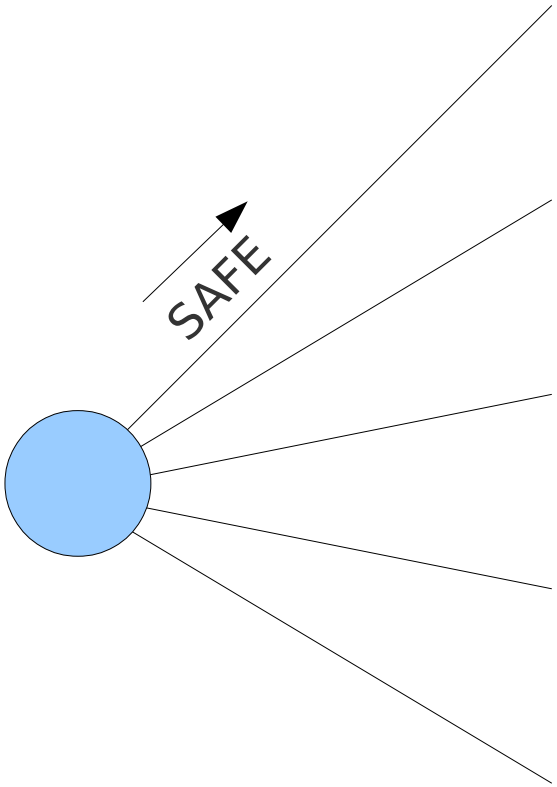
Synchronizatory

1. Czyli jak uruchomić alg „synch” w „asynch” sieci...
2. Po co to? Alg synch łatwiej się projektuje...
3. Wirtualne rundy, zegar generuje „pulsy”...

4. Potwierdzanie dostarczenia komunikatu (ACK)
   bo w modelu asynch nadawca nie wie kiedy kom został dostarczony!!
5. A–alg synch; **def:** wierz jest „bezpieczny” w k-tej rundzie A
   jeśli wszystkie kom k-tej rundy alg A zostały potwierdzone
6. Kiedy mogę przejść do k+1 (wirtualnej) rundy ?
   odp: gdy jestem bezpieczny i moi sąsiedzi są bezpieczni
   to jest minimalny warunek każdego synchron. !!!
7. Kilka typów synchron. α, β, γ, … (niżej oznaczane S)

8. Wada synchronizatorów: „narzut” czyli dodatkowe kom/czas;
   narzut komunikatowy M(S) i czasowy T(S) // na 1 rundę wirt.
   A – alg synch, A' – alg asynch (A'=A+S),
   M(A') = Minit(S) + M(A) + T(A)*M(S)
   T(A') = Tinit(S) + T(A)*T(S)
   ??? co to jest czas alg asynch ???

Synchronizator α

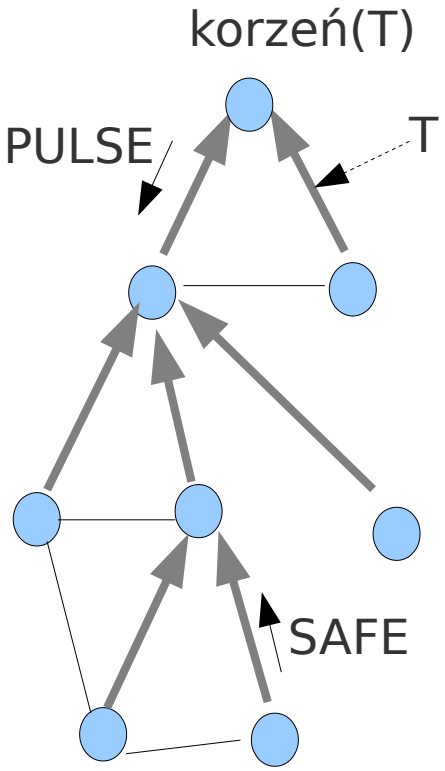


Zasada działania:
1. wierz wysyła kom k-tej rundy A, **z potw!**
2. wysyła SAFE do wszystkich sąsiadów
3. czeka na SAFE od wsz sąsiadów
4. przechodzi do (k+1) wirt rundy

Dlaczego to działa ?
* po pkt 1 wierz jest bezpieczny
  po pkt 3 wie, że jego sąsiedzi są bezpieczni
* zauważmy, że pkt 3 wstrzymuje wierz
  przed przejściem do następnej wirt rundy!
  (czyli to jest wirt odpowiednik fiber yield!)
* $T(\alpha)=O(1)$, $M(\alpha)=O(|E|)$

Pytanie:
mamy dwa wierz a i b w odległ x;
jaki jest nr bieżącej wirt rundy w a i b ?

