D-FLER: A Distributed Fuzzy Logic Engine for Rule-based Wireless Sensor Networks

Mihai Marin-Perianu and Paul Havinga

University of Twente, Enschede, The Netherlands
{m.marinperianu, p.j.m.havinga}@utwente.nl

Abstract. We propose D-FLER, a distributed, general-purpose reasoning engine for WSN. D-FLER uses fuzzy logic for fusing individual and neighborhood observations, in order to produce a more accurate and reliable result. Thorough simulation, we evaluate D-FLER in a fire-detection scenario, using both fire and non-fire input data. D-FLER achieves better detection times, while reducing the false alarm rate. In addition, we implement D-FLER on real sensor nodes and analyze the memory overhead, the numerical accuracy and the execution time.

1 Introduction

Initially perceived as a cost-effective method for monitoring large geographical areas in detail, Wireless Sensor Networks (WSN) exhibit today a rapidly increasing uptake in various industrial and business-related fields [20]. There are two main reasons for this trend. First, the comprehensive set of features (digital I/O, storage and processing, wireless communication) available on current sensor nodes makes the term "intelligent sensor" real to the industrial automation community. Second, WSN are self-organizing, collaborative networks, capable of executing logic in a distributed way, at the point of action. This is a desirable property because it decreases the load on the back-end system [12] and improves the overall reliability and responsiveness. There are, however, several important challenges in this regard. The sensor nodes have limited resources and usually execute only simple logic, such as concise *business rules* [20]. This may lead to erroneous decisions in the presence of inaccurate or faulty sensors. Sensor data fusion techniques can reduce the errors, but also require complex computations that sensor nodes cannot handle, or prior information that is infeasible to obtain practically [15].

In this paper, we explore fuzzy logic as an alternative solution to these problems. The field of fuzzy logic has been developing for more than forty years, with many successful applications in diverse areas as automotive industry, artificial intelligence, medicine, behavioral science, just to mention some [11]. Fuzzy inference systems (FIS) match two of the most challenging requirements of WSN: (1) they are simple and can be executed on limited hardware, and (2) tolerate imprecise, unreliable data. In addition, FIS have several properties that are less mentioned in the WSN literature, but are equally important from a practical point of view. First, fuzzy logic can reduce the development time compared

with other techniques. In Bayesian calculus for example, prior probabilities need to be acquired by means of a statistical analysis requiring massive amounts of data [15]. With fuzzy logic, it is possible to have a running system by using only an intuitive, common-sense description of the problem. Second, fuzzy logic is flexible; it can be built on top of the expert knowledge, mixed with conventional control methods and easy to add or change functionality. Third, FIS are computationally fast [15], which is important because the processing capabilities of sensor nodes are limited. Fourth, FIS can be implemented with little memory overhead, which is a desirable property in WSN because of (1) the limited memory on the sensor nodes and (2) the latency of the network reprogramming. Finally, FIS, potentially combined with neural networks, are adaptive, i.e. can be trained with examples or can learn at runtime from feedback.

To the extent of our knowledge, there is little research effort in the field of WSN that involves fuzzy logic. In this paper, we make a step forward and propose a lightweight, distributed fuzzy logic engine operating with simple IF-THEN rules applied over both the individual sensor readings and the neighborhood observations. We choose event detection (more precisely, fire detection) as an application example to evaluate our approach compared with conventional crisp logic methods. We believe that many other practical WSN applications can benefit from using a distributed inference engine, capable of fusing multi-sensor, multi-node unreliable information, in order to produce a more reliable result.

2 Related Work

Previous research has also considered the idea of reasoning engines for embedded devices. Cooperative Artefacts [23], for example, can autonomously reason about their situation by means of an inference engine based on a Prolog interpreter. The functionality of the interpreter is however limited, since it has to run on resource-poor sensor nodes. An application specific virtual machine, such as Maté [19], offers more flexibility for programming WSN, but at the price of significant overhead. The ubiquitous chip [14] is an event-driven I/O control device based on ECA (Event, Condition, Action) rules. A similar model is used by the business rules for sensor nodes [20], which are designed to express service-oriented business logic in a compact way. Compared to these examples, a fuzzy logic inference engine has the advantage of preserving the simplicity of rule-based logic, while handling unreliable and imprecise numeric information.

3 D-FLER

In this section we describe D-FLER, a distributed rule-based fuzzy logic engine designed for collaborative WSN. We start with a short overview of general fuzzy logic systems and than present the detailed design of D-FLER.

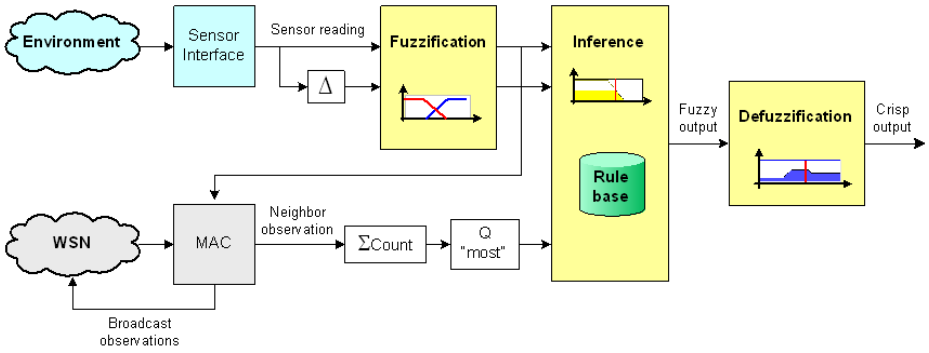


Fig. 1. D-FLER structure.

