vrcholovom grafe G so zmyslom pre orientáciu je tiež veľmi zaujímavý a z časového hľadiska veľmi priaznivý a prekvapujúci.

Veta 2.2.2. *Pri simultánnom zobudení všetkých iniciátorov v n-vrcholovom kompletnom grafe G existuje pre jednoduchý prahový model algoritmus na voľbu šéfa pracujúci v čase $O(n \log n)$.*

Algoritmus využíva podobný koncept *objaviteľských* a *pomocných* správ, ako tomu bolo v časti 2.1.2 opisujúcej postup pre broadcasting v kompletných grafoch bez zmyslu pre orientáciu. Hlavnou myšlienkou celého algoritmu je koncept akýchsi *tokenov*, ktoré putujú po sieti a snažia sa navštevovať nové a nové vrcholy. Správy si so sebou nesú okrem iných aj informáciu vo forme (D, r, R), pričom D je pre daný token množina doteraz nenavštívených vrcholov, $r - 1$ je počet *pomocníkov*, ktorých ešte treba regrutovať a R je množina už regrutovaných pomocníkov.

Označme množinu všetkých vrcholov grafu G ako V. Ak sa informácia v správe dostane do nového vrcholu u, ten sa ju snaží exportovať ďalej. Aby tak mohol spraviť, musí odoslať v jednom časovom okamihu $(n - 1)$ správ do doteraz nenavštívených vrcholov patriacich do množiny D, pričom v sieti nemôže byť odoslaná správa idúca do už známeho vrchola (nepriateľ by si totiž vybral práve túto správu). Poslať taký počet objaviteľských správ však on sám nie je schopný urobiť, preto okrem týchto objaviteľských správ do vrcholov patriacich D odošle aj prosbu o pomoc do už navštívených vrcholov $(V - D)$. Chceme totiž postupom času docieliť situáciu, aby sa v určitom časovom momente odoslalo $(n - 1)$ objaviteľských správ do množiny D, pričom v sieti sa nepošle žiadna neobjaviteľská správa (istota doručenia aspoň jednej objaviteľskej správy, teda informovania jedného nového vrchola).

Ak $|D| = d$, v jednom časovom momente bude musieť odosielať do množiny D svoje správy $\lceil (n-1)/d \rceil$ informovaných vrcholov z množiny $V - D$. Vrchol u bude v každom okamihu až do uplynutia času $\lceil (n-1)/d \rceil$ posielať objaviteľské správy do množiny D. Ak by sa totiž do posledného kroku daného časového intervalu neodoslala ani jedna objaviteľská správa, počas tohto posledného časového okamihu bude práve $\lceil (n-1)/d \rceil$ vrcholov (u a jeho pomocníci) posielať objaviteľské správy do vrcholov patriacich množine D. Preto u odošle vo svo-

jich správach okrem D-množiny ešte nenavštívených vrcholov daného tokena aj r- počet potrebných naregrutovaných vrcholov. Okrem toho bude medzi správami putovať aj informácia o už regrutovaných pomocníkoch, pretože chceme regrutovať nových a nových (túto informáciu si budeme pamätať v R). Všetky správy teda budú obsahovať trojicu (D, r, R). Ak správa dôjde k vrcholu patriacemu do množiny $V - D$ (teda k nejakému naregrutovanému pomocníkovi), ten o jedno zníži r (keďže on sám už je naregrutovaný), do R pridá samého seba a takéto nové informácie v trojici (D, r, R) odošle v prvom kroku ako pomocné správy všetkým vrcholom z množiny $V - (D \cup R)$. Okrem toho posiela počas r krokov objaviteľské správy vrcholom patriacim do D. Takýmto spôsobom si token vynúti to, že buď sa medzitým odoslala objaviteľská správa, alebo sa po istom čase odošle $n - 1$ objaviteľských správ do doteraz neznámych vrcholov, teda aspoň jeden musí byť informovaný.

Takýmto spôsobom rôzne tokeny od rozličných iniciátorov putujú po sieti. Po nejakom čase aspoň jeden z vrcholov grafu (označme ho t) dostane správu, v ktorej bude množina neznámych vrcholov D prázdna (ako keby sa nám podarilo broadcastovať túto správu). Do putujúcich správ zapíše každý novoobjavený vrchol (ak patril do množiny D) svoje ID, čiže t bude vedieť na základe všetkých ID rozhodnúť o jednom šéfovi (napr. ten s najnižším ID). Po skončení tejto prvej fázy potom t (možno aj ďalšie vrcholy) iniciuje ďalší broadcasting, v ktorom sa rozposiela ID šéfa.

Kapitola 3

Šéf v k-súvislom grafe s orientáciou

3.1 Úvod

Popíšme si uvažovanú situáciu: Na začiatku sa v sieti reprezentovanej grafom G zobudí niekoľko vrcholov (napr. lokálnych počítačov), ktoré budeme volať *iniciátormi*. Tí poznajú svoje vlastné ID číslo a tiež celkovú topológiu v sieti. Úlohou jedného z nich je presvedčiť ostatných, aby ho "zvolili za svojho šéfa", čo znamená, že po skončení algoritmu (požadujeme exaktnú termináciu, teda každý vrchol v určitom momente bude vedieť, že už nemusí očakávať nijakú činnosť súvisiacu s voľbou) bude každý vrchol v sieti (iniciátori, ale aj vrcholy, ktoré boli pri spustení algoritmu neaktívne) poznať ID tohto jediného vrchola. Navrhnutý a ďalej predstavený algoritmus využíva spôsob pretláčania správ prostredníctvom dopredu naplánovaných vlákien (threadov - budeme používať oba pojmy; niekedy budeme hovoriť aj o cestách, keďže tieto vlákna budú vlastne cesty), ktorý bol využívaný pri broadcastingu v hranovo k-súvislom grafe (predstavíme ho ďalej), ale aj špeciálnu štruktúru správ, ktoré musia "byť odolné" voči prípadným kolíziám (práve na takýchto kolíziách a akomsi strácaní niektorých threadov je celý algoritmus založený).

Veta 3.1.1 (Globálna verzia Mengerovej vety)**.** *Pre ľubovoľný graf platí nasledovné:*

1. *Graf je k-súvislý práve vtedy, keď obsahuje k nezávislých ciest medzi ľubovoľnými dvoma vrcholmi.*
2. *Graf je hranovo k-súvislý práve vtedy, keď obsahuje k hranovo disjunktných ciest medzi ľubovoľnými dvoma vrcholmi.*

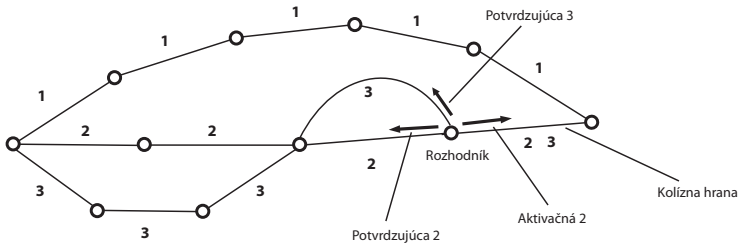
Každý iniciátor si v úvodnej fáze algoritmu vyberie ľubovoľný iný vrchol (tzv. cieľ, cieľový vrchol - tento pojem budeme používať aj pre iniciátorov inej než úvodnej fázy) a na základe topologickej znalosti celého grafu aj k hranovo disjunktných ciest (akoby vopred vybudovaných kanálov) $P = \{P_1, \ldots, P_k\}$ vedúcich k nemu (existenciu týchto vlákien máme zaručenú vďaka vete 3.1.1). Postupne sa iniciátor snaží pretlačiť správu, ktorá obsahuje okrem ďalších informácií aj jeho ID, k vybranému vrcholu. Na tom by nebolo nič zvláštne, keby boli všetky cesty od iniciátorov k ich cieľom navzájom disjunktné. Takáto "ideálna situácia" však nie je možná v priebehu celého algoritmu.

