Network Synchronization

Janna Burman

Reference

“Complexity of Network Synchronization” 1985 - Baruch Awerbuch.

Synchronous network Model

- Messages are allowed to be sent only at integer times or ***pulses*** of a ***global clock***.
- Each node has an access to this clock.
- At most one message can be sent over a given link (from a given node) at a certain pulse.
- The delay of each link is at most one time unit of the global clock.

- Time complexity is the number of pulses passed from the start to the termination of the algorithm.

Compare to asynchronous network

- Each message sent by a node to its neighbor arrives within some finite but unpredictable time.
- Time complexity is the worst-case number of time units (time unit is a worst-case link delay) from the start to the termination of the algorithm.

Problem

Asynchronous algorithms (A_a) are in many cases substantially inferior in terms of their complexity to corresponding synchronous algorithms (A_s), and their design and analysis are much more complicated.

Goal

- To develop a general and efficient simulation technique – a ***synchronizer*** that enables any synchronous algorithm to run in any asynchronous network.

➢ Thus, to develop a methodology for designing efficient distributed algorithms in asynchronous networks.

Paper results

TABLE I. COMPLEXITIES OF SYNCHRONOUS ALGORITHMS

Problem	Adapted from PRAM algorithm of	Communication complexity	Time complexity
Breadth-first search	[4]	$\lvert E \rvert$	$\lvert V \rvert$
Maximum flow	[13]	$\lvert V \rvert^3$	$\lvert V \rvert^2$

TABLE II. COMPLEXITIES OF ASYNCHRONOUS ALGORITHMS

Problem	Reference	Communication complexity	Time complexity	Values of parameters
Breadth-first search	[6]	$\lvert V \rvert \, \lvert E \rvert$	$\lvert V \rvert$	
	[6]	$\lvert V \rvert^{2+x}$	$\lvert V \rvert^{2-2x}$	$0 \leq x \leq 0.25$
	This paper	$k \lvert V \rvert^2$	$\lvert V \rvert \dfrac{\log_2 \lvert V \rvert}{\log_2 k}$	$2 \leq k < \lvert V \rvert$
Maximum flow	[11]	$\lvert V \rvert \lvert E \rvert^2$	$\lvert V \rvert^2 \lvert E \rvert$	
	This paper	$k \lvert V \rvert^3$	$\lvert V \rvert^2 \dfrac{\log_2 \lvert V \rvert}{\log_2 k}$	$2 \leq k < \lvert \mathrm{V} \rvert$

Solution

Given: Asynchronous network and A_s

→ ***Synchronizer:***

- Is a distributed algorithm running in async. network.
- Is a ‘clock-pulse’ generator:
  - A new pulse is generated at a node only after it receives all the messages of the synchronous algorithm, sent to that node by its neighbors at the previous pulses.

- So, how node knows that it received all pulse messages of A_s?

Safe node

- ***Safe node*** - node is said to be safe with respect to a certain pulse if each message of the A_s sent by that node at that pulse has already arrived at its destination.
- If we require **ACKs** on each A_s message, each node may detect that it is safe whenever all its messages have been acknowledged.
- Then, a new pulse may be generated at a node whenever all the neighbors of that node are known to be safe with respect to the previous pulse.
  - It remains to find a way to deliver this information to each node with small communication and time costs.

Synchronizer *alpha*

- Each node detects eventually that it is safe and then reports this fact directly to all its neighbors.
- Whenever a node learns that all its neighbors are safe, a new pulse is generated.

- <u>Complexities</u> that are added to A_s at each pulse:
  - Communication: C(*alpha*) = O(|E|) = O($|V|^2$)
  - Time: T(*alpha*) = O(1)

Synchronizer *beta*

- <u>Initialization phase</u>:
  1. ***Leader* s** is chosen.
  2. Spanning tree rooted at s is constructed.
- Whenever a node learns that it is safe and all its descendants in the tree are safe, it reports this fact to its father (this communication pattern is referred to as a ***convergecast***).
- s will eventually learn that all the nodes in the network are safe; at that time it broadcasts a certain message along the tree, notifying all the nodes that they may generate a new pulse.

