Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Voľba šéfa s nespoľahlivými správami

Diplomová práca

2018
Martin Červeň, bc.

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Voľba šéfa s nespoľahlivými správami

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	prof. RNDr. Rastislav Královič PhD.

Bratislava, 2018
Martin Červeň, bc.

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Červeň
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Voľba šéfa s nespoľahlivými správami
Leader election with unreliable messages

Anotácia:

Cieľ: Cieľom práce je preskúmať riešiteľnosť a zložitosť komunikačných problémov (voľba šéfa) v synchrónnom modeli s možnosťou straty správ.

Vedúci: prof. RNDr. Rastislav Kráľovič, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 07.11.2016

Dátum schválenia: 14.12.2016 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Rád by som sa poďakoval môjmu školiteľovi prof. RNDr. Rastislavovi Královičovi PhD. za jeho pomoc a podporu, za jeho čas a vedomosti, s ktorými sa so mnou počas písania práce podelil.

Abstrakt

V tejto diplomovej práci sme sa zamerali na problém voľby šéfa na 2D torusoch v jednoduchom prahovom modeli. V jednoduchom prahovom modeli máme zaručené doručenie iba jednej správy, za predpokladu, že ich pošleme dostatočné množstvo. Navrhli algoritmus, ktorého zložitosť je $O(n^2)$ aj pri znalosti orientácie, aj bez nej. Okrem tohto algoritmu uvádzame aj iný algoritmus, ktorý má síce horšiu zložitosť, ale ukazuje problém, ktorý prináša strata orientácie v 2D torusoch.

Kľúčové slová: voľba šéfa, 2D torus, jednoduchý prahový model

Abstract

In this master thesis we focused on the leader election problen on 2D tori in the simple threshold model. In the simple threshold model, there is guaranteed delivery only of a one message provided, when we send sufficient quantity of them. We designed an algorithm, that has $O(n^2)$ complexity with sense of direction and also without it. Besides this algorithm, we introduce another algorithm, that has worse complexity, but it shows the problem caused by loss of orientation in 2D tori.

Keywords: leader election, 2D torus, simple threshold model

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Vo svete sa čím ďalej tým viac používajú distribuované systémy skladajúce sa z veľkého množstva počítačov a procesorov, ktoré sú navzájom poprepájané. Základným problémom v takýchto sieťach je problém voľby šéfa, ktorému sa budeme venovať aj v tejto diplomovej práci, lebo bez toho, aby bol zvolený šéf, nie je možné riešiť niektoré ďalšie problémy alebo by nebol využitý celý potenciál všetkých počítačov.

Keďže v reálnom svete môže dochádzať ku chybám a výpadkom pri komunikácií, tak sú vytvárané rôzne modely, na ktorých je voľba šéfa skúmaná, pričom niektoré sa približujú k realite viac a iné menej. Pre našu prácu sme si vybrali na skúmanie jednoduchý prahový model. Ide o synchrónny model fungujúci nad k-súvislými grafmi, v ktorom keď pošleme v celom modeli aspoň k správ, tak je garantované doručenie iba jednej. V prípade, že pošleme celkovo menej ako k správ v jednom tiku globálnych hodín, tak sa môžu stratiť všetky správy.

Problém voľby šéfa bol v tomto modeli skúmaný nad viacerými topológiami sietí, konkrétne na kruhoch, úplných grafoch a bol vymyslený aj všeobecný algoritmus pre akýkoľvek k-súvislý graf. V našej práci skúmame jednoduchý prahový model na topológii 2D torusu. Ide o 4-súvislý graf, takže budeme naše algoritmy konfrontovať so všeobecným algoritmom pre k-súvislé grafy a jeho zložitosťou. Našim cieľom je navrhnúť algoritmus s lepšou zložitosťou, akú má všeobecný algoritmus pre k-súvislé grafy.

Najprv sa pozrieme na problém voľby šéfa vo všeobecnosti - jeho neformálnu aj formálnu definíciu a dôvod, prečo sa tento problém skúma. Následne uvedieme niekoľko dosiahnutých výsledkov a aj jednotlivé princípy, ktorými boli dosiahnuté, aby sa čitateľ lepšie oboznámil ako s prácou v jednoduchom prahovom modeli, tak aj s prístupmi voľby šéfa na 2D toruse. V ďalšej časti uvádzame jeden zvolený prístup, pomocou ktorého sa nám nepodarilo dosiahnuť lepšiu zložitosť, ako má všeobecný algoritmus a v poslednej kapitole uvádzame algoritmus, v ktorom sme zvolili iný prístup k voľbe šéfa, ktorý pomohol a jeho zložitosť je lepšia.

Kapitola 1

Definície

V tejto kapitole definujeme základné pojmy, ktoré budeme v práci využívať a tiež si definujeme model, v ktorom budeme skúmať možnosti voľby šefa.

1.1 Voľba šéfa

Samotný problém voľby šéfa bol sformulovaný v roku 1971 A. R. I. Smithom. [17] Úlohou pri voľbe šéfa je navrhnúť distribuovaný algoritmus, ktorý pre danú sieť procesorov začínajúcich v rovnakom stave dokáže zvoliť šéfa. Procesory sú nezávislé entity bez zdieľanej pamäte, ktoré medzi sebou komunikujú posielaním správ. Výsledkom tohto algoritmu je konfigurácia, v ktorej sa práve jeden procesor nachádza v stave *šéf* pričom všetky ostatné procesory sa nachádzajú v stave *prehra*. Dôvodom voľby šéfa je, aby ďalšie procesy v sieti boli koordinované a bolo možné efektívnejšie využiť túto sieť procesorov. Zvolenie šéfa je dôležité pre ďalšie distribuované algoritmy, medzi ktoré patrí prehľadávanie grafu, tvorba minimálnej kostry, broadcast atď. [2]

Formálnu definíciu voľby šefa preberáme z knihy od Gerarda Tela [18]:

Definícia 1.1.1. Algoritmus na voľbu šéfa je algoritmus spĺňajúci nasledovné vlastnosti:

- Každý proces má lokálne rovnaký algoritmus.
- Algoritmus je decentralizovaný, t.j. výpočet môže byť inicializovaný ľubovoľnou neprázdnou podmnožinou procesov.
- Algoritmus dosiahne terminálnu konfiguráciu v každom výpočte a v každej terminálnej konfigurácii je práve jeden proces v stave šéf a všetky ostatné procesy sú v stave prehra.

Angluin dokázala [1], že v kruhu anonymných procesorov nie je možné deterministickým algoritmom zvoliť šéfa. Dôkaz založila na modeli, ktorý obsahoval štyri procesory,

z ktorých v rovnakom stave sa nachádzali protiľahlé procesory. Takúto symetriu nie je možné rozbiť a teda by došlo k nekonečnému výpočtu alebo chybnému výstupu.

Voľba šéfa sa najčastejšie rieši nad procesormi, ktoré sú rozlíšiteľné. Procesory sa vzájomne líšia pomocou pridelených identifikátorov, pričom každý identifikátor je unikátny. Pri procesoroch rozlíšených pomocou identifikátorov sa zvyčajne hľadá procesor s identifikátorom, ktorý je minimálny alebo maximálny. Takýto spôsob možno označiť ako decentralizovaný problém hľadania extrému. [5]

V prípade voľby šéfa na celulárnych automatoch symetriu možno rozbiť aj inými spôsobmi, medzi ktoré patrí napríklad priradenie bodu v súradnicovej sústave Z^d. [12]

1.2 Modely pre problém voľby šéfa

Voľba šéfa sa skúma na rôznych modeloch, ktoré sa líšia topológiou siete, informáciami, ktoré jednotlivé procesory majú na začiatku k dispozícii (procesor môže mať zmysel pre orientáciu, poznať veľkosť siete atď.) synchrónnosťou, spoľahlivosťou prenosu správ atď.

Výpočtové modely pre problém voľby šéfa sa líšia vo viacerých parametroch. Medzi hlavné oblasti rozdielov patria vlastnosti procesorov a vlastnosti liniek, ktoré túto sieť tvoria. [3]

Medzi vlastnosti procesorov patria:

- Identita procesora: Jednotlivé procesory majú pridelené identifikátory, ktoré sú najčastejšie prirodzené čísla. V rámci komunikačnej siete bývajú unikátne, ale v niektorých modeloch nemusia byť všetky unikátne a iba niektoré sú odlišné a tie rozbíjajú symetriu. Niekedy sa identifikátor vytvára priamo v procesore, napríklad z pripojeného hardvéru, úrovne nabitia batérie atď.

- Identita susedov: Vlastnosť určujúca, koľko informácie vedia jednotlivé procesory o svojich susedoch. Procesor môže vedieť o svojich susedoch, ale nemusí vedieť ich identifikátory a teda nie je možné priamo adresovať správy. V takomto prípade je adresácia vykonávaná na základe lokálnych názvov liniek.

- Informácie o topológii: Procesor už vopred má znalosť alebo vie na základe lokálnych informácií vypočítať počet procesorov v celej komunikačnej sieti, priemer grafu, vzájomnú polohu susedných procesorov, prípadne ďalšie vlastnosti topológie.

- Lokálna pamäť: lokálna pamäť môže byť obmedzená alebo môže byť potenciálne nekonečnej veľkosti.

Vlastnosti liniek:

- Spoľahlivosť: Jednotlivé linky môžu byť spoľahlivé alebo nespoľahlivé. Pri spoľahlivých linkách je každá odoslaná správa doručená a to práve raz a v nezmenenej podobe. Pri spoľahlivých linkách taktiež nedochádza k vytváraniu nových správ. Nespoľahlivosť liniek sa môže prejavovať tak, že jednotlivé správy nemusia byť doručené alebo sú doručené čiastočne alebo sa ich obsah mohol cestou modifikovať. V sieťach s nespoľahlivými linkami môže dôjsť aj k vytváraniu nových správ, ktoré neboli odoslané žiadnym z procesorov. Spoľahlivosť linky sa môže v čase meniť, ak ju tak zadefinujeme v našom modeli (napríklad prvých n správ prejde spoľahlivo a potom už prestane byť linka spoľahlivá).

- Vlastnosť FIFO: Táto vlastnosť určuje, či budú prijímané správy v poradí, v ktorom boli odoslané alebo môžu prísť aj v pomiešanom poradí.

- Kapacita: Linky môžu byť schopné v jednom momente prenášať buď neobmedzené alebo obmedzené množstvo správ. Kapacita môže byť limitovaná pre jeden alebo aj oba smery.

- Smer: Linky môžu byť jednosmerné alebo obojsmerné. Po jednosmernej linke sa môžu dať posielať správy buď iba jedným smerom alebo obomi smermi, s tým obmedzením, že na nej môžu byť naraz iba správy idúce jedným smerom.

Okrem týchto dvoch oblastí veľkú rolu zohrávajú aj globálne vlastnosti celého modelu a vlastnosti správ. Medzi globálne vlastnosti modelu patrí synchronizácia. Podľa synchronizácie sa modely delia na asynchrónne a synchrónne. Pri synchrónnych modeloch sa synchronizácia dosahuje pomocou globálnych hodín, súčasťou synchronizačného signálu môže a nemusí byť počítadlo kôl. Základnou vlastnosťou správ je dĺžka správy, kde sa väčšinou berú do úvahy správy neobmedzenej dĺžky, ale niekedy bývajú skúmané modely, kde je dĺžka správ obmedzená nejakou konštantou. V prípade dĺžky správ býva skúmaný aj počet bitov na jednu správu, ktorý je potrebný, aby správa obsahovala všetky potrebné informácie. Keď vieme ohraničenie na počet bitov, tak môžeme skúmať, ako sa bude meniť zložitosť voľby šéfa, ak budeme limitovať veľkosť správ, prípadne ako veľmi môžeme limitovať veľkosť správ, aby zložitosť ostala nezmenená.

Pri vlastnostiach liniek sme spomínali spoľahlivosť. V praxi nevieme zaručiť absolútnu spoľahlivosť a preto má zmysel skúmať modely, kde nie je spoľahlivosť stopercentná. V praxi môže dochádzať aj ku kompletným výpadkom, kedy nie sú doručované žiadne správy. Ak by sa mohli strácať všetky správy, tak by sme nevedeli urobiť voľbu šéfa a preto sa v teoretických modeloch určujú rôzne hranice na možnosti straty správ, pričom niektoré modelujú reálnu skutočnosť lepšie a niektoré horšie.

Modely počítajúce so stratou správ môžeme rozdeliť do dvoch kategórií, na deterministické a pravdepodobnostné modely. V pravdepodobnostných modeloch dôjde ku chybe na linke s pravdepodobnosťou $p < 1$.

Z topologického hľadiska delíme chyby na statické a dynamické. Pri statických chybách už dopredu vieme, na ktorých linkách bude dochádzať k chybám, zatiaľčo pri dynamických musíme počítať s tým, že o mieste výskytu chýb dopredu nevieme, akurát musíme počítať s tým, že k nim môže dochádzať a nemôžeme sa spoliehať na to, že správa, ktorú v danom momente potrebujeme doručiť, tak aj bude naozaj doručená.

Do topologických informácií, ktoré môže mať dostupné procesor, patrí zmysel pre orientáciu. Zmysel pre orientáciu, ako ho definujú Flocchini, Mans a Santoro vo svojej práci [8], sa skladá z troch vlastností. Prvou z nich je vzťah medzi označeniami a schopnosťou rozlíšiť jednotlivé cesty. Každý vrchol má priradený unikátny identifikátor ku každej hrane, ktorá je s ním incidentná. Ak je zmysel pre orientáciu tvorený značením, tak potom na základe značiek jednotlivých hrán vieme rozlíšiť, či dve rôzne cesty končia v jednom vrchole alebo v rozličných vrcholoch. Druhá vlastnosť prirodzene nadväzuje na prvú a je ňou existencia vzťahu medzi označením hrán a lokálnymi pomenovaniami. Každý vrchol sa odkazuje na iné vrcholy pomocou lokálnych mien. V označenom grafe so zmyslom pre orientáciu existuje funkcia, ktorá ku postupnosti označení z vrcholu x do vrcholu y priradí lokálne mená používané vrcholom x na odkazovanie sa na vrchol y. Treťou vlastnosťou je schopnosť vrcholov prekladať konzistentne lokálne označenia susedných vrcholov.

