

Problem Bizantyjskich Generałów: Metafora

Grupa dowódców wojskowych otacza wrogie miasto i musi uzgodnić wspólny plan bitwy.

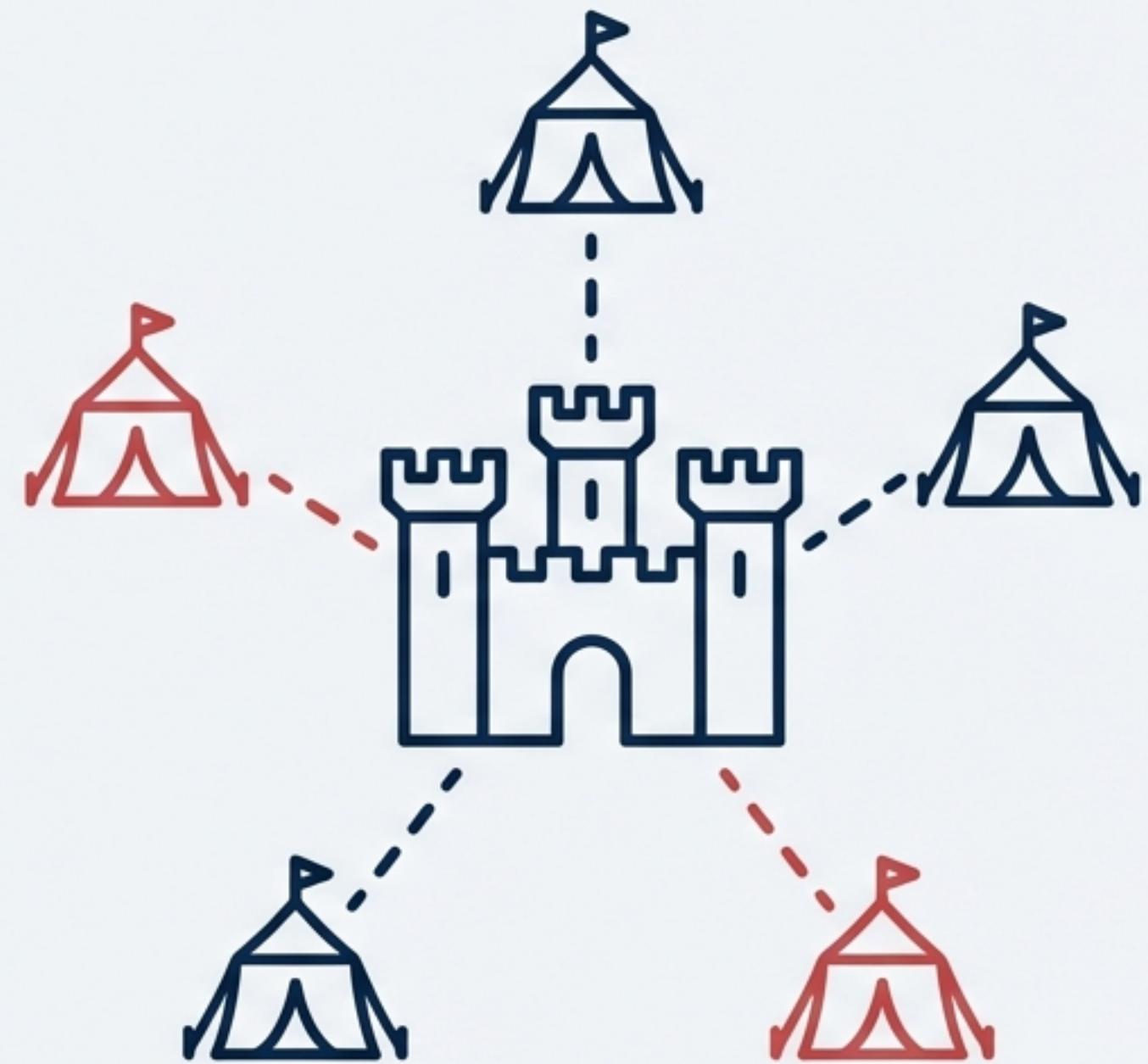
Komunikują się wyłącznie za pośrednictwem posłańców.

Problem: Niektórzy generałowie mogą być zdrajczami, próbującymi zmylić pozostałych.