3.1 Overview of Fuzzy Logic

Fuzzy logic systems are in general non-linear input-output mappings [21]. FIS operate with fuzzy sets, which extend the ordinary notion of crisp sets. A fuzzy set F is characterized by a membership function $\mu_F(x)$, which gives the degree of similarity of x to F. An important consequence is that FIS are universal approximators, capable of approximating any continuous function with an arbitrary bound ϵ. In engineering, the most widely used are the rule-based FIS. These systems are composed of four basic components. First, the *fuzzifier* maps crisp inputs into fuzzy sets by using the membership functions. Second, the fuzzified values activate the *rules*, which are provided by experts or extracted from numerical data. The rules are expressed as a collection of IF-THEN statements, having fuzzy propositions as antecedents and consequences. Third, the *fuzzy inference engine* combines the rules to obtain an aggregated fuzzy output. Finally, the *defuzzifier* maps the fuzzy output back to a crisp number that can be used for making decisions or control actions.

3.2 D-FLER Design

WSN usually try to compensate the resource limitations and the lack of reliability through cooperative algorithms, which exploit the high density of nodes deployed. This is why FIS for WSN have to go distributed and embed collaborative mechanisms of reasoning on the observed data and taking decisions or actions in a coordinated manner.

Following this idea, D-FLER uses two types of inputs: individual observations (sensor readings of the current node) and neighborhood observations (fuzzified sensor data from the neighboring nodes). Fig. 1 illustrates how D-FLER fuses these inputs. We distinguish the following main operations:

1. *Fuzzification of individual observations.* D-FLER obtains the sensor readings from the sensor interface of the current node. Both the sensor raw values and

their differential variations Δ are fuzzified through the predefined membership functions. The fuzzified values are scheduled for being broadcast within the local neighborhood by the MAC layer.

2. *Quantification of neighborhood observations.* When receiving neighborhood observations over the radio, the MAC layer forwards them to D-FLER. Then, D-FLER updates the *sigma-count* factor [25], which is formally defined as

$$\sum Count(F) = \sum_i \mu_F(x_i) \tag{1}$$

where $X = \{x_1, ..., x_n\}$ is the set of neighbors and F is a property of interest related to their observations, e.g. "smoke level is high".
Optionally, the neighborhood observations can be given weighting factors based on the confidence of each neighbor (e.g. given by the precision of its sensors, distance from the event observed, past accuracy computed as false alarm and rejection rates, etc.). In this case, we have a *weighted sigma-count*

$$\sum Count(F; w) = \sum_i w_i \mu_F(x_i) \tag{2}$$

The neighborhood observations are eventually characterized through a fuzzy majority *quantifier*, such as *most* [18]

$$\mu_{most}(\frac{\sum Count(F)}{|X|}) = \mu_{most}(\frac{\sum_i \mu_F(x_i)}{n}) \tag{3}$$

where

$$\mu_{most}(x) = \begin{cases} 0 & \text{if } x \leq 0.3; \\ 2x - 0.6 & \text{if } 0.3 < x < 0.8; \\ 1 & \text{if } x \geq 0.8. \end{cases} \tag{4}$$

The *most* quantifier gives a fuzzified indication of the consensual neighborhood opinion that the current node can add to its own observations, in order to take a more accurate decision. However, the decision of the node is not fed back into the neighborhood, in order to avoid an artificial increase of confidence due to ping-pong effects.

3. *Inference.* In addition to conventional fuzzy inference, the D-FLER rules incorporate both the fuzzified individual observations and the quantified neighborhood observations. Such a "distributed" rule has the following structure

$$\begin{aligned} \text{IF } & s_1 \text{ is } F_{i_1} \text{ AND } s_2 \text{ is } F_{i_2} \text{ AND } ... \ s_p \text{ is } F_{i_p} \text{ AND} \\ & Q\ n_1 \text{ is } F_{j_1} \text{ AND } Q\ n_2 \text{ is } F_{j_2} \text{ AND... } Q\ n_q \text{ is } F_{j_q} \\ \text{THEN } & o \text{ is } G \end{aligned} \tag{5}$$

where s_i are fuzzified sensor readings, n_j are neighborhood observations, Q is the majority quantifier, o is the output, F_{i_k,j_l} and G are input and output fuzzy sets, respectively.
Considering the fire detection example (see Sec. 4), a rule can be written as

$$\begin{aligned} \text{IF } & Smoke \text{ is } High \text{ AND } Temp \text{ is } Low \text{ AND} \\ & most(SmokeNeigh) \text{ is } High \text{ AND } most(TempNeigh) \text{ is } High \\ \text{THEN } & FireDecision \text{ is } High \end{aligned} \tag{6}$$

During the inference process, several rules are activated and contribute to the combined fuzzy output. Optionally, the combination of rules can be weighted according to the degree of belief to each rule, if such information is available.

4. *Defuzzification*. The last phase, defuzzification, produces a crisp output computed using one of the common methods: maximum, mean of maxima, centroid, etc.

