Algorytmy rozproszone odporne na błędy

Autoreferat pracy doktorskiej

Michał Strojnowski

1 Wprowadzenie

W pracy analizowana jest złożoność komunikacyjna dwóch popularnych problemów z dziedziny obliczeń rozproszonych: *plotkowania* i *konsensusu*. Plotkowanie to problem efektywnej wymiany posiadanych informacji pomiędzy wszystkimi jednostkami w systemie. Konsensus to problem podjęcia wspólnej decyzji przez wszystkie jednostki w systemie. Dotychczasowe prace nad tymi problemami skupiały się głównie na uzyskaniu optymalnej złożoności czasowej algorytmu i jego odporności na błędy (patrz np. [1, 6, 7, 8, 11, 12]). W tej pracy dodatkowo minimalizowana jest liczba użytych komunikatów oraz ich sumaryczna długość.

Rozważany tu model systemu rozproszonego to synchroniczny system z przesyłaniem wiadomości (*synchronous message-passing system* [11]). W systemie takim jednostki dysponują zsynchronizowanymi zegarami i mogą się komunikować wysyłając wiadomości "jeden do jednego". Model zakłada że jednostki są podatne na awarie, następujące w sposób nieprzewidywalny w trakcie wykonania algorytmu. Rozwiązanie musi zapewniać uzyskanie poprawnego wyniku niezależnie od tego które jednostki i w jakiej kolejności ulegną awarii. W przypadku plotkowania przedstawione rozwiązania są odporne na dowolną liczbę awarii. W przypadku przedstawione rozwiązania zapewniają sukces jeśli nie więcej niż jedna trzecia jednostek ulegnie awarii.

Praca składa się z dwóch części. Pierwsza część jest poświęcona problemowi plotkowania, analizowanym dla trzech różnych typów awarii: zatrzymań, gubienia komunikatów i złośliwego zachowania (Bizantyjskiego) [2]. Dla każdego typu awarii przedstawiany jest algorytm odporny na dowolną liczbę błędów, asymptotycznie optymalny pod względem liczby użytych komunikatów. Do każdego algorytmu dołączony jest dowód jego optymalności, w postaci dolnego ograniczenia na złożoność komunikacyjną tego problemu. Wszystkie algorytmy zaprojektowane są w oparciu o wspólny schemat rozproszonej komunikacji, wprowadzony w tej pracy. Schemat ten tworzy kilka rodzin grafów z wysokimi współczynnikami ekspansji i niewielką liczbą krawędzi. Grafy te określają które pary jednostek mają prawo wysyłać do siebie komunikaty w kolejnych fazach algorytmu. Istnienie odpowiednich grafów jest dowodzone przy użyciu metod probabilistycznych.

Druga część pracy przedstawia rozwinięcie przedstawionego schematu komunikacji i stworzenie na jego bazie dwóch algorytmów konsensusu: deterministycznego i kwantowego. W odróżnieniu od pierwszej części, optymalność tych algo-

rytmów nie jest dowiedziona. Jednak w obu przypadkach koszt komunikacyjny jest niższy niż najlepszych dotychczas znanych rozwiązań działających w tym samym czasie.

1.1 Opis modelu

Rozważany tutaj model obliczeń rozproszonych to standardowy synchroniczny system z wymianą komunikatów [11]. System taki zkłada się ze zbioru jednostek obliczeniowych współpracujących w celu wykonania jakiegoś zadania. Każda jednostka ma wiedzę o wszystkich pozostałych jednostkach i może do każdej z nich wysyłać komunikaty. Wykonanie algorytmu następuje w *rundach*. W każdej rundzie każda jednostka może odebrać dowolną liczbę komunikatów, wykonać dowolne obliczenie i wysłać dowolną liczbę komunikatów. Wysłane komunikaty docierają do odbiorców w kolejnej rundzie, o ile nie nastąpi awaria nadawcy lub odbiorcy.

Każda jednostka posiada unikalny identyfikator, od którego może uzależniać swoje zachowania. Ponadto każda jednostka może posiadać jakieś dane wejściowe. Każdy problem jest zdefiniowany jako zależność pomiędzy danymi wejściowymi i danymi posiadanymi przez jednostki przy zakończeniu działania. Rozważamy tutaj wyłącznie rozwiązania w których wszystkie jednostki kończą działanie jednocześnie, po wykonaniu ustalonej z góry liczby rund.

1.2 Awarie jednostek

W trakcie wykonania algorytmu poszczególne jednostki mogą ulegać awariom. W niniejszej pracy rozważane są trzy typy awarii: zatrzymanie, gubienie komunikatów oraz złośliwe zachowanie (Bizantyjskie).

