An implementation of firefly-inspired network synchronicity without leaders on a group of small wireless devices

Fujio Yamamoto

Kanagawa Institute of Technology, Kanagawa 243-0292, Japan
(Tel: 81-46-291-3181, Fax: 81-46-242-8490)

yamamoto@ic.kanagawa-it.ac.jp

Abstract: Synchronicity observed in a group of fireflies was reconstructed using SunSPOTs. It is based on the software simulation with phase delay model. The main purpose is to provide a method for performing synchronicity among electronic fireflies, rather than obtaining clock synchronization. Some problems arising from the use of real wireless devices, not occurred in the simulation, were addressed and solved. The most important problem was how to send and receive phase delay information among SunSPOTs using real ratio signal. This is not the problem in the simulation because the detection of other flashes can be immediately obtained by simple calculations. The results from the proposed implementation exhibit relatively good synchronicity although its accuracy is not so high. These can be useful to observe synchronous behavior in the biological population without leaders.

Keywords: network synchronicity, sensor network, synchronization

1 INTRODUCTION

Many applications using wireless sensor network technology require time synchronization among each local clock on the devices connected to the network. But it is hard to get an efficient method for it. On the other hand, *synchronicity* is not the same as *time synchronization,* and the both are complementary each other. Synchronicity is considered as the ability to organize simultaneous collective action across a network (Werner-Allen et al [1]). Many research papers on synchronicity inspired by fireflies have been published (Werner-Allen [1], Tyrrell [2], Leidenfrost [3]). However, most of them except Werner-Allen [1] take software simulation, where uncertain behavior and delay in communication are not considered.

In this paper, an implementation method for synchronicity that does not use a centralized signal is presented, and demonstrated on a realistic experimental radio network. This method is based on the synchronicity that has been observed in nature in large populations of fireflies. My paper does not ask for very accurate synchronous behavior because the objective is to demonstrate fireflies' synchronicity on electronic devices. The accuracy of about several milliseconds is good enough to observe flashing of fireflies by human eyes. I proposed a method for synchronicity, and implemented it on the devices as shown below. In the following, at first, flashing behavior of fireflies without leaders is analyzed by a simulation using NetLogo (Wilensky [4], Wilensky [5]). Next, my implementation on a group of SunSPOTs (Smith [6], Simon [7]) is presented. Finally some experimental results are presented.

2 ANALYSIS OF FIREFLY SYNCHRONICITY USING NETLOGO

In the NetLogo's model library, there is an application to simulate behavior of populations of fireflies. It exhibits simultaneous flashing of all the fireflies without leader. I made concise version of it to investigate the mechanism of such synchronous actions. The interface window is illustrated in Fig. 2.1. Some related parameters are set as follows:

[number of fireflies]= 4
[cycle-length] = 6 time ticks
[flash-length] = 2 time ticks
[flashed-to-rest] = 1

Four fireflies are deployed horizontally to one line. They are identified as Firefly-0 (#0), .., Firefly-3 (#3). The distance between adjacent fireflies is set to one (1). This numbering of fireflies is made in round-robin manner, so that the distance between #0 and #3 becomes one. Each firefly tries to adjust its flashing phase to the neighboring one that other emission. It is assumed here that one can receive the emission within the distance 1. The firefly that receives such emission reset its time tick to the value of the flash-length. This resetting occurs only after the flashing and never occurs during its own flashing. Therefore, this makes the receiving firefly delay the phase after its flashing.

Detailed procedures in NetLogo are shown in Fig. 2.2.

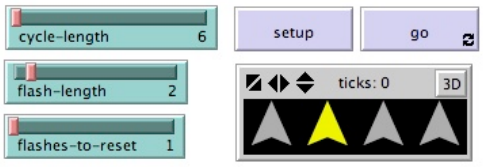
cycle-length 6
setup
go
flash-length 2
flashes-to-reset 1
ticks: 0
3D