Definícia 3.1.1. Keďže iniciátorov môže byť v ľubovoľnej fáze niekoľko, označme vybrané hranovo navzájom disjunktné cesty vrchola i ako $P_{i1}, P_{i2}, \ldots, P_{ik}$, cieľ tohto iniciátora ako v. Vrchol i sa snaží postupne pretláčať cez tieto cesty informáciu do cieľového vrchola v. Nech $P_{ij} = (i = u_0, u_1, u_2, \ldots, u_{l_i} = v)$ je cesta vedúca od iniciačného vrchola i k cieľovému vrcholu v. Potom orientovaná hrana $e = \langle u_m, u_{m+1} \rangle$ môže byť v niektorom z nasledujúcich stavov:

- e je *pasívna* vzhľadom na cestu P_{ij} práve vtedy, keď cez ňu "pretiekla" správa pochádzajúca od vrchola i (t.j. obsahujúcu ID iniciátora i) v smere threadu P_{ij} a tiež v opačnom smere (to bude iný typ správy - o tom píšem neskôr), teda cez orientovanú hranu $\langle u_{m+1}, u_m \rangle$
- e je *aktívna* vzhľadom na cestu P_{ij} práve vtedy, keď nie je na ňu *pasívna* a správa od iniciátora i už niekedy prešla hranou $\langle u_{m-1}, u_m \rangle$. V prípade prvej hrany danej cesty P_{ij}, teda hrany $\langle u_0, u_1 \rangle$, hovoríme o *aktívnosti* od začiatku danej fázy až do okamihu, keď sa táto hrana stane *pasívnou* (teda po orientovanej hrane $\langle u_1, u_0 \rangle$ "pretečie" tzv. potvrďujúca správa)
- e je *spiaca* práve vtedy, keď nie je *aktívna* a ani *pasívna*

Vrchol u_m môžeme nazvať *aktívny*, *pasívny* alebo *spiaci* vzhľadom na P_{ij} v závislosti od toho, v akom stave sa nachádza hrana $\langle u_m, u_{m+1} \rangle$ prislúchajúca k jednej z ciest P_{ij} vedúcej cez tento vrchol.

Definícia 3.1.2. Hranu $\langle u, v \rangle$, v ktorej sa stretávajú aspoň dve cesty od dvoch rôznych iniciátorov, nazveme *kolízna hrana*. Vrchol u v takomto prípade nazveme *rozhodník*.



Obr. 3.1: ta tu je popis

3.2 Kolízie a ich riešenie

Rozhodník sa v prípade, že bude musieť rozhodnúť, ktorú správu prepustí ďalej po spoločne vyznačenom úseku dvojice niektorých ciest patriacich dvom iniciátorom, vyberie tú s väčším ID svojho iniciátora. Takáto situácia nastane, keď je rozhodník s aktívny vzhľadom na niektorú z ciest P_{ij} iniciátora i (pričom táto cesta má pokračovať po hrane $\langle s, t \rangle$) a v niektorom z ďalších krokov (kým sa nedostane do stavu pasívny) do neho príde správa iniciátora l, ktorá tiež požaduje pokračovanie po hrane $\langle s, t \rangle$. Rozhodník už po threade vedúcom od iniciátora s nižším ID nebude viac v danej fáze posielať aktivačné správy (tento pojem si vysvetlíme neskôr). Vyššie popísanú situáciu nazveme *kolízia*.

Definícia 3.2.1. Kolízia je stav, keď by jedna hrana mala byť aktívna vzhľadom na dve cesty dvoch rôznych iniciátorov.

Pri takto navrhnutom riešení kolízií budú niektoré thready zakapávať a s nimi postupne aj niektorí iniciátori. Ten s najväčším ID, naopak, bude postupne rozširovať a dobývať nové a nové územia (informácia o jeho ID sa bude šíriť týmto grafom). Po skončení jednej fázy sa všetky cieľové vrcholy, ktoré dostali v jej priebehu aspoň jednu správu s ID iniciátora, stanú novými iniciátormi a vyberú si na základe informácie v prijatej správe nový cieľ - vrchol doteraz neinformovaný o ID ich počiatočného iniciátora (v správe je ukladaná množina vrcholov doteraz navštívených a informovaných o ID tohto iniciátora). Ak sa v jednej fáze doručí do cieľového vrchola viac správ s ID rôznych iniciátorov,

daný vrchol "preberie" a ďalej šíri tú s najväčším ID. Takýmto spôsobom teda zakape iniciátor s nižším ID.

Zhrnutie algoritmu:
Iniciačný vrchol i si na začiatku fázy vyberie cieľový vrchol v, ktorý ešte nefiguruje v množine "príjemcov" S správy M a tiež k hranovo disjunktných ciest, cez ktoré sa bude snažiť pretlačiť správu M k príjemcovi v. V správe bude popísaný cieľový vrchol, preto sa v prípade, v ktorom nenastane kolízia, nemôže stať, že cesta niektorej správy sa skončí v inom vrchole (ak by cesta viedla cez ešte nenavštívený vrchol, ktorý si myslí, že správa je určená práve jemu). Pomocou akejsi procedúry popísanej ďalej (s popísaným princípom tejto procedúry) sa vrchol snaží počas trvania jednej fázy pretlačiť správu cez aspoň jedno vlákno (thread) k cieľovému vrcholu, ktorý sa v ďalšej fáze stáva novým iniciátorom (ak k nemu prešla správa cez aspoň jedno vlákno).

Keďže iniciátorov môže byť v jednej fáze viac, ich cesty sa môžu navzájom prepletať. Ak cieľový vrchol dostane počas jednej fázy viac ako jednu správu, do ďalšej fázy vstupuje ako iniciátor so správou s najvyšším ID. Pri vzájomnej kolízii správ v niektorom z rozhodníkov posúva vrchol ďalej tiež iba tú s najvyšším ID. Tento rozhodník je však naďalej "spätne aktívny" aj voči threadu, ktorého správa v ňom zakapala - vrcholy na tomto vlákne totiž netušia, že majú byť neaktívne, keďže nemôžu vedieť, že ich správa sa dostala v niektorom z kôl danej fázy až k rozhodníkovi a ten rozhodol o deaktivácii daného vlákna. Rozhodník je teda voči tomuto vláknu naďalej aktívny a posiela potvrďujúce správy vrcholu, od ktorého dostal správu o danom vlákne (posielanie týchto potvrďujúcich správ je riadené algoritmom). Takýmto spôsobom sa rozširujú správy obsahujúce ID niektorých iniciátorov. Postupne všetci iniciátori s nie najvyšším ID "zakapú" (ich rozširovanú správu pohltí iná), pričom vrchol s najvyšším ID spomedzi všetkých iniciátorov "zostane nažive" a po maximálne $(n-1)$ fázach rozšíri informáciu o svojom ID do všetkých ostatných vrcholov.

3.3 Posúvanie správ v algoritme

Procedúra KOLO() je hlavnou "posuvnou" konštrukciou celého algoritmu. Neskôr ukážeme, čo presne sa v jednom takomto obehu stane, pričom sa tiež zamyslíme nad tým, koľko takýchto kôl je potrebných spraviť v jednej fáze.

Ako uvidíme už za chvíľu, algoritmus používa dva typy správ - tzv. *aktivačnú* (dopredu smerom cez jednotlivé vlákna) a potom tzv. *deaktivačnú* (potvrďujúcu správu) - ack správu, ktorá putuje smerom dozadu cez jednotlivé vlákna. Čo je ich úlohou?

- *Aktivačná správa* - snaží sa po niektorom vlákne aktivovať nový vrchol vzhľadom na tento už dopredu určený thread, teda poslať správu smerom od iniciátora k jeho cieľu po vlákne, pre ktoré bola nadväznosť správ určená (iniciátor na začiatku vysiela aktivačné správy smerom k najbližším vrcholom patriacim do určených ciest, pričom tieto správy si so sebou nesú presnú informáciu o tom, kadiaľ, teda po ktorých vrcholoch a hranách, bude viesť ich cesta). V prípade kolízie viacerých aktivačných správ (mali by vo svojej ceste pokračovať po spoločnej hrane) sa vrchol na základe informácie v týchto správach o ich ID rozhodne, ktorú z nich prepošle ďalej a ktorú už ďalej nepustí a iba jej predchodcovi (vrcholu, odkiaľ daná správa prišla) bude posielať deaktivačné správy.
- *Deaktivačná (potvrďujúca, pasívna, ack) správa* - keďže o progrese aktivačnej správy vzhľadom na niektoré vlákno P_{ij} posielajúci vrchol nemá žiadnu informáciu (nemôže si byť istý, či tá bola poslaná ďalej - nepriateľ ju mohol zo siete v danom tiku hodín odobrať), posiela túto správu až do prijatia tzv. deaktivačnej správy. Adresát sa po prijatí aktivačnej správy snaží "upovedomiť" svojho predchodcu, odkiaľ správa prišla, o tom, že mu už nemusí posielať aktivačné (informačné) správy. To robí práve prostredníctvom tejto deaktivačnej správy. Po jej prijatí už vrchol nebude posielať po danom threade aktivačné správy a s určitosťou vie, že informácia bola pretlačená ďalej.

