Fireflies Synchronization in Small Overlay Networks

Iva Bojić, Mario Kušek
Faculty of Electrical Engineering and Computing
Department of Telecommunications
Unska 3, Zagreb, 10000, Croatia
E-mail: { iva.bojic, mario.kusek }@fer.hr

Abstract - Ad-hoc networks are very dynamic and nodes are entering and exiting network frequently. In such networks self-organization without centralized control is crucial for efficient network operation. To accomplish self-organization in small ad-hoc networks, the pulse-coupled oscillator model based on biological mutual synchronization such as that observed in flashing fireflies, is used. Fireflies are known to emit flashes at regular intervals when isolated, but in the group they entrain the pulsing of their lights to converge upon the same rhythm until synchronicity is reached. This paper investigates the influence of different overlay network topologies to synchronization time and network traffic.

I. INTRODUCTION

In certain parts of South-East Asia along side riverbanks, male fireflies gather on trees at dawn, and start emitting flashes regularly. Over time synchronization emerges from a random situation, which makes it seem as though the whole tree is flashing in perfect synchrony [1]. Computer scientists have often looked to nature for inspiration, this is one of that cases.

In all different species of fireflies, an emission of light represents communication that helps female fireflies distinguish males of its own species. This allows a female to recognize a specific male. She responds to him with a flash after a species-specific time delay [2].

Although the reason behind this synchronization is not fully understood, fireflies are not the only biological system displaying a synchronized behavior. This emergent pattern is present in human body but also among humans. For example, in heart cell [3] and neurons [4], or on the street where people, walking next to each other, tend to walk in synchrony.

Fireflies are used for synchronization in distributed systems. This paper investigates how different overlay network topologies affect synchronization time and network traffic in distributed system because we believe that fully connected network does not give the best results. In section II is related work. Section III explains conducted simulation while section IV describes used network topologies. Section V deals with conducted experiments and results are discussed in section VI. Finally, section VII concludes the paper and gives directions for future work.

II. RELATED WORK

In biological systems distributed synchronization is commonly modeled using the theory of coupled oscillators. A theoretical framework for the convergence to synchrony in fully-meshed networks was published in [5]. That Mirollo and Strogatz model has been applied to all kinds of problems that use fireflies as role model. For fireflies, an oscillator represents the internal clock dictating when to flash, and upon reception of a pulse from other oscillators, this clock is adjusted. Over time, pulses of different oscillators are transmitted simultaneously.

The Mirollo and Strogatz model has been applied to work which explores realistic networks effects such as transmission delays [6]. It has also been used for heartbeat synchronization protocol for overlay networks [7] and mechanisms for policy distribution and synchronization over a number of active servers [8].

In [5] was even proposed to explore how different network topologies affect on neighborhood of fireflies. But, to our knowledge, that haven't been done until now. The main contribution of our work is observation of different overlay network topologies affecting on two parameters: synchronization time and network traffic.

III. SIMULATION

The behavior that is modeled here is governed by the following rules [2]:

- Each firefly has an intrinsic flashing frequency determined with *threshold* parameter, and when left alone it will flash at periodic intervals;
- The flashes are timed by the progressive excitation within each firefly; the excitation increases until it reaches a *threshold*, at which point a flash is emitted and the excitation is reset to zero;
- If a firefly senses a certain amount of luminescence from its neighbors, it will reset its excitation to zero in order to flash simultaneously with those neighbors in the future; but, if the excitation is close enough to the flashing *threshold*, the flash has already been started and will proceed as planned even though the excitation is reset to zero; this is determined with *buffer* parameter.

A. The simulation skeleton

In this simulation, there are two threads; passive and active one (see Fig. 1).

```
1:  loop
2:      wait until excitation = threshold
3:      P ← findNeighbors();
4:      send flash to all peers in P;
5:  end loop
```

(a) active thread

```
1: loop
2:      receiveFlash();
3:      processFlash();
4: end loop
```

(b) passive thread

Fig. 1 The simulation skeleton: active and passive threads

In active thread each firefly sends its flash-message to neighbors. In related works [6], [7], [8], firefly neighbors are all other fireflies in the network. That is called fully connected network. Here are used different network topologies to narrow that neighborhood. So depended on used topology, each firefly has one or more neighbors, but not all of them. Further in the paper is explored how this affects on synchronization time and network traffic.

