Synchronizers

Thursday, July 24, 2014
8:13 AM

Have seen:

- asynch. model / synch. model

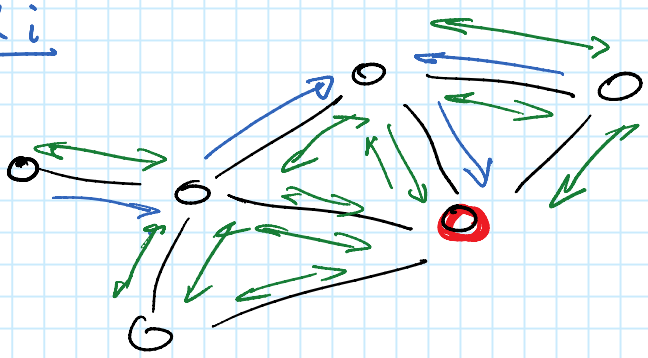
	synch. model	asynch. model
sync. alg.	✓	?
asynch. alg.	✓	✓

Can we run a synchronous algorithm in an asynchronous system?

Ideas:

- longest message time?
  Problem: don't know largest msg. delay
- "simulate" rounds
- once a node has received all msg. of the current round, it can proceed to the next round
- problem: don't which messages we should receive...
- everybody sends a message to all neighbors

Round i



Synchronizer = algorithm that simulates rounds

easiest solution

- every node sends a message to all neighbors in every round

⟹ works ✓

⟹ cost?

problem: many additional messages

synchronous alg. A

(synch.) time compl. of A : $T_S(A)$

" msg. " " " : $M_S(A)$

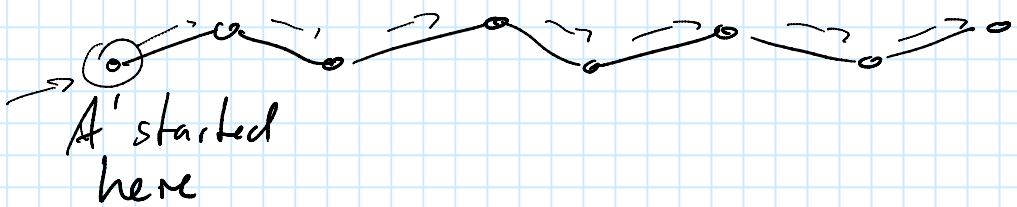
asynch. alg. A'

- (asynch.) msg. compl. of A': $\overbrace{\Theta(T_S(A) \cdot m)}^{M_A(A')}$

problem: potentially $M_A(A') \gg M_S(A)$

- (asynch.) time compl. of A':

$$T_A(A') = O(T_S(A))$$



example: BFS tree construction

synch. alg. A : flooding, $T_S(A) \leq D$, $M_S(A) \leq 2m$

asynch. alg. A' : $T_A(A') \leq D$, $M_A(A') = \Theta(D \cdot m)$

Can we improve the msg. compl.?

Synchronizers

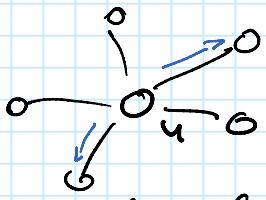
nodes generate clock pulses

Valid clockpulse (at node u)

- if node u has received all msg. from neighbors of prev. clock pulses

safe node (w.r.t. to clock pulse i)

- u is safe if all msg. sent by u in round (cl.pulse) i have been received by neighbors of u



How can we detect safety?

send acknowledgements!

Lemma: If all neighbors of a node u are safe, u can generate the next clock pulse.

Proof: ✓ (from the defs.)

Synchronizer S

(transforms synch. alg. A into asynch. alg. A')

$T(S)$: time per round of A
$M(S)$: #msg. per round of A
$T_{init}(S)$, $M_{init}(S)$: initialization cost (precomputation)

$$T_A(A') \leq T_{init}(S) + T_s(A)(1 + T(S))$$

$$M_A(A') \leq M_{init}(S) + M_s(A) + T_s(A) \cdot M(S)$$

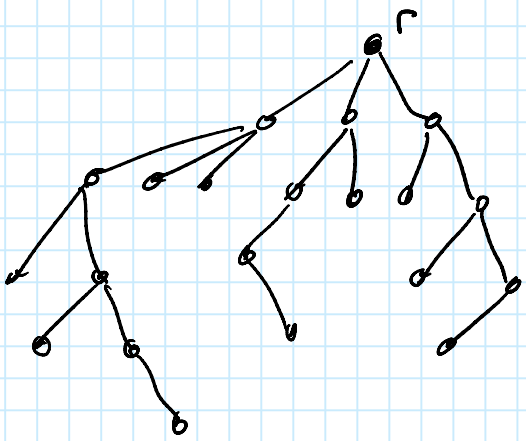
Synchronizer α

1. u waits until u is safe
2. u sends msg. to all neighbors (u is safe)
3. once all safety msg. received, proceed to next round (generate clock pulse)

