Exercise 12: Pulse!

Task 1: Again, and Again, and Again...

In this task, the goal is to show that once a single synchronized pulse is generated by the main state machine from the lecture, the algorithm stabilized. This follows by induction, once we prove that another synchronized pulse will be generated (within the period bounds) without any correct node producing any pulse in between. Of course, this requires some bounds on the timeouts. Throughout this exercise, we assume that each $v \in V_g$ transitions to PULSE during $[0, 2d]$.

a) Which inequality does T_1 need to satisfy so that it is guaranteed that each $v \in V_g$ transitions to WAIT during $(T_1/\vartheta, T_1 + 2d)$?

b) If the constraint from a) holds, which inequalities do T_{listen} and T_{wait} need to satisfy so that it is guaranteed that each $v \in V_g$ transitions to INPUT 1 during $(T_1/\vartheta, T_1+3d)$?

c) If the above two constraints are satisfied, which inequality does T_{wait} need to satisfy so that it is guaranteed that each $v \in V_g$ transitions to RUN 1 during $((T_1+T_2)/\vartheta, T_1+T_2+3d)$?

d) If the resulting consensus instance is correctly working, argue that another synchronized pulse satisfying the period bounds given in the lecture is generated.

e) Provide (minimal) constraints under which the resulting consensus instance is guaranteed to be executed correctly.

Task 2: NEXT!

In this task, we modify the self-stabilizing pulse synchronization algorithm from the lecture such that it provides the interface required to make the Lynch-Welch algorithm self-stabilizing using the technique from the lecture.

a) Add an intermediate state to the auxilliary state machine that "delays" an output of 1 by the consensus routine. The transitions to PULSE and to LISTEN are triggered when one of the following three events occurs: (i) the NEXT signal is triggered, (ii) Guard G4 is satisfied, or (iii) a timeout of $\mathcal{O}((1 + (\vartheta - 1)R)d)$ expires. Explain why the modified algorithm stabilizes regardless of the NEXT signals, provided suitable timeout assignments can be found. How are the constraints on the timeouts affected (no details necessary)?

b) Argue that the self-stabilizing pulse synchronization algorithm can be made to work in its modified form, i.e., if the original algorithm had a suitable assignment of timeouts and $\vartheta - 1$ is sufficiently small, so does the modified algorithm. (Handwaving is ok, no formal proof required.)

c) In the terminology of Section 9.1, determine σ_h, B_1, B_2, and B_3 for the modified algorithm.

d) Arguing as in Task 1 of exercise sheet 9, show that one can choose timeouts so that the machinery from Section 9.1 works and $T, \mathcal{S}(1) \in \mathcal{O}(\sigma_h)$. Here, you will have to choose T larger than in the lecture. You may assume that $\vartheta - 1$ is a sufficiently small constant. What is the resulting skew bound after stabilization? (Hint: T will only increase by a constant factor, which will not cause any real trouble; deal with this in the very end. First, recall that it's not necessary to choose $B_3 = \alpha B_2 = \alpha^2 B_1$. It's good enough to ensure that B_1 beats an $\mathcal{O}(\sigma_h + d)$ term, which T_2 in the modified algorithm will have to do anyway, and choose $B_3 = \alpha B_2$ for some $\alpha \in 1 + \mathcal{O}(\vartheta - 1)$.)

e*) The result is not actually a good solution to the original problem, as the Lynch-Welch algorithm now ends up having a bad skew of $\Omega(d)$. However, this issue can be resolved by further modifying how the pulses are generated. Can you see the solution?