

Wyzwanie Rozproszonego Konsensusu

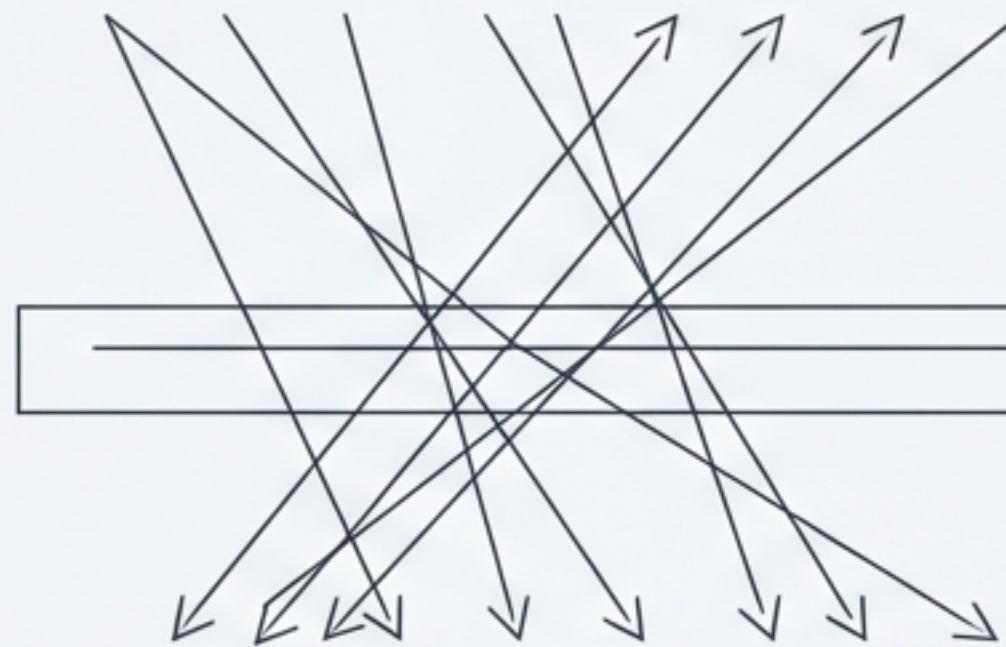
Jak sprawić, by wiele komputerów w zawodnej sieci zgodziło się co do jednej prawdy?

- Wyobraźmy sobie replikowane bazy danych banku: kluczowe jest, aby wszystkie zgadzały się co do kolejności transakcji, nawet jeśli serwery ulegają awarii.
- W systemach rozproszonych wiadomości mogą być opóźnione, docierać w złej kolejności, a serwery mogą nagle zniknąć z sieci.
- Główne wyzwanie: budowanie niezawodnych, spójnych systemów na bazie zawodnej infrastruktury. To problem, który rozwiązuje replikacja maszyny stanów (State Machine Replication - SMR).
- Paxos to fundamentalny protokół leżący u podstaw krytycznych systemów, takich jak Google Chubby, Apache ZooKeeper, etcd i Google Spanner.