$T(\alpha) = O(1)$, $M(\alpha) = O(m)$

Synchronizer β

1. precompute a rooted spanning tree T



idea: root finds out when all nodes are safe
→ use convergecast

node u:

1. waits until receives "safe" message from all children
2. if u is safe, report "safe" to parent

"safe" = subtree of u is safe

root uses flooding on T to tell everyone that all nodes are safe

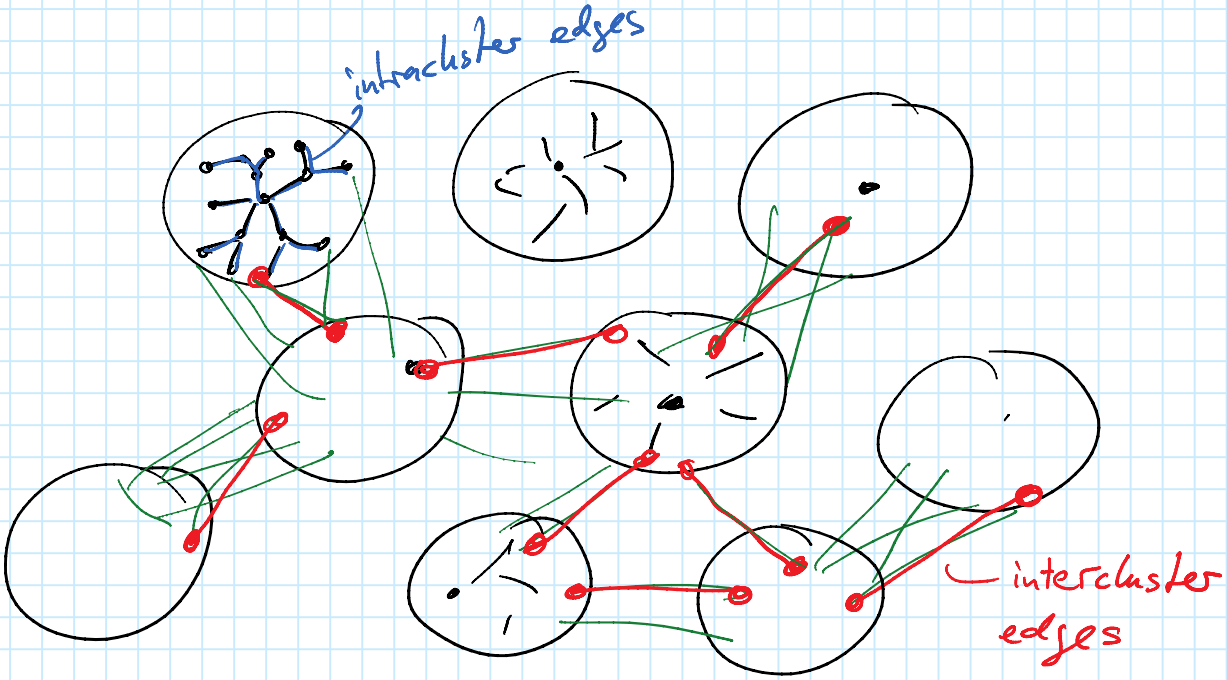
$T_{init}(\beta) = O(n)$, $M_{init}(\beta) = O(m + n \log n)$

$T(\beta) = O(n)$, $M(\beta) = O(n)$

Synchronizer γ

combination of α and β

precompute a clustering of G



Idea:

- use synchronizer β inside clusters
- use synchronizer α between clusters

1. synchronizer β inside clusters

for each cluster C, leader node $l(C)$

$l(C)$ uses convergecast to find out when all nodes of C are safe

$\Rightarrow$ can be done in time r_C

r_C: radius of the cluster tree of C

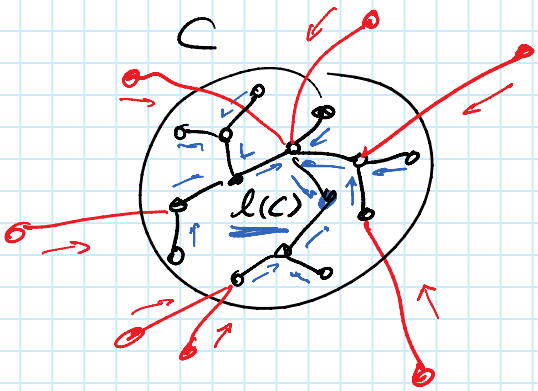
$l(C)$ reports to all nodes in C that the cluster C is safe

node u in cluster C

if u learns that C is safe

u sends "cluster safe" message over all its intercluster edges

cluster C needs to learn when it has received a "cluster safe" message over each of its intercluster (red) edges



once $l(C)$ knows that all neighboring clusters are safe, it can tell the nodes in C to start the next round (generate a clock pulse)