Fig. 2.1 Interfaces in the simulation

```
turtles-own [ clock ] ;; each firefly's clock
to setup
  clear-all
  put-firefly 0 0 3
  put-firefly 1 0 0
  put-firefly 2 0 4
  put-firefly 3 0 5
  ask turtles [recolor]
end
to put-firefly [ix iy iclock] ;; create ordered turtle
  cro 1 [setxy ix iy set clock iclock]
end
to go
  ask turtles [
    increment-clock-upto-cycle-length
    reset-clock-upon-perceiving-other-flash
  ]
  ask turtles [recolor]
  plot-number-of-flashes
  plot-firefly-phase
  tick ;;advance the tick counter by one
end
to increment-clock-upto-cycle-length
  set clock (clock + 1)
  if clock = cycle-length [ set clock 0 ]
end
to reset-clock-upon-perceiving-other-flash
  if  (clock >= flash-length)[ ;; own flash never changed
    if (count turtles in-radius 1 with [color = yellow]
       >= flashes-to-reset) [ set clock flash-length ]
  ]
end
to recolor
  ifelse (clock < flash-length)
    [ set color yellow ] [ set color gray ]
end
```

Fig. 2.2 Procedures in the NetLogo simulation

The results obtained by executing these procedures are illustrated in Fig. 2.3. At the starting point (time tick = 0), four fireflies are in different phase. Only #1 is in flashing interval. The #0 immediately detects the flash from #1 and resets its phase to the value of the flash-length. The same action occurs in the #2. When time tick advances to one, the #1 is still in flashing mode. Therefore, phase delay occurs again in both #0 and #2. When time tick advances to two, the #1 completes its flashing, but the #3 is still in flashing mode. In this situation, both #0 and #2 detects flashing from the #3. This causes the #0 and #2 again reset their phase to the value of flash-length. After that, others affect no fireflies for a while.

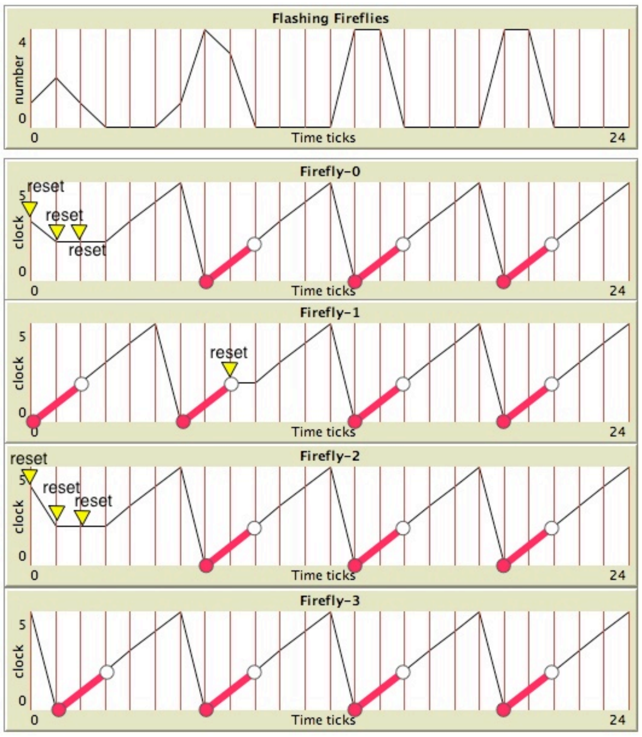

Fig. 2.3 The analysis of the phase delay model

The next notable point is the time when time tick becomes eight. At this point, the #1 detects flashing from either #0 or #2. Therefore, when time tick advances to nine, the #1 resets its phase to the value of flash-length. After the time tick becomes thirteen, all the fireflies exhibit synchronous behavior, and this synchronicity will be continued forever.

3 IMPLEMENTATION OF SYNCHRONICITY ON SUNSPOT

Flashing of fireflies was implemented on a group of small devices with wireless communications. The proposed method for it is based on the NetLogo simulation mentioned above. That is, the phase delay mechanism affecting to the flash cycle, is adopted. However, there are some important issues, which were not appeared in the simulation. They are discussed below.

3.1 Issues in implementation on SunSPOT

The present implementation of the proposed method has been performed on a set of tiny Java machine SunSPOTs Smith [10], Simon [11], Akriboulos [12]. One SunSPOT corresponds to one firefly as illustrated in Fig. 3.1. Each SunSPOT has ZigBee-based radio communication facilities, which exchange clock delay information.

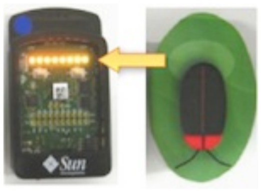
Fig. 3.1 Electronic firefly using SunSPOT

In NetLog simulation the problem of how to detect the flashes of other fireflies does not occur because other nearby flashes are obtained immediately by a simple calculation based on positional information. On the other hand, with SunSPOT, the flash is expressed and emitted by a radio signal, so that other SunSPOTs should be able to receive the flash. Since flashing occurs erratically throughout the population, each SunSPOT generates multiple threads on which other flashes starting at different times can be received.