Główne wyzwanie: Znaleźć algorytm, który zapewni, że wszyscy lojalni generałowie osiągną porozumienie.

Jak sformułowano w oryginalnej pracy: „Sytuację tę można abstrakcyjnie wyrazić w kategoriach grupy generałów armii bizantyjskiej obozujących ze swoimi wojskami wokół wrogiego miasta”.

Sformalizowany w 1982 r. przez Leslie Lamporta, Roberta Shostaka i Marshalla Pease'a.



Warunki Spójności Interaktywnej (Interactive Consistency)

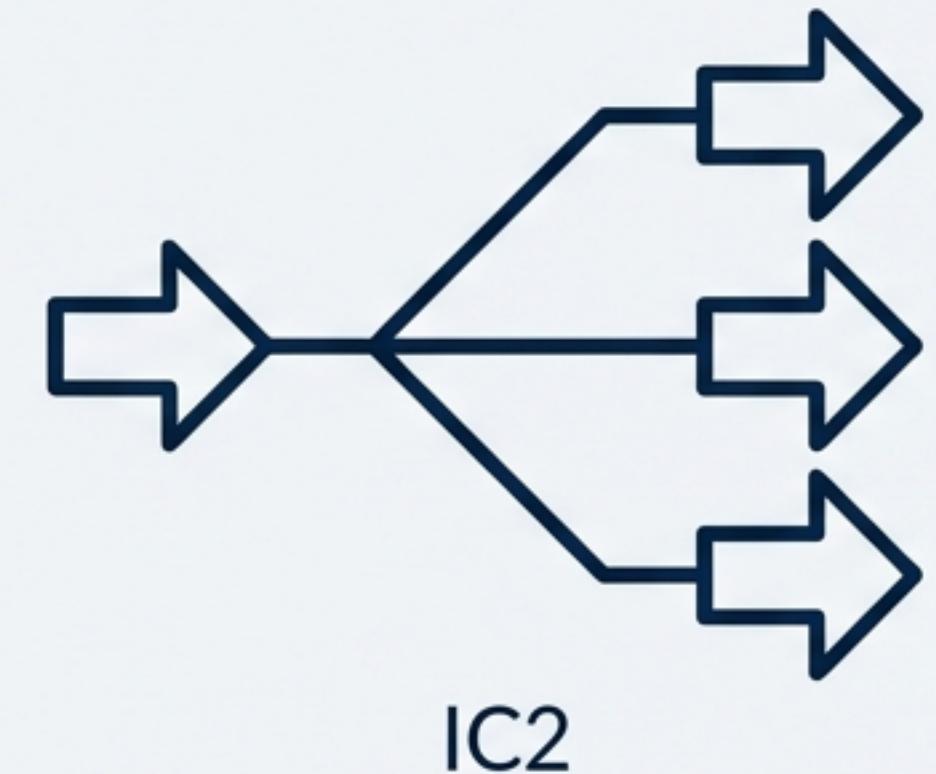
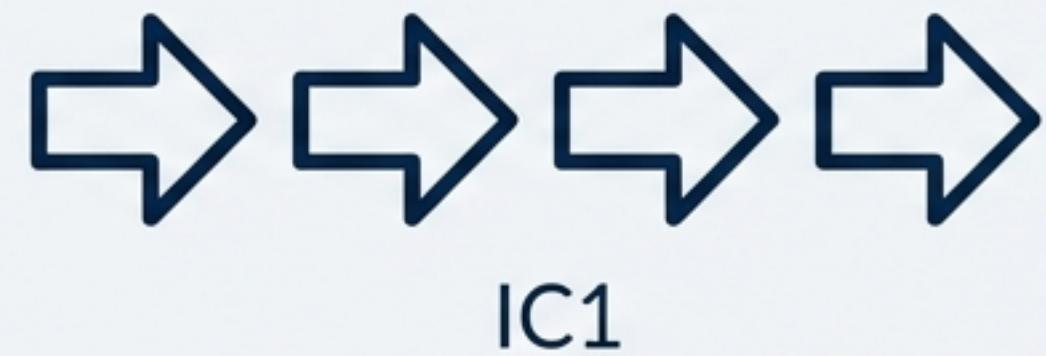
Każde rozwiązanie musi spełniać dwa rygorystyczne warunki:

- **IC1:** Wszyscy lojalni porucznicy wykonują ten sam rozkaz.
- **IC2:** Jeśli głównodowodzący generał jest lojalny, każdy lojalny porucznik musi wykonać jego rozkaz.