- *Zatrzymanie* to awaria przy której dana jednostka przerywa swoje działanie i nie uczestniczy dalej w wykonaniu algorytmu, w szczególności nie wysyłając ani nie odbierając żadnych komunikatów. Modeluje to zjawiska takie jak przerwanie działania programu z powodu wyjątku, czy fizyczne wyłączenie jednostki.

- *Gubienie komunikatów* to awaria powodująca że niektóre komunikaty wysłane do danej jednostki nie są przez nią odbierane, a niektóre wysyłane przez nią nie docierają do odbiorców. Modeluje to zjawiska takie jak uszkodzenie sieci, błędy interfejsów itp.

- *Złośliwe zachowanie* to ogólna kategoria błędów obejmująca dowolne odstępstwa działania jednostki od założonego algorytmu. W szczególności obejmuje wysyłanie komunikatów niepoprawnych czy zawierających nieprawidłowe dane. Modeluje to przejęcie kontroli nad jednostką przez inny algorytm.

Dla każdego z tych typów awarii definiujemy klasę algorytmów które są odporne na ich występowanie. Dodatkowym parametrem tej klasy może być próg odporności t, określający przy ilu maksymalnie awariach poprawność działania algorytmu jest zapewniona.

2 Schematy komunikacji

Przed zaprezentowaniem samych algorytmów, wprowadzamy kilka definicji.

Grafem komunikacyjnym będziemy nazywać dowolny graf nieskierowany, którego węzłami są jednostki obliczeniowe systemu rozproszonego. Każdy taki graf definiuje pewien *schemat komunikacji*, w którym jednostki komunikują się wyłącznie ze swoimi sąsiadami w tym grafie. Maksymalna liczba komunikatów jakie mogą być wysłane w jednej rundzie jest wtedy ograniczona przez dwukrotność liczby krawędzi tego grafu.

Nasze rozwiązania będziemy budować ze specyficznych schematów komunikacji, zbudowanych w oparciu o grafy o potrzebnych nam własnościach i o niewielkiej liczbie krawędzi. W tej sekcji prezentujemy potrzebne nam grafy.

2.1 Dystrybutor

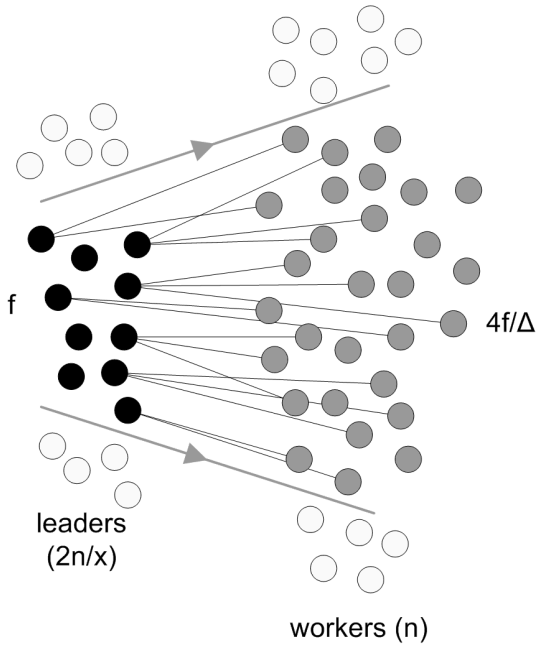
Dystrybutor jest grafem w którym mały zbiór wierzchołków L jest połączony z dużym zbiorem wierzchołków W niewielką liczbą krawędzi, w taki sposób żeby każdy wystarczająco duży podzbiór W miał dużo sąsiadów w zbiorze L. Formalna definicja jest następująca:

Niech $G = (W, L, E)$ będzie dwudzielnym grafem w którym W i L są rozłączymi zbiorami węzłów, a $E \subseteq W \times L$ jest zbiorem krawędzi. Powiemy że G jest (n, x, Δ)-*dystrybutorem*, jeśli spełnia trzy warunki:

- $|W| = n$, $|L| = \lfloor \frac{2n}{x} \rfloor$;
- maksymalny stopień wierzchołków w zbiorze W wynosi Δ
- dla każdego $f < \frac{2n}{x^2}$, każdy zbiór $Y \subseteq W$ o rozmiarze $\frac{4f}{\Delta}$, ma co najmniej f sąsiadów

Twierdzenie 2.1 *Dla dowolnych parametrów n, $\Delta \geq 4$ i $x \geq 8$, istnieje (n, x, Δ)-dystributor.*

Szkic dowodu: Dla zadanych parametrów tworzymy losowy graf spełniający pierwsze dwa warunki i dowodzimy że prawdopodobieństwo spełnienia trzeciego warunku jest większe od zera. Argument probabilistyczny pokazuje, że istnieją zatem grafy spełniające wszystkie trzy warunki.



