# Distributed Symbolic Network Quality Assessment for Resource-constrained Devices

Andrea Augello*, Salvatore Gaglio*†, Giuseppe Lo Re*, and Daniele Peri*
*{andrea.augello01, salvatore.gaglio, giuseppe.lore, daniele.peri}@unipa.it
* Department of Engineering, University of Palermo, Viale delle Scienze, Ed. 6, 90128 Palermo, Italy
†ICAR-CNR, 90146 Palermo, Italy

***Abstract*—After a Wireless Sensor Network (WSN) is deployed it is subject to significant variations of the quality of its radio links during its lifetime. Knowledge of the condition of the wireless links can be useful to optimize power consumption and increase the reliability of the network. However, resource-constrained nodes may not be able to spare the storage space for network monitoring code. Also, reprogramming deployed nodes can be costly or unfeasible. In this work, we show how an approach based on the exchange of symbolic executable code among nodes enables the assessment of the network status in terms of Packet Reception Rate (PRR) with no extra storage requirements on deployed networks. We also compare the predictions made through this estimate with the actual network behavior.**

## I. Introduction

A Wireless Sensor Network (WSN) consists of sensors, also called nodes, that exchange data through radio interfaces. WSNs are used in a wide range of endeavors, like military surveillance, agriculture, environment monitoring, and vehicular area networks [1].

WSNs typically use low-powered radios, whose signal quality is highly susceptible to external environmental variables that can change over time [2]. The quality of radio links can be subject to fluctuations [3] and may exhibit characteristics that differ from the ones observed during pre-deployment testing [4]. Message exchanges have a great impact on energy consumption and network lifetime [5], therefore link failures can cause a significant increase in energy consumption [6]. Moreover, knowledge on the link quality can improve the routing inside a WSN [7]. Thus, assessing the network quality is of the utmost importance [8].

Traditional approaches to network quality estimation require some code to be already loaded on the nodes for this specific purpose [9]–[11]. In this work instead we adopt a symbolic approach as the basis to implement network quality assessment on deployed WSNs.

Real-time executable symbolic code exchange between nodes can introduce complex behaviors into already-deployed WSNs via runtime code injection [12]. Indeed, the symbolic approach enables many testing schemes without requiring special-purpose routines on the nodes [13], leaving more resources available for other applications.

A symbolic approach is, moreover, easily extended to application-specific formalisms, such as fuzzy logic [14].

The remainder of the paper is organized as follows: in Section II we present an overview of the features of the used software platform, Section III outlines the proposed quality assessment scheme, in Section IV we provide experimental results. Finally, Section V reports our conclusions.

## II. Symbolic execution platform

Popular general-purpose operating systems for WSNs like TinyOS and Contiki [15] primarily focus on efficient resource usage and reduced power consumption, and support the development of high-level applications.

With these frameworks, application code is cross-compiled and uploaded on the device storage through a wired connection, with little flexibility and extensibility at runtime, making reprogramming already deployed nodes a complex endeavor [16].

Alternatively, interpreted languages enable the dynamic execution of programs speeding up the development process, and allowing debugging and modifications to the programs even at runtime. These interpreters usually run atop a general-purpose operating system or need to be embedded inside an application [17], dramatically increasing resource consumption and memory occupation [18], and limiting the access to the underlying hardware.

In this work we leveraged the software platform DC4CD [19], which permits symbolic code injection and execution on resource-constrained WSN nodes due to its compact memory footprint.

In DC4CD, which is based on the stack-based programming language Forth [20], programs consist of sequences of symbols, called *words*. Words are stored in a dictionary that can easily be expanded.

Colon (:) and semicolon (;) are the Forth words to start and end the definition of new words in terms of a sequence of already-defined executable words:

```
: <new word> <word 1> <word 2>...<word n> ;
```

It is also possible to define special symbols, known as *markers*, to create restore points in the dictionary: executing a marker rolls back the dictionary to its state before the marker was defined, deleting all the words defined after it. These mechanisms permit the interactive development on resource-constrained devices [21], even of distributed applications, reducing development time [22].

