Problem konsensusu

Plan wykładu

Celem wykładu jest wprowadzenie w tematykę konsensusu i jego zastosowań w systemach rozproszonych. Omówione zostaną problemy konsensusu podstawowego, konsensusu jednolitego, konsensusu probabilistycznego oraz ich przykładowe rozwiązania. Przedstawione też będą związki między problemami konsensusu i innymi istotnymi problemami przetwarzania rozproszonego.

Konsensus: definicja nieformalna

Nieformalnie, *konsensusem* nazywa się problem uzgadniania przez zbiór procesów jednej wartości spośród zbioru wartości zaproponowanych wstępnie przez te procesy.

Algorytm rozwiązywania tego problemu nazywać też będziemy mechanizmem konsensusu. W systemach rozproszonych mechanizm konsensusu jest oczywiście pewnym specyficznym przetwarzaniem rozproszonym.

Problemy uzgadniania

Należy zauważyć, że konsensus jest szczególnym przypadkiem problemów uzgadniania, w których zbiór procesów podejmuje decyzje powiązane pewnymi zależnościami.

W przypadku konsensusu relacja między decyzjami różnych procesów jest bardzo prosta: wymaga się, by wszystkie procesy decydowały się na tę samą wartość. W przypadku problemów elekcji jeden proces musi zdecydować, że został wybrany, a wszystkie pozostałe, że zostały pokonane (nie wybrane). W problemach zatwierdzania (ang. *commit-abort*) procesy decydują, czy należy zatwierdzić, czy też odrzucić transakcję – przy czym transakcja nie może być zatwierdzona przy sprzeciwie choć jednego procesu (jest to szczególny przypadek konsensusu binarnego). W konsensusie aproksymowanym, inaczej problemie przybliżonej zgody (ang. *approximate agreement*), wszystkie procesy podejmują decyzję, których wartości nie mogą się różnić więcej niż o pewną stałą i muszą należeć do zbioru wartości wejściowych.

Konsensus: Motywacja

Problem konsensusu jest trudny do rozwiązania w systemie rozproszonym, jednocześnie jednak jest on problemem bardzo ważnym. Mając do dyspozycji mechanizm konsensusu można znaleźć rozwiązanie wielu innych zagadnień, między innymi takich jak zgodne rozgłaszanie niezawodne z globalnym uporządkowaniem wiadomości, komunikacja zsynchronizowana z obrazami grup, nieblokujące zatwierdzanie atomowe, uzgadnianie składu grupy.

Jak przedstawiono już na poprzednim wykładzie, zgodne rozgłaszanie niezawodne z globalnym uporządkowaniem wiadomości (ang. *total order reliable broadcast*), polega na zapewnieniu w systemie rozproszonym dostarczenia rozgłaszanych wiadomości w takim samym porządku do wszystkich odbiorców, mimo potencjalnych awarii i nieprzewidywalnych opóźnień komunikacyjnych.

Problem składu grupy (ang. *group membership*) polega na uzgodnieniu przez zbiór procesów spójnego obrazu składu grupy zmieniającej się w czasie.

Obraz składu grupy, identyczny dla wszystkich procesów będących jej członkami, nazywa się *widokiem grupy*.

Problem komunikacji zsynchronizowanej z obrazami grup (ang. *view synchronous multicast*) polega na rozgłaszaniu tej samej sekwencji wiadomości do wszystkich procesów należących do określonego obrazu (widoku) grupy.

Problem nieblokującego zatwierdzania atomowego (ang. *non-blocking atomic commitment*) polega na zatwierdzeniu lub odrzuceniu transakcji przez grupę procesów, przy czym transakcja może być zatwierdzona tylko w przypadku, w którym wszystkie procesy głosowały za jej zatwierdzeniem.

Możliwość rozwiązania wymienionych problemów z użyciem mechanizmu konsensusu nie oznacza, że problemy te są łatwiejsze. Konsensus można bowiem rozwiązać również mając na przykład do dyspozycji mechanizm niezawodnego rozgłaszania z globalnym uporządkowaniem wiadomości.

Konsensus a zgodne rozgłaszanie niezawodne z globalnym uporządkowaniem wiadomości

Konsensus można łatwo sprowadzić do problemu zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości. Aby rozwiązać konsensus z wykorzystaniem istniejącej abstrakcji rozgłaszania, wystarczy wykonać następujące kroki:

- Wszystkie procesy rozsyłają proponowaną wartość za pomocą zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości
- Procesy decydują się na pierwszą dostarczoną wartość.

Z właściwości użytego mechanizmu rozgłaszania wynika, że wiadomości są dostarczone do wszystkich procesów w tym samym porządku, a więc ta sama wiadomość zostanie dostarczona jako pierwsza.

Problem zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości można z kolei sprowadzić do konsensusu wymagając, by dostarczenie *k*-tej wiadomości było decydowane w wyniku kolejnego, *k*-tego konsensusu.

W końcowej części wykładu zostanie przedstawiony algorytm implementujący zgodne rozgłaszanie niezawodne z globalnym porządkiem wiadomości za pomocą zgodnego rozgłaszania niezawodnego i wybranego mechanizmu konsensusu.

Konsensus podstawowy: Specyfikacja

Mechanizm konsensusu podstawowego charakteryzują własności ważności, zakończenia, integralności oraz zgodności.

Własność *ważności* (ang. *validity*) oznacza, że jeżeli którykolwiek proces P_i decyduje się na wartość v_i, to wartość ta została wcześniej zaproponowana przez pewien proces. Innymi słowy, procesy nie mogą decydować się na dowolną wartość, lecz muszą zgodzić się na którąś z przedstawionych propozycji.

Własność *zakończenia* (ang. *termination*) oznacza, że wszystkie poprawne procesy ostatecznie decydują się na jakąś wartość. W systemie asynchronicznym, czas podejmowania decyzji jest przy czym skończony, ale nieznany.

Własność *integralności* (ang. *agreement*) oznacza, że każdy proces decyduje się tylko raz. Procesy nie mogą więc zmieniać raz podjętych decyzji.

Własność *zgodności* (ang. *agreement*) oznacza, że nie istnieją dwa poprawne procesy, które zdecydowałyby się na inne wartości. Tym samym, poprawne procesy uzgadniają wspólną wartość.

Dodatkowo wymagane jest by mechanizm konsensusu miał własność *nietrywialności rozwiązania* (ang. *non-triviality*), którą zazwyczaj pomija się definiując konsensus z uwagi na jej oczywistość. Własność ta mówi, że wartość decyzji nie może być ustalona z góry.

