Mechanizmy rozgłaszania niezawodnego

Plan wykładu

Celem wykładu jest prezentacja zagadnień związanych z implementacją niezawodnej komunikacji w zawodnym środowisku rozproszonym. Wykład obejmie omówienie abstrakcyjnych mechanizmów rozgłaszania wiadomości, takich jak: podstawowe rozgłaszanie niezawodne (ang. *best-effort*), zgodne rozgłaszanie niezawodne (ang. *regular reliable broadcast*), jednolite rozgłaszanie niezawodne (ang. *uniform reliable broadcast*) i probabilistyczne rozgłaszanie niezawodne (ang *probabilistic reliable broadcast*). Przedstawione zostaną wybrane algorytmy implementujące wspomniane operacje rozgłaszania wiadomości, w tym: algorytm podstawowego rozgłaszania niezawodnego pasywny i aktywny, algorytmy zgodnego rozgłaszania niezawodnego z potwierdzeniami od wszystkich i z potwierdzeniami od większości, aktywny i pasywny algorytm probabilistycznego rozgłaszania niezawodnego, algorytmy zgodnego rozgłaszania niezawodnego z przyczynowym i globalnym uporządkowaniem wiadomości.

Rozgłaszanie niezawodne – definicja nieformalna

Nieformalnie, rozgłaszanie to mechanizm komunikacyjny, za pomocą którego proces może wysłać wiadomość do grupy procesów. Mechanizmy rozgłaszania niezawodnego gwarantują pewne własności, pomimo występowania awarii. Gwarancje te tyczą się przede wszystkim zapewnienia dostarczenia wiadomości do adresatów, identyczności zbioru wiadomości dostarczanych do wszystkich procesów poprawnych, oraz kolejności dostarczanych wiadomości.

Przykładowe zastosowania mechanizmów niezawodnego rozgłaszania

Mechanizmy niezawodnego rozgłaszania mają bardzo wiele zastosowań. Stąd wynika ich duże znaczenie teoretyczne i praktyczne.

Przykład 1: Systemy z wieloma uczestnikami

> W systemach z wieloma uczestnikami (ang. *multi-participant systems*) działa wiele procesów, współuczestnicząc w skomplikowanych zadaniach. Przykładem są wszelkiego rodzaju wirtualne środowiska z wieloma użytkownikami (na przykład gry MMRPG – *massive multiplayer role-playing games*). Dostępność abstrakcyjnych operacji niezawodnego rozgłaszania jest przydatna dla twórców tego rodzaju systemów.

Przykład 2: Zwielokrotnianie (replikacja)

> Często stosowanym rozwiązaniem w celu zwiększenia odporności systemu na awarie, lub zwiększenia jego efektywności, jest zwielokrotnianie (replikacja) elementów. Pojawia się wówczas jednocześnie problem spójności replik. Konstrukcja protokołów zapewniających spójność replik jest znacząco uproszczone w wypadku możliwości skorzystania z mechanizmów niezawodnego rozgłaszania.

Klasy mechanizmów rozgłaszania niezawodnego

W dalszej części wykładu zostaną omówione abstrakcje (mechanizmy) komunikacyjne podstawowego, zgodnego, jednolitego i probabilistycznego rozgłaszania niezawodnego.

Nieformalnie, w podstawowym rozgłaszaniu niezawodnym (ang. *best-effort broadcast*) odpowiedzialność za dostarczenie wiadomości spoczywa na nadawcy, o którym zakłada się, że jest poprawny. W zgodnym rozgłaszaniu niezawodnym (ang. *regular reliable broadcast*) system gwarantuje identyczność zbioru wiadomości dostarczonych przez wszystkie poprawne procesy. Zostaną przedstawione dwa algorytmy implementujące ten mechanizm:

- pasywny algorytm zgodnego rozgłaszania niezawodnego (ang. *lazy reliable broadcast*),
- aktywny algorytm zgodnego rozgłaszania niezawodnego (ang. *eager reliable broadcast*)

W jednolitym rozgłaszaniu niezawodnym (ang. *uniform reliable broadcast*) dodatkowo gwarantuje się, że wiadomość dostarczona przez jeden proces (być może niepoprawny) zostanie dostarczona przez

wszystkie poprawne procesy. Przedstawione zostaną dwa algorytmy niezawodnego rozgłaszania jednolitego:

- algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (ang. *all-ack uniform reliable broadcast*)
- algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od większości (ang. *majority-ack uniform reliable broadcast*)

W probabilistycznym rozgłaszaniu niezawodnym (ang. *probabilistic reliable broadcast*) gwarancje dotyczące dostarczenia wiadomości są zachowane jedynie z pewnym prawdopodobieństwem.

W zgodnym rozgłaszaniu niezawodnym z przyczynowym uporządkowaniem wiadomości gwarantowane jest dodatkowo, że wiadomości przyczynowo zależne będą dostarczone do odbioru zgodnie z relacją poprzedzania.

W zgodnym rozgłaszaniu niezawodnym z globalnym uporządkowaniem wiadomości zapewniona jest natomiast taka sama kolejność dostarczania wiadomości do wszystkich poprawnych procesów.

Warto zaznaczyć, że własności uporządkowania przyczynowego i globalnego są niezależne. Oznacza to, że uporządkowanie przyczynowe nie implikuje i nie jest implikowane przez uporządkowanie globalne.

Nie są to oczywiście jedyne możliwe abstrakcyjne operacje rozgłaszania. Wspomnieć warto jeszcze choćby o quasi-niezawodnym rozgłaszaniu.

Podstawowe rozgłaszanie niezawodne: Specyfikacja

Najprostszym rodzajem rozgłaszania jest *podstawowe rozgłaszanie niezawodne* (ang. *best-effort broadcast*). Ten podstawowy mechanizm (abstrakcja) komunikacyjny składa odpowiedzialność za niezawodność na nadawcę. Nie zapewnia przy tym dostarczenia wiadomości do wszystkich procesów, jeżeli nadawca ulegnie awarii.

Mechanizm podstawowego rozgłaszania niezawodnego ma z definicji następujące własności: ważności, braku powielania, braku samogeneracji.

Własność *ważności* (ang. *best-effort validity*) oznacza, że jeżeli procesy P_i oraz P_j są poprawne, to każda wiadomość rozgłaszana przez P_i jest ostatecznie dostarczona do P_j. Ogólnie, jest to własność żywotności. Zauważmy, że mimo tej własności zbiory wiadomości dostarczanych przez różne poprawne procesy mogą się różnić.

Własność *braku powielania* (ang. *no duplication*) oznacza, że jeżeli wiadomość jest dostarczona, to jest dostarczona co najwyżej raz. Jest to ważna gwarancja: jej sens łatwo wyjaśnić za pomocą przykładu odwołującego się do operacji bankowych. Użytkownik składający zlecenie przelewu chciałby mieć pewność, że zostanie ono wykonane jedynie raz.

Własność *braku samogeneracji* (ang. *no creation*) oznacza, że jeżeli jakaś wiadomość jest dostarczona do procesu P_j, to została wcześniej rozgłoszona przez jakiś proces P_i. Innymi słowy, własność ta gwarantuje, że kanały nie generują samorzutnie wiadomości.

Te dwie ostatnie własności są czasami w literaturze przedmiotu określane jako *integralność* lub *zwartość* (ang. *integrity*).

Zauważmy, że własność braku samogeneracji oraz braku powielania są własnościami bezpieczeństwa.

Algorytm podstawowego rozgłaszania niezawodnego: Operacje komunikacyjne

Operację podstawowego rozgłaszania niezawodnego wiadomości M przez P_i do zbioru procesów $\mathcal{P}$ oznaczać będziemy przez $\mathbf{send}^{\mathrm{BRB}}(P_i, \mathcal{P}, M)$. Odpowiadające tej operacji zdarzenie zapisywać

będziemy jako $e\_send^{BRB}(P_i, \mathcal{P}, M)$. Analogicznie, przez $\mathbf{deliver}^{\mathrm{BRB}}(P_j, P_i, M)$ oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{BRB}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M wysłaną przez proces P_j do procesu aplikacyjnego P_i.