Algoritmus pre "posun" v hranovo k-súvislom grafe:
V každom kole každej fázy vykonávajú zúčastnené vrcholy nasledujúcu procedúru (tá bola uvedená v [DKKS]):

Procedure KOLO (vrchol w)
Nech A je množina aktívnych hrán vedúcich z vrchola w na začiatku daného podkola
for i:= 0 to k do // jedno podkolo
for $B \subseteq \{1, \ldots, k\}$ také, že $|B| = i$ do begin
// jedna iterácia sa udeje v jednom kroku - všetky správy v jednej iterácii sú
// posielané paralelne

Nech C je množina hrán vedúcich z w, cez ktoré došla do w akákoľvek "aktiváciu spôsobujúca správa" v aktuálnom kole (berieme všetky hrany, z ktorých došla akákoľvek správa bez ohľadu na to, či pri jej doručení nastala kolízia alebo nie - správa teda nemusela vyvolať aktiváciu, ale jej pôvodný zámer bol aktivovať daný vrchol)
for $e = \langle m, n \rangle$ také, že $\langle n, m \rangle \in C$ do
pošli ack správu cez e
for $e \in A$ také, že $e \in P_z$ a zároveň $z \notin B$ do
pošli aktivačnú správu cez e (v prípade, že by si cez e mal poslať viac aktivačných správ, vyber si tú s najväčším ID)
end for;
end for;
end procedure;

Keďže ide o synchronizovaný graf, pomocou "informácie o čase" vieme zosynchronizovať ľubovoľné vrcholy. Iniciátori a tiež ostatné posielajúce vrcholy v danej fáze totiž v správe odosielajú aj aktuálnu časovú hodnotu, ktorú si pri každom tiku hodín aktualizujú. Takýmto spôsobom vieme synchronizovať začiatky fáz, kôl a zaručiť, že jednotlivé kroky v rôznych vrcholoch nebudú navzájom kolidovať (dôležité pri dokazovaní správnosti a časovej zložitosti algoritmu). Vrchol teda po obdržaní správy prečíta aj obsiahnutú informáciu o čase a podľa nej sa zapojí do tej-ktorej časti vykonávania procedúry KOLO().

3.4 Spasívňovanie hrán a threadov

Definícia 3.4.1. Dané vlákno P_{ij} je *aktívne*, ak v ňom existuje *aspoň jedna aktívna hrana*. Naopak, vlákno P_{ij} nazveme *pasívne* vtedy, keď už cez neho "pretiekla" správa od iniciátora k svojmu cieľu a neexistuje v ňom žiadna hrana, cez ktorú by neprešla potvrďujúca ack správa (všetky vrcholy v danom vlákne sú *pasívne*). Vlákno nazveme *pasívnym* aj v prípade, že sa dostalo do kolízie s vláknom so správou s vyšším ID a v danom úseku (od iniciátora po rozhodníka) neexistuje aktívny vrchol, t. j. predchodca rozhodníka a všetky ďalšie vrcholy až po iniciátora obdržali potvrďujúcu ack správu.

Lema 3.4.1. *Dovtedy, kým nie sú v danej fáze pasívne pre nejaké číslo* $r \in \{1, \ldots, k\}$ *všetky thready s týmto poradovým číslom od všetkých iniciátorov danej*

fázy platí, že sa v každom kole spasívni minimálne jedna hrana (môže sa spasívniť aj potom, to nás však už nezaujíma).

Dôkaz. Každé aktívne vlákno má podľa definície aspoň jeden aktívny vrchol (hranu). V prípade vlákna, po ktorom ešte neprešla ani jedna správa, je týmto aktívnym vrcholom (aktívnou hranou) iniciátor (hrana vedúca od iniciátora k ďalšiemu vrcholu tejto cesty).
Lemu dokážeme *sporom.* Nech tvrdenie neplatí, t.j. snažíme sa, aby v danom kole nenastala situácia, v ktorej sa doručí nejaká ack správa. Na základe hore uvedeného cieľa ukážeme, že v želanom prípade musí existovať na začiatku každého i-teho podkola (podmnožina B obsahuje postupne všetky i-prvkové podmnožiny množiny $\{1, 2, \ldots, k\}$) daného algoritmu KOLO() minimálne i aktívnych ciest s rozličnými poradovými číslami, po ktorých v danom kole prešla aktivačná správa - $P_{r_1 l_1}, P_{r_2 l_2}, \ldots, P_{r_i l_i}$, pričom r_s sa môže rovnať r_t, ale $l_s \neq l_t$ $\forall s, t$. Tvrdenie dokážeme *matematickou indukciou.*

Tvrdenie 3.4.1. *Predpokladajme, že až do posledného podkola (keď podmnožina B obsahuje iba celú množinu $\{1, 2, \ldots, k\}$) nebola poslaná žiadna potvrďujúca ack správa. Potom na začiatku každého i-teho podkola (podmnožina B obsahuje postupne všetky i-prvkové podmnožiny množiny $\{1, 2, \ldots, k\}$) daného algoritmu KOLO() existuje minimálne i aktívnych ciest s rozličnými poradovými číslami, po ktorých v danom kole prešla aktivačná správa.*

Dôkaz. Toto tvrdenie triviálne platí pre úvodné nulté podkolo kola a tiež pre prvé podkolo - podľa predpokladu neexistuje poradové číslo, s ktorým by sme už nemali ani jedno aktívne vlákno. To znamená, že sa v úvodnom nultom podkole pošle minimálne k aktivačných správ, takže na začiatku prvého podkola budeme mať aspoň jednu aktivovanú cestu, po ktorej v danom kole prešla aktivačná správa.

Nech tvrdenie platí pre p-te podkolo, t.j. existuje práve $P_{l_1 m_1}, P_{l_2 m_2}, \ldots, P_{l_p m_p}$, pričom l_s sa môže rovnať l_t, ale $m_s \neq m_t$ $\forall s, t$ (v prípade, že je takýchto ciest s rozličnými poradovými číslami viac, tvrdenie je splnené aj pre $(p+1)$-té podkolo - preto uvažujeme *práve* p takýchto ciest). Označme $P = \{m_1, \ldots, m_p\}$, pričom existuje aspoň jedna cesta s doručenou aktivačnou správou prenesenou v danom kole s poradovým číslom z P pre každé číslo $r \in P$. Pozrime sa na prípad, keď sa B v danom podkole rovná P. V danom podkole je teda poslaných p potvrďujúcich ack správ (po cestách s poradovým číslom $m_1, \ldots, m_p$ niektorých

iniciátorov $l_1, \ldots, l_p$). Navyše, keďže pre každé číslo $r \in 1, \ldots, k$ existuje aspoň jedna aktívna cesta (podľa predpokladu a na základe toho, že doteraz nebola doručená žiadna spasívňujúca správa), v danom podkole bude poslaných aspoň $k-p$ aktivačných správ. To znamená, že v danom kroku podkola bude poslaných minimálne k správ, takže aspoň jedna bude musieť byť doručená. Keďže však nechceme, aby bola doručená ack správa, musela byť doručená aktivačná správa. Tá ale nepatrila ceste s poradovým číslom patriacim do množiny P, takže bola "zaktívnená" cesta s novým poradovým číslom, teda toto tvrdenie platí aj pre $(p+1)$-te podkolo. □

Na základe dokázaného tvrdenia 3.4.1 ale musí byť na začiatku k-teho podkola (keď $B = \{1, \ldots, k\}$) poslaných minimálne k rôznych aktivačných správ, pričom aspoň jedna bola poslaná po ľubovoľnej ceste s poradovým číslom t pre akékoľvek $t \in \{1, \ldots, k\}$. To ale znamená, že v k-tom podkole bude poslaných k potvrďujúcich ack správ, zatiaľ čo ani jedna aktivačná správa (pretože v tomto podkole $B = \{1, \ldots, k\}$). Akokoľvek sme sa teda snažili vyhýbať sa ack správam, v tomto podkole bude aspoň jedna poslaná, čo je v spore s predpokladom. Lema je tým pádom dokázaná. □

Predpokladajme teraz na chvíľu, že v jednej fáze je dostatočný počet kôl na to, aby sme mohli vysloviť nasledujúce tvrdenie. V ďalšej časti ukážeme, koľko kôl potrebujeme vykonať, aby sme mohli s istotou tvrdiť, že máme potrebný dostatočný počet kôl.