Words use a stack to pass return values and parameters, for example, when executing the sequence of words

```
1 2 +
```

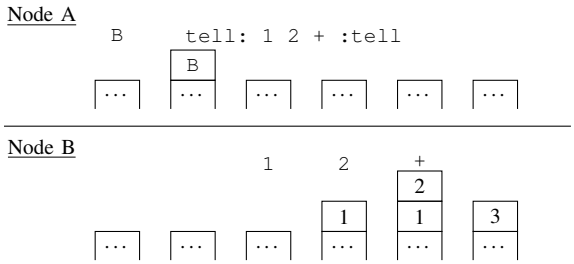


Fig. 1. Stack effects of the execution of the "`B tell: 1 2 + :tell`" Forth code. `B` is placed on the stack of Node A and is consumed by `tell:`, Node B executes the arithmetic operations laves the result on top of its stack.

literals `1` and `2` leave their values on top of the stack, to be consumed by the word `+`, which then leaves their sum, `3`, on the stack.

To facilitate the exchange of executable symbolic code, the platform natively provides support to distributed computing schemes through two special-purpose words, `tell:` and `:tell`. These words build IEEE 802.15.4-2003-compliant messages containing all the sequence of words enclosed between them as payload of datalink level packets. These messages are sent to the node whose MAC address is on top of the stack when `tell:` is executed, placing that value in the destination address field of the packet. Upon reception, the destination node immediately executes the received instructions without any further translation step. So if a node were to execute the code

```
1D01 tell: VALVE OPEN :tell
```

then said node would send the message `VALVE OPEN` to the MAC address `1D01`. The receiving node, provided that its dictionary holds definitions for the received words, would run that code producing the desired effect, i.e. opening the valve.

As a more practical example, given two nodes, Node A and Node B, with A and B being hexadecimal IDs as in the previous example, when Node A executes the symbolic code

```
B tell: 1 2 + :tell
```

`B` is put on top of Node A stack, and is then consumed by the `tell: :tell` construct. All the following words up to `:tell` are not executed and are instead put in the payload of a packet sent to node B as soon as `:tell` is encountered. Upon reception, Node B will execute the symbolic code `1 2 +` in the payload leaving the final result on top of its stack. The effects on the stacks of both nodes are shown in Fig. 1.

## III. Point-to-point packet reception rate estimate

As shown in Section II, executable code can be sent as a chain of symbols inside a regular message. Hence, by sending the appropriate executable code, it is possible to have the nodes execute non-trivial actions without programming them beforehand. Imposing no extra burden on the limited flash memory of the resource-constrained device.

In this work, we used this characteristic of the adopted platform to generate a connectivity matrix of the WSN.

We assume a 1-hop star topology, meaning that the bridge node `Node0` can communicate with every other node in the WSN. To obtain an estimate for the packet reception ratio (PRR) in each directed link (`Node1`, `Node2`) in the network we use a recursive code exchange, employing nested `<destination address> tell: <code> :tell` primitives, building a chain of messages to force communication between nodes.

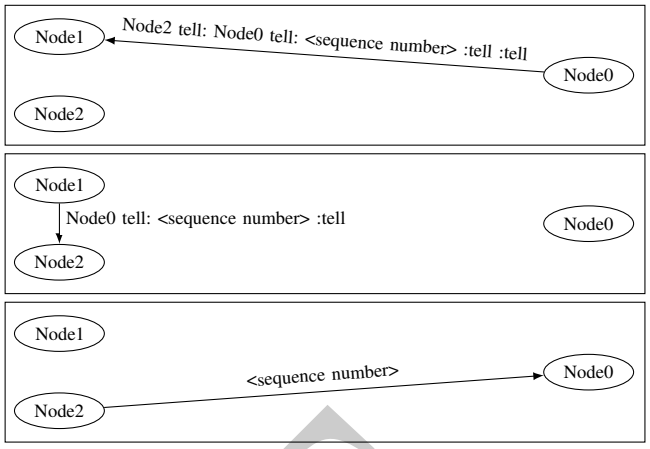


Fig. 2. Messages sent during one iteration of the point-to-point packet reception rate estimation.

The devised scheme entails the following steps:

1) through a serial connection we send instruction to the bridge node to have it transmit a message to the first node in the pair;
2) the sent message contains `tell: :tell` instructions to forward some payload to the second node in the pair;
3) finally, the second node receives a message with instructions to send the bridge node a sequence number uniquely identifying this message;
4) after the time required for `Node2` to transmit this message has elapsed, the last received sequence number can be retrieved from the top of the stack of the bridge node, if present;
5) it is then possible to check whether the expected sequence number was received or some error occurred during this process.

