Fault-Tolerant Wireless Sensor Networks using Evolutionary Games

Ricardo Villalón
Computer Science Department
The University of New Mexico
Albuquerque, NM 87131
villalon@cs.unm.edu

Patrick G. Bridges
Computer Science Department
The University of New Mexico
Albuquerque, NM 87131
bridges@cs.unm.edu

ABSTRACT

Wireless sensor networks are commonly used to implement applications for habitat monitoring, bridge structure monitoring, or monitoring large networks such as power grids. In these environments, faults are very frequent, maintenance and replacement of system components is complicated or expensive, and survivability of the network itself becomes a requirement.

In this research, we take an evolutionary approach to implement protocols for reliable and survivable applications for wireless sensor networks. We propose a system with a virtual biological community of organisms living inside an ecosystem composed of sensor nodes and data packets. These organisms execute some of the functions assigned to the network and implement these functions in several ways. Over time, organisms with the best strategies are selected by the natural selection process.

As an example, we present a collection protocol called Evolutionary Collection Protocol (ECP) to collect information in a sensor network and send it towards the root node. Organisms in the community execute the routing and data transportation operations in the context of an evolutionary game. Some preliminary results about population dynamics and network operation are presented.

Keywords

Fault-tolerance, wireless sensor networks, evolutionary games, natural selection.

1. INTRODUCTION

The complex fault scenarios of wireless sensor networks (WSN) applications are usually caused by the harsh environmental conditions where they are deployed or by the physical characteristics of the sensor devices. Additionally, long-term applications such as habitat monitoring, bridge monitoring or power grid monitoring require high levels of survivability for individual network components and long connectivity times for the whole network.

Existing research about fault-tolerance of wireless networks is insufficient for real-time WSN with complex failure environments [12], [13]. Most existing fault models assume constant failure rates for all components of the network instead of considering a more dynamic approach, with different failure rates for every component. Therefore, network operation can be partially or totally interrupted when several failures produce chaotic situations because they were not considered appropriately. Additionally, most current approaches ignore the real-time issues associated with failures because they abstract the real time to rounds or iterations of the algorithm.

We propose an evolutionary network model inspired by biological structures and supported by evolutionary game theory to provide an efficient framework for reliability and survivability of WSN, even in the presence of complex failures. In this approach, a *virtual biological community* hosted in the nodes helps to execute the functions assigned to the network. The community is composed of several species of organisms having different roles, some of them execute functions for the high-level application, others support and preserve the community, or manage the environment and resources.

The functions of the network are implemented in the context of one or more simultaneous evolutionary games. Virtual organisms are players of the game and the network functions are implemented as strategies of players. Strategies can be inherited between generations of players, supporting the evolution of the best strategies through the process of natural selection.

Section 2 describes the contributions of this research. Sections 3 and 4 give a more detailed presentation of the problem we try to solve and the main components of the evolutionary approach. Sections 5 and 6 describe an example evolutionary protocol called ECP and present some preliminary results. Sections 7, 8, and 9 describe future and related work and present the conclusions.

2. RESEARCH CONTRIBUTIONS

An important number of research projects are dedicated to fault-tolerant systems using bio-inspired approaches but, to the best of our knowledge, there are no implementations where an evolutionary approach considers real-time faults as part of the system itself. In our approach, reliability and fault-tolerance are included in the design of the system components; they are embedded into the rules of the evolutionary game and implemented as strategies of the players.

Furthermore, our approach uses evolutionary games as a formal methodology to compare and evaluate different algorithms in a practical environment, including algorithms migration when players move between nodes over the network.

The main contributions of this research are:

- A useful framework and a set of strategies to deal with simple and complex failures in unreliable WSN.
- A software platform to create fault-tolerant WSN, with dynamic fault models, even for individual nodes of the network.
- A software platform to evaluate the evolution of strategies in WSN.
- A software platform to create adaptive applications for dynamic execution environments.

- A platform to create self-healing and self-configuring wireless sensor networks.

3. PROBLEM DESCRIPTION

Existing models for reliability and survivability of wired networks present some issues when applied to WSN because of contrasting conditions [17]. WSN are usually built from small, low-cost, unreliable hardware devices that have limited processing capabilities and low memory capacity. These architectural restrictions together with the environmental conditions where they are usually deployed make these networks susceptible to failures of different types.