Twierdzenie Fisher-Lynch-Paterson

W 1985 zostało udowodnione (Michael Fisher, Nancy Lynch, Michael Paterson) fundamentalne twierdzenie o nierozwiązywalności konsensusu w pełni asynchronicznym środowisku rozproszonym, o ile jest możliwe załamanie (awaria) chociaż jednego procesu. Innymi słowy, jeżeli nie ma ograniczeń na maksymalny czas przesyłania wiadomości i prędkość procesów, to problem konsensusu nie można w ogólności rozwiązać.

Dowód tego twierdzenia opiera się na wykazaniu, że nie jest możliwe odróżnienie procesu niepoprawnego od takiego, który po prostu jest bardzo wolny lub jest połączony bardzo wolnymi łączami.

Konsensus w systemach częściowo synchronicznych

Twierdzenie 13.2

> Nie istnieje algorytm rozwiązujący problem konsensusu w środowisku (modelu) z synchronicznymi procesami i asynchronicznymi kanałami, jeżeli istnieje możliwość załamania (awarii) choćby jednego procesu.

Wynika to oczywiście z braku ograniczeń na czas przesyłania wiadomości, co powoduje, że nie można odróżnić procesu niepoprawnego od takiego, który połączony jest wyjątkowo wolnymi łączami.

Twierdzenie 13.3

> Nie istnieje algorytm rozwiązujący problem konsensusu w modelu z asynchronicznymi procesami i kanałami o znanych maksymalnych opóźnieniach, jeżeli istnieje możliwość choć jednej awarii procesu.

Wynika to oczywiście z braku ograniczeń na różnice prędkości między procesami, co powoduje, że nie można odróżnić procesu niepoprawnego od takiego, który jest bardzo wolny.

Detektory awarii a konsensus

Konsensus posiada rozwiązanie dla systemów, w których dostępne są doskonałe detektory awarii. Wystarczającym warunkiem jest dostępność detektora klasy S, a w niektórych rodzajach systemów rozproszonych nawet detektora ostatecznie słabego (◊W). Należy zwrócić uwagę, że jeżeli można rozwiązać konsensus z użyciem pewnego detektora awarii, to oznacza to, że detektora tego nie da się zaimplementować w pełni asynchronicznym systemie rozproszonym.

W szczególności, można rozwiązać konsensus w systemach asynchronicznych z dostępnym detektorem awarii P, lub z użyciem ◊S pod warunkiem jednak, że większość procesów jest poprawna, tzn. dla $n > 2f$, gdzie n oznacza liczbę wszystkich procesów a f jest maksymalną liczbą procesów niepoprawnych.

Konsensus podstawowy – operacja propose

Proponowana do uzgodnienia przez proces P_i wartość v_i określana jest w wyniku wykonania operacji $\mathbf{propose}^{\mathbf{RC}}(P_i, v_i)$. Operacja ta inicjuje mechanizm konsensusu.

Zdarzenie zachodzące w procesie P_i w wyniku wykonania tej operacji zapisywane będzie $e\_propose^{RC}(P_i, v_i)$.

Konsensus podstawowy – operacja decide

Określenie przez proces P_i (monitor Q_i) decyzji co do ostatecznie uzgodnionej wartości v następuje w wyniku wykonania operacji $\mathbf{decide}^{\mathbf{RC}}(P_i, v)$. Wykonanie tej operacji kończy mechanizm konsensusu. Zdarzenie $e\_decide^{RC}(Pi, v)$ zachodzące w procesie P_i w wyniku wykonania operacji $\mathbf{decide}^{\mathbf{RC}}(P_i, v)$, oznacza osiągnięcie przez proces aplikacyjny P_i konsensusu, a więc uzgodnienia wartości v z wszystkimi procesami.

Mechanizm konsensusu

Schematycznie, miejsce mechanizmu konsensusu ilustruje powyższy slajd. Mechanizm jest inicjowany zajściem zdarzeń $e\_propose^{RC}(P_i, v_i)$ w procesach aplikacyjnych P_i. Jego efektem jest natomiast podjęcie przez każdy monitor decyzji co do ostatecznej wartości v. Podjęcie tej decyzji odpowiada wykonaniu operacji $\mathbf{decide}^{\mathbf{RC}}(P_i, v)$ i zajściu zdarzenia $e\_decide^{RC}(P_i, v)$, oznaczającego osiągnięcie konsensusu przez P_i, a więc uzgodnienia wartości v z wszystkimi procesami.

Rozgłoszeniowy algorytm konsensusu podstawowego: Założenia

Przedstawiony zostanie obecnie algorytm konsensusu podstawowego, stosujący mechanizmy podstawowego rozgłaszania niezawodnego oraz doskonałego detektora awarii. Z założeń tych wynika, że algorytm przeznaczony jest dla modelu jawnych awarii.

Rozgłoszeniowy algorytm konsensusu podstawowego: Koncepcja

Algorytm działa w rundach, i każda wiadomość jest oznaczona numerem rundy. Decyzja podejmowana jest, gdy proces otrzyma wiadomości od wszystkich poprawnych procesów i nie wykrył żadnej nowej awarii - a więc ma pewność, że otrzymał wszystkie propozycje, które będą wzięte pod uwagę przez wszystkie pozostałe procesy. Jeżeli wykryta została awaria, proces nie ma pewności, czy posiadany przez niego zbiór wartości propozycji jest taki sam, jak widziany przez wszystkie pozostałe procesy. Różnica może mieć na przykład miejsce, jeżeli jakiś błędny proces zdążył przesłać wiadomość tylko do części pozostałych procesów. W takim wypadku rozpoczynana jest kolejna runda. Do podjęcia decyzji używana jest dowolna deterministyczna funkcja, znana wszystkim procesom. Na przykład wybierana może być najmniejsza wartość spośród zaproponowanych.

Rozgłoszeniowy algorytm konsensusu podstawowego (1)

W algorytmie przesyłane są wiadomości dwóch typów. Typ MYPROP zawiera pole *proposedValue*, które jest zbiorem wartości zaproponowanych w danej rundzie przez poszczególne procesy, oraz pole *roundNo* określające numer rundy. Typ DECIDED zawiera pole *decidedValue* określające decyzję podjętą przez nadawcę wiadomości, oraz pole *roundNo* określające rundę, w której została ona podjęta.

Wiadomości typu DECIDED wysyłane są w momencie podjęcia decyzji przez proces (monitor).