In passive thread each firefly first receives message in function *receiveFlash()* and then processes it in function *processFlash()* (see Fig. 1). In function *processFlash()* (see Fig. 2) firefly senses a certain amount of luminescence from its neighbors. If that amount of luminescence is larger than *trigger* and firefly's excitation is smaller than buffer, firefly will flash and reset its excitation to zero.

```
1:  P ← findNeighbors();
2:  sumLights = 0;
3:  for each Neighbor in P
4:      sumLights += Neighbor.light * 100 / distance
5:  end for each
6:  if (sumLights > trigger && excitation < buffer)
7:      excitation = 0;
8:  end if
```

Fig. 2 The function processFlash();

The *trigger* sets the amount of luminescence required for a firefly to reset its excitation prematurely. Each firefly's flash is counted as 100 units of light that decreases with the inverse square of the distance from source.

In simulation is used Euclid distance (see Fig 3), but that's not quit possible for real implementation. The real implementation will use different metrics – propagation delay. Propagation delay is time taken for a burst to propagate from the emitting to the receiving node. This time is proportional to the distance between two nodes.

```
1:  distance = sqrt ((x1 – x2) ^ 2 + (y1 – y2) ^ 2);
```

Fig. 3 The Euclid distance

B. The simulation parameters

The *numberFireflies* sets the number of fireflies in the simulation.

The *threshold* sets the excitation threshold at which point a firefly will flash and reset its excitation to zero.

The *buffer* sets how many time steps are necessary for the flashing signal to evolve and terminate in a flash. If a firefly is triggered to reset its excitation when the excitation is within "buffer" of the threshold, the flash will proceed as planned despite the resetting.

Different *topologies* are explained in next section.

C. The simulation restrictions

There are two differences between simulation and real world. The simulation does not support message delay as neither the possibility that message can be lost which happens in the real world. Also the firefly in the real world cannot transmit and receive flash at the same time and this is possible in the simulation.

IV. TOPOLOGIES

Simulation described in previous section has already been implemented in STARLOGO [9] but only for fully connected networks. This paragraph shows connection between all implemented topologies (see Fig. 4 – Fig. 8) and neighborhood.

Overlay network is build over a small ad-hoc mobile network which is constituted of mobile phones in range of Bluetooth signal.

Firefly synchronization is used to synchronize small ad-hoc mobile network where for each different overlay network topology function *findNeighbors*() (Fig. 1) returns different neighbors.

Implemented topologies are fully connected (fully meshed or all-to-all topology), ring, star, line and mesh topology.

A. Fully connected

Every firefly has (*numberFireflies* - 1) neighbors. That means we have fully connected network.

For example, neighbors from node 1 in full connected topology are nodes 2, 3, 4, 5 and 6, and from node 2 nodes 1, 3, 4, 5 and 6 (see Fig. 4).

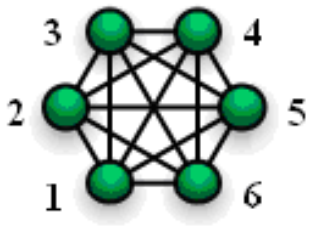

Fig. 4 Implemented network topologies – fully connected

B. Ring

Every firefly has exactly two neighbors, but not necessary the closest ones. Important is that total distance between all fireflies is the smallest.

For example, neighbors from node 1 are nodes 2 and 6, and from node 2 nodes 1 and 3 (see Fig. 5).

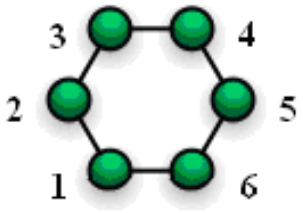

Fig. 5 Implemented network topologies – ring

C. Star

Center of the star must be found first. That is firefly closest to the center of all fireflies. It has (*numberFireflies* - 1) neighbors. Every other firefly has just one neighbor - center firefly (see Fig. 6).

For example, neighbors from node 6 are nodes 1, 2, 3, 4 and 5, but nodes 1, 2, 3, 4 and 5 have only one neighbor – node 6.

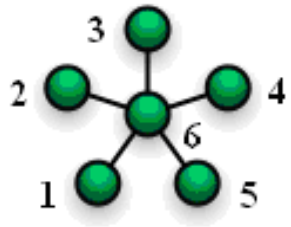

Fig. 6 Implemented network topologies – star

D. Line

First and the last firefly in line have only one neighbor and every other firefly has two neighbors. This topology is very alike to ring topology.