To make this situation worse, the cost of repairing failed nodes can be high because of difficulties in accessing the physical location of the network. In these cases, the network itself has to execute temporary procedures to keep the system online while more stable solutions are applied.

Additionally, some real-time failures can not be well analyzed with conventional failure models because such models assume constant failure rates, instead of using a model with different failure functions for every component of the network. A good example of this are hybrid fault models [13] that are not well represented with constant failure rates.

In a real-time WSN, every component can have a different failure function, and failures can not be statically analyzed. Occurrences of complex or hybrid faults at different nodes and different times can produce important damages to the network. Therefore, it is worth considering time-dependent failure rates and time-dependent failure modes as defined in [13], these models are called *dynamic hybrid failure models.*

4. EVOLUTIONARY APPROACH

4.1 Evolutionary Games

Evolutionary Game Theory [14] applied to the Natural Selection process and Darwinian Dynamics [9] provides an intuitive and formal path to start building an evolutionary framework to host a biological community inside a WSN. Evolution by natural selection is an evolutionary game because it has players, strategies, strategy sets, and payoffs. Players are the individual organisms. Strategies are heritable phenotypes or visible features of players. A player's strategy set is the set of all evolutionarily feasible strategies. Payoffs are expressed in terms of fitness, and fitness is the expected per capita growth rate of a given strategy within an ecological circumstance [18].

In the proposed virtual community, players are virtual organisms. Strategies are functions or behaviors of the network to be executed by the players (e.g., picking the next hop on the path to the root node, or selecting the right time to move from node to node). Payoff is defined as the per capita growth rate over time of the players' strategies for the surviving individuals.

4.2 First evolutionary model

We first consider a typical sensor network application. It is a *collection network* where nodes have sensors of different types such as temperature, light, or humidity. Sensor nodes collect values and send them to the root node for processing.

Then, we create a community of individuals with capabilities to execute some functions of this application (Figure 1):

- *Messengers:* carry collected values from originating nodes to the root node.
- *Advertisers:* advertise and share routing information between neighbor nodes. For example, Advertisers can collect information that can be used by Messengers to reach the root node.
- *Guardians:* execute management functions inside the node, such as resources administration, CPU usage or memory management.

The physical location of the organisms is very dynamic; they move from node to node while executing some function according to their own capabilities. Organisms can migrate to other nodes for replication purposes. They can also die or be killed by other organisms.

When a node crashes, all organisms living in that node die, but Advertisers located at neighbor nodes rapidly detect that the crashed node is no longer active. Messengers use this information to exclude crashed nodes from the path to the root node. Guardians execute memory and time management functions inside the node. They also play an important role when more complex failures are detected because they can declare alert or emergency states in the nodes to enable complex interactions between organisms of several nodes.

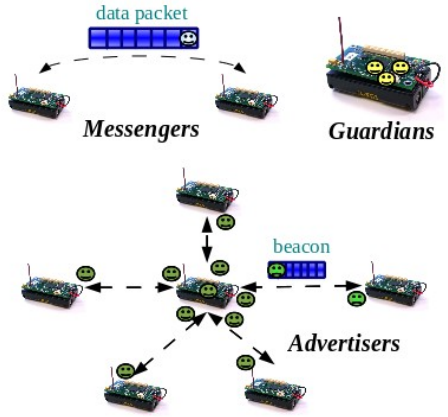

Figure 1: a) Messengers move application data between nodes; b) Guardians take care of internal node management; c) Advertisers maintain routing tables and do fault-tolerance monitoring.

4.3 Micro-component Framework

The proposed game approach imposes some requirements on the underlying software infrastructure because it allows several simultaneous implementations (values of the strategies) for some components of the system (strategies of players) .

There are some issues related to the design of this infrastructure, and the proposed evolutionary platform. For example:

- Some functions require several implementations because they are strategies of players, but some other functions require only one implementation without the complexity of the strategies.
- The system has to be flexible enough to add, change or remove games, according to the requirements of new applications.
- Fault-tolerance and optimization components are very dynamic. The system has to support this dynamic behavior.