Rozgłoszeniowy algorytm konsensusu podstawowego (2)

Używane są dalej wiadomości *decOut* typu DECIDED oraz *propOut* typu MYPROP. Zmienna $decided_i$ zawiera wartość, na którą chce się zdecydować proces P_i, początkowo równa pewnej wyróżnionej wartości pustej. Zbiór $proposalSet_i$ zawiera zestaw wszystkich widzianych przez proces P_i (monitor Q_i) propozycji. Numer rundy zapisany jest w zmiennej $roundNo_i$. Zbiór $correct_i$, początkowo obejmujący identyfikatory wszystkich procesów, oznacza procesy, które według wiedzy dostępnej dla P_i są poprawne. Analogicznie, $correctThisRound_i$ oraz $correctLastRound_i$ zawierają identyfikatory procesów poprawnych w obecnej i poprzedniej rundzie. Zbiór procesów poprawnych w poprzedniej rundzie początkowo obejmuje wszystkie procesy, podczas gdy zbiór procesów poprawnych w rundzie obecnej początkowo jest pusty.

Rozgłoszeniowy algorytm konsensusu podstawowego (3)

Zainicjowanie rozgłoszeniowego algorytmu konsensusu podstawowego związane jest z zajściem zdarzenia $e\_propose^{RC}(\ P_i,\ v_i\ )$ w procesie aplikacyjnym P_i. W efekcie tego zdarzenia, monitor Q_i rozpoczyna wymianę wiadomości z innymi monitorami w celu uzgodnienia wspólnej decyzji co do wartości v. Uzgodnienie to prowadzi do zajścia zdarzeń $e\_decide^{RC}(P_i, v)$.

Rozgłoszeniowy algorytm konsensusu podstawowego (4)

W wyniku wykrycia awarii z użyciem detektora awarii P, proces P_j jest usuwany ze zbioru poprawnych procesów $correct_i$ widzianych przez P_i.

W przypadku otrzymania propozycji od monitora Q_j wartość z pola *propValue* otrzymanej wiadomości MYPROP dodawana jest do zbioru propozycji $proposalSet_i$. Sam proces P_j dodawany jest do zbioru $correctThisRound_i$ procesów poprawnych w obecnej rundzie.

Rozgłoszeniowy algorytm konsensusu podstawowego (5)

W przypadku, gdy monitor Q_i otrzyma wiadomości od monitorów wszystkich poprawnych procesów i dotąd nie podjął decyzji, sprawdza, czy w obecnej rundzie zbiór poprawnych procesów nie uległ zmianie, tzn. czy $correctThisRound_i = correctLastRound_i$. Jeżeli ten warunek jest spełniony, to monitor Q_i podejmuje decyzję wykonując operację $\mathbf{decide}^{\mathbf{RC}}(\ P_i,\ decided_i\ )$. Prowadzi to do zajścia zdarzenia

$e\_decide^{RC}(P_i, decided_i)$ w procesie P_i. Po wykonaniu operacji **decide**$^{RC}(P_i, decided_i)$, monitor Q_i rozgłasza swoją decyzję korzystając z mechanizmu podstawowego rozgłaszania niezawodnego.

W przeciwnym razie, tzn. gdy $correctThisRound_i \neq correctLastRound_i$, monitor przechodzi do kolejnej rundy i rozgłasza aktualny zbiór propozycji.

Rozgłoszeniowy algorytm konsensusu podstawowego (6)

W przypadku, gdy monitor Q_i otrzyma powiadomienie o decyzji pewnego monitora, natychmiast podejmuje identyczną decyzję i rozgłasza ten fakt wszystkim pozostałym monitorom.

Działanie rozgłoszeniowego algorytmu konsensusu podstawowego: Przykład 1

Przykład prezentowany na slajdzie przedstawia cztery procesy uzgadniające wspólną decyzję za pomocą rozgłoszeniowego algorytmu konsensusu podstawowego. Wszystkie procesy proponują pewną wartość, a odpowiadające im monitory rozsyłają ją za pomocą podstawowego rozgłaszania niezawodnego. Załóżmy, że proces P_1 ulega jednak awarii w trakcie rozgłaszania, w wyniku czego jego propozycję otrzymuje jedynie monitor Q_2. Ponieważ monitor Q_2 zebrał wiadomości od wszystkich monitorów, więc może podjąć decyzję wybierając minimalną wartość, na przykład v_1, i rozgłosić ją w systemie. Monitory otrzymując wiadomość o decyzji Q_1 podporządkowują się jej, podejmując identyczną decyzję. Należy zauważyć tutaj, że monitory Q_3 i Q_4 nie mogą same podjąć decyzji, gdyż nie otrzymały wiadomości od wszystkich monitorów procesów, które były uznawane za poprawne na początku tej rundy. Co więcej, po wykryciu awarii procesu P_1 monitory mogą rozpocząć nową rundę, rozgłaszając wszystkie znane sobie propozycje.

Fakt wykrycia awarii procesu P_1 nie wpływa na decyzję monitora Q_2, gdyż otrzymał on już wcześniej wiadomość od tego procesu dodając go do zbioru $correctThisRound_2$. Proces P_1 usunięty jest więc wprawdzie z zbioru $correct_2$, ale ponieważ zbiór $correct_2$ zawiera się w zbiorze $correctThisRound_2$, zachodzi warunek z wiersza 14-tego algorytmu prowadzący do podjęcia decyzji.

Działanie rozgłoszeniowego algorytmu konsensusu podstawowego: Przykład 2

W kolejnym przykładzie dochodzi w pierwszej rundzie do dwóch awarii. Proces P_1 ulega awarii zaraz po przesłaniu swojej propozycji v_1 do monitora Q_2, natomiast P_4 ulega awarii po wysłaniu wiadomości do monitorów Q_1 oraz Q_3. Żaden więc z dwóch pozostałych monitorów nie może podjąć decyzji, gdyż żaden z nich nie otrzymał wiadomości od wszystkich monitorów. Następuje wiec kolejna runda, w której P_2 ulega awarii. Tym samym, po wykryciu awarii procesu P_2, monitor Q_3 przechodzi do kolejnej rundy i podejmuje wreszcie decyzję.

Rozgłoszeniowy algorytm konsensusu podstawowego : Złożoność

Załóżmy, że topologię przetwarzania reprezentuje graf w pełni połączony.