Lema 3.4.2. *V každej fáze platí, že sa doručí správa od iniciátora obsahujúca najvyššie rozposielané ID k jeho cieľovému vrcholu (môžu sa popritom doručiť aj správy niektorých ostatných iniciátorov).*

Dôkaz. Ako sme si ukázali v predchádzajúcej leme 3.4.1, v každom kole sa až do momentu, keď sú pasívne všetky thready s rovnakým poradovým číslom r od všetkých iniciátorov danej fázy pre akékoľvek $r \in \{1, \ldots, k\}$, spasívni aspoň jeden vrchol. Pozrime sa na iniciačný vrchol, ktorého správa obsahuje najvyššie ID spomedzi všetkých iniciátorov. Všetkých jeho k threadov končí v cieľovom vrchole. Aj v prípade kolízie s inými vláknami totiž vlákno idúce z tohto iniciátora "prerazí" ostatné thready, pretože jeho ID je najvyššie spomedzi všetkých. Postupne sa "spasívňujú" jednotlivé hrany a vrcholy v threadoch, ktoré buď dorazia až do cieľa, alebo končia svoju cestu v nejakom rozhodníkovi.

Keďže sme predpokladali, že vo fáze je dostatočný počet kôl, musí raz prísť aj na spasívnenie aspoň jedného threadu vedúceho od iniciátora s najvyšším ID k jeho cieľu, t.j. tento cieľ určite dostane správu od daného iniciátora. Ak by totiž aj najprv "spasívneli" všetky ostatné vlákna okrem threadov patriacich iniciátorovi s najvyšším ID, stále nie je porušená podmienka predchádzajúcej lemy, a síce "dovtedy, kým nie sú pasívne všetky thready s rovnakým poradovým číslom od všetkých iniciátorov danej fázy". Stále je totiž pre ľubovoľné $i \in \{1, \ldots, k\}$ aktívne vlákno s daným poradovým číslom i tohto iniciátora s najvyšším ID. Teda postupne sa budú "spasívňovať" aj hrany idúce po vláknach vedúcich z tohto iniciátora, až kým sa aspoň jedno jeho vlákno nestane pasívnym - máme istotu, že správa sa od iniciátora s najvyšším ID dostala až k jeho cieľovému vrcholu a teda lema platí. □

3.5 Algoritmus a jeho zložitosť

Teraz sa poďme pozrieť na to, koľko kôl musí obsahovať jedna fáza na to, aby sme s určitosťou mohli tvrdiť horeuvedenú lemu 3.4.2. Poďme si postupne rozanalyzovať danú situáciu. V každej fáze môže byť maximálne n iniciátorov (aj keď je to možné iba v úvodnej fáze, nadhodnotíme tento počet takýmto spôsobom). Jeden konkrétny iniciátor má "vybraných" k hranovo disjunktných ciest - nemôže využiť žiadnu hranu dvakrát. Jeho thready teda môžu prechádzať po maximálne m hranách (toľko je totiž hrán v celom grafe). Ak by sa teda v každom kole spasívnela hoci len jedna hrana patriace nejakému threadu (čo máme za horeuvedených podmienok zaistené), po mn kolách sme určite v stave, že pre niektoré poradové číslo $r \in \{1, \ldots, k\}$ sú pasívne všetky thready s týmto poradovým číslom od všetkých iniciátorov danej fázy, a teda sa nám s určitosťou podarilo "pretlačiť" správu od iniciátora s najvyšším ID k jeho cieľu. Každý akýmkoľvek spôsobom pracujúci vrchol danej fázy môžeme (na základe rozposielaného "časového údaju") nastaviť tak, že po poslednom (mn)-tom tiku hodín akoby zaspí (v prípade, že nebol cieľovým vrcholom) a čaká, čo sa bude diať v novej fáze. Môžeme teda vysloviť nasledujúcu vetu:

Veta 3.5.1. *Po nm kolách sa uskutoční jedna fáza, teda iniciátor s najvyšším ID pretlačí informáciu do niektorého nového vrchola.*

Vidíme, že po $(n-1)$ fázach sa všetky vrcholy dozvedia o najvyššom ID v sieti. V prípade, že počas behu algoritmu sa dostalo do niektorého z vrcholov viac

správ s rôznymi ID, daný vrchol ich jednoducho porovná a vyberie si spomedzi nich najvyššie ID. Šéf bude o sebe vedieť na základe toho, že nedostane správu s vyšším ID ako je jeho vlastné. Takto môžeme po $(n-1)$ fázach ukončiť beh algoritmu s istotou, že bola uskutočnená úspešná voľba šéfa. Na záver sa pozrime na časovú veľkosť procedúry KOLO(). Keďže procedúra využíva všetky podmnožiny k-prvkovej množiny, jej časová zložitosť je 2^k.

Veta 3.5.2. *Existuje algoritmus, ktorý v jednoduchom prahovom modeli s konkrétnym hranovo k-súvislým grafom so zmyslom pre orientáciu zvolí šéfa v čase $O\left(2^k n^2 m\right)$, pričom n je počet vrcholov daného grafu a m je počet jeho hrán.*

3.6 Hyperkocka

Na záver tejto kapitoly pridávam zaujímavé aplikovanie predchádzajúcej vety na prípade *hyperkocky*. Je viacero ekvivalentných definícií tejto konštrukcie, ja uvádzam nasledujúcu:

Definícia 3.6.1. Hyperkocka s dimenziou d (d-dimenzionálna hyperkocka) je graf, ktorého vrcholy tvoria všetky možné binárne vektory dĺžky d. Hrana medzi dvojicou vrcholov existuje práve vtedy, keď sa vektory týchto vrcholov líšia práve v jednom bite.

Nasledujúce tvrdenie udávam bez dôkazu, pretože ten nie je predmetom tejto časti (aj keď je veľmi jednoduchý - jednotlivé časti sa dajú dokázať napr. *matematickou indukciou*).

Tvrdenie 3.6.1. *Pre d-dimenzionálnu hyperkocku platí, že počet jej vrcholov $n = 2^d$ a počet jej hrán $m = \frac{d2^d}{2}$, čo sa dá zapísať aj ako $m = \frac{nlogn}{2}$.*

Pre hyperkocku využijeme predchádzajúci algoritmus pre hranovo k-súvislý graf, pričom ako cieľový vrchol si ľubovoľný iniciátor u na začiatku každej fázy vyberie svojho nasledovníka v vo virtuálnej hamiltonovskej ceste prechádzajúcej cez hyperkocku. V prípade vytvorených threadov medzi u a v využijeme nasledujúcu lemu (opäť bez dôkazu, ten však nie je nijako náročný):

Lema 3.6.1. *Medzi vrcholmi u a v existuje d navzájom hranovo disunktných ciest, pričom dĺžka ľubovoľnej z nich nie je viac ako 3.*

Jedna fáza teda trvá v prípade hyperkocky namiesto času $O\left(2^d nn\log n\right) = O\left(n^3 \log n\right)$ iba $O\left(2^d nd\right) = O\left(n^2 \log n\right)$. Na základe lemy 3.6.1 totiž každá z d disjunktných ciest obsahuje maximálne 3 hrany, teda v každej fáze musia správy prejsť po maximálne $3d$ namiesto $m = O\left(n \log n\right)$ hranách.

Veta 3.6.1. *V d-dimenzionálnej hyperkocke existuje algoritmus pre jednoduchý prahový model, ktorý zvolí šéfa v čase $O\left(n^3 log n\right)$*

Dôkaz. Hore sme už uviedli trvanie jednej fázy. Keďže fáz je $n - 1$, celkový čas algoritmu je $O\left(n\right)$-násobok trvania jednej fázy. □

Kapitola 4

Šéf v k-súvislom grafe bez orientácie

4.1 Úvod

Na rozdiel od predchádzajúceho prípadu nevedia jednotlivé vrcholy o topológii grafu, preto nemôžu "dopredu naplánovať" k ciest (threadov) smerujúcich k jednému cieľovému vrcholu. Preto táto časť nového algoritmu bude musieť byť oproti predchádzajúcej časti upravená (stále však budeme využívať koncept threadov, ibaže trochu pozmenený). Zmenená musí byť aj informácia, ktorú nesie správa. Tieto zmeny uvedieme, vysvetlíme a okomentujeme v nasledujúcej časti. Treba si však uvedomiť, že každý vrchol pozná lokálnu topológiu, t. j. vie od seba odlíšiť jednotlivé porty, ktoré majú zároveň svoje pevne dané čísla. Čo však zostáva, je schopnosť "zladenia času" medzi vrcholmi a tiež synchronizácia začiatkov fáz, kôl a jednotlivých krokov. Podobne ako v predchádzajúcej časti sa tiež budú riešiť kolízie.