- <u>Complexities</u>: C(*beta*) = O(|V|) and T(*beta*) = O(|V|) (ignoring the initialization phase)

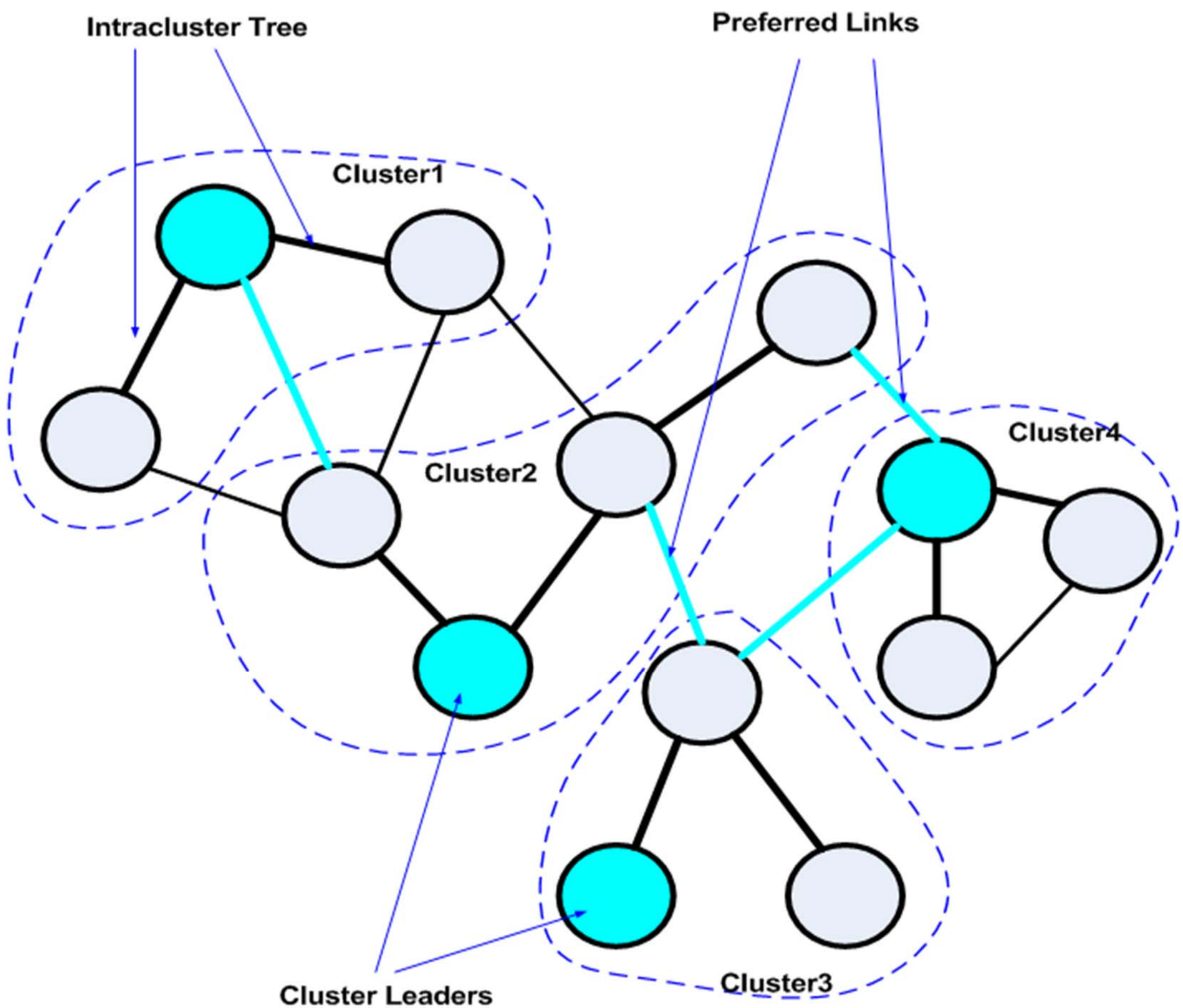
Exercises

1. Apply synchronizer *alpha* to the synchronous BFS construction by PI and calculate time and message complexities of the resulting asynchronous algorithm.