Jeżeli żaden proces nie ulega awarii, procesy podejmują decyzję w pierwszej rundzie. Wymaga ona wymiany n^2 wiadomości typu MYPROP przed osiągnięciem decyzji i drugie tyle wiadomości typu DECIDED. Złożoność czasowa wynosi więc 1, a komunikacyjna $2n^2$. W przypadku, w którym kolejno ulegają awarii wszystkie procesy, algorytm wymaga n kroków (rund), a więc złożoność czasowa wynosi n. W każdej dodatkowej rundzie wymienianych jest dodatkowe $O(n^2)$ wiadomości, z czego wynika, że złożoność komunikacyjna w przypadku pesymistycznym wynosi $O(n^3)$.

Hierarchiczny algorytm konsensusu podstawowego: Założenia

Kolejnym rozwiązaniem problemu konsensusu jest algorytm hierarchiczny, w którym kolejno każdy proces (monitor) pełni rolę lidera rundy. Przyjęto przy tym, że hierarchia procesów jest zdefiniowana przez uporządkowanie procesów według indeksów. Lider rundy rozgłasza swoją decyzję, która jest przejmowana przez pozostałe procesy, po czym zaczyna się kolejna runda z nowym liderem. Jeżeli lider ulegnie awarii, jego rolę przejmuje kolejny proces i tak dalej. Algorytm stosuje mechanizm podstawowego rozgłaszania niezawodnego oraz mechanizm doskonałego detektora awarii. Z założeń tych wynika, że algorytm ten jest przeznaczony dla modelu jawnych awarii.

Hierarchiczny algorytm konsensusu podstawowego (1)

Jedynym używanym typem wiadomości jest typ DECIDED, niosący wartość podjętej decyzji w polu *decidedValue* oraz numer rundy w polu *roundNo,* w której tę decyzję podjęto.

Hierarchiczny algorytm konsensusu podstawowego (2)

Wiadomość *decOut* jest wiadomością powiadamiającą inne procesy o podjętej przez lidera decyzji. Zmienna $proposal_i$ zawiera propozycję procesu P_i a zmienna $propRoundNo_i$ (ang. *proposer round number*) zawiera identyfikator procesu, który zaproponował tę wartość. Tablica $delivered_i$ zawiera wartość *True*, jeżeli proces (monitor) otrzymał powiadomienie o decyzji lidera. Tablica $liderSet_i$ zainicjowana jest identyfikatorami procesów, będących kolejnymi kandydatami na lidera. Identyfikator lidera *k*-tej rundy jest zapisany w *k*-tym elemencie tej tablicy. Tablica ta posiada identyczną zawartość dla wszystkich procesów biorących udział w algorytmie. Tablica $broadcast_i$ zawiera wartość *True*, jeżeli monitor podjął już decyzję i rozgłosił ją wśród procesów. Numer kolejnej rundy zawarty jest w zmiennej $roundNo_i$. Wreszcie zmienna $suspected_i$ posiada reprezentuje zbiór procesów podejrzewanych o to, że są niepoprawne. Zmienna *k* jest używana lokalnie w obsłudze zdarzenia odbioru wiadomości DECIDED.

Hierarchiczny algorytm konsensusu podstawowego (3)

Monitor Q_i zapisuje propozycję procesu P_i w zmiennej $proposal_i$. W przypadku wykrycia awarii jakiegoś procesu P_j jest on dołączany do zbioru $suspected_i$. Z własności *silnej dokładności* doskonałego detektora awarii P wynika, że żaden poprawny proces nigdy nie będzie podejrzewany.

Hierarchiczny algorytm konsensusu podstawowego (4)

Jeżeli monitor Q_i stwierdza, że zostaje liderem bieżącej, *k*-tej rundy (co dzieje się wtedy, gdy jego identyfikator jest równy identyfikatorowi zapisanemu w *k*-tym elemencie tablicy $liderSet_i$) oraz jeżeli zmienna $proposal_i$ nie jest pusta, a także Q_i jeszcze nie podejmował decyzji, to Q_i decyduje się na wartość zapisaną w zmiennej $proposal_i$, zapamiętuje ten fakt i powiadamia o tym pozostałe monitory. Z własności *ważności* podstawowego rozgłaszania niezawodnego wynika, że każdy proces (monitor) poprawny otrzyma rozgłoszoną wiadomość, o ile rozgłaszający proces (monitor) również jest poprawny. Jeżeli więc monitor podejmuje decyzję i jest poprawny, to wszystkie inne poprawne monitory otrzymają wiadomość o podjętej przez niego decyzji.

Hierarchiczny algorytm konsensusu podstawowego (5)

Jeżeli monitor Q_i podejrzewa lidera bieżącej rundy lub już otrzymał od niego wiadomość, zwiększa numer rundy. Z własności dokładności doskonałego detektora awarii P wynika, że jest niemożliwe, by poprawny proces był kiedykolwiek podejrzewany. Z własności kompletności detektora awarii wynika natomiast, że każdy niepoprawny proces będzie podejrzewany.

Jeżeli monitor Q_i otrzymuje powiadomienie o decyzji, to ignoruje ją, gdy pochodzi ona od monitora skojarzonego z procesem znajdującym się niżej w hierarchii procesów niż P_i, lub gdy numer rundy nadesłanej wiadomości jest mniejszy niż największy numer rundy, w której monitor Q_i otrzymał decyzję. Może to bowiem oznaczać, że wiadomość ta została wysłana przez niepoprawny monitor tuż przed jego awarią (warunek w wierszu 18). W przeciwnym wypadku monitor Q_i zapisuje otrzymaną propozycję decyzji do zmiennej $proposal_i$ oraz zapamiętuje, że w rundzie, zapisanej w polu *roundNo* odebranego komunikatu, otrzymał decyzję lidera.

Działanie hierarchicznego algorytmu konsensusu podstawowego: Przykład 1

W przedstawionym przykładzie liderem pierwszej rundy jest monitor Q_1. Rozgłasza on swoją decyzję, po czym proces P_1 i tym samym monitor Q_1 ulegają awarii. Liderem nowej rundy zostaje Q_2. Nowa runda zaczyna się w każdym monitorze albo w momencie otrzymania wiadomości od Q_1, albo w momencie wykrycia jego awarii. Monitor Q_2 rozgłasza wartość, na którą chce się zdecydować i podejmuje decyzję. Wartością tą jest albo v_1, jeżeli Q_2 otrzymał wiadomość od Q_1 przed wykryciem awarii (tak jak w przykładzie), albo v_2. Monitory Q_3 oraz Q_4 po otrzymaniu wiadomości od Q_2 rozpoczynają nową rundę, której liderem jest Q_3. Monitor Q_3 rozgłasza tę wiadomość, a więc wysyła ją również do Q_2. Ponieważ monitor Q_2 podjął decyzję już wcześniej, wiadomość otrzymana od Q_3 jest

przez Q_2 ignorowana – dlatego została pominięta na slajdzie. Po wysłaniu wiadomości monitor Q_3 podejmuje decyzję – w rozważanym przykładzie v_1. Monitor Q_4 otrzymując wiadomość przechodzi do nowej rundy, stwierdza, że jest jej liderem i podejmuje decyzję rozsyłając ją do wszystkich monitorów. Pozostałe monitory ignorują tę wiadomość, gdyż podjęły już decyzję, dlatego wiadomości te pominięto na slajdzie jako nieistotne.