Cost of synchronizer γ

ignore initialization

($T_{init}(\gamma) = O(n)$, $M_{init}(\gamma) = O(m + n \log n)$)

parameters of clustering

k : max. cluster radius
(max. radius of the cluster trees)

m_C : # of intercluster (red) edges

$T(\gamma) = O(k)$

$O(1)$ many convergecasts / broadcasts in each tree

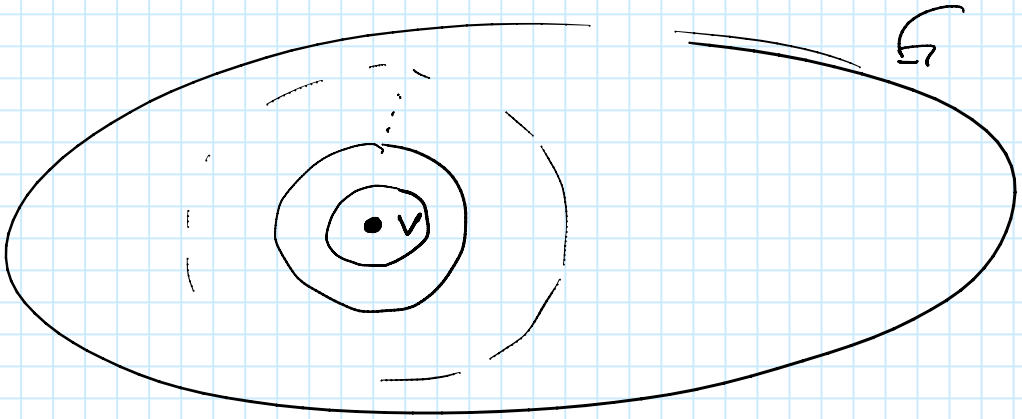
$M(\gamma) = O(n + m_C)$

$O(1)$ many convergecasts / broadcasts

≤ 2 msg. over red edges

What is the trade-off between k and m_C

Construct clustering (parameter $g > 1$)



$B(v,r)$: ball of radius r around v

$$B(v,r) := \{u \in V : d(u,v) \leq r\}$$

$r := 0$

while $|B(v,r+1)| > g\,|B(v,r)|$ do

  $r := r + 1$

output $B(v,r)$ as cluster

$\rightarrow$ remove $B(v,r)$ from G and repeat

cluster radius is r

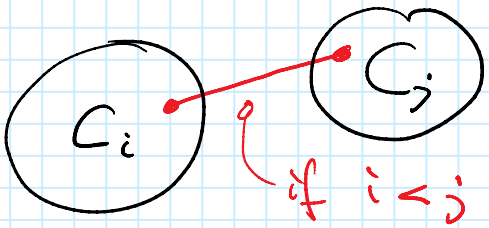
What is k?

$$k \leq \log_g n$$

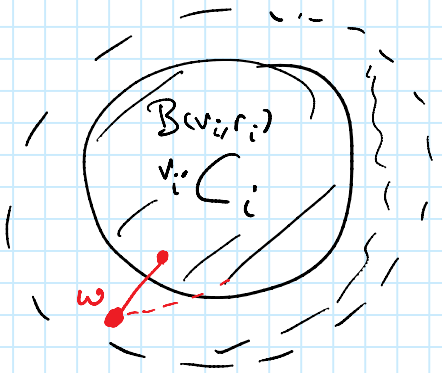
Number of intercluster edges?

order the clusters by the order in which they are constructed

$C_1, C_2, C_3, \ldots$



call this an intercluster edge of cluster C_i



$\Rightarrow$ # of intercluster edges of cluster C_i is at most

$$|B(v_i, r_i+1) \setminus B(v_i, r_i)|$$

w.r.t. graph when constructing cluster C_i

we know: $|B(v_i, r_i+1)| \leq g \cdot |B(v_i, r_i)|$

$$\Rightarrow |B(v_i, r_i+1) \setminus B(v_i, r_i)| \leq (g-1) \cdot |B(v_i, r_i)|$$

number of nodes in cluster C_i $\quad |C_i|$

$$w_c \leq \sum_i (g-1) \cdot |C_i|$$

$$= (g-1) \cdot \sum_i |C_i| = (g-1) \cdot n$$

In summary:

$k \leq \log_g n$, $m_c \leq (g-1) \cdot n$

$T(\gamma) = O(\log_g n)$, $M(\gamma) = O(g \cdot n)$

if $g = 2$: $T(\gamma) = O(\log n)$, $M(\gamma) = O(n)$

BFS-tree:

$T_A = O(n \cdot D \cdot \log n)$

$M_A = O(m + n(D + \log n))$

if $g = n^{1/s}$: $T(\gamma) = O(s)$, $M(\gamma) = O(n^{1+1/s})$

In α, β, γ:

all nodes simulate all rounds

$\Rightarrow$ trade-off of γ is optimal

If nodes only simulate rounds in which they participate

best known cost is $O(\log^3 n)$ for both time and msg. complexity