AKADEMIA GÓRNICZO-HUTNICZA

IM. STANISŁAWA STASZICA W KRAKOWIE



Wydział Inżynierii Metali i Informatyki Przemysłowej



PROJEKT ZALICZENIOWY

pt.

„Zagadnienie uzgadniania w systemach rozproszonych”

Zespół w składzie: **Krystian Cymerys**

**Konrad Boguń**

**Lucyna Hajder**

**Marcin Czanasz**

**Michał Góra**

**Karol Gałęzia**

Kierunek studiów: **Informatyka Stosowana**

Kraków 2016

**Spis treści**

[**1.** **Analiza zagadnienia** 3](#_Toc451346144)

[**2.** **Algorytmy rozwiązujące zagadnienie uzgadniania** 3](#_Toc451346145)

[**3.** **Analiza porównawcza algorytmów** 3](#_Toc451346146)

[**4.** **Opis sposobu oceniania algorytmów oraz wybór najlepszego** 3](#_Toc451346147)

[**5.** **Technologie rozproszone dostępne na rynku** 3](#_Toc451346148)

[**6.** **Analiza porównawcza implementacji** 3](#_Toc451346149)

[**7.** **Wnioski** 3](#_Toc451346150)

[**8.** **Bibliografia** 3](#_Toc451346151)

# **Analiza zagadnienia**

Problem uzgadniania, czyli innymi słowy konkretny problem osiągnięcia konsensusu, polega na tym, iż każdy węzeł w systemie rozproszonym posiada swoją wartość początkową. Następnie wszystkie węzły muszą zgodzić się na jedną z wartości określonych poprzez poszczególne pozostałe węzły. W przypadku kiedy w systemie nie dochodzi do awarii algorytm rozwiązania problemu uzgadniania jest prosty do wytłumaczenia i zrozumienia, a mianowicie: każdy z istniejących węzłów w systemie wysyła wybraną przez siebie wartość, przypuśćmy decyzję, do pozostałych węzłów. Wykonywany jest wówczas algorytm wyboru większościowego. Rezultatem takiego działania jest osiągnięcie porozumienia pomiędzy węzłami, ponieważ dysponują one takimi samymi danymi oraz wykonują ten sam algorytm, co prowadzi do zdecydowania się na tą samą wartość vel decyzję.

W niniejszym problemie istnieje możliwość wystąpienia dwóch rodzajów awarii:

* Załamania (crash failures) – awaria ta polega na tym, iż pewien węzeł przestaje wysyłać komunikaty
* Awarie bizantyjskie – polegają na tym, że istnieje jeden lub więcej wadliwy węzeł, który wysyła dowolne komunikaty

Sformułowanie problemu uzgadniania można przedstawić w sposób obrazowy, jak to jest dokonywane w większości literatury. Z tego powodu na potrzeby niniejszego projektu również skorzystano z tego sformułowania:

„Grupa armii bizantyjskich otacza miasta nieprzyjaciela. Rozkład sił jest taki, że jeśli wszystkie armie zaatakują razem, to będą w stanie zdobyć miasto. Innym sposobem uniknięcia porażki jest odwrót wszystkich armii. Generałowie poszczególnych armii mają zaufanych posłańców, którzy z powodzeniem dostarczą każdy komunikat od jednego generała do innego. Jednak niektórzy generałowie mogą być zdrajcami usiłującymi doprowadzić do porażki armii bizantyjskich. Należy opracować taki algorytm, który umożliwi wszystkim wiernym generałom uzgodnienie pewnego planu działania. Ostateczna decyzja powinna być z grubsza taka, jaka zostałby podjęta w drodze głosowania większościowego nad decyzjami poszczególnych generałów. W przypadku nierozstrzygnięcia głosowania końcową decyzją ma być odwrót.” [[[1]](#endnote-1)].

W systemie rozproszonym odpowiednikiem generałów będą węzły, natomiast posłańcy będą odpowiadać kanałom komunikacyjnym.