In addition to these operations, D-FLER can be trained and can learn at runtime if feedback is available. To achieve this, two parameters can be adjusted:

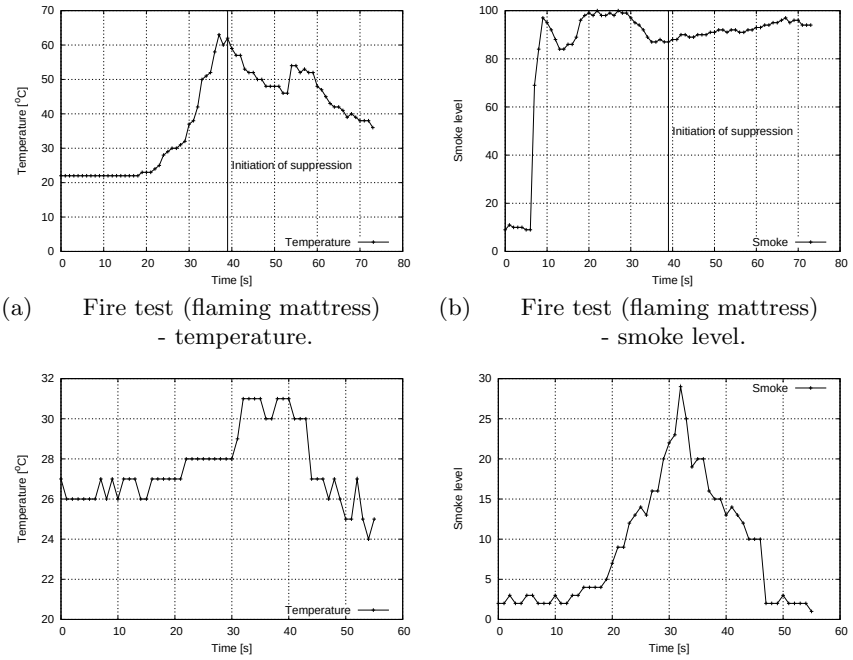
- The confidence of each neighbor in the *weighted sigma-count* factor. For example, nodes that constantly report observations in contrast with the right decision will receive a lower weight in the result of the quantification.
- The importance of each rule in the inference process. For example, after training, the rules contributing to correct decisions will be assigned a larger weight in forming the combined fuzzy output.

4 Application Example - Fire Detection

We propose large-scale fire detection as an application example for evaluating D-FLER. The potential usage of WSN for real-time fire detection and firefighting assistance is currently under investigation in several research projects [2, 3]. Advanced fire detection algorithms consider distributed sensing as an alternative for improving time to alarm over single-station detectors [6]. In addition, comprehensive experiments show that the use of combined sensors [9] (e.g. smoke and CO) can significantly reduce the false alarm rate, while increasing sensitivity (i.e., decreasing the detection time for real fires). Since fuzzy logic systems can fuse naturally multi-sensor data, they appear as a promising solution for robust fire detection algorithms [7].

In this paper, we show that a distributed FIS, running within the WSN, can improve the overall detection time and reliability, while providing better coverage for monitoring large hazardous areas. We utilize as input data the fire tests carried on by Bukowski et al. [13] to evaluate the performance of modern residential alarms. Both the fire and non-fire (i.e., nuisance alarms) test data is publicly available on the NIST website and well documented. We consider four representative tests: two fire scenarios (flaming mattress and flaming chair) and two nuisance scenarios (fried hamburgers and toasted bagel halves). For the fire scenarios, we follow the temperature and smoke data for approximately one minute after the moment of ignition. Similarly, for the nuisance scenarios, we follow the temperature and smoke data for approximately one minute around the time when the smoke alarms used in the tests reach medium alarm thresholds.

In Fig. 2 we plot two example data sets for a fire and a nuisance scenario, respectively. Fig. 2(a) and 2(b) show the temperature and smoke data for the first fire test used in our simulations (test SDC05 - flaming mattress in bedroom [13]). The ignition takes place at time 0. We notice that the temperature increases much slower than the smoke level, which raises abruptly after 5s and reaches



(a) Fire test (flaming mattress) - temperature.

(b) Fire test (flaming mattress) - smoke level.

(c) Nuisance test (toasted bagel) - temperature.

(d) Nuisance test (toasted bagel) - smoke level.

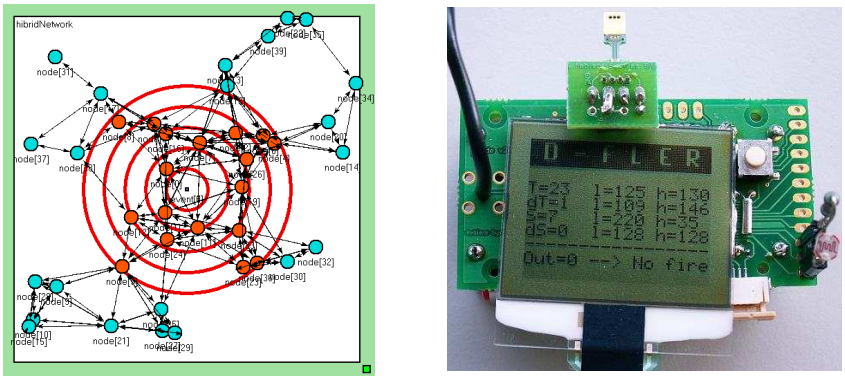
Fig. 2. Example of fire and nuisance test data.

a maximum at 9s. However, the temperature has also to be monitored for a reliable detection because smoke can also occur in the case of a failed ignition. The temperature starts to raise quickly after 29s and reaches a maximum at 38s, right before the initiation of suppression.