Działanie w Świecie Częściowej Synchronii

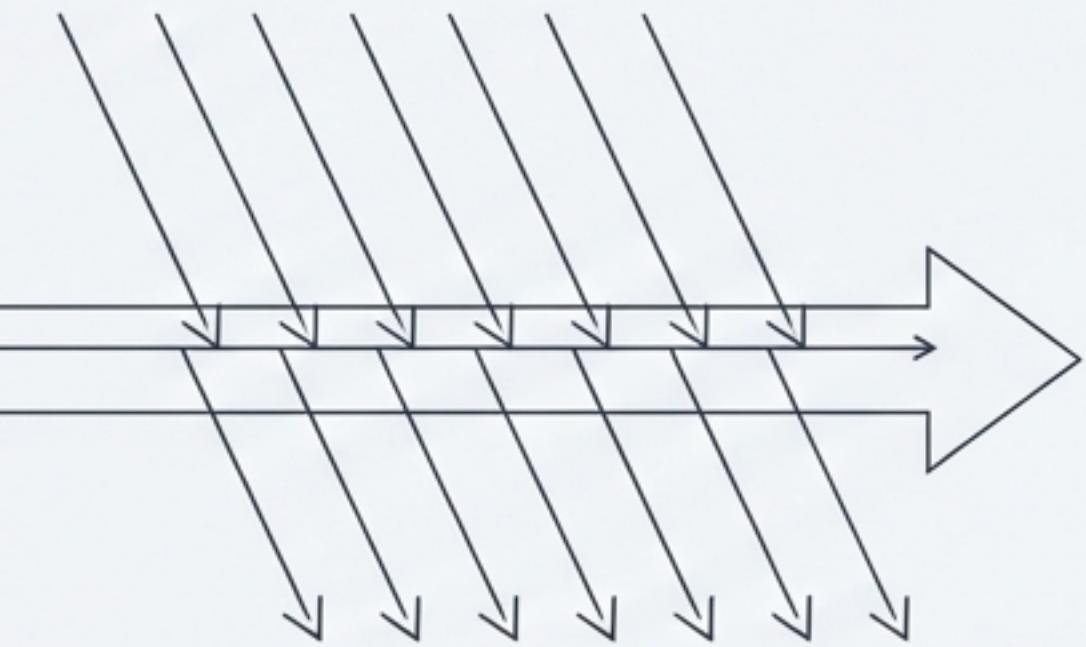
Model asynchroniczny



Model częściowej synchronii



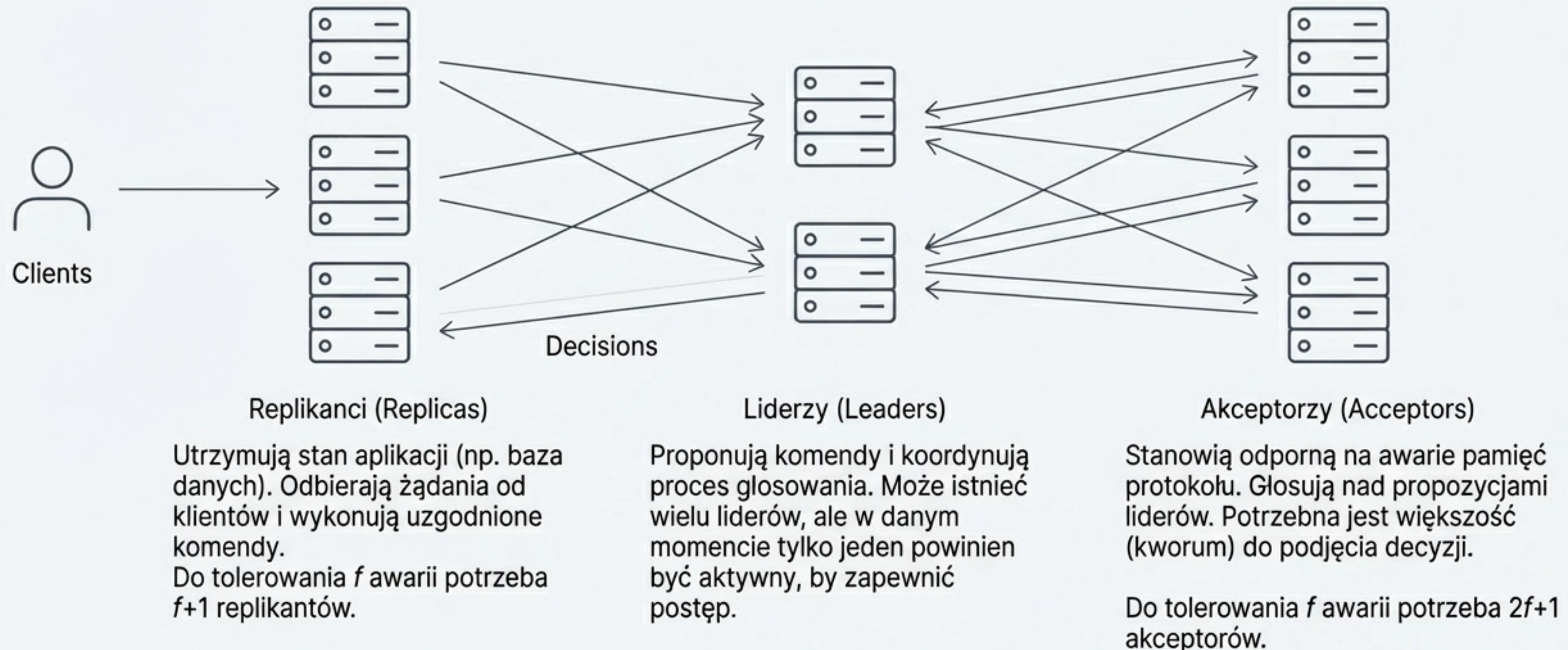
Model synchroniczny



- Brak jakichkolwiek gwarancji czasowych. Słynny wynik niemożliwości FLP dowodzi, że w tym modelu konsensus jest niemożliwy w przypadku choćby jednej awarii.
- Realistyczny kompromis. Zakłada, że sieć, choć przez pewien czas może być asynchroniczna, ostatecznie się stabilizuje i wiadomości zaczynają docierać w określonym (choć nieznanym) czasie.
- Paxos jest zaprojektowany do działania w tym modelu. Gwarantuje **bezpieczeństwo** (safety - nigdy nie zostanie podjęta zła decyzja) przez cały czas, a **życiowość** (liveness - decyzja w końcu zostanie podjęta) jest zapewniona, gdy sieć się ustabilizuje.