V nasledujúcej časti budeme narábať s niektorými novými, ale aj starými pojmami. Ak neupresníme inak, vyskytujúci sa pojem má rovnakú definíciu, ako sme ho predstavili a objasnili v predchádzajúcej časti. Nové pojmy, samozrejme, zadefinujeme a objasníme si ich význam. Vopred prezradíme, že na pretláčanie správ po jednotlivých threadoch sa bude rovnako ako v predchádzajúcej časti používať procedúra KOLO(vrchol w). Keďže nám však nie je známa topológia

siete, musíme vyriešiť a pristaviť sa pri niektorých vzniknutých problémoch.

Definícia 4.1.1. O nadväznosti správ M a N hovoríme vtedy, ak je N poslaná na základe prijatia správy M niektorým vrcholom v. To sa môže stať pri šírení informácie cez thread (vrchol v je súčasťou niektorého threadu a informácia je cez neho iba pumpovaná a posúvaná ďalej) alebo ak je vrchol v iniciátorom v nasledujúcej fáze, pričom správa M s cieľom vo vrchole v bola do neho doručená v tej predchádzajúcej fáze (čiže N je niektorou zo správ odoslaných vrcholom v ako iniciátorom tejto novej fázy).

4.2 Lokálne známy svet

Pred vysvetlením nových skutočností už vopred objasníme pojem cieľ správy. K nemu sa vrátime aj neskôr, pretože na jeho úplné vysvetlenie potrebujeme zadefinovať a vysvetliť ďalšie pojmy. Aby však pri čítaní nasledujúceho textu nevznikali zbytočné otázky, podávame aspoň čiastočné vysvetlenie tohto pojmu už tu. V algoritme totiž nebude cieľ správy (resp. threadu - cieľ niektorého z threadov vedúcich od iniciátora) daný exaktne, pretože nepoznáme topológiu siete. Cieľ threadu, správy bude v nadväznosti správ putujúcej cez daný thread opísaný ako "niektorý z nasledovníkov niektorého (presne vypísaného) vrcholu". Nemôžeme hovoriť o exaktnom cieli, nakoľko nepoznáme doteraz nenavštívených nasledovníkov vrchola (ľubovoľného).
A ako sám seba rozpozná cieľový vrchol, keď nie je v správe exaktne definovaný? V správe je totiž v istej forme odosielaná aj informácia, kto je jej odosielateľom. Ak správa (resp. thread, ktorý reprezentuje daná správa) obsahuje svoj cieľ opísaný ako "nasledovník vrchola v" a príjemca indikuje odosielateľa ako vrchol v, hneď vie, že práve on je cieľovým vrcholom daného vlákna.

V tejto časti budeme narábať s pojmom *port*. Jeho význam je intuitívne čitateľovi jasný a presne v takomto význame s ním budeme narábať aj my. *Port* je "kanál" vedúci od jedného vrchola k druhému (je to orientovaná hrana v grafe). Keďže hovoríme o hranovo k-súvislom grafe, každý vrchol má aspoň k portov (každá hrana $\langle u, v\rangle$ sa skladá z dvoch portov $\langle u, v\rangle$ a $\langle v, u\rangle$).

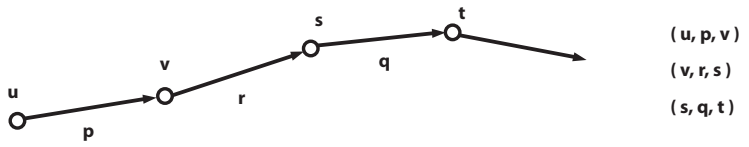
Pre algoritmus bez topologickej znalosti zavedieme pojem *známa trojica*. Ak vrchol u posiela prostredníctvom portu p správu doteraz neznámemu vrcholu v (tento pojem si podrobnejšie vysvetlíme neskôr, zatiaľ sa uspokojíme s intuitívnym významom tohto pojmu), táto informácia bude obsiahnutá aj v ďal-

ších nadväzujúcich správach odoslaných vrcholom v, ako aj vo všetkých ďalších nadväzujúcich správach (táto znalosť bude figurovať vo všetkých nadväzujúcich správach N odoslaných niektorým vrcholom na základe prijatia správy M obsahujúcej danú znalosť). Ako vlastne vrchol v získa novú vedomosť? Ten totiž dostane znalosť o tom, že u odosielal prostredníctvom portu p, pričom pridá k dvojici "svoju značku" (identifikátor), čím vytvorí trojicu *(u, p, v)*.

Definícia 4.2.1. Známa trojica je akási znalosť (informácia) zapísaná vo forme (u, p, v), ktorá hovorí, že vrchol u odoslal prostredníctvom portu p správu vrcholu v.

Ako sa budú nadväzujúce správy šíriť a šíriť ďalej (k novým, doteraz neznámym vrcholom a od nich ešte ďalej), bude ako ich súčasť postupne pribúdať známych trojíc *(u, p, v)* a narastať akási vedomosť. Pomenujme množinu všetkých známych trojíc nachádzajúcich sa v správe M ako *vedomosť známych trojíc* tejto správy a označne ju $K(M)$.

Definícia 4.2.2. Množinu všetkých trojíc (u, p, v) nachádzajúcich sa v správe M nazývame vedomosť známych trojíc, označujeme ju K(M).



Obr. 4.1: Vedomosť známych trojíc

V situácii, keď vrchol u dostane správu, ktorej cieľovým vrcholom je on sám, počká tento vrchol na koniec danej fázy (na základe času zaznamenaného v danej správe, tikotu hodín a vedomosti o dĺžke ľubovoľnej fázy) a v tej ďalšej sa stane jedným z iniciátorov (možno jediným). V správe však dostane aj vedomosť známych trojíc. Niektoré zo známych trojíc obsiahnutých v tejto vedomosti môžu mať ako svojho odosielateľa samotný vrchol u - (u, p, v). Vrchol vďaka tejto vedomosti známych trojíc, ako i ďalším, akoby pozná istú časť sveta (napr. ak vedomosť známych trojíc obsahuje známe trojice (u, p, v) a tiež (v, r, s), vrchol vie, že do vrchola s sa dostane prostredníctvom portu p, pričom vrchol v má použiť port r). Ostatná časť sveta je pre neho neznáma.

Pri tejto príležitosti objasníme aj vyššie spomenutý pojem *doteraz neznámy vrchol*. To je taký vrchol v, do ktorého prišla správa z vrchola u cez port p, pričom

sa trojica (u, p, v) až doteraz nevyskytovala v množine známych trojíc. Názov doteraz neznámy vrchol môže byť trochu mätúci - nemusí ísť totiž o vrchol, ktorý doteraz nebol navštívený. Mohol byť navštívený, ale z iného vrchola ako u, teda pre u je port vedúci k tomuto vrcholu "skokom do neznáma". Nadväzujúce správy už ale budú obsahovať znalosť o trojici (u, p, v) a z tohto pohľadu už vrchol v nebude neznámy.

Definícia 4.2.3. Známa cesta je ľubovoľná postupnosť vrcholov $u_1, \ldots, u_i$, pričom vo vrchole $u = u_1$ sa vyskytuje správa M obsahujúca vedomosť známych trojíc $(u_1, p_1, u_2), \ldots, (u_{i-1}, p_{i-1}, u_i)$.
Neznáma cesta je postupnosť skladajúca sa zo známej cesty zväčšená o vrchol u_{i+1}, pričom (u_i, p_i, u_{i+1}) nepatrí medzi vedomosť známych trojíc.

Definícia 4.2.4. Neznámy vrchol z pohľadu iniciátora u je vrchol v, pričom z vedomosti známych trojíc obsiahnutej v správe iniciátora (na základe ktorej sa stal iniciátorom) sa dá vyskladať neznáma cesta s koncom vo vrchole v (čiže neznámy vrchol z pohľadu iniciátora u môže byť aj jeho susedný vrchol v, pričom trojica (u, p, v) nepatrí medzi vedomosť známych trojíc). Známy vrchol z pohľadu iniciátora je ľubovoľný vrchol ležiaci na niektorej známej ceste.