For example, neighbors from node 3 in line topology (see Fig. 7) are nodes 2 and 4, but node 1 has only one neighbor – node 2.

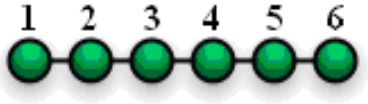

Fig. 7 Implemented network topologies – line

E. Mesh

Mesh topology has its own parameter *pointsForMesh*. If that parameter is equal to (*numberFireflies* – 1) then we have fully connected network, otherwise just partially connected. But if it's too small (for example 1 or 2), we probably won't have a connected graph, so in our simulation we use *pointsForMesh* greater than 2.

Also, it is important to mention that mesh topology doesn't give bidirected graph like other topologies. Fig. 8 shows one example of mesh topology where connection (red line) between two nodes without arrow is bidirected and connection (black line) with arrow is undirected.

For example, neighbors from node 1 are nodes 4, 5, 6, but node 1 isn't neighbor from node 4. Neighbors from node 4 are nodes 2, 3, 5 (see Fig 8).

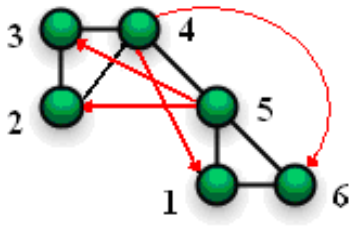

Fig. 8 Implemented network topologies – mesh(3)

V. Simulation Evaluation

Simulation is evaluated by using a simulator called MASON. Fig. 9 shows picture of fireflies' 3D simulation made in it.

A. Simulation environment

MASON (Multi-Agent Simulator Of Neighborhoods) is a single-process discrete-event simulation core and visualization toolkit written in Java [10].

In order to visualize the influence of different network topologies, the graphical user interface (GUI) is made. The GUI enables the user to modify various parameters during the simulation.

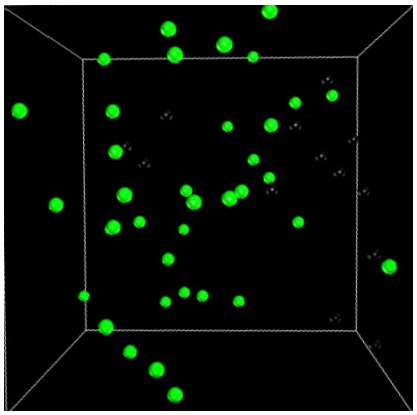

Fig. 9 Fireflies' 3D simulation in MASON

MASON models can be attached to a GUI toolkit, which enables visualization and manipulation of the model in both 2D and 3D.

B. Evaluation metrics

In order to compare the simulation results with different network topologies, the evaluation metrics are similar. Therefore, the two important parameters are: the amount of time units (discrete time Δt) until the system achieves synchronicity and the network traffic.

- *Time to sync:* This metric defines the time until all nodes have entered the synchronization state.
- *Network traffic:* Amount of messages sent between nodes (fireflies) during the synchronization process.

C. Parameter settings

Several parameter settings are the same for all experiments and are adapted to simulate the behavior of our tested environment (see Table I).

Table I Parameter choice used in all simulator experiments

Parameter	Value
threshold	20
buffer	2
numberFireflies	10

VI. Simulation Results

The Table II shows the time to sync and the amount of sent messages during the synchronization process in simulation. Each simulation is executed one hundred times and the average time to sync is used. In some cases synchronization is not achieved and unsuccessfulness of synchronization is in column "Unsuccessful sync (%)". The last column shows the average number of messages needed to achieve synchronization.

Table II The time to sync and the amount of sent messages for different topologies

	Average time to sync	Unsuccessful sync (%)	Average amount of messages
Fully connected	43,80	24%	202,14
Ring	100,38	32%	71,29
Star	39,54	2%	29,30
Line	125,27	38%	83,98
Mesh(4)	55,63	19%	90,91

Usage of some topologies on one hand increases amount of sent messages, but on the other hand this also increases the possibility that the network achieves synchronicity.

A. Fully connected

According to the Table II, in 76% of cases synchronization has been achieved, but not always in the shortest time. Amount of sent messages was the largest.

