Large-Scale Distributed Systems
Project 3: Firefly-inspired synchronization

Laurent Hayez

December 9, 2015

Table of contents

1 Introduction

In nature, fireflies produce light in order to attract mates or prey. One interesting feature of these beetles is that when they emit light in group, at some point, they do it in a synchronized manner, just by looking at when their neighbours emit light. This feature is interesting in large-scale distributed systems, as synchronization might be required, but one node does not know every other nodes.

The objective is thus to inspire ourselves from fireflies to try to synchronize nodes in a decentralized manner. At first we will detail the protocol skeleton and explain how the core of the protocols will work. Then we will look at a two models called "phase-advance" and "phase-delay" and briefly analyze them. The main and final part will be the "adaptive Ermentrout model" which is more representative of the reality. We will explain the implementation specifities and analyze this model in different situations.

2 Protocol skeleton

According to the paper "Firefly-inspired Heartbeat Synchronization in Overlay Networks", the skeleton for the different algorithms is composed of two main functions, namely activeThread and passiveThread. We provide the pseudo code for the implementation in Algorithm 2.1.

In the different protocols, a node is an oscillator characterized by its phase φ and the cycle length Δ. We define φ as a sawtooth function with domain $[0, 1]$ such that $\frac{d\varphi}{dt} = \frac{1}{\Delta}$. This is represented in Figure 1.

When φ reaches 1, the node will send a flash to a set of neighbour nodes, and φ is reset to 0. The cycle length, depending on the model chosen, can be the same or different for all nodes.

Algorithm 2.1 Skeleton for the Firefly algorithms

Variables:

φ ▷ phase

Δ ▷ cycle length

update_phi_init $\leftarrow$ false

update_phi_period $= \begin{cases} \frac{\Delta}{10} & \text{if } \Delta < 1 \\ \frac{1}{10\Delta} & \text{if } \Delta \geq 1 \end{cases}$

function sendFlash()
    $P \leftarrow$ view from PSS
    send flash to all peers in P
end function

function processFlash()
    depends on the implementation
end function

function updatePhi()
    if $\varphi < 1$ **then**
        $\varphi \leftarrow \varphi + \frac{1}{\Delta} \cdot$update_phi_period
    else
        fire event "Flash!"
        $\varphi \leftarrow 0$
    end if
end function

function activeThread()
    if $\neg$ update_phi_init **then**
        update_phi_init $\leftarrow$ true
        new periodic thread "updatePhi" with period update_phi_period
    end if
    wait for the event "Flash!"
    sendFlash()
end function

function passiveThread()
    receive flash
    processFlash()
end function

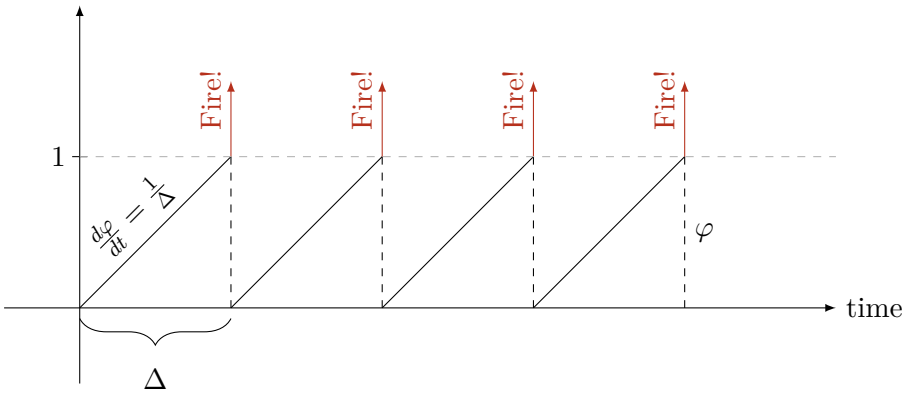

Figure 1: Representation of φ and its relation with Δ

3 Conclusion