Działanie hierarchicznego algorytmu konsensusu podstawowego: Przykład 2

Kolejny przykład zaczyna się podobnie jak poprzednio. Monitor Q_1 rozgłasza proponowaną wartość do wszystkich, jednakże ponieważ ulega awarii, z własności podstawowego rozgłaszania niezawodnego wynika, że wiadomość może nie dotrzeć do monitora Q_2. Po wykryciu awarii procesu P_1, monitor Q_2 decyduje o rozpoczęciu rundy, której jest liderem. Pozostałe monitory również przechodzą do nowej rundy, po otrzymaniu wiadomości od Q_1. Monitor Q_2 decyduje się na v_2, gdyż nigdy nie widział wartości v_1, i rozsyła tę wartość do pozostałych monitorów. Odebranie tej wiadomości powoduje, że zarówno Q_3 jak i Q_4 przyjmują wartość v_2 jako nową propozycję decyzji (por. wiersz 18 algorytmu). Powoduje to przejście do rundy trzeciej, której liderem ma być Q_3. Ponieważ jednak ulega on awarii, Q_4 przechodzi do nowej rundy i podejmuje decyzję, wybierając wartość v_2.

Hierarchiczny algorytm konsensusu podstawowego: Złożoność

W przypadku, gdy początkowy lider nie ulega awarii, algorytm kończy się po pierwszym kroku. Złożoność czasowa wynosi więc 1, a złożoność komunikacyjna wynosi *n*. W przypadku pesymistycznym, w którym wszystkie procesy po kolei ulegają awarii, algorytm kończy się po co najwyżej *n* krokach, przy czym w każdym kroku jest wysyłanych *n* wiadomości. Złożoność czasowa wynosi więc *n* a komunikacyjna n^2.

Konsensus jednolity: Specyfikacja

Kolejnym mechanizmem konsensusu omawianym dalej jest mechanizm *konsensusu jednolitego*, posiadający własności ważności, zakończenia, integralności i jednolitej zgodności.

Pierwsze trzy własności są identyczne jak w konsensusie podstawowym. Własność *ważności* (ang. *validity*) oznacza, że jeżeli proces P_i decyduje się na wartość v_i, to wartość ta została wcześniej zaproponowana przez pewien proces.

Własność *zakończenia* (ang. *termination*) oznacza, że wszystkie poprawne procesy ostatecznie decydują się na jakąś wartość.

Własność *integralności* (ang. *agreement*) oznacza, że każdy proces decyduje się tylko raz.

Konsensus jednolity odróżnia się dodatkową własnością *jednolitej zgodności*.

Własność *jednolitej zgodności* (ang. *uniform agreement*) oznacza, że żadne dwa procesy, niekoniecznie poprawne, nie decydują się na inne wartości. Innymi słowy, jeżeli jakikolwiek proces podejmie decyzję – nawet jeżeli jest on niepoprawny – to wszystkie inne procesy muszą podjąć identyczną decyzję.

Konsensus jednolity – operacja propose

Proponowana do uzgodnienia przez proces P_i wartość v_i określana jest w wyniku wykonania operacji $\mathbf{propose}^{\mathbf{UC}}(P_i, v_i)$. Operacja ta inicjuje mechanizm konsensusu jednolitego.

Zdarzenie zachodzące w procesie P_i w wyniku wykonania tej operacji zapisywane będzie $e\_propose^{UC}(P_i, v_i)$.

Konsensus jednolity – operacja decide

Określenie przez proces P_i (monitor Q_i) decyzji co do ostatecznie uzgodnionej wartości v następuje w wyniku wykonania operacji $\mathbf{decide}^{\mathbf{UC}}(P_i, v)$. Wykonanie tej operacji kończy mechanizm konsensusu jednolitego. Zdarzenie $e\_decide^{UC}(P_i, v)$ zachodzące w procesie P_i w wyniku wykonania operacji $\mathbf{decide}^{\mathbf{UC}}(P_i, v)$, oznacza osiągnięcie przez proces aplikacyjny P_i konsensusu, a więc uzgodnienia wartości v z wszystkimi procesami.

Rozgłoszeniowy algorytm konsensusu jednolitego: Założenia

Przedstawiona zostanie obecnie modyfikacja rozgłoszeniowego algorytmu konsensusu podstawowego, dzięki której algorytm ten rozwiązuje problem konsensusu jednolitego. Nowy algorytm podobnie jak poprzednio zakłada dostępność mechanizmu podstawowego rozgłaszania niezawodnego oraz doskonałego detektora awarii. Zasadniczą różnicą jest fakt wstrzymania się wszystkich procesów z decyzją aż do *n*-tej rundy, gdzie *n* oznacza liczbę procesów.

Rozgłoszeniowy algorytm konsensusu jednolitego (1)

W przeciwieństwie do algorytmu rozwiązującego konsensus podstawowy, w prezentowanej wersji używany jest tylko jeden typ wiadomości, typ MYPROP. Jego struktura jest identyczna jak poprzednio. Typ ten zawiera więc pole *proposedValue*, które jest zbiorem wartości zaproponowanych w danej rundzie przez poszczególne procesy, oraz pole *roundNo* określające numer rundy.

Rozgłoszeniowy algorytm konsensusu jednolitego (2)

Wszystkie zmienne używane przez wersję algorytmu rozgłoszeniowego dla jednolitego konsensusu są identyczne jak dla wersji podstawowej. Dodana jest jedynie tablica $delivered_i$ zawierająca identyfikatory procesów, od których w danej rundzie otrzymano wiadomość, natomiast zmienna $decided_i$ jest zmienną logiczną określającą, czy proces podjął już decyzję.

Rozgłoszeniowy algorytm konsensusu jednolitego (3)

Rozgłoszeniowy algorytm konsensusu jednolitego jest modyfikacją algorytmu konsensusu zgodnego. Stąd pierwsze kroki są bardzo podobne.