To satisfy these requirements, we propose a software framework based on small functional units called *micro-components*. Micro-components implement control flows representing the processes of the system, similar to the micro-protocols proposed in [6] with some extensions to provide specific functionality required by a sensor network application, such as split-phase or two-phase components and virtual components to implement strategies.

Figure 2 shows the flow control for Advertising Receive process explained below. Each box represents a micro-component, solid gray boxes represent micro-components that are not strategies of any player, green boxes (with 45 degrees pattern) represent strategies for Advertisers, yellow boxes (with vertical pattern) are strategies of Guardians. All player strategies have a similar behavior to virtual functions in C++: the specific implementation of the function is selected at run-time, depending on the strategy value of the player executing the function.

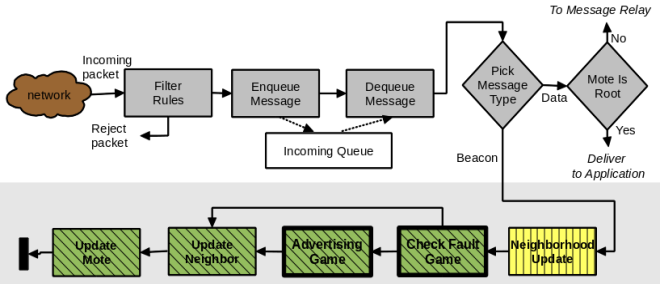

Figure 2: Process flow for Advertising Receive for ECP

5. EVOLUTIONARY COLLECTION PROTOCOL

5.1 Protocol Description

ECP is a collection protocol based on Collection Tree Protocol (CTP) [10], a well-known protocol used in sensor networks that collects information at sensor nodes and send it towards the root node. The original CTP is a best-effort, multi-hop delivery protocol. ECP is designed to have the same basic features but adds an evolutionary fault-tolerance mechanism and some performance and energy-optimization features.

ECP is implemented as an evolutionary game using a virtual biological community. Players are the same as described in section 4.2, with Messengers, Advertisers and Guardians.

The protocol is constructed using the following steps:

1. Create a process flow for each high-level function. ECP has five of these functions: Message Send, Message Receive, Advertising Send, Advertising Receive and Check Fault. Figure 2 shows the details for Advertising Receive.
2. For each flow created in the previous step, pick the components to optimize and assign them to some player species. In Figure 2, Neighborhood Update is assigned to Guardians; Check Fault Game, Advertising Game, Update Neighbor and Update Mote are assigned to Advertisers.
3. Create additional components representing internal one-to-one games to satisfy the optimization or fault-tolerance requirements, and to speed up the evolutionary process. In Figure 2, components Check Fault Game and Advertising Game are created to detect crashed nodes and decrease the overhead generated by routing tables updates.

Figure 3 shows a layered architecture for a sample application, the ECP layer is composed of the five process mentioned in step 1. The colored-faces layer contains the game and player definitions. The operating system layer is based on TinyOS, the same platform used for the reference CTP implementation.

The five functions for the current ECP implementation are:

- *Message Send:* send data messages to other nodes.
- *Message Receive:* receive and process data and beacon messages received from other nodes.
- *Advertising Send:* periodically advertise the node to the network.
- *Advertising Receive:* provide specific processing for beacon packets. This flow contains the two internal games (advertising and check fault) required to optimize the advertising process and fault-tolerance.
- *Check fault:* implement monitoring of neighbor nodes. This flow is complementary to Check Fault Game.

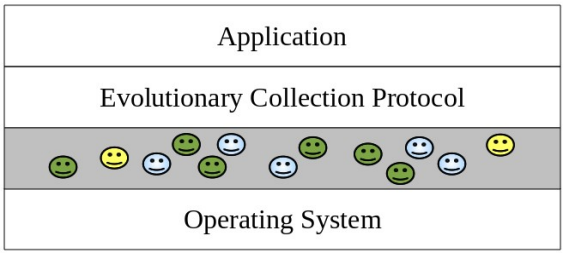

Figure 3: Layered design for ECP

5.2 Game Design

The proposed game structure is simple enough that we can have many players on each sensor node, and the population dynamics generated by the natural selection process can help to produce the desired results. Each player is composed of three elements:

- A numeric identifier for the species.
- A state value, usually composed of a few bit fields.
- A strategy set represented by an array of bits (32 bits for the current implementation), where independent strategies are represented by subgroups of bits.