Rysunek 1: Dystrybutor. Każdy wystarczajšco duży podzbiór L (czarne) posiada duży zbiór sšsiadów w W

2.2 Komunikator

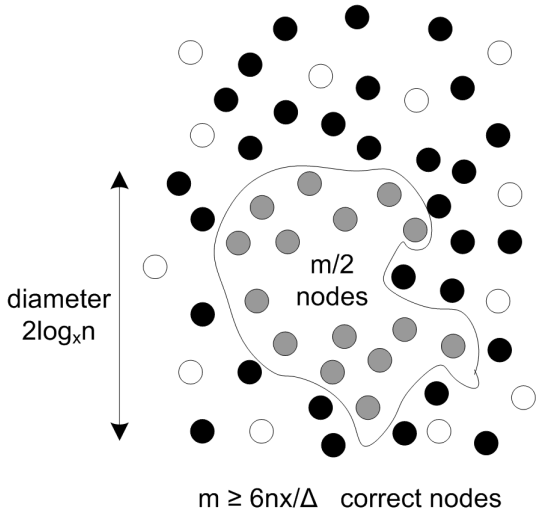
Komunikator jest grafem w którym każdy wystarczająco duży zbiór wierzchołków zawiera duży podzbiór o małej średnicy.

Formalnie, graf $G = (L, E)$ o n wierzchołkach i stopniu Δ, nazywamy (n, x, Δ)-*komunikatorem*, jeśli:

> Dla dowolnego $B \subseteq L$ rozmiaru $m \geq \frac{6nx}{\Delta}$, istnieje $C \subseteq B$ zawierający co najmniej połowę wierzchołków z B i mający średnicę co najwyżej $2 \log_x n$.

Twierdzenie 2.2 *Dla dowolnych n, Δ i $x \geq 2 \log n$, istnieje (n, x, Δ)-komunikator.*

Szkic dowodu: Tworzymy losowy graf o stopniu Δ i rozważamy wszystkie jego podzbiory o wielkości $b = \frac{6nx}{\Delta}$. W każdym z tych podzbiorów liczymy wierzchołki które mają w tym zbiorze mniej niż $\frac{b}{2}$ sąsiadów w odległości $\log_x n$. Obliczamy jakie jest prawdopodobieństwo że takich wierzchołków jest więcej niż $\frac{b}{2}$. Jeśli takich wierzchołków jest mniej, pozostałe tworzą podgraf o zakładanej średnicy. Dowodzimy że prawdopodobieństwo równoczesnego takiego zdarzenia dla wszystkich podzbiorów losowego grafu jest wystarczająco duże, żeby istniał graf mający taką własność.



Rysunek 2: Komunikator: po usunięciu dowolnego zbioru wezłów (czarne), wśród pozostałych węzłów istnieje duży podzbiór o małej średnicy (szare)

2.3 Adaptatywny komunikator

Adaptatywny komunikator jest specjalnym typem komunikatora, w którym usunięcie niewielkiego zbioru wierzchołków nie może spowodować oddzielenia większego zbioru od głównej części grafu.

Graf $G = (L, E)$ o n wierzchołkach i stopniu Δ nazywamy (n, s, Δ)-*adaptatywnym-komunikatorem*, jeśli:

> Dla dowolnego $f \leq \frac{n}{s}$ i dowolnego zbioru $B \subseteq L$ o rozmiarze $n - f$, istnieje $C \subseteq B$ rozmiaru co najmniej $n - 2f$, taki że podgraf indukowany przez C w G ma średnicę co najwyżej $2 \log_d n$.

Twierdzenie 2.3 *Dla dowolnych n, $x \geq 6$ i $\Delta \geq 36$, istnieje (n, x, Δ)-adaptatywny-komunikator.*

Szkic dowodu: Pokazujemy że grafy Ramanujana [10] o wystarczająco dużym stopniu mają powyższą własność.

2.4 Podprocedury

Korzystając z powyższych grafów, możemy zdefiniować procedury z których będą korzystać nasze algorytmy:

Rozproszone zapytanie - to procedura korzystająca z dystrybutora jako grafu komunikacji. W procedurze tej mały zbiór liderów (L) wysyła zapytanie do wybranych

wierzchołków w dużym zbiorze i czeka na odpowiedzi. Własności dystrybutora gwarantują ograniczenie na liczbę wierzchołków które nie zostaną odpytane z powodu awarii niektórych liderów.

Mieszanie - to procedura korzystająca z komunikatora lub adaptatywnego komunikatora. W procedurze tej wierzchołki we wskazanym zbiorze wysyłają swoją wiedzę do sąsiadów przez ustaloną liczbę rund. W efekcie każdy wierzchołek zbiera wiedzę od wszystkich wierzchołków w ustalonej odległości. Własności grafu gwarantują wielkość zbioru wierzchołków które całkowicie wymieniają się wiedzą.

