The Network as a Computer: a Framework for Distributed Computing over IoT Mesh Networks

Emanuele Di Pascale, Irene Macaluso, Avishek Nag, Mark Kelly, Linda Doyle

***Abstract*—Ultra-dense Internet of Things (IoT) mesh networks and machine-to-machine communications herald an enormous opportunity for new computing paradigms and are serving as a catalyst for profound change in the evolution of the Internet. The collective computation capability of these IoT devices is typically neglected in favor of cloud or edge processing. We propose a framework to tap into this resource pool by coupling data communication and processing. Raw data captured by sensing devices can be aggregated and transformed into appropriate actions as it travels along the network towards actuating nodes.**

This paper presents an element of this vision, whereby we map the operations of an artificial neural network onto the communication of a multi-hop IoT network for simultaneous data transfer and processing. By exploiting the principle of locality inherent to many IoT applications, the proposed approach can reduce the latency in delivering processed information. Furthermore, it improves the distribution of energy consumption across the IoT network compared to a centralized processing scenario, thus mitigating the "energy hole" effect and extending the overall lifetime of the system.

***Index Terms*—Internet of Things, Artificial Neural Networks, Wireless Sensor Networks, Wireless Mesh Networks, Distributed Computing**

I. Introduction

THE Internet of Things (IoT) is rapidly progressing due to manufacturing advancements with respect to size, weight, power, and cost of next-generation low-power radio frequency transceivers and micro-controllers. As a result, IoT networks have significantly grown across a wide variety of domains and the number of IoT devices is forecasted to grow to about 20 billion by 2020 [1]. IoT mesh networks in particular are witnessing a resurgence both in the industry and in academia; for example, Qualcomm has recently announced a mesh networking platform for home IoT devices [2].

While most of the early IoT applications focused on passive data gathering and monitoring, actuation has been receiving more and more attention [3]. By coupling sensing devices with actuators capable of directly interacting with their environment, it is possible to design systems that can dynamically react to events and perform complex automated control tasks. As an example, in a factory setting we might want to use IoT to automatically detect faults and take appropriate recovery measures; in a smart city, to dynamically control lights at a set of junctions in response to the incoming traffic; in a smart home, to coordinate electric appliances so that their combined power usage stays below a set of thresholds based on the time of day.

Naturally, in all such applications some sort of data aggregation and processing is required to transform the input measurements into an appropriate output for the actuators; typically this computation is offloaded to a cloud platform. In the cloud-based model, data from the input sensors is routed through a local gateway to remote servers, where it is aggregated and processed; once an appropriate course of action has been determined, the required output signal is sent back to the IoT gateway and, from there, it is relayed to the actuator nodes.

In many application domains, however, the response time of the IoT system to external events must satisfy strict timing constraints, either due to the mission-critical nature of the process being supervised or simply to minimize the costs incurred by unnecessarily delaying the appropriate control measures [4]. Hence, it is paramount to limit the latency of these applications, i.e. the amount of time it takes to go from the raw input readings of the sensors to the final output signal at the actuating nodes. While the cloud-based processing model presents many advantages, e.g. in terms of cost-efficiency and ease of development, it also introduces a significant amount of latency by virtue of the round-trip to the remote servers and back that data has to take. For some classes of time-sensitive applications this can represent an unacceptable trade-off.

For this reason, many authors have proposed to move data processing from the cloud to the edge of the network [5], i.e., performing aggregation and computation at the local gateway and avoiding the cloud for all but the most resource-intensive operations. Even in the edge processing model, however, data still needs to reach a centralized collection point; in mesh networks this can be sub-optimal in terms of latency, as the source and destination devices might be located at the periphery of the network. More importantly, nodes that are directly connected to the gateway will be necessarily involved in the relaying of all the messages directed to the gateway itself; this causes an uneven distribution of the energy consumption among the nodes in the network, in what is called the energy hole effect [6]. As the first of these nodes runs out of power, the remaining 1-hop neighbors of the gateway will face an even greater load, thus creating an avalanche effect that can quickly disable the network. It is worth noting that this is an inherent problem of centralized data processing in mesh networks, regardless of whether the actual computation is performed locally at the edge or remotely in the cloud.

The next logical step then is in-network computation [7],

A preliminary version of this paper was published in the proceedings of the International Communications Conference (ICC) held in Paris, France on May, 2017.

Emanuele Di Pascale, Irene Macaluso, Avishek Nag and Linda Doyle are at CONNECT, Trinity College Dublin. Mark Kelly is at Intel Labs Europe.

an even more extreme form of edge computing whereby data processing is distributed among the nodes of the network. However, most of the works on in-network computation are limited in scope, focusing on simple functions (e.g. averaging of the readings from multiple sources) and typically assuming the existence of a single output sink for the processed data.

In this paper, we propose a generalized framework for in-network computation, in which the data aggregation and processing steps are implemented through an Artificial Neural Network (ANN) distributed across the IoT mesh network. More specifically, we map a neural network onto the IoT devices by designating a number of "hidden neuron" nodes that will collectively perform the required operations on the input data. Because in our framework computation is performed in parallel with communication as data travels towards its final destination, we call this approach Network as a Computer (NaC).

The theoretical foundation for this concept is provided by the Universal Approximation Theorem [8], which roughly states that a feed-forward neural network with a single layer of hidden neurons can approximate any continuous function of interest arbitrarily well. This theorem assures us that a wide range of functions of interest for IoT computation will be implementable with a single intermediate hidden neuron layer between the input and output nodes. Hence, we can extend in-network computations to use cases with more complex functions or a larger set of sink nodes than those typically considered in literature.

The underlying assumption is that, in many applications, sensing and actuating nodes are typically located close to each other, even when they are part of a larger IoT mesh network. Intuitively, in most control applications the actuators will need to interact with the environment in close proximity of the locations where the readings were taken; this idea is often referred to as the *locality of information processing* principle. The advantage of our approach is then twofold. Firstly, by appropriately placing hidden neurons near the input and output nodes we can reduce the length of the path that messages need to traverse, and hence decrease the latency experienced by the system. Secondly, by avoiding the central gateway and its surrounding nodes altogether we can better distribute the energy consumption across the IoT network and mitigate the energy hole problem. Combined, these two factors make for more responsive applications that can run for longer. These claims are investigated and backed by numerical results and simulations in sections V and VI.

The remainder of the paper is structured in the following way: firstly we discuss related work in section II. In section III we further clarify the NaC concept by defining the target scenarios referenced throughout the paper. Section IV formally defines the optimization framework for the placement of hidden neuron nodes in the network of IoT devices. Section V presents numerical results on the energy consumption of the nodes in the network, which show how our approach is able to improve the lifetime of the network compared to a centralized solution. In section VI we describe the simulation experiments carried out to validate our approach in terms of latency reduction. Finally, we present our conclusions.

II. Related Work

While we believe that the framework presented here is fundamentally novel, there are some previous related works targeting specific aspects of the problem at hand, or sharing the same aim but with a different approach. In this section we present a summary of the related literature.

A. In-Network Computation in Wireless Sensor Networks (WSNs)

Many works address the more general problem of in-network computation in WSNs. Most of the approaches in literature focus on determining the maximum achievable computation rate of specific functions [7], [9], [10]. Typically these works consider symmetric functions and assume a single collector node for the processed data. More recently, approaches considering the distributed implementation of a generic function have been considered [11], [12]. The function of interest is described as a weighted directed acyclic graph in [11] and as a directed tree in [12]. Both works, unlike ours, do not consider the resource availability in each node, which shapes the search space of the optimal mapping.

B. Distributed ANNs in WSNs or IoT Networks

Some existing research works report implementing distributed neural network architectures overlaying a sensor network or IoT networks similar to our proposal. For example, in [13] the authors develop neural network architectures that can be used to report information from a sensor network in a more "cognitive" manner. While there are certain similarities to the work we are presenting here, their proposal largely comes down to a method for data aggregation and filtering without any specific mention of resource-optimal mapping of neurons on sensor network nodes as we do.

The authors in [14] and [15] propose to distribute the different layers of a deep neural network between the low power IoT nodes and the cloud. Similarly, in [16] Preferred Networks Inc. propose to distribute different layers of a deep neural network into different edge devices. The authors in [17] define a sparse Neural Network (NN) architecture, where multiple subsets of a distributed data-set are placed into edge machines; the parameters of the NN are trained from the dataset and communicated to the cloud in order to complete the NN function. Our proposal is different from all these cited works in that we keep the neural network completely localized to the IoT nodes, bypassing cloud and edge processing entirely.