Warunek IC1 zapobiega podziałowi lojalnych sił, co prowadziłoby do katastrofy.

Warunek IC2 zapewnia, że system nie pozwoli zdrajcom zagłuszyć rozkazu uczciwego dowódcy.

Uwaga: problemem nie są proste awarie, ale komponenty, które aktywnie wysyłają sprzeczne i wprowadzające w błąd informacje.



Błąd Bizantyjski: Aktywny Sabotaż, a Nie Pasywna Awaria

Zwykła awaria (Fail-stop): Komponent po prostu przestaje działać lub wysyłać komunikaty.

Błąd bizantyjski: Uszkodzony komponent wykazuje nieprzewidywalne, złośliwe zachowanie. Może wysyłać sprzeczne informacje do różnych części systemu.

Przykład: Wysyła `wartość=5` do jednego procesora, a `wartość=10` do innego.

Przekładając metaforę na systemy komputerowe:

- **Generałowie** → Procesory w systemie rozproszonym
- **Posłańcy** → Sieć komunikacyjna
- **Zdrajcy** → Wadliwe komponenty o złośliwym zachowaniu

Błąd Bizantyjski



Zwykła Awaria



Wynik Niemożliwości: Próg 1/3 dla Wiadomości Ustnych

Przy użyciu ‘wiadomości ustnych’ (nieweryfikowalnych, jak standardowe pakiety sieciowe), problem jest **nierozwiązywalny**, jeśli liczba zdrajców stanowi co najmniej $1/3$ wszystkich generałów.

To nie jest kwestia trudności – to jest **matematycznie niemożliwe**. Żaden algorytm nie może zagwarantować spójności w takich warunkach.

Dowód: „Jeden zdrajca może zdezorientować dwóch lojalnych generałów”.

Aby system był odporny na **M** zdrajców, musi składać się z co najmniej **$3M + 1$** generałów:

1 zdrajca wymaga minimum 4 generałów.

2 zdrajców wymaga minimum 7 generałów.



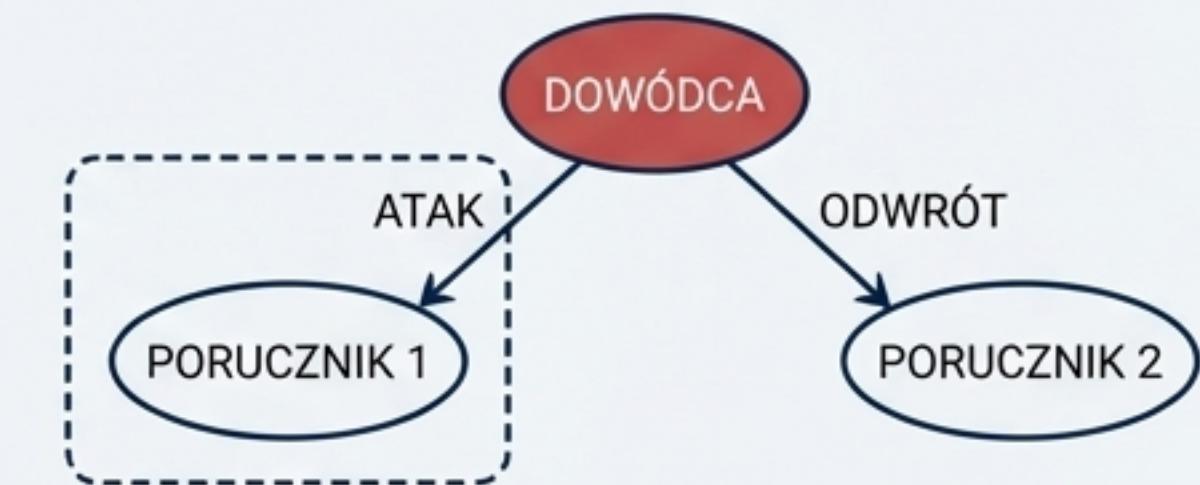
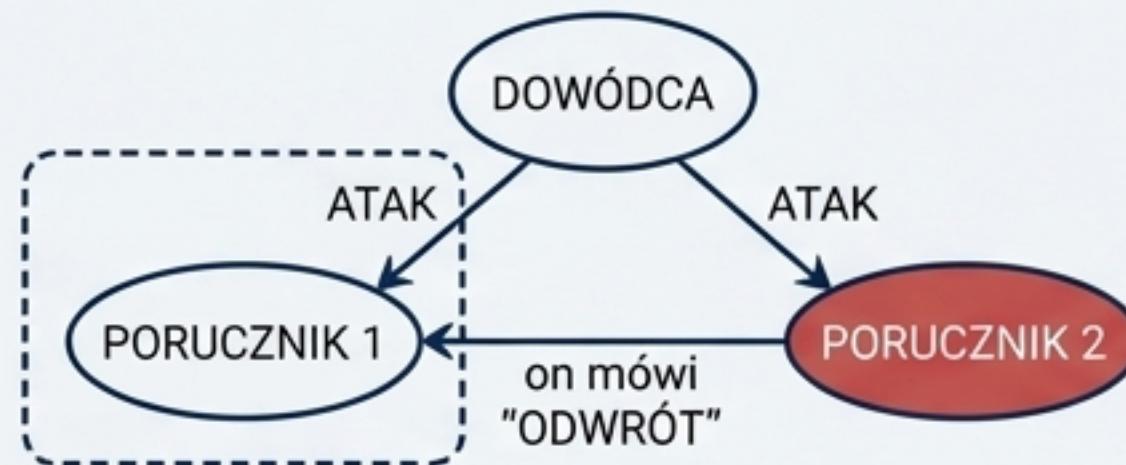
Generals $\geq 3M + 1$