3 Plotkowanie

Pierwszym problemem rozważanym w tej pracy jest problem plotkowania. Jego definicja jest następująca: Inicjalnie każda jednostka posiada pewien zasób wiedzy (plotkę). W momencie zakończenia działania, każda jednostka powinna znać plotki wszystkich pozostałych jednostek. W przypadku wystąpienia awarii, wymaganie to jest ograniczane: każda poprawnie działająca jednostka musi znać plotki wszystkich poprawnie działających jednostek [9, 11].

Dla każdego z rozważanych trzech typów awarii przedstawiamy algorytm odporny na dany typ awarii. Algorytm odporny na zatrzymania działa dla dowolnej liczby awarii. Algorytmy odporne na gubienie komunikatów i złośliwe zachowania, parametryzowane są progiem t określającym przy ilu maksymalnie awariach poprawność działania jest gwarantowana. Koszt działania algorytmów uzależniony jest od liczby jednostek n, progu t oraz faktycznej liczby awarii które nastąpiły w trakcie wykonania f.

3.1 Zatrzymania

Zanim przedstawimy algorytm, udowodnimy dolne ograniczenie na złożoność komunikacyjną plotkowania przy awariach zatrzymania.

Twierdzenie 3.1 *Każdy algorytm plotkowania odporny na zatrzymania i kończący działanie w ciągu c rund, musi używać $\Omega(n+fn^{\frac{1}{c+1}})$ komunikatów przy wystąpieniu f zatrzymań.*

Szkic dowodu: Rozważamy scenariusz w którym awarii ulega każda jednostka która otrzyma co najmniej $n^{1/(c+1)}$ komunikatów. Jeśli mniej niż połowa jednostek ulegnie awarii, to w ciągu c rund żadna jednostka nie będzie w stanie zebrać plotek wszystkich poprawnie działających jednostek. Tym samym żaden algorytm używający mniej niż $fn^{\frac{1}{c+1}}$ komunikatów, nie może kończyć się poprawnie w takim scenariuszu.

W szczególności, dla algorytmów działających w czasie stałym otrzymujemy następujący wniosek.

Wniosek 3.2 *Każdy algorytm plotkowania odporny na zatrzymania i działający w stałym czasie, musi używać* $\Omega(n+fn^{\varepsilon})$ *komunikatów przy wystąpieniu* f *zatrzymań, dla pewnej stałej* $\varepsilon > 0$.

Twierdzenie 3.3 *Dla każdego* $s \geq 2$*, istnieje algorytm plotkowania odporny na* $n-1$ *zatrzymań, działający w czasie* $O(\log_s^3 n)$ *i używający* $O(ns^2 \log_s^3 n)$ *komunikatów.*

Zaprezentowany jest dwuczęściowy algorytm. Pierwsza część przeprowadza próbę plotkowania przy użyciu $\frac{n}{s}$ liderów. Ta część wymaga $O(n \log_s n)$ komunikatów i gwarantuje poprawność jeśli co najmniej połowa liderów zadziała poprawnie. W drugiej części biorą udział tylko te jednostki które nie zebrały wystarczającej liczby plotek w pierwszej części. Ta część jest podzielona na $\log_s n$ faz. W każdej kolejne fazie każda aktywna jednostka wysyła s razy więcej komunikatów niż w poprzedniej. Jeśli w którejkolwiek fazie więcej niż $\frac{1}{s}$ początkowych jednostek zadziała poprawnie, faza ta poprawnie wykonuje plotkowanie i w kolejnych fazach jednostki nie wysyłają już komunikatów. Zatem jeśli w ostatniej fazie co najmniej jedna jednostka działa poprawnie, plotkowanie jest zakończone. Maksymalna liczba komunikatów wysłanych w każdej fazie jest ograniczona przez $O(ns^2 \log_s^2 s)$.

Twierdzenie 3.4 *Dla każdego* n *i dowolnego* $\varepsilon > 0$ *istnieje algorytm plotkowania działający w czasie stałym, odporny na* $n-1$ *zatrzymań i używający* $O(n + fn^{\varepsilon})$ *komunikatów gdy wystąpi* f *zatrzymań.*

Poszukiwanym algorytmem jest przedstawiony powyżej algorytm dla $s = n^{\varepsilon}$.