Again in [18], the authors investigate the placement of artificial neural networks in wireless sensor nodes to detect forest fires. However in their work each node implements a full neural network, while we propose to distribute the different components of the neural network over the IoT network optimally subject to resource constraints.

In [19] the authors propose a middleware layer to distribute neural network building blocks over multiple heterogeneous devices, not unlike what can be achieved by TensorFlow when multiple processing resources are available. Most importantly, their reference "IoT network" is composed of powerful machines that have little in common with the constrained IoT

devices we consider here. Finally, the authors did not develop optimization algorithms to distribute these functionalities over the available devices, although this is mentioned as possible future work.

In [20] the authors present a similar vision with a more specialized focus on self-adapting WSNs. They propose using a Hopfield neural network to solve a heuristic algorithm for determining the minimum weakly-connected dominating set of nodes in the network – a problem which is of relevance in many wireless communication protocols. Since every node is mapped to a neuron, they do not introduce an optimization framework in their proposal. General-purpose computation over a WSN through neural networks is mentioned as a possibility but not further investigated.

Lastly, the patent described in [21] describes a method to map a neural network to a network of embedded devices. However, their solution also lacks the mapping optimization aspect of our approach – nodes are instead uniformly mapped to available embedded devices. Their work also rigidly requires a full mesh network between the embedded devices, which can be a very restrictive assumption in many sparse deployment scenarios. Our solution does not require full mesh connectivity, using intermediate node relays instead whenever necessary.

C. Distributed Intelligence (non-NN-based) in WSNs or IoT Networks

A few other works in the literature also aim to put some form of distributed intelligence in sensor networks/IoT networks but not specifically point to a NN-based solution. The authors in [22] argue that it is not enough to connect everyday objects through the IoT infrastructure but that we also need to enable them with some level of intelligence to understand their environment. Among several approaches of establishing some form of cognition in the IoT infrastructure, the authors in [22], present an approach of non-cloud-based decentralized data processing using the Alternate Direction Method of Multipliers (ADMM). While ADMM is a good candidate to solve a distributed optimization problem using a massive dataset or even a large-scale machine learning algorithm [23], the authors in [22] do not explicitly discuss applications of ADMM to implement cognitive intelligence for IoT sensor nodes.

In [24], a strong case for developing distributed intelligence in IoT network nodes is presented. The authors share our belief in the importance of exploiting locality of information in WSNs, but they do not propose anything similar to our approach. In fact, none of the works discussed in this section account for resource-availability-aware optimal mapping of artificial intelligence (whether through a neural network or not) on IoT nodes.

III. Reference Scenarios

The high-level description of the concept presented in the introduction necessarily abstracts many implementation details that would depend on the specific context of application. Naturally, we must be able to map the concept to a specific IoT network instance for the purpose of experimentation and analysis. In this section we will formally introduce our two reference scenarios, explicitly detailing the assumptions made and defining key parameters of interest. In doing so, we have an opportunity to discuss how the NaC concept can be applied in practice on a given IoT network. The scenarios described here will be used both for the energy consumption analysis presented in section V and for the latency simulations detailed in section VI.

For our reference experiments we use a lattice of IoT devices. A grid topology was chosen both for its simplicity and because it matches many real-world deployments, such as a grid of parking sensors, or evenly spaced devices in a factory plant. However, our framework is applicable to any mesh network topology: to showcase this, we briefly present some results obtained on randomly generated graphs at the end of section V. With that exception, all the other experiments presented in the paper implement our reference lattice scenario.

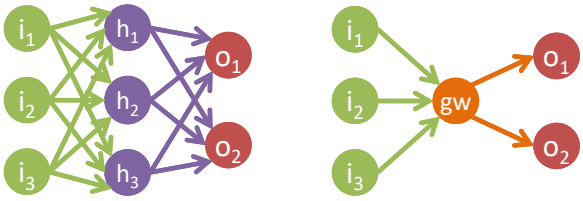
The IoT sensors in the grid form a mesh network and use message relaying to send measurements to a central gateway node for cloud storage and processing. We refer to these messages as IoT background traffic in the remainder of the paper. The sensors are characterized by η_{iot}, which represents the daily number of measurements collected by each device. Furthermore, a subset of the nodes in the grid act respectively as the input nodes and actuators of our target application. These nodes operate at a regular interval in what we define as messaging rounds. There are η_{nn} such messaging rounds each day; note that this parameter is uncorrelated with η_{iot}. During each of these rounds, each of the input nodes produces a measurement; these measurements need to be aggregated and processed in order to produce the required output at the actuators. The way in which this aggregation and processing is performed is what differentiates our two reference scenarios.

In the Network as a Computer (NaC) scenario, computation is performed in a distributed fashion through a feed-forward neural network [8]. As described in section IV, a predefined number of IoT nodes are selected by our optimization framework to behave as hidden neurons. During each round, the measurements from the input sensors also represent the input to our neural network. As such, they are directed to the IoT devices acting as hidden neurons, through what we call NN messages. Once all the required inputs have been collected at a hidden neuron node, the computed output signal can be forwarded to the actuators. Similarly, each actuator needs to collect the messages from all the hidden neuron nodes to extract the final output. In other words, the IoT network will implement the data flow model of Fig 1a. The resulting data traffic over the IoT network is referred to as NaC traffic in the rest of the paper.

On the other hand, in the centralized Gateway (GW) scenario input messages from the sensing nodes have to reach the gateway for aggregation and processing purposes. Once that is done, the desired output is forwarded back from the gateway to the actuators. This is exemplified by the data flow model shown in Fig. 1b. This represents our reference baseline scenario, which is used to evaluate the performance of our proposed approach.

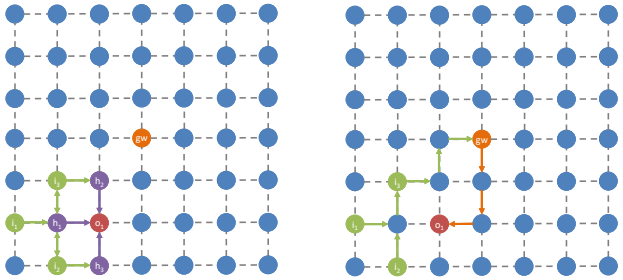
An example of how the data flow models described above

would map to an actual instance of the scenarios with a single actuator node is shown in Fig. 2.



(a) (b)

Fig. 1: Data flow model for the NaC scenario (a) and for the centralized GW scenario (b). Here and throughout the paper figures will adopt the following coloring convention: input nodes are depicted in green, hidden neuron nodes in purple, output nodes in red, and gateway nodes in orange. Messages originating from one of these node types will be represented with arrows of the same color.



(a) (b)

Fig. 2: Implementation of the data flow models of Fig. 1 for a particular deployment instance, respectively for the NaC scenario (a) and for the centralized GW scenario (b).

IV. NaC Optimization Framework

The optimal placement of hidden neuron nodes over the available IoT devices is intuitively crucial for the success of the NaC concept. On one hand, we must pay attention not to compromise the network by overloading devices beyond their capabilities, potentially exhausting their power budget or simply impacting on the performance of the application. At the same time, a sub-optimal placement could generate longer routing paths than those we would have with centralized processing; this could introduce additional latency and generate unnecessary traffic, potentially creating congestion in the network. Clearly, the choice must be performed in an optimal way, taking into account the constraints of the network and the cost of data transmission.

For this purpose, in [25] we introduced an optimization framework for hidden neuron placement. Specifically, the framework takes a topology of IoT devices with some input nodes and some actuators, and maps a given number of hidden neuron nodes on a subset of the available devices to optimize an objective function of interest (e.g. one that minimizes the transmission power used for messaging), subject to constraints on resource availability.

TABLE I: Optimization Framework Symbols

Symbol	Definition
$d_{k,j}$	Cost of the optimal path between nodes k and j.
E_B	Energy available at each node daily.
E_r	Energy consumed by one node for receiving one message.
E_s	Energy consumed for sensing by one node.
E_t	Energy consumed by one node for transmitting one message.
H	Set of hidden nodes for the neural network.
N	Set of nodes in the IoT network.
O	Set of output nodes for the neural network.
$P_{k,j}$	Optimal path between nodes k and j.
S	Set of source nodes for the neural network.
$T(i)$	Upper bound on the number of hidden neurons that can be mapped on node i.
x_i	Binary variable that takes value '1' if node i is selected as one or more hidden nodes.
y_i	Integer variable representing how many hidden nodes are deployed in node i.
η_{iot}	Number of background IoT messages generated by each node daily.
η_{nn}	Daily number of neural network rounds.