//MOŻNA JESZCZE COŚ DOPISAĆ

# **Algorytmy rozwiązujące zagadnienie uzgadniania**

|  |  |  |
| --- | --- | --- |
| Nazwa algorytmu | Twórca | Rok powstania |
| Algorytm generałów bizantyjskich [[[2]](#endnote-2)] | Leslie Lamport  Robert Shostak  Marshall Pease | 1982 |
| Algorytm rozpływowy (flooding) |  |  |
| Algorytm króla | Piotr Berman  Juan Garay |  |
| Algorytm jednorundowy |  |  |
| Algorytm randomizowany (randomized algorithm) |  |  |
| Algorytm dwóch armii |  |  |
| Algorytm królowej (The Queen Algorithm) |  |  |
| Algorytm tratwy (Raft algorithm) |  |  |
| Rozgłoszeniowy algorytm konsensusu podstawowego |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Algorytm generałów bizantyjskich (The Byzantine Generals Problem) [[[3]](#endnote-3)] | Algorytm rozpływowy (flooding algoritm) | Algorytm randomizowany (Randomized algorithm) [[[4]](#endnote-4)] |
| ]skończoność (termination) | algorytm składa się z m+1 rund w pierwszej fazie, oraz z rundy podjęcia decyzji na podstawie otrzymanych danych w drugiej fazie. Rozwiązanie gwarantuje, że wszystkie poprawne procesy w końcu podejmą decyzję dochodząc tym samym do porozumienia | algorytm składa się z t+1 rund | algorytm składa się z stałej liczby rund. Jeżeli niektóre procesy nie mają ustawionej losowo lokalnej wartości, to ustawiają ją taką samą jak pozostałe. |
| zgodność (agreement) | wartość decyzji wszystkich poprawnych procesów jest taka sama. Jeśli p­i oraz pj są prawidłowe oraz podjęły decyzję, wówczas di = dj |  | zakładając, że niektóre poprawne procesy decydują o X. Proces ten musi uzyskać X z n-3f prawidłowych procesów. Każdy inny właściwy proces musi otrzymać X co najmniej n-4f razy (wszystkie poprawne procesy ustawiają swoje lokalne wartości X, oraz proponują ją, a także decydują się na X w następnej rundzie) |
| integralność (integrity) | jeśli dowódca jest poprawny, to wszystkie poprawne procesy decydują się na wartość decyzji, którą zaproponował dowódca |  | jeżeli wszystkie poprawne procesy posiadają tą samą wartość X, otrzymają n‑2f propozycji zawierających X w pierwszej rundzie oraz będą decydować o X |
| dla jakich kroków jest niemożliwy | poprawny tylko dla ilości procesów n>3m (zdrajców powinno być 3 razy mniej) [[[5]](#endnote-5)] |  | poprawny tylko dla binarnych danych wejściowych. Nie działa w przypadku kiedy istnieją procesy bizantyjskie. Działa tylko wtedy gdy istnieją f<n/9 crash failures |
| ilość wiadomości | n\*[(n‑1)\*n‑k)\*(n‑k‑1)] wiadomości [1] |  | ?? |
| złożoność obliczeniowa | O(Nf+1) – dla komunikatów niepodpisanych  O(N2) – dla komunikatów podpisanych [[[6]](#endnote-6)] |  | O(2n) – wszystkie poprawne procesy zdecydowały  O(1/2n-f) – najgorszy przypadek, kiedy wszystkie n-f poprawnych procesów wybiera tą samą wartość losowo |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Algorytm króla (Phase King Algorithm)[[[7]](#endnote-7)] | Algorytm królowej (The Queen Algorithm)[[[8]](#endnote-8)] | Algorytm dwóch armii |
| skończoność (termination) | *Algorytm zawiera 3(f + 1)* rund dla (f+1) faz.  Jeżeli król jest prawidłowy, to wszystkie prawidłowe procesy mają tą samą wartość. | *Algorytm zawiera 2(f + 1)* rund dla (f+1) faz.  Jeżeli królowa jest prawidłowa, to wszystkie prawidłowe procesy mają tą samą wartość. | Rozwiązanie może być nieskończone. Każdy proces decyduje o wartości po skończonej ilości wiadomości.[[[9]](#endnote-9)] |
| zgodność (agreement) | Wszystkie procesy mają wpływ na ostateczną wartość (króla).  Jeśli wszystkie poprawne procesy mają wartość x na początku fazy , to  nadal będą mieć x w końcowym etapie. | Wszystkie procesy mają wpływ na ostateczną wartość (królowej).  Jeśli wszystkie poprawne procesy mają wartość x na początku fazy , to  nadal będą mieć x w końcowym etapie. | Wszystkie poprawne procesy mają decydować o tej samej wartości.  Wartość jest wybrana przez proces, jeśli wystąpiła n/2 razy[[[10]](#endnote-10)] |
| integralność (integrity) | Wszystkie poprawne procesy otrzymują >= n-f odpowiedzi.  Król ustawia swoją wartość na taką, która wystąpiła więcej niż n - 2f razy i przesyła ją do wszystkich procesów. | Wszystkie poprawne procesy otrzymują >= n-f odpowiedzi.  Królowa ustawia swoją wartość na taką, która wystąpiła więcej niż n/2 + f razy i przesyła ją do wszystkich procesów. | Każdy proces otrzymuje n-f odpowiedzi. Jeśli otrzymana wartość wystąpiła >n/2 ilość, to proces zmienia wartość i przesyła ją dalej. |
| dla jakich kroków jest niemożliwy | poprawny tylko dla ilości procesorów n>3f (zdrajców powinno być 3 razy mniej) | poprawny tylko dla ilości procesorów n>4f (zdrajców powinno być 4 razy mniej) | Poprawny tylko dla ilości procesorów n > 2f |
| ilość wiadomości | (f + 1)[(n − 1)(n + 1)] wiadomości[1] | =(n-f)/2+(n-f)/2 > n/2+f wiadomości | Ilość wiadomości rośnie wykładniczo |
| złożoność obliczeniowa | O(t^3) | ??? | ??? |