Algorytm podstawowego rozgłaszania niezawodnego: Założenia

Przedstawiony algorytm wykorzystuje niezawodne kanały komunikacyjne. Algorytm zakłada, że procesy działają deterministycznie i poprawnie do ewentualnego załamania (ang. *crash*). Algorytm nie odwołuje się jednak do detektorów awarii, a więc przyjmuje model przetwarzania z ukrytymi awariami (ang. *fail silent*).

Algorytm podstawowego rozgłaszania niezawodnego (1)

W rozważanym algorytmie wiadomości aplikacyjne są umieszczone w pakietach typu PACKET.

Pakiety są przesyłane między monitorami. Z odebranych pakietów monitory wyodrębniają wiadomości aplikacyjne i przekazują je procesom aplikacyjnym.

Algorytm podstawowego rozgłaszanie niezawodnego (2)

Przypomnijmy, że $\mathbf{send}^{\mathbf{PL}}(Q_i, Q_j, pcktOut)$ jest mechanizmem kanałów niezawodnych. Operacja ta gwarantuje zatem, że każda wiadomość wysłana przez poprawnie działający proces, ostatecznie dotrze do adresata i umożliwi wówczas zajście zdarzenia $e\_receive^{PL}(Q_j, Q_i, M)$ oraz wykonanie dalej operacji $\mathbf{deliver}^{\mathbf{BRB}}(P_j, P_i, M)$. Warto przypomnieć, że w algorytmie implementującym kanały niezawodne monitor otrzymując dowolną wiadomość dostarcza ją tylko wtedy, jeżeli nie była już wcześniej dostarczona.

W przedstawionym algorytmie podstawowego rozgłaszania niezawodnego, zajście zdarzenia $e\_send^{BRB}(P_i, \mathcal{P}, msgOut)$ powoduje wysłanie przez monitor Q_i procesu P_i wiadomości kolejno do wszystkich procesów zbioru $\mathcal{P}$. Odebranie z kolei pakietu przez monitor Q_i adresata P_i, implikuje przekazanie wiadomości wyodrębnionej z pakietu procesowi P_i.

Algorytm podstawowego rozgłaszania niezawodnego: Złożoność

Niech topologią rozważanego przetwarzania rozproszonego jest graf w pełni połączony. Przy tym założeniu, rozgłaszanie wiadomości wymaga jednego kroku algorytmu związanego z wysyłaniem wiadomości. Ponieważ z kolei algorytm realizujący podstawowe rozgłaszanie niezawodne wymaga, by nadawca wysłał wiadomość do wszystkich procesów (łącznie z sobą samym), jego złożoność komunikacyjnego wynosi *n*, gdzie *n* jest liczbą procesów przetwarzania rozproszonego.

Ilustracja podstawowego rozgłaszania niezawodnego

W przedstawionym przykładzie proces P_1 chce rozesłać wiadomość M. Podstawowe rozgłaszanie niezawodne polega na wysyłaniu tej wiadomości oddzielnie do każdego procesu należącego do zbioru procesów $\mathcal{P}$, korzystając oczywiście z niezawodnych kanałów. Przypomnijmy, że niezawodne kanały autonomicznie ponawiają transmisję, pod warunkiem jednak, że proces nadawcy działa poprawnie.

Zgodne rozgłaszanie niezawodne: Specyfikacja

Mechanizm zgodnego rozgłaszania niezawodnego (ang. *regular reliable broadcast*) zapewnia dostarczenie wiadomości nawet w przypadku, gdy nadawca ulegnie awarii, jeżeli tylko jakikolwiek proces poprawny odebrał rozgłaszaną wiadomość.

Mechanizm ten ma z definicji następujące własności: ważności, braku powielania, braku samogeneracji, zgodności.

Własność *ważności* (ang. *validity*) oznacza, że jeżeli proces P_i jest poprawny, to każda wiadomość rozgłaszana przez ten proces jest ostatecznie dostarczona do P_i. Warto zauważyć różnicę tej własności w porównaniu z własnością ważności podstawowego rozgłaszania niezawodnego.

Własność *braku powielania* (ang. *no duplication*) oznacza, że jeżeli wiadomość jest dostarczona, to jest dostarczona co najwyżej raz.

Własność *braku samogeneracji* (ang. *no creation*) oznacza, że jeżeli jakaś wiadomość jest dostarczona do procesu P_j, to została wcześniej rozgłoszona przez jakiś proces P_i.

Własność *zgodności* (ang. *agreement*) oznacza, że jeżeli jakaś wiadomość została odebrana przez pewien poprawny proces P_i (dostarczona do pewnego poprawnego procesu P_i), to ostatecznie wszystkie poprawne procesy odbiorą tę wiadomość. Ta własność określa podstawową różnicę w stosunku do podstawowego rozgłaszania niezawodnego, gdyż zapewnia, że wiadomości mogą zostać dostarczone nawet w przypadku, gdy ich nadawca uległ awarii. Równocześnie, własność ta w połączeniu z własnością ważności oznacza, że wiadomości rozgłaszane przez poprawny proces ostatecznie zostaną odebrane przez wszystkie poprawne procesy.

Zgodne rozgłaszanie niezawodne określane jest też często jako *niepodzielne* lub *atomowe* (ang. *atomic broadcast*), gdyż gwarantuje, że albo wszystkie poprawne procesy odbiorą rozgłaszaną wiadomość, albo żaden z nich.

Zgodne rozgłaszanie niezawodne: Operacje komunikacyjne

Operację zgodnego rozgłaszania niezawodnego wiadomości M przez P_i do zbioru procesów $\mathcal{P}$ oznaczać będziemy przez $\mathbf{send}^{\mathbf{RRB}}(P_i, \mathcal{P}, M)$. Odpowiadające tej operacji zdarzenie zapisywać będziemy jako $e\_send^{RRB}(P_i, P_j, M)$. Analogicznie, przez $\mathbf{deliver}^{\mathbf{RRB}}(P_j, P_i, M)$ oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{RRB}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M wysłaną przez proces P_j do procesu aplikacyjnego P_i.

Pasywny algorytm zgodnego rozgłaszania niezawodnego: Założenia

Pasywny algorytm zgodnego rozgłaszania niezawodnego (ang. *lazy reliable broadcast*) używa doskonałego detektora awarii oraz mechanizm podstawowego rozgłaszania niezawodnego. Algorytm przeznaczony jest zatem dla modelu przetwarzania z jawnymi awariami (ang. *fail-stop*). Zagwarantowanie własności zgodności uzyskuje się dzięki retransmisjom wiadomości, dokonywanym w chwili wykrycia awarii nadawcy przez doskonały detektor awarii.

Pasywny algorytm zgodnego rozgłaszania niezawodnego (1)

W rozważanym algorytmie wiadomości aplikacyjne są umieszczane w pakietach typu PACKET, zawierających dodatkowo identyfikator pierwotnego (oryginalnego) nadawcy wiadomości.

Pasywny algorytm zgodnego rozgłaszania niezawodnego (2)

Zmienne $pcktIn$ oraz $msgOut$ posiadają swoje zwykłe znaczenie. Zbiór $correct_i$ zawiera identyfikatory procesów uznawanych za poprawne przez proces P_i. Zbiór $delivered_i$ tworzą wiadomości, które już zostały odebrane przez P_i. Tablica $from_i$ zawiera zbiory, których elementami są pary składające się z identyfikatora pierwotnego nadawcy wiadomości i samej wiadomości aplikacyjnej. Odpowiednio, *j*-ty element tej tablicy to wiadomości przekazane przez monitor Q_j. Zmienne pId oraz msg są używane lokalnie w programie obsługi zdarzenia wykrycia awarii procesu.