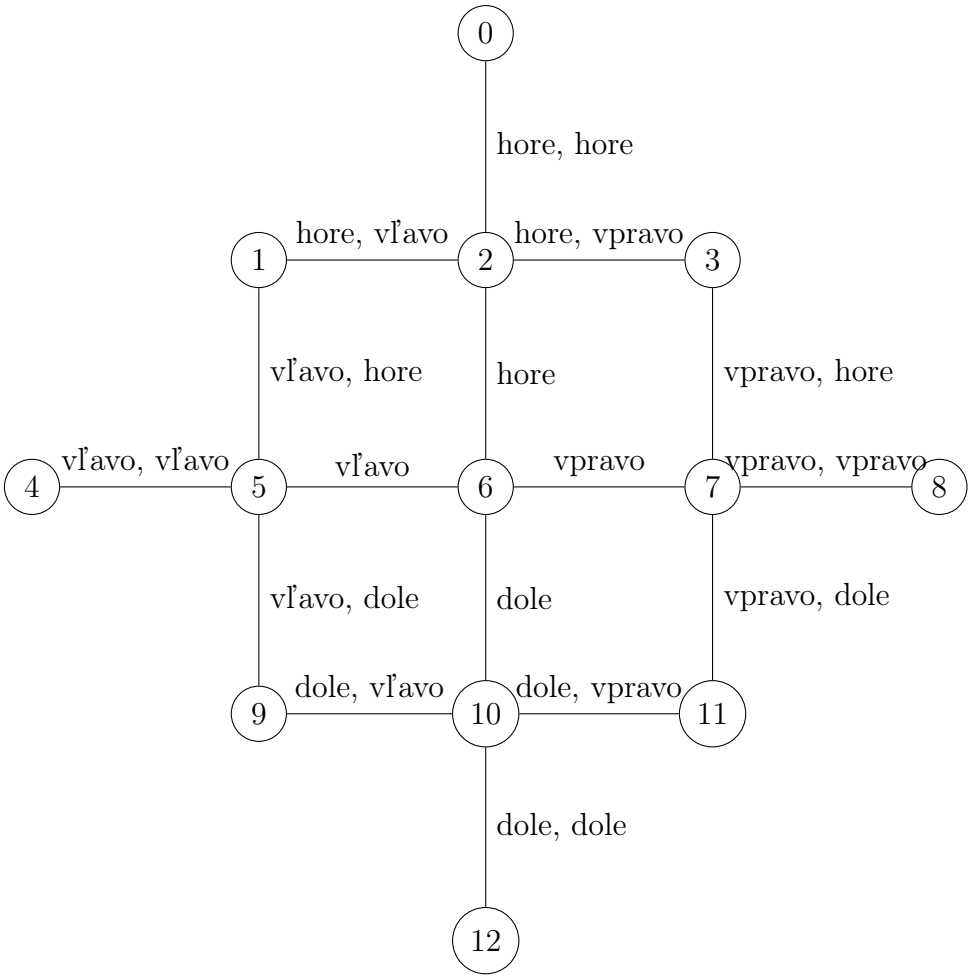
1.3 Model skúmaný v tejto práci

Výpočtový model, na ktorom skúmame algoritmy v tejto diplomovej práci, pozostáva z množiny identických procesorov, ktoré sú pospájané komunikačnou sieťou. Pamäť jednotlivých procesorov je nezdieľaná a každý má svoju vlastnú. Správy, ktoré si medzi sebou procesory posielajú, sa prenášajú prostredníctvom liniek.

Náš model je synchrónny. Všetci iniciátori sa zobudia simultánne a ostatné procesory sa zobúdzajú prvou prichádzajúcou správou. Na začiatku každého kola procesory príjmu procesory správy, potom urobia lokálny výpočet a na základe neho na konci kola odošlú správy (je taktiež prípustné, že nemusia odoslať žiadnu správu).

Vlastnosti procesorov v našom modeli:

- Identita procesora: Jednotlivé procesory majú pridelené unikátne identifikátory, ktoré sú vybrané z množiny prirodzených čísel.
- Identita susedov: Procesor nevie identifikátory svojich susedov. Správy môže adresovať iba pomocou lokálnych označení liniek.



Obr. 1.1: Zmysel pre orientáciu v časti 2D torusu.

- Topologické informácie: Skúmame model, v ktorom budú mať procesory zmysel pre orientáciu a taktiež model, kedy budú mať informáciu o počte susedov, ale nie o ich rozmiestnení. Zmysel pre orientáciu na 2D toruse z pohľadu procesora číslo 6 ilustrujeme na obrázku 1.1.

Vlastnosti liniek v našom modeli:

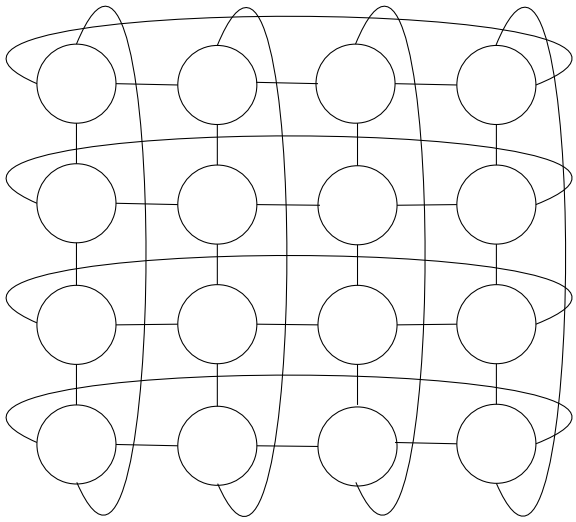
- Spoľahlivosť: Správy, ktoré sú doručené, sú doručené bez zmeny. Taktiež v našom modeli nevznikajú žiadne nové správy. Platí však, že nie všetky odoslané správy sú doručené (môže sa stať, že v niektorom kole nie je doručená žiadna správa).

- Kapacita: Jedna linka dokáže v jednom kole prenášať jedným smerom iba jednu správu.

- Vlastnosť FIFO: V našom modeli je táto vlastnosť irelevantná, lebo v jednom kole môže byť odoslaná iba jedna správa a tá je buď doručená na začiatku nasledujúceho kola alebo nie je doručená vôbec. Dá sa teda o našom modeli povedať aj že má vlastnosť FIFO, aj že nemá vlastnosť FIFO.

- Smer: Linky v našom modeli sú obojsmerné a je možné po nich posielať naraz správy oboma smermi.

Komunikačnú sieť nášho modelu predstavuje graf $G = (V, E)$. Vrcholy tohto grafu predstavujú procesory a hrany tohto grafu predstavujú linky. V našom prípade ide o neorientovaný graf, ktorý neobsahuje žiadne viacnásobné hrany ani slučky. Pojmy ako vrchol a procesor a tiež hrana a linka budeme v texte voľne zamieňať. V našom prípade je grafom modelujúcim sieť pospájaných procesorov 2D torus (cyklická mriežka). Ukážku štvorcového torusu veľkosti 4x4 uvádzame na obrázku 1.2.



Obr. 1.2: 2D torus veľkosti 4x4.

Jedna z najpodstatnejších vlastností, ktorú má nami skúmaný model, je tá, že ide o jednoduchý prahový model. V takomto modeli existuje hranica, ktorá určuje, koľko správ sa musí v celej sieti v jednom kole poslať, aby bolo garantované doručenie aspoň jednej správy. Pokiaľ nie je poslané dostatočné množstvo správ, tak môže dôjsť k tomu, že nebude doručená žiadna z nich.

Predtým, ako si formálne definujeme jednoduchý prahový model, potrebujeme si definovať hranovú súvislosť.

Definícia 1.3.1. Hranová súvislosť c(G) grafu G je minimálna veľkosť množiny hrán, ktoré keď odstránime z hrán grafu G, tak výsledný graf bude nesúvislý.

S formálnou definíciu hranovej súvislosti môžeme pokračovať definíciou jednoduchého prahového modelu. [6]

Definícia 1.3.2. Jednoduchý prahový model je taký model, v ktorom pre počty stratených správ platí nasledovné:

- Ak je m správ prenášaných po tom, ako všetky procesory na konci kola odoslali svoje správy, tak najviac $max\{c_G - 1, m - 1\}$ správ sa môže stratiť (c_G je hranová súvislosť komunikačnej topológie).

K strácaniu správ dochádza v každom kole nezávisle na ostatných kolách. Napríklad v prípade kruhovej siete dostávame, že ak je odoslaná iba jedna správa, tak môže dôjsť k jej strateniu. Ak sú poslané aspoň dve správy, tak potom aspoň jedna bude doručená. Pri úplných grafoch o n vrcholoch je nutné poslať aspoň $n - 1$ správ, aby bolo zaručené, že sa aspoň jedna doručí. V našom prípade, kedy sieť procesorov tvorí 2D torus, potrebujeme poslať aspoň štyri správy, aby bola aspoň jedna z nich doručená.

Z vyššie uvedených definícií vyplýva, že sa budeme zaoberať deterministickým a nie pravdepodobnostným modelom.

Kapitola 2

Doterajší stav problematiky

V tejto kapitole si povieme o výsledkoch, ktoré sa podarilo dosiahnuť v oblasti voľby šéfa v sieťach s nespoľahlivými správami. Budeme sa snažiť písať podrobnejšie, aby čitateľ dostal lepší náhľad do danej oblasti a bolo mu prirodzenejšie pracovať v jednoduchom prahovom modeli. Okrem výsledkov dosiahnutých v jednoduchom prahovom modeli budeme prezentovať aj výsledky dosiahnuté pri voľbe šéfa na 2D torusoch. Sumarizáciu výsledkov uvádzame v tabuľkách 2.1 a 2.2

2.1 Voľba šéfa v jednoduchom prahovom modeli

V jednoduchom prahovom modeli bola skúmaná voľba šéfa na troch topológiách. Konkrétne išlo o obojsmerné kruhy, úplné grafy a k-súvislé grafy. [4, 7] V prípade k-súvislých grafov bol skúmaný aj model so zmyslom pre orientáciu, aj model bez zmyslu pre orientáciu.

2.1.1 Voľba šéfa na kruhoch

Pri voľbe šéfa môžeme dostať rôznu zložitosť v prípade, že vynútime simultánne zobudenie všetkých iniciátorov a v prípade, že povolíme samovoľné prebúdzanie počas

kruh	prebudenie naraz	$O(n \log n)$
	svojvoľné prebudenie	$O(n^2)$
úplné grafy	so zmyslom pre orientáciu	$O(n^2 \log n)$
	bez zmyslu pre orientáciu	$O(n^3)$
k-súvislé grafy	so zmyslom pre orientáciu	$O(2^k n^2 m)$
	bez zmyslu pre orientáciu	$O(2^k n^2 m)$

Tabuľka 2.1: Sumarizácia výsledkov týkajúcich sa problému voľby šéfa (n je počet vrcholov, k je hranová súvislosť a m je celkový počet hrán)

voľba po jednotlivých kruhoch	so zmyslom pre orientáciu	$O(n)$
	bez zmyslu pre orientáciu	$O(n)$
voľba značením územia	so zmyslom pre orientáciu	$O(n)$
	bez zmyslu pre orientáciu	$O(n)$
s chybnými linkami	so zmyslom pre orientáciu	$O(n+f)$

Tabuľka 2.2: Sumarizácia výsledkov dosiahnutých pri voľbe šéfa na štvorcových 2D torusoch (veľkosti $\sqrt{n} \times \sqrt{n}$, f je počet chybných liniek).

behu algoritmu. K rozdielnej zložitosti dochádza aj pri voľbe šéfa v jednoduchom prahovom modeli na kruhu. Pri simultánnom prebudení k iniciátorov dostávame zložitosť $O(n \log k)$ krokov, aj keď vedomosť o počte procesorov na kruhu je neznáma. Každý ďalší procesor môže byť zobudený iba prichádzajúcou správou.

Samotný algoritmus je zložený z dvoch ideí. Prvou z nich je algoritmus na voľbu šéfa na kruhu, ktorého zložitosť je $O(n \log k)$ pri štandardnom posielaní a doručovaní správ. Druhou ideou je idea vlákien, ktorá bola použitá na broadcasting v jednoduchom prahovom modeli. [7]

Spomínaný algoritmus na voľbu šéfa je založený na Franklinovom algoritme [9] na voľbu šéfa na kruhu, ktorý bol upravený, ale aj po úprave si zachoval rovnakú asymptotickú zložitosť. Na začiatku sa iniciátori nachádzajú v stave kandidát a procesory, ktoré nie sú kandidáti, sú porazení a ich jediná aktívna účasť v protokole je preposielanie správ. Každý kandidát postupuje vo fázach, pričom začína vo fáze 0. V každej fáze sa snaží kandidát zajať oboch svojich susedov, ktorí sú v rovnakej fáze a v prípade, že sa mu to podarí, tak prejde do ďalšej fázy. Ak sa mu to nepodarí, tak iba čaká, kým nebude sám porazený nejakou inou správou.

Zo samotného fungovania môžeme vidieť, že procesor v najväčšej fáze s najlepším ID vždy vyhrá a teda nemôže dôjsť k zablokovaniu. Taktiež procesor môže prejsť do ďalšej fázy, iba vtedy ak obaja jeho aktívni susedia sú v rovnakej fáze, ale s horším ID, pričom daný procesor sa stane porazeným. Keďže sme na kruhu, tak jeden procesor môže byť obeťou pre maximálne dvoch kandidátov a teda najviac polovica kandidátov môže z jednej fázy postúpiť do ďalšej. Z toho vyplýva, že do fázy j sa môže dostať $k(2/3)^j$ kandidátov a že celkový počet fáz p potrebných na voľbu je $p \leq 1 + \log_{2/3} k$.

Zatiaľ sme nespomínali žiadne modifikácie, ktoré by umožnili tomuto algoritmu fungovať aj pri strácaní správ. Na to je použitá rovnaká metóda pretláčania správ pomocou vlákien ako v algoritme na broadcasting na kruhu. [6, 7] Popíšeme ju trochu podrobnejšie, lebo dobre ilustruje prístup k posielaniu správ v jednoduchom prahovom modeli, ktorý je potrebný pre fungovanie algoritmov v tomto modeli. Keď si zoberieme procesor p, ktorý chce posielať správy na oba smery, tak sa postupne vytvárajú reťaze následných informovaných procesorov ako vpravo, tak aj vľavo od procesora p, ktoré

voláme vlákna. Každý informovaný procesor sa snaží posunúť informáciu ďalej v zodpovedajúcom smere, až kým nedostane potvrdenie, že sa informáciu podarilo doručiť. Procesory vrámci jedného vlákna môžeme rozdeliť na aktívne, ktoré sa snažia rozšíriť informáciu a procesory pasívne, ktoré už dostali potvrdenie a teda vedia, že informácia sa dostala ďalej a už sa ju nesnažia šíriť. Komunikácia medzi procesormi je vykonávaná v kolách. Cieľom každého kola je, aby sa na jeho konci aspoň jeden vrchol stal pasívny. Jedno kolo pozostáva zo štyroch nasledovných krokov:

- Každý vrchol prepošle správu k jeho potenciálne neinformovanému susedovi.
- Každý aktívny vrchol v pravej časti prepošle správu k jeho potenciálne neinformovanému susedovi. Každý aktívny vrchol v ľavej časti, ktorý prijal správu v prvom kroku, odpovie na túto správu odoslaním potvrdzujúcej správy.
- Rovnaký krok ako predošlý, ale ľavá a pravá strana sú vymenené.
- Každý vrchol, ktorý dostal správu v krokoch 1-3, ktorá bola informačná a nie potvrdenie, odošle potvrdenie na danú informačnú správu.