4.3 Rozširovanie územia a riešenie kolízií

Ako teda vlastne funguje šírenie informácie z iniciátora? On sám si vyberie k *portovo disjunktných vlákien* (analógia *hranovej disjunktnosti*) vedúcich ku maximálne k neznámym vrcholom (viesť môžu ku $1, \ldots, k$ neznámym vrcholom). Môže využiť susedné neznáme vrcholy, ako aj neznáme vrcholy prislúchajúce niektorej neznámej ceste. Tu treba poznamenať, že kým existuje aspoň k doteraz neprejdených portov (umiestnených na ľubovoľných miestach v sieti), stále sa dá nájsť k portovo disjunktných vlákien k aspoň jednému neznámemu vrcholu.
Ak iniciátor vyberie k portovo disjunktných ciest, do správ posielaných cez tieto cesty zakóduje aj presný popis danej cesty (aby známe vrcholy vedeli, kam ďalej "pretláčať" správu). Ak dostane vrchol v správu, pomocou tohto popisu cesty, informácie o čísle danej cesty (rovnako ako v prípade úplnej topologickej znalosti si jednotlivé cesty očíslujeme číslami $1, \ldots k$), poslanom časovom údaji (na synchronizáciu) a ďalších informácií (napr. sa musí pozrieť, či on nebol cieľovým vrcholom) rozhodne, čo ďalej robiť. V princípe je ale tento krok (rozhodovanie a narábanie so správami) rovnaký ako v prípade úplnej topologickej znalosti

(keďže procedúra KOLO() je pre oba prípady rovnaká). Aj v tomto prípade narábame s aktivačnými a potvrďujúcimi ack správami.

Čo však vtedy, ak vrchol v dostane v jednej fáze správy z viacerých threadov (v prípade *kolízie*)? Môže nastať viacero prípadov, pred ich vymenovaním si ešte musíme zadefinovať nový pojem, ktorý nám pomôže vyriešiť jednu z nechcených situácií.

Keďže jeden iniciátor nemá iba jedného, ale až k cieľových vrcholov, môže sa z neho v jednej fáze potenciálne doručiť až k rôznych správ do cieľových vrcholov, pričom títo sa v ďalšej fáze stanú iniciátormi "pretláčajúcimi" správu s rovnakým ID. Následne môžu ich cesty kolidovať, čo by robilo problémy (rozpisujeme sa o tom v ďalšej časti). Preto si v nadväznosti správ budeme postupne pamätať aj čísla ciest, cez ktoré putovali nadväzujúce správy.

Definícia 4.3.1. Dodatočné ID je postupnosť čísel z množiny $\{1, \ldots, k\}$. Iniciátor hociktorej z fáz pri príprave správy pre vlákno j vytvorí novú postupnosť na základe tej starej, prijatej od iniciátora v predchádzajúcej fáze (keď bol terajší iniciátor cieľovým vrcholom), a to tak, že k nej na jej koniec pripojí číslo j.

Pred vyslovením a dokázaním nasledujúceho tvrdenia si prezradíme, ako budeme v algoritme postupovať v prípade, ak je vrchol v jednej fáze cieľovým vrcholom pre viacero správ. V takomto prípade cieľový vrchol vyberie pre "iniciáciu" ďalšej fázy (ako iniciačnú správu) jednu zo správ s najvyšším ID, a to tú s najväčším dodatočným ID (až do konca toho-ktorého aktuálneho kola však v určených časových krokoch posiela na základe algoritmu potvďujúce ack správy odosielateľom, od ktorých dostal tieto "koncové správy"). Toto zaručí, že v každej fáze budeme mať maximálne n iniciátorov - v každom vrchole najviac jedného.

Tvrdenie 4.3.1. *Žiadne dve putujúce správy patriace do rozličných threadov (od rozličných iniciátorov) s rovnakým ID nemajú rovnaké dodatočné ID.*

Dôkaz. Túto lemu dokážeme *sporom*. Nech by mali dve správy rovnaké ID aj dodatočné ID a zároveň patrili do rozličných threadov. Nech ich dodatočné ID má tvar $d_1 d_2 \ldots d_g$. Znamená to, že obe patria do threadu s rovnakým poradovým číslom d_g. Tiež to znamená, že v predchádzajúcich fázach $(g-1), \ldots, 1$ mali "dozadu nadväzujúce správy" (správy v predchádzajúcich threadoch, cez ktoré sme sa postupne dostali až do aktuálneho vlákna) poradové čísla $d_{g-1}, \ldots, d_1$.

Keďže do dodatočného ID pridávame číslo iba v iniciátorovi novej fázy, v l-tej fáze majú všetky putujúce thready správy zložené z l čísel (keďže nijaké nemohli vzniknúť samovoľne v priebehu algoritmu - iba na základe prijatých správ - teda všetky majú svojich počiatočných predkov v iniciátoroch algoritmu).

Treba tiež podotknúť, že v prípade kolízie v niektorom cieľovom vrchole sa na základe vyššie spomenutého ďalej "pretláča" iba správa s najvyšším ID, prípadne najvyšším dodatočným ID, ktorá obsahuje dodatočné ID obsahujúce čísla threadov od počiatku správy v iniciácii algoritmu až po uvažovaný okamih. Nemôže sa teda stať, že správa akoby vo svojom dodatočnom ID obsahovala časť z dodatočného ID niektorého zakapaného threadu alebo že by počas svojho života od iniciácie algoritmu menila svoje ID (najvyššie ID putuje stále, ostatné ID zakapávajú). Dodatočné ID obsiahnuté v správe je presný popis "chodu" danej správy s daným presným ID od začiatku algoritmu až po daný okamih.

Z popisu je tiež jasné, že z jedného iniciátora nemôže smerovať viac threadov s jedným konkrétnym poradovým číslom (práve jedno vlákno pre ľubovoľné poradové číslo). Z tohto všetkého vyplýva, že správy síce postupne putovali po vláknach s rovnakým poradovým číslom $(d_1, \ldots, d_g)$, ale ich iniciátormi boli dva rozličné vrcholy s nerovnakým ID. To je však spor s tým, že tieto dve správy nesú rovnaké ID, teda tvrdenie je dokázané a my v ďalšej časti môžeme bez nejakých problémov využívať koncept dodatočných ID. □

V nasledujúce časti sa už nebudeme zaoberať cieľovými vrcholmi, ale vrcholmi nachádzajúcimi sa na niektorej zo známych ciest a popíšeme si, ako riešiť prípadné kolízie vo vrchole v. Pri rovnakých, resp. rozličných iniciátoroch máme na mysli iniciátorov danej fázy (nie iniciátorov pri spustení algoritmu). Čo teda robiť, ak do vrchola v príde viac správ:

Ak pochádzajú z rovnakého iniciátora (je možné zistiť vďaka popisu cesty - je v nej zaznačený aj iniciátor) a žiadna správa neobsahuje ako svoj cieľ "niektorý z nasledovníkov v":
Tento prípad je tu spomenutý len pre úplnosť, inak nie je tento prípad ničím výnimočný. Iniciátor totiž vyberá k portovo disjunktných ciest, nie vrcholo disjunktných, a teda vrchol v iba "posunie" tieto správy v smere k cieľu po už vopred daných threadoch.

Ak pochádzajú z rovnakého iniciátora, pričom jedna zo správ má ako svoj cieľ zaznačené "niektorý z nasledovníkov v":

Opäť veľmi jednoduchý prípad. Správu obsahujúcu "neznámy" cieľ sa vrchol v snaží "pretlačiť" cez niektorý svoj neznámy port (neznámy vzhľadom na túto správu - port, ktorý nie je obsiahnutý vo vedomosti známych trojíc), ostatné správy "pretláča" po vopred daných vláknach ďalej po známych cestách.

Ak pochádzajú z rovnakého iniciátora, pričom aspoň dve správy majú ako svoj cieľ zaznačené "niektorý z nasledovníkov v":
Ak je takýchto správ s "neznámym" cieľom s, musí existovať aspoň s doteraz neznámych portov. Toto má na starosti iniciátor - na základe vedomosti známych trojíc zapísaných v prijatej správe vie, koľko aspoň má ktorý vrchol neznámych portov (odvíja sa od toho, že každý vrchol má aspoň k portov, keďže ide o hranovo k-súvislý graf). Daný vrchol sa snaží týchto s správ "pretláčať" cez jednotlivé svoje neznáme porty, pričom každú správu cez práve jeden port. Ak by to totiž takto nerobil, nevedeli by sme zaručiť korektnosť algoritmu (ktorá bude ukázaná v ďalšej časti).

Ak pochádzajú z rôznych iniciátorov s rozličnými ID:
Vrchol v "pretláča" ďalej po threadoch iba správy s najvyšším ID. Tie s nižším identifikátorom "stopne", t. j. neposúva ich už ďalej po ich vyznačenom vlákne. Nemôže ale stopnúť činnosť celého vlákna, nakoľko predchodca posielajúci danú správu s nižším ID si nemôže byť istý o jej doručení. Vrchol v teda v tom-ktorom kole danej fázy posiela potvrďujúcu ack správu, aktivačné už ale po danom threade neposiela (podobný princíp ako pri prípade so známou topológiou).