Pasywny algorytm zgodnego rozgłaszania niezawodnego (3)

Monitor procesu P_i chcącego rozesłać wiadomość M, umieszcza ją w pakiecie zawierającym dodatkowo pole $origin$, oznaczające pierwotnego (oryginalnego) nadawcę wiadomości i rozsyła otrzymany pakiet za pomocą podstawowego rozgłaszania niezawodnego. Przypomnijmy, że $\mathcal{Q}$ oznacza zbiór monitorów wszystkich procesów tworzących zbiór $\mathcal{P}$.

Pasywny algorytm zgodnego rozgłaszania niezawodnego (4)

Odebranie pakietu przez monitor Q_i od monitora Q_j powoduje sprawdzenie, czy wiadomość aplikacyjna zawarta w pakiecie nie była już wcześniej dostarczona do P_i. Jeżeli nie, to wiadomość ta zostaje dostarczona.

Następnie wiadomość zostaje dodana do zbioru będącego *j*-tym elementem tablicy $from_i$. Jeżeli proces P_j jest niepoprawny, to monitor Q_i rozsyła ponownie otrzymany pakiet za pomocą podstawowego rozgłaszania niezawodnego.

Pasywny algorytm zgodnego rozgłaszania niezawodnego (5)

W przypadku wykrycia awarii procesu P_j przez doskonały detektor awarii P, proces P_j usuwany jest ze zbioru $correct_i$ oraz ponownie są rozsyłane wszelkie pakiety kiedykolwiek otrzymane od tego procesu.

Pasywny algorytm zgodnego rozgłaszania niezawodnego: złożoność

Załóżmy, że topologią przetwarzania rozproszonego jest graf w pełni połączony.

Jeżeli początkowy nadawca wiadomości nie ulega awarii (przypadek optymistyczny), pasywny algorytm rozgłaszania niezawodnego kończy się w jednym kroku po wysłaniu *n* komunikatów. Stąd złożoność czasowa wynosi 1, a komunikacyjna *n*. W przypadku, w którym kolejno ulegają awarii wszystkie procesy odbierające tę wiadomość, i wiadomość jest ostatecznie odebrana tylko przez jedyny poprawny proces, złożoność czasowa wynosi *n*. Ponieważ w każdym kroku komunikaty wysyłane są do wszystkich procesów, a każdy z nich retransmituje każdą wiadomość co najwyżej raz, złożoność komunikacyjna wynosi n^2.

Aktywny algorytm zgodnego rozgłaszania niezawodnego: Założenia

Zgodne rozgłaszanie niezawodne można także zrealizować za pomocą algorytmu aktywnego (ang. *eager reliable broadcast*), który nie wymaga użycia doskonałego detektora awarii. Algorytm został przeznaczony zatem dla modelu ukrytych awarii (ang. *fail-silent*). Podstawowa różnica w stosunku do poprzedniego algorytmu polega na retransmisji wiadomości przez każdy proces natychmiast po jej dostarczeniu.

Aktywny algorytm zgodnego rozgłaszania niezawodnego (1)

Algorytm używa tylko jednego typu wiadomości PACKET o strukturze identycznej jak w poprzedniej realizacji rozgłaszania niezawodnego. Również znaczenie stosowanych zmiennych nie różni się od analogicznych zmiennych w algorytmie pasywnego rozgłaszania niezawodnego.

Aktywny algorytm zgodnego rozgłaszania niezawodnego (2)

W porównaniu z algorytmem poprzednim, rozesłanie wiadomości aplikacyjnej różni się tylko natychmiastowym dostarczeniem jej do lokalnego procesu. Rozgłaszanie rozsyła wiadomości do wszystkich procesów *łącznie* z nadawcą.

Aktywny algorytm zgodnego rozgłaszania niezawodnego (3)

Podczas odbioru pakietu monitor Q_i sprawdza, czy zawarta w pakiecie wiadomość nie została już wcześniej dostarczona. Jeżeli nie, to dostarcza tę wiadomość do procesu aplikacyjnego i zapamiętuje ten fakt dodając wiadomość do zbioru $delivered_i$. Następnie monitor rozsyła otrzymany pakiet za pomocą podstawowego rozgłaszania niezawodnego (ang. *best-effort broadcast*) do wszystkich monitorów z zbioru $\mathcal{Q}$.

Aktywny algorytm zgodnego rozgłaszania niezawodnego: Przykład (1)

W pokazanym przykładzie widać zasadniczą różnicę w stosunku do wcześniej pokazywanego podstawowego rozgłaszania niezawodnego. Jeżeli P_1 rozgłosił wiadomość M, to każdy otrzymujący ją proces, zarówno P_2 jak i P_3, natychmiast rozsyłają ją do wszystkich innych procesów. Widać, że nawet gdyby proces P_1 uległ awarii przed wysłaniem wiadomości do innych procesów niż P_2, to poprawność procesu P_2 zapewniłaby ostateczne dostarczenie tej wiadomości do wszystkich innych poprawnych procesów.

Aktywny algorytm zgodnego rozgłaszania niezawodnego: Przykład (2)

W tym przykładzie, procesy P_1 oraz P_2 ulegają awarii zanim zdążą rozesłać wiadomość do wszystkich procesów. Nie narusza to poprawności algorytmu, gdyż wiadomość nie została odebrana przez żaden poprawny proces, a więc spełniony jest warunek zgodności.

Aktywny algorytm zgodnego rozgłaszania niezawodnego: Złożoność

Przyjmujemy, że topologia przetwarzania rozproszonego jest reprezentowana przez graf w pełni połączony.

Jeżeli początkowy nadawca wiadomości nie ulega awarii (przypadek optymistyczny) aktywny algorytm zgodnego rozgłaszania niezawodnego kończy się w 1 kroku. Stąd jego złożoność czasowa wynosi 1, a złożoność komunikacyjna wynosi w tym przypadku n^2.

Jeżeli wszystkie procesy kolejno ulegają awarii, złożoność czasowa wynosi n. Każdy z procesów ulegających awarii może wysłać n wiadomości, ale być może się tylko jedna z nich dotrze do adresata.

Jednolite rozgłaszanie niezawodne: Specyfikacja

Kolejną z omawianych abstrakcji komunikacyjnych rozgłaszania niezawodnego jest jednolite rozgłaszanie niezawodne (ang. *uniform reliable broadcast*). Główną różnicą w stosunku do mechanizmów omawianych wcześniej jest wymóg, by zbiór wiadomości dostarczonych przez procesy niepoprawne zawsze był podzbiorem zbioru wiadomości odebranych przez procesy poprawne.

Mechanizm ten ma z definicji własności ważności, braku powielania, braku samogeneracji i jednolitej zgodności.

Własność *ważności* (ang. *validity*) oznacza, podobnie jak poprzednio, że jeżeli proces P_i jest poprawny, to każda wiadomość rozgłaszana przez ten proces jest ostatecznie dostarczona do P_i.

Własność *braku powielania* (ang. *no duplication*) oznacza, że jeżeli wiadomość jest dostarczona to jest dostarczona co najwyżej raz.

Własność *braku samogeneracji* (ang. *no creation*) oznacza, że jeżeli jakaś wiadomość jest dostarczona do procesu P_j, to została wcześniej rozgłoszona przez jakiś proces P_i.

Wreszcie, własność *jednolitej zgodności* (ang. *uniform agreement*) oznacza, że jeżeli jakaś wiadomość została odebrana przez pewien proces P_i (poprawny bądź niepoprawny) to ostatecznie wszystkie poprawne procesy odbiorą tę wiadomość. Własność ta jest szczególnie ważna, jeżeli proces dokonuje interakcji ze światem zewnętrznym, zwłaszcza, jeżeli efekty tej interakcji są nieodwracalne (np. wydruk dokumentu, przelew bankowy i tak dalej).