Fig. 2(c) and 2(d) show the temperature and smoke sample data for the second nuisance test used in our simulations (test MHN20 - two frozen bagel halves toasted [13]). During this test, the frozen bagels were toasted to a medium brown color and did not char significantly. We notice an approximately 5°C rise in temperature, with a peak between 32s and 40s. Similarly, the smoke level increases to a peak level until 32s. After stopping the toaster, the conditions come back to normal at approximately 48s.

The sensor nodes are assumed to sample the temperature and smoke values under two simple fire models:

– *Basic model.* All the nodes deployed on the area measure the same values at the same time, with the variations introduced by their different accuracy and measurement errors.



(a) OMNeT++ simulation. (b) μNode platform.

Fig. 3. D-FLER simulation and prototype.

– *Radial model*. The fire spreads from a central point in circular rings with a specified speed. The nodes therefore measure the fire parameters shifted in time, according to their position on the area.

In case of fire, it is desirable that the large majority of nodes within the event radius report fire in the shortest possible time. Likewise, in the presence of nuisance conditions, a correct behavior minimizes the number of false alarms. Therefore, we take the percentage of nodes agreeing upon the real event status (fire/non-fire) as a measure of reliability, and the time within which the percentage converges to 100% as a measure of responsiveness.

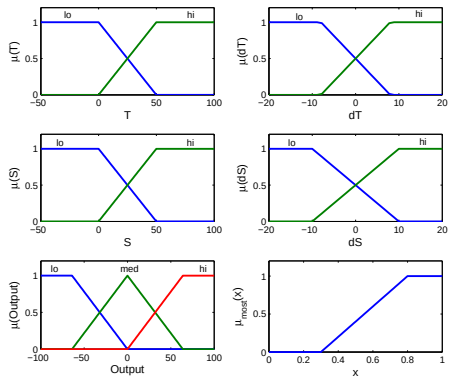
We are now ready to introduce our simulation model and present the performance results.

5 Simulation

5.1 Simulation setting

We consider a random deployment of 100 nodes within a rectangular area of 500m x 500m. The radio range is set to 150m. The nodes are considered to be randomly equipped with temperature and/or smoke sensors. The temperature sensor data is expressed in Celsius degrees. The smoke sensor data is derived from the ionization smoke alarm analog output used in the NIST tests, and scaled to 100. The accuracy of the sensors is modeled according to the characteristics of a real sensor [4], by taking into account the linearity, offset and gain errors, plus Gaussian white noise. The overall accuracy lies in the interval ±1–4%.

We implement the following four detection methods in the OMNet++ simulation environment [5] (see Fig. 3(a) for a graphical image of the simulation setting):



(a) Membership functions and μ_{most}.

Rule	T	dT	S	dS	most(T_n)	most(S_n)	Out
1	LO	LO	LO	LO	LO	LO	**LO**
2	LO	LO	LO	LO	LO	HI	**LO**
3	LO	LO	LO	LO	HI	LO	**LO**
4	LO	LO	LO	LO	HI	HI	**LO**
5	LO	LO	LO	HI	LO	LO	**LO**
6	LO	LO	LO	HI	LO	HI	**MED**
7	LO	LO	LO	HI	HI	LO	**MED**
8	LO	LO	LO	HI	HI	HI	**MED**
9	LO	LO	HI	LO	LO	LO	**LO**
10	LO	LO	HI	LO	LO	HI	**LO**
11	LO	LO	HI	LO	HI	LO	**MED**
12	LO	LO	HI	LO	HI	HI	**MED**
13	LO	LO	HI	HI	LO	LO	**MED**
14	LO	LO	HI	HI	LO	HI	**MED**
15	LO	LO	HI	HI	HI	LO	**HI**
16	LO	LO	HI	HI	HI	HI	**HI**
...	...	...	...	...	...	...	...

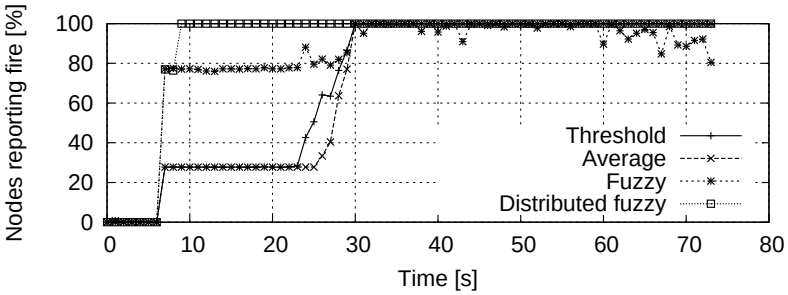
(b) Rule set (partial).

Fig. 4. Fuzzy-logic fire detection engine.

