Synchronizers

Overview

- Synchronizers: general simulation techniques that allow to run synchronous algorithms in asynchronous networks

Definition 12.1 (valid clock pulse). *We call a clock pulse generated at a node* v valid *if it is generated after* v *received all the messages of the synchronous algorithm sent to* v *by its neighbors in the previous pulses.*

Synchronous vs asynchronous

- Given a synchronous algorithm A, A can be turned into an asynchronous algorithm as follows: as soon as v generates the i-th pulse, v performs the action of the i-th round

Theorem 12.2. *If all generated clock pulses are valid according to Definition 12.1, the above method provides an asynchronous algorithm that behaves exactly the same way as the given synchronous algorithm.*

Synchronous vs asynchronous/cont.

- How
  - Special (signalling) messages
  - Acks
- Complexity
  - T(S)/M(S) = time/message complexity of synchronizer S per pulse
  - $T_{tct} = T_{init}(S) + T(A)(1 + T(S))$
  - $M_{tct} = M_{init}(S) + M(A) + T(A)M(S)$

Synchronizer α

Definition 12.3 (Safe Node). *A node v is* safe *with respect to a certain clock pulse if all messages of the synchronous algorithm sent by v in that pulse have already arrived at their destinations.*

Algorithm 42 Synchronizer α (at node v)

1: **wait** until v is safe
2: **send** SAFE to all neighbors
3: **wait** until v receives SAFE messages from all neighbors
4: start new pulse

Synchronizer α/cont.

- In practice:

1. Send message to all neighbors, include round information i and actual data of round i (if any)
2. Wait for message of round i from all neighbors, and go to next round

Theorem 12.5. *The time and message complexities of synchronizer α per synchronous round are*

$$T(\alpha) = O(1) \quad \textit{and} \quad M(\alpha) = O(m).$$

Synchronizer β

- <u>Initialization</u>: compute a spanning tree rooted at some leader l

Algorithm 43 Synchronizer β (at node v)

```
wait until v is safe
wait until v receives SAFE messages from all its children in T
if v ≠ ℓ then
    send SAFE message to parent in T
    wait until PULSE message received from parent in T
end if
send PULSE message to children in T
start new pulse
```

Synchronizer β/cont.

Theorem 12.6. *The time and message complexities of synchronizer β per synchronous round are*

$$T(\beta) = O(\text{diameter}(T)) \le O(n) \quad \textit{and} \quad M(\beta) = O(n).$$

The time and message complexities for the initialization are

$$T_{\text{init}}(\beta) = O(n) \quad \textit{and} \quad M_{\text{init}}(\beta) = O(m + n \log n).$$

- Synchronizer α is time efficient
- Synchronizer β is message efficient
- Can we trade-off? Yes

Synchronizer γ/cont.

Algorithm 44 Synchronizer γ (at node v)

wait until v is safe
wait until v receives SAFE messages from all children in intracluster tree
if v is not cluster leader **then**
    send SAFE message to parent in intracluster tree
    wait until CLUSTERSAFE message received from parent
end if
send CLUSTERSAFE message to all children in intracluster tree
send NEIGHBORSAFE message over all intercluster edges of v
wait until v receives NEIGHBORSAFE messages from all adjacent intercluster edges and all children in intracluster tree
if v is not cluster leader **then**
    send NEIGHBORSAFE message to parent in intracluster tree
    wait until PULSE message received from parent
end if
send PULSE message to children in intracluster tree
start new pulse

Synchronizer γ/cont.

Theorem 12.7. *Let* m_C *be the number of intercluster edges and let* k *be the maximum cluster radius (i.e., the maximum distance of a leaf to its cluster leader). The time and message complexities of synchronizer* γ *are*

$$T(\gamma) = O(k) \quad \text{and} \quad M(\gamma) = O(n + m_C).$$

Building a partition

Algorithm 45 Cluster construction

while unprocessed nodes **do**
  select an arbitrary unprocessed node v;
  $r := 0$;
  while $|B(v, r+1)| > \rho |B(v, r)|$ **do**
    $r := r + 1$
  end while
  makeCluster($B(v, r)$)   // all nodes in $B(v, r)$ are now processed
end while

- B(v, r): ball of radius r aroung v
- This is a *centralized algorithm*

Building a partition/cont.

Theorem 12.8. *Algorithm 45 computes a partition of the network graph into clusters of radius at most $\log_\rho n$. The number of intercluster edges is at most $(\rho - 1) \cdot n$.*

- The trade-off between intracluster radius and number of intercluster edges is asymptotically optimal
- If ρ >= 2 it is possible to give a distributed algorithm with time and msg complexities O(n) and O(m + nlog n) respectively