Each value for a strategy represents a different implementation of the corresponding function. For example, Messengers have a *pick next hop* strategy, the system can have several implementations of this function, the natural selection process provides the evaluation over time of each implementation.

The full set of strategies for Messengers, Guardians and Advertisers is:

Messengers strategy set

- *Next hop:* pick the next hop towards the root node.
- *Message timing:* pick the time to move to the next hop.
- *Replication:* pick the time and the number of replicas for next replication.

Advertisers strategy set

- *Beacon timing:* pick the time to send a beacon, when playing the advertiser role.
- *Check-fault timing:* pick the time to check for neighbor failures, when playing the neighbor monitor role.
- *Energy saving:* provide thresholds for some parameters of the advertising game.
- *Replication*: pick the time and the number of replicas for next replication.

Guardians strategy set

- *Timer management:* implement the strategy for managing the system timer.
- *Neighborhood management:* implement the strategy for adding, updating, and replacing neighbors from the routing table.
- *Replication:* pick the time and the number of replicas for next replication.

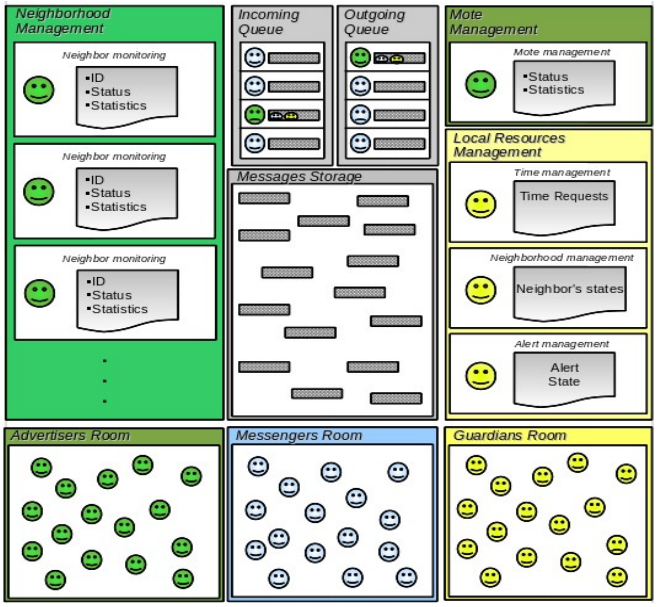

Figure 4: ECP game structures

Figure 4 shows a layout of the game data structures created at each sensor node. The neighborhood, node management, local resources management, and the queues are used for players to execute all their functions, replication is executed inside the rooms.

6. PRELIMINARY RESULTS

6.1 Evolutionary Software Framework

A software tool to start running evolutionary applications is the first important result of the project. ECP protocol is the first evolutionary example. Figure 5 shows a block diagram of the framework:

- *Micro Component Framework* provides the underlying support to create applications using micro-components.
- *Evo Game System* enable the game definition and the creation of strategies as micro-components.
- *Evo Components* and *Other Evolutionary Components* are complementary components supporting the implementation of higher level components.

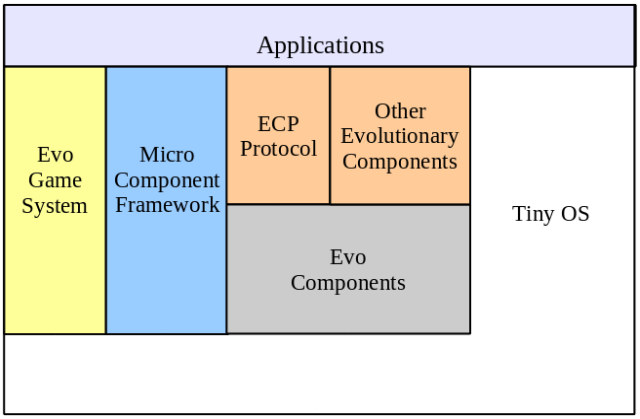

Figure 5: Block diagram of the software tool created

6.2 Population Dynamics and Evolution

The evolution of strategies for Advertisers can be tested with a simple TinyOS application that starts running the ECP protocol, no transmission of data packets is required because the routing tables are updated automatically by Advertisers.