Cieľom týchto krokov je zaručiť, aby počas nich bolo doručené aspoň jedno potvrdenie. Ak sa totiž počas prvých troch krokov nepodarilo doručiť žiadne potvrdenie, tak potom sa vo štvrtom kroku posielajú súčasne dve potvrdenia, z ktorých jedno musí byť doručené. Spojenie pretláčania správ so samotnou voľbou šéfa je urobené nasledovne: Procesor p chce poslať dve správy, jednu ľavému a jednu pravému susedovi. Na posielanie týchto správ iniciuje dve vlákna, ktoré postupujú podľa vyššie popísaného algoritmu. Postupne pri šírení vlákien sa vrcholy menia na pasívne a sú označované párom [fáza, ID], ktorý patrí iniciátorovi. Toto sa deje, kým vlákno nenarazí na iného kandidáta v rovnakej fáze, ale s horším ID. V takom prípade sa vlákno otočí s informáciou o víťazstve a pokračuje.

V jednom momente môže byť celkovo veľa aktívnych vlákien, ale keďže sme vynútili simultánne zobudenie všetkých iniciátorov, tak potom sú jednotlivé kolá synchronizované medzi všetkými vláknami. Práve vďaka tomu ak v každom kroku platí nasledovné:[7]

- Je aspoň jeden aktívny vrchol v pravom vlákne a tiež aspoň jeden aktívny vrchol v ľavom vlákne.
- Najviac jedna správa je preposielaná jedným smerom po jednej hrane.
- Ak sú po jednej hrane posielané dve správy v opačných smeroch, tak sú rovnakého typu.

Tak potom je doručené aspoň jedno potvrdenie v niektorom vlákne počas tohto kola. Na to, že platí prvá podmienka, sa stačí pozrieť na pravé a ľavé vlákno s najvyšším párom [fáza, ID]. Tieto vlákna nikdy nezaniknú, takže každé má aspoň jeden aktívny vrchol. Druhú podmienku je jednoduché splniť, v prípade, že by po jednej hrane malo byť poslaných viac správ, tak sa pošle iba tá s najväčšou dvojicou [fáza, ID]. Zvyšné vlákna by aj tak neskôr zanikli. Posledná podmienka taktiež platí, lebo dve správy sú poslané po jednej hrane iba, keď sa vlákno otáča.

Na záver dochádza k jednej výnimke a to v momente, keď už ostal iba jeden kandidát a jeho pravé a ľavé vlákno sa stretnú. V tomto momente je identita šéfa známa jednému, prípadne dvom procesorom. Tieto vrcholy potom potrebujú počkať dostatočný počet krokov, ktorý je daný hornou hranicou času behu algoritmu, aby sa zabezpečilo, že už nie sú v obehu žiadne správy a následne môže vrchol, ktorý vie, kto je šéf, urobiť broadcast, ktorý bude obsahovať ID šéfa, na kruhu v lineárnom čase.

Použitím popísaného algoritmu nie je možné zvoliť viac ako jedného šéfa, lebo všetky procesory, ktoré sú vo vlákne, sú porazené a až keď sa stretnú dve vlákna jedného procesoru, tak je zvolený šéf.

2.1.2 Voľba šéfa na úplných grafoch

Úplné grafy sú také grafy, v ktorých je každý vrchol spojený hranou so všetkými ostatnými vrcholmi grafu. Ide teda o súvislý $(n-1)$-regulárny graf a teda v jednoduchom prahovom modeli potrebujeme poslať aspoň $(n-1)$ správ, aby sme mali garantované doručenie aspoň jednej. Voľba šéfa pri úplných grafoch sa dá realizovať pomocou upraveného algoritmu na broadcast v jednoduchom prahovom modeli. V podstate ide o robenie n paralelných broadcastov, s tým, že iba najlepší vrchol prežije a stane sa šéfom. Takto je k pôvodnému algoritmu pridaný multiplikatívny faktor veľkosti n. V prípade, že máme k dispozícii zmysel pre orientáciu, tak vieme voľbu šéfa realizovať v $O(n^3)$ a v prípade neorientovaného grafu je to možné v $O(n^4)$. [6]

2.1.3 Voľba šéfa na k-súvislých grafoch

S kompletnou znalosťou topológie

V k-súvislých grafoch platí Mengerova veta, ktorá hovorí o tom, že medzi ľubovoľnými dvoma vrcholmi grafu existuje k ciest, ktoré sú navzájom hranovo disjunktné. Pre k-súvislý graf, v ktorom každý vrchol pozná úplnú topológiu, existuje algoritmus na voľbu šéfa v jednoduchom prahovom modeli, ktorého zložitosť je $O(2^k n^2 m)$.[4]

Tento algoritmus využíva práve spomínanú Mengerovu vetu. Technika pretláčania správ je podobná, ako pri algoritme na voľbu šéfa na kruhu, až na to, že sa využíva k vlákien, ktoré sú vedené práve po k nezávislých cestách na grafe. V každej fáze si vrchol vyberie iný vrchol, ktorému bude chcieť poslať správu a vyberie si na to

k hranovo disjunktných ciest. Cieľový vrchol, ktorý dostane správu obsahujúcu ID iniciátora, preberie jeho úlohu a vyberie si iný cieľový vrchol, ktorému sa snaží doručiť správu, do ktorej doplnil svoje ID, aby si nejaký ďalší vrchol nevybral ako cieľ niektorý z už navštívených a informovaných vrcholov.

V jednej fáze môže byť aktívnych aj viac iniciátorov ako jeden a preto aj keď vlákna jedného iniciátora idú po rôznych cestách, môže dochádzať k situácií, že po jednej hrane sa má posielať viac ako jedna správa. V takejto situácii sa vyberie správa s najlepším ID, tá je posielaná ďalej a ostatné vlákna zaniknú. V prípade, že si jeden vrchol ako svoj cieľ vybralo viacero iniciátorov, tak ten si podobne zo správ vyberie správu s najlepším ID a v ďalšej fáze sa stane iniciátorom s vybranou správou.

Pri postupnom šírení iniciátorov prestanú byť niektoré správy preposielané, či už z toho dôvodu, že sa stretnú s inými vláknami po ceste do cieľových vrcholov alebo priamo v cieľových vrcholoch. Zo všetkých správ ostane iba správa s najlepším ID pôvodného iniciátora a po maximálne $(n-1)$ fázach bude táto informácia rozšírená do všetkých ostatných vrcholov.

Pri posielaní správ sa využíva podobný princíp ako pri pretláčaní správ na kruhoch a označuje sa ako kolo. Na začiatku sa pošle k informačných správ, z ktorých musí byť jedna doručená. Potom sa postupne vyberajú všetky možné podmnožiny k liniek (od veľkosti 1 až po k), po ktorých nebudeme pretláčať správu. Vrchol, ktorý prijal informačnú správu na začiatku, bude posielať potvrdenie. Vynechaním jedného vrcholu chceme dosiahnuť, aby sa posielalo $k-1$ informačných správ ešte neinformovaným vrcholom. Keďže nevieme, ktorý vrchol bol na začiatku informovaný, musíme vyskúšať všetky možnosti, ktorý vrchol vynechať a neposielať mu informačnú správu. Keď nastane situácia, že vynecháme už informovaný vrchol, ktorý posiela potvrdenie, tak sa buď doručí potvrdenie, alebo bude informovaný ďalší vrchol. Ak sa doručujú iba informačné správy, tak pri každej veľkosti vynechanej podmnožiny vrcholov nastane situácia (pri každej veľkosti podmnožiny nastane práve raz), že už informované vrcholy posielajú potvrdenia a informačné správy sú posielané iba neinformovaným vrcholom. Na záver sú potom všetky vrcholy informované a naraz posielajú potvrdenia, z ktorých sa jedno musí doručiť. Ak sa doručilo nejaké potvrdenie skôr, tak nemusia byť všetci informovaní, ale vieme, že bol informovaný vrchol, od ktorého sme dostali potvrdenie a kolo splnilo svoju úlohu.

Ak zoberieme, že by bolo v jednej fáze n iniciátorov a správy každého by chceli ísť po všetkých m hranách grafu, tak po nm kolách už určite bola doručená správa iniciátora s najlepším ID do svojho cieľa. Celkovo teda potrebujeme 2^k krokov na kolo, kde pretláčame správy, nm kôl na jednu fázu a spolu $(n-1)$ fáz, z čoho dostávame časovú zložitosť $O(2^k n^2 m)$.

Bez znalosti topológie

V tomto prípade jednotlivé vrcholy nemajú znalosť kompletnej topológie, ale vedia iba, koľko majú susedov a vedia tiež rozlíšiť svoje jednotlivé porty. Rovnako ako v prípade so znalosťou kompletnej topológie, tak aj teraz sa budú využívať vlákna na šírenie informácie, ale nie je možné pre ne dopredu vybrať cesty, po ktorých sa tak bude diať. Postupne ako sa budú šíriť správy, tak sa bude rozširovať známe územie, lebo každý vrchol bude do správ pridávať svoje ID a port, cez ktorý danú správu posiela. Ak teda bude chcieť vrchol poslať správu ďalej, tak na základe informácií, ktoré táto správa obsahuje, vie vybrať k portov, cez ktoré ešte nešla a opäť sa známe územie rozšíri. Ak ešte neprejdený port nie je priamo port iniciátora, tak cesta k nemu je pridaná do správy, aby vrcholy zo známeho územia vedeli, kade majú správu posielať.

Čiže teraz sa nebudú posielať správy v jednej fáze jednému cieľovému vrcholu, ale potenciálne až k neznámym vrcholom, ktorí sa v nasledujúcej fáze stanú iniciátormi. Pokým ešte existuje k portov, cez ktoré daná správa nebola posielaná, tak môže existovať vrchol, ktorý zatiaľ nebol informovaný a existuje k portovo disjunktných ciest, po ktorých môžeme smerovať jednotlivé vlákna.

Keďže správy od jedného pôvodného iniciátora môže posielať viac iniciátorov v neskorších fázach, tak môže dochádzať k ich vzájomným kolíziám a preto bolo do algoritmu pridané aj dodatočné ID, ktoré sa postupne vytvára, ale nebudeme o ňom podrobnejšie písať.

V tomto prípade je potrebných $2m$ fáz na oboznámenie všetkých vrcholov o najlepšom ID, o informovaní aspoň jedného zatiaľ neznámeho vrcholu je potrebné $2mn$ kôl a jedno kolo aj bez znalosti topológie ostáva veľkosti 2^k krokov. Výsledná zložitosť voľby šéfa v ľubovoľnom k-súvislom grafe bez znalosti topológie je $O(2^k m^2 n)$. [4]

2.2 Voľba šéfa na 2D torusoch

2D torus je 4-súvislý graf a teda by sme na voľbu šéfa mohli využiť generický algoritmus na voľbu šéfa pre k-súvislé grafy. Z tabuliek 2.1 a 2.2 na začiatku tejto kapitoly však môžeme vidieť, že je pomerne veľký rozdiel medzi generickým algoritmom pre k-súvislé grafy a algoritmami na voľbu šéfa na 2D torusoch. Je pravda, že tie fungujú v modeloch, kde sa nestrácajú správy, ale mohol by tam byť priestor na zlepšenie. V nasledujúcom texte uvádzame výsledky dosiahnuté pri voľbe šéfa na 2D torusoch. Ide o výsledky dosiahnuté v modeloch bez straty správ a v jednom prípade ide o model, v ktorom sa nachádzajú chybné linky.

2.2.1 Voľba šéfa po jednotlivých kruhoch

Model, v ktorom funguje nasledujúci algoritmus, pozostáva zo siete n procesorov s nezdieľanou pamäťou, ktoré sú usporiadané v 2D toruse rozmerov $x \times y = n$, a ktoré sú označené unikátnymi identifikátormi od 0 po $n - 1$. Okrem identifikátora má každý procesor pridelenú aj pozíciu. Všetky linky sú obojsmerné. Výpadok šéfa sa môže udiať kedykoľvek a môže ho zaznamenať jeden alebo viac procesorov (v najhoršom prípade všetky ostatné procesory). Pod výpadkom šéfa sa v tomto prípade rozumie neschopnosť fungovať ako šéf, ale daný procesor dokáže preposielať správy v priebehu voľby nového šéfa.

Samotná voľba šéfa prebieha v štyroch fázach: [14]

- Prvá fáza: Vrchol, ktorý detegoval výpadok šéfa, o tom informuje vo svojom riadku.
- Druhá fáza: Vykoná sa voľba šéfa medzi kandidátmi na jednotlivých stĺpcoch, pričom výsledky sa pošlú v každom stĺpci do prvého riadku.
- Tretia fáza: V rámci prvého riadku sa urobí voľba šéfa z výsledkov volieb v stĺpcoch.
- Štvrtá fáza: Vrchol, ktorý vie o zvolenom šéfovi, informuje o tom všetky ostatné vrcholy.

Správu o výpadku šéfa posiela vrchol, ktorý ten výpadok detegoval, doprava a doľava. Vrchol, ktorý danú správu prijme, ju prepošle opačným smerom a prejde do druhej fázy, pričom ako svoje ID vyberie väčšie ID.

Druhá fáza prebieha smerom hore. Keď nejaký vrchol dostane správu zdola, tak porovná svoje ID s ID v správe a pošle správu hore s väčším z nich. Keď správa obíde celý stĺpec a vráti sa k iniciátorovi, tak je výsledok poslaný do prvého riadku.

V tretej fáze iniciuje najľavejší vrchol (so súradnicami 0,0) voľbu šéfa z výsledkov volieb na jednotlivých stĺpcoch. Táto voľba prebieha smerom doľava a keď najľavejší vrchol dostane správu s výsledkom, tak je fáza tri dokončená.

Posledná štvrtá fáza pozostáva z broadcastu, ktorým sú informované všetky vrcholy. Najprv sa urobí broadcast v prvom riadku, tento broadcast sa robí obomi smermi. Každý vrchol z jedného riadku po prijatí správy s výsledkom, urobí broadcast na stĺpci, pričom aj tento broadcast sa vykonáva naraz obomi smermi, aj hore, aj dole. Celkovo tento algoritmus potrebuje $O(n)$ správ a v prípade štvorcového torusu je jeho časová zložitosť $O(\sqrt{n})$.