It is important to clarify a few assumptions underlying the framework. Firstly, it should be noted that the same IoT network can implement multiple applications, for each of which a separate instance of this framework can be run. Secondly, it is assumed that the structure of the neural network, namely the input and output nodes and the appropriate number of hidden neurons to perform the computation of interest, are available as an input to the optimization framework. These are parameters that depend on the specific application and are not discussed in this work. Finally, the optimal mapping requires a representation of the IoT topology. In particular, it is assumed that the routing tables used in the IoT network and relevant network-state information (e.g the cost associated with the use of a link in terms of delay) are made available to the optimal mapping. Moreover, the optimal mapping must be updated anytime a change in the topology occurs, e.g. when a new device joins the network or a link breakage is detected. It is worth noting that routing protocols for IoT networks typically include mechanisms for recovery in case of link or node failure and the gateway could easily be informed of changes in the network.

In this paper, we introduce two extensions to the optimization framework originally proposed in [25]. Firstly, our original model assumed that Dijkstra-like shortest path routing was available between any pair of nodes in the network. In practice, however, this is typically not the case in the low-complexity, low-power IoT networks that we are considering. For this reason, we modified our original model to take into account more realistic routing protocols for constrained networks. This first extension is detailed in subsection IV-A below.

The second extension is related to the energy constraints of the available devices. As briefly explained above, the choice of IoT devices and wireless paths that implement the neural

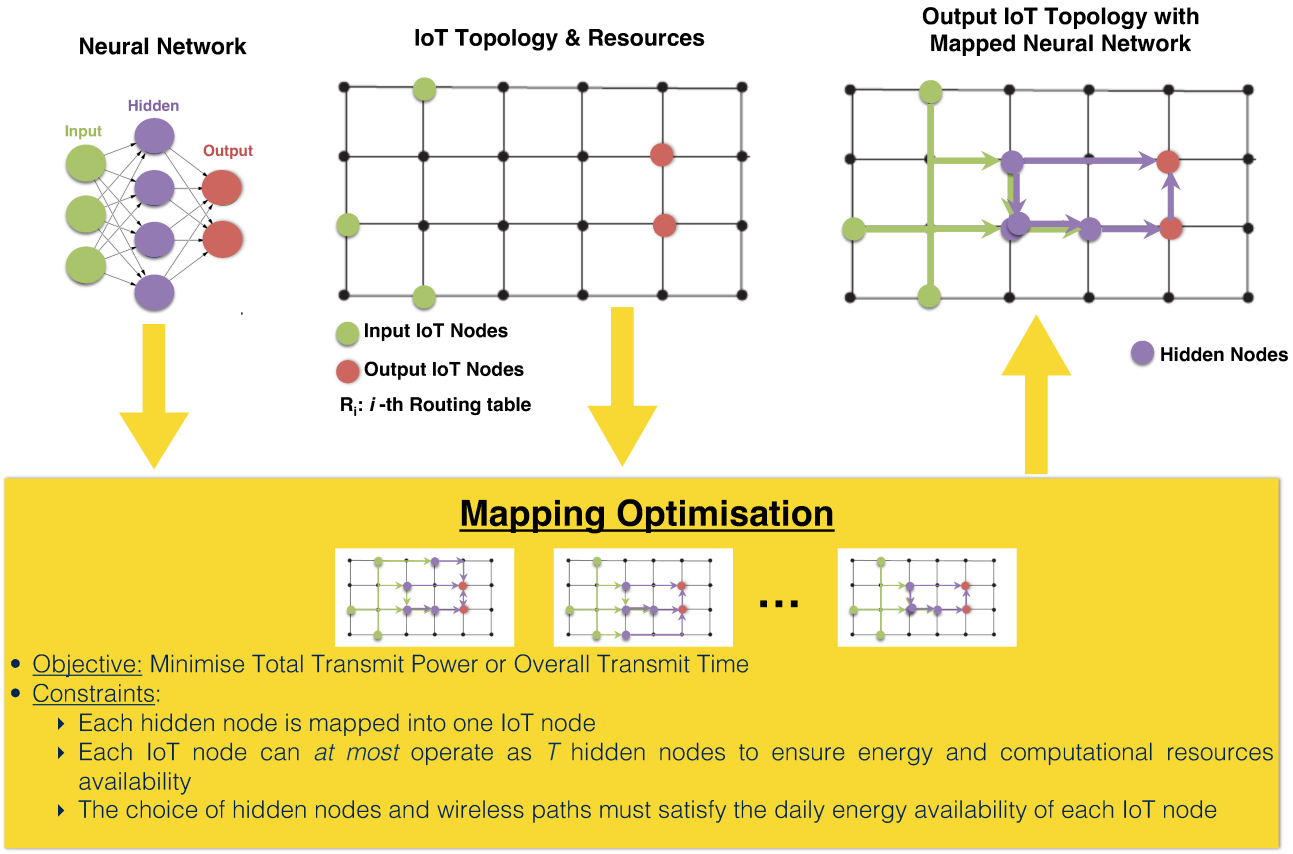


Fig. 3: The proposed optimization framework

network is constrained by the resource usage on each IoT device, which shapes the search space of the optimal mapping problem. In the optimization framework proposed in [25], we assumed that those constraints were known in advance, so that each IoT node n could at most operate as $T(n)$ hidden neurons. Here we extend the original model to take into account the energy availability at each node and the energy consumed in transmitting/receiving the NN messages as well as the background IoT messages. We will first formalize the problem in the case the neural network is trained offline, and then extend it to account for the exchange of training messages. While the process of training the neural network is not trivial, and performing it on a network of constrained devices would neither be an efficient choice nor bring significant benefits compared to an offline training, the online weights adaptation might be useful for certain applications, e.g. process monitoring or fault diagnosis in which the data distribution changes over time. The extended framework is detailed in subsection IV-B below. Table I details the symbols used in the remainder of the paper. The updated optimization framework is depicted in Fig. 3.

A. Constrained Routing in IoT Networks

IoT networks come in many shapes and flavors, depending on the domain of interest and the requirements of the service to be implemented. Arguably the applications that are gathering the highest momentum are deployed on small and inexpensive devices, typically battery-operated, which are tasked with simple monitoring and control tasks. The limited capabilities of these nodes is compensated by their low operational costs and their consequent pervasiveness, allowing IoT business opportunities to flourish in domains that would otherwise be prohibitively expensive to automate. These devices also represent an inherently more interesting target for distributed computing, which offers a way to implement complex tasks that would otherwise exceed the capabilities of the individual nodes of the network. For these reasons, in this paper we focus our attention on low-power IoT mesh networks.

For example, Low-Power and Lossy Networks (LLNs) [26] are networks in which constrained nodes (i.e. with limited processing power, memory or energy) are interconnected by lossy links, often supporting low data rates. Multi-hop routing in LLNs comes in either of two fundamental flavors: *route-over*, where routing is implemented at layer 3 of the network protocol stack (i.e. at the network layer) at each hop, or *mesh-under*, where the multi-hop forwarding is handled directly at layer 2 (i.e. at the data-link layer), presenting the abstraction of a single layer 3 link between source and destination. The most widespread approach is the route-over one, in particular using the RPL protocol [27]. RPL is based

on the use of Destination-Oriented DAGs (DODAGs), that is, optimized Directed Acyclic Graphs (DAGs) connecting the nodes of the network to a single destination root, such as for example the network gateway. If a node wants to send a message to a different destination from the root of the DODAG, the message will still have to traverse the graph structure, potentially reaching the root before being sent back down again towards the destination; this is done to reduce the workload at intermediate nodes, as many of them may work in *non-storing mode*, i.e., may not keep a full view of the network topology. The assumption of the protocol is that most messages will go from a generic source to a single shared sink - or a small subset of sinks, as in principle it is possible to have multiple DODAGs rooted at different nodes. This is highly inefficient in decentralized networks like the ones we envision in this paper.