Jednolite rozgłaszanie niezawodne: Operacje komunikacyjne

Operację jednolitego rozgłaszania niezawodnego wiadomości M przez P_i do zbioru procesów $\mathcal{P}$ oznaczać będziemy przez $\mathbf{send}^{\mathbf{URB}}(P_i, \mathcal{P}, M)$. Odpowiadające tej operacji zdarzenie zapisywać będziemy jako $e\_send^{URB}(P_i, \mathcal{P}, M)$. Analogicznie, przez $\mathbf{deliver}^{\mathbf{URB}}(P_j, P_i, M)$ oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{URB}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M wysłaną przez proces P_j do procesu aplikacyjnego P_i.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich: Założenia

Algorytm jednolitego rozgłaszanie z potwierdzeniami od wszystkich (ang. *all-ack uniform reliable broadcast*) realizuje usługę jednolitego rozgłaszania w modelu jawnych awarii (ang. *fail-stop*). Używa on podstawowego rozgłaszania niezawodnego, kanałów niezawodnych oraz doskonałego detektora awarii. W algorytmie tym wiadomość jest odbierana przez proces P_i tylko wtedy, gdy proces ten zaobserwował retransmisję tej wiadomości przez wszystkie poprawne procesy.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (1)

Algorytm używa typu PACKET o zwykłej strukturze.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (2)

Większość zmiennych używanych przez prezentowany teraz algorytm ma znaczenie identyczne jak w przypadku zgodnego rozgłaszania niezawodnego. Dwie nowe zmienne to zbiór $pending_i$ zawierający pakiety z wiadomościami oczekujące na dostarczenie oraz zbiór ack_i zawierający pary: pakiet, zbiór identyfikatorów procesów. Dalej omawiana funkcja SELECT używa lokalnych zmiennej *result*, będącej zbiorem identyfikatorów procesów, zmiennej *pcktAck* typu PACKET oraz zmiennej *pId* typu PROCESS_ID. Zmienna *pckt* typu PACKET używana jest lokalnie podczas próby odnalezienia pakietów, które mogą zostać dostarczone do procesu aplikacyjnego.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (3)

Zauważmy, że funkcja SELECT zwraca identyfikatory wszystkich procesów, które dokonały już retransmisji pakietu *pcktIn*.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (4)

Wysyłając wiadomość *msgOut* monitor umieszcza ją w pakiecie *pcktOut*, który jest dodany do zbioru $pending_i$, po czym pakiet zostaje rozgłoszony za pomocą podstawowego rozgłaszania niezawodnego.

Obsługa awarii procesu odbywa się w zwykły sposób, jako efekt działania detektora awarii.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich (5)

Odbierane pakiety monitor Q_i dodaje ją do zbioru ack_i razem z identyfikatorem nadawcy. Nadawca pakietu Q_j może się różnić od pierwotnego nadawcy, będąc jedynie pośrednikiem w przekazaniu wiadomości. Jeżeli dostarczony pakiet nie zawiera się jeszcze w zbiorze $pending_i$ to jest do niego dodawany a następnie rozsyłany za pomocą podstawowego rozgłaszania niezawodnego.

Wiadomość jest dostarczana do procesu aplikacyjnego dopiero wtedy, kiedy monitor Q_i otrzyma tę wiadomość w wyniku retransmisji od wszystkich poprawnych procesów.

Ilustracja algorytmu jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich

W podanym przykładzie wiadomość M jest rozgłaszana przez proces P_1 (Q_1), który następnie ulega awarii. Dla zwiększenia czytelności, pominięto strzałki oznaczające fakt wysłania przez proces wiadomości do samego siebie oraz od procesów P_3 i P_4. W nawiasach klamrowych podano zawartość zbioru ack_i procesów. Zauważmy, że wiadomość M ta dotarła jedynie do monitora Q_2, który przesłał ją dalej do pozostałych procesów, ale nie dostarczył do P_2. Po otrzymaniu wiadomości M od Q_2, monitor Q_3 rozsyła ją ponownie. Wiadomość ta dociera do Q_2 w wyniku czego do zbioru ack_i dodawany jest proces P_3. Monitor Q_4 postępuje identycznie: odbierając wiadomość M od Q_2, rozsyła ją, wstrzymując jednak dostarczenie do procesu aplikacyjnego P_4. Gdy wiadomość dotarła do monitora Q_2 od Q_3 i Q_4, zbiór ack_i został powiększony o identyfikatory procesów P_3 i P_4. W tym momencie monitor Q_2 może już dostarczyć wiadomość M do procesu P_2, gdyż wie, że dotarła ona do monitorów wszystkich poprawnych procesów. Analogicznie postępują monitory Q_3 oraz Q_4.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od wszystkich: Złożoność

Jak zwykle rozważając złożoność czasową i komunikacyjną przyjmiemy, że topologia przetwarzania rozproszonego ma postać grafu pełnego.

Jeżeli początkowy nadawca wiadomości nie ulega awarii (przypadek optymistyczny) algorytm *all-ack* jednolitego niezawodnego rozgłaszania kończy się w ciągu 2 kroków, po wysłaniu n komunikatów przez oryginalnego nadawcę i ich retransmisji przez wszystkie n-1 pozostałych procesów, z których każdy wysyła n komunikatów. Złożoność czasowa wynosi więc 2, a złożoność komunikacyjna n^2. W przypadku pesymistycznym, w którym kolejno ulegają awarii wszystkie procesy, złożoność czasowa wynosi $n + 1$, a komunikacyjna n^2.

Algorytm jednolitego rozgłaszania niezawodnego z potwierdzeniami od większości: Założenia

Rozważany algorytm zakłada dostępność mechanizm podstawowego rozgłaszania niezawodnego. Nie jest wymagany detektor awarii, a więc przyjmuje się model przetwarzania z ukrytymi awariami. Wymagana jest jednak, by większość procesów nie ulegała awarii. Przy tych założeniach jednolite rozgłaszanie można zaimplementować za pomocą algorytmu z potwierdzeniami od większości. Różni się on od poprzednio omawianego tylko tym, że nie jest używany doskonały detektor awarii, a także warunkiem dostarczenia wiadomości sformułowanym w wierszu 15. Pozostałe kroki algorytmu są identyczne.

Jednolite rozgłaszanie niezawodne z potwierdzeniami od większości: Algorytm

Warunek w wierszu 15, odróżniający algorytm od poprzednio przedstawionego, oznacza tutaj oczekiwanie na nadejście wiadomości od większości procesów, w tym od co najmniej jednego poprawnego.

Problem implozji potwierdzeń

Jak widać z wcześniej przedstawionych algorytmów, implementacja niezawodnego rozgłaszania w zawodnym środowisku, w którym zarówno łącza jak i procesy mogą ulegać awariom, może wymagać zbierania potwierdzeń od wielu procesów. Staje się to problematyczne w przypadku, gdy liczba procesów biorących udział w przetwarzaniu rozproszonym rośnie. Zadanie zebrania potwierdzeń od dużej grupy procesów może w efekcie zajmować poważną część zasobów (pamięci, czasu procesora) węzłów systemu rozproszonego.

Rozwiązanie problemu implozji potwierdzeń

Problem ten można rozwiązać za pomocą grupowania procesów w hierarchiczne struktury, na przykład drzewa binarne. Dzięki temu obciążenie każdego procesu ulega zmniejszeniu, kosztem jednakże zwiększenia czasu potrzebnego na zebranie wszystkich potwierdzeń. Niestety, w przypadku awarii któregoś z procesu, pojawia się potrzeba często czasochłonnej rekonfiguracji używanych struktur.