V skutočnosti bol tento algoritmus prezentovaný spolu s tým, že dokázal fungovať aj pri nefunkčnosti jednej linky, bol však neskôr rozšírený, aby vedel fungovať aj pri výpadku viac liniek a preto sme vynechali časti spomínajúce riešenie nefunkčnosti linky.

2.2.2 Voľba šéfa s chybnými linkami

Ide o rozšírenie pôvodného algoritmu na voľbu šéfa na 2D toruse, aby zvládal koreknte zvoliť šéfa aj pri výpadku a nefunkčnosti viacerých liniek. Mohammed Al Refai zaviedol pravidlá na obchádzanie, ktoré sa použijú v prípade, že sa zistí nefunkčnosť niektorej linky. [15]

Ak je nefunkčná linka smerom hore, tak sa správa posiela najprv doprava, potom hore a nakoniec doľava k určenému príjemcovi. V prípade nefunkčnosti linky smerom doprava je obchádzka smerovaná hore, doprava a nakoniec smerom dole. Pokiaľ nie je funkčná linka smerom dole, tak sa využíva obchádzka doprava, dole a doľava. Na záver, ak nefunguje linka smerom doľava, tak obchádzka vedie hore, doľava a dole. V prípade detekcie výpadku ďalšej linky, ktorá sa nachádzala na trase obchádzky, sa začne robiť ďalšia obchádzka. Takto sa môžu jednotlivé obchádzky na seba reťaziť, až kým správa nedorazí do svojho cieľa.

Takto modifikovaný algoritmus má väčšiu časovú zložitosť práve o obchádzky, ktoré musia jednotlivé správy prekonať, keď narazia na nefunkčnú linku a tiež potrebuje viac správ, ktoré vykonajú obchádzky a teda časová zložitosť sa zvýši na $O(\sqrt{n} + f)$ a potrebný počet správ sa zvýši na $O(n + f)$ (f v oboch prípadoch znamená počet nefunkčných liniek).

2.2.3 Voľba šéfa značením územia

Ide o Petersonov algoritmus [13], v ktorom sa potupne zmenšuje počet aktívnych procesorov o konštantný faktor, aby bola dosiahnutá zložitosť $O(n)$.

Základ celého algoritmu je označovanie štvorcového územia, ktorého veľkosť jednej strany je d. Označovanie prebieha "označovacou" správou, ktorá toto územie obchádza. Najprv sa posiela do vzdialenosti d doprava, potom o d dole, následne d krát doľava a na záver d ráz hore. Vzdialenosť d sa postupne v jednotlivých fázach zväčšuje, konkrétne vzťahom $d = \alpha^i$, kde i je číslo fázy a $\alpha \approx 1.1795$.

Počas označovania môže správa naraziť na už označené územie iným procesorom v rovnakej fáze. Ak správa procesora x vo fáze i narazí na územie už označené procesorom y v rovnakej fáze, tak sa porovná ID v správe a ID uložené v označenom procesore. Pokiaľ správa obsahuje menšie ID, tak pokračuje v označovaní ďalej, ale pridá sa do nej informácia, že narazila na územie označené procesorom s väčším ID. V prípade, že ID v správe je väčšie ako ID už označeného územia, tak "označovacia" správa procesora x zanikne a po už označenom území procesora y sa pošle správa "videnéVäčším(x,i)".

Procesor x môže postúpiť do ďalšej fázy buď keď jeho "označovacia" správa nenarazila na iné označené územie, alebo keď narazila a vrátila sa k procesoru x s informáciou, že narazila na už označené územie, tak môže postúpiť do ďalšej fázy, iba ak bola prijatá aj správa "videnéVäčším".

Algoritmus týmto spôsobom pokračuje, kým d_i nepresiahne $\sqrt{n}$. V takom prípade procesor x nedostane svoju "označovaciu" správu zdola, ale zľava. Keď teda dostane svoju správu zľava, tak pošle správu smerom hore a čaká na jej príchod zdola. Ak sa aj tá vráti, tak potom vo všetkých ďalších fáza pošle správu smerom doprava a očakáva ju z ľavej strany a následne pošle správu smerom hore a očakáva ju zdola. Takýchto fáz je len konštantný počet, lebo iba konštantný počet kandidátov sa môže dostať do stavu, kedy ich správy obídu celý torus. Procesor, ktorý zostane aktívny po tomto konštantnom počte fáz, sa stane šéfom a oznámi to všetkým ostatným.

Petersenov algoritmus [13] funguje na 2D torusoch so znalosťou orientácie, ale bol modifikovaný Bernardom Mansom [11], aby vedel fungovať aj bez znalosti informácie s rovnakou zložitosťou $O(n)$. Lokálnu znalosť orientácie si vytvoria procesory na začiatku, pošlú svoju identitu svojim susedom a následne aj identitu všetkých svojich susedov. To stačí na vytváranie "zábradlia", ktoré je tvorené z ID procesorov, o ktorých vedia jednotliví susedia a hovorí o tom, na ktorú stranu má správa zatočiť, aby zatáčala vždy do jednej strany.

Kapitola 3

Algoritmus so znalosťou orientácie

V tejto kapitole najprv uvedieme náš algoritmus pre voľbu šéfa na 2D torusoch so znalosťou orientácie a potom dokážeme jeho správnosť a zložitosť.

3.1 Úvod

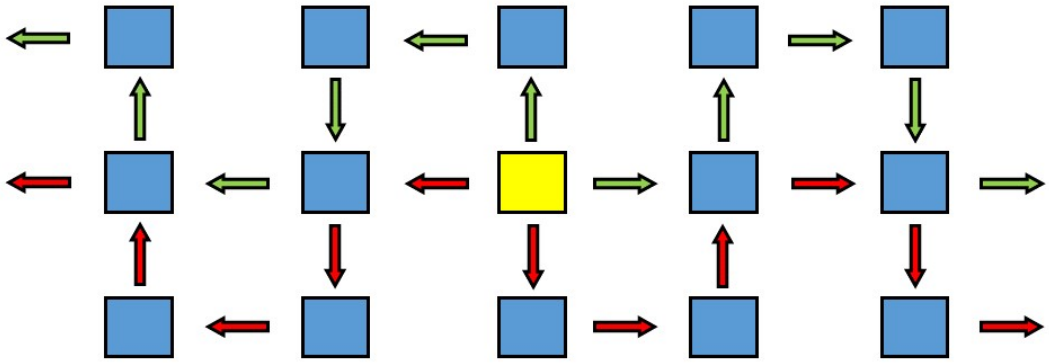
Náš algoritmus na voľbu šéfa vychádza z algoritmu na voľbu šéfa na 2D torusoch bez straty správ, v ktorom je torus rozdelený na kruhy vodorovne a tiež zvislo. Na každom z týchto kruhov prebieha voľba šéfa zvlášť. Najprv sa vo vodorovnom kruhu informuje o potrebe voľby nového šéfa, následne sa urobí voľba šéfa na jednotlivých zvislých kruhoch, potom sa urobí na vodorovnom kruhu voľba šéfa z výsledkov jednotlivých zvislých kruhov a nakoniec sa urobí broadcast, ktorý obsahuje informáciu o tom, kto je zvolený ako šéf.

Tým, že sa nachádzame v prostredí, kde môže dochádzať k strácaniu správ, tak sme na pretláčanie správ v našom algoritme použili podobný spôsob ako je v algoritme na voľbu šéfa na kruhoch v jednoduchom prahovom modeli, ale museli sme tento algoritmus rozšíriť, lebo potrebujeme v každom kole poslať minimálne dvojnásobok správ oproti kruhu, aby sme mali garantované, že sa aspoň jedna správa doručí.

3.2 Základná myšlienka pretláčania správ

Na komunikáciu medzi jednotlivými susedmi využívame aj susedné kruhy pôvodného torusu. Konkrétne ide o štyri cesty (ak sa na to pozeráme z pohľadu cyklickosti torusu, tak môžeme hovoriť o dvoch cestách), ako môžeme vidieť na obrázku 3.1. Týmito cestami sa budeme snažiť pretláčať správy pomocou vlákien podobne, ako to bolo pri voľbe šéfa na kruhoch. Ak sa stretnú v jednom procesore dve vlákna, ktoré boli iniciované opačnými smermi, tak je na danom kruhu zvolený šéf.

Na pretláčanie správ bude náš algoritmus pracovať v kolách pozostávajúcich zo 16



Obr. 3.1: Znázornenie ciest, po ktorých posielame správy po jednotlivých kruhoch. Iniciátor je znázornený žltou farbou.

fáz, pričom v každej z fáz sa posielajú iné kombinácie správ, t.j. ani v jednej z fáz sa neopakujú linky a smery, ktorými sa správy posielajú. Každá správa obsahuje tiež informáciu, v ktorej fáze a ktorým smerom je posielaná, aby sa prebúdzané procesory vedeli zapojiť do komunikácie v správnej fáze. Iniciátori sa prebúdzajú simultánne, takže tí vedia, kedy sa zapojiť do komunikácie na základe smeru, ktorým je správa posielaná.

Definícia 3.2.1. Jedno kolo algoritmu na pretláčanie správ v jednoduchom prahovom modeli v 2D toruse pozostáva z nasledovných fáz:

- Fáza 1: pošli informačnú správu všetkými smermi.
- Fázy 2-5: pošli informačnú správu na 3 smery, v jednotlivých fázach meň smery, ktorými posielaš dané správy, informovaný procesor z prvej fázy posiela potvrdzujúcu správu.
- Fázy 6-11: pošli informačnú správu na 2 smery, v jednotlivých fázach meň smery, ktorými posielaš dané správy, informované procesory z predošlých fáz posielajú potvrdzujúcu správu.
- Fázy 12-15: pošli jednu informačnú správu, v jednotlivých fázach meň smery, ktorými posielaš danú správu, informované procesory z predošlých fáz posielajú potvrdzujúcu správu.
- Fáza 16: všetky informované procesory posielajú potvrdzujúcu správu.

V prvej fáze bude doručená aspoň jedna správa. V druhej až piatej fáze sa budeme snažiť informovať ďalší procesor, ale keďže nevieme, ktorý bol informovaný v prvom kole, tak musíme meniť smery, ktorými posielame informačné správy, lebo inak by

nám inteligentný útočník doručoval stále iba tú jednu správu, ktorej adresátom je už informovaný procesor. V týchto štyroch fázach musí nastať situácia, že informačné správy sa posielajú trom ešte neinformovaným procesorom a už informovaný procesor posiela potvrdzujúcu správu. Ak sa doručí potvrdzujúca správa, tak je informovaný jeden nový vrchol, pričom o tom odosielateľ vie. Ak sa v spomínanej situácií doručí niektorá z informačných správ, tak potrebujeme pokračovať ďalšími fázami.

Vo fázach šesť až jedenásť posielame informačnú správu na dva smery, ktoré postupne meníme a predpokladáme, že už sú informované dva procesory (ak by neboli, tak muselo byť doručené potvrdenie), ktoré posielajú zvyšné dve potrebné správy. Opäť musí nastať situácia, v ktorej posielame správy dvom ešte neinformovaným procesorom a druhé dva už informované procesory posielajú potvrdzujúce správy. V tejto situácii sa buď doručí potvrdenie alebo je informovaný aj tretí procesor.

V priebehu fáz dvanásť až pätnásť je buď doručené potvrdenie od niektorého z informovaných procesorov alebo je informovaný posledný štvrtý procesor. Ak počas predošlých fáz nebolo doručené žiadne potvrdenie, tak v poslednej šestnástej fáze musí byť doručené aspoň jedno potvrdenie, lebo ho posielajú všetky štyri informované procesory a teda platí, že po šestnástich fázach je informovaný minimálne jeden nový procesor a bolo doručené aspoň jedno potvrdenie.

Veta 3.2.1. *Na konci každého kola, v ktorom posiela správy k iniciátorov, bude informovaný aspoň jeden nový procesor a bude doručené aspoň jedno potvrdenie.*

Dôkaz. Všetci iniciátori sa prebúdzajú naraz, takže majú zosynchronizované jednotlivé fázy v kole. Ostatné procesory sa prebúdzajú prichádzajúcou správou, ktorá obsahuje informáciu o fáze a smere, ktorým je posielaná, takže sa prebudený procesor vie správne zapojiť do komunikácie. V prvej fáze sa posielajú v každej inštancii štyri informačné správy, z ktorých jedna musí byť doručená a teda prvá časť vety triviálne platí.

Teraz si postupne rozoberieme jednotlivé prípady, kedy môže byť doručené potvrdenie. Pred fázou 2 sme v stave, že je informovaný jeden procesor. Vo fázach 2 až 5 tento procesor posiela potvrdenie. Počas týchto fáz sa tri razy posiela správa aj už informovanému vrcholu a teda môže sa stať, že bude práve tá správa doručená, ale vďaka meneniu príjemcov pre informačné správy nastane situácia, v ktorej sa posielajú informačné správy na tri smery, kde sú vo všetkých inštanciách iba neinformované procesory a už informovaný procesor posiela potvrdenie. Ak je doručené potvrdenie, tak veta platí.

Ak nie je doručené potvrdenie, tak musí byť doručená niektorá informačná správa a teda sú informované dva procesory. Vo fázach 6 až 11 je päť takých fáz, že sa posiela informačná správa aj tým smerom, kde v niektorej inštancii už je informovaný vrchol a jedna taká fáza, že sa informačné správy posielajú smermi, v ktorých nie je v žiadnej inštancii informovaný vrchol a už informované vrcholy posielajú potvrdenie. V tejto

fáze sa buď doručí potvrdenie a veta platí, alebo je doručená informačná správa.

Pokračujme teda ďalej v prípade, že sa stále nedoručilo žiadne potvrdenie. Máme už tri informované procesory, ktoré posielajú potvrdenia a okrem toho je posielaná ešte jedna informačná správa v každej inštancii. Vo fázach 12 až 15 sú tri fázy, kedy posielame informačnú správu smerom k niektorému už informovanému vrcholu v niektorej inštancii a teda informácia sa nešíri a nemusí byť doručené žiadne potvrdenie, ale je tam aj jedna fáza, kedy informačná správa smeruje smerom, v ktorom nie je informovaný vrchol v žiadnej inštancii a informované vrcholy posielajú potvrdenie. Ak je doručené potvrdenie, tak veta platí. V prípade, že nie je doručené potvrdenie, tak musí byť doručená informačná správa a teda v poslednej fáze sú informované štyri procesory (môžu to byť štyri procesory v jednej, ale aj v štyroch rôznych inštanciách) a všetky posielajú potvrdenia, z ktorých aspoň jedno musí byť doručené, čím sme dokázali, že veta platí. □

3.3 Voľba šéfa

Procesor, ktorý deteguje zlyhanie šéfa, iniciuje voľbu nového šéfa. Voľbu šéfa môže iniciovať jeden alebo viac procesorov (v najhoršom prípade $n - 1$).