Fig. 2 shows an example of the exchanged messages in an instance of this procedure.

These steps are repeated $k$ times for each ordered pair to obtain an estimate of the ratio of messages lost between the two nodes. We consider ordered pairs as it is frequent in WSNs that links are asymmetric [23].

The execution of these steps for each pair yields a connectivity matrix containing the PRR for each directed link in the WSN (Fig. 3).

The order in which the pairs are tested is randomized to minimize spurious correlations with possible external interferences. In this work, for simplicity, we assume lossless communication between the nodes and the bridge. In fact, the bridge may not even be permanently part of the deployed network and act as a probing node only occasionally instead (e.g. a mobile node on an unmanned aerial vehicle).

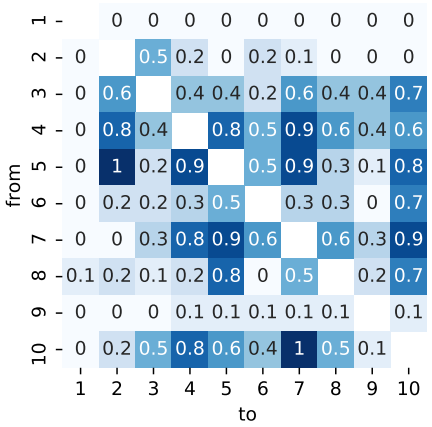


Fig. 3. Connectivity matrix of the WSN. Each cell contains the PRR between a pair of deployed nodes computed through 10 iterations of the quality assessment scheme.

## IV. Experimental evaluation

To test this system we deployed ten nodes plus a bridge node in a domestic environment (Fig. 4). All the nodes are IRIS motes with an IEEE 802.15.4 compliant RF transceiver, a 4 KB EEPROM, 128 KB of Flash memory, and an 8 KB static RAM.

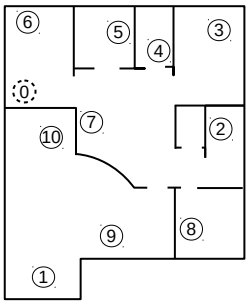

Fig. 4. Nodes distribution inside the home environment where the nodes were deployed.

The quality assessment was carried out through $k = 10$ repetitions for each pair of nodes. The resulting connectivity matrix, expectedly asymmetric, is reported in Fig. 3.

To verify that the predictions provided by our model are in agreement with the actual behavior of the network we devised a simple distributed application involving the following steps:

1) the bridge node broadcasts an initialization message to all the nodes in the network;
2) each node, upon receiving the initialization message, resets a counter for the received messages and starts a timer counting down to a value proportional to its ID;
3) when its timer expires each node broadcasts the update message;
4) whenever a node receives an update message, it increments its counter.

At the end of the execution, each node is queried for the number of received messages. Given that part of this test

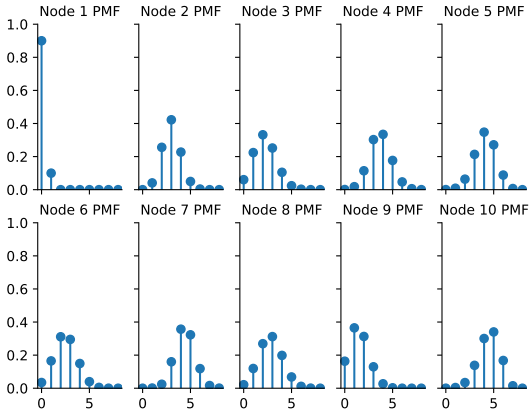


Fig. 5. Probability mass function of the random variable "total received messages" for each node.

scheme deterministically depends on node IDs, we performed the evaluation by repeating this test ten times assigning each time a different ID from the 0-65535 range to every node.

When a node broadcasts a message, its reception by all the other nodes is considered independent. Moreover, we also assume independence between the reception by a node of messages coming from different sources.

Under these premises, the reception of each message by a specific node is a Bernoulli trial, with chances of success depending on the message source. The total amount of messages received by a node is then the outcome of a series of independent Bernoulli trials with distinct distributions. Thus it can be modeled as a Poisson binomial distribution [24].