Another option is IEEE's 802.15.5 standard [28], a mesh-under protocol designed for low-power Wireless Personal Area Networks (WPANs). At its core 802.15.5 is similar to RPL, using tree structures that are nearly equivalent to RPL's DODAGs, but with some additional measures to limit the inefficiencies of routing through a single-root structure. Specifically, nodes can optionally keep track of the local link state by exchanging periodic messages with other surrounding nodes. Using this information, they can build a k-hop neighbors list (with $k \geq 2$), i.e. a list of nodes reachable within k hops disregarding the constraints of the shared routing tree structure. When forwarding multi-hop messages, a node first checks whether it can reach the destination through some of its local connections recorded in the k-hop neighbor lists, and only if that is not possible it defaults to sending the message along the shared tree.

Naturally, if the placement algorithm in our optimization framework uses a routing mechanism not available in the actual IoT network we are considering, the resulting hidden neuron deployment might be inefficient or even counterproductive. For this reason, we have modified the optimization problem presented in [25] to better match the nature of the constrained routing environments we are dealing with. Specifically, we assume an 802.15.5-like routing protocol, with a tree structure rooted in the network gateway, and 2-hop neighbor lists in each node to allow for local shortcuts. The process that leads to the generation of the routing table is exemplified in Fig. 4: from the physical grid topology of IoT devices (Fig. 4a) we build a minimum spanning tree rooted at the gateway in the middle to represent the single DODAG structure (Fig. 4b). Then, for each node, we add routing entries to allow for shortcuts through the 2-hop neighbors list - e.g., in node 5 we would add node 4 as the next hop for messages directed to nodes 3 or 4 (Fig. 4c). Without the local neighbor list, messages directed to node 3 would instead be routed to node 10 along the minimum spanning tree, taking a much longer path before reaching their destination.

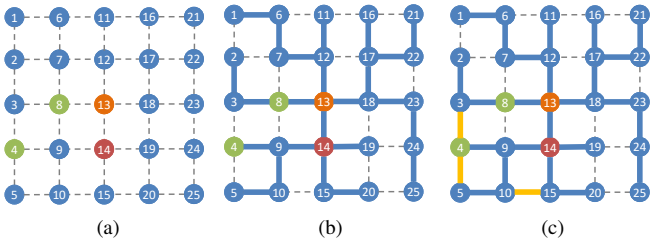


Fig. 4: Routing table generation for constrained routing: (a) the physical topology of devices, (b) the minimum spanned tree rooted at the gateway, (c) additional logical links for the 2-hop neighbors of node 5. See Fig. 1 for the color coding.

B. Energy-Aware Optimal Mapping

The optimal mapping, which identifies the IoT nodes that act as hidden neurons and the optimal paths from the inputs to the outputs via the hidden neurons, can be formulated as an integer linear program that minimizes a cost function subject to constraints on the resource availability of each node.

In this paper we consider two objective functions for the optimal mapping of neurons into IoT nodes: i) minimizes the overall cost of communication, while ii) minimizes the maximum cost of communication. The optimal path $P_{i,j}$ between every two pair of nodes (i, j) is precomputed based on the 802.15.5-like routing protocol explained above. In the first case, the cost of $P_{i,j}$, denoted by $d_{i,j}$, is the sum of the transmit power required to transmit a message between each pair of adjacent nodes in $P_{i,j}$[1]. The overall cost of communication corresponds to the total transmit power required to deliver the processed information to the output nodes. The corresponding objective function is given by:

$$\min_{x_i} \sum_{i \in N} (\sum_{s \in S} d_{s,i} + \sum_{o \in O} d_{i,o}) x_i \tag{1}$$

where x_i is a binary variable that takes value 1 if node i is selected as one or more hidden nodes.

In the second case, $d_{i,j}$ corresponds to the sum of the expected transmit time between each pair of adjacent nodes in $P_{i,j}$. The objective function, which corresponds to the maximum transmit time to deliver the processed information to the output nodes, is given by:

$$\min_{x_i} \sum_{i \in N} (\max_{s \in S}(d_{s,i}) + \max_{o \in O}(d_{i,o})) x_i \tag{2}$$

The choice of IoT devices and wireless paths that implement the neural network is constrained by the resource usage on each device. The constraints, which shape the search space of the optimal mapping problem, are given in (3), (4), (5), and (6). Constraint (3) ensures the integrity of the mapping model by guaranteeing that exactly $|H|$ hidden nodes are allocated. Constraints (4) and (5) ensure that: i) if node n is not selected to implement hidden neurons ($x_n = 0$), the number of hidden neurons deployed in node n is 0 ($y_n = 0$); ii) if node n is selected to implement hidden neurons ($x_n = 1$), then $1 \leq y_n \leq T(n)$. Constraint (6) ensures that choice of hidden nodes

[1]In the context of the optimal mapping discussed in this section we assume an ideal MAC layer, with no collisions and retransmissions. The impact of imperfect MAC in the presence of interference is evaluated using Cooja, as detailed in section VI.

$$\sum_{i \in N} y_i = |H| \tag{3}$$

$$x_n - y_n \leq 0 \qquad \forall n \in N \tag{4}$$

$$-T(n)x_n + y_n \leq 0 \qquad \forall n \in N \tag{5}$$

$$\mathbb{1}_S(n)E_s + \sum_{s \in S}\sum_{i \in N}(\mathbb{1}_{P_{-s,i}}(n)E_r + \mathbb{1}_{P_{s,-i}}(n)E_t)x_i + \sum_{o \in O}\sum_{i \in N}(\mathbb{1}_{P_{-i,o}}(n)E_r + \mathbb{1}_{P_{-i,-o}}(n)E_t)x_i + y_n E_t|O| \leq E_u(n) \qquad \forall n \in N \tag{6}$$

$$x_n \in \{0,1\}, y_n \in \{0,1,\ldots,|H|\} \qquad \forall n \in N \tag{7}$$

and wireless paths satisfies the daily energy availability of each node. The right-hand side of (6), $E_u(n)$, represents the energy available at node n for each neural network computation and it is computed as

$$\frac{E_B - \eta_{iot}E_{iot}(n)}{\eta_{nn}},$$

where $E_{iot}(n)$ is the energy consumed by node n for each IoT round. In other words, the energy available for each neural network round is obtained by subtracting the energy consumed by the node for all the IoT-related operations in a day ($\eta_{iot}E_{iot}(n)$) from the overall daily energy budget (E_B) and then dividing the result by the number of neural network computations in a day (η_{nn}). $E_{iot}(n)$ is computed based on the model presented in [29]. In addition to sensing and transmitting its own IoT data, a device may also have to relay the IoT data of other nodes towards the gateway.

For each IoT round, $E_{iot}(n)$ can be computed as follows:

$$E_{iot}(n) = E_s + \mathrm{in(n)}E_r + \mathrm{out(n)}E_s \tag{8}$$

where $\mathrm{in(n)}$ and $\mathrm{out(n)}$ are the number of incoming and outgoing paths to node n towards the gateway, and are precomputed based on the 802.15.5-like routing protocol. In the left-hand side of (6) $\mathbb{1}_S(n)$ is equal to 1 if node $n \in S$, and 0 otherwise; $\mathbb{1}_{P_{-s,i}}(n)$ is equal to 1 if node $n \neq s$ belongs to the path from source s to node i , and 0 otherwise; $\mathbb{1}_{P_{s,-i}}(n)$ is equal to 1 if node $n \neq i$ belongs to the path from source s to node i , and 0 otherwise; $\mathbb{1}_{P_{-i,o}}(n)$ is equal to 1 if node $n \neq i$ belongs to the path from node i to output o; $\mathbb{1}_{P_{-i,-o}}(n)$ is equal to 1 if node $n \neq o$ and $n \neq i$ belongs to the path from node i to output o, and 0 otherwise. The first term in (6) is equal to energy consumed by node n to sense, if node n is one of the input nodes of the neural network, and 0 otherwise. The last term ($y_n E_t|O|$) is 0 if no hidden neurons are allocated to node n; otherwise it is the energy consumed to transmit the messages processed by each hidden neuron allocated to node n to each output node. The assumption here is that messages cannot be compressed. Hence, node n has to send to each output node a number of messages equal to the assigned number of hidden neurons (y_n). The first summation in (6) represents the energy consumed by node n to relay messages from the input nodes to the hidden nodes. The second summation represents the energy consumed by node n to relay messages from the hidden nodes to the output nodes.