Definícia 3.3.1. Voľba šéfa na 2D torusoch prebieha v nasledujúcich štyroch krokoch:

- Procesor informuje procesory na vodorovnom kruhu o potrebe voľby nového šéfa
- Tieto procesory spustia voľbu šéfa na zvislých kruhoch
- Urobí sa voľba šéfa na pôvodnom vodorovnom kruhu z výsledkov volieb na zvislých kruhoch
- Broadcastom sa informujú všetky procesory o novo zvolenom šéfovi

Ak by bola voľba šéfa na vodorovnom kruhu iniciovaná skôr ako bola dokončená voľba šéfa na všetkých zvislých kruhoch, tak procesor, ktorý dostane správu o voľbe šéfa z výsledkov, ale ešte nemá výsledok zo svojho zvislého kruhu, počká, kým ho dostane a následne pokračuje voľba šéfa aj na vodorovnom kruhu.

Voľba šéfa na jednotlivých kruhoch bude prebiehať iným spôsobom podľa toho, či povolíme jedného alebo aj viacerých iniciátorov. Dôvodom je to, že pri jednom iniciátorovi vieme urobiť voľbu šéfa na jednotlivých kruhoch efektívnejšie ako pri viacerých iniciátoroch.

V prípade jedného iniciátora voľba šéfa na zvislých kruhoch prebieha tak, že procesor, ktorý iniciuje voľbu šéfa na zvislom kruhu, pošle na oba smery po dve vlákna, ktoré postupne prechádzajú kruhom a snažia sa nájsť procesor s najmenším ID. Keď

sa vrátia k iniciátorovi, tak obsahujú toto najmenšie ID. Prípad jedného iniciátora popisujeme preto, lebo sa na ňom jednoduchšie vysvetľujú kolízie správ, ktoré nastanú pri voľbe šéfa na zvislých kruhoch. V prípade viacerých iniciátorov môže dochádzať ku kolíziám častejšie, ale spôsob ich riešenia bude veľmi podobný.

3.4 Kolízie správ

Ak voľbu šéfa iniciuje jeden procesor, tak v prvej časti nemusíme riešiť možnosť kolízie viacerých správ na jednej linke, lebo cesty, po ktorých sa posielajú jednotlivé správy, sú disjunktné. V druhej časti, kde prebieha voľba šéfa na jednotlivých zvislých kruhoch, môže nastať kolízia, lebo správy sa šíria aj po vedľajších kruhoch.

V takomto prípade procesor, ktorý prijal dve správy z rôznych kruhov naraz, ktoré majú byť posielané po jednej linke, skontroluje, ktorá správa obsahuje ako kandidáta procesor s menším ID, následne porovná lepšie ID so svojím a odošle jednu správu, ktorá bude obsahovať najlepšie ID spomedzi porovnávaných troch a tiež informáciu o tom, ktorými smermi má správa pokračovať. Po "kolíznej" linke prejde jedna správa, ktorá potom pokračuje dvomi smermi, ktoré zodpovedajú cestám pôvodných dvoch správ.

Následne môže nastať situácia, v ktorej budú v jednom kruhu kolovať správy s rôznymi ID, lebo vďaka kolízii sa vymenilo ID v správe z toho kruhu za lepšie. V takejto situácii môže v tomto kruhu doraziť prvá správa s lepším ID (obísť celý kruh), ktoré na danom kruhu vyhrá. V prípade, že prvá dorazí správa s horším ID, tak lepšie ID muselo pochádzať z druhého kruhu a horšie ID, ktoré dorazilo skôr, bude neskôr porazené v sumárnych voľbách. (Ak by bolo lepšie ID z aktuálneho kruhu, tak by pri kolízii vyhralo a horšie ID by sa nemalo ako dostať do daného kruhu.) Sumárne voľby sú voľby z výsledkov volieb na zvislých kruhoch.

Počas kolovania správ, ktoré obsahujú rôzne ID, na jednom zvislom kruhu môžu nastať dve možnosti, kde do jedného procesoru prídu správy s dvoma rôznymi ID, smer myslíme z pohľadu kruhu, lebo každým smerom putujú dve vlákna. Prvá možnosť je, že najprv príde správa s horším ID, ktorú on prepošle a až neskôr príde správa s lepším ID a túto tiež iba pošle ďalej po jej ceste. Vždy keď budeme spomínať, že správu prepošle, tak to znamená, že ID v správe porovná so svojím ID a odošle správu s lepším z tých dvoch. Druhá možnosť, v akom poradí prídu správy, je tá, že najprv príde správa s lepším ID a až neskôr príde správa s horším ID. V tomto prípade prvú správu pošle ďalej, ale zapamätá si ID, ktoré v nej poslal (ak bolo jeho ID lepšie, tak si to nutne nemusí pamätať). Keď príde druhá správa, ktorá obsahuje horšie ID ako to, ktoré bolo poslané v predošlej správe ďalej, tak v tejto správe zmení ID na to zapamätané. Ak by prišli správy naraz, tak je to v podstate iba upravený druhý prípad, pričom porovná

všetky tri ID (dve zo správ a svoje ID) a odošle dve správy s najlepším z nich.

Spájanie dvoch správ do jednej, keď sa posielajú po "kolíznej" linke nie je problém, lebo v celom systéme sú aspoň 4 správy, ktoré obsahujú najlepšie ID a teda minimálne tie budú v každej fáze kola posielané. Dokonca je pre nás výhodné, ak sa spájajú správy, lebo keď sa doručí správa, ktorá bola spojená z viacerých správ, tak sa informácia šíri rýchlejšie.

3.5 Počet správ potrebný na voľbu šéfa pri jednom iniciátorovi

V predošlých prácach, ktoré skúmali jednoduchý prahový model, sa skúmala časová zložitosť, my sme si vybrali zložitosť z pohľadu počtu správ. Pri analýze potrebného počtu správ budeme pracovať s 2D torusom veľkosti $X \times Y = N$. Najprv si rozoberieme algoritmus po častiach a potom výsledky spojíme, čím dostaneme celkový počet potrebných správ.

Lema 3.5.1. *Na informovanie o potrebe voľby šéfa po vodorovnom kruhu potrebujeme* $O(N)$ *správ.*

Dôkaz. Jedno kolo má 16 fáz a na jeho konci je informovaný aspoň jeden procesor s tým, že je doručené aj potvrdenie. V každej fáze posielame 4 správy. Spolu máme Y procesorov, ktoré je treba informovať o potrebe voľby šéfa. Medzi týmito procesormi vedú cesty dĺžky 1, ale aj dĺžky 3, ktoré musí správa prekonať, aby sme informovali ďalší procesor. Potrebný počet správ je teda nasledovný:

$$16 \cdot 4 \cdot 3 \cdot Y = O(N) \tag{3.1}$$

□

Lema 3.5.2. *Na voľbu šéfa na zvislých kruhoch potrebujeme* $O(N^2)$ *správ.*

Dôkaz. Voľba šéfa na každom zvislom kruhu prebieha rovnako ako informovanie o potrebe voľby šéfa a teda každý jeden kruh potrebuje $16 \cdot 4 \cdot 3 \cdot X$ správ. Týchto kruhov máme celkovo Y. Máme garantované doručenie iba jednej správy, tak sa nám v najhoršom prípade iba v jednom kruhu posunie informácia. Keďže máme $X \cdot Y$ procesorov, cez ktoré musí správa prejsť a v každom kole sa posiela správa v každom z Y kruhov, tak celkový počet správ potrebných na voľbu šéfa na zvislých kruhoch je:

$$(16 \cdot 4 \cdot 3 \cdot X \cdot Y) \cdot Y = O(N^2) \tag{3.2}$$

□

Veta 3.5.1. *Na voľbu šéfa v 2D toruse v jednoduchom prahovom modeli so znalosťou orientácie a jedným iniciátorom je treba $O(N^2)$ správ.*

Dôkaz. Z predchádzajúcich liem dostávame, že na informovanie po vodorovnom kruhu potrebujeme $O(N)$ správ, na voľbu šéfa na zvislých kruhoch potrebujeme $O(N^2)$ správ, voľba šéfa z výsledkov správ je tiež $O(N)$, lebo nám stačí, aby iniciátor poslal 4 vlákna, ktoré nájdu najlepšie ID z výsledkov volieb na zvislých kruhoch, na čo potrebujeme rovnaké množstvo správ ako v časti, kde informujeme o potrebe voľby šéfa. Na záver ešte potrebujeme urobiť broadcast, na ktorý potrebujeme spolu $O(N) + O(N^2)$ správ. Z predošlého dostávame:

$$O(N) + O(N^2) + O(N) + O(N) + O(N^2) = O(N^2) \tag{3.3}$$

□

3.6 Problém viacerých iniciátorov

Pri viacerých iniciátoroch potrebujeme ošetriť, aby jednotlivé inštancie, ktoré boli spustené viacerými iniciátormi, nespôsobili vzájomnou interakciou nekorektnosť celého algoritmu. Mohlo by to mať za následok, že sa nepodarí zvoliť šéfa alebo ich bude zvolených viac súčasne. Okrem toho chceme zastaviť nadbytočné správy, aby nešírili nepodstatnú alebo viackrát tú istú informáciu. Ak by sme to nerobili, tak nám môže počet správ výrazne narásť.

Pri prvotnom informovaní o potrebe voľby šéfa na vodorovných kruhoch stačí, aby ak procesor poslal jedným smerom informáciu o potrebe voľby šéfa, tak túto informáciu už neposielal po rovnakej linke druhý raz. Takto zaručíme, že po každej linke prejde správa práve raz a teda nebude dochádzať k opätovnému posielaniu rovnakej informácie.

V druhej a tretej časti musíme zmeniť spôsob, akým sa volí šéf. Už nestačí, aby cez kruh prešli štyri vlákna, ktoré postupne nájdu na kruhu najmenšie ID a doručia ho iniciátorovi voľby šéfa na kruhu, lebo môžeme mať na každom kruhu viac iniciátorov a teda by nám počet správ mohol narásť na $O(N^3)$. Ak by správy obsahovali okrem najlepšieho ID aj ID iniciátora (aby iniciátor vedel, že prijatá správa je jeho a už obišla celý kruh), tak by voľba šéfa na zvislých kruhoch prebehla korektne, len by prebehla potenciálne toľko krát, koľko je na kruhu iniciátorov. Preto bude v týchto častiach prebiehať voľba šéfa použitím algoritmu Hirschberga a Sinclaira, ktorý pracuje s počtom správ $O(N \log N)$. [10] V tomto algoritme sa iniciátor voľby šéfa na kruhu snaží zajať 2^k susedných procesorov na každej strane, keď sa mu to podarí, tak zvýši k o jedna a skúsi zajať väčšie územie, až kým nezajme celý kruh alebo sa nenájde kandidát, ktorý ho porazí.

Keďže pracujeme v prostredí, v ktorom sa môžu správy strácať, v tomto prostredí budeme potrebovať pri viacerých súbežne bežiacich voľbách N násobne viac správ (lebo všetci iniciátori sa snažia pretláčať správy naraz, ale iba jedna je doručená aj s potvrdením) a teda výsledný počet správ potrebný v druhej a tretej časti bude $O(N \cdot N \log N)$.

Pri takto zmenenej voľbe šéfa na jednotlivých kruhoch musíme zmeniť aj spôsob, akým sa riešia kolízie správ. Už nemôžeme v správach ľubovoľne nahrádzať ID, ale musíme ho zachovať a kolidujúce správy iba spojiť do jednej a potom po prechode linkou ich opäť rozdeliť, aby mohli pokračovať každá svojou cestou.

Lema 3.6.1. *Počet správ potrebný na informovanie procesorov na vodorovných kruhoch o potrebe voľby šéfa je* $O(N^2)$.

Dôkaz. V najhoršom prípade iniciuje voľbu šéfa všetkých N procesorov. V takomto prípade každý procesor iniciuje 4 vlákna, ktoré majú za úlohu pretláčať informáciu po vodorovnom kruhu. Každé takéto vlákno sa dostane iba ku susedovi iniciátora, lebo ten ďalej posiela svoje vlákno. Keďže máme N procesorov, ktoré treba informovať a v jednom kroku je garantované doručenie jednej správy, celkovo máme N liniek, po ktorých musia správy prejsť, tak celkový počet správ potrebných na informovanie o potrebe voľby šéfa je:

$$O(N \cdot N) = O(N^2). \tag{3.4}$$

□

Lema 3.6.2. *Na urobenie voľby šéfa na všetkých zvislých kruhoch pri viacerých iniciátoroch je potrebných* $O(N^2 \log N)$ *správ.*

Dôkaz. Algoritmus na voľbu šéfa na jednom kruhu má zložitosť $O(N \log N)$. V jednoduchom prahovom modeli potrebuje tento algoritmus X krát viac správ, lebo máme garantované doručenie iba jednej správy a v každom kroku sa všetky procesory snažia posielať správy. Tento algoritmus beží v Y inštanciách naraz a keďže v jednom kole je garantované doručenie aj s potvrdením iba jednej správy, tak dôjde k posunu iba v jednej inštancií na jednom zvislom kruhu. Z uvedeného dostávame:

$$X \cdot Y \cdot O(N \log N) = O(N^2 \log N) \tag{3.5}$$

□

Lema 3.6.3. *Na urobenie sumárnej voľby šéfa na vodorovných kruhoch pri viacerých iniciátoroch je potrebných* $O(N^2 \log N)$ *správ.*

Dôkaz. Analogický dôkaz ako dôkaz predošlej lémy. Akurát smer posielania správ je otočený. □

Veta 3.6.1. *Na voľbu šéfa v 2D toruse v jednoduchom prahovom modeli so znalosťou orientácie viacerými iniciátormi je treba* $O(N^2 \log N)$ *správ.*

Dôkaz. Na informovanie potrebujeme $O(N^2)$ správ, na voľby šéfa potrebujeme ako zvislo, tak aj vodorovne potrebujeme $O(N^2 \log N)$ správ. Na následný broadcast nám stačí $O(N^2)$ správ, lebo každý iniciátor už vie, ktorý procesor bol zvolený ako šéf a teda ide o rovnaký proces ako pri informovaní o potrebe voľby šéfa s tým rozdielom, že toto informovanie je najprv vykonané po vodorovných kruhoch a následne ešte aj po zvislých kruhoch. Zhrnutím čiastkových výsledkov dostávame:

$$O(N^2) + O(N^2 \log N) + O(N^2 \log N) + O(N^2) = O(N^2 \log N) \tag{3.6}$$

□

Kapitola 4

Algoritmus bez znalosti orientácie - horšia zložitosť

V predošlej kapitole sme rozoberali prípad, kedy jednotlivé procesory mali znalosť o orientácii, čo v prípade 2D torusu znamenalo, že vedeli, ktorý port vedie ku susedovi, ktorý je "hore", "dole", "vľavo" a ktorý "vpravo". Teraz budeme rozoberať prípad, kedy procesory túto vedomosť nemajú.