Dowód Niemożliwości dla Trzech Generałów (1 Dowódca, 2 Poruczników)

Scenariusz A: Dowódca jest lojalny, Porucznik 2 to zdrajca.*

1. Dowódca wysyła "ATAK" do obu poruczników.
2. Zdrajca (P2) mówi P1: "Dowódca rozkazał ODWRÓT".
3. P1 otrzymuje sprzeczne wiadomości: "ATAK" (od dowódcy) i "ODWRÓT" (od P2).

Zgodnie z IC2 musi usłuchać dowódcy i **atakuje**.



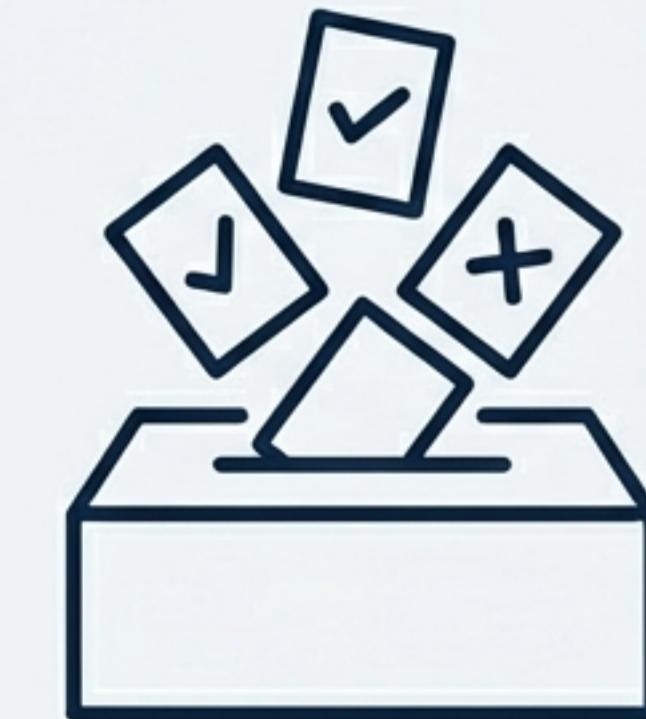
Scenariusz B: Dowódca to zdrajca.*

1. Dowódca wysyła "ATAK" do P1, a "ODWRÓT" do P2.
2. Z perspektywy P1, sytuacja jest **identyczna** jak w Scenariuszu A. Otrzymał "ATAK" od dowódcy.
3. Ponieważ P1 nie jest w stanie odróżnić tych dwóch scenariuszy, musi postąpić tak samo – **atakuje**.