Another problem is that no global clock exits in the implementation on SunSPOTs differently in the case of NetLogo simulation. In the simulation explained above, all the fireflies refer to one global clock, even though they are in different phases. Namely, as shown in Fig.2.3, all fireflies act based on the common time ticks. Therefore, delay value to synchronize with others always becomes an integer multiples of one time tick. This is not true on SunSPOTs because fireflies must calculate the difference between the starting time of its own flash and the receiving time of other flash by their own local clock. This difference does not necessarily become an integer multiples of one time tick. This implies that synchronicity and time-synchronization are not the same, as shown in the introduction.

3.2 Principles of the implementation

Each firefly on a SunSPOT counts up time ticks in its own clock during the cycle-length. When it reaches the max (i.e. cycle-length), the count is reset to zero. When a firefly becomes time tick zero, it begins flashing and continues it during the flash-length, in which other flashes do not affect the firefly although it can receive other flash. Flashing is realized by broadcasting a packet that has special header. Once the firefly detects other flashes, then it does not receive other flashes for a time. When the firefly receives other flashes, the firefly can calculate the necessary delay time. The delay is the difference between the receiving time of other flash and the starting time of its own flash. The delay is added at the end of the current flash cycle. It is expected that the firefly synchronizes its own flashing at the next cycle with that of the firefly sending the packet.

However, in the real world, several problems can be observed. For example, some pairs of SunSPOTs may maintain fixed value of delay each other for a long time, depending on the startup timing, as shown in the next section. Also, cycle time of each SunSPOT may gradually become disordered. Due to this, synchronicity that is observed may lose accuracy even though essential framework of synchronicity is preserved. In order to settle these problems, it is necessary for a SunSPOT that has been received other flash can get again another flashing signal after several cycles. Such correction seems unavoidable when using many devices with limited timing accuracy.

4 RESULTS AND ANALYSIS

Experiment of this implementation using four to eight SunSPOTs has been performed. The radio output power was set to the minimum so that broadcast range is limited within 20 cm. All the SunSPOTs are aligned into a line. The distance between two SunSPOTs is set to 15 cm. Therefore, each SunSPOT can receive broadcast message only from the neighbors. In this situation, each SunSPOT is started in different time, and human eyes observed the process to the synchronicity. For the group of four SunSPOTs, synchronicity was obtained after two to four cycles. For the group of eight SunSPOTs, it took more cycles but less than six cycles until steady state of synchronicity. Occasionally, disorder of synchronicity was observed. However, the synchronicity was recovered approximately after four to six cycles. Typical two cases observed in the experiment are analyzed in Fig 3.2 and Fig 3.3.

Fig.3.2 shows the group of four SunSPOTs, in which each SunSPOT begins its flashing at random time point. The #1 detects flashing of #2 during its own flashing interval. The delay d1 (difference between the start time of #1 and the start time of #2) was added at the end of the current cycle of #1. This causes synchronicity between #1 and #2 at the next cycle. On the other hand, the #4 detects the second flashing of #3 outside of its flashing interval. The delay d3 (the delay against #3) was added at the end of normal cycle of #4. Any other SunSPOT never disturbed the #2. After two cycles of #1, all four SunSPOTs agreed to synchronize.

Fig.3.3 illustrates different behavior compared to Fig3.2. Note that the cycle-length is changed against the case of Fig. 3.2. to make it easy to explain. One time tick regularly shifts the starting time of three SunSPOTs. The #2 was delayed for d2 (against #1) because #2 detects flashing from #1. By this, the #2 agreed to synchronize with #1. In the same way, #3 tried to synchronize with #2 by adding the delay d1 (against #2) at the end of its cycle. But the #2 and the #3 did not synchronize each other, because the #2 was already delayed for d2. No synchronicity seems to be obtained after that. But at this point, the flag for permission of getting other flash was set as on. (How often this permission is set is indicated as an optional parameter.) Then, reversely, #2 detects the flash from #3, and #1 detect the flash from #2. As a result, all three SunSPOTs went into synchronicity after several time ticks.