Architektura Protokołu Paxos: Podział Ról

- Protokół opiera się na trzech kluczowych rolach. Jeden węzeł może pełnić wiele ról jednocześnie.



Balloty vs. Propozycje: Mechanizm Głosowania

Propozycja (Proposal): Para `(numer slotu, komenda)`.

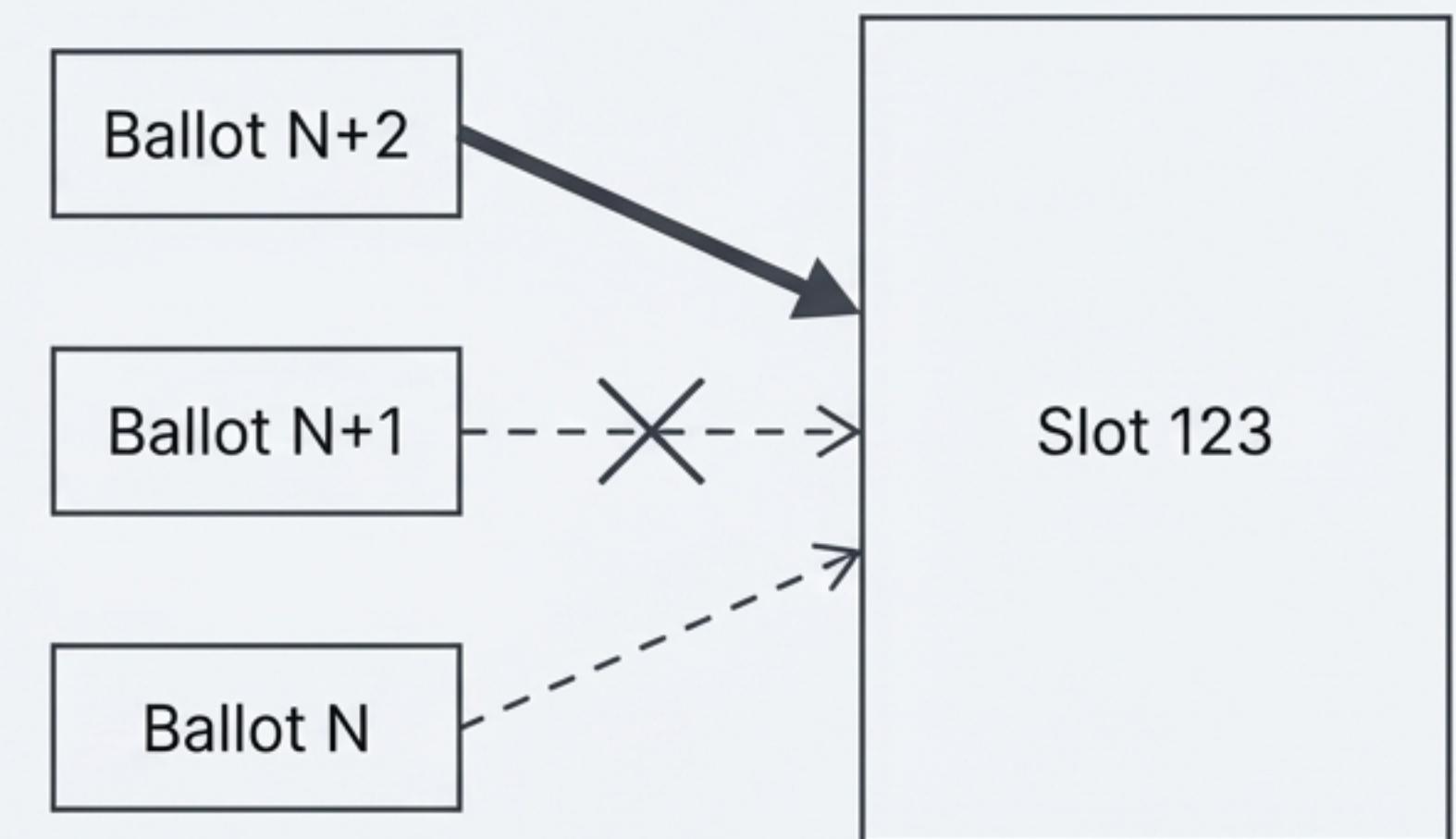
Komenda to operacja, którą chcemy uzgodnić dla danego miejsca w sekwencji (slotu). Np. `(slot 123, "przelej 100 zł z A na B")`.

Ballot: Formalna runda głosowania zainicjowana przez lidera. Każdy ballot ma unikalny, monotonicznie rosnący numer.

Numer Ballotu: Działa jak numer wersji dla rund głosowania. Zapobiega zakłóceniom ze strony starych, opóźnionych wiadomości z poprzednich prób głosowania.

Wyższy numer ballotu zawsze "wygrywa" z niższym, unieważniając jego postęp.