Zainicjowanie rozgłoszeniowego algorytmu konsensusu jednolitego związane jest z zajściem zdarzenia $e\_propose^{UC}(P_i, v_i)$ w procesie aplikacyjnym P_i. W efekcie tego zdarzenia, monitor Q_i rozpoczyna wymianę wiadomości z innymi monitorami w celu uzgodnienia wspólnej decyzji co do wartości v. Uzgodnienie to prowadzi do zajścia zdarzeń $\mathbf{decide}^{\mathbf{UC}}(P_i, v)$.

W wyniku wykrycia awarii z użyciem detektora awarii P, proces P_j jest usuwany ze zbioru poprawnych procesów $correct_i$ widzianych przez P_i.

W przypadku otrzymania propozycji od monitora Q_j wartość z pola *proposedValue* otrzymanej wiadomości MYPROP dodawana jest do zbioru propozycji $proposalSet_i$. Identyfikator procesu dodawany jest do zbioru $delivered_i[roundNo_i]$ identyfikatorów procesów, od których w danej rundzie $roundNo_i$ otrzymano wiadomość z proponowanym zbiorem wartości.

Rozgłoszeniowy algorytm konsensusu jednolitego (4)

W rozgłoszeniowym algorytmie konsensusu jednolitego, w porównaniu do algorytmu konsensusu podstawowego, inny jest warunek podjęcia decyzji. Mianowicie, decyzja podejmowana jest dopiero w *n*-tej rundzie, po warunkiem oczywiście, że monitor nie podjął jeszcze decyzji oraz otrzymano wiadomość od wszystkich monitorów skojarzonych z niepodejrzewanymi w tej rundzie procesami. Monitor decyduje się na minimalny element z zbioru wszystkich nadesłanych propozycji. Jeżeli natomiast $roundNo_i < n$, to następuje przejście do następnej rundy oraz rozesłanie zbioru wartości proponowanych $proposedSet_i$ w kolejnej rundzie.

Hierarchiczny algorytm konsensusu jednolitego: Założenia

Przedstawiona zostanie obecnie modyfikacja hierarchicznego algorytmu konsensusu podstawowego, dzięki której algorytm ten rozwiązuje problem konsensusu jednolitego. Założenia poprawności algorytmu są nadzbiorem założeń dla wersji rozwiązującej problem konsensusu podstawowego, a więc algorytm ten wymaga dostępności mechanizmu podstawowego rozgłaszania niezawodnego i doskonałego detektora awarii. Dodatkowo algorytm wymaga dostępności mechanizmów zgodnego rozgłaszania niezawodnego oraz kanałów niezawodnych.

Podobnie, jak poprzednio przyjmuje się, że hierarchia procesów jest zdefiniowana przez uporządkowanie procesów według ich indeksów. Zasadnicza różnica w algorytmach konsensusu podstawowego i jednolitego związana jest z podziałem każdej rundy na dwa etapy: w pierwszym etapie lider wysyła propozycję decyzji, a w drugim używa zgodnego rozgłaszania niezawodnego do

rozesłania podjętej decyzji. Przejście do nowej rundy następuje tylko wówczas, gdy lider jest podejrzewany.

Hierarchiczny algorytm konsensusu jednolitego (1)

Hierarchiczny algorytm konsensusu jednolitego używa trzech typów wiadomości. Typ PROPOSE, przechowuje wartość proponowanej decyzji w polu *proposedValue* oraz numer rundy (pole *roundNo*), w której tę propozycję wysłano. Typ DECIDE zawiera wartość decyzji w polu *decidedValue*. Typ ACK zawiera numer rundy w polu *roundNo*.

Hierarchiczny algorytm konsensusu jednolitego (2)

Wyjaśnienia wymaga użycie zmiennych, które nie występowały w wersji algorytmu dla konsensusu podstawowego. Oprócz wiadomości *ackOut* typu ACK oraz *propOut* typu PROPOSE pojawia się dodatkowo zbiór $ackSet_i$ zawierający identyfikatory procesów, od których otrzymano potwierdzenia propozycji, oraz tablica $proposed_i$ zawierająca wszystkie propozycje, gdzie *k*-ty element oznacza propozycję nadchodzącą w *k*-tej rundzie.

Hierarchiczny algorytm konsensusu jednolitego (3)

Monitor Q_i zapisuje propozycję procesu P_i w zmiennej $proposal_i$. W przypadku wykrycia awarii jakiegoś procesu P_j, proces ten jest dołączany do zbioru $suspected_i$. Z własności *silnej dokładności* doskonałego detektora awarii P wynika, że żaden poprawny proces nigdy nie będzie podejrzewany.

Hierarchiczny algorytm konsensusu jednolitego (4)

Jeżeli monitor odkrywa, że ma być liderem bieżącej rundy i nie podjął dotąd jeszcze decyzji, rozsyła swoją propozycje za pomocą podstawowego rozgłaszania niezawodnego.

Hierarchiczny algorytm konsensusu jednolitego (5)

Otrzymanie propozycji o numerze rundy *k* powoduje zapamiętanie jej w *k*-tym elemencie tablicy $proposed_i$. Jeżeli numer rundy zawarty w polu *roundNo* odebranej wiadomości PROPOSE jest nie mniejszy od numeru bieżącej rundy, wysyłane jest potwierdzenie do nadawcy. Warunek ten znaczy, że odebrana wiadomość nie pochodzi od niepoprawnego procesu. Przejście do nowej rundy następuje bowiem tylko wtedy, gdy lider jest niepoprawny. Z własności doskonałego detektora awarii wynika, że lider nie byłby podejrzewany, gdyby był poprawny.

Hierarchiczny algorytm konsensusu jednolitego (6)

Jeżeli lider bieżącej rundy jest podejrzewany, rozpoczynana jest nowa runda. W przypadku otrzymania potwierdzenia, dołączane jest ono do zbioru $ackSet_i$.

Hierarchiczny algorytm konsensusu jednolitego (7)

Jeżeli lider otrzymał potwierdzenia od wszystkich poprawnych monitorów, rozsyła swoją decyzję za pomocą niezawodnego rozgłaszania (ang. *regular reliable broadcast*). Po otrzymaniu wiadomości typu DECIDED od lidera, każdy monitor również podejmuje identyczną decyzję.