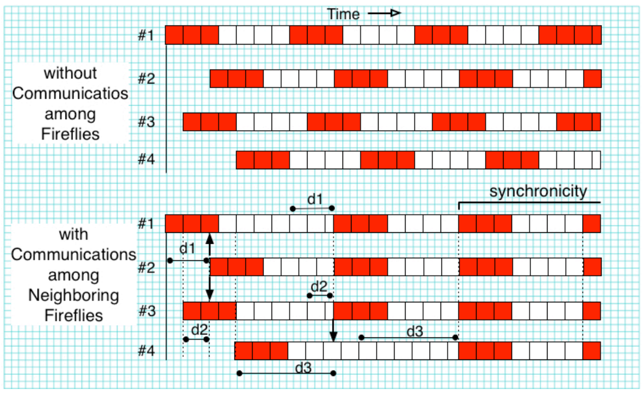
Time
#1
#2
#3
#4
without Communicatios among Fireflies
d1
synchronicity
with Communications among Neighboring Fireflies
#1
d1
#2
d2
#3
d2
d3
#4
d3

Fig. 3.2 Analysis of randomly started fireflies

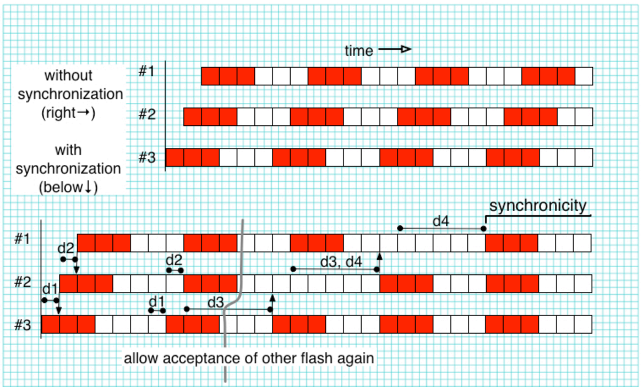
time
without synchronization (right→)
with synchronization (below↓)
#1
#2
#3
synchronicity
d4
#1
d2
d2
d3, d4
#2
d1
d1
d3
#3
allow acceptance of other flash again

Fig. 3.3 Analysis of regularly started fireflies

5 CONCLUSION

Synchronicity observed in a group of fireflies was reconstructed using four to eight SunSPOTs. It is based on the software simulation with phase delay model. Some problems arising from the use of real wireless devices, not occurred in the simulation, were discussed and solved. The most important problem was how to communicate between SunSPOTs using real ratio signal. This is not the problem in the simulation because the detection of other flashes is immediately obtained by simple calculations. The results from the proposed implementation exhibit relatively good synchronicity although its accuracy is not so high. These can be useful to observe synchronous behavior in the biological population without leaders. Also, it is meaningful to demonstrate the proposed implementation in the education of embedded software development.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Education, Culture, Sports, Science and Technology Grant-in-Aid for Scientific Research (C) 23501176.

REFERENCES

[1] Werner-Allen G, Tewari G, Pater A, Welsh M and Nagpal R (2005), Firefly-Inspired Sensor Network Synchronicity with Realistic Radio Effects, Proceedings of the 3rd ACM Conference on Embedded Network Sensor Systems (SenSys'05), San Diego, California, pp. 142-153.
[2] Tyrrell A, Auer G (2007), Imposing a Reference Timing onto Firefly Synchronization in Wireless Networks, IEEE 65th Vehicular Technology Conference, VTC2007-Spring. pp. 222 – 226.
[3] Leidenfrost R and Elmenreich W (2009), Firefly Clock Synchronization in an 809.15.4 Wireless Network", EURASIP Journal on Embedded Systems, Volume 2009, Article ID 186406, 17 pages.
[4] Wilensky U (1997), NetLogo Fireflies model, http://ccl.northwestern.edu/netlogo/models/Fireflies, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
[5] Wilensky U (1999), "NetLogo", http://ccl.northwestern.edu/netlogo/, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
[6] Smith R B (2007), SPOTWorld and the Sun SPOT, Proceedings of the 6th international conference on Information processing in sensor networks, pp. 565-566.
[7] Simon D, Cifuentes C, Cleal D, Daniels J and White D (2006), Java on the bare metal of wireless sensor devices: the squawk Java virtual machine, Proceedings of the 2nd international conference on Virtual execution environments, pp. 78-88.