In each step every firefly sends its flash to all other fireflies. Light decreases with the inverse square of the distance from the source. In this case source is firefly that sends its flash-message to others fireflies. Some fireflies are just too far away from that firefly so they can't sense the amount of luminescence required for them to reset their excitation prematurely. That means that some flash-messages for some fireflies are useless and unnecessary overweight for network traffic.

Simulation results of this topology have shown that it isn't the best one. They justified our belief that other network topologies might be better choice for finding more suitable neighbors.

B. Ring

Results of ring topology haven't brought many surprises. Time to synchronize is larger then in fully connected topology, but amount of sent messages is far smaller.

Ring topology is a good choice. In 32% of cases synchronization hasn't been achieved. That is good result considering 24% in fully connected topology.

C. Star

It seams that this topology is the most suitable one. In only 2% synchronization hasn't been established. In all others it has been established within 40 steps (2 * threshold) and 30 messages. This means the shortest time to sync and the smallest amount of sent messages.

Considering all results, star topology is the best one.

D. Line

Although, this topology is very alike to the ring topology, results are much worse. Line topology has the longest time to sync, and the smallest chances to establish it (62%). The amount of sent messages is the second worse, just after fully connected topology.

Considering all results, line topology is the worst one.

E. Mesh

Mesh topology has average time to sync and average amount of sent messages. Table II shows results for mesh(4) topology. Mesh(4) means that every firefly is connected with its four closest neighbors.

This topology actually gives pretty good results. Changing *pointsForMesh* parameter, different mesh topologies can be obtained. For each realistic problem, optimal *pointsForMesh* parameter must be found.

In our simulation *pointsForMesh* parameter is equal to 4. There is no particular reason why it has been chosen the number 4. However, in section IV it has been explained why it is important to use *pointsForMesh* grater than 2, so we chose this one.

VII. Future Work And Conclusion

Fireflies provide an amazing spectable with their ability to synchronize using simple rules: each node maintains an internal clock dictating when to emit, and in return, this clock is adjusted when receiving.

These synchronization rules are particularly simple and well suited for a deployment in ad hoc networks. However, they are not directly applicable for all-to-all overlay network topologies.

We have tested different overlay network topologies to find the most suitable one. Although, there is no best one, star topology is close to the best.

However, this topology has one big disadvantage - single point of failure (SPOF); the assessment of a potentially single location of failure identifies the critical components of a complex system that would provoke a total systems failure in case of malfunction. Highly reliable systems may not rely on any such component.

In this topology, a single point of failure is center of a star, if it fails, will stop the entire system from working. Therefore, some strategies to prevent from total systems failure must be implemented. Most common one is redundancy. Redundant systems include a double instance for any critical component. In this case that means two centers of the star, where second center will begin working if first stops.

Fig. 10 shows what happens if first center of star (point number 0) fails. Then second one (point number 6) stars working, and the system is reliable.

Future work will rely on implementation of this algorithm on real mobile phones using Bluetooth. Some work has already been done, but there is still much work to be done.

Clearly, some modifications should be made to solve the problem with system reliability to use a star topology in real implementation.

Fig. 11 shows mobile's application GUI of our implementation on mobile phones. For now only fully connected topology has been implemented. However, this topology doesn't give gut results, so another approach is required.

Furthermore, we want to compare the results of our firefly synchronization with the use of existing synchronization approaches.

Acknowledgments

This work was carried out within research project 036-0362027-1639" Content Delivery and Mobility of Users and Services in New Generation Networks", supported by the Ministry of Science, Education and Sports of the Republic of Croatia.