To produce some initial results about population dynamics and evolution of strategies, we have simulated tests with the following parameters:

- A network with 16 nodes, organized in a grid topology, with a distance of 6 meters between nodes.
- The noise model for wireless transmission was set to a very a low level, to simulate ideal conditions.
- Tests include only creation of routing tables, then only strategies for Advertisers are tested for beacon timing optimization. Eight values for time between beacon packets are tested for evolution.
- The rooms of participating players are initialized with random samples of players, using only values of allowed strategies, to 50% of the capacity of the room.
- The set of possible strategies is restricted to test independent parameters of the system.

For these tests the timing between beacon packets is optimized and the strategies with greater time are expected to survive when the metric or cost to move from node to node is stable.

Figure 6 shows snapshots of population dynamics. Columns represent the 8 possible strategies, rows represent the simulation time for each snapshot. Each entry (i, j) of the grid contains the number of organisms using strategy j at time i. Starting at row 3, some entries have a zero value meaning that all players using the corresponding strategy died.

Last row shows strategy 4 having the majority of individuals, only a few players using strategy 1, and all other strategies did not survive. Note that strategy 4 starts with less individuals than strategy 1, but over time it evolves and become the evolutionarily stable strategy (ESS).

	Strategy ID							
Time	1	2	3	4	5	6	7	8
00:05:22	38	36	27	15	61	41	91	68
00:20:30	94	2	9	52	119	3	190	110
00:40:02	187	1	1	110	121	0	109	44
00:50:47	276	1	0	134	80	0	63	26
01:22:32	341	0	0	189	29	0	13	8
01:47:25	391	0	0	174	13	0	3	0
03:07:01	439	0	0	144	0	0	1	0
03:57:18	455	0	0	128	0	0	0	0
30:01:45	185	0	0	403	0	0	0	0
50:06:50	5	0	0	581	0	0	0	0
99:02:52	4	0	0	586	0	0	0	0

Figure 6: Evolution of strategies for Beacon Timing

7. FUTURE WORK

Interesting next steps for this research include identification of evolutionary strategies for specific topologies and network environments, and identification of sections of the network with organisms using different strategies because of different failure rates and changes in the environment.

Another worthwhile experiment would be to devise new metrics, parameters and algorithms for more complex network behaviors. For example: adaptations of ant colony optimization algorithms for advertising.

Implementation of stochastic and reinforcement learning strategies, or the creation of more complex organisms with higher-level functions are also considered as future improvements.

8. RELATED WORK

Many theoretical and practical research projects on WSN are inspired by biological systems because of the amazing results we can find in complex systems in nature. In this section we describe the basic differences from our research project with some previous projects having a similar approach.

[11], [12], [13] propose a theoretical layered architecture to create fault-tolerant sensor networks. In this approach, sensor nodes are players for some evolutionary game. They propose extensions to classical failure models to represent real-time and dynamic hybrid models. Our project uses some ideas about fault-tolerance models proposed on this work but. On another hand, they propose evolutionary games as a layer of the model but there is no clear process for evolution of strategies, and they do not define a population dynamics based on a fitness function.

[3], [4], [5] presents a different example of biologically-inspired sensor network platform, based on the behavior of bees. They define a system with an evolutionary process; they also have players, population dynamics, migration from node to node, adaptation, and a selection process. The model defines a procedure for player replication and death, based on energy levels of players. But this system also lacks a well defined group of strategies to characterize the species of players. They do not have a clear evolutionary process for the strategy sets of players, with a fitness function to provide a metric for the resulting evolution.

[8] propose evolution of cooperation in a large WSN with a static population. They define network nodes as players in the context of an evolutionary game motivated by the iterated prisoners dilemma game with strategies and fitness functions. They have a fitness function but there is no population dynamic because population is fixed, and they only prpopose the solution for a specific problem involving cooperation.

Related to fault-tolerance, [2] presents a survey of different fault-tolerant routing techniques, based on retransmission and replication schemes, with information being transmitted several times, or replicated for redundancy. [16] describes a layered structure including node, network, sink and back-end to propagate faults in WSN. They describe detection techniques for self-diagnosis, group detection, hierarchical detection, and recovery techniques such as active replication and passive replication, but none of these techniques consider different failure models for each device on the network, and there are not considerations for real-time behavior of components.

