CS321 Project extension: The TANGO Desynchronisation algorithm

Eric Choi, John Jersin, Harry Robertson
(*<iechoi*, *jjersin*, *harryrob>* *@stanford.edu*)

1 Principles of TANGO

1.1 Randomization

The Desync algorithm does not extend naturally to multi-hop networks. To overcome this we have modified the algorithm to randomize time slot selection. Instead of deterministically drifting towards the center of the time interval between its predecessor and successor as in Desync, a node now simply chooses a random new firing time if it is firing too close to a 1-hop or 2-hop neighbor.

1.2 Toe stepping

A node can detect if it is firing too close to a neighbor, as it can hear that neighbor's messages. However, in the network configuration A-B-C (where B can hear A and C, but A and C cannot hear each other), if A and C are "toe-stepping" (emitting too close to each other) then they cannot detect this directly.

To avoid this problem, if a node B hears A and C toe-stepping, it sends a toe-step packet to one of them to tell it to re-allocate. If a node receives a toe-step packet, it randomly chooses a new firing time.

1.3 "Too close"

If node A observes nodes B and C emit at t_B and t_C, it decides they are too close iff:

$$(|t_C - t_B| \,\% T) \leq \frac{\epsilon T}{n+1}$$

(where T is the period, n is the number of A's neighbors, and ϵ is a constant).

This definition intrinsically gives us a lower bound on the time slot spacing metric for the network. Our performance on the metric can be adjusted by modifying ϵ.

1.4 Enhancements

When a node has to choose a new firing time, the most basic approach would be to choose it perfectly randomly. However, if the new firing time is too close to another node's firing time, one or the other will have to re-allocate, and so on. This leads to many useless re-allocations, and poor convergence time. To improve this situation, each node records its neighbor's firing times, and only chooses a new firing time which is not too close to those.

This does not take into account conflicts with 2-hop neighbors. To avoid these, we include neighbor information in the toe-step packets: when A sends B a toe-step packet, it includes it's neighbor's firing times. This way B can avoid emitting too close to those (2-hop) neighbors.

2 Algorithm statement

Each node N basically operates in the following manner:

- store the period, our next firing time and our neighbor's IDs and most recent firing times
- pick a random intial firing time
- if two neighbors N_1 and N_2 emit too close to one another: send a toe-step packet to N_1 (where $ID(N_1) < ID(N_2)$)
- if N' emits too close to us:

  - if $ID(N') > ID(N)$: randomly pick a new firing time which is not too close to our neighbors'
  - if $ID(N') < ID(N)$: continue as is

- if we receive a toe-step packet:
  - if we have recently changed firing time: continue and ignore duplicate toe-step packets
  - otherwise: randomly pick a new firing time which is not too close to our neighbors' OR to the nodes contained in the toe-step packet

3 Performance

3.1 Time slot spacing

Once the algorithm has converged (i.e. no node see any two nodes as "too close"), we have by definition of the property "too close":

$\forall$ node A, $\forall$ A's neighbors B,C

$$(|t_C - t_B| \% T) > \frac{\epsilon T}{n+1}$$

Using the metric presentation paper's notations, we have for the node N_j and its neighbors:

$$\forall i \leq n_j, (t_{(i+1)\% n_j} - t_i)\% T > \frac{\epsilon T}{n_j}$$

These differences naturally sum to T, so we also have:

$$\forall i \leq n_j, (t_{(i+1)\% n_j} - t_i)\% T \leq T - \epsilon T \frac{n_j - 1}{n_j}$$

So $\forall i \leq n_j$,

$$\left| t_{(i+1)\% n_j} - t_i)\% T - \frac{T}{n_j} \right|$$

$$\leq max(\frac{\epsilon T - T}{n_j}, T - \epsilon T \frac{n_j - 1}{n_j} - \frac{T}{n_j})$$

$$\leq T - T\frac{\epsilon(n_j - 1) + 1}{n_j}$$

In particular if $\epsilon \approx 1$ (and $\epsilon < 1$), then

$$\forall i \leq n_j, \left| t_{(i+1)\% n_j} - t_i)\% T - \frac{T}{n_j} \right| \approx 0$$

Thus by making ϵ approach 1, we can obtain arbitrarily low values of the time slot spacing error described in the metric paper.

3.2 Convergence time

Without information in toe-step packets about timing of 2-hop neighbors, our convergence time is unbounded in any case with nodes that have at least one 2-hop neighbor. This is due to the fact that a node can randomly but repeatedly choose a time that will collide with the 2-hop neighbor. In this version of the algorithm the probability of converging quickly decreases with the number of 2-hop neighbors. The worst case is a star network (a tree with many leaves, all at depth 1).

Adding information about 2-hop neighbors into toe-step packets provides an upper bound on the number of periods it takes to converge for star networks which is linear in the number of nodes in the neighborhood of the root (all nodes in the star). This is a huge improvement, from a possibly infinite convergence to a linear convergence time for what was the worst case.

A proof by induction of linear convergence time for star networks follows.

Definitions We define *settled* to mean a node will not change its timing again until the network has converged.

We define *converged* to mean a set of nodes which are settled and such that no other node fires too close to any of them.

We order the nodes according to decreasing ID numbers.

All collisions between pairs of nodes will be resolved by re-randomizing the firing time of the node with the lower ID.

Base case The first node will not need to change its timing (as it has the highest ID) and we consider it settled.

- If the root node is the first node, then all nodes will be able to detect if they are too close to it and rerandomize so that they are not longer too close.

- If the first node is a child of the root node and the root node is too close to the first node, the root will detect this and re-randomize its own timing so that it no longer collides with the first node.

- If the first node is a child of the root node, the root will detect collisions between the first node and any of its children. Toe-step packets will be sent to all nodes that collides with the first node and they will re-randomize such that they no longer fire too close to the first node.

Therefore after period 1, the first 1 node will have converged. Our inductive step is to show that after period N the first N nodes will have converged.

Inductive step We assume that for period N the first N-1 nodes have converged.

The Nth node will not be firing too close the any nodes before it in the ordering by definition of convergence.

Therefore all nodes which may be firing too close to the Nth node will have lower ID than it, and the Nth node will be settled.

All collisions with the Nth node will be settled as they were for the first node above. When nodes re-randomize, they avoid collisions with any neighbors or 2-hop neighbors that they know about, therefore they will not collide with any of the first N-1 nodes in the process of moving away from the Nth node.

Therefore after the Nth period, the first N nodes will have converged.

Therefore convergence time is linear in the number of nodes for the star network.

4 Conclusion and Future work

Our current implementation of the algorithm shows fast convergence time and reasonable (though not optimal) time slot spacing performance.

Ideas for future work include:

- Currently if two nodes emit so close to one another that their packets collide, and there are no third parties observing this, then the two nodes will not notice each other[1]. To avoid this we could add a slight random "wobble" to each firing time, so that firings can never systematically collide.

- We have a bounded convergence time only when each node has at most one neighbor which could send it a toe-step packet. By remembering (and aging) the information contained in toe-step packets from all neighbors, we should be able to bound convergence time in all cases.

- Make the algorithm independent of node IDs (i.e. use a different way of deciding which of two toe-stepping nodes should re-allocate).

[1] Actually the nodes will eventually fall apart due to clock drift, so this is not a pathological case in practise.