Ak pochádzajú z rôznych iniciátorov s rovnakými ID:
Táto možnosť je najproblémovejšia. Vyriešime ju ale jednoducho, pomocou konceptu dodatočných ID. Stačí nám totiž porovnať dodatočné ID - podľa horeuvedenej lemy totiž vieme, že správy s rovnakými ID putujúce v jednej fáze majú rozličné dodatočné ID. Takto opäť "pretláčame" ďalej iba správu s vyšším dodatočným ID, pričom pri správe s nižším dodatočným ID posielame (na základe inštrukcií algoritmu KOLO()) potvrďujúce ack správy.

4.4 Spasívňovanie hrán a threadov

Po malých krokoch (aby sme sa vyhli prípadným problémom) a objasnení ďalších vecí postupne ukážeme korektnosť vyššie popísaného algoritmu.

Lema 4.4.1. *Ak je v sieti bez topologickej znalosti v danej fáze iba jeden iniciátor s nejakou svojou vedomosťou známych trojíc, pričom existuje vzhľadom na neho aspoň k doteraz neprejdených portov, až do úplného spasívnenia niektorého z jeho threadov sa v jednom kole aspoň jedna jeho orientovaná hrana (vrchol) ležiaca na niektorom z threadov stane pasívnou, t. j. doručí sa jej potvrďujúca ack správa.*

Dôkaz. Dodatok "pričom existuje vzhľadom na neho aspoň k doteraz neprejdených portov" nám zaručuje to, že môžeme v danej fáze vytvoriť k threadov ku k neznámym vrcholom. Inak je dôkaz tejto lemy veľmi podobný ako dôkaz lemy 3.4.1, preto nebudeme uvádzať jeho detaily, iba si ukážeme jeho koncept.

Na začiatku každého (i-teho) podkola si vynútime zaručenie "aktívnosti" (bola doručená aktivujúca správa) aspoň i hrán na i threadoch (tie "fungujú" ako v prípade topologickej znalosti - ich princíp je rovnaký, iba cieľ sa trochu zmenil). Preto v k-tom podkole máme zaktívnené všetky vlákna (ak sme medzitým neposlali ack správu), pričom v každom sa pošle aspoň jedna potvrďujúca ack správa. Žiadne aktivačné správy sa neposielajú, preto musí byť doručená aspoň jedna ack správa. Takéto vynútenie si k aktívnych threadov môžeme robiť dovtedy, keď nie je niektoré vlákno pasívne (už sa na ňom neposielajú žiadne aktivačné správy). □

Lema 4.4.2. *Ak je v sieti bez topologickej znalosti v danej fáze iba jeden iniciátor s vedomosťou, pričom existuje vzhľadom na neho aspoň k doteraz neprejdených portov, na maximálne $2m$ kôl sa mu podarí pretlačiť správu do niektorého (aspoň jedného) z jeho cieľových vrcholov.*

Dôkaz. Iniciátor si vyberie k portovo disjunktných threadov ku k neznámym vrcholom (reálne môže ísť o $1, \ldots, k$ rozličných vrcholov v sieti). Keďže jednotlivé vlákna sú portovo disjunktné a portov v algoritme je $2m$ (každá hrana sa skladá z dvoch portov), môžu tieto thready viesť iba po maximálne $2m$ portoch. Na základe vyššie spomenutej lemy vieme, že až do pretlačenia správy do niektorého z cieľových vrcholov sa spasívni v každom kole aspoň jeden vrchol (port). Keďže zároveň sa popri spasívňovaní doručujú v každom kole v každom threade aj aktivujúce správy (nie však od pasívnych vrcholov), postupne sa v niektorom z threadov odošle aktivujúca správa až do cieľového vrchola. □

Teraz sa už poďme pozrieť na reálnu situáciu, keď je v jednej fáze viacero iniciátorov s rovnakým alebo rozličným ID.

Lema 4.4.3. *V každom kole sa v prípade, že pre každé poradové číslo* $1, \ldots, k$ *existuje aspoň jedno aktívne vlákno s týmto poradovým číslom, spasívni aspoň jeden vrchol (hrana) nachádzajúca sa na niektorom aktívnom vlákne.*

Dôkaz. Nezaujíma nás, ktorému iniciátorovi dané vlákno s poradovým číslom i patrí, pretože v procedúre KOLO() túto skutočnosť (číslo iniciátora) vôbec neberieme do úvahy. Na úplnom začiatku ľubovoľnej procedúry KOLO(), pričom táto bola spustená za splnenia horeuvedených podmienok, spojíme šetky cesty s pevným poradovým číslom j - $P_{i_1 j}, P_{i_2 j}, \ldots, P_{i_l j}$ patriace iniciátorom $i_1, i_2, \ldots, i_l$ - do jedného nového vlákna (nie nutne spojitého) P_j, pričom za jeho inciátora zvolíme niektorý z vrcholov siete (iniciátora "dodáme" iba formálne, v skutočnosti tam vôbec nemusí figurovať). Všetky aktívne hrany patriace do jednotlivých threadov $P_{i_1 j}, P_{i_2 j}, \ldots, P_{i_l j}$ vyhlásime za aktívne aj vo vlákne P_j.

V prípade nespojitosti, t. j. $(u, v) \in P_j, (v, s) \in P_j$, ale $(u, v) \notin P_j$, uvažujeme o hrane (u, v) ako o spasívnenej hrane (je nám jedno, že je tam skok, pre jednotlivé vrcholy vlákna P_j je to informovaná hrana, po ktorej už nemusia pretláčať v danej fáze správu). V prípade nespojitosti iniciátora daného vlákna (t.j. neleží na žiadnej aktívnej hrane vlákna) uvažujeme podobne - hrana patriaca iniciátorovi je už spasívnená a nemusíme sa ňou nijako extra zaoberať. Keďže sme už vyššie spomínali prípady kolízií, tieto môžeme vylúčiť z uvažovaných možností. Prípad, že z jedného vrchola u môže vychádzať aj viacero hrán patriach do threadu j tiež nemusíme nijako obzvlášť riešiť. Takúto situáciu sme totiž (po teoretickej stránke) nikdy nevylúčili, preto sa o nej nebudem zmieňovať ani v tomto dôkaze.

Horeuvedené "spájanie" spravíme pre ľubovoľné poradové číslo $j = 1, \ldots, k$. Pre ľubovoľné vlákno P_j sme si mohli vybrať iniciátora spomedzi vrcholov celého grafu. Vďaka tejto možnosti vyberieme pre každý thread P_j ($j \in \{1, \ldots, k\}$) jedného a toho istého iniciátora u. Poznamenajme ešte, že existuje aspoň jedno aktívne vlákno pre každé poradové číslo, teda každé vlákno P_j ($j = 1, \ldots, k$) je stále aktívne. Dostali sme sa však do prípadu rozoberaného v leme 4.4.1. V sieti bez topologickej znalosti totiž existuje iba jeden iniciátor, pričom existuje aspoň k doteraz neprejdených portov (aspoň k aktívnych hrán v threadoch) a žiadny z jeho threadov ešte nie je pasívny. Teda v danom kole sa spasívni aspoň jedna hrana nachádzajúca sa na niektorom jeho (minimálne až doteraz) aktívnom threade. □

Ako vieme z vyššie spomenutej lemy 4.4.2, v sieti bez topologickej znalosti s jedným iniciátorom sa na maximálne $2m$ kôl podarí pretlačiť správu do niektorého z jeho cieľových vrcholov. Na základe riešenia kolízií vieme, že na začiatku každej fázy môže existovať maximálne n iniciátorov. Ak by "pretláčal" každý inciátor svoje správy samostatne, na najviac $2mn$ kôl by všetci "pretlačili" správu do aspoň jedného svojho cieľového vrcholu. My však nepotrebujeme, aby svoju správu pretlačili všetci. Nám stačí zaručiť, aby ju do aspoň jedného cieľového vrchola pretlačil niektorý iniciátor s najvyšším ID. Vďaka tejto požiadavke môžeme opomenúť požiadavku sériovosti (ktorú nevieme zaručiť, keďže jednotlivé vrcholy by nemali ako poznať, kedy sa nachádzajú vo "svojom určenom časovom úseku").