The optimal mapping formulated above takes into account the flow of data from the input to the output nodes. In other words, it is assumed that the neural network is trained offline and is used by the IoT network to perform the required operations on the input data. It is possible to extend the optimal mapping so as to account for the messages exchanges required to update the neural network weights when using the backpropagation algorithm [30]. Backpropagation is the most commonly used algorithm for training neural networks. The algorithm works by first forward-propagating the inputs to compute the corresponding output of the neural network, and then back-propagating the error from the outputs to the hidden nodes. Hence, the objective function and the energy availability constraint (6) have to be modified by adding the contribution of the back-propagation step. For example, denoting by ρ the fraction of daily NN rounds that are also used to update the network weights[2], the objective function corresponding to the total transmit power can be expressed as follows:

$$\min_{x_i} \sum_{i \in N} \Big(\sum_{s \in S} d_{s,i} + \sum_{o \in O} (d_{i,o} + d_{o,i}\rho) \Big) x_i. \tag{9}$$

When including the contribution of the back-propagation messages, the energy availability constraint is given in (10), where $\mathbb{1}_{P_{-o,i}}(n)$ is equal to 1 if node $n \neq o$ belongs to the path from output o to node i; $\mathbb{1}_{P_{-o,-i}}(n)$ is equal to 1 if node $n \neq o$ and $n \neq i$ belongs to the path from output o to node i, and 0 otherwise; $\mathbb{1}_O(n)$ is equal to 1 if node n is one of the output nodes, and 0 otherwise.

In terms of complexity, the problem of embedding a neural network in a network of IoT devices belongs to a class of network embedding problems which, in the absence of constraints on resources availability, are shown in [11] to be NP-complete. In the optimization problem formulated in the paper there are $|N|$ binary variables (x_i) and $|N|$ integer variables (y_i), each upper bounded by the number of hidden nodes in the neural network ($|H|$). Each integer variable can be expressed as $|H|+1$ binary variables. Therefore, the total number of binary variables in the problem is $|N| \times (|H|+2)$. As expected, the problem size grows with the size of the IoT network determined by $|N|$ and the size of the neural

[2]If the neural network is first trained offline, only a subset of the new measurements, for which the target outputs are known, is used for online training.

$$\mathbb{1}_S(n)E_s + \sum_{s\in S}\sum_{i\in N}(\mathbb{1}_{P_{-s,i}}(n)E_r + \mathbb{1}_{P_{s,-i}}(n)E_t)x_i + \sum_{o\in O}\sum_{i\in N}(\mathbb{1}_{P_{-i,o}}(n)E_r + \mathbb{1}_{P_{-i,-o}}(n)E_t)x_i + y_nE_t|O| + \rho\Big(\sum_{o\in O}\sum_{i\in N}(\mathbb{1}_{P_{-o,i}}(n)E_r + \mathbb{1}_{P_{-o,-i}}(n)E_t)x_i + \mathbb{1}_O(n)E_t|H|\Big) \leq E_u(n) \qquad \forall n \in N \tag{10}$$

TABLE II: Recap of the simulation parameters for energy consumption evaluation

Parameter Name	Parameter Value(s)
E_t	$195.6e-9$ Joules per bit
E_r	$220.8e-9$ Joules per bit
Gateway nodes	1 (middle of the grid)
Grid size	7x7
Hidden neuron nodes	3
Input nodes	3
Network lifetime	6 months
Output nodes	2
Packet size	10^3 bits
η_{iot}	12 (every hour)
η_{nn}	1 (every day)

network determined by $|H|$. Despite this, our tests show that the optimization problem remains tractable when the network size increases; as an example, it takes less than a minute to complete on a standard laptop for a neural network with 15 hidden nodes over a 25×25 grid, that is to say, for a network of the order of magnitude of 625 nodes.

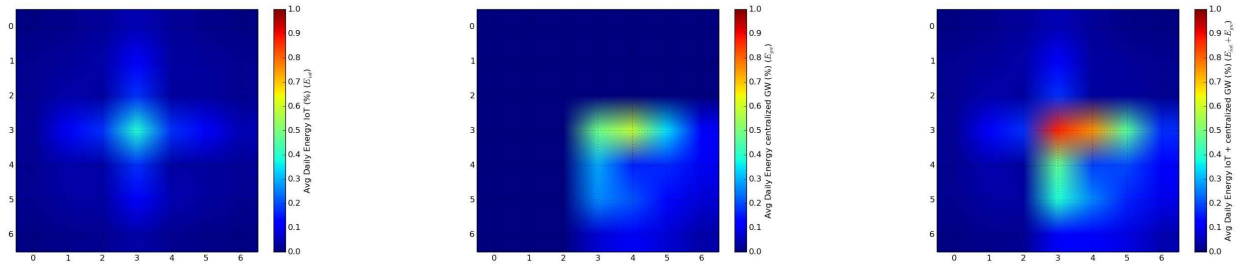
V. Energy Consumption Evaluation

In this section we analyze the properties of the NaC solution with respect to the energy consumption of the IoT devices. We consider a grid of nodes, where each node can communicate with its closest neighbors (Von Neumann neighborhood) and the center node of the grid is the gateway used by both the IoT background traffic and the centralized GW traffic. Input and output nodes of the NN are selected randomly from a single quadrant of the grid (south-east of the gateway) to ensure that the locality of information processing principle can be applied. We focus on the core of the NaC formulation, i.e without the backpropagation elements.

The results presented in Figures 5 , 6, 7 and 8 refer to a scenario with a grid of 7×7 IoT nodes, 3 input nodes, 3 hidden neurons and 2 output nodes for the NN. We also conducted experiments with larger networks (up to 11×11 IoT nodes), and the results show similar trends. It is important to note that the number of nodes in multi-hop networks is typically in the order of tens per gateway for a wide range of applications, from networked control systems in industrial environments [31] to smart city deployments [32]. The results shown in Figures 5 , 6, 7 and 8 are averaged over 10^2 independent selections of input and output nodes, and the corresponding optimal hidden nodes in the case of NaC. Moreover, we consider $\eta_{nn} = 12$ NN daily rounds and $\eta_{iot} = 1$ IoT daily rounds. Finally, we assume $E_t = 195.6e-9$ Joules per bit [33], $E_r = 220.8e-9$ Joules per bit [33], a packet size of 10^3 bits, and a battery of 5 Joules [33]. Assuming a network lifetime of 6 months, $E_B = 5/(0.5 * 365) = 0.0274$ (Joules per day) is the daily energy budget available to each node. The parameters of interest for the simulations are summarized in Table II.

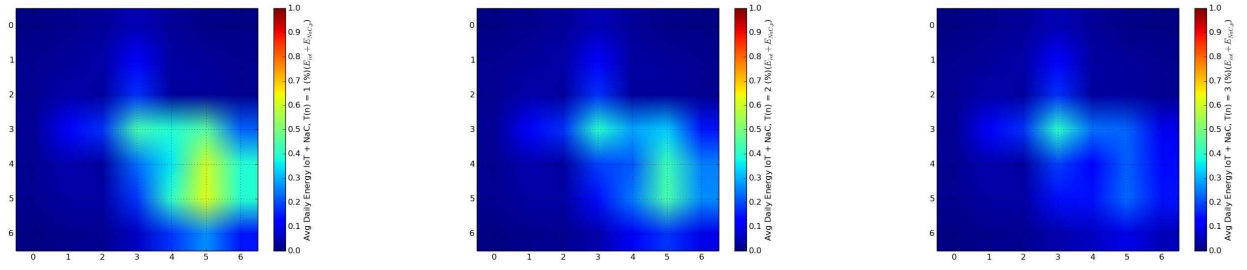
Figures 5 and 6 show the percentage of the daily energy budget (E_B) used by each node in the grid for the different types of traffic. In particular, Figure 5a shows the percentage of energy used by IoT messages (E_{iot}), i.e. to collect IoT measurements at the gateway (node $(3, 3)$ in the grid). Obviously, the energy consumption is higher for nodes responsible to relay messages, as it can be observed by the cross-like pattern in figure. Figure 5b shows the percentage of energy used by the centralized GW messages (E_{gw}). Since input and output nodes are localized in the south-east quadrant, only those nodes consume energy. Since the data has to be collected at the gateway to be processed before being sent to the output nodes, the gateway and the nodes responsible to relay messages show a higher energy usage. Figure 5c shows the combined energy used by IoT messages and the centralized GW messages ($E_{iot} + E_{gw}$). Figures 6a, 6b, and 6c show the combined energy used by IoT messages and the NaC messages corresponding to the objective function in (1) ($E_{iot} + E_{NaC,p}$) for $T(n) = 1$, $T(n) = 2$, and $T(n) = 3$ respectively, i.e. when each IoT node can implement at most 1, 2, and 3 hidden neurons. We denote by $E_{NaC,p}$ the energy consumed by the distributed NaC in correspondence to the objective function in (1). Results corresponding to the objective function in (2) - for which the energy consumption is denoted by $E_{NaC,t}$ - show a similar trend. Regardless of the value of $T(n)$, the comparison between NaC (Figures 6a, 6b, and 6c) and the centralized GW solution (Figure 5c) highlights a key feature of the NaC solution: by processing information locally we mitigate the so-called "energy-hole" problem. In other words, the NaC solution avoids overloading those nodes that already have a higher traffic load due to the background IoT traffic. By comparing results Figures 6a, 6b, and 6c, we can observe that distributing the information processing among more nodes, i.e. considering a lower value of $T(n)$, results in a higher energy usage. This is expected, in that the lower is $T(n)$, i.e. the higher the number of IoT devices selected as hidden neurons, the higher is the required number of messages. However, the increase in energy usage is localized in nodes that are less critical for the network.