4.1 Úvod

Keďže náš algoritmus z predošlej kapitoly vo veľkej miere využíva znalosť orientácie pri posielaní správ, tak by sa mohlo zdať, že keď o túto znalosť prídeme, tak prestane fungovať alebo bude potrebovať veľké množstvo správ. Ukazuje sa však, že je možné s pomerne malým množstvom správ získať dostatočné informácie o lokálnom svete. Toto zisťovanie je potrebné robiť počas samotnej voľby šéfa. Ak by sme sa snažili na začiatku zistiť informácie o susedoch a zistiť si ich polohu voči ich susedom, tak by mohla nastať situácia, že by už v celom systéme neboli aspoň štyri správy a nie všetky procesory by získali túto znalosť. Pokiaľ to však spojíme so samotnou voľbou šéfa, tak vtedy tento problém nenastane a počet správ potrebný na voľbu šéfa narastie pri použití nami navrhnutého algoritmu, v každom kroku iba o konštantný počet, čiže zložitosť ostane zachovaná.

4.2 Zisťovanie informácie o susedoch a ich vzájomnej polohe

Na začiatku má každý procesor iba informáciu o tom, že má štyroch susedov, ale o ich vzájomnej polohe nevie nič a taktiež nevie, kto sú jeho susedia (nevie ich ID). Na správne fungovanie algoritmu potrebuje každý procesor vedieť, kto sú jeho susedia a

taktiež, kto sú ich susedia. Na základe tejto vedomosti si už vie každý procesor vytvoriť "mapu" lokálneho sveta a posielať správy potrebné na voľbu šéfa.

Správy, ktoré zisťujú susedov, budeme volať zisťovacie. Zisťovacie správy budú mať obmedzenú životnosť a to konkrétne na štyri. Procesor, ktorý chce zistiť informácie o lokálnom svete, pošle na všetky svoje porty zisťovaciu správu. Príjemcovia takejto správy sa ju snažia poslať ďalej so zníženou životnosťou o jedna po všetkých troch zvyšných portoch. Pokiaľ je životnosť prijatej správy rovná nule a procesor, ktorý ju prijal, nie je jej iniciátor, tak s takouto správou nič nerobí.

Správy sú pretláčané rovnako ako správy pri algoritme so znalosťou orientácie, teda v každom kole je 16 fáz a procesor podľa toho, ktorá je fáza a informácie v správe, vie, kedy má danú správu posielať ďalej a kedy má posielať potvrdenia o prijatí správy.

Cieľom zisťovacej správy je zozbierať ID procesorov, cez ktoré prechádza a vrátiť sa k pôvodnému odosielateľovi, teda odosielateľ cez jeden port odošle správu a tá sa s pozbieranými ID vráti cez jeho iný port. Nie je možné, aby sa vrátila cez rovnaký port, lebo žiaden procesor neposiela správu späť po príchodzom porte a taktiež životnosť správy je dostatočne malá a teda správa nedokáže urobiť slučku niekde inde, ako bezprostredne pri iniciátorovi. Vďaka obmedzenej životnosti správ, vetvy, ktoré nevedú naspäť k pôvodnému odosielateľovi, skončia. Keď sa vrátia zisťovacie správy, tak procesor vie, ktoré procesory s ním susedia a ako sú rozmiestnené.

```
1  spracujZisťovaciuSprávu(správa, port):
2    if správa.iniciátor == mojeID and správa.životnosť == 0:
3      využi informácie z prijatej správy a pokračuj vo voľbe šéfa
4      return
5    if správa not in známeSprávy: #ak to je nová správa
6      známeSprávy.add(správa) #ulož si ju
7      prícho dzíPort = port #port, cez ktorý prišla správa
8      cieľovéPorty = všetkyPorty - prícho dzíPort #ostatné porty označ ako
       nevyskúšané
9      správa.životnosť -= 1
10     správa.IDs.append(mojeID)
11   správyNaOdoslanie = zlepSprávyNaRovnakýPort(známeSprávy, potvrdenie) #
       spojí správy, ktoré majú ísť cez rovnaký port, pridá do správ
       potvrdenie o prijatí správy
12   send(správyNaOdoslanie, správyNaOdoslanie.cieľovéPorty) #pošle správy
       na porty, ktoré sa ešte nedoručili
13
14 spracujPotvrdenie(správa, port):
15   zapamätajPríjemcu(sprava.ID) #zapamätá si, komu daným portom vie doruči
       ť správu
16   známeSprávy.get(správa).cieľovéPorty -= port #cez port sa podarilo
       poslať správu
17   if známeSprávy.get(správa).cieľovéPorty == {}: #ak už nie je žiaden
       port kam poslať správu
```

```
18      známeSprávy.remove(správa)  #tak správu zabudni
```

Kód 4.1: Spracovanie zisťovacej správy a potvrdenia o prijatí zisťovacej správy.

4.3 Prepojenie zisťovania a voľby šéfa

Zisťovanie okolia prebieha úplne bezprostredne pred posielaním správ s voľbou šéfa a ani sa nečaká, kým všetky správy so zisťovaním okolia prídu nazad. Celkovo sa môže vrátiť najviac osem správ po štyroch kruhoch, pričom tieto správy tvoria dvojice, ktoré prejdú po rovnakých kruhoch, ale v opačnom smere.

Už keď sa vráti prvá zisťovacia správa, tak sa procesor rozhodne, ktorý procesor si vyberie ako príjemcu pre správu o voľbe šéfa. Konkrétne si vyberie tretí procesor v poradí z ID procesorov v prijatej zisťovacej správe. Tomuto procesoru pošle správu s voľbou šéfa, vie k nemu nasmerovať dve vlákna, jedno priamo a druhé po ceste zisťovacej správy. Následne potrebuje procesor zistiť, ktorý port vedie k procesoru naproti procesoru, ktorému sme už poslali správu o voľbe šéfa.

Pri prijatí druhej zisťovacej správy môže nastať viacero prípadov, pričom pri niektorých nemusí procesor vedieť rozhodnúť ako pokračovať a bude musieť počkať na ďalšiu zisťovaciu správu, ktorá mu dodá potrebné informácie. Možné prípady prijatia ďalšej zisťovacej správy sú:

- druhá správa obsahuje rovnaké ID ako prvá, len v opačnom poradí
- druhá správa obsahuje jedno rovnaké ID
- prvá správa neobsahovala žiadne ID z druhej zisťovacej správy

Ukážeme si, že vo všetkých prípadoch je v systéme dostatok správ a nemôže sa stať, že by sa všetky správy stratili a algoritmus sa zasekol. Do úvahy nebudeme brať správy, ktoré vedú preč od iniciátora zisťovania okolia a nemajú šancu sa k nemu vrátiť. Ak by sme s nimi počítali, tak by ich inteligentný útočník postupne doručil a ostali by sme v situácii, že ich už nemáme, preto potrebujeme mať dostatok poslaných správ v jednom kroku bez týchto správ.

V prípade, že druhá prijatá zisťovacia správa obsahuje rovnaké ID procesorov ako prvá, len v opačnom poradí, tak procesor musí čakať na ďalšiu zisťovaciu správu. Celkovo máme v systéme dve správy o voľbe šéfa, dve zisťovacie správy, ktoré jednu linku prešli spoločne s prvou alebo druhou už doručenou zisťovacou správou a dve až štyri zisťovacie správy, ktoré boli poslané z iniciátora cez porty, z ktorých sa ešte nevrátila žiadna zisťovacia správa (dve až štyri, lebo nevieme, či už prešli po prvej linke alebo ešte nie). Aj v najhoršom prípade, keď by posledné dve spomínané správy ešte neprešli po prvej linke a stretli by sa s druhými dvoma zisťovacími správami, ktoré sú v systéme

(budú sa posielať po rovnakých linkách, len v opačnom smere, takže môžu navzájom kolidovať správy a potvrdenia), tak v systéme budeme mať spolu 4 správy na rôznych linkách a teda máme dostatok správ, aby sme mali garantované doručenie aspoň jednej z nich.

Ak druhá zisťovacia správa obsahuje prvé alebo tretie ID zhodné s prvým alebo tretím ID z prvej zisťovacej správy, tak procesor potom už vie rozlíšiť, ktoré porty vedú k náprotivným procesorom, ale ešte nevie vytvoriť štyri disjunktné cesty. V tomto stave nám však v systéme zostávajú dve správy o voľbe šéfa a minimálne dve zisťovacie správy. Zisťovacích správ môže byť v danom momente aj viac a to v prípade, že už stihli zisťovacie správy prejsť aspoň cez prvú linku.

Posledná možnosť prijatia druhej informačnej správy je tá, že prvá a druhá informačná správa obsahujú disjunktné ID procesorov, cez ktoré prechádzali. V tomto prípade procesor nevie rozlíšiť, ktoré procesory sú si naproti sebe a opäť musí čakať na ďalšiu zisťovaciu správu. V systéme máme dve správy o voľbe šéfa, dve zisťovacie správy, ktoré sa "oddelili" od už doručených zisťovacích správ (tieto správy sa môžu stretnúť, konkrétne jedna správa a potvrdenie tej druhej) a ešte minimálne jednu zisťovaciu správu.

V prvom a treťom prípade nevie procesor poslať žiadnu ďalšiu správu o voľbe šéfa, v druhom môže poslať jednu novú správu o voľbe šéfa. Prvý prípad môže po doručení ďalšej zisťovacej správy prejsť do druhého alebo tretieho prípadu s tým, že už je doručená jedna zisťovacia správa naviac, ale stále bude dosť správ v systéme. Ak je prijatá ďalšia zisťovacia správa v druhom prípade (nie taká, ktorá obsahuje rovnaké ID ako niektorá z už prijatých správ), tak už procesor vie smerovať správy s voľbou šéfa k susedným procesorom tak, aby mali disjunktné cesty. V treťom prípade tiež po prijatí ďalšej zisťovacej správy s novým ID procesor vie smerovať správy s voľbou šéfa po disjunktných cestách.

Ďalšie procesory na ceste správ s voľbou šéfa budú pokračovať podobne ako prvý iniciátor. Jediná zmena bude v tom, že správu, ktorá prišla priamo, sa snažia preposlať priamo a správu, ktorá prešla cez tri linky, sa budú snažiť poslať tiež cez tri linky ďalej (ak prišla správa vrchom, tak ju pošlú spodkom a naopak). Po prijatí prvej správy o voľbe šéfa iniciujú zisťovanie okolia a keď budú mať dostatok informácií, tak budú posielať správy s voľbou šéfa ďalej.

Samotná voľba šéfa prebieha rovnako ako v prípade so znalosťou orientácie.

4.4 Kolízie správ

Keďže sa správy pohybujú blízko seba, tak môže ľahko dochádzať k ich kolíziám. Kolízne správy však stačí spojiť do jednej, poslať spolu a v ďalšom procesore ich rozpojiť

a spracovať. Dôvod, prečo to môžeme urobiť, je ten, že stále budeme mať dostatok správ v systéme a ich spájaním dokonca dosiahneme to, že ak prejde správa, ktorá je spojená z viacerých správ, tak cez danú linku prejde viac informácie. Dostatok správ v systéme je zaručený tým, že správy s voľbou šéfa navzájom nekolidujú (pri voľbe šéfa na jednotlivých zvislých kruhoch môžu kolidovať, ale vždy budú posielané aspoň štyri správy s najlepším ID) a okrem nich je v systéme aj dostatok správ o zisťovaní susedov.

4.5 Počet správ potrebný na voľbu šéfa pri jednom iniciátorovi

Ak máme iba jedného iniciátora, môžeme použiť spôsob voľby šéfa rovnaký, ako pri jednom iniciátorovi pri znalosti orientácie, t.j. informujeme po kruhu o potrebe voľby šéfa (smer závisí od toho, ktorá zisťovacia správa príde ako prvá), následne sa v smere kolmo urobí voľba šéfa na jednotlivých kruhoch, potom sa urobí sumárna voľba a na záver broadcast s informáciou o zvolenom šéfovi. Keďže zisťovanie okolia nám pridáva vždy iba konštantný počet správ, tak zložitosť zostane zachovaná a to konkrétne $O(N^2)$.

4.6 Problém viacerých iniciátorov

Pri viacerých iniciátoroch máme ten problém, že si každý iniciátor na začiatku vyberá smer, ktorým pošle informáciu, že sa má urobiť voľba šéfa v smere kolmo na danú správu. Ak by si všetci iniciátori vybrali rovnaký smer, tak z pohľadu zložitosti sa dostávame do pozície, ako keď sme mali znalosť orientácie, lebo samotné zisťovanie okolia nám pridá v každom kroku iba konečný počet správ.

Problém nastáva, ak si aspoň jeden iniciátor vyberie opačný smer. Vtedy bude prebiehať voľba šéfa súčasne aj na zvislých, aj na vodorovných kruhoch. Súčasná voľba na zvislých aj vodorovných kruhoch by viedla k zhoršeniu zložitosti na:

$$[Y \cdot X \cdot (\log X)] \cdot [X \cdot Y \cdot (\log Y)] \cdot N = O(N^3 \cdot (\log X) \cdot (\log Y)) \tag{4.1}$$

V tejto rovnici prvá časť zodpovedá Y zvislým inštanciám, druhá X vodorovným inštanciám a ešte sú tieto počty správ vynásobené N, lebo bežia všetky tieto inštancie súčasne, ale je garantované doručenie iba jednej správy.

Ak by sme vedeli dosiahnuť, že sa iniciátori rozhodnú rovnako alebo vedeli zastaviť dostatočne skoro voľbu šéfa v jednom smere, tak by sme dosiahli podstatne lepšiu zložitosť.

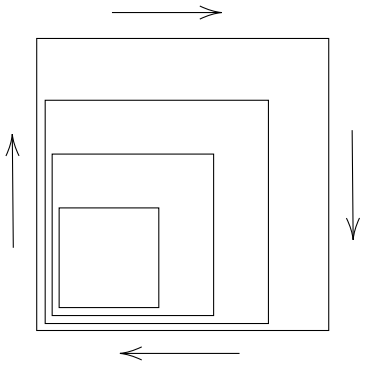
Kapitola 5

Ďalší algoritmus so znalosťou aj bez znalosti orientácie - lepšia zložitosť

Po tom, ako sme zistili, že nami navrhnutý algoritmus má bez znalosti orientácie horšiu zložitosť ako generický algoritmus pre k-súvislé grafy, snažili sme sa zmeniť náš prístup k problému voľby šéfa.