3.2 Gubienie komunikatów

Dolne ograniczenie na liczbę komunikatów potrzebnych do przeprowadzenia plotkowania w przypadku wystąpienia błędów gubienia komunikatów przedstawia następujące twierdzenie:

Twierdzenie 3.5 *Każdy algorytm plotkowania odporny na* t *błędów gubienia komunikatów musi używać* $\Omega(n + ft)$ *wiadomości gdy wystąpi* f *błędów.*

Szkic dowodu: Rozważamy scenariusz w którym istnieje zbiór f jednostek nie odbierających żadnych komunikatów, ponieważ albo same są wadliwe albo jednostki wysyłające do nich komunikaty są wadliwe. Jeśli do którejkolwiek jednostki w tym zbiorze zostanie wysłanych mniej niż $t - f$ komunikatów, jest możliwe że jednostka ta działa poprawnie i nigdy nie otrzymuje żadnego komunikatu - a tym samym plotkowanie nie może zakończyć się poprawnie. W przypadku gdy $f < \frac{t}{2}$, otrzymujemy dowód że żaden algorytm wysyłający mniej niż $\frac{ft}{4}$ komunikatów nie może działać poprawnie w takim scenariuszu.

Twierdzenie 3.6 *Dla każdego $t < n$, istnieje algorytm plotkowania odporny na t błędów gubienia komunikatów, działający w stałym czasie i używający $O(n + tf)$ wiadomości gdy wystąpi f błędów.*

Przedstawiony jest algorytm działający w kilku fazach. W pierwszej, nazwanej *małym mieszaniem*, jednostki wymieniają się informacjami korzystając z małego zbioru liderów, podobnie jak w pierwszej części algorytmu dla błędów zatrzymań. Następnie jednostki które nie zebrały wystarczającej ilości plotek wykonują *duże mieszanie*, wymieniając się informacjami przy użyciu komunikatora zdefiniowanego na wszystkich jednostkach. Celem tych dwóch faz jest uzyskanie grupy jednostek które znają prawie wszystkie plotki, tzn. co najmniej $n - 6t$ plotek. W kolejnej fazie część jednostek z tej grupy wysyła zapytania do wszystkich jednostek których plotek do tej pory nie udało im się zebrać i czeka na odpowiedzi. W ten sposób uzyskiwany jest zbiór referencyjny - który posiada dla każdej jednostki albo jej plotkę, albo ponad t potwierdzeń o niemożliwości komunikacji. Następnie wykonywane jest ponownie małe i duże mieszanie, aby większość jednostek zebrała informacje od zbioru referencyjnego. W ostatniej fazie te jednostki którym nie udało się zebrać tej informacji wykonują zapytanie bezpośrednio do zbioru referencyjnego.

Algorytm działa w stałej liczbie rund i każda faza używa $O(n + tf)$ komunikatów.

3.3 Złośliwe zachowania

Złośliwe zachowania jednostek, najbardziej utrudniając przeprowadzenie plotkowania, wymuszają bardzo wysokie dolne ograniczenia na koszt komunikacyjny.
Ograniczenia te były już w literaturze przedstawiane [5]. Tutaj podajemy jedynie prosty dowód pokazujący że w przeciwieństwie do pozostałych typów awarii, dolne ograniczenie nie zależy od ilości błędów które faktycznie wystąpią w trakcie wykonania algorytmu (co oznacza że nie istnieje algorytm adaptacyjny dla tego problemu).

Twierdzenie 3.7 *Każdy algorytm plotkowania odporny na t złośliwych zachowań, musi używać $\Omega(nt)$ wiadomości, nawet gdy w wykonaniu nie wystąpi ani jedna awaria.*

Dowód opiera się na prostej obserwacji, że każda jednostka może uzyskać jedynie taką wiedzę jaką zechcą jej przekazać te jednostki z którymi się komunikuje. Jeśli jakakolwiek jednostka otrzymuje informacje od co najwyżej t innych, nie może wykluczyć że wszystkie te jednostki są wadliwe i przekazana przez nie informacja jest fałszywa. Aby móc zakładać że uzyskane dane są poprawne, każda jednostka musi otrzymać komunikaty od ponad t innych, co wymusza użycie nt komunikatów nawet gdy wszystkie jednostki działają prawidłowo.

Wysokie dolne ograniczenie na koszt algorytmu powoduje że optymalny algorytm jest znacznie prostszy od przedstawianych dla pozostałych typów awarii. Cytujemy go tutaj za pracą [5].

Twierdzenie 3.8 [5] *Dla każdego t istnieje algorytm plotkowania odporny na t złośliwych zachowań, działający w stałym czasie i używający $O(nt)$ wiadomości.*

W algorytmie uczestniczy $2t + 1$ liderów, którzy zbierają plotki z całego systemu i następnie rozsyłają je do wszystkich jednostek w systemie. Jeśli jakaś jednostka uzyska od liderów sprzeczne informacje, wybiera informację pojawiającą się najczęściej. Ponieważ przynajmniej $t + 1$ liderów działa prawidłowo, prawdziwe dane zawsze stanowią większość. Jeśli $2t + 1 > n$, powyższej operacji nie da się przeprowadzić. Wtedy po prostu każda jednostka wysyła swoją plotkę do wszystkich jednostek w systemie.