In order to assess the impact of the optimal hidden nodes placement on the energy usage, we evaluated the energy consumption in the network in case hidden nodes are randomly selected in the same quadrant of the input and output nodes.



(a) (b) (c)

Fig. 5: Percentage of the daily energy budget used by IoT traffic (a), centralized GW (b), combined IoT traffic and centralized GW (c).



(a) (b) (c)

Fig. 6: Percentage of the daily energy budget used by the combined IoT traffic and NaC with $T(n) = 1$ (a), $T(n) = 2$ (b), and $T(n) = 3$ (c).

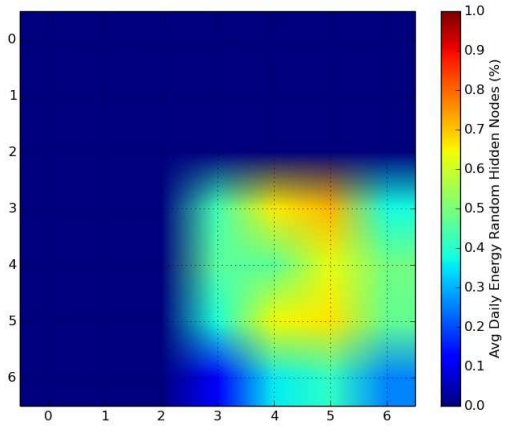


Fig. 7: Percentage of the daily energy budget used by the combined IoT traffic and NaC with random hidden nodes.

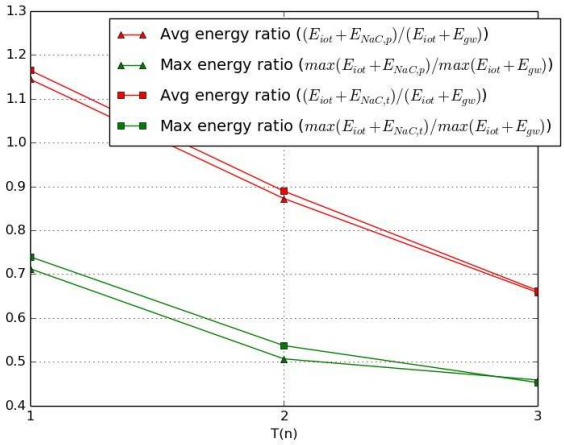


Fig. 8: Ratio of the energy used in the centralized GW solution to the distributed NaC solution for the two objective functions. The curves with triangular markers refer to objective function (1), while the curves with the square markers refer to objective function (2)

Figure 7 shows the combined energy used by IoT messages and the NaC messages for randomly selected hidden nodes, assuming each IoT device can implement at most 1 hidden neuron. By comparing results in Figures 7 and 6a we can observe that the optimal placement of hidden neuron nodes plays a key role in decreasing the energy usage in the network when implementing the NaC scenario. On average, the network energy consumption in case of optimally placed hidden nodes is 65% of the energy used in case of randomly selected hidden nodes.

Figure 8 shows the ratio of the average energy used by the combined IoT/NaC solution (for the two objective functions) to the energy used by the combined IoT/centralized GW

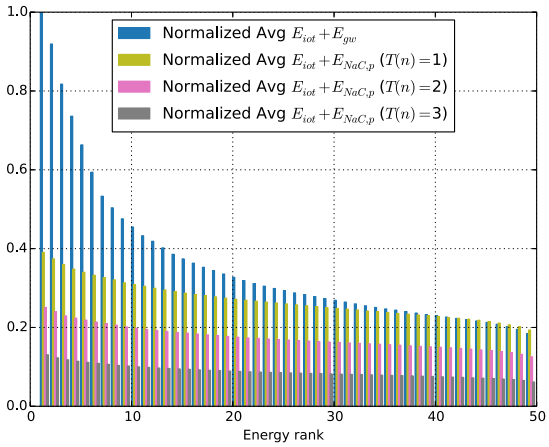


Fig. 9: Normalized energy consumption of nodes in a random mesh IoT network.

solution as a function of $T(n)$. The figure also shows the ratio of the maximum energy usage in the network for both cases. It can be observed that both NaC solutions result in a decrease of the average energy used when $T(n) \geq 2$. In the case of $T(n) = 1$, the slight increase in the average consumed energy is compensated by the decrease in the maximum energy usage. In other words, even when more energy is used on average, the energy consumption is distributed among nodes with a higher reserve of energy.

We also evaluated the NaC solution in the more general case of random mesh IoT topologies. Figure 9 shows the normalized energy consumption averaged over 10^3 random topologies with 49 nodes. Each topology is generated by uniformly distributing the devices over a target square area, and then interconnecting devices that are located at a distance lower than the transmission range. As in the case of the grid deployment, the optimal path between every two pair of nodes is precomputed based on the 802.15.5-like routing protocol. The gateway in each topology is randomly selected from the center of the graph, i.e. the set of nodes that minimize the maximal distance from other nodes in the graph. In order to preserve the locality of information principle, for each topology, input and output nodes are randomly selected so that the maximum distance between nodes is less than 5 hops, i.e., within a 2-hop neighborhood of a randomly selected origin node. The remaining parameters are the same as those used for the previous simulations, as summarized in Table II. For each random topology, nodes are ranked according to the combined energy consumed by the centralized GW and IoT messages, and according to the combined energy consumed by the NaC solutions (with $T(n) \in \{1, 2, 3\}$) and IoT messages. The ranked energy profiles are averaged over the 10^3 samples and normalized with respect to the energy used by the node with the highest consumption. As shown in figure, the energy consumption corresponding to NaC, for each value of $T(n)$, is more evenly distributed than in the case of the centralized GW approach. Among the different NaC solutions, higher values of $T(n)$ result in more evenly distributed energy consumption profiles. As we showed for the regular lattice deployment, by processing information locally once again the NaC approach avoids overloading the nodes already carrying a higher traffic load due the background IoT traffic, even in the case of random mesh topologies.

VI. Latency Evaluation

A. Choosing a Simulation Framework

One of the benefits that we claim for our distributed processing approach is the reduction of latency in all those cases where input and actuating nodes are physically located close to each other – or, in other words, when the principle of locality of information processing applies. Short of a full implementation of the concept, to verify this claim we needed a simulation framework capable of providing an estimation of the performance of the system under reasonably realistic assumptions. The main constraint that dictated our choice was the availability of multi-hop routing models, preferably in a low-power setting.

The best fit turned out to be Cooja [34], the simulation/emulation environment that comes with Contiki. Contiki is an Operating System for constrained IoT devices which is widely adopted both in industry and academia. Each node in the Cooja simulation environment is an actual compiled and executed Contiki instance, running the same code that would run on a physical mote; this results in accurate and realistic simulations. It also boasts an extensive community of active developers and support for a large set of Internet standards.

Specifically, Contiki provides two separate networking stacks: uIP and Rime. The former is an implementation of the TCP/IP stack for constrained devices; focusing our attention on mesh networking, uIP implements natively RPL, the multi-hop protocol briefly described in subsection IV-A. However, support for multiple instances of DODAGs is not fully implemented nor well documented, and simulations using a single DODAG would not perform well in our decentralized scenario; furthermore, there is no implementation of mesh-under protocols in uIP. For these reasons, we decided to look elsewhere.