Dla jednego slotu może odbyć się wiele prób głosowania (wiele ballotów), zanim zostanie podjęta ostateczna decyzja.

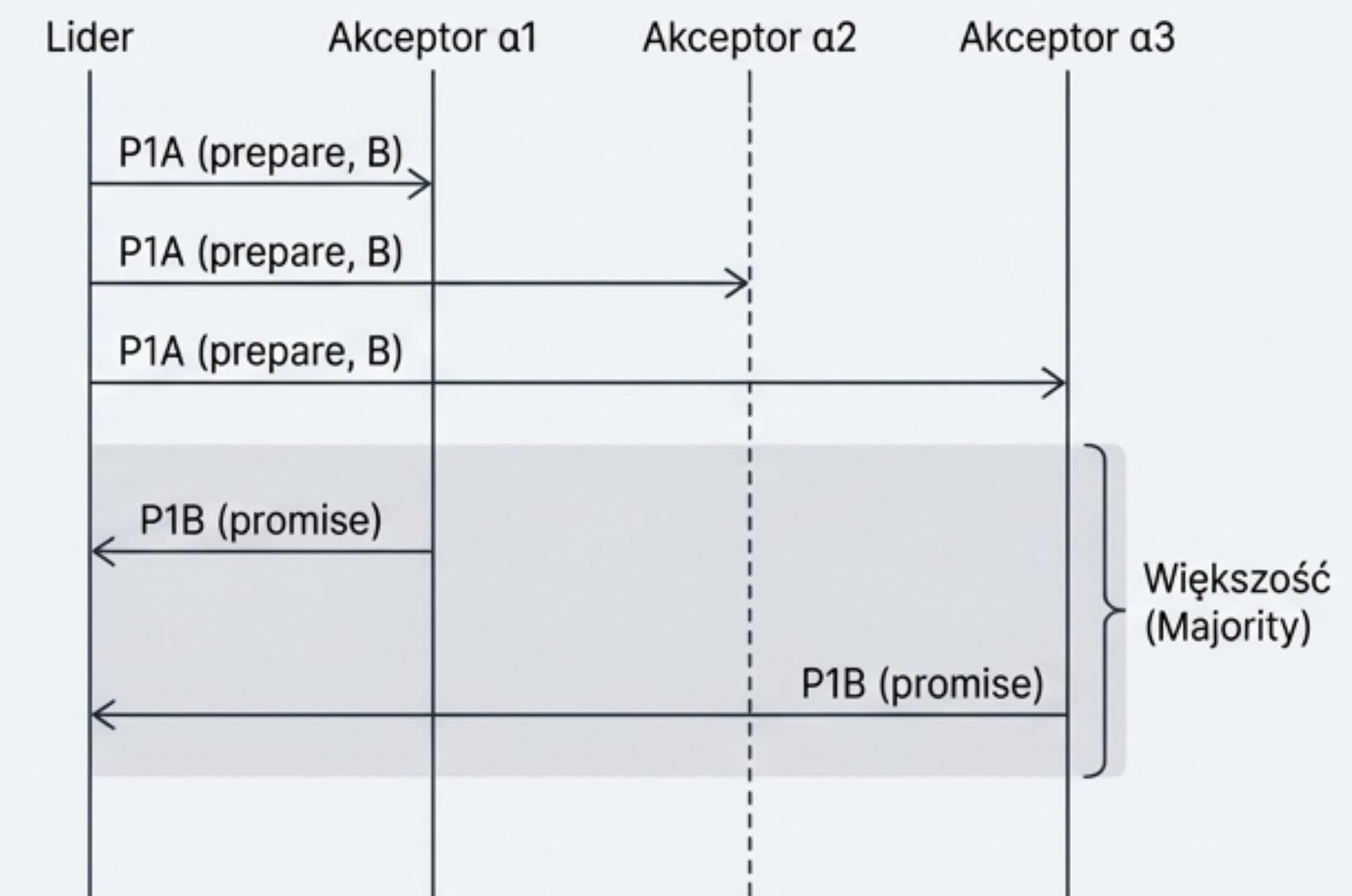


Faza 1: Przygotowanie i Obietnica (Prepare & Promise)

Cel: Lider musi uzyskać zgodę większości akceptorów na rozpoczęcie nowej rundy głosowania (ballotu) i dowiedzieć się, czy coś już zostało wcześniejsza zaakceptowane.

Kroki protokołu:

- 1. **Lider -> Akceptorzy (P1A - Prepare):** Lider wysyła wiadomość `prepare` z nowym, unikalnym numerem ballotu B do wszystkich akceptorów.
 - Pytanie: 'Czy obiecujecie nie akceptować propozycji z ballotów o numerze niższym niż B ?'
- 2. **Akceptorzy -> Lider (P1B - Promise):** Jeśli B jest wyższe niż jakikolwiek numer ballotu, który akceptor widział wcześniej, odpowiada wiadomością `promise`.
 - Odpowiedź: 'Obiecuję. Dodatkowo, oto propozycja o najwyższym numerze ballotu, jaką do tej pory zaakceptowałem.'
- Jeśli lider otrzyma `promise` od większości akceptorów, wie, że jego ballot został przyjęty i może przejść do Fazy 2.