Finally, [1] presents a theoretical definition to design congestion control protocols, using the TCP protocol as reference. With that approach, you can compare different versions of the protocol by changing values for some parameters. This allows to compare and select the best version according to the selected values. We have used some of the conceptual ideas proposed on this research to design our example protocol ECP.

9. CONCLUSIONS

This paper presents a biologically-inspired software framework to create improved fault-tolerant protocols for WSN, using an evolutionary game approach.

The system provides a software tool to test and compare different strategies. It also produce adaptive behaviors by selecting the best strategies, according to the conditions of the environment, and the available strategy sets.

Preliminary results show that some strategy sets can become evolutionarily stable (ESS) if the defined internal games provides the right guidelines for the evolutionary process, according to predefined application requirements.

10. REFERENCES

[1] Altman, E. et al. An evolutionary game approach for the design of congestion control protocols in wireless networks. 6th International Symposium on WiOPT, 2008.

[2] Alwan, H. and Agarwal, A. A Survey on Fault Tolerant Routing Techniques in Wireless Sensor Networks. Third International Conference on Sensor Technologies and Applications, 2009.

[3] Boonma, P. and Suzuki, J. Biologically-Inspired Adaptive Power Management for Wireless Sensor Networks. In G. Aggelou(ed.), Handbook for Wireless Mesh & Sensor Networking, Chapter 3.4.8, pp. 190-202, McGraw-Hill, September, 2008.

[4] Boonma, P. and Suzuki, J. A Biologically-Inspired Architecture for Self-Managing Sensor Networks. In Proc. of the 3rd IEEE Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), IWWAN subtrack, Reston, VA, September, 2006.

[5] Boonma, P. et al. BiSNET: A Biologically-Inspired Architecture for Wireless Sensor Networks. In Proc of the 2nd IEEE International Conference on Autonomic and Autonomous Systems, Santa Clara, CA, July, 2006.

[6] Bridges, P. et al. A Configurable and Extensible Transport Protocol. ACM/IEEE Transactions on Networking, Vol. 15, Issue 6, 2007.

[7] Charalambous C. and Cui Sh. A Biologically Inspired Networking Model for Wireless Sensor Networks. IEEE Network, May/June 2010.

[8] Crosby, G. and Pissinou, N. Evolution of Cooperation in Multi-Class Wireless Sensor Networks. 32nd IEEE Conference on Local Computer Networks, 2007.

[9] Darwin, C. The Origin of Species. Barnes & Noble Classics, 446 pp, 2004.

[10] Gnawali, O. et al. Collection Tree Protocol. Technical Report SING-09-01

[11] Ma, Z. and A. W. Krings, Bio-Robustness and Fault Tolerance: A New Perspective on Reliable, Survivable and Evolvable Network Systems. Proc. IEEE Aerospace Conference, March 1-8, 2008, Big Sky, MT.

[12] Ma, Z. and A. W. Krings. Dynamic Hybrid Fault Modeling and Extended Evolutionary Game Theory for Reliability, Survivability and Fault Tolerance Analyses. IEEE Transactions on Reliability, Vol. 60, No. 1, March 2011.

[13] Ma, Z. and A. W. Krings. Dynamic Hybrid Fault Models and the Applications to Wireless Sensor Networks. MSWiM'08, October 27-31, 2008, Vancouver, BC, Canada.

[14] Maynard Smith, J. Evolution and the Theory of Games. Cambridge University Press, 224 pp, 1982.

[15] Michod, R. Darwinian Dynamics, Evolutionary Transitions in Fitness and Individuality. Princeton University Press, 262 pp, 1999.

[16] Moreira, L. et al. A Survey on Fault Tolerance in Wireless Sensor Networks. http://www.cobis-online.de

[17] Tanenbaum, A., Steen, M. Distributed Systems Principles and Paradigms. Prentice-Hall, pp 361-367, 2002.

[18] Vincent, T. L. and J. L Brown. Evolutionary Game Theory, Natural Selection and Darwinian Dynamics. Cambridge University Press, 382 pp, 2005.