Given a set $S$ of nodes, if each node sends one message, the expected number of received messages by node $n$ is given by Eq. 1 and its variance by Eq. 2.

$$\mu_n = \sum_{i \in S \setminus n} p_{n,i} \tag{1}$$

$$\sigma_n^2 = \sum_{i \in S \setminus n} (1 - p_{n,i}) p_{n,i} \tag{2}$$

with $p_{i,j}$ corresponding to the PRR in the $i$-th column and $j$-th row of the connectivity matrix in Fig. 3.

Each node has a different probability of receiving messages from any given origin node and, consequently, a distinct probability mass function (PMF) for the random variable "total received messages". The computed probability mass function for each node used in the experimental setup is shown in Fig. 5.

The average outcome of ten runs of our experiment compared to the theoretical value given by the computed PMF is reported in Fig. 6. Most observed values are within $1.5\,\sigma$ of the predicted value. The prediction tends to be a lower bound of the actual number of received messages, partially because the lossless bridge assumption underestimates link quality and also because the test application exchanges messages that are much shorter than the assessment ones (Fig.2) thus having smaller chances of being corrupted.

As a possible extension, the connectivity matrix, together with the computed PMFs, could be used as an input to a

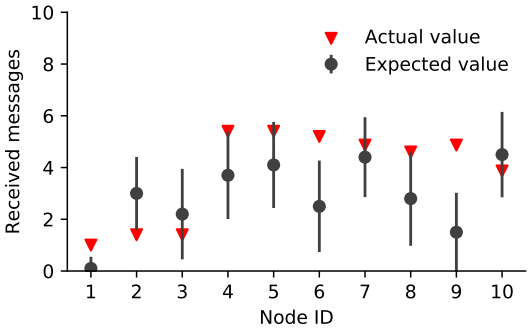


Fig. 6. Expected and actual number of received messages. The error bars correspond to $1.5\,\sigma$.

Bayesian reasoner in order to diagnose issues in the network and detect malfunctioning or poorly positioned nodes. Moreover, the proposed scheme could also be used to programmatically tune transmission power control parameters and compute the Expected Transmission Count between nodes during the network lifetime.

Links characteristics can vary significantly over time, and the quality assessment is an expensive operation with a required time that is quadratically proportional to the number of nodes. The application used to test the quality of the PRR estimation then can also be used to ascertain whether the previous estimate is still valid or a new execution is needed.

## V. Conclusions

In this paper we proposed a network quality assessment scheme rooted in a symbolic programming paradigm based on the exchange of executable code among nodes. This approach enables complex network monitoring operations on resource-constrained devices without the need for any additional code stored on the nodes. This goal is achieved by sending out messages containing appropriate instructions at runtime.

Future work will consider extending the proposed scheme to multi-hop topologies, using the collected data as an input for a diagnostic Bayesian reasoner, and assessing the impact of the proposed scheme on node power consumption.

## References


[1] A. Ali, Y. Ming, S. Chakraborty, and S. Iram, "A comprehensive survey on real-time applications of WSN," *Future internet*, vol. 9, no. 4, p. 77, 2017.

[2] J. Hughes, P. Lazaridis, I. Glover, and A. Ball, "An empirical study of link quality assessment in wireless sensor networks applicable to transmission power control protocols," *IET Conference Proceedings*, January 2017. [Online]. Available: https://doi.org/10.1049/cp.2017.0274

[3] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "An empirical study of low-power wireless," *ACM Transactions on Sensor Networks (TOSN)*, vol. 6, no. 2, pp. 1–49, 2010.

[4] A. Schoofs, G. O'Hare, and A. Ruzzelli, "Debugging Low-Power and Lossy Wireless Networks: A Survey," *IEEE Communications Surveys Tutorials*, vol. 14, no. 2, pp. 311–321, 2012.

[5] I. Das, R. N. Shaw, and S. Das, "Analysis of Energy Consumption of Energy Models in Wireless Sensor Networks," in *Innovations in Electrical and Electronic Engineering*, M. N. Favorskaya, S. Mekhilef, R. K. Pandey, and N. Singh, Eds. Singapore: Springer Singapore, 2021, pp. 755–764.