Synchronizator β



drzewo spinające T
w sieci G

Zasada działania:

1. wierz wysyła kom k-tej rundy A, **z potw!**
2. liść T wysyła SAFE do parenta w T
3. nie-liść T (i nie-korzeń)
   czeka aż dostanie SAFE od wsz synów,
   a następnie wysyła SAFE do parenta
4. gdy korzeń(T) dostanie SAFE od wsz synów,
   to „rozsyła” po T kom PULSE...
5. gdy wierz dostanie PULSE to przechodzi
   do (k+1) wirt rundy

* zauważmy, że gdy wierz dostaje PULSE
  to wszystkie wierz G są bezpieczne
* możemy zadań to samo pytanie
  co poprzednio...
* $T(\beta) = O(diam(T))$, $M(\beta) = O(|V|)$

Synchronizator γ

- wada synch β duży narzut czasowy O(diam(T))
- synch γ używa „klastrów” (ale nie tych do MWIS w planarnych!!)
  w każdym klastrze C jest drzewko spinające T_C
- *idea:* w klastrach stosuje się β, w wierz/klastrach stosuje się α
- jeśli dwa klastry są połączone kraw, to jedna z nich jest **pref**
- *narzuty*: niech h_p oznacza max wysokość drzewka w klastrach,
  niech E_p oznacza zb kraw drzewek w klastrach + kraw pref
  dla param „k”, można zbudować klastry takie, że:

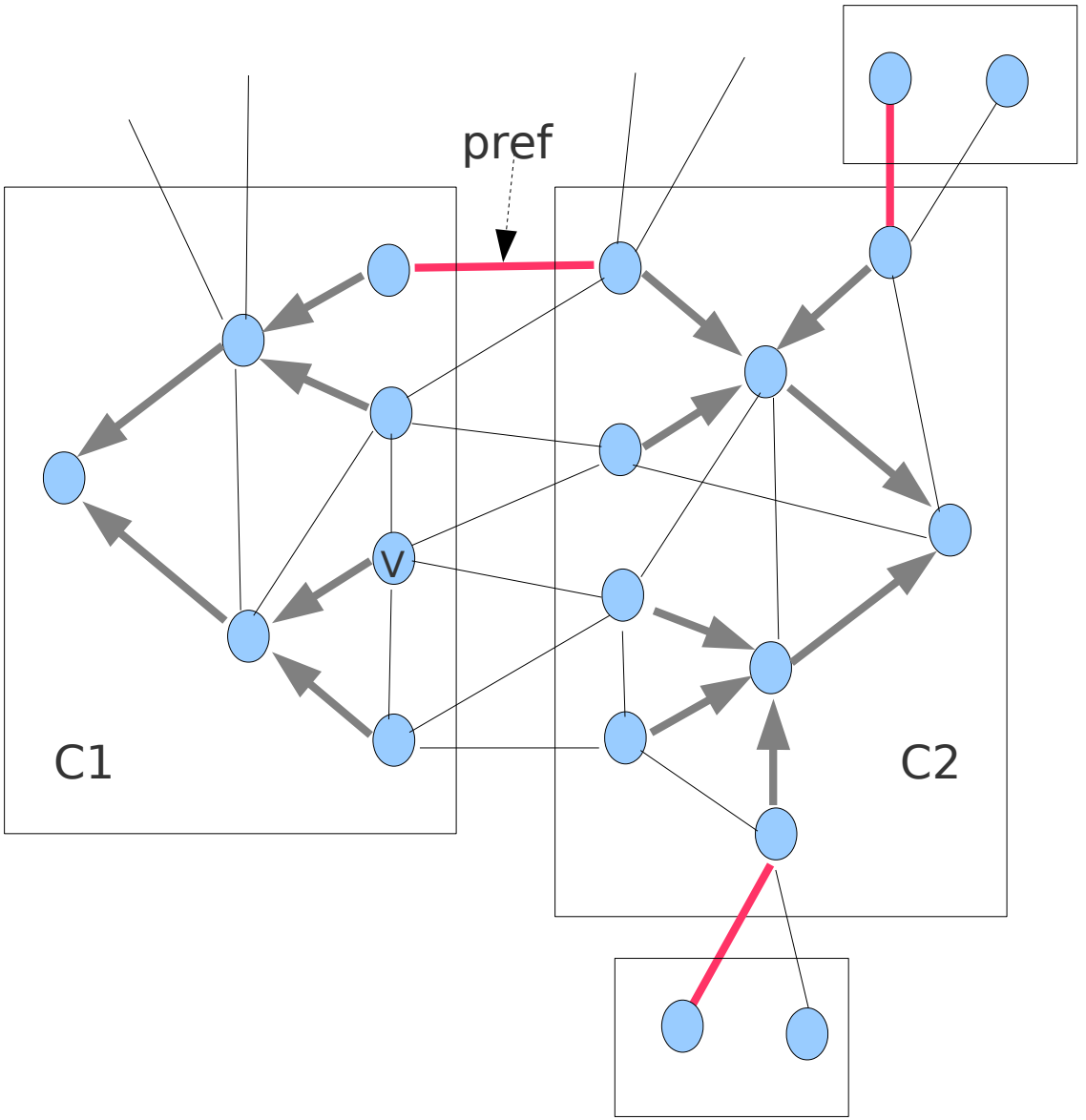
  $h_p < \log|V| / \log k$

  $|E_p| < k * |V|$

  $T(\gamma)=O(h_p)$, $M(\gamma)=O(|E_p|)$

  ??? skąd to wiemy, patrz zasada działania synch γ ???

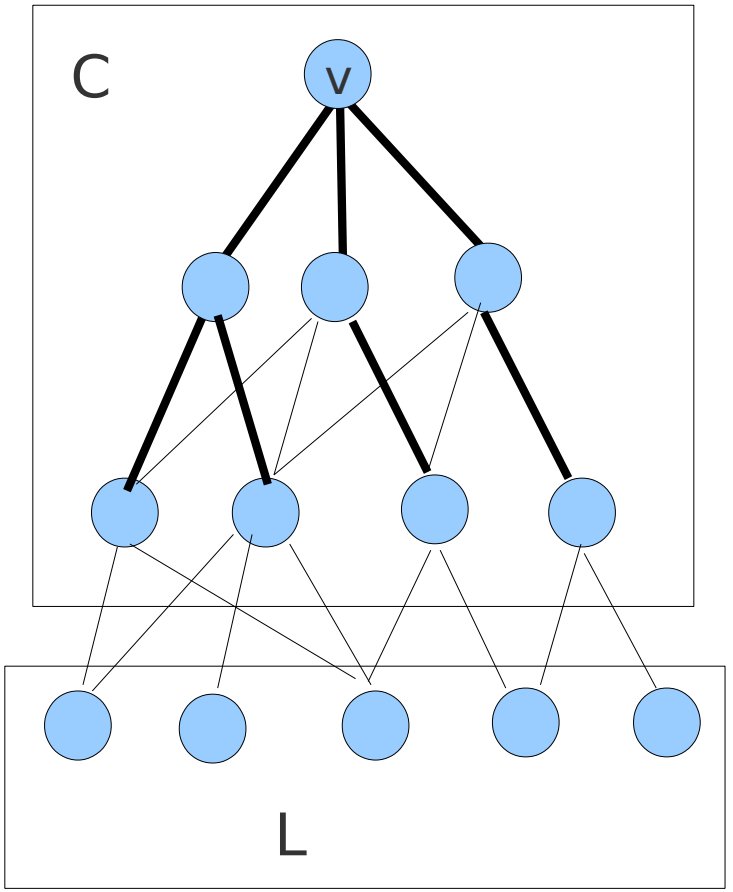
Synchronizator γ (c.d.)



Zasada działania:

- nie wystarczy używać β
  w klastrach!!!
  wierz „v” ma sąsiadów
  poza C1...
  (nie będą bezpieczni!)
- używamy kom:
  SAFE,
  CLUSTER_SAFE,
  ^ działają jak β w C1
  READY,
  ^ gdy CLUSTER_SAFE
  dotrze do kraw pref,
  to jej drugi koniec
  wysyła READY w C2
  PULSE,
  ^ gdy korzeń(T_C2)
  dostanie READY
  od wsz kraw pref,
  to rozsyła PULSE po T_C2

Jak się oblicza klastry dla synch γ ?