3.4 Problemy otwarte

Przedstawione wyniki rozstrzygają kwestię złożoności problemu plotkowania w większości rozważanych przypadków. Wśród nierozstrzygniętych szczególnie interesująca jest kwestia trade-offu pomiędzy czasem a złożonością komunikacyjną problemu. Przedstawiony algorytm pokazuje ten trade-off dla krótkich czasów (od stałego do logarytmicznego), uzyskując minimalną złożoność $O(n \log^3 n)$. Nie wiadomo czy dla dłuższych czasów da się uzyskać mniejszą złożoność. W szczególności otwarta pozostaje kwestia rozwiązania plotkowania przy użyciu $O(n)$ komunikatów dla dowolnej liczby awarii, nawet jeśli dopuścimy wykładniczy czas działania.

Nasze algorytmy są zaprojektowane dla synchronicznego systemu, ale ich podstawowe idee prawdopodobnie mogą być zastosowane również w systemach tylko częściowo synchronicznych [4]. Należy pamiętać że w systemie całkowicie asynchronicznym sam problem plotkowania według powyższej definicji jest nierozwiązywalny, ponieważ żadne jednostki nie są w stanie zakończyć działania jeśli brakuje im jakiś plotek, nie będąc w stanie odróżnić jednostki która się zatrzymała od takiej, której komunikaty wędrują wystarczająco wolno.

Kolejnym kierunkiem w którym można rozwinąć te badania jest kwestia wielokrotnego używania plotkowania na tym samym zbiorze jednostek. W przypadku błędów typu zatrzymań, jednostki wadliwe są natychmiast rozpoznawane przez sąsiadów i usuwane z dalszego obliczenia. W przypadku złośliwych zachowań nic takiego nie jest możliwe, ponieważ dodatkowy koszt nie zależy od ilości jednostek które faktycznie działają nieprawidłowo. W przypadku awarii gubienia komunikatów kwestia pozostaje otwarta. Choć teoretycznie do oznaczenia jakiejś jednostki jako wadliwej wystarczy informacja od $t + 1$ jednostek o tym że nie można nawiązać z nią komunikacji, w praktyce żadna jednostka może tyle komunikatów nigdy nie zgubić. Aby czynnik $O(tf)$ nie pojawiał się w koszcie każdego kolejnego wywołania, algorytm

działający wielokrotnie powinien w jakiś sposób unikać wysyłania komunikatów do jednostek "podejrzanych", które zgubiły wcześniej wiele komunikatów.

4 Konsensus

Druga część pracy dotyczy problemu konsensusu. Jego definicja jest następująca: Inicjalnie każda jednostka ma jakąś opinię (wartość z jakiegoś zbioru). Na końcu wszystkie jednostki muszą mieć jedną opinię, wybraną spośród pierwotnie posiadanych. W przypadku wystąpienia błędów wymagane jest aby na końcu wszystkie poprawnie działające jednostki posiadały taką samą opinię.

4.1 Algorytm deterministyczny

Twierdzenie 4.1 *Dla każdego $t < \frac{n}{3}$, istnieje deterministyczny algorytm konsensusu odporny na t zatrzymań, działający w czasie $O(t)$ i używający wiadomości o łącznej długości $O(n \log^2 n)$ bitów.*

Działanie zaproponowanego algorytmu opiera się na pojęciu *dobrze połączonej większości* w grafie. Jej definicja jest następująca: załóżmy że będziemy usuwać z grafu wszystkie wierzchołki które mają mniej niż c sąsiadów. Usunięcie jednych może spowodować że kolejne będą miały mniej niż c sąsiadów i wtedy je również usuwamy. Jeśli po tym procesie pozostanie nam spójny zbiór zawierający ponad połowę początkowych wierzchołków, nazywamy ten zbiór *dobrze połączoną większością*.

Do naszego algorytmu potrzebujemy grafu komunikacji o niewielkiej liczbie krawędzi oraz mającego następującą własność: jego dowolny podzbiór o $\frac{2n}{3}$ elementach zawiera dobrze połączoną większość o niewielkiej średnicy. Dowodzimy istnienia takich grafów za pomocą metody probabilistycznej, analogicznie jak w przypadku komunikatorów. Używamy rodziny takich grafów jako grafów komunikacji w naszym algorytmie.

Sam algorytm składa się z trzech faz:

1. **Faza propagacji głosów**
   W tej fazie jednostki mające głos o wartości 1 wysyłają go do sąsiadów w grafie komunikacji, a każda jednostka która otrzyma 1 przyjmuje go jako swój i dalej propaguje do sąsiadów. Ta faza trwa przez $12t + 23 \log n$ rund. Wynikiem tej fazy jest że wszystkie jednostki w dobrze połączonej większości mają identyczny głos.