Wniosek: W scenariuszu B, P1 (lojalny) atakuje, a P2 (lojalny) wykonuje rozkaz "ODWRÓT". Warunek IC1 (jedność działania) zostaje złamany.

Rozwiązanie 1: Algorytm Wiadomości Ustnych (OM)

- Algorytm **OM(M)** rozwiązuje problem dla **3M+1** lub więcej generałów, tolerując do **M** zdrajców.
- Działa na zasadzie **rekurencji** – problem jest rozbijany na mniejsze podproblemy,niczym rosyjska matrioszka.
- Kroki algorytmu **OM(M)**:
 1. Dowódca wysyła swoją wartość do każdego porucznika.
 2. Każdy porucznik i działa jako dowódca w algorytmie **OM(M-1)**, aby wysłać otrzymaną wartość do n-2 pozostałych poruczników.
 3. Każdy porucznik podejmuje ostateczną decyzję na podstawie głosowania większościowego ze wszystkich zebranych wartości (swojej od dowódcy i tych przekazanych przez innych poruczników).



Przykład Algorytmu OM(1): 4 Generałów, 1 Zdrajca

Lojalny Dowódca (D) wysyła 'ATAK' do trzech poruczników: P1, P2 i zdrajcy Z3.

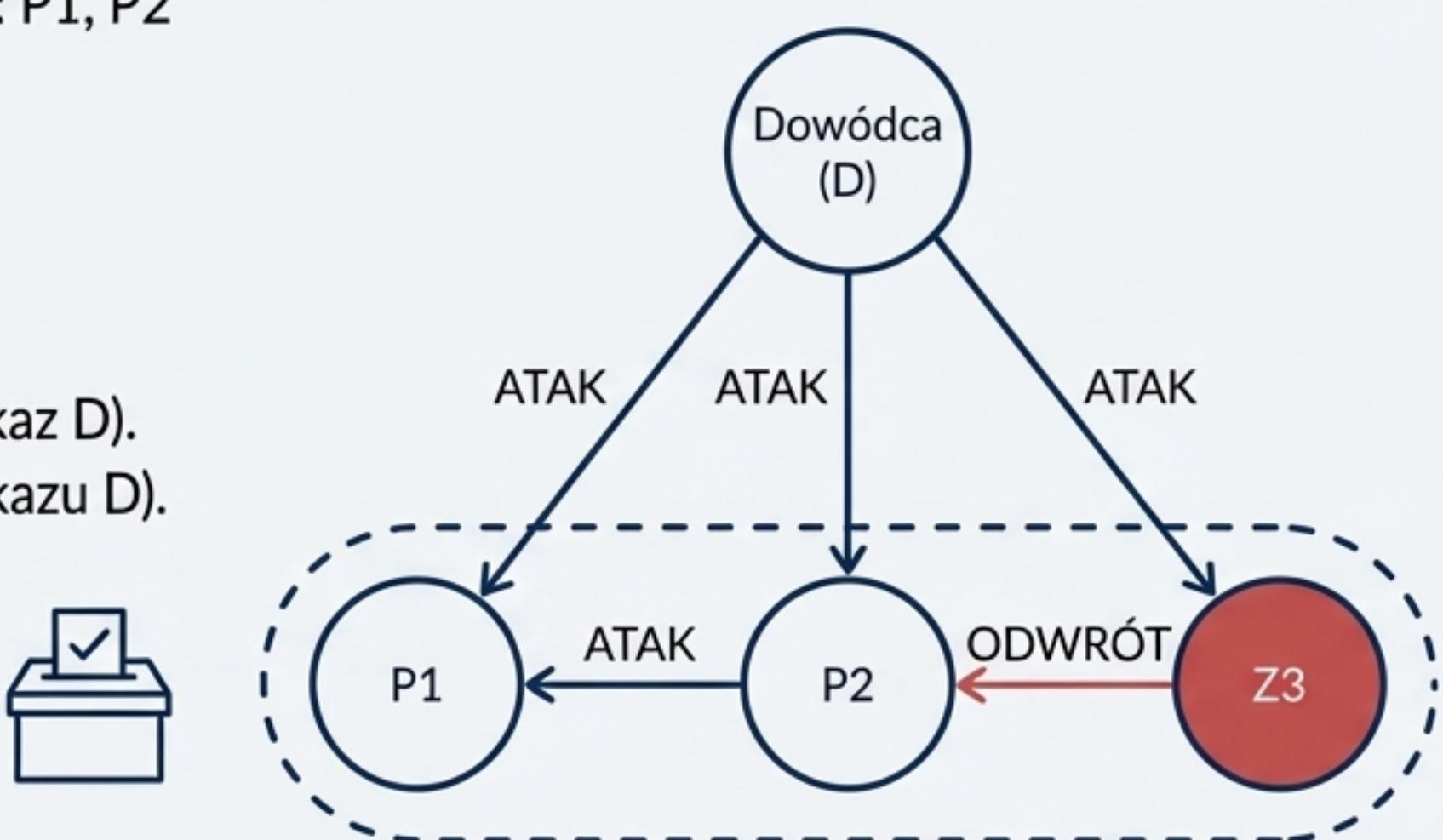
Z perspektywy lojalnego Porucznika P1:

1. Otrzymuje 'ATAK' bezpośrednio od Dowódcy (D).
2. Następnie pyta pozostałych. Otrzymuje:
 - 'ATAK' od lojalnego P2 (który wiernie przekazuje rozkaz D).
 - 'ODWRÓT' od zdrajcy Z3 (który kłamie na temat rozkazu D).

Głosowanie większościowe Porucznika P1:

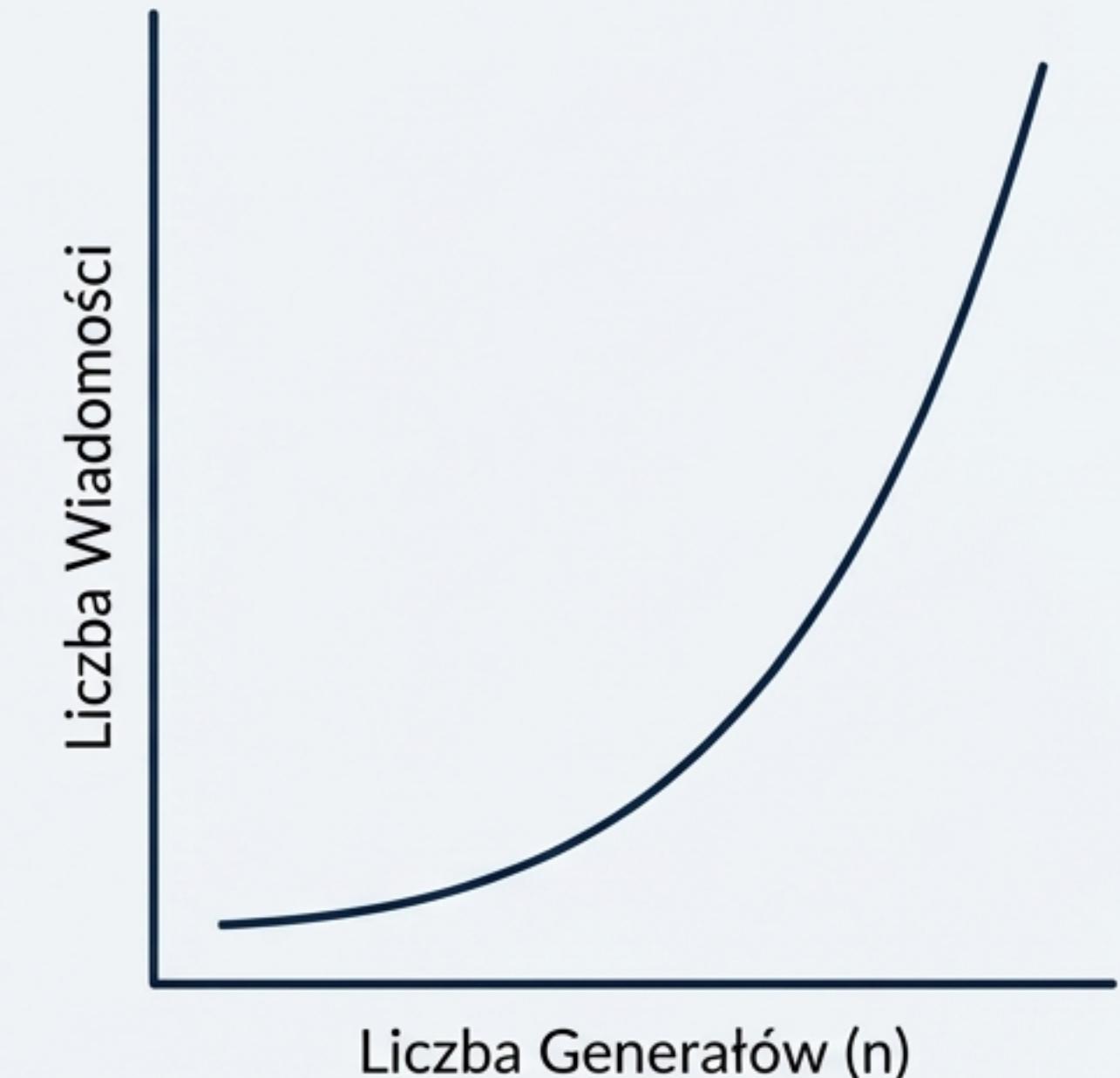
- Wektor wartości: $v = ['ATAK', 'ATAK', 'ODWRÓT']$
- Wynik: $\text{majority}(v) = \text{ATAK}$