[6] R. Kotian, G. Exarchakos, S. Stavros, and A. Liotta, "Impact of Transmission Power Control in multi-hop networks," *Future Generation Computer Systems*, vol. 75, pp. 94–107, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X16303910

[7] V. N. Medeiros, P. C. G. de Brito, B. Silvestre, and V. da CM Borges, "Rall: Routing-aware of path length, link quality and traffic load for wireless sensor networks," in *Proceedings of the Symposium on Applied Computing*, 2017, pp. 594–601.

[8] J. Brown, U. Roedig, C. A. Boano, and K. Römer, "Estimating packet reception rate in noisy environments," in *39th Annual IEEE Conference on Local Computer Networks Workshops*, 2014, pp. 583–591.

[9] W. Liu, Y. Xia, R. Luo, and S. Hu, "Lightweight, Fluctuation Insensitive Multi-Parameter Fusion Link Quality Estimation for Wireless Sensor Networks," *IEEE Access*, vol. 8, pp. 28 496–28 511, 2020.

[10] T. Attia, M. Heusse, B. Tourancheau, and A. Duda, "Experimental Characterization of LoRaWAN Link Quality," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[11] M. Chincoli and A. Liotta, "Self-Learning Power Control in Wireless Sensor Networks," *Sensors*, vol. 18, no. 2, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/2/375

[12] S. Gaglio, G. Lo Re, G. Martorella, and D. Peri, "A Lightweight Network Discovery Algorithm for Resource-constrained IoT Devices," in *2019 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2019, pp. 355–359.

[13] A. Augello, R. D'Antoni, S. Gaglio, G. Lo Re, G. Martorella, and D. Peri, "Verification of Symbolic Distributed Protocols for Networked Embedded Devices," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1177–1180.

[14] S. Gaglio, G. Lo Re, G. Martorella, and D. Peri, "High-level programming and symbolic reasoning on IoT resource constrained devices," *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, vol. 150, pp. 58–63, 2015.

[15] A. Yaqoob, M. A. Ashraf, F. Ferooz, A. H. Butt, and Y. D. Khan, "WSN Operating Systems for Internet of Things (IoT): A Survey," in *2019 International Conference on Innovative Computing (ICIC)*. IEEE, 2019, pp. 1–7.

[16] K. Lehniger, S. Weidling, and M. Schölzel, "Heuristic for page-based incremental reprogramming of wireless sensor nodes," in *2018 IEEE 21st International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, 2018, pp. 61–66.

[17] K. Grunert, "Overview of JavaScript Engines for Resource-Constrained Microcontrollers," in *2020 5th International Conference on Smart and Sustainable Technologies (SpliTech)*, 2020, pp. 1–7.

[18] N. Reijers and C.-S. Shih, "Improved ahead-of-time compilation of stack-based JVM bytecode on resource-constrained devices," *ACM Transactions on Sensor Networks (TOSN)*, vol. 15, no. 3, pp. 1–44, 2019.

[19] S. Gaglio, G. Lo Re, G. Martorella, and D. Peri, "DC4CD: A Platform for Distributed Computing on Constrained Devices," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 1, pp. 27:1–27:25, Dec. 2017. [Online]. Available: http://doi.acm.org/10.1145/3105923

[20] D. M. Hanna, B. Jones, L. Lorenz, and S. Porthun, "An embedded Forth core with floating point and branch prediction," in *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2013, pp. 1055–1058.

[21] J. S. Furter and P. C. Hauser, "Interactive control of purpose built analytical instruments with Forth on microcontrollers - A tutorial," *Analytica Chimica Acta*, vol. 1058, pp. 18–28, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003267018313278

[22] S. Gaglio, G. Lo Re, G. Martorella, and D. Peri, "A Fast and Interactive Approach to Application Development on Wireless Sensor and Actuator Networks," in *Emerging Technology and Factory Automation (ETFA), 2014 IEEE*, Sept 2014, pp. 1–8.

[23] L. Sang, A. Arora, and H. Zhang, "On Link Asymmetry and One-Way Estimation in Wireless Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 6, no. 2, Mar. 2010. [Online]. Available: https://doi.org/10.1145/1689239.1689242

[24] Y. Hong, "On computing the distribution function for the Poisson binomial distribution," *Computational Statistics & Data Analysis*, vol. 59, pp. 41–51, Mar 2013.