2. **Faza wykrywania większości**
   W tej fazie jednostki analizują komunikacje z sąsiadami aby wykryć czy należą do dobrze połączonej większości. Faza trwa przez $23 \log n$ rund.

W każdej rundzie, każdy procesor który otrzymał co najmniej 2δ komunikatów wysyła komunikaty do sąsiadów. Procesor który otrzymał mniej komunikatów nie wysyła nic. Po zakończeniu fazy wszystkie procesory które w ostatniej rundzie wysyłały komunikaty, zatwierdzają posiadany głos. Faza zapewnia że tylko jednostki z dobrze połączonej większości zatwierdzą posiadane głosy, co dzięki poprzedniej fazie oznacza że wszystkie zatwierdzone głosy będą identyczne.

3. **Faza wykrywania decyzji**
   W tej fazie jednostki które jeszcze nie znają zatwierdzonej decyzji próbują się skontaktować z jednostkami które już ją znają. Faza trwa $2 \log n$ rund. Co co drugiej rundzie każda jednostka która nie zna jeszcze zatwierdzonej decyzji wysyła komunikaty do sąsiadów w grafach o coraz wyższym stopniu. Jednostki które znają zatwierdzoną wartość odpowiadają. Pod koniec tej fazy wszystkie poprawnie działające jednostki znają zatwierdzoną wartość.

4.2 Algorytm kwantowy

Twierdzenie 4.2 *Dla każdego $k > 0$ istnieje kwantowy algorytm konsensusu odporny na $\frac{n}{3}$ zatrzymań, działający w czasie $O(k \log n)$, używający komunikatów o łącznej długości $O(kn \log^3 n)$ qubitów i kończący się poprawnie z prawdopodobieństwem co najmniej $1 - 2^{-k}$.*

Działanie algorytmu opiera się na kwantowej wersji wyboru losowego głosu. W tym algorytmie każda jednostka losuje liczbę z ustalonego dużego zakresu i wysyła razem ze swoim głosem. Jako końcową decyzję, każda jednostka wybiera ten spośród otrzymanych głosów, do którego dołączona jest najwyższa wartość. Algorytm ten ma duże prawdopodobieństwo sukcesu w losowym przypadku, ale adaptatywny przeciwnik może łatwo doprowadzić do jego błędnego zakończenia, zatrzymując jednostkę która wylosowała najwyższą wartość w trakcie rozsyłania wiadomości. Kwantowa wersja tego algorytmu [3] polega na rozesłaniu superpozycji zamiast losowej liczby, tak że wartość przypisana do każdego głosu jest ustalana dopiero po dostarczeniu wszystkich wiadomości. W takiej sytuacji żaden przeciwnik nie jest w stanie przewidzieć która z wartości zostanie ostatecznie wybrana i algorytm zawsze działa tak jak w losowym przypadku. Jego powtórzenie odpowiednią liczbę razy pozwala uzyskać dowolnie poprawny wynik z dowolnie dużym prawdopodobieństwem.

Przedstawione tutaj usprawnienie kwantowego algorytmu polega na istotnym zmniejszeniu liczby wysyłanych komunikatów i ich sumarycznej dlugości (z $\Theta(n^2)$ do $\Theta(n \log n)$, za cenę przedłużenia działania o czynnik $O(\log n)$. Używany do tego jest graf komunikacji zdefiniowany dla algorytmu deterministycznego. Wszystkie operacje w trakcie działania algorytmu są przeprowadzane na superpozycjach otrzymywanych wartości, bez ich odczytywania. W efekcie powstaje stan splątany

obejmujący pamieć wszystkich jednostek biorących udział w algorytmie. Jego pomiar z dużym prawdopodobieństwem prowadzi do spójnego wyniku odczytanego przez większość jednostek, niezależnie od tego które z nich uległy awarii.

Algorytm składa się z dwóch faz:

1. **Faza propagacji głosów**
   W tej fazie jednostki przygotowują superpozycję wszystkich możliwych wartości losowych, dołączają je do swoich głosów i rozsyłają do sąsiadów w grafie komunikacji. Po otrzymaniu każdej wiadomości jednostka nie odczytuje jej, a zamiast tego odpowiednio splątuje jej wartość z dotychczas przechowywaną wartością. Faza trwa $23 \log n$ rund. Po zakończeniu każda jednostka odczytuje posiadaną wartość. Każda jednostka która przez wszystkie rundy otrzymywała co najmniej δ wiadomości, zatwierdza otrzymaną wartość jako obowiązującą.

2. **Faza wykrywania decyzji**
   Ta faza jest identyczna jak trzecia faza algorytmu deterministycznego. Przez $2 \log n$ rund, jednostki które jeszcze nie znają zatwierdzonej decyzji wysyłają komunikaty do sąsiadów w grafach o coraz wyższym stopniu i czekają na odpowiedzi.