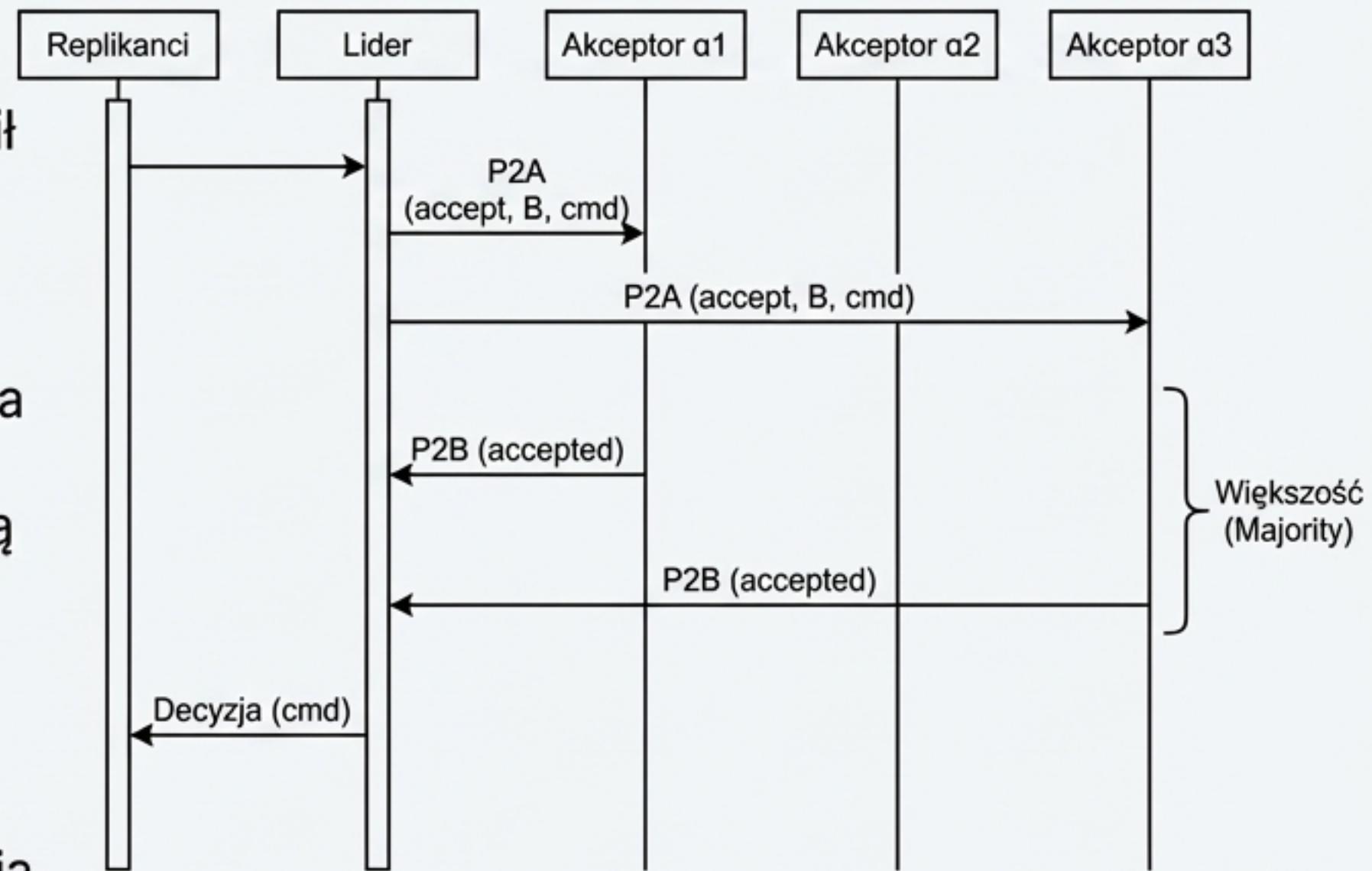


Faza 2: Akceptacja (Accept & Accepted)

Cel: Lider, mając obietnicę od większości, proponuje konkretną komendę do zaakceptowania.