- budujemy po 1 klastrze
- zaczynamy od wierz „v” (LE?)
- jeśli |L| >= (k-1)* |C|
  to C:= C u L

Argument:
- jeśli promień klastra = x, to:
  n >=|C|>(k-1)^x; // MH: w przybliżeniu...
  zatem: n > (k-1)^x, x<logn / log(k-1)
  czyli: h_p < log |V| / log(k-1)
- E_p to kraw drzew + pref...
  gdy kończymy, to: |L|<(k-1)*|C|
  l. kraw pref C <= |L|<(k-1)*|C|
  l. kraw pref wsz klastrów <=(k-1)*|V|
  l. kraw drzew wsz klastrów < |V|
  czyli: |E_p| < k*|V|

Synch eta_1 i teta

Klastry typu „partition” vs typu „cover”...
w drugim typie każdy wierz należy do >=1 klastra

S i T to zb. klastrów typu cover... („małe” i „duże” klastry)
Def: T *poprawia* S jeśli dla każdego **a** z S istnieje **b** z T
t. że **a** zawiera się w **b**

Vol(T) to suma mocy klastrów z T, Vol(T)>=n
potrafimy obliczyć zb. klastrów T, poprawiający S, t. że:
1. Vol(T) < n^(1+1/z)
2. diam(T) < (z+1)*diam(S)

(robi się to alg b. podobnym do tego z poprz slajdu!!)

synch eta_1:
- S := {N[v]: v in V}, N[v] = „v + sąsiedzi v”
- dla z:=log|V|/logk: Vol(T)< k|V|, diam(T)< O(log|V|/logk)
- T poprawia S, zatem każdy wierz posiada klaster z T w którym są wszyscy jego sąsiedzi (*home cluster*)
- we wszystkich klastrach T działa synch beta
- gdy wierz dostanie PULSE z home cluster to przechodzi do następnej wirt rundy...

Dalszy rozwój: spannery Pelega + uogólniony synch α (!)

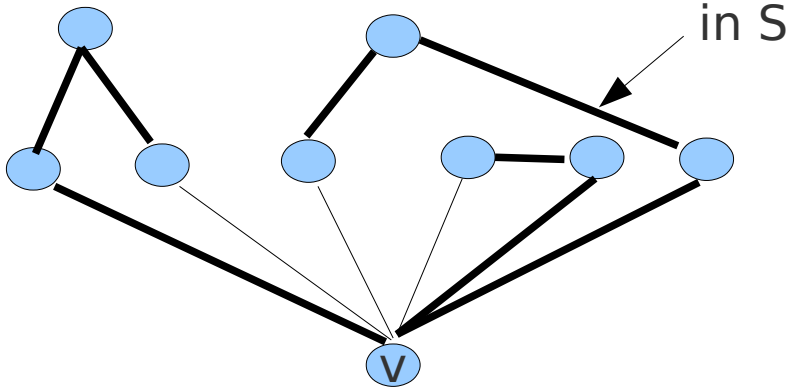
Synch alfa można uogólnić:
- następna runda gdy kula o prom „q” jest bezpieczna;
- T(alfa) = O(q), M(alfa)= O(q|E|)

Reszta układanki:
- *spanner* Pelega (multiplikatywny)
- jest to podgraf S grafu G, spinający i spójny, taki, że:
  - dist_S(a,b) <=q dist_G(a,b)

Praca o uogólnieniu synch alfa:
- Peleg, Ullman, „*An optimal synchronizer for the hypercube*”, 1987

Prace o obliczaniu spannera Pelega:
- Derbel, Gavoille, Peleg "*Deterministic Distributed Construction of Linear Stretch Spanners in Polylogarithmic Time*", 2007
- w tej^ pracy opisano jak zbudować:
  - 3-spanner, o O(n^1.5) kraw, w czasie O(logn), w modelu rozpr/synch

Jak to razem działa?



Kula(v,3) zawiera sąsiadów ”v” w G;
- Czyli jeśli ta kula jest bezp to sąsiedzi w G są bezp (min warunek synch !!)

T(alfa w span)=O(3)
M(alfa w span)=O(3 n^1.5)