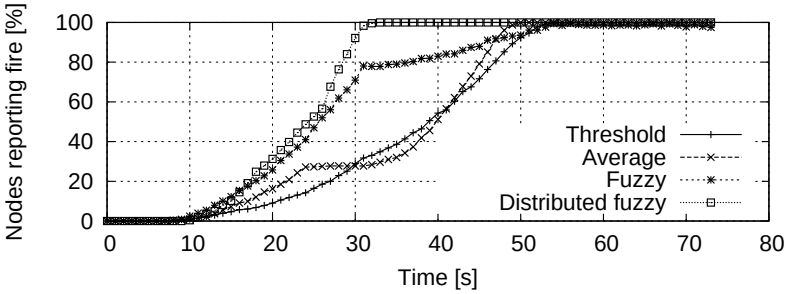
1. *Threshold*. Each node decides that a fire occurred when the temperature and smoke values exceed predefined thresholds. The thresholds correspond to the activation time of the medium alarm level in the NIST fire tests.
2. *Average*. A simple distributed mechanism is implemented, where each node takes the mean value between its own samples and the average of the readings reported by its neighbors. The decision it made according to the same thresholds as in the previous case. This method is expected to decrease the false alarm rate in the case of nuisance scenarios, by reducing the effect of individual sensor errors.
3. *Fuzzy*. Each node uses a local fuzzy logic engine. If the node has both temperature and smoke sensors, then four inputs are analyzed: T - temperature, S - smoke level, dT - temperature change (current sample minus previous sample) and dS - smoke level change (current sample minus previous sample). For the nodes with only one sensor, the fuzzy logic engine takes two inputs (T and dT or S and dS). The membership functions are shown in Fig. 4(a). Similar to the first method, no information is exchanged among the nodes.
4. *D-FLER*. As described in Sec. 3, the nodes broadcast their fuzzified temperature and smoke values within the local neighborhood. D-FLER operates with both the inputs mentioned in the previous method (T, S, dT and dS) and the quantified neighborhood observations. The *most* quantifier defined in Eq. 4 is used (see Fig. 4(a)). The rules have the format described in Eq. 5 and 6 and are partially shown in Fig. 4(b).

5.2 Results

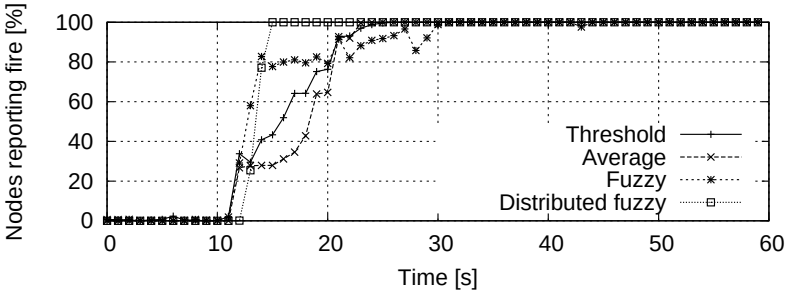
The four data sets (fire and nonfire) are input to the four simulated detection methods. We collect the decisions of the nodes at every time step. The results



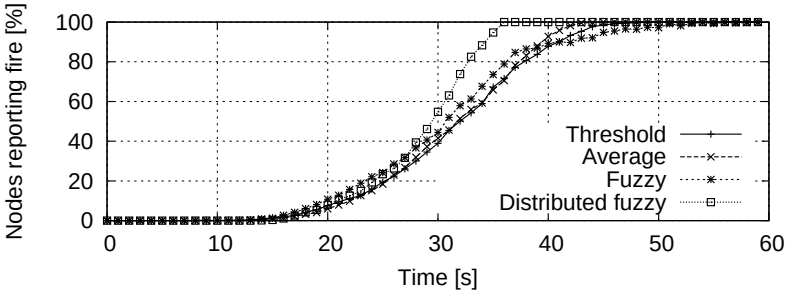
(a) First fire test - basic model.



(b) First fire test - radial model.

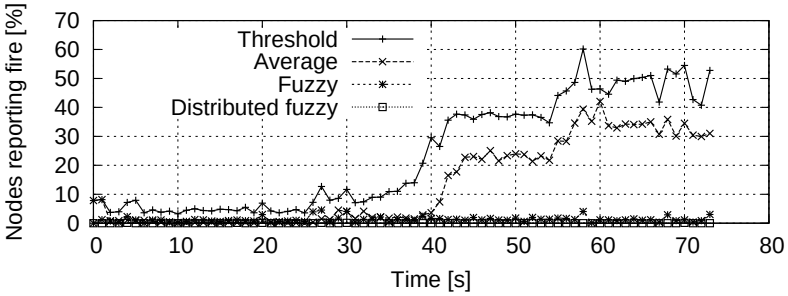


(c) Second fire test - basic model.

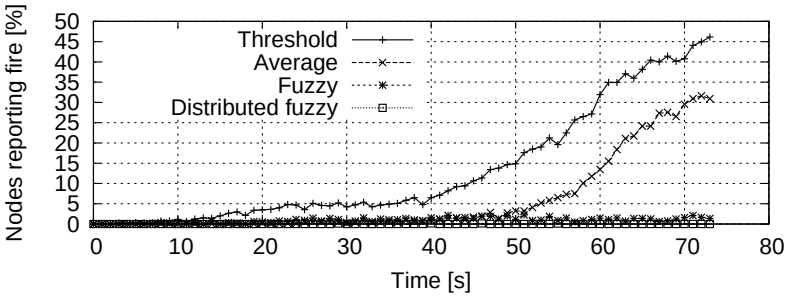


(d) Second fire test - radial model.

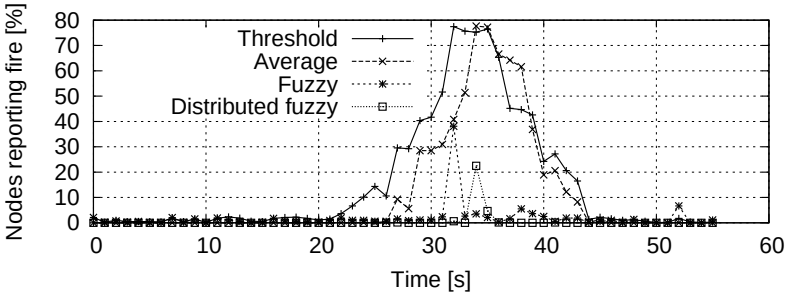
Fig. 5. Simulation results of fire tests.