Kroki protokołu:

- **1. Wybór Wartości:** Lider analizuje odpowiedzi `promise` z Fazy 1. Jeśli jakikolwiek akceptor zgłosił wcześniej zaakceptowaną propozycję, lider **musi** zaproponować tę samą komendę. Jeśli nie, lider może zaproponować nową komendę.
- **2. Lider -> Akceptorzy (P2A - Accept):** Lider wysyła do akceptorów (tych, którzy złożyli obietnicę) wiadomość `accept` z numerem ballotu B i wybraną komendą.
- **3. Akceptorzy -> Lider (P2B - Accepted):** Jeśli akceptor nie obiecał niczego nowszemu ballotowi, akceptuje propozycję i odpowiada `accepted`.
- Gdy lider otrzyma `accepted` od większości, decyzja jest podjęta. Lider informuje o tym replikantów.



Optymalizacja: Multi-Paxos dla Wyższej Wydajności

Problem z podstawowym Paxosem:

Uruchamianie pełnego, dwufazowego protokołu (4 kroki komunikacji) dla każdej pojedynczej komendy jest bardzo kosztowne i powolne.

Rozwiązanie Multi-Paxos:

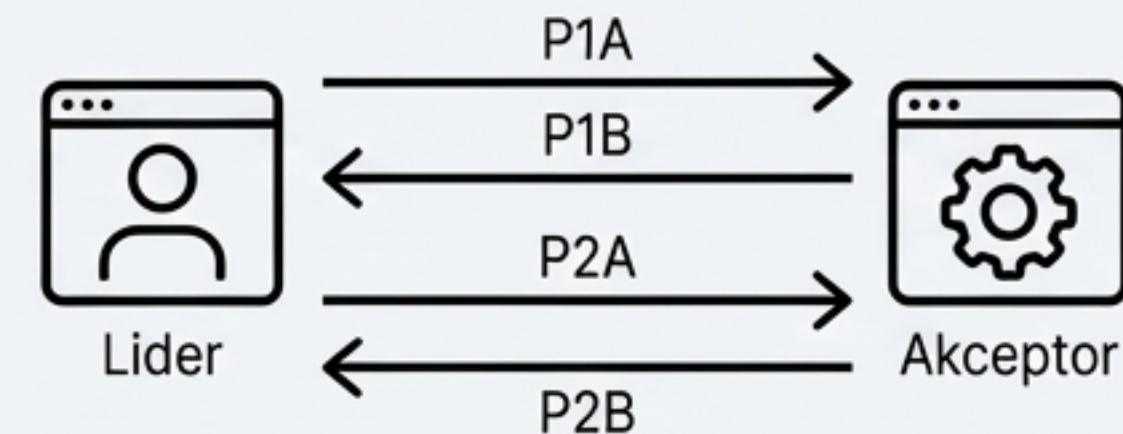
Jeśli jeden lider pozostaje stabilny (nie jest zastępowany przez innego), może pominąć Fazę 1 (Prepare/Promise) dla kolejnych slotów.

- Lider jednorazowo "zdobywa władzę" za pomocą Fazy 1.
- Następnie może proponować komendy dla kolejnych slotów, używając tylko Fazy 2 (Accept/Accepted).

Rezultat:

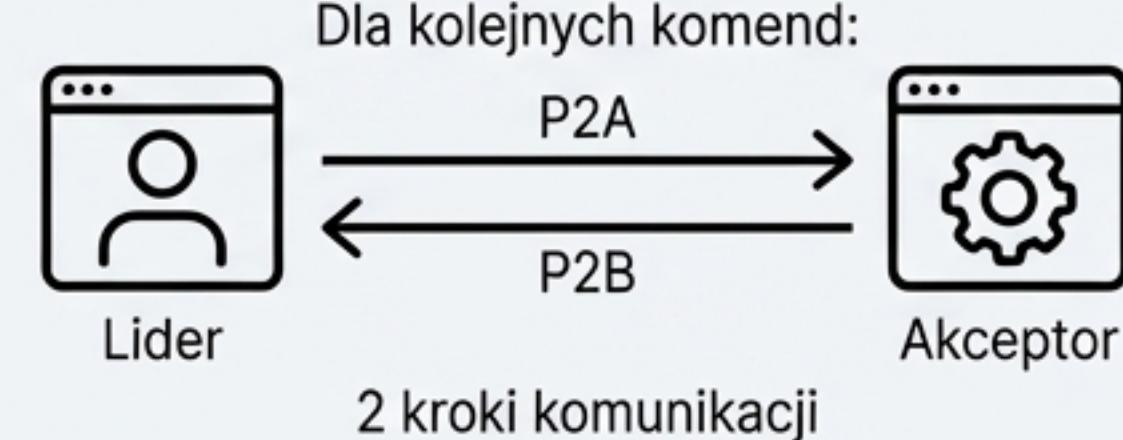
Redukcja opóźnienia z 4 do 2 kroków komunikacji na komendę. To fundamentalna optymalizacja używana w prawie wszystkich rzeczywistych implementacjach Paxos.

