Odporność na błędy (fault tolerance)

Są 2 rodzaje błędów sieci: złośliwe i łagodne (ang. benign)
* łagodne – wierz lub kraw przestaje działać „na zawsze”

Problemy które rozważymy:
* 2 generałów (coordinated attack problem)
  chodzi o podjęcię wspólnej decyzji o ataku...
  psują się połączenia (komunikaty nie są dostarczane)
  x_i – chęć Pi, y_i – ostateczna decyzja Pi
  formalnie: mamy 2 procesory P1 i P2; mają być spełnione warunki:
  1. „agreement”, y1=y2, czyli wspólna decyzja...
  2. „validity”, if x1=x2=0, żaden kom. nie został dost., to y1=y2=0
  3. „nontriviality”, istnieje egzekucja, w której y1=y2=1
* consensus
  każdy procesor Pi ma liczbę x_i na wejściu,
  chodzi o wybranie wspólnej liczby y_i spośród x_i,
  przez te proc które się nie popsuły... („crash” procesorów)
  formalnie: mamy procesory P1, .., Pn; mają być spełnione warunki:
  1. „agreement”, y_i=y_j dla wsz niepopsutych Pi, Pj
  2. „validity”, y_i in {x1, ..., xn}, dla wsz niepopsutych Pi

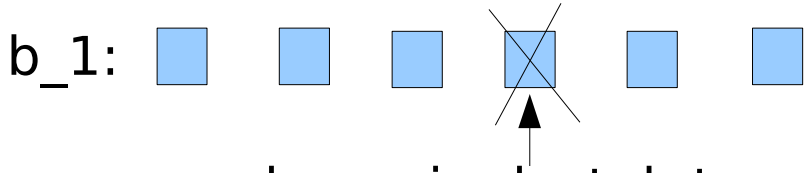
2 generałów

Tw: nie istnieje alg rozwiązujący problem 2 generałów

Dowód:
def: a|P = egzekucja „a” z pkt widzenia procesora P
def: a_1/P/a_2 = egz a_1 i a_2 są nierozróżnialne z pkt widzenia P

Zakładamy, że alg istnieje, czyli istnieje egz b_1, w której dostarcza się „k” komunikatów oraz y1=y2=1 (z nontriviality).

Konstruujemy egzekucje a_k, która wygląda tak:



Zauważmy, że:
b_1/P1/a_k oraz w a_k dostarcza się k-1 kom, y1=y2=1 (?)
Następnie tworzymy egz a_{k-1} z a_k *analogicznie* (jak a_k z b_1)
czyli w a_{k-1} dostarcza się k-2 kom, y1=y2=1
...
w a_1 dostarcza się 0 kom, y1=y2=1, czy to już sprzeczność ???

2 generałów (c.d.)

W egz a_1 dostarcza się 0 kom, y1=y2=1,
czy to już sprzeczność z validity ?
  NIE, no nie wiemy czy x1=x2=0 ?!?!

Konstruujemy egz b'_0 = „a_1 z tą zmianą, że x1=0”
a_1/P2/b'_0 => w egz b'_0: y1=y2=1 (P2 nie widzi innej wart x1)

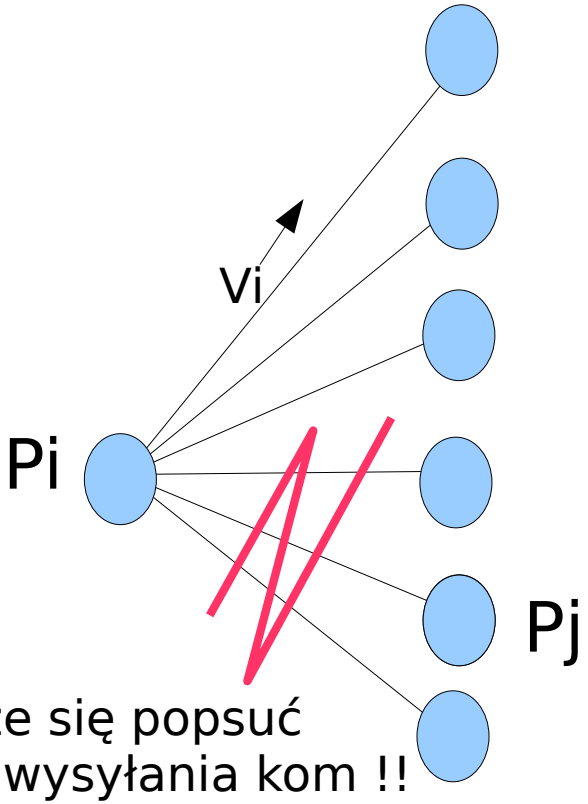
Konstruujemy egz b_0 = „b'_0 z tą zmianą, że x2=0”
b'_0/P1/b_0 => w egz b_0: y1=y2=1 (P1 nie widzi innej wart x2)
tzn. b_0 jest egz w której x1=x2=0, y1=y2=1, dost się 0 kom

!!! SPRZECZNOŚĆ (z validity) !!!

Consensus

Zał: graf pełny, we: x_i, wy: y_i,
f:= liczba psujących się procesorów

Proc consensus (dla procesora Pi)
Vi= {x_i}
for k=1 to f+1 do:
wyślij Vi do wszystkich sąsiadów
czytaj V1, ..., Vn od sąsiadów
Vi:= suma_j Vj (suma zbiorów)
y_i= min Vi



Proc może się popsuć
w czasie wysyłania kom !!

Tw: dla wsz niepopsutych Pi, Pj, mamy Vi=Vj, czyli yi=yj
Dowód: udowodnimy implikację „x in Vi => x in Vj”
r:= pierwsza iter, w której x został dodany do Vi
1. r<=f ??? x in Vj (bo jest jeszcze kolejna iteracja...)
2. r=f+1 to ???
musi istnieć ciąg proc, psujących się zanim wysłały x do Pi

$$P_{i_1}, P_{i_2}, \ldots, P_{i_f}, P_{i_{f+1}}, P_i$$

iteracje: 1 f-1 f f+1

Consensus (c.d.)

$$P_{i_1}, P_{i_2}, \ldots, P_{i_f}, P_{i_{f+1}}, P_i$$

iteracje: 1 f-1 f f+1

pierwszy proces ma x na WE...

„iteracje”: w której dany proces dostaje x,
oraz go wysyła w następnej iteracji (ew. crash) !

Są 2 możliwości:
1. przedostatni proces się psuje – sprzeczność (co do f) !!!
2. -''- się NIE psuje – wysyła x do Pj

Intuicja algorytmy consensus:
wysyłam cenna informację gdzie się da, może przetrwa...