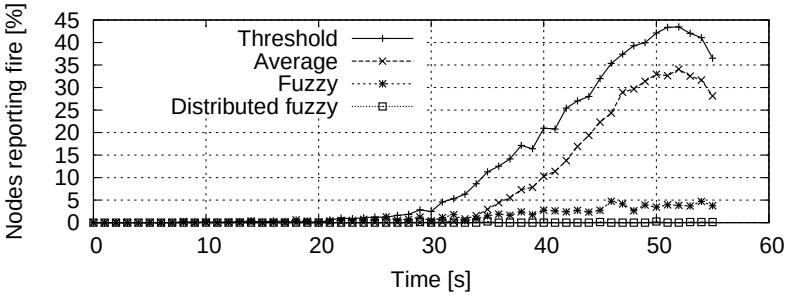
(a) First nuisance test - basic model.



(b) First nuisance test - radial model.



(c) Second nuisance test - basic model.



(d) Second nuisance test - radial model.

Fig. 6. Simulation results of nuisance tests.

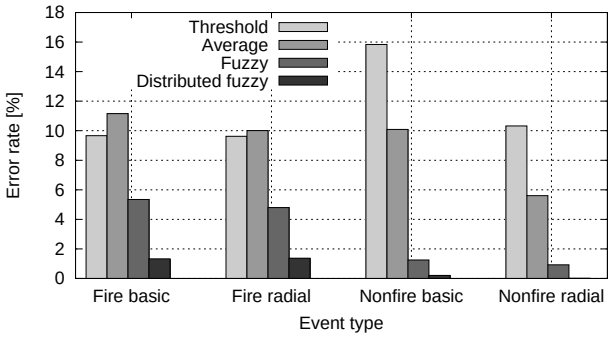


Fig. 7. Summary of simulation results - error rates of the four methods.

presented in Fig. 5 and 6 are averaged over 10 simulation runs with different random topologies.

In the case of fire, we are interested in a rapid and reliable detection. The more nodes detect the fire, the more reliable is the decision on the average case and the information about the event can be transported faster toward the sink or gateway nodes. Fig. 5 plots the percentage of nodes reporting fire during the two fire tests, considering both the basic and the radial spread model. We can make the following observations:

- D-FLER achieves the shortest detection times both for the basic and radial fire models, followed by the individual fuzzy engine. This indicates that fuzzy logic can model with more granularity the event of interest and provide a more robust inference result.
- The individual fuzzy engine has a few decision oscillations (especially noticeable in Fig. 5(a)), while D-FLER has a clear transition from non-fire to fire state in all the situations. This shows that the individual reasoning method is more sensitive to sensor errors than the distributed one.
- The average method is slightly slower to converge to 100% nodes reporting fire than the direct threshold method because of the slow temperature raise and the influence of the neighbors samples.

In the case of nuisance tests, we are interested in reducing the false alarm rate. Fig. 6 presents the simulation results of the two nuisance scenarios. We can make the following observations:

- The average method reduces the false alarm rate compared to the direct threshold method, proving to be less sensitive to individual sensor errors.
- The fuzzy logic-based methods are clearly more robust to nuisance conditions, with D-FLER approaching to 0% erroneous nodes.

Fig. 7 gives a summarized view of all the simulation results. The error rate is computed as the percentage of erroneous decisions, over all the nodes and the

entire simulation duration. The fuzzy logic methods prove to be more reliable than the threshold solutions, D-FLER having an average error rate less than 2% in the case of fire and approaching 0% in the case of nuisance tests.

6 Implementation

We implemented D-FLER on the Ambient μNode 2.0 platform [1] (see Fig. 3(b)), which is based on the MSP430 micro-controller. The μNodes run AmbientRT [16], a real-time multitasking operating system based on publish/subscribe inter-task communication. The publish/subscribe model simplifies the implementation, as D-FLER can abstract from various types of inputs by subscribing only to their data. Similar to our previous work [20], the reasoning engine is decoupled from the *drivers* (custom interfaces giving the sensor readings or radio protocol stack delivering the neighbors messages) that provide the input data. The system is therefore modular and easy to reconfigure over the air.

The following list summarizes the implementation details of the D-FLER components (see also Sec. 3):

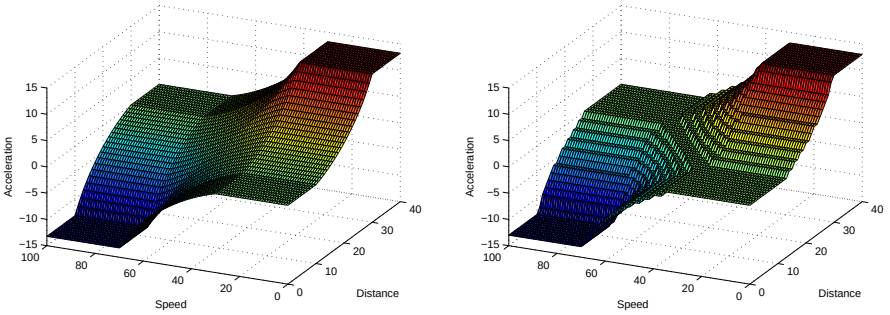
1. *Fuzzification.* The local sensor readings are fuzzified according to the user-specified membership functions. D-FLER currently accepts only triangular membership functions. Our implementation of the fuzzification is *computational oriented* [8] due to the small amount of RAM available on the node. To reduce the computational complexity, the maximum fuzzified value is scaled to a power of 2, as recommended by Dannenberg [10].
2. *Quantification of neighborhood observations.* The data from the neighboring nodes is processed through the *sigma-count* factor and μ_{most} operator (see Eq. 1 - 4).
3. *Inference.* The rules are formatted as in Eq. 5, taking into account all the fuzzified inputs. D-FLER uses *max-min* inference for computational simplicity.
4. *Defuzzification.* The aggregated result of the rule evaluation is defuzzified using the *centroid* method.