Probabilistyczne rozgłaszanie niezawodne: Specyfikacja

Problem implozji potwierdzeń wymusił poszukiwanie rozwiązań lepiej skalowalnych niż rozważane poprzednio. Interesującą propozycją jest podejście probabilistyczne, oparte na przykładach zaczerpniętych z rzeczywistości – takich jak szerzenie się chorób zakaźnych albo rozpowszechnianie plotek. Stąd tego typu mechanizmy (algorytmy) nazwano algorytmami *epidemicznymi* (ang. *epidemic*) czy też *plotkarskimi* (ang. *gossiping*, *rumor mongering*). Korzystają one z abstrakcji komunikacyjnej (mechanizmu) określanej jako probabilistyczne rozgłaszanie niezawodne (ang. *probabilistic reliable broadcast*).

Mechanizmy probabilistycznego rozgłaszania niezawodnego charakteryzują się, podobnie jak poprzednie mechanizmy rozgłaszania, własnościami ważności, braku powielania i braku samogeneracji. Zupełnie inaczej jest tu zdefiniowana własność ważności (ang. *validity*) oznaczająca, że dane jest prawdopodobieństwo, takie że jeżeli procesy P_i oraz P_j są poprawne, to każda wiadomość rozgłaszana przez proces P_i jest ostatecznie dostarczona do P_j z tym prawdopodobieństwem. Dopuszczalne (choć oczywiście niepożądane) są zatem sytuacje, w których wiadomość rozgłaszana nie jest dostarczona do wszystkich poprawnych procesów, nawet jeżeli nadawca wiadomości był poprawny. Widać tutaj podobieństwo do podstawowego rozgłaszania niezawodnego. W pewnym sensie podstawowe rozgłaszanie niezawodne również można by określić mianem *probabilistycznego*. W przypadku probabilistycznego rozgłaszania niezawodnego prawdopodobieństwo nie jest jednak bezpośrednio powiązane z możliwością awarii procesu.

Pozostałe dwie własności nie różnią się od odpowiednich własności uprzednio przedstawionych mechanizmów rozgłaszania:

Własność *braku powielania* (ang. *no duplication*) oznacza, że jeżeli wiadomość jest dostarczona, to jest dostarczona co najwyżej raz.

Własność *braku samogeneracji* (ang. *no creation*) oznacza, że jeżeli jakaś wiadomość jest dostarczona do procesu P_j, to została wcześniej rozgłoszona przez jakiś proces P_i.

Probabilistyczne rozgłaszanie niezawodne:Operacje komunikacyjne

Operację probabilistycznego rozgłaszania niezawodnego wiadomości M przez P_i do zbioru procesów $\mathcal{P}$ oznaczać będziemy przez $\mathbf{send}^{\mathbf{PRB}}(P_i, \mathcal{P}, M)$. Odpowiadające tej operacji zdarzenie zapisywać będziemy jako $e\_send^{PRB}(P_i, P_j, M)$. Analogicznie, przez $\mathbf{deliver}^{\mathbf{PRB}}(P_j, P_i, M)$ oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{PRB}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M wysłaną przez proces P_j do procesu aplikacyjnego P_i.

Idea probabilistycznego rozgłaszania niezawodnego

Ogólna idea algorytmu wykorzystującego mechanizm plotkowania przedstawiona jest na rysunku. Każdy proces wybiera losowo *k* procesów, do których następnie wysyła wiadomość. Identycznie postępuje każdy proces po odebraniu wiadomości. Im większe *k*, tym więcej wymienianych wiadomości, ale zarazem mniejsza liczba rund potrzebna by wiadomość dotarła do wszystkich poprawnych procesów z danym prawdopodobieństwem.

Aktywny algorytm probabilistycznego rozgłaszania niezawodnego: Założenia

W przeciwieństwie do dotychczas przedstawionych algorytmów, aktywny algorytm niezawodnego rozgłaszania probabilistycznego (ang. *eager probabilistic reliable broadcast*) nie posiada żadnych szczególnych wymagań. W istocie, każde rzeczywiste użyteczne łącze komunikacyjne powinno spełniać warunki pozwalające uznać je za kanał rzetelny. Algorytm składa się z co najwyżej *r* rund, przy czym w każdej rundzie rozsyłane jest co najwyżej *k* wiadomości, gdzie *r* oraz *k* są liczbami dobranymi przez użytkownika. W zapisie algorytmu będą one reprezentowane odpowiednio przez zmienne *maxRoundNo* oraz *fanout*.

Aktywny algorytm probabilistycznego rozgłaszania niezawodnego (1)

Struktura wiadomości PACKET używana przez aktywne probabilistyczne rozgłaszanie niezawodne (ang. *eager probabilistic broadcast*) różni się od dotychczasowych dodatkowym polem *roundNo*, zawierającym licznik pozostałego czasu życia pakietu.

Aktywny algorytm probabilistycznego rozgłaszania niezawodnego (2)

Zmienna $fanout_i$ oznacza, ilu adresatów ma zostać wylosowanych przy rozsyłaniu wiadomości. Im większa jej wartość, tym większe obciążenie procesów i większa liczba nadmiarowo przesyłanych wiadomości – ale zarazem tym większe prawdopodobieństwo dostarczenia wiadomości do wszystkich procesów w określonej maksymalnej liczbie rund (kroków). Zmienna $maxRoundNo_i$ oznacza czas życia wiadomości wyróżniony maksymalną liczbą kolejnych jej przesłań. Zbiór *targets*, używany lokalnie w procedurze `GOSSIP`, oznaczać będzie zbiór procesów, które zostały wybrane jako adresaci rozsyłanych wiadomości. Zbiór *candidate* jest używany lokalnie w przedstawionej dalej funkcji `GOSSIP`. Pozostałe zmienne posiadają znaczenie jak poprzednio.

Aktywny algorytm probabilistycznego rozgłaszania niezawodnego (3)

W procedurze `GOSSIP` najpierw dokonywany jest wybór adresatów rozsyłanych wiadomości. Są oni wybierani losowo ze zbioru wszystkich procesów. Używana jest tu funkcja `RANDOM`, która dokonuje losowego wyboru jednego elementu z podanego zbioru, w tym wypadku ze zbioru procesów $\mathcal{P}$. Następnie pakiet *pcktOut* jest wysyłany do wszystkich wybranych procesów z użyciem mechanizmu kanałów rzetelnych.

Aktywny algorytm probabilistycznego rozgłaszania niezawodnego (4)

Kiedy proces aplikacyjny P_i chce rozgłosić wiadomość *msgOut*, monitor Q_i wypełnia pole *origin* wysyłanego pakietu wstawiając tam identyfikator procesu P_i, ustawia wartość pola *roundNo* na maksymalny czas życia i wysyła pakiet za pomocą procedury `GOSSIP`.

Aktywny algorytm probabilistycznego rozgłaszania niezawodnego (5)

Każdy monitor odbierając wiadomość sprawdza, czy nie została ona przedtem dostarczona. Jeżeli nie, wiadomość jest dostarczana teraz. Równocześnie, jeżeli pole *roundNo* nadesłanego pakietu jest większe od zera, to monitor dekrementuje to pole i rozsyła ten pakiet dalej za pomocą procedury `GOSSIP`.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego: Koncepcja

Wadą rozgłaszania opartego o mechanizm plotkowania jest duża liczba potencjalnie nadmiarowych wiadomości. Sposobem pozwalającym na ominięcie tego problemu i zmniejszenie zużycia zasobów systemu jest wykorzystanie pewnego istniejącego, zawodnego mechanizmu rozgłaszania. Na początku wiadomość jest rozsyłana za pomocą dowolnej (być może zawodnej) operacji rozgłaszania, a następnie dodatkowo używany jest mechanizm plotkowania z użyciem zawodnych operacji komunikacyjnych łączy rzetelnych, w celu retransmisji zagubionych wiadomości. Retransmisje takie mogłyby być oczywiście wykonywane jedynie przez oryginalnego nadawcę. Aby jednak zwiększyć prawdopodobieństwo dostarczenia zagubionych wiadomości, przyjmuje się, że retransmisje mogą być wykonywane także przez inne procesy. Rezultatem tego jest wymóg czasowego przechowywania przez każdy proces pewnej liczby odebranych wiadomości.