Wykonanie tych dwóch faz zapewnia poprawne wykonanie konsensusu z prawdopodobieństwem co najmniej $\frac{1}{2}$. Algorytm może być powtórzony dowolnie wiele razy, tak aby uzyskać wystarczająco wysokie prawdopodobieństwo sukcesu.

4.3 Problemy otwarte

Przedstawione algorytmy pokazują możliwość znalezienia w rozproszonym systemie grupy większościowej, zdolnej do podjęcia wspólnej decyzji, bez zebrania globalnej informacji kto do tej grupy należy. Każda jednostka, na podstawie odbieranych komunikatów, lokalnie decyduje czy należy do tej grupy i odpowiednio dostosowuje swoje zachowanie. Umożliwia to znaczne zmniejszenie ilości przekazywanych w systemie komunikatów. Wadą tych rozwiązań jest odporność jedynie na ograniczoną liczbę błędów. Niezależnie od metody wyboru grupy decydującej, jeśli potem znaczna część tej grupy zostanie usunięta, jej decyzja może nie być wiążąca dla reszty systemu. Interesującym kierunkiem badań byłoby więc zmodyfikowanie tego algorytmu, np. przez dodanie kolejnych faz, tak żeby uzyskać poprawność dla dowolnej liczby błędów bez drastycznego zwiększenia kosztu komunikacyjnego.

Przedstawiony algorytm kwantowy ma dodatkową wadę, polegającą na tym że konsensus uzyskiwany jest tam jedynie z odpowiednim prawdopodobieństwem. Mimo że to prawdopodobieństwo można wybrać dowolnie bliskie 1, bardziej przydatny byłby algorytm kończący się zawsze z poprawnym wynikiem. Istnieją techiki pozwalające przekształcić taki algorytm typu Monte Carlo w algorytm typu Las Vegas, gdzie wynik jest zawsze poprawny a jedynie czas działania może być nieustalony

[6]. Techniki te wymagają jednak użycia $O(n^2)$ komunikatów, na co w tym przypadku nie możemy sobie pozwolić. Możliwość uzyskania pełnej poprawności przy niższym koszcie komunikacyjnym pozostaje otwarta.

Kolejnym otwartym problemem jest dalsze poprawienie złożoności bitowej algorytmu consensusu. Dotychczas jedyne znane algorytmy o złożoności $O(n)$ działają w czasie wykładniczym. Niewykluczone że rozwinięcie technik zaprezentowanych w tej pracy mogłoby umożliwić stworzenie algorytmu który osiągałby to samo w czasie wielomianowym.

Literatura

[1] M.K. Aguilera, S. Toueg, A Simple Bivalency Proof that t-Resilient Consensus Requires t+1 Rounds, *Information Processing Letters*, 71 (1999) 155-158.

[2] H. Attiya, J. Welch, Distributed Computing, John Willey & Sons, 2004 (second edition)

[3] M. Ben-Or and A. Hassidim, Fast quantum byzantine agreement, *Proc. of the 37th Annual ACM Symposium on Theory of Computing (STOC)* 2005, pp. 481 - 485.

[4] C. Dwork, N. Lynch, and L. Stockmeyer, Consensus in the presence of partial synchrony, *Journal of ACM*, 35:2, 1988, pp. 288 - 323

[5] D. Dolev and R. Reischuk, Bounds on information exchange for Byzantine Agreement, *Journal of ACM*, 32:1, 1985, pp. 191 - 204.

[6] P. Feldman, S. Micali, An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement, *SIAM Journal of Computing*, 26 (4), 1997, pp. 873-933.

[7] M. Fisher, and N. Lynch, A lower bound for the time to assure interactive consistency, *Information Processing Letters*, 14, 1982, pp. 183 - 186.

[8] M. Fisher, N. Lynch, M. Paterson, Impossibility of distributed consensus with one faulty process, *A. ACM*, 32, 1985, pp. 374 - 382.

[9] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, Dissemination of information in communication networks: broadcasting, gossiping, leader election, and fault-tolerance, Theoretical Computer Science, EATCS Series, Springer, 2005

[10] A. Lubotzky, R. Phillips, and P. Sarnak, Ramanujan graphs, *Combinatorica*, 8, (1988), pp. 261 - 277

[11] N. Lynch, Distributed Algorithms, Morgan Kaufmann Publishers, 1996

[12] M. Saks, N. Shavit, H. Woll, Optimal time randomized consensus - making resilient algorithms fast in practice, *Proc. of* 2*nd SIAM-ACM Symposium on Discrete Algorithms (SODA)*, 1991, pp. 351 - 362.