We evaluate the D-FLER implementation by following three properties of interest: the memory overhead, the numerical accuracy and the execution time.

The code memory footprint amounts to ≈1kB FLASH memory (out of 48kB available), leaving thus enough space for the OS kernel, sensor drivers, network stack etc. In addition, D-FLER occupies 20 bytes RAM for static variables and allocates dynamically heap space for the inputs, outputs and rules. To estimate the memory consumption M at runtime, we use the following formula

$$M = I(4m_i + 1) + NIm_i + 2R(Im_i + 1) + O(4m_o + 1) \tag{7}$$

where I is the number of inputs, each input having m_i membership functions, N is the number of neighbors providing same type of inputs (fuzzified), R is the number of rules and O is the number of outputs, each output having m_o



(a) Matlab. (b) D-FLER.

Fig. 8. The car control problem.

membership functions. An important optimization can be made if every input has two membership functions ($m_i = 2$). In this case, the rules can be represented only as their consequence parts ("o is G" from Eq. 5). Then, the binary representation of the rule index gives the combination of inputs and fuzzy input sets in the antecedents, e.g. rule number 5 (binary 101) means "i_0 is F_1^0 AND i_1 is F_0^1 AND i_2 is F_1^2". Using this optimization, a running instance of D-FLER with 5 inputs and 2 outputs allocates 326 bytes RAM out of 10kB available.

D-FLER performs the computations on two-byte signed integers. In order to evaluate the precision error, we run a simple car control problem [24] with two inputs (distance and speed) and one output (acceleration). Fig. 8(a) and 8(b) plot the rule surfaces generated with Matlab and D-FLER, respectively, where the latter is obtained by iterating through the whole input space with unit step. The absolute error of D-FLER (due to integer approximation) is 1% on average, with a maximum of 3.33%.

D-FLER execution time is depicted in Fig. 9 (at logarithmic scale) for different problem complexities, given by the number of inputs, outputs and rules (the number of membership functions is fixed at $m_i = 2$ and $m_o = 3$). For each case, the execution time is computed by averaging over the entire input space, as in the previous example. We can make the following observations:

- The fuzzification and inference times are in the same range, varying approximately linearly with the number of inputs and rules, respectively. The fuzzification operation takes between 180 and 360μs, while the inference process takes between 80μs and 1.13ms.
- The defuzzification is clearly the most time-consuming operation (between 6.56 and 25.86ms) because of the computation involved in calculating the centroid of the output area. There are several alternatives to reduce the defuzzification time: (1) use rectangular output membership functions [10], (2) use a fast centroid approximation method [22] or (3) use a simpler defuzzification method, e.g. maximum.

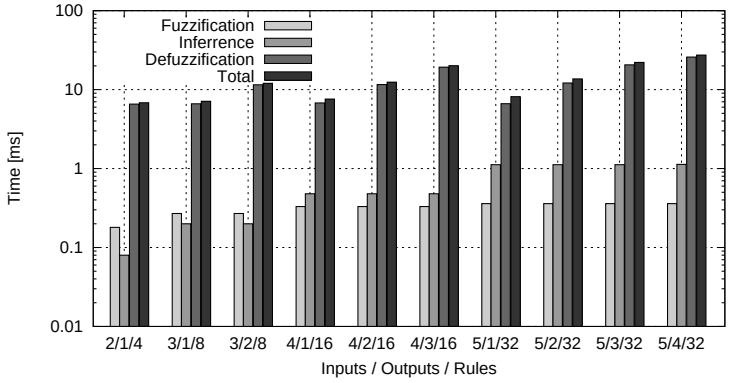


Fig. 9. Performance of D-FLER on μNode platform.

– The total execution time is basically given by the defuzzification speed and, consequently, varies linearly with the number of outputs.

7 Discussion

In this section, we briefly discuss the impact of several factors on the performance of D-FLER, pointing out the main advantages and limitations.

Communication. Exchanging the observed data within the one-hop neighborhood increases the communication overhead and, consequently, the energy consumption. On average, each node would send one packet and receive $N\pi r^2/A$ packets per time step, where N is the total number of nodes, A is the area size and r is the radio range. Reducing the duty cycle alleviates the problem, but at the price of increasing the detection time. A better approach is to use cross-layer integration and piggyback the data to the periodic heartbeat messages of the MAC or routing protocols.

Sensing errors. In practical deployments, cheap sensors with low precision may be used due to cost concerns. Likewise, the phenomenon to be detected (e.g. fire) may itself introduce considerable sensing errors. We repeated the simulations from Sec. 5.1 considering sensors with two times lower accuracy (2-8%). The results are shown in Fig. 10. In the case of fire, there is little increase in the detection error rate ($< 0.3\%$). However, for the nuissance scenarios, the false alarm rate grows with up to 3.7% for all the methods except D-FLER, which stays lower the 0.5%.