Fakt zagubienia wiadomości (dopuszczalny z powodu przyjętej zawodności mechanizmu rozgłaszania) można wykryć za pomocą utrzymywanych przez każdy proces liczników wysyłanych wiadomości. Liczniki te są dołączane do każdej wiadomości jako jej numer sekwencyjny. Proces podejrzewa zagubienie wiadomości, jeżeli numer sekwencyjny aktualnie dostarczonej wiadomości od procesu P_j różni się o więcej niż jeden od numeru sekwencyjnego wiadomości uprzednio otrzymanej od P_j. W takim wypadku proces może rozgłosić prośbę o nadesłanie brakujących wiadomości za pomocą mechanizmu plotkowania.

Proces odbierający prośbę o retransmisję wiadomości spełnia ją, jeżeli wiadomość ta znajduje się w zbiorze wiadomości przechowywanych przez ten proces. W przeciwnym wypadku używa mechanizmu plotkowania, by przesłać tę prośbę do innych procesów.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (ang. *lazy probabilistic broadcast*) realizuje przedstawioną ideę.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego: Założenia

Wymagania pasywnego algorytmu probabilistycznego rozgłaszania niezawodnego (ang. *lazy probabilistic broadcast*) wynikają z podziału rozgłaszania na dwie fazy. Na początku wiadomość jest rozsyłana za pomocą dowolnej (być może zawodnej) operacji rozgłaszania, a następnie dodatkowo używany jest mechanizm plotkowania za pomocą zawodnych operacji komunikacyjnych kanały rzetelnych w celu retransmisji zagubionych wiadomości.

Ponieważ na operacje rozgłaszania nie są narzucone żadne wymagania, będziemy je oznaczać **send***(P_i, $\mathcal{P}$, M) a odpowiadające jej zdarzenie wysłania *e_send**(P_i, $\mathcal{P}$, M). Analogicznie, zdarzenie odebrania wiadomości oznaczać będziemy przez *e_receive**(P_j, P_i, M). Mechanizmowi temu może odpowiadać pewien dowolny mechanizm rozgłaszania oparty o rzetelne kanały komunikacyjne.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (1)

Algorytm używa trzech typów wiadomości. Wiadomości aplikacyjne są przesyłane za pomocą wiadomości typu PACKET, które poza polem *origin* określającym identyfikator pierwotnego (oryginalnego) nadawcy komunikatów, zawierają też numer sekwencyjny wiadomości w polu *seqNo*. Typ REQEST zawiera wszystkie pola takie same jak PACKET (dziedziczy po strukturze PACKET) oraz dodatkowo posiada pole *roundNo* określające maksymalny czas życia wiadomości. Wreszcie typ ANSWER służy do dostarczania brakujących wiadomości.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (2)

Zmienne $fanout_i$, $maxRoundNo_i$, $targets_i$ oraz *pcktOut* posiadają znaczenie jak poprzednio. Wiadomość *ansOut* jest typu ANSWER, a wiadomość *reqOut* - typu REQUEST. Zmienna $seqNo_i$ (ang. *sequence number*) przechowuje ostatni numer sekwencyjny wysyłanej wiadomości. Zmienna $storeThr_i$ (ang.

store threshold) jest parametrem określającym prawdopodobieństwo zapamiętania otrzymanej wiadomości (w celu jej ewentualnej retransmisji). Zbiór $stored_i$ zawiera pakiety przechowywane w celu ewentualnej retransmisji. Pakiety te powinny być po pewnym czasie usunięte z tego zbioru, co zostało pominięte dla prostoty prezentacji. Zbiór $pending_i$ zawiera pakiety otrzymane, których wiadomości aplikacyjne jeszcze nie zostały dostarczone do procesów aplikacyjnych. Tablica $vSeqNo_i$, zawiera numery sekwencyjne ostatnio dostarczonych wiadomości. *K*-ty element tej tablicy określa numer sekwencyjny ostatnio dostarczonej wiadomości od P_k. Zmienna *pckt*, typu PACKET jest używana lokalnie w funkcji DELIVERPENDING. Przypomnijmy, że zmienna $targets_i$, używana lokalnie w procedurze GOSSIP oznacza zbiór procesów, które zostały wybrane jako adresaci rozsyłanych wiadomości. Zbiór *candidate* jest używany lokalnie w przedstawionej dalej funkcji GOSSIP.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (3)

Procedura GOSSIP jest identyczna jak w poprzednim algorytmie. Jej efektem jest rozesłanie pakietu *pcktOut* do zbioru procesów, których liczba określona jest przez zmienną $fanout_i$.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (4)

Procedura DELIVERPENDING przegląda zbiór $pending_i$ sprawdzając, czy może dostarczyć do P_i kolejną wiadomość wysłaną pierwotnie przez proces P_k. Warunkiem dostarczenia jest, by pakiet ze zbioru $pending_i$ pochodzący od Q_k posiadał numer sekwencyjny o jeden większy od numeru sekwencyjnego ostatnio dostarczonej wiadomości od P_k, Jeżeli dostarczenie jest możliwe, to odpowiednio uaktualniona jest tablica $vSeqNo_i[k]$ oraz zmienna $pending_i$. Warto zaznaczyć, że dostarczenie jednej wiadomości może pozwolić do dostarczenie kolejnej.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (5)

Kiedy proces aplikacyjny P_i chce rozesłać wiadomość *msgOut*, monitor Q_i zwiększa licznik wysłanych wiadomości $seqNo_i$ i rozsyła następnie pakiet za pomocą zawodnej operacji rozgłaszania.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (6)

Monitor Q_i odbierając pakiet, po pierwsze z pewnym prawdopodobieństwem zapamiętuje go w zmiennej $stored_i$, umożliwiając tym samym późniejszą jego retransmisję. Następnie monitor sprawdza, czy numer sekwencyjny pakietu jest równy numerowi następnego oczekiwanego pakietu od Q_j. Jeżeli tak, dostarcza wiadomość do procesu aplikacyjnego P_i. W przeciwnym wypadku wstawia pakiet do zbioru $pending_i$ i używając procedury GOSSIP wysyła prośbę do wszystkich procesów o przysłanie mu brakujących pakietów. Istnieje pewne prawdopodobieństwo, że żaden z procesów, do których dotrą prośby, nie posiada tego pakietu, gdyż pakiet został zagubiony w czasie rozgłaszania i nie dotarł do żadnego monitora, lub prośby o retransmisję nie dotarły do właściwych monitorów. W takim wypadku, po jakimś czasie monitor Q_i zwiększa numer sekwencyjny następnej oczekiwanej wiadomości od P_j, akceptując utratę jednej wiadomości (dla uproszczenia prezentacji, ta funkcjonalność została tu pominięta).

Niezdefiniowana funkcja RANDOM* użyta w zapisie algorytmu oznacza funkcję zwracającą losową liczbę z określonego przedziału liczb naturalnych.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (7)

Monitor Q_i otrzymując prośbę o retransmisję pakietu, spełnia ją, jeżeli posiada poszukiwany pakiet od określonego nadawcy i o wymaganym numerze sekwencyjnym.

W przeciwnym wypadku, jeżeli pole *round* otrzymanej prośby jest większe od zera, monitor Q_i rozsyła tą prośbę dalej za pomocą procedury GOSSIP.