Before: Basic Paxos



4 kroki komunikacji

After: Multi-Paxos (Stabilny Lider)



Dla kolejnych komend:

2 kroki komunikacji

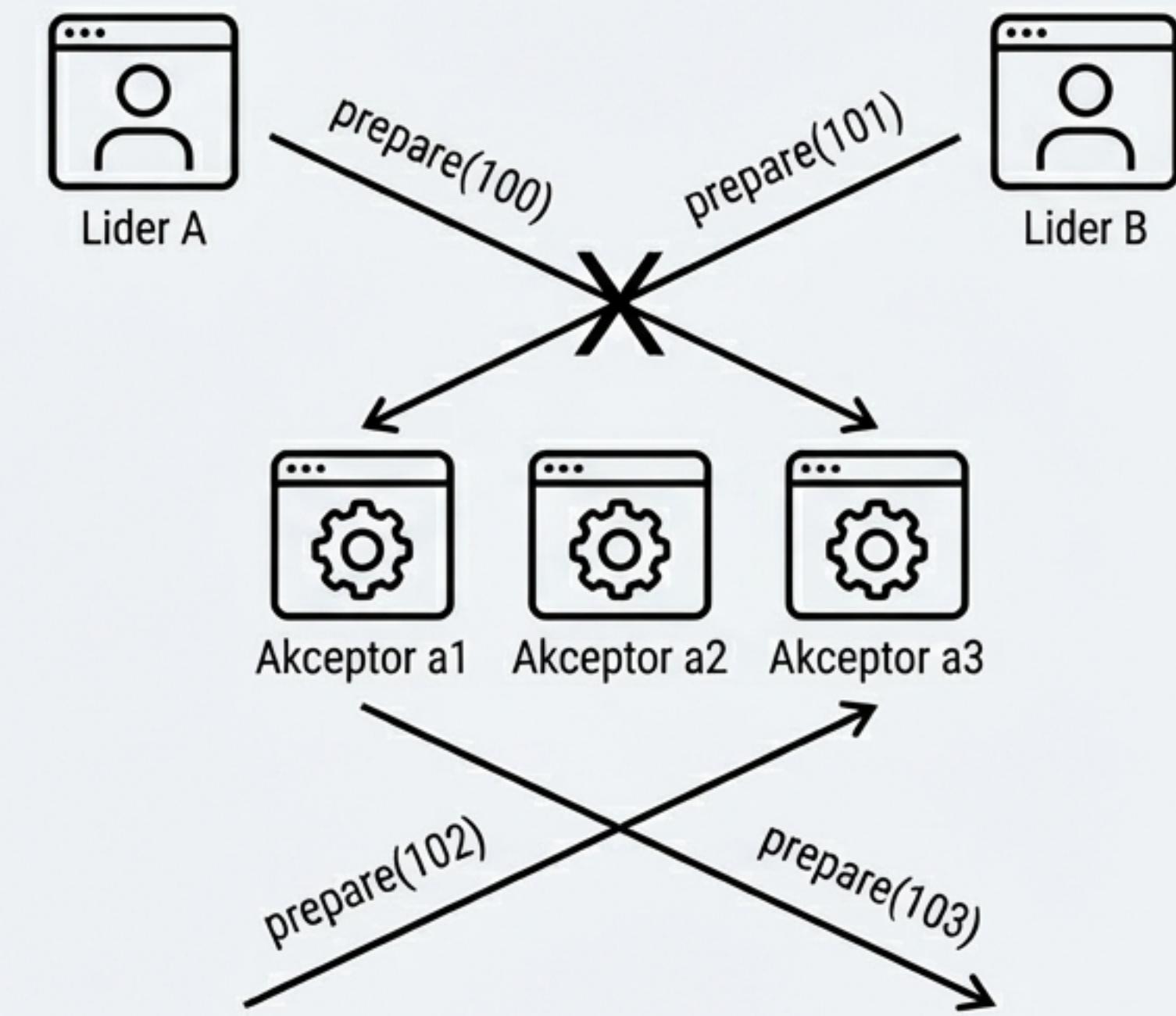
Wybór Lidera i Problem Rywalizujących Liderów

Jak lider staje się aktywny?

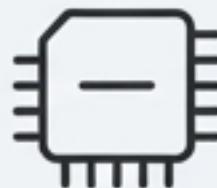
- Nowy lider musi najpierw pomyślnie ukończyć Fazę 1, aby 'przejąć' większość akceptorów. Ten proces jest realizowany przez tzw. wątek 'Scout'.
- Gdy lider jest aktywny, używa wątków 'Commander' do prowadzenia Fazy 2 dla poszczególnych komend.

Problem: Livelock (wzajemne blokowanie)

- Dwóch (lub więcej) liderów może wpaść w pętlę, w której nieustannie unieważniają sobie nawzajem balloty.
- Przykład: Lider A rozpoczyna ballot 100. Lider B natychmiast go unieważnia, rozpoczynając ballot 101. Lider A w odpowiedzi rozpoczyna ballot 102, i tak w nieskończoność.
- Rozwiążanie: Losowe czasy oczekiwania (randomized timeouts). Zamiast natychmiastowej reakcji, lider, którego ballot został unieważniony, odczuwa losowy czas przed rozpoczęciem nowego. To 'rozsynchonizuje' konflikt.