Hierarchiczny algorytm konsensusu jednolitego: Złożoność

Algorytmy rozwiązujące problem konsensusu jednolitego mają większą złożoność niż rozwiązujące problem konsensusu podstawowego. I tak, rozgłoszeniowy algorytm konsensusu jednolitego kończy się w n krokach wymieniając n^3 wiadomości. Złożoność czasowa wynosi więc *n*, a złożoność komunikacyjna wynosi n^3. Algorytm hierarchiczny, jeżeli lider nie ulegnie awarii, kończy się po 3 krokach: 2 kroki w pierwszej rundzie i dodatkowy krok na niezawodne rozgłaszanie, wymieniając przy tym 3*n* wiadomości. Złożoność czasowa w przypadku optymistycznym wynosi więc 3, a komunikacyjna 3*n*. Każda awaria lidera dodaje dwa dodatkowe kroki i 2*n* wiadomości. Maksymalna liczba awarii wynosi oczywiście *n* -1, a więc w przypadku pesymistycznym, złożoność komunikacyjna jest $O(n^2)$, a złożoność czasowa wynosi 2*n*+1.

Konsensus probabilistyczny: Specyfikacja

Poprzednio omawiane algorytmy wymagały spełnienia dość silnego założenia dostępności doskonałego detektora błędów. Mechanizm konsensusu probabilistycznego (ang. *randomized consensus, probabilistic consensus*), charakteryzujący się własnościami ważności, zgodności, integralności i zakończenia, nie ma tak silnych wymagań.

Pierwsze trzy własności konsensusu probabilistycznego są identyczne, jak w przypadku konsensusu podstawowego.

Własność *ważności* (ang. *validity*) oznacza, że jeżeli proces P_i decyduje się na wartość v_i, to wartość ta została wcześniej zaproponowana przez pewien proces.

Własność *zgodności* (ang. *agreement*) oznacza, że żadne dwa poprawne procesy nie decydują się na inne wartości.

Własność *integralności* (ang. *agreement*) oznacza, że każdy proces decyduje się tylko raz.

Konsensus probabilistyczny odróżnia się jednak sformułowaniem warunku zakończenia.

Własność *zakończenia probabilistycznego* (ang. *probabilistic termination*) oznacza, że wszystkie poprawne procesy, z prawdopodobieństwem równym 1, ostatecznie decydują się na jakąś wartość.

Konsensus probabilistyczny – operacja propose

Proponowana do uzgodnienia przez proces P_i wartość v_i określana jest w wyniku wykonania operacji $\mathbf{propose}^{\mathbf{PC}}(P_i, v_i)$. Operacja ta inicjuje mechanizm konsensusu.

Zdarzenie zachodzące w procesie P_i w wyniku wykonania tej operacji zapisywane będzie $e\_propose^{PC}(P_i, v_i)$.

Konsensus probabilistyczny – operacja decide

Określenie przez proces P_i (monitor Q_i) decyzji co do ostatecznie uzgodnionej wartości v następuje w wyniku wykonania operacji $\mathbf{decide}^{\mathbf{PC}}(P_i, v)$. Wykonanie tej operacji kończy mechanizm konsensusu. Zdarzenie $e\_decide^{PC}(P_i, v)$ zachodzące w procesie P_i w wyniku wykonania operacji $\mathbf{decide}^{\mathbf{PC}}(P_i, v)$, oznacza osiągnięcie przez proces aplikacyjny P_i konsensusu, a więc uzgodnienia wartości v z wszystkimi procesami.

Algorytm konsensusu probabilistycznego: Założenia

Obecnie zostanie przedstawiony algorytm rozwiązujący problem konsensusu probabilistycznego. Algorytm ten jest modyfikacją dobrze znanego probabilistycznego algorytmu rozwiązującego konsensus binarny. Używa on mechanizmów zgodnego oraz podstawowego rozgłaszania. Wymaga on również, by większość procesów pozostała poprawna. Nie wymaga natomiast obecności detektora awarii.

Każda runda podzielona jest na dwie fazy. W pierwszej fazie używane jest podstawowe rozgłaszanie niezawodne, a w drugiej zgodne rozgłaszanie niezawodne do rozsyłania decyzji. W każdej rundzie istnieje prawdopodobieństwo większe od zera, że algorytm się zakończy; tak więc ostatecznie prawdopodobieństwo zakończenia wynosi 1.

Algorytm konsensusu probabilistycznego (1)

Algorytm używa czterech typów wiadomości. Typ INIVAL służy do przesyłania wartości proponowanych przez procesy, która to wartość przenoszona jest w polu *proposedValue*. Typ DECIDE posiada pole *decidedValue* zawierające decyzję oraz numer rundy w polu *roundNo*, w której decyzję tę podjęto. Pozostałe typy posiadają pole *proposedValue* zawierające propozycję decyzji oraz numer w polu *roundNo*, oznaczający numer rundy, w czasie której wysłano wiadomość.

Algorytm konsensusu probabilistycznego (2)

Wiadomość *decOut* jest komunikatem typu DECIDED. Z kolei, *phase1Out, phase2Out* oraz *iniOut* są wiadomościami typu PHASE1, PHASE2 oraz INIVAL. Zmienna $estimate_i$ zawiera przybliżenie decyzji, natomiast zmienna $decided_i$ zawiera decyzję procesu P_i, przy czym zmienne te są zainicjowane specjalnymi wyróżnionymi wartościami pustymi. Zmienna $roundNo_i$ zawiera numer rundy. Tablica

$proposedValues_i$ zawiera wartości początkowe nadesłane przez pozostałe procesy. Tablice zbiorów $phase1_i$ oraz $phase2_i$ zawierają propozycje nadesłane przez procesy w odpowiednio, pierwszej i drugiej fazie danej rundy.

Algorytm konsensusu probabilistycznego (3)

Proponując daną wartość monitor rozsyła ją za pomocą podstawowego rozgłaszania niezawodnego, zapamiętuje swoją propozycję, przypisuje ją do zmiennej $estimate_i$, zwiększa numer rundy oraz następnie rozsyła swoją propozycję ponownie jako wiadomość typu PHASE1.

Algorytm konsensusu probabilistycznego (4)

Otrzymując wartość początkową zaproponowaną przez inny monitor Q_j, monitor Q_i dodaje ją do zbioru wartości początkowych $proposedValues_i$. Z kolei otrzymując wiadomość typu PHASE1 pochodzącej z *k*-tej rundy, monitor Q_i dodaje przenoszoną przez nią wartość do zbioru będącego *k*-tym elementem tablicy $phase1_i$.