|  |  |  |
| --- | --- | --- |
|  | Terminating Reliable Broadcast[[11]](#endnote-11) | k-set Consensus[[12]](#endnote-12) |
| Skończoność | Każdy poprawny proces dostarcza jakąś wiadomość | Każdy poprawny proces musi zdecydować się na jakąś wartość. |
| Ważność | Jeśli nadawca wiadomości m nie jest uszkodzony to wszystkie poprawne procesy dostarczają wiadomość m | Jeśli jakiś proces decyduje się na jakąś wartość to musi być to wartość zaproponowana przez inny proces.  Jeden z warunków:  STRONG:  1. Decyzja dowolnego prawidłowego procesu jest równa wartości początkowej jakiego poprawnego procesu.  2. Jeśli wszystkie poprawne procesy zaczynają z wartością v to poprawne procesy decydują się na wartość v.  REGULAR:  1. Decyzja dowolnego prawidłowego procesu jest równa wartości początkowej jakiego procesu.  2.Jeśli wszystkie procesy zaczynają z wartością v to poprawne procesy decydują się na wartość v.  WEAK:  1.Jeśli nie wystąpiły błędy to decyzja dowolnego procesu jest równa wartości początkowej jakiegoś procesu.  2.Jeśli nie wystąpiły błędy i wszystkie procesy zaczęły z wartością v to każdy proces decyduje się na wartość v.  tl;dr Jeśli jakiś proces decyduje się na jakąś wartość to musi być to wartość zaproponowana przez inny proces. |
| Integralność | Każdy poprawny proces dostarcza co najwyżej jedną wiadomość. Jeśli proces dostarcza wiadomość różną od null wtedy nadawca musi dostarczyć wiadomość m. |  |
| Zgodność | Jeśli poprawny proces dostarcza wiadomość m to każdy inny poprawny proces rówież dostarcza m. | Wszystkie poprawne procesy muszą podjąć decyzję. Zbiór wartości na które procesy mogą się zdecydować wynosi co najwyżej k (>f) wartości. |
| Złożoność | D+ƒd | ? |
|  | Poprawny proces zawsze otrzymuje wiadomość nawet jeżeli wystąpi błąd podczas wysyłania wiadomości (w tym przypadku proces dostaje wiadomość null). |  |