Lema 4.4.4. *Za jednu fázu trvajúcu $2mn$ kôl pretlačí aspoň jeden iniciátor danej fázy s najvyšším ID do niektorého svojho cieľového vrchola aspoň jednu správu.*

Dôkaz. Uvažujme iniciátora u s najvyšším ID, ktorý má navyše spomedzi všetkých iniciačných vrcholov s týmto ID aj najvyššie dodatočné ID. To znamená, že všetky vlákna idúce z tohto iniciátora "skončia" vo svojich cieľových vrcholoch, pretože v prípade kolízie s ďalším vláknom iného iniciátora sa ich cesta nemôže zatarasiť (pretože odosielajúci vrchol ich uprednostní).

Cieľom "nepriateľa" je pokúšať sa čo najdlhšie pozdržať správy na vláknach iniciátora u a ak to bude možné, ani ich nedoručiť do dopredu vybraných cieľových vrcholov. Kým sa to nepriateľovi darí, máme však zaručene splnenú podmienku predchádzajúcej lemy hovoriacu o aktívnosti aspoň jedného vlákna s poradovým číslom rovným j pre všetky $j = 1, \ldots, k$. Teda v každom kole sa spasívni nejaká hrana patriaca threadu P_{ij} pre i - niektorý z iniciátorov a $j \in \{1, \ldots, k\}$. Poďme sa pozrieť, koľko spasívnených hrán môžeme teoreticky mať dovtedy, kým nebudeme musieť spasívniť aj niektorý z cieľových vrcholov iniciátora u s najvyšším ID:

- $2m\,(n-1)$ portov prislúchajúcich ostatným iniciátorom (okrem u)
- $2m - k$ portov prislúchajúcich iniciátorovi u (potenciálne všetky porty siete okrem k portov vedúcich k cieľovým vrcholom)

Dokopy teda môžeme spasívniť $2mn - k$ portov dovtedy, kým nebudeme musieť spasívniť niektorý z cieľových vrcholov iniciátora u. My však máme k dispozícii

$2mn$ kôl, teda v maximálne $(2mn - k + 1)$-tom kole musíme doručiť správu aj k niektorému z cieľových vrcholov iniciátora u. □

Takto pretláčame správy s najvyšším ID do nových, doteraz "neznámych" vrcholov a po maximálne $2m$ fázach môžeme s istotou tvrdiť, že sme s týmto ID oboznámili všetky vrcholy (pretože portov je $2m$, aj neznámych vrcholov ležiacich na týchto portoch je $2m$). Toto môžeme vysloviť a dokázať aj na základe nasledujúceho tvrdenia.

Tvrdenie 4.4.1. *Ak má niektorý iniciátor u danej fázy g správu obsahujúcu najvyššie ID aj najvyššie dodatočné ID, jeho potomkovia v ďalších fázach budú v prípade kolízie s potomkom iného iniciátora v fázy g pokračovať vo svojej ceste, t. j. kolízny vrchol zastaví cestu tohto potomka iniciátora v, kým potomok iniciátora u bude pokračovať vo svojej ceste.*

Dôkaz. Keďže nové dodatočné ID vytvoríme na základe starého tak, že na koniec starého dodatočného ID dodáme číslo prislúchajúceho threadu (číslo tohto threadu má v dodatočnom ID najnižšiu váhu), potomkovia iniciátora u budú vždy disponovať vyššími číslami ako potomkovia v. □

4.5 Algoritmus a jeho zložitosť

Horeuvedené dokázané tvrdenie 4.4.1 nás uisťuje o tom, že iniciátor s najvyšším počiatočným ID postupne cez svojich nadväzujúcich potomkov oboznámi všetky vrcholy siete o svojom ID a bude mu to trvať maximálne $2m$ fáz. Tvrdiť to môžeme na základe toho, že každý potomok v ľubovoľnej fáze má oproti svojmu predkovi o jeden "známy vrchol" viac (o nejaký vrchol ležiaci na doteraz neznámom porte), teda v l-tej fáze algoritmu poznajú všetci iniciátori l známych vrcholov (prislúchajúcich l portom). Keďže iniciátori sú navzájom nezávislí (nijako nespolupracujú, ani si nevymieňajú svoje informácie), ten s najvyšším ID a dodatočným ID pretlačí prostredníctvom niektorého zo svojich potomkov informáciu o tomto ID do všetkých ostatných zostávajúcich vrcholov (neznámych vrcholov ležiacich na neznámych portoch).

Veta 4.5.1. *V ľubovoľnom hranovo k-súvislom grafe bez zmyslu pre orientáciu vieme v simple threshold modeli zvoliť šéfa v čase $O(2^k m^2 n)$, kde m je počet hrán daného grafu a n je počet vrcholov tohto grafu.*

Dôkaz. Korektnosť algoritmu sme ukázali v postupných krokoch. O časovej zložitosti sme tiež hovorili pri postupnom vysvetľovaní algoritmu a môžeme si ju zapísať aj nasledovne (čím vlastne ukážeme aj hodnoty pre synchronizáciu jednotlivých krokov):

- celý algoritmus: oboznámenie všetkých vrcholov s najvyšším začiatočným ID patriacim jednému z iniciátorov algoritmu obsahuje $2m$ fáz
- jedna fáza: cieľom jednej fázy je "prenos" informácie o najvyššom ID k aspoň jednému "doteraz neznámemu" vrcholu - potrebujeme na to $2mn$ kôl
- jedno kolo: v jednom kole spasívnime aspoň jednu aktívnu hranu na niektorom z threadov niektorého iniciátora - procedúra KOLO() sa oproti predchádzajúcemu prípadu (s topologickou znalosťou) nezmenila a jej zložitosť je 2^k

□

Záver

Cieľom práce bolo navrhnúť algoritmus pre voľbu šéfa v jednoduchom prahovom modeli pre rozličné topológie, prípadne aj pre ľubovoľný hranovo k-súvislý graf. Cieľ sa čiastočne podarilo splniť, keďže som navrhol a dokázal algoritmus pre hranovo k-súvislý graf. Okrem toho som sa však chcel zamerať na špeciálne topológie, čo sa mi nepodarilo celkom podľa predstáv. V navrhnutých riešeniach som totiž vždy objavil chybu, ktorá zmarila moje úsilie. Do budúcnosti teda zostáva otvorený tento problém.

Podarilo sa mi však viac ponoriť do sveta chýb v sieťach a ich možných riešení. Táto oblasť je ešte stále dostatočne otvorená a nachádza sa tu ešte veľa nedoriešených problémov. Modelovanie sietí s chybami je veľmi zaujímavé a v tejto oblasti už napriek mnohým problémom existuje množstvo výsledkov. Časť z nich som poskytol v jednej kapitole tejto diplomovej práce, iné som pre rôzne dôvody neuviedol. Do budúcnosti by však bolo zaujímavé urobiť prehľad najzaujímavejších výsledkov z danej oblasti. Otvorené zostávajú aj dolné odhady počtu odoslaných správ pre riešenia problémov v jednotlivých navrhovaných modelov. Aj táto časť tejto širokej oblasti je otvorená pre potenciálnych uchádzačov.

Navrhnuté algoritmy nadväzujú na publikované výsledky, obsahujú však aj nový pohľad na celú vec. Svojou komplexnosťou ponúkajú ďalšie možnosti svojho využitia. Ich škálovaním by sa dalo prísť na ďalšie výsledky, to už ale nebolo cieľom tejto práce.

Literatúra

[AGHK96] R. Ahlswede, L. Gargano, H. S. Haroutunian, and L. H. Kchachatrian. Fault-tolerant minimum broadcast networks. *Networks 27*, 1996.

[BDP97] P. Berman, D. Diks, and A. Pelc. Reliable broadcasting in logarithmic time with byzantine link failures. *Journal of Algorithms*, 22 (2):199–211, 1997.

[DKKS] S. Dobrev, Rastislav Královič, Richard Královič, and N. Santoro. On fractional dynamic faults with threshold.

[DKP] S. Dobrev, R. Královič, and D. Pardubská. Leader election in extremely unreliable rings and complete networks.

[Dob04] S. Dobrev. Computing input multiplicity in anonymous synchronous networks with dynamic faults. *Journal of Discrete Algorithms 2*, pages 425–438, 2004.

[FLP85] M. J. Fisher, N. A. Lync, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM 32 (2)*, 1985.

[Fra82] R. Franklin. On an improved algorithm for decentralized extrema finding in circular configurations of processors. *Commun. AC*, 25(5):336–337, 1982.

[KKR03] R. Královič, R. Královič, and P. Ružička. Broadcasting with many faulty links. *Proc. 10th Colloquium on Structural Information and Communication complexity (SIROCCO '03)*, pages 211–222, 2003.