5.1 Úvod

Keďže spôsob voľby po kruhoch viedol bez znalosti orientácie k výraznému zhoršeniu, rozhodli sme sa preskúmať druhý spôsob, ktorý bol navrhnutý na voľbu šéfa na 2D toruse. Ide o algoritmus, v ktorom sa každý iniciátor snaží postupne vo fázach označiť väčšie územie.



Obr. 5.1: Znázornenie postupne zväčšujúcich sa štvorcov, ktoré predstavujú označované územie.

5.2 Algoritmus na voľbu šéfa značením územia

Ide o Petersonov algoritmus [13] modifikovaný Bernardom Mansom [11]. Základným cieľom každého aktívneho procesora je vo fáze i označiť štvorcovú oblasť veľkosti d, keď sa mu to podarí, tak postúpi do ďalšej fázy $(i + 1)$ a skúsi označiť väčšiu oblasť, ako je to ilustrované na obrázku 5.1, Veľkosť, ktorú sa snaží označiť, je daná číslom fázy a konštantou α nasledovne: $d = \alpha^i$. Na začiatku je označovaným územím samotný kandidát/iniciátor. Označovanie územia je vykonávané "označovacou" správou, ktorá je preposielaná vždy do vzdialenosti d jedným smerom, najprv sa posiela smerom hore, potom doprava, potom smerom dole a nakoniec doľava, aby sa vrátila k pôvodnému odosielateľovi "označovacej" správy.

Pri označovaní územia môže dôjsť k dvom situáciám. Prvá je tá, že "označovacia" správa procesoru x sa vráti bez toho, aby preťala územie označované nejakým iným procesorom. V tomto prípade procesor x postúpi do ďalšej fázy a pokúsi sa označiť väčšie územie.

Ak "označovacia" správa príde do procesoru už označeného iným kandidátom y v rovnakej fáze, tak pokiaľ má y väčšie ID, správa pokračuje v označovaní územia s tým, že sa v nej nastaví boolovská premenná *videnéVäčšie* na *true*. V prípade, že má y menšie ID, tak príchodzia "označovacia" správa procesoru x už nebude preposielaná a je poslaná nová správa "videnéVäčším(x,i)", ktorá bude preposielaná po území označovanom procesorom y.

Ak procesor x príjme aj svoju "označovaciu" správu s nastavenou premennou *videnéVäčšie* na *true* a aj správu "videnéVäčším", tak postúpi do fázy $(i + 1)$ a začne označovať väčšiu oblasť.

Zatiaľ sme spomínali situácie, kedy správa a procesor, ktorý ju prijal, boli v rovnakej fáze. Ak nastane situácia, že správa je vo väčšej fáze, tak procesor je automaticky označený a preposiela danú správu ďalej. Pokiaľ je správa v nižšej fáze, tak daná správa zaniká.

Je dobré si všimnúť, že keď x dostane správu "videnéVäčším(y,i)", tak y sa už v ďalších fázach nebude snažiť označovať väčšie územie, lebo jeho "označovacia" správa sa k nemu nevrátila, ale sa zmenila na správu "videnéVäčším(y,i)", ktorú prijal procesor x. Z uvedeného vyplýva, že ak procesor x prežil, tak buď jeho správa nenarazila na žiadne iné označené územie alebo aspoň jeden iný kandidát neprežil.

V prípade, keď sa v jednom vrchole naraz stretne viac "označovacích" správ, tak sa daný vrchol stane súčasťou územia označeného správou s najmenším ID.

Pre správnosť algoritmu je nutné, aby ak procesor z bol súčasťou označeného územia procesora x a teraz sa stane súčasťou územia označeného procesorom y, následná správa "videnéVäčším" určená pre x bola posielaná po území procesora y. Na to, aby sme udržali počet správ malý, tak potrebujeme limitovať počet správ typu "videnéVäčším"

poslané jedným procesorom. To zabezpečíme tak, že procesor prepošle iba jednu takú správu (obmedzenie jednej správy sa vzťahuje na jednu fázu).

Popísaným spôsobom funguje algoritmus až dovtedy, kým nenastane situácia, v ktorej platí $d_i \geq \sqrt{n}$. V takomto prípade procesor dostane "označujúcu" správu zdola namiesto z pravej strany, je to z dôvodu cyklickosti torusu a budeme tento prípad volať zacyklenie. Následne pošle správu napravo a očakáva jej príchod z ľavej strany. Ak sa mu to podarí, tak všetky následné fázy označovania sa stanú jednoduchšími.

Všetky fázy po zacyklení pošle procesor x "označovaciu" správu najprv hore a očakáva ju zdola a potom ju pošle doľava a očakáva ju sprava. Situácia, keď ostane už iba jeden kandidát, je určite dosiahnutá po konštantnom počte p fáz po zacyklení. Ak teda procesor x prežije p fáz po zacyklení, tak sa stáva šéfom a oznámi všetkým ostatným výsledok a nastáva koniec.

Zložitosť tohto algoritmu sa pre $\alpha \approx 1.1795$ rovná $\theta(n)$. [11, 13]

5.3 Nami modifikovaný algoritmus

Algoritmus, ako sme ho popísali vyššie, by v prostredí jednoduchého prahového modelu nefungoval správne. Je v ňom treba zmeniť viacero častí, aby sme zaručili jeho korektné fungovanie, aj keď môže dochádzať k strácaniu správ a je nutné poslať aspoň štyri správy v každom tiku hodín, ak chceme mať istotu, že aspoň jedna bude doručená.

Asi najväčšou zmenou je to, že "označovacie" správy budeme posielať dvomi smermi. Každým smerom sa bude posielať dvojica správ v podstate rovnakým spôsobom, ako je v kapitole 3 znázornené na obrázku 3.1, teda striedavo priamo a obchádzkou. Rozdiel je v tom, že nepôjdu proti sebe na kruhu, ale na označovanom území, čiže na začiatku sa dve pošlú smerom hore a dve smerom doprava. V pôvodnom algoritme išla jedna správa v smere hodinových ručičiek, zatiaľ čo v našom algoritme pôjdu dve správy v smere a dve správy proti smeru hodinových ručičiek, ale stačí, aby jedna z nich prešla celé označované územie, zvyšné tri sú na to, aby sme mali zaručený dostatok správ v systéme.

V momente, keď prvá zo štyroch "označovacích" správ prejde cez celé označované územie, tak síce už nemusia byť štyri správy v celom systéme, ale už bola potrebná informácia doručená. Ak táto správa neobsahovala premennú *videnéVäčšie* nastavenú na *true*, tak môže procesor prejsť do ďalšej fázy a opäť pošle štyri "označovacie správy" označovať väčšie územie. Ak správy z neskoršej fázy dobehnú správy zo skoršej fázy alebo správy zo skoršej fázy prídu do procesoru označeného rovnakým procesorom, ale neskoršou fázou, tak potom správy zo skoršej fázy zaniknú, lebo informácia, ktorú nesú, už bola doručená inou správou a teda nie sú potrebné. Ak príde správa do procesoru označeného rovnakým ID, ako nesie ona sama a v rovnakej fáze, tak je posielaná ďalej,

lebo sa nedá zistiť, či správa, ktorá daný procesor označila, už prišla späť k iniciátorovi alebo je ešte na ceste a teda, ak by sme ju neposielali ďalej, mohlo by sa stať, že ostane v systéme menej správ ako štyri a žiadna by sa nevedela doručiť.

V prípade, že doručená "označovacia" správa obsahovala premennú *videnéVäčšie* nastavenú na *true*, tak musí procesor čakať na správu "videnéVäčším". Rovnako ako "označovacie" správy, tak aj správy typu "videnéVäčším" sa namiesto jednej posielajú štyri, po dve každým smerom označovaného územia, čiže sa nemusíme obávať, že by bol v systéme nedostatok správ a mohla by nastať situácia, že sa správa typu "videnéVäčším" nedoručí. Takéto správy posielame rovno všetky štyri. V tomto spôsobe sa môže stať, že správa "videnéVäčším" príde skôr ako "označovacia" správa. Tento jav v skutočnosti nepredstavuje žiaden problém, lebo ak procesor dostane správu "videnéVäčším" skôr, tak počká na "označovaciu" správu a bez ohľadu na to, či má alebo nemá táto správa nastavenú premennú *videnéVäčšie* na *true*, tak môže prejsť do ďalšej fázy (nemusí ju mať nastavenú z toho dôvodu, že mohla ísť opačným smerom a prejsť cez vrchol, v ktorom došlo k stretu skôr).

Vzhľadom na to, že posielame vždy správy aj vodorovne aj zvislo, tak keď d_i bude väčšie ako $\sqrt{n}$ a dôjde k zacykleniu, nečakáme, ako v pôvodnom algoritme na zvislý smer a potom neposielame správu vodorovne. Musíme však dávať pozor na to, aby bol dostatok správ v systéme. Keď nastane prvý raz situácia, že správa poslaná smerom hore bude doručená zdola namiesto sprava alebo správa poslaná doprava bude doručená zľava namiesto zhora, tak vtedy detegujeme zacyklenie a potrebujeme poslať správu smerom dolu, ak prišla správa zľava, alebo doľava, ak prišla správa zdola (je to naproti správam z opačného smeru, ako je smer správy, ktorá bola doručená po zacyklení). Keď sa doručí ďalšia správa z rovnakého smeru ako prvá, tak postupujeme analogicky.

V prípade, že sa doručí druhá správa zo smeru kolmého na smer prvej prijatej správy, tak už nemusíme posielať ďalšie "označovacie" správy v danej fáze, lebo aspoň jedna správa označila vodorovný kruh a aspoň jedna zvislý kruh, ale môžeme prejsť do ďalšej fázy (za predpokladu, že boli splnené podmienky na prechod do ďalšej fázy). V prípade, že sa procesoru x podarí prežiť p fáz po zacyklení, rovnako ako v pôvodnom algoritme, tak sa stáva šéfom a ostáva mu už iba o tom všetkých informovať.

5.4 Kolízie správ

Správy rovnakého typu týkajúce sa jedného procesora sú vždy posielané po disjunktných cestách a teda ku kolíziám medzi správami môže dochádzať iba medzi správami rôznych procesorov alebo medzi správami rôzneho typu.

Z "označovacích" správ rôznych procesorov sa vyberá tá od najlepšieho kandidáta a teda síce sa môžu stretnúť v jednom procesore, ale možnosť pokračovať bude mať iba

jedna. Správa typu "videnéVäčším" sa po jednej linke posiela vždy iba jedna v rámci jednej fázy a ak by sa stretli viaceré správy určené pre jeden procesor, tak potom sa pošle iba tá s najväčšou fázou (ak existuje správa s väčšou aj menšou fázou, tak bola doručená iná správa s menšou fázou a teda túto informáciu už nie je potrebné šíriť).

V prípade stretu "označujúcej" správy a správy "videnéVäčším" môžeme tieto správy spojiť. Ak by totiž ešte nedokončila označovanie žiadna správa, tak aj po spojení budeme mať stále aspoň štyri správy v systéme. Ak už nejaká správa dokončila označovanie, tak ostatné označovacie správy už nenesú novú informáciu, je ich v systéme menej ako štyri, ale sú v systéme aspoň štyri správy "videnéVäčším" a teda opäť môžeme správy spojiť. Ak už bola doručená aj správa "videnéVäčším", tak spojením správ môže nastať situácia, že nie je v systéme dostatok správ, ale už bola doručená potrebná informácia.

Ak nastane stretnutie správ vo finálnej fáze, keď sa stretne finálna správa s inou správou, tak automaticky má prednosť správa z finálnej fázy a ostatné správy sa už neposielajú ďalej. Pre finálne správy navzájom platia rovnaké pravidlá, ako pri "označovacích" správach a správach typu "videnéVäčším".

5.5 Korektnosť a potrebný počet správ

Na to, aby sme ukázali korektnosť, musíme ukázať, že zmeny, ktoré sme v algoritme urobili, majú vplyv iba na počet správ a nie na fungovanie algoritmu ako takého.

Modifikácie, ktoré sme robili, nám prinášajú vždy iba konštantne-krát viac správ voči pôvodnému algoritmu. V prípade "označovacích" správ a správ typu "videnéVäčším" ide zhodne o štvornásobok. Tým, že posielame štyri správy naraz a potrebujeme disjunktné cesty, niektoré správy musia prejsť po troch linkách, kým sa dostanú k ďalšiemu procesoru na označenie. Ide však opäť o konštantné predĺženie.

Lema 5.5.1. *Maximálny počet aktívnych procesorov v jednotlivých fázach je rovnaký ako v pôvodnom algoritme, t.j.* $n/\alpha^{2i}(2-\alpha^2)$.

Dôkaz. V nami upravenom algoritme sme nezmenili veľkosť označovaného územia. Zmenili sme iba spôsob, akým sa posielajú správy, ale samotné podmienky na prechod do vyššej fázy sme nemenili a teda zostávajú platné argumenty prezentované Petersonom [13].

Zoberme možné maximum n_i z aktívnych procesorov vo fáze i. Do vyššej fázy sa môžu dostať procesory dvoma spôsobmi. Prvý je, že na nikoho pri označovaní územia nenarazili a druhý je, že síce narazili na hranicu označenú procesorom s väčším ID, ale zároveň ich hranica bola videná správou nesúcou väčšie ID. Nech a_i je počet procesorov, ktoré sa do vyššej fázy dostali prvým spôsobom, je zrejmé, že musí platiť $a_i \leq n/d_i^2$. Počet procesorov, ktoré sa dostali do vyššej fázy druhým spôsobom, môže byť nanajvýš

$(n_i - a_i)/2$, lebo každý porazený mohol zabezpečiť prežitie najviac jedného kandidáta. Z toho dostávame:

$$n_{i+1} \leq a_i + (n_i - a_i)/2 = (n_i + a_i)/2 \leq (n_i + n/d_i^2)/2 \tag{5.1}$$

Keďže $d_i = \alpha^i$ sa v každej fáze zväčšuje, tak potom horné ohraničenie n_i na počet kandidátov sa zmenšuje. Riešením rekurentných vzťahov dostávame [16]:

$$n_{i+1} \leq n/\alpha^{2i}(2 - \alpha^2) \tag{5.2}$$

□

Veta 5.5.1. *Počet správ potrebných na voľbu šéfa na 2D toruse v jednoduchom prahovom modeli je $O(n^2)$.*

Dôkaz. Každý aktívny procesor sa vo fáze i snaží označiť územie veľkosti $4d_i = 4\alpha^i$. Obomi smermi po okraji tohto územia posiela po dve "označovacie" správy, pričom vždy ide jedna z nich priamo a druhá obchádzkou dĺžky 3. Okrem "označovacích" správ môžu po rovnakých cestách byť posielané štyri správy typu "videnéVäčším". Z toho dostávame spolu $2 \cdot 4 \cdot 3 \cdot 4 \cdot \alpha^i$ poslaných správ za každý aktívny procesor. Keď dosadíme počet správ potrebných jedným procesorom do horného ohraničenia maximálneho počtu aktívnych procesorov uvedeného v nerovnici 5.2 upraveného pre fázu i $(n/\alpha^{2(i-1)}(2-\alpha^2))$, dostaneme, že je v danej fáze potrebných $O(n/\alpha^{i-2}(2-\alpha^2))$ správ.