The Rime communication stack [35] provides a set of lightweight communication primitives ranging from best-effort local area broadcast to reliable network flooding. Specific "transformation modules" are then provided to map those communication primitives to existing protocols that physical devices can understand. The main advantage of using Rime lies in its simplicity, as we can combine and build upon its bare-metal primitives to mimic mesh protocols that better fit our requirements. Specifically, we were able to emulate the behavior of 802.15.5 by implementing neighboring lists at the routing table of each node. Furthermore, the fine-grained control offered by Rime over the forwarding logic used by the multi-hop primitive at run time ensures that the routing decisions taken by the IoT devices match the model used in the optimization framework.

B. Implementation Details

The simulations implement the reference scenarios described in section III. The objective is to compare the messaging latency respectively using our distributed NaC concept or with a more traditional cloud-based approach. In all cases, input and actuator nodes are selected randomly from a single quadrant of the grid to ensure that locality of information processing applies.

Rime offers two multi-hop transmission primitives: best-effort multi-hop (mh) and reliable multi-hop (rmh). The former simply attempts to forward a message to a final destination through intermediate hops that need to be provided through a custom-defined routing function. The latter, on the other hand, requires each intermediate hop to acknowledge the reception of the message to the previous hop; the maximum number of re-transmissions (5) and the ACK timeout (1s) are modifiable parameters of the protocol, as is the maximum number of hops that the primitive will support before dropping the packet.

No matter what ACK timeout was chosen, the usage of reliable multi-hop primitives also created the problem of duplicate messages, which had to be properly dealt with at the receiver to avoid byzantine behaviors. This was done by keeping track at each node of the expected sequence number, identifying the current "round" of NN operations, as well as the packets already received from each of the involved actors (e.g. each of the input nodes, in the case of a hidden node). Furthermore, there is no notification of the source node in case of a message loss somewhere down the forwarding chain. In Rime, only the last successful intermediate hop is aware of the packet loss, and there is no mechanism in place to back-propagate negative acknowledgements. For this reason, losses are detected only posthumously at the reception of an NN message with a unexpected sequence number.

Routing tables for this multi-hop forwarding are saved to file by the optimization framework algorithm and loaded in each node at run time; this is done to make sure that the routes computed by the optimal node placement algorithm correspond to the paths taken by messages in the actual simulation. These routes are shared by both NN messages (to whatever destination) and background IoT messages to the gateway. The process used to compute these routing tables is described at the end of subsection IV-A

Interference between nodes is simulated using Coojas built-in Unit Disk Graph Radio Medium (UDGM) model: transmission range is modeled as a circular range, with interference reaching a larger circular range. These ranges are set so that nodes can only communicate with adjacent nodes in the lattice (i.e. the 4 nodes next to it on the vertical and horizontal axis), while interfering also with their diagonal neighbors, as shown in Fig. 10. By taking into account data loss due to interference, these simulations can help us better understand the impact of a non-perfect MAC on the performance of our framework.

Latency is measured for each of the NN messaging rounds described in subsection III. Specifically, we define it as the interval of time between the instant the last of the input measurements of that round is available to the instant the last output node receives the processed data. We recall here that, at each round, we need to aggregate data from each of the input nodes in order to produce the desired output. Cooja has internal tools to track networking metrics of interests, including latency, but naturally it has no way to understand this end-to-end definition of latency, spanning several source-destination pairs. For this reason, we take these measurements through specific log parsers that were developed in Python.

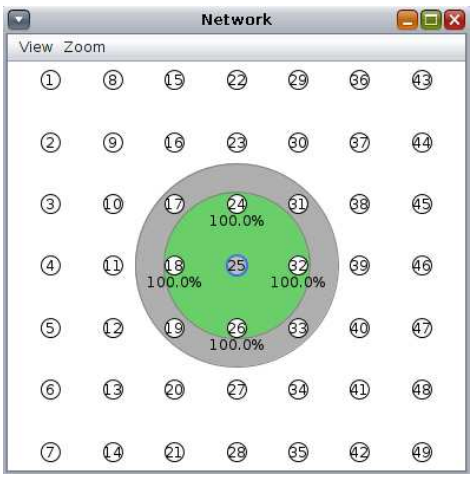


Fig. 10: Interference model in Cooja: the green and gray circles represent respectively the transmission and interference range of the middle node.

On the other hand, the remaining IoT nodes in the grid behave in the same way in both the centralized and distributed scenarios, and thus their latency measurements are not included in the results; their main purpose is to enhance the realism of the simulation by generating a sparse background traffic which can interfere with the transmission of our messages of interest.

Various versions of both the NaC and GW based simulation scenarios were implemented, using different combinations of reliable and best-effort multi-hop forwarding. A lesson that we learned early on is that the proximity of input nodes can work against us by causing many collisions and hence loss of messages. For this reason, the results shown here correspond to scenarios in which the NN messages from input nodes are forcibly spaced in time; in other words, if the length of a NN messaging round is 60 seconds, we might have the first node sending its messages to the hidden neuron nodes at $t = 0s$ of a round, the second input node sending its messages at $t = 10s$ of the same round, and so on.

Furthermore, we soon realized that using reliable multi-hop for both NN and IoT messages led to many lost packets, as both mechanisms attempt to recover from a collision by re-transmitting their message, hence causing even more congestion; better results were achieved using reliable transmission primitives only for the neural network messages, while background IoT messages use the best-effort primitive. All the results presented in subsection VI-C follow this convention. Other parameters of interest for the simulations are gathered in Table III. Note that the messaging frequency is much higher than what we would observe in most WSN deployments; this

TABLE III: Recap of the parameters for the Cooja simulations

Parameter Name	Parameter Value(s)
ACK timeout	1 second
Gateway nodes	1 (middle of the grid)
Grid size	7x7, 11x11
Hidden neuron nodes	3
Input nodes	3
Max. retransmissions	5
NN rounds per run	160
Output nodes	1
η_{iot}	144 (every 10 minutes)
η_{nn}	1440 (every minute)

is simply to reduce the length of the simulation runs, which is quite significant.

C. Simulation Results

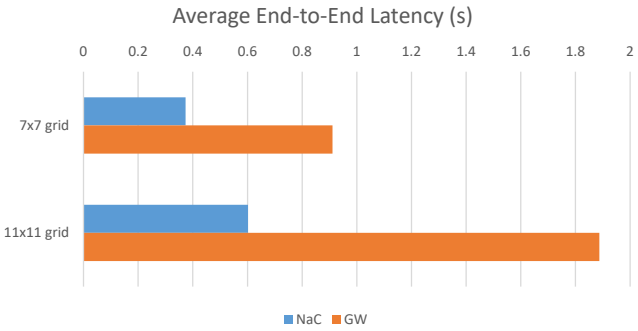


Fig. 11: Average latency of a NN messaging round.

Fig. 11 shows the averaged end-to-end latency for NN messaging rounds, using either NaC or a centralized gateway processing, for two simulation batches using respectively a 7x7 and an 11x11 grid of IoT devices, similarly to what was detailed in section V. Thanks to the clustering of input and output nodes, in the 7x7 results the latency of NN messages is more than halved for NaC compared to the GW solution. These averaged results include a "worst case" scenario in which input nodes were hand-picked at the extremities of the quadrant of choice and the output node was placed adjacently to the gateway; even in these conditions the NaC approach is able to reduce the overall latency of NN messages from $0.67s$ to $0.44s$ on average.

On the larger 11x11 grid, latency results are even more flattering for the NaC approach. This is intuitively explained by the increased distance that a multi-hop message needs to travel from the input node to the gateway and then finally to the output node as the size of the grid increases; thanks to locality of information and the optimal placement of hidden neuron nodes in the network, our distributed solution scales better than a centralized one.

Fig. 12 shows results on the percentage of failed messaging rounds for the same batches of simulations, i.e. those where some of the NN messages were lost due to collisions before reaching the actuator. For the 7x7 grid batch using NaC, we registered a 1.7% increase in the number of failed rounds compared to the centralized case – that is, from 0.15% to

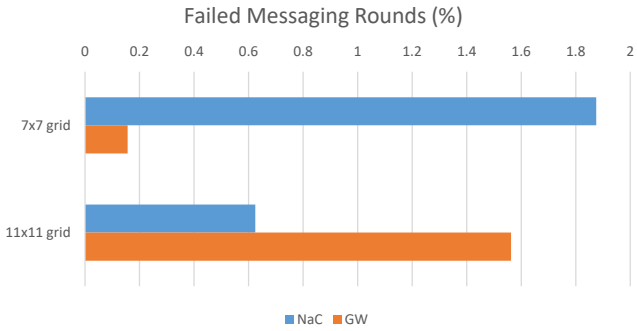


Fig. 12: Average percentage of failed messaging rounds.