P2 przeprowadza identyczne rozumowanie i również decyduje się na **ATAK**. Konsensus zostaje osiągnięty wbrew działaniom zdrajcy.



Słabość Algorytmu OM: Wykładniczy Koszt Komunikacji

- Algorytm OM(M) jest skuteczny, ale niezwykle kosztowny.
- Liczba wymaganych wiadomości rośnie wykładniczo wraz z liczbą tolerowanych zdrajców (M). Zgodnie z pracą, algorytm wymaga wysłania do $^{(n-1)(n-2)...(n-m-1)}$ wiadomości.
- To czyni go niepraktycznym dla dużych systemów rozproszonych.
- Istnieje fundamentalny kompromis między poziomem niezawodności a wydajnością.
- Problem dodatkowo komplikuje się, gdy sieć nie jest w pełni połączona (nie każdy generał może komunikować się z każdym innym).



Rozwiążanie 2: Algorytm Wiadomości Podpisanych (SM)

Wprowadzamy nowe założenie (A4):

- (a) Podpis lojalnego generała nie może być podrobiony.
- (b) Każdy może zweryfikować autentyczność podpisu generała.

Jak to działa:

1. Dowódca wysyła rozkaz z cyfrowym podpisem:
`'rozkaz:podpis_D'`.
2. Gdy porucznik P1 otrzymuje rozkaz, weryfikuje podpis, a następnie dodaje własny i przesyła go dalej:
`'rozkaz:podpis_D:podpis_P1'`.
3. Tworzy to **weryfikowalny łańcuch dowodów**.

Efekt: Zdrada staje się jawną. Jeśli zdradziecki dowódca wyśle dwa różne podpisane rozkazy, porucznicy będą mieli kryptograficzny dowód jego zdrady. Algorytm ten może poradzić sobie z dowolną liczbą zdrajców.



[rozkaz:podpis_D]

→ [rozkaz:podpis_D:podpis_P1]

→ [rozkaz:podpis_D:podpis_P1:podpis_P2]

Problem Bizantyjski w Prawdziwym Świecie

Zasady spójności interaktywnej są fundamentem dla krytycznych systemów, w których pojedynczy błąd może mieć katastrofalne skutki.