2. Repeat the same exercise with synchronizer *beta.*

Synchronizer *gamma*
(= *alpha* and *beta* hybrid)

- <u>Initialization phase</u>:
  - Network is partitioned into ***cluster***s.
  - This partition P is defined by any spanning forest of the communication graph.
  - Each forest tree is the cluster – ***intracluster tree***.
  - Between 2 neighboring clusters, <u>one</u> ***preferred link*** is chosen.
  - Inside each cluster, a ***leader*** is chosen.



Intracluster Tree
Preferred Links
Cluster1
Cluster2
Cluster4
Cluster3
Cluster Leaders

Synchronizer *gamma* (cont.)

- ***Cluster is safe*** – if all its nodes are known to be safe
- Synchronizer *gamma* is performed in 2 phases:
  – <u>1st phase:</u> Synchronizer *beta* is applied separately in each cluster along the intracluster trees
    - Whenever the leader of a cluster learns that its cluster is safe, it reports this fact to all the nodes in the cluster as well as to all the leaders of the neighboring clusters.

  – <u>2nd phase:</u> Synchronizer *alpha* is applied among clusters
    - Nodes of the cluster wait until all the neighboring clusters are known to be safe and then generate the next pulse.

Synchronizer *gamma* - details

1st phase:

- After terminating its part in the algorithm, a node enters the first phase of the synchronizer, in which SAFE messages are convergecast along each intracluster tree, as in Synchronizer *beta*.
- Eventually, leader learns that its cluster is safe and reports this fact to all neighboring clusters by starting the broadcast of a CLUSTER-SAFE message. Each node forwards this message to all its sons and along all incident preferred links.

2nd phase:

- Convergecast process: A node sends a READY message to its father whenever all the clusters neighboring it and any of its descendants are known to be safe. This situation is detected by a node whenever it receives READY messages from all its sons and CLUSTER-SAFE messages from all the incident preferred links and from its father.
- Whenever the above conditions are satisfied at the leader of the cluster, the leader knows that all the neighboring clusters as well as its own cluster are safe.
- Now, in order to start a new pulse, a cluster leader broadcasts along the tree a PULSE message.

Synchronizer *gamma* - Complexity

- ***Ep*** – set of all the tree links and all the preferred links in a partition P.
- ***Hp*** – maximum height of a tree in the forest of P.

- At most 4 messages are sent over each link of Ep. Thus **C(*gamma*) = O(|Ep|).**
- O(|Hp|) time requires for each cluster to verify its safety and additional O(|Hp|) to verify all neighboring clusters safety. Thus **T(*gamma*) = O(|Hp|).**

- Goal: To find P for which both Ep and Hp are small.
- Above parameters depend only on the structure of the forest.

Partition algorithm

- Using the partition algorithm of the paper:
  Ep ≤ k|V| and **Hp ≤ $\log_k$|V|** $(\log_2|V|/\log_2 k)$
  k is a parameter of the partition algorithm and may be chosen arbitrarily in the range $2 \le k < |V|$.
- By increasing k in the range from 2 to $|V|^{1/10}$, C(***gamma***) increases from O(|V|) to **$O(|V|^{1.1})$** while T(***gamma***) decreases from $O(\log_2|V|)$ to **O(10)**.

- <u>Complexity of the partition algorithm</u>:
  C_{init}(*gamma*) = $O(k|V|^2)$ and **T_{init}(*gamma*)** = $O(|V|\log_k|V|)$

Thank you!

Good Luck!