1.86% on average. In a relatively small network such as the one considered, the additional traffic generated by messages to the hidden neuron nodes seems indeed to have a negative effect on the reliability of message delivery, albeit with values that are still very reasonable for an LLN.

However, on the larger 11x11 grid topology this trend is reversed, with NaC outperforming the centralized GW approach even in terms of failed messaging rounds. Together with the latency results detailed above, we see this as a further proof of the inherent scalability advantage of NaC compared to a non-distributed solution.

VII. Conclusions

This paper extends the work in [25], where we first introduced a neural-network-based framework for in-network computation, exploiting the communications between the devices of an IoT mesh network to perform data processing and aggregation.

Specifically, in this work we refined our optimization framework to take into account the constraints of the routing protocols typically implemented in a low-power context. Furthermore, we enriched the system model to account for the energy consumption of the messaging process, in order to optimize the placement of the hidden neurons and improve the lifetime of the network. Finally, we used Cooja simulations to validate the feasibility of the concept and to measure the latency gains that could be achieved compared to a centralized gateway processing solution on a reference grid topology scenario.

Our results show that NaC improves the distribution of energy consumption among the devices of the network, thus mitigating the energy hole effect and increasing the expected lifetime of the network. Furthermore, it is able to exploit the typical proximity of sensing and actuating devices to reduce latency compared to a cloud or edge-based solution. Finally, when running more realistic simulations in Cooja, our results show that data loss due to interference affects the centralized gateway approach just as much as (and in some cases more than) our proposed distributed approach.

Future work will focus on increasing the scalability of this approach, and to extend it to other forms of in-network processing that do not require a pre-mapping step on top of

a static topology, as this is a problem in scenarios with high mobility, such as for example smart vehicles.

ACKNOWLEDGMENT

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) and is co-funded under the European Regional Development Fund under Grant Number 13/RC/2077.

REFERENCES

[1] R. van der Meulen. (2015) Gartner says 6.4 billion connected things will be in use in 2016, up 30 percent from 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3165317

[2] P. Lancia and J. Sinnott. (2018) Qualcomm expands mesh networking platform to smart home products and launches new reference platform capabilities. [Online]. Available: https://www.qualcomm.com/news/releases/2018/01/08/qualcomm-expands-mesh-networking-platform-smart-home-products-and-launches-new

[3] (2015) R&I on IoT integration and platforms. [Online]. Available: http://ec.europa.eu/research/participants/portal/desktop/en/opportunities/h2020/topics/iot-03-2017.html

[4] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, February 2017.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[6] A. F. Liu, M. Ma, Z. G. Chen, and W. h. Gui, "Energy-hole avoidance routing algorithm for wsn," in *2008 Fourth International Conference on Natural Computation*, vol. 1, Oct 2008, pp. 76–80.

[7] A. Giridhar and P. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, 2006.

[8] S. S. Haykin, *Neural Networks and Learning Machines*. Pearson Education Upper Saddle River, 2009, vol. 3.

[9] N. Khude, A. Kumar, and A. Karnik, "Time and energy complexity of distributed computation in wireless sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 4, 2005, pp. 2625–2637.

[10] L. Ying, R. Srikant, and G. E. Dullerud, "Distributed symmetric function computation in noisy wireless sensor networks," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4826–4833, 2007.

[11] P. Vyavahare, N. Limaye, and D. Manjunath, "Optimal embedding of functions for in-network computation: Complexity analysis and algorithms," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2019 – 2032, 2015.

[12] V. Shah, B. K. Dey, and D. Manjunath, "Network flows for function computation," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 4, pp. 714–730, 2013.

[13] L. Reznik and G. Von Pless, "Neural Networks for Cognitive Sensor Networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2008, pp. 1235–1241.

[14] E. De Coninck, T. Verbelen, B. Vankeirsbilck, S. Bohez, P. Simoens, P. Demeester, and B. Dhoedt, "Distributed neural networks for internet of things: the big-little approach," in *Internet of Things. IoT Infrastructures: Second International Summit, IoT 360 2015, Rome, Italy, October 27-29, 2015, Revised Selected Papers, Part II*. Springer, 2016, pp. 484–492.

[15] S. Leroux, S. Bohez, T. Verbelen, B. Vankeirsbilck, P. Simoens, and B. Dhoedt, "Resource-constrained classification using a cascade of neural network layers," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–7.

[16] "http://on-demand.gputechconf.com/gtc/2015/presentation/S5813-Nobuyuki-Ota.pdf."

[17] Y. Sheng, J. Wang, and Z. Zhao, "A communication-efficient model of sparse neural network for distributed intelligence," in *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2016, pp. 515–520.

[18] Y. Liu, Y. Yang, C. Liu, and Y. Gu, "Forest fire detection using artificial neural network algorithm implemented in wireless sensor networks," *ZTE Communications*, pp. 16–20, May 2015.

[19] E. De Coninck, T. Verbelen, B. Vankeirsbilck, S. Bohez, S. Leroux, and P. Simoens, "DIANNE," in *Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT - M4IoT 2015*, New York, New York, USA, 2015, pp. 19–24.

[20] J. Li and G. Serpen, "Adaptive and Intelligent Wireless Sensor Networks through Neural Networks: an Illustration for Infrastructure Adaptation through Hopfield Network," *Applied Intelligence*, mar 2016.

[21] N. Kambhatla, D. Kanevsky, and W. W. Zadrozny, "US6418423 B1-Method and Apparatus for Executing Neural Network Applications on a Network of Embedded Devices," 2002.

[22] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive internet of things: A new paradigm beyond connection," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 129–143, April 2014.

[23] R. Zhang and J. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *International Conference on Machine Learning*, 2014, pp. 1701–1709.

[24] B. McBeath, "Distributed Intelligence in the IoT," Oct 2014. [Online]. Available: http://www.clresearch.com/research/detail.cfm?guid=4E088DB8-3048-79ED-9964-69B73742171B

[25] N. Kaminski, I. Macaluso, E. Di Pascale, A. Nag, J. Brady, M. Kelly, K. Nolan, W. Guibene, and L. Doyle, "A Neural-Network-Based Realization of In-Network Computation for the Internet of Things," in *2017 International Conference on Communications (ICC)*. IEEE, may 2017.

[26] J. Ko, A. Terzis, S. Dawson-Haggerty, D. E. Culler, J. W. Hui, and P. Levis, "Connecting low-power and lossy networks to the internet," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 96–101, April 2011.

[27] A. Brandt, J. Vasseur, J. Hui, K. Pister, P. Thubert, P. Levis, R. Struik, R. Kelsey, T. H. Clausen, and T. Winter, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550, Mar. 2012. [Online]. Available: https://rfc-editor.org/rfc/rfc6550.txt

[28] M. Lee, R. Zhang, C. Zhu, T. R. Park, C.-S. Shin, Y.-A. Jeon, S.-H. Lee, S.-S. Choi, Y. Liu, and S.-W. Park, "Meshing wireless personal area networks: Introducing ieee 802.15. 5," *IEEE communications magazine*, vol. 48, no. 1, 2010.

[29] J. Li and P. Mohapatra, "An analytical model for the energy hole problem in many-to-one sensor networks," in *IEEE vehicular technology conference*, vol. 62, no. 4. IEEE; 1999, 2005, p. 2721.

[30] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[31] C. P. Kruger and G. P. Hancke, "Implementing the Internet of Things vision in industrial wireless sensor networks," in *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*. IEEE, 2014, pp. 627–632.

[32] S. Andreev, O. Galinina, A. Pyattaev, M. Gerasimenko, T. Tirronen, J. Torsner, J. Sachs, M. Dohler, and Y. Koucheryavy, "Understanding the IoT connectivity landscape: a contemporary M2M radio technology roadmap," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 32–40, 2015.

[33] Moteiv Corporation. (2006) Tmote sky datasheet. [Online]. Available: http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

[34] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, Nov 2006, pp. 641–648.

[35] A. Dunkels, F. Österlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *Proceedings of the 5th international conference on Embedded networked sensor systems*. ACM, 2007, pp. 335–349.