Computation. The fuzzification using triangular or trapezoidal membership functions and the max-min inference prove computationally fast in our implementation. However, the usage of other membership functions, such as Gaussian, would impose a memory-oriented implementation [8], less suitable for the low

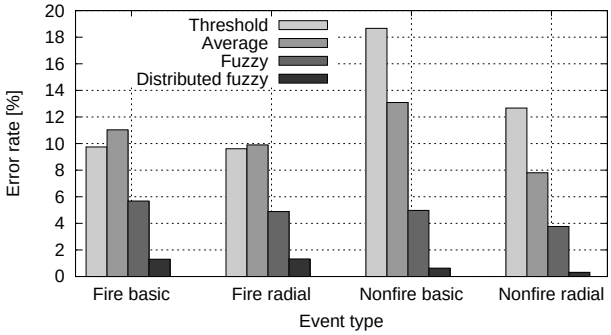


Fig. 10. Error rates for sensors with lower accuracy.

amount of RAM typically available on sensor nodes. The centroid-based defuzzification introduces the highest latency and should be optimized or replaced with a simpler method if execution times less than 6ms are needed.

Training and learning. In order to produce optimal results, tedious tuning of the membership functions may be required. In practice, standard methods such as ANFIS [17] and fuzzy clustering are used to build a fuzzy model based on both human knowledge and stipulated input-output data pairs. Since D-FLER is a distributed solution, learning at runtime about the confidence of the neighborhood observations would also be a valuable feature in real-life deployments.

8 Conclusions

In this paper, we described D-FLER, a distributed fuzzy logic reasoning engine for WSN. By combining individual sensor inputs with neighborhood observations, D-FLER produces more accurate results and is more robust to sensor errors. For performance evaluation, we use fire detection as an application scenario. The simulation results show that distributed fuzzy logic is a promising alternative for event detection with WSN, as it improves the detection time, while reducing the false alarm rate. From the implementation point of view, D-FLER proves effective and feasible to run on resource-constrained sensor nodes. As future work, we want to study the impact of WSN communication characteristics (low duty cycle, multihop, high error rates) on the performance of D-FLER. In addition, we plan to conduct an application trial in a real setting for firefighting assistance and distributed coordination, as part of the AWARE project [2].

References

1. Ambient Systems. http://www.ambient-systems.net.
2. AWARE project. http://grvc.us.es/aware.

3. Fire Information and Rescue Equipment (FIRE). http://fire.me.berkeley.edu.
4. National Semiconductor LM92 temperature sensor. http://www.national.com.
5. OMNeT++. http://www.omnetpp.org.
6. T. Cleary and K. Notarianni. Distributed sensor fire detection. In *International Conference on Automatic Fire Detection*, 2001.
7. T. Cleary and T. Ono. Enhanced residential fire detection by combining smoke and co sensors. In *International Conference on Automatic Fire Detection*, 2001.
8. A. Costa, A. De Gloria, F. Giudici, and M. Olivieri. Fuzzy logic microcontroller. *IEEE Micro*, 17(1):66–74, 1997.
9. R. J. Roby D. T. Gottuk, M. J. Peatross and C. L. Beyler. Advanced fire detection using multi-signature alarm algorithms. *Fire Safety Journal*, 37:381–394, 2001.
10. A. Dannenberg. Fuzzy logic motor control with msp430x14x. Technical Report SLAA235, Texas Instruments, 2005.
11. J. Espinosa, J. Vandewalle, and V. Wertz. *Fuzzy logic, identification and predictive control*. Springer-Verlag, 2004.
12. M. Marin-Perianu et al. Decentralized enterprise systems: A multi-platform wireless sensor networks approach. Technical Report TR-CTIT-07-31, CTIT, University of Twente, 2007.
13. R. W. Bukowski et al. Performance of home smoke alarms. Technical Report 1455, NIST, 2004.
14. T. Terada et al. Ubiquitous chip: A rule-based i/o control device for ubiquitous computing. In *Pervasive*, pages 238–253, 2004.
15. S. J. Henkind and M.C. Harrison. An analysis of four uncertainty calculi. *IEEE Transactions on Systems, Man and Cybernetics*, 18(5):700–714, 1988.
16. T. Hofmeijer, S. Dulman, P. G. Jansen, and P. J. M. Havinga. AmbientRT - real time system software support for data centric sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 61–66, 2004.
17. J. S. Roger Jang. Anfis: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:665–684, 1993.
18. J. Kacprzyk. Group decision making with a fuzzy linguistic majority. *Fuzzy Sets and Systems*, 18(2):105–118, 1986.
19. P. Levis and D. Culler. Maté: a tiny virtual machine for sensor networks. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 85–95, 2002.
20. M. Marin-Perianu, T.J. Hofmeijer, and P. J. M.Havinga. Implementing business rules on sensor nodes. In *11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 292–299, 2006.
21. J. M. Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83:345–377, 1995.
22. T. A. Runkler and M. Glesner. Decade - fast centroid approximation defuzzification for real time fuzzy control applications. In *ACM Symposium on Applied Computing (SAC '94)*, pages 161–165, 1994.
23. M. Strohbach, H. W. Gellersen, G. Kortuem, and C. Kray. Cooperative artefacts: Assessing real world situations with embedded technology. In *Ubicomp*, pages 250–267, 2004.
24. K. Tanaka. *An Introduction to Fuzzy Logic for Practical Applications*. Springer-Verlag, 1996.
25. L. A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Computers and Mathematics*, 9:149–184, 1983.