Algorytm konsensusu probabilistycznego (5)

W przypadku, gdy monitor Q_i otrzymał od większości procesów wiadomości typu PHASE1 oraz jeszcze do tej pory nie podejmował decyzji, monitor Q_i sprawdza, czy wszystkie wartości w zbiorze $phase1_i$ dla tej rundy są identyczne, równe v. Jeżeli tak, przyjmuje jako $estimate_i$ właśnie v. Jeżeli nie, jako $estimate_i$ przyjmuje wartość pustą. Następnie wartość $estimate_i$ jest rozsyłana za pomocą podstawowego rozgłaszania niezawodnego rozpoczynając drugą fazę rundy.

Otrzymanie wiadomości typu PHASE2 z *k*-tej rundy powoduje dodanie przenoszonej przez nią wartości do zbioru będącego *k*-tym elementem tablicy $phase2_i$.

Algorytm konsensusu probabilistycznego (6)

Po otrzymaniu wiadomości PHASE2 w *r*-tej rundzie od większości procesów, monitor Q_i, o ile jeszcze dotąd nie podjął decyzji, sprawdza najpierw, czy istnieje wartość v różna od wyróżnionej wartości pustej, taka, że wszystkie propozycje ze zbioru będącego *r*-tym elementem tablicy $phase2_i$ równają się v. Jeżeli tak, to monitor Q_i decyduje się na tą wartość i decyzję tę rozsyła za pomocą zgodnego rozgłaszania niezawodnego.

W przeciwnym razie, to znaczy, jeżeli wartości należące do tablicy $phase2_i$ są różne, monitor Q_i wybiera jako swoją nową propozycję w zmiennej $estimate_i$ dowolną, różną od wartości pustej, propozycję ze zbioru będącego *r*-tym elementem tablicy $phase2_i$. Jeżeli taka wartość propozycji nie istnieje (wszystkie elementy tego zbioru równają się wartości pustej), to losuje wartość ze zbioru otrzymanych propozycji początkowych i rozpoczyna fazę pierwszą nowej rundy, rozsyłając wiadomość PHASE1 zawierającą nową propozycję monitora Q_i.

Algorytm konsensusu probabilistycznego (7)

Po otrzymaniu wiadomości typu DECIDE, monitor Q_i podejmuje decyzję, wykonuje operację $\mathbf{decide}^{PC}(P_i, decided_i)$, i tym samym inicjuje zdarzenie $e\_decide^{PC}(P_i, decided_i)$.

Potrzeba randomizacji

Można sobie zadać pytanie, dlaczego w wierszu 37 algorytmu występuje losowe wybranie wartości, zamiast dokonania wyboru za pomocą ściśle określonej, deterministycznej funkcji. Uzasadnienie przedstawia rysunek obrazujący przykładowe wykonanie algorytmu. Wyobraźmy sobie, że każdy monitor, zamiast w wierszu 37 losować wartość, wybiera pierwszą wartość różną od wyróżnionej wartości pustej.

Monitor Q_1 dostaje wartość od Q_2. Ponieważ obie wartości się różnią ($1 \neq 2$), więc Q_1 zgodnie z wierszem 19, wybierze wartość pustą jako propozycję wartości do uzgodnienia. Identyczna sytuacja zajdzie w monitorze Q_2, natomiast monitor Q_3 otrzyma propozycję wartości 2, i ponieważ wszystkie otrzymane wartości są identyczne, zaproponuje dalej 2.

W drugiej fazie może z kolei zajść sytuacja, w której Q_1 otrzymałby wartość pustą od Q_2 i dlatego deterministycznie (zamiast losowo) wybrałby wartość 1. Monitor Q_2 natomiast otrzymałby wartość 2 od Q_3 i ponieważ jest ona różna od wartości pustej, przyjąłby ją jako swoją propozycję. Q_3 z kolei otrzymuje od Q_2 wartość pustą, ale ponieważ sam wcześniej zaproponował wartość różną od wartości pustej, przyjmuje ją jako podstawę do kolejnej rundy. Jak widać, w kolejnej rundzie wszystkie procesy proponują dokładnie takie same wartości jak poprzednio – a ponieważ taka sytuacja może się powtarzać, skutkuje to niebezpieczeństwem, że algorytm nigdy by się nie zakończył. Stąd właśnie wynika potrzeba randomizacji.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości: Założenia

W poprzedniej części wykładu zostały przedstawione różne algorytmy implementujące mechanizm konsensusu. Zostało już wcześniej wspomniane, że dostęp do takich mechanizmów umożliwia łatwe skonstruowanie mechanizmu (abstrakcji) zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości.

Obecnie zostanie prezentowany algorytm korzystający z mechanizmu rozgłaszania niezawodnego i mechanizmu konsensusu podstawowego, zapewniający dostęp do operacji rozgłaszania o własnościach zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (1)

Wiadomości typu PACKET zawierają wiadomości aplikacyjne w polu *data*, oraz identyfikator pierwotnego nadawcy wiadomości w polu *origin*.

Zbiór $delivered_i$ zawiera wiadomości dostarczone już do procesu aplikacyjnego, zaś zbiór $unordered_i$ otrzymane pakiety zawierające wiadomości oczekujące na dostarczenie. Zbiór $decided_i$ zawierać będzie efekt decyzji podjętej w wyniku konsensusu - posortowany zbiór pakietów o identycznym składzie dla każdego procesu.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (2)

Jeżeli proces aplikacyjny chce rozgłosić wiadomość, monitor z nim skojarzony buduje pakiet zawierający tę wiadomość i rozgłasza ją z użyciem mechanizmu zgodnego rozgłaszania niezawodnego.

Monitor otrzymujący pakiet dodaje go do zbioru wiadomości niedostarczonych, czekających na kolejny konsensus.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości (3)

Jeżeli zbiór wiadomości *unordered* jest niepusty, i monitor nie jest w trakcie uzgadniania kolejności wiadomości do odebrania, to proponuje on własny zbiór jako zbiór wiadomości do dostarczenia, na przykład w porządku, w jakim sam je otrzymał.

W ramach konsensusu monitory decydują się na zbiór wiadomości przeznaczonych do dostarczenia. Po uzyskaniu konsensusu każdy monitor dostarcza wiadomości z tego zbioru w uzgodnionym porządku.

W wyniku zastąpienia mechanizmu konsensusu podstawowego mechanizmem konsensusu jednolitego, przedstawiony algorytm realizuje mechanizm jednolitego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości.

Algorytm zgodnego rozgłaszania niezawodnego z globalnym uporządkowaniem wiadomości: Złożoność

Przedstawiony algorytm wymaga 3 kroków i wymiany 3*n* komunikatów. Jego złożoność czasowa wynosi więc 3, a złożoność komunikacyjna wynosi 3*n*.