Zatiaľ sme nebrali do úvahy, že pracujeme v jednoduchom prahovom modeli, v ktorom máme garantované iba to, že sa v jednom tiku hodín doručí jedna správa, za predpokladu, že bude v celom systéme poslané dostatočné množstvo správ. Reálne sa každý snaží poslať svoje správy, ale doručiť sa môže vždy iba jedna (o ktorej sa aj vie, že je doručená a teda sa nebude už po danej linke posielať znovu). Potrebujeme preto počet správ potrebných na jednu fázu ešte raz vynásobiť počtom procesorov v danej fáze a tiež počtom fáz, čím dostávame počet správ vo fázach pred zacyklením $O(n^2\alpha^2/(2-\alpha^2)^2(\alpha-1))$. Do fázy, kedy dôjde k zacykleniu, sa dostane iba konštantný počet procesorov [13], pričom sa pošle vo finálnych fázach celkovo $O(\sqrt{n})$ správ (Tu sa nemení zložitosť voči modelu, v ktorom sa nestrácajú správy. Je to preto, lebo nie všetky správy jednej fázy sa posielajú naraz, ale vždy iba konštantný počet). Pri zvolení $\alpha = 2$ dostávame požadované $O(n^2)$. □

5.6 Fungovanie pri strate orientácie

V prípade, že jednotlivé procesory nemajú zmysel pre orientáciu, budeme postupovať podobne, ako pôvodný algoritmus a to s využitím "zábradlia". Ako sme už písali v predošlej kapitole, zisťovanie lokálneho okolia nemôžeme urobiť ako v pôvodnom algoritme

na začiatku, ale potrebujeme ho spojiť so samotnou voľbou šéfa. Spojenie zisťovania lokálneho okolia a samotnej voľby šéfa budeme robiť aj v tomto prípade rovnako ako v predošlej kapitole. Rozdiel bude v spracovaní správ týkajúcich sa samotnej voľby šéfa, lebo tie sa nebudú vždy posielať priamo, ale potrebujú niekedy aj zmeniť smer.

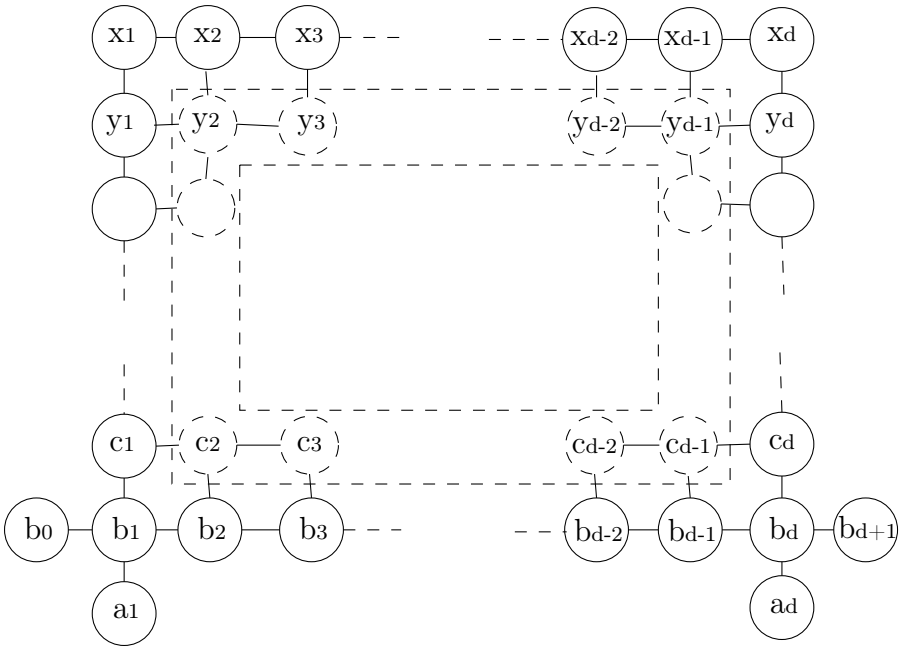
Pri posielaní správ týkajúcich sa voľby šéfa potrebujeme vedieť zaručiť, aby bola správa preposielaná správny počet krát priamo a aby všetky tri odbočenia boli v rovnakom smere. Preposielanie priamo je jednoduchšia úloha, lebo jednotlivé procesory na základe zisťovania svojich susedov vedia rozlíšiť, ktoré procesory sú si navzájom susedné a ktoré sú naproti sebe. Problémom je rozlíšiť strany, lebo tie si môže každý procesor určiť inak. Napríklad jeden procesor si určí, že daný port vedie smerom hore, no v skutočnosti z globálneho pohľadu môže tento port smerovať dole. Každý procesor si však potrebuje určiť, ktorý port smeruje na ktorú stranu, aby sa v každej fáze nerozhodoval náhodne, ale vždy sa snažil zväčšovať územie jedným smerom.

Z globálneho pohľadu neprekáža, ak nebude každý procesor v ľavom dolnom rohu označovaného územia [11, 13], ale môže byť v ktoromkoľvek z rohov a taktiež smer, ktorým je územie označované, môže byť ako v smere hodinových ručičiek, tak aj v protismere. Podstatné je, aby správa vedela odbočovať do rovnakého smeru. Ak by totiž raz odbočila doprava, potom doľava a potom znovu doprava, tak by sa nemusela vrátiť k pôvodnému odosielateľovi, ale s najväčšou pravdepodobnosťou by skončila v nejakom úplne inom procesore.

Rovnaký smer odbočovania správ je dosiahnutý pomocou "zábradlia", ktoré ilustrujeme na obrázku 5.2, kde "zábradlie" reprezentujú čiarkované kruhy medzi čiarkovanými obdĺžnikmi. Lokálne známe okolie dvoch bezprostredne susedných procesorov obsahuje prienik, ktorý tvoria štyri ďalšie procesory. Práve táto informácia známa obidvom susedným procesorom nám poslúži na vybudovanie "zábradlia", ktoré je budované dynamicky a to tak, že procesor do správy pridá informáciu, na stranu ktorého procesora má odbočiť. Keď jeho sused správu príjme, skontroluje, o ktorú stranu ide a aktualizuje túto informáciu tak, že vymení ID procesora, ktorého smerom sa má urobiť zatočenie, za ID procesora, ktorý je na rovnakej strane a vidí ho aj nasledujúci príjemca. Ak už má správa odbočiť, tak ju môže poslať priamo procesoru, ktorý je uvedený ako "zábradlie" a do správy dať ID procesoru, ktorý poslúži ako začiatok novej časti zábradlia.

5.7 Kolízie správ pri strate orientácie

Pribudol nám ďalší typ správ, správy zisťujúce okolie, ktorý môže kolidovať s inými správami ("označovacími", "videnéVäčším" a správami z finálnych fáz). Samotné kolízie správ zisťujúcich okolie sme už popísali v kapitole 4 a aj teraz budeme postupovať rovnakým spôsobom, že zisťovacie správy budeme spájať s inými správami, ak majú



Obr. 5.2: "Zábradlie" v 2D toruse tvoria vrcholy, ktoré sú medzi čiarkovanými obdĺžnikmi.

ísť po jednej linke. Kolízie ostatných typov správ zostávajú taktiež riešené rovnako ako pri znalosti orientácie.

5.8 Korektnosť a zložitosť pri strate orientácie

Využívanie "zábradlia" vo svojej práci uviedol a dokázal jeho správnosť Bernard Mans [11] a jeho korektnosť ostáva zachovaná aj v jednoduchom prahovom modeli, lebo ide o informáciu naviac v správe a v našom modeli nedochádza k zmene obsahu správ.

Veta 5.8.1. *Počet správ potrebných na voľbu šéfa na 2D toruse bez orientácie v jednoduchom prahovom modeli je* $O(n^2)$.

Dôkaz. Samotné využívanie zábradlia nám nepridáva žiadne správy naviac, len zväčšuje veľkosť správ. Počet správ nám zväčšuje iba zisťovanie okolia jednotlivých procesorov. Ide však iba o konštantný počet správ pre každý procesor a teda zložitosť algoritmu ostáva zachovaná rovnaká ako pri algoritme so znalosťou orientácie, ktorého zložitosť je $O(n^2)$. □

Záver

Voľba šéfa je pomerne silno skúmaný problém, existuje veľa modelov, ktoré boli na skúmanie tohto problému vytvorené, pričom niektoré viac a niektoré menej odzrkadľujú realitu. Cieľom tejto práce bolo preskúmať možnosti jednoduchého prahového modelu na topológii 2D torusu a navrhnúť v tomto modeli algoritmus na voľbu šéfa.

Mali sme aj určité obmedzenie na zložitosť, lebo už existuje všeobecný algoritmus na voľbu šéfa v jednoduchom prahovom modeli pre k-súvislé grafy. Naším cieľom bolo skúsiť využiť špecifické vlastnosti 2D torusu, aby nami navrhnutý algoritmus mal lepšiu zložitosť alebo v prípade, že by sa nám to nedarilo, skúsiť dokázať dolný odhad zložitosti, že lepšie sa to nedá.

Niektoré spôsoby, ktoré sme skúšali, sa nepodarilo adaptovať do prostredia, v ktorom dochádza k strácaniu správ. Prvý prezentovaný algoritmus, ktorý funguje aj v jednoduchom prahovom modeli, má lepšiu zložitosť, pokiaľ majú jednotlivé procesory znalosť orientácie. V prípade, keď túto znalosť procesory nemajú, tak sa zložitosť rapídne zhoršila a výsledná zložitosť bola nie iba rovnaká, ale dokonca horšia ako všeobecný algoritmus. Problémom bolo to, že aj keď sme vedeli získať znalosť lokálnej topológie, nevedeli sme zaručiť jej konzistentnosť naprieč všetkými procesormi a to potom viedlo k spomínanému zhoršeniu zložitosti.

Preto sme sa rozhodli zmeniť náš prístup, ale zároveň využiť už získané znalosti. Druhý algoritmus, ktorý sme navrhli, už mal lepšiu zložitosť ako všeobecný algoritmus pre k-súvislé grafy. Dokonca sa nám podarilo zlepšiť algoritmus tak, že náš druhý algoritmus je lepší aj pri znalosti orientácie, aj bez znalosti orientácie, ako náš prvý prezentovaný algoritmus so znalosťou orientácie. Podarilo sa nám dosiahnuť zložitosť $O(n^2)$ aj so znalosťou orientácie, aj bez nej. Všeobecný algoritmus na voľbu šéfa na k-súvislých grafoch má zložitosť $O(2^k m^2 n)$ (k je súvislosť, m je celkový počet hrán a n je počet vrcholov), čo pri 2D toruse môžeme napísať ako $O(n^3)$. Zložitosť nášho algoritmu je v počte správ a zložitosť všeobecného algoritmu je časová, ale odoslať $O(n^2)$ správ netrvá viac ako $O(n^2)$ času.

Ďalšie skúmanie v tejto oblasti by mohlo byť dokázanie dolného odhadu zložitosti na potrebný počet správ na voľbu šéfa v jednoduchom prahovom modeli na 2D torusoch a tiež skúmanie ďalších špecifických topológií, kde by mohla byť zložitosť lepšia ako má všeobecný algoritmus na voľbu šéfa na k-súvislých grafoch.

Literatúra

[1] Dana Angluin. Local and global properties in networks of processors. In *Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 82–93. ACM, 1980.

[2] Peter Banda. *Anonymous Leader Election in One-and Two-Dimensional Cellular Automata.* PhD thesis, Comenius University, Bratislava, Slovakia, 2014.

[3] bc. Martin Námešný. Voľba šéfa na cayleho grafoch s lineárnym počtom správ. Master's thesis, Fakulta matematiky, fyziky a informatiky Univerzity Komenského, 2010.

[4] bc. Ondrej Pašuth. Voľba šéfa v sietach s chybnými linkami. Master's thesis, Fakulta matematiky, fyziky a informatiky Univerzity Komenského, 2011.

[5] Ernest Chang and Rosemary Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes. *Communications of the ACM*, 22(5):281–283, 1979.

[6] Stefan Dobrev, Rastislav Královič, Richard Královič, and Nicola Santoro. On fractional dynamic faults with threshold. In *International Colloquium on Structural Information and Communication Complexity*, pages 197–211. Springer, 2006.

[7] Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Leader election in extremely unreliable rings and complete networks. In *International Conference On Principles Of Distributed Systems*, pages 512–526. Springer, 2008.

[8] Paola Flocchini, Bernard Mans, and Nicola Santoro. Sense of direction: Definitions, properties, and classes. *Networks*, 32(3):165–180, 1998.

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

[10] Daniel S. Hirschberg and James B. Sinclair. Decentralized extrema-finding in circular configurations of processors. *Commun. ACM*, 23(11):627–628, 1980.

[11] Bernard Mans. Optimal distributed algorithms in unlabeled tori and chordal rings. *J. Parallel Distrib. Comput.*, 46(1):80–90, 1997.

[12] Codrin M. Nichitiu, Jacques Mazoyer, and Eric Rémila. Algorithms for leader election by cellular automata. *J. Algorithms*, 41(2):302–329, 2001.

[13] Gray Lynn Peterson. *Efficient algorithms for elections in meshes and complete networks*. Department of Computer Science, University of Rochester, 1985.

[14] Mohammed Refai, Ahmad Sharieh, and Fahad Alshammari. Leader election algorithm in 2d torus networks with the presence of one link failure. *Int. Arab J. Inf. Technol.*, 7(2):105–114, 2010.

[15] Mohammed Al Refai. Dynamic leader election algorithm in 2d torus network with multi links failure. *International Journal of Computer Science Trends and Technology (IJCST)*, 2(5), 2014.

[16] Nicola Santoro. *Design and analysis of distributed algorithms*, volume 56. John Wiley & Sons, 2006.

[17] Alvy Ray Smith. Two-dimensional formal languages and pattern recognition by cellular automata. In *Switching and Automata Theory, 1971., 12th Annual Symposium on*, pages 144–152. IEEE, 1971.

[18] Gerard Tel. *Introduction to distributed algorithms*. Cambridge university press, 2000.