Praktyczne Optymalizacje Implementacyjne



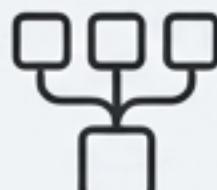
- Redukcja stanu: Akceptorzy nie muszą przechowywać pełnej historii. Wystarczy, że dla każdego slotu pamiętają tylko propozycję z najwyższym zaakceptowanym numerem ballotu.



- Usuwanie zbędnych danych (Garbage Collection): Gdy decyzja dla danego slotu jest już znana wszystkim replikantom, liderzy i akceptorzy mogą bezpiecznie usunąć powiązane z nim dane.



- Optymalizacja operacji tylko do odczytu: Zapytania, które nie modyfikują stanu, nie muszą przechodzić przez pełny protokół Paxos. Można je obsłużyć szybciej, np. za pomocą mechanizmu dzierżawy (leases).



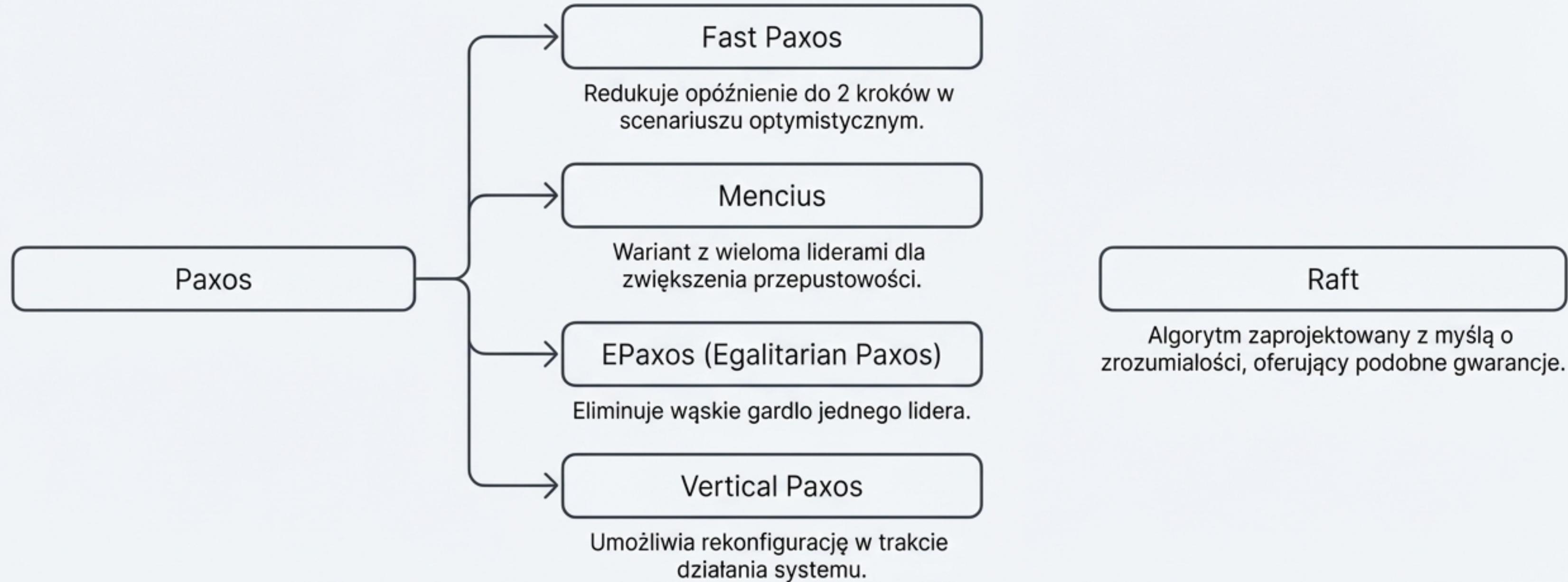
- Batching: Łączenie wielu małych komend w jedną dużą propozycję, aby zmniejszyć narzut protokołu.



- Pipelining: Lider może rozpoczętać proces głosowania dla slotu 'N+1', nie czekając na ostateczne zakończenie głosowania dla slotu 'N'.

Ewolucja Konsensusu: Warianty i Alternatywy Paxos

Paxos stał się podstawą dla wielu innych algorytmów, które optymalizują różne aspekty, takie jak opóźnienie, przepustowość czy złożoność.



Paxos dostarcza jednej potężnej odpowiedzi. Ale poszukiwania trwają.

Jak osiągnąć konsensus w świecie, gdzie nic nie jest pewne?