Dla uproszczenia zapisu, zastosowano tu funkcję INDEXOF(*pId*), która zwraca indeks procesu o identyfikatorze *pId*.

Pasywny algorytm probabilistycznego rozgłaszania niezawodnego (8)

Po otrzymaniu odpowiedzi na wysłaną prośbę monitor sprawdza, czy może dostarczyć odebraną wiadomość, a jeśli tak, czy może dostarczyć któreś z wiadomości umieszczonych w zbiorze $pending_i$. W przeciwnym wypadku dołącza pakiet do zbioru $pending_i$.

Zgodne rozgłaszanie niezawodne z przyczynowym uporządkowaniem wiadomości: Specyfikacja

Mówimy, że wiadomość M_1 poprzedza przyczynowo M_2, jeżeli zdarzenie wysłania M_2 zależy przyczynowo od zdarzenia wysłania M_1. Innymi słowy, oznacza to, że zachodzi któryś z poniższych przypadków:

1) Obie wiadomości zostały rozgłoszone przez ten sam proces i wiadomość M_1 została rozgłoszona przed M_2,

2) Wiadomość M_1 została odebrana przez pewien proces P_i, a M_2 została rozgłoszona przez P_i po odebraniu M_1,

3) Istnieje wiadomość M' taka, że dla M_1 i M', oraz M' i M_2 zachodzi 1) bądź 2)

Relacja przyczynowego porządku między wiadomościami jest zachowana przez kolejny omawiany w ramach wykładu mechanizm niezawodnego rozgłaszania, mianowicie zgodnego rozgłaszania niezawodnego z przyczynowym uporządkowaniem wiadomości (ang. *Causally order reliable broadcast*). Mechanizm ten posiada wszystkie własności zgodnego rozgłaszania niezawodnego i dodatkowo własność *przyczynowego uporządkowania* (ang. *causal delivery*) wiadomości.

Własność przyczynowego uporządkowania oznacza, że przed dostarczeniem dowolnej wiadomości M, proces musi dostarczyć wszystkie wiadomości od których M przyczynowo zależy.

Zgodne rozgłaszanie niezawodne z przyczynowym uporządkowaniem wiadomości: Operacje komunikacyjne

Operację zgodnego rozgłaszania niezawodnego M przez P_i do zbioru procesów $\mathcal{P}$, z zachowaniem przyczynowego uporządkowania wiadomości oznaczać będziemy przez $\mathbf{send}^{\mathbf{RRB\_CO}}(P_i, \mathcal{P}, M)$. Odpowiadające tej operacji zdarzenie zapisywać będziemy jako $e\_send^{RRB\_CO}(P_i, \mathcal{P}, M)$. Analogicznie, przez $\mathbf{deliver}^{\mathbf{RRB\_CO}}(P_j, P_i, M)$ oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{RRB\_CO}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M wysłaną przez proces P_j do procesu aplikacyjnego P_i.

Algorytm zgodnego rozgłaszania niezawodnego z przyczynowym uporządkowaniem wiadomości: Założenia

Zostanie obecnie przedstawiony algorytm realizujący mechanizm zgodnego rozgłaszania niezawodnego zachowującego przyczynowe uporządkowanie wiadomości. Algorytm ten zakłada dostępność mechanizmu (abstrakcji) zgodnego rozgłaszania niezawodnego, lecz nie wymaga detektora awarii. Algorytm odpowiedni jest zatem dla modelu przetwarzania z ukrytymi awariami.

Algorytm zgodnego rozgłaszania niezawodnego z przyczynowym uporządkowaniem wiadomości (1)

Wiadomości aplikacyjne przesyłane są w komunikatach typu PACKET. Pakiety te mają dodatkowo pole *past*, zawierające zbiór informacji o wiadomościach poprzedzających przyczynowo wiadomość aplikacyjną przesyłaną w tym pakiecie.

Zbiór $past_i$ dla każdego procesu P_i zawiera zbiór informacji o wiadomościach, które poprzedzają przyczynowo wysyłaną przez proces wiadomość aplikacyjną.

Zbiór $delivered_i$ zawiera wiadomości dostarczone do procesu P_i.

Algorytm zgodnego rozgłaszania niezawodnego z przyczynowym uporządkowaniem wiadomości (2)

Wysyłając pakiet z wiadomością aplikacyjną *msgOut*, monitor Q_i procesu P_i inicjuje odpowiednio pola pakietu, dołączając do niego wiadomości, od których wiadomość *msgOut* jest przyczynowo zależna.

Następnie, wysyłana wiadomość aplikacyjna jest dołączana do zbioru $past_i$ wraz z identyfikatorem nadawcy - P_i. Pakiet jest rozgłaszany z użyciem mechanizmu zgodnego rozgłaszania niezawodnego.

Algorytm zgodnego rozgłaszania niezawodnego z przyczynowym uporządkowaniem wiadomości (3)

Wiadomość aplikacyjna przenoszona przez pakiet jest dostarczana tylko i wyłącznie wtedy, jeżeli zostaną dostarczone wszystkie wiadomości, od których jest ona przyczynowo zależna. W tym celu monitor Q_i przegląda i ewentualnie dostarcza wiadomości zawarte w polu *past* otrzymanego pakietu. Dla uproszczenia zapisu przyjmuje się, że zbiór $pcktIn.past$ jest uporządkowany i wchodzące w jego skład wiadomości są dostarczane w poprawnej kolejności Każda dostarczona wiadomość dołączana jest przy tym do zbioru $past_i$.

Algorytm zgodnego rozgłaszania niezawodnego z przyczynowym uporządkowaniem wiadomości : Złożoność

Jak łatwo zauważyć, złożoność czasowa i komunikacyjna pakietowa przedstawionego algorytmu jest identyczna jak wykorzystywanego mechanizmu zgodnego rozgłaszania niezawodnego.

Poważną wadą algorytmu jest jednak rosnący rozmiar zbioru $past_i$. Zbiór ten rośnie w czasie. Potrzeba dołączania zbioru $past_i$ do wysyłanych wiadomości powoduje wzrost złożoności komunikacyjnej bitowej. W efekcie, algorytm ten nie ma większego znaczenia w rzeczywistych zastosowaniach. W praktyce, zgodne rozgłaszanie niezawodne z przyczynowym uporządkowaniem wiadomości można jednak łatwo uzyskać modyfikując stosownie przedstawione wcześniej mechanizmy komunikacji punkt-punkt, zachowujące uporządkowanie przyczynowe wiadomości.

Zgodne rozgłaszanie niezawodne z globalnym uporządkowaniem wiadomości: Specyfikacja

Mechanizm zgodnego rozgłaszania niezawodnego z globalnym (całkowitym) uporządkowaniem wiadomości (ang. *total order broadcast*), nazywany też czasem rozgłaszaniem atomowym (ang. *atomic broadcast*), oprócz własności zgodnego rozgłaszania niezawodnego ma dodatkowo własność globalnego uporządkowania wiadomości. Ta ostatnia własność oznacza, że wszystkie procesy odbierają wiadomości w takiej samej kolejności.