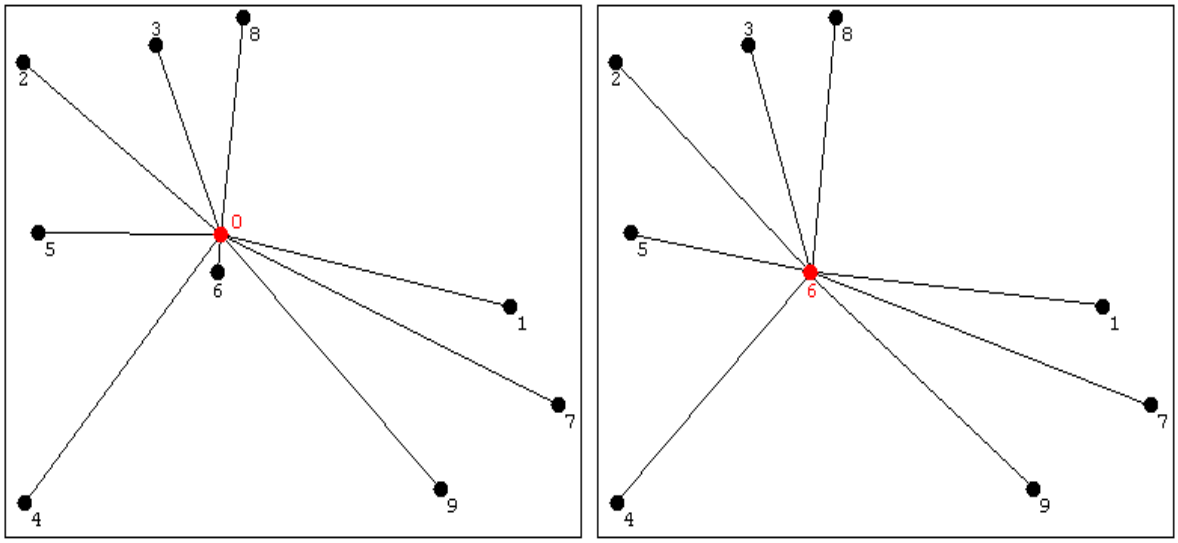

Fig. 10 Single point of failure - redundancy

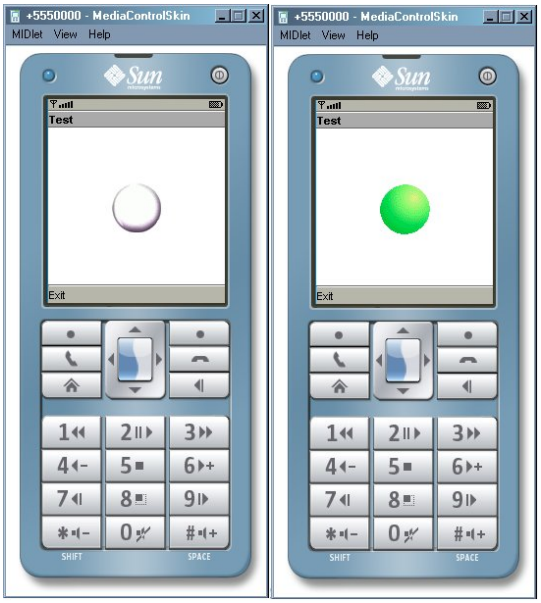

Fig. 11 Fireflies on mobile phone

REFERENCES

[1] D. Attenborough, "BBC Trials of Life: Talking to Strangers", 1990

[2] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau, *Self-Organization in Biological Systems*, Princeton University Press, 2001

[3] C. Peskin, "Mathematical Aspects of Heart Physiology", New York: Courant Institute of Mathematical Sciences, 1975

[4] J. Hopfield and A. Herz, "Rapid local synchronization of action potentials: Toward computation with coupled integrate-and-fire neurons", Proc. National Academy of Sciences USA 92, vol. 92, pp. 6655–6662, July 1995

[5] R. Mirollo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators", SIAM Journal on Applied Mathematics, vol. 50, no. 6, pp. 1645–1662, Dec. 1990

[6] A. Tyrrell, G. Auer, and C. Bettstetter, "Firefly synchronization in ad hoc networks," in Proc. MiNEMA Workshop 2006, Feb. 2006

[7] O. Babaoglu, T. Binci, M. Jelasity and A. Montresor, "Firefly-inspired Heartbeat Synchronization in Overlay Networks", First International Conference on Self-Adaptive and Self-Organizing Systems, July 2007

[8] I. Wokoma, I. Liabotis, O. Prnjat, L. Sacks, I. Marshall, "A Weakly Coupled Adaptive Gossip Protocol for Application Level Active Networks", Proceedings of IEEE 3rd International Workshop on Policies for Distributed Systems and Networks – Policy, 2002

[9] StarLogo web site, http://education.mit.edu/starlogo/, visited on 12.01.2009

[10] MASON web site, http://www.cs.gmu.edu/~eclab/projects/mason/, visited on 12.01.2009