# **Analiza porównawcza algorytmów**

# **Opis sposobu oceniania algorytmów oraz wybór najlepszego (?)**

# **Technologie rozproszone dostępne na rynku**

# **Analiza porównawcza implementacji**

# **Wnioski**

# **Bibliografia (oczywiście trzeba będzie ją poprawić)**

1. Ben-Ari M., „Podstawy programowania współbieżnego i rozproszonego, wyd. 2 zm. i rozsz.”, rozdział 12, str. 237-240, Wydawnictwa Naukowo-Techniczne, 2009, Warszawa [↑](#endnote-ref-1)
2. The Byzantine Generals Problem (with Marshall Pease and Robert Shostak), ACM Transactions on Programming Languages and Systems 4, 3 July 1982, <http://research.microsoft.com/users/lamport/pubs/byz.pdf>, (dostęp: 22.05.2016) [↑](#endnote-ref-2)
3. Coulouris G., Dollimore J., Kindberg T., Gordon B., “Distributed Systems. Concepts and Design. Fifth Edition”, str. 662, Pearson Education, 2012, Boston [↑](#endnote-ref-3)
4. Mads Dam, “Parallel and Distributed Computing. Distributed Algorithms. Lecture 7: Consensus”, slajdy 72-82 oraz 93, <http://www.csc.kth.se/utbildning/kth/kurser/DD2451/pardis11/DD2451_lecture7.pdf>, (dostęp: 26.05.2016) [↑](#endnote-ref-4)
5. Nelson M., “The Byzantine Generals Problem”, artykuł: <http://marknelson.us/2007/07/23/byzantine/>, (dostęp: 22.05.2016) [↑](#endnote-ref-5)
6. “Systemy Równoległe i rozproszone”, wykład nr 13 – „Rozgłaszanie i uzgadnianie”, strona z materiałami dydaktycznymi dr hab. inż. K. Banasia, <http://www.metal.agh.edu.pl/~banas/SRR/SRR_W13_Rozglaszanie_Uzgadnianie.pdf>, (dostęp: 22.05.2016) [↑](#endnote-ref-6)
7. Ajay Kschemkalyani, Mukesh Singhal, „Distributed Computing: Principles, Algorithms and Systems. Chapter 14: Consensus and Agreement”, Cambridge University Press, <https://www.cs.uic.edu/~ajayk/Chapter14.pdf> (dostęp: 22.05.2016) [↑](#endnote-ref-7)
8. Mads Dam, “Parallel and Distributed Computing. Distributed Algorithms. Lecture 7: Consensus”, slajdy 48-55, <http://www.csc.kth.se/utbildning/kth/kurser/DD2451/pardis11/DD2451_lecture7.pdf>, (dostęp: 22.05.2016) [↑](#endnote-ref-8)
9. „Computer Science & Engineering. Courses. CSEP552: Distributed Systems.Lecture 5: Intro to consensus”, University of Washington, <http://courses.cs.washington.edu/courses/csep552/13sp/lectures/5/intro.pdf> ostatni dostęp 21.05.2016 [↑](#endnote-ref-9)
10. Paul Krzyzanowski, „Consensus. Reaching agreement”, <https://www.cs.rutgers.edu/~pxk/417/notes/content/consensus.html> (dostęp: 22.05.2016) [↑](#endnote-ref-10)
11. <https://www.cs.uic.edu/~ajayk/Chapter14.pdf> (dostęp 26.05.2016) [↑](#endnote-ref-11)
12. <https://www.cs.unc.edu/~reiter/papers/1999/PODC.pdf> (dostęp 26.05.2016) [↑](#endnote-ref-12)