Zgodne rozgłaszanie niezawodne z globalnym uporządkowaniem wiadomości: Operacje komunikacyjne

Operację zgodnego rozgłaszania niezawodnego wiadomości M przez proces P_i do zbioru procesów $\mathcal{P}$, zachowującą uporządkowanie globalne wiadomości, oznaczać będziemy przez $\mathbf{send}^{\mathbf{RRB\_TO}}(P_i, \mathcal{P}, M)$. Odpowiadające tej operacji zdarzenie zapisywać będziemy jako $e\_send^{RRB\_TO}(P_i, \mathcal{P}, M)$. Analogicznie, przez $\mathbf{deliver}^{\mathbf{RRB\_TO}}(P_j, P_i, M)$ oznaczać będziemy operację uaktywniającą zdarzenie $e\_receive^{RRB\_TO}(P_j, P_i, M)$ i przekazującą w efekcie wiadomość M wysłaną przez proces P_j do procesu aplikacyjnego P_i.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości: Założenia

Zostanie obecnie zademonstrowany algorytm realizujący mechanizm rozgłaszania z globalnym uporządkowaniem wiadomości, poprawny przy następujących założeniach: dostępności niezawodnych kanałów FIFO, oraz poprawności wszystkich procesów.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (1)

Algorytm używa dwóch typów wiadomości. Wiadomości aplikacyjne są przesyłane w pakietach typu PACKET, które zawierają dodatkowo numer sekwencyjny wiadomości w polu $seqNo$ oraz identyfikator pierwotnego nadawcy wiadomości w polu $origin$. Wiadomości typu SNUPDATE (ang. *sequence number update*) służą do aktualizacji numerów sekwencyjnych (znaczników czasowych) i zawierają pole numeru sekwencyjnego $seqNo$.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (2)

Zmienne *pckt* oraz *pcktOut* są używane lokalnie podczas rozstrzygania o globalnej kolejności odebranych wiadomości. Wiadomości *seqNoOut* oraz *updateIn* są typu SNUPDATE. Zbiór $pending_i$ zawiera otrzymane pakiety, w których zawarte są wiadomości aplikacyjne jeszcze nie dostarczone do procesu aplikacyjnego. Tablica $vSeqNo_i$ zawiera numery sekwencyjne pakietów otrzymanych i jest swego rodzaju zegarem wektorowym.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (3)

Wysyłając wiadomość monitor Q_i procesu P_i zwiększa licznik $vSeqNo_i[i]$ a następnie inicjuje odpowiednio pola pakietu. Pakiet jest także dołączony do zbioru pakietów $pending_i$.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (4)

Odbierając wiadomość od monitora Q_j monitor Q_i, ustawia wartość *j*-tego elementu tablicy $vSeqNo_i$ na wartość numeru sekwencyjnego przysłanego pakietu. Następnie dodaje go do zbioru $pending_i$. Jeżeli *i*-ty wpis w tablicy $vSeqNo_i$ jest mniejszy niż numer sekwencyjny odebranego pakietu, monitor zmienia wartość elementu $vSeqNo_i[j]$ na wartość numeru sekwencyjnego z pola *seqNo* odebranego pakietu, i następnie informuje o tym fakcie pozostałe monitory rozgłaszając wiadomość *seqNoOut*.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (5)

Odbierając wiadomość *updateIn* typu SNUPDATE od monitora Q_j, monitor Q_i ustawia wartość *j*-tego elementu tablicy $vSeqNo_i$ na wartość numeru sekwencyjnego przysłanego pakietu.

Wiadomość jest dostarczana, jeżeli posiada najmniejszy licznik (znacznik czasowy) spośród wszystkich otrzymanych wiadomości (w przypadku konfliktów rozstrzyga identyfikator procesu) oraz jeżeli ten licznik jest nie większy od wszystkich elementów tablicy $vSeqNo_i$.

Ilustracja algorytmu zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości

Przeanalizujmy teraz przykład przedstawiony na slajdzie. Początkowo wszystkie pozycje wszystkich tablic *vSeqNo* są wyzerowane. Monitor Q_1 procesu P_1 wysyła pakiet z wiadomością M_1 i $seqNo = 1$. W wyniku tego wartość jego tablicy $vSeqNo_1[1] := 1$, a pakiet z wiadomością M_1 dołączana jest do zbioru $pending_1$. Analogicznie, monitor Q_3 procesu P_3 wysyła pakiet z wiadomością M_3 i $seqNo = 1$,, przypisując $vSeqNo_3[3] := 1$, a pakiet z wiadomością M_3 dołączana jest do zbioru $pending_3$.

Monitor Q_2 po otrzymaniu pakietu z wiadomością M_1 modyfikuje swoją tablicę *vSeqNo*, wykonując przypisanie $vSeqNo_2[1] := 1$. Ponieważ nadesłany pakiet posiada licznik sekwencyjny większy niż licznik lokalny zawarty w $vSeqNo_2[2]$, więc dodatkowo wykonuje operację przypisania $vSeqNo_2[2] := 1$ oraz rozsyła wiadomości SNUPDATE do pozostałych monitorów. Odpowiada to wierszom 12-16 przedstawionego algorytmu. Pozostałe monitory otrzymując tę wiadomość aktualizują odpowiednie pozycje lokalnej tablicy *vSeqNo*.

Załóżmy, że pakiet od monitora Q_3 dociera do Q_1 oraz Q_2. Powoduje to odpowiednią modyfikację tablic *vSeqNo*. W tej chwili dla obu monitorów zachodzi warunek z wiersza 21 algorytmu – numer sekwencyjny pakietu z wiadomością M_1 jest równy wszystkim pozycjom tablicy *vSeqNo*, a identyfikator jej nadawcy jest mniejszy od identyfikatorów nadawców wszystkich pozostałych wiadomości. Po dostarczeniu M_1 i usunięciu odpowiedniego pakietu ze zbioru $pending_i$, oba monitory Q_1 oraz Q_2 mogą następnie dostarczyć wiadomość M_3. Wreszcie, kiedy pakiet z wiadomością M_1 dotrze ostatecznie do Q_3, zostanie ona uznana za możliwą do dostarczenia (znowu zajdzie warunek z wiersza 21 algorytmu), i będzie można następnie dostarczyć również M_2.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości : Złożoność

Rozważając złożoność czasową i komunikacyjną przyjmiemy, że topologia przetwarzania ma postać grafu pełnego.

Jak łatwo zauważyć, w przedstawionym algorytmie rozgłoszenie wiadomości wymaga 2 kroków: w pierwszym nadawca rozgłasza wiadomość, w drugim otrzymuje komunikaty typu SNUPDATE. W pierwszym kroku wysłanych jest *n* komunikatów, zaś w drugim co najwyżej (*n* - 1) × n komunikatów. Stąd, złożoność komunikacyjna wynosi więc $O(n^2)$.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości: awaria procesu

Rozważmy teraz przykład, w którym jeden proces ulegnie awarii.

Załóżmy, że tak jak poprzednio monitor Q_1 procesu P_1 wysyła pakiet z wiadomością M_1. Powoduje to modyfikację elementu $vSeqNo_1[1]$:= 1, oraz dołączenie pakietu z wiadomością M_1 do zbioru $pending_1$. Analogicznie, monitor Q_3 wysyłając pakiet z wiadomością M_3, przypisuje $vSeqNo_3[3]$:= 1, dołączając równocześnie pakiet z wiadomością M_3 do zbioru $pending_3$.

Załóżmy tym razem jednak, że proces P_2, i tym samym monitor Q_2 uległ awarii po otrzymaniu pakietu z wiadomością M_1.

Po pewnym czasie pakiet z wiadomością od procesu P_3 dociera do Q_1. Powoduje to oczywiście odpowiednią modyfikację tablic $vSeqNo_1$. Analogicznie, gdy pakiet z wiadomością M_1 dotrze ostatecznie do Q_3, monitor ten zmodyfikuje odpowiednio tablicę $vSeqNo_3$. Żaden proces nie jest w stanie odebrać wiadomości, gdyż oba czekają na zwiększenie drugiej pozycji tablicy $vSeqNo[2]$ (odpowiadającej procesowi P_2). Zwiększenie to może jednak nastąpić tylko w wyniku otrzymania pakietu od Q_2 – co nigdy nie nastąpi z powodu awarii P_2 i Q_2.

Widzimy tutaj, że w przypadku możliwości awarii choć jednego procesu przedstawione rozwiązanie jest niepoprawne.