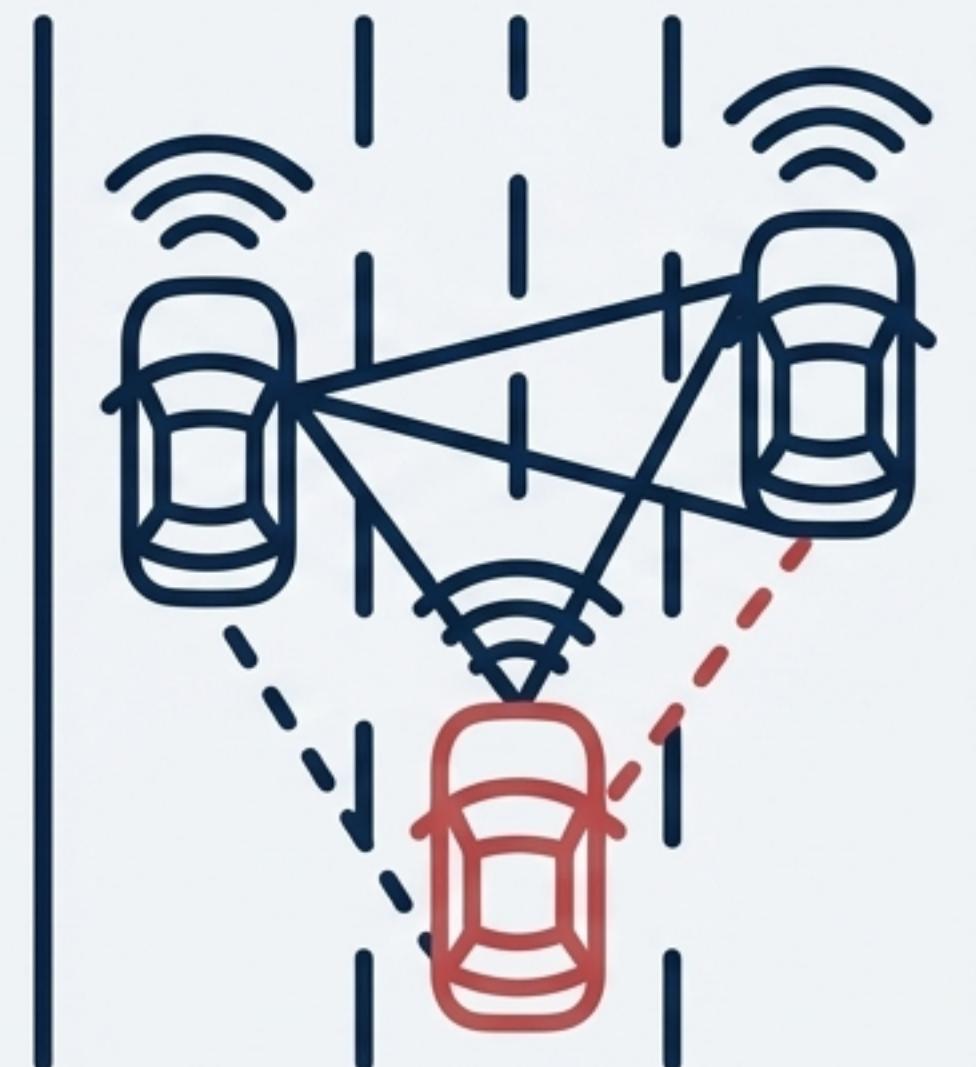
- **Lotnictwo:** Systemy sterowania lotem (fly-by-wire) używają wielu komputerów, które nieustannie 'głosują' nad decyzjami, aby odfiltrować błędne dane z jednego wadliwego czujnika.
- **Systemy rozproszone:** Infrastruktura chmurowa i globalne bazy danych muszą zapewniać spójność danych na serwerach na całym świecie, mimo potencjalnych awarii.
- **Blockchain i Kryptowaluty:**
 - Cała sieć Bitcoin jest systemem odpornym na błędy bizantyjskie (BFT).
 - Górnicy (minerzy) to tysiące 'generałów' próbujących osiągnąć globalny konsensus co do jednej, prawdziwej wersji historii transakcji.



Pytanie na Przyszłość: Generałowie w Erze AI

Jak Problem Bizantyjskich Generałów wygląda w świecie autonomicznych systemów?

Wyobraź sobie flotę połączonych siecią autonomicznych pojazdów, które muszą wspólnie podjąć decyzję w ułamku sekundy, aby uniknąć kolizji na autostradzie.



Jeden z pojazdów ma uszkodzony czujnik (lub został zhakowany) i transmituje fałszywe, "złośliwe" dane o swoim położeniu i prędkości.

W jaki sposób reszta floty może osiągnąć bezpieczny, ratujący życie konsensus, ignorując dane od "zdrajcy" i unikając katastrofy?