Original Article

Enabling distributed intelligence assisted Future Internet of Things Controller (FITC)

Hasibur Rahman *, Rahim Rahmani

Department of Computer and Systems Sciences (DSV), Stockholm University, Nod Building, SE-164 55 Kista, Sweden

ABSTRACT

The unprecedented prevalence of ubiquitous sensing will revolutionise the Future Internet where state-of-the-art Internet-of-Things (IoT) is believed to play the pivotal role. In the fast forwarding IoT paradigm, hundreds of billions of things are estimated to be deployed which would give rise to an enormous amount of data. Cloud computing has been the prevailing choice for controlling the connected things and the data, and providing intelligence based on the data. But response time and network load are on the higher side for cloud based solutions. Recently, edge computing is gaining growing attention to overcome this by employing rule-based intelligence. However, requirements of rules do not scale well with the proliferation of things. At the same time, rules fail in uncertain events and only offer pre-assumed intelligence. To counter this, this paper proposes a novel idea of leveraging the belief-network with the edge computing to utilize as an IoT edge-controller the aim of which is to offer low-level intelligence for IoT applications. This low-level intelligence along with cloud-based intelligence form the distributed intelligence in the IoT realm. Furthermore, a learning approach similar to reinforcement learning has been proposed. The approach, i.e. enabling a Future IoT Controller (FITC) has been verified with a simulated SmartHome scenario which proves the feasibility of the low-level intelligence in terms of reducing rules domination, faster response time and prediction through learning experiences at the edge.

1. Introduction

Future Internet is expected to be driven by the prevalence of Internet of Things (IoT) where it is envisioned that *anything* can be connected [1]. The hype around IoT is that it is the next technological revolution of the current world [2] where hundreds of billions of things will be interconnected. IoT has started to shape into reality from its hype by and large due to recent advancements in ubiquitous technologies such as Radio Frequency Identification (RFID)/Near Field Communication (NFC), Wireless Personal Area Network (WPAN), high speed communication (4G/5G), Bluetooth Low Energy (BLE), etc. Advanced developments in the sensing and actuating technologies also contribute to the rise of the IoT popularity. This rise in connected things has already taken its

number beyond current world's population and expected to impact every aspect of human life. Currently, there are almost two connected things for every human. The ratio is expected only to accelerate in the coming days. The challenge of collecting and sharing the context information (ConIn) from these connected things has been addressed in earlier research [3–8]. The challenge has been addressed by architecting IoT platforms via mostly middleware solutions. Each middleware solution addresses different IoT challenges; for example, device management, context information collection and sharing, context-awareness, interoperability, etc. [4]. However, there is no single middleware solution or IoT platform that solves all these IoT challenges. An ideal IoT platform capable of providing solutions to all IoT aspects has not yet been designed [4]. Furthermore, most of the IoT platforms solutions are cloud centric [3–5,8]; recently Cisco coined the term fog computing, i.e. edge computing closer to the actual devices [9].

Lately resource constrained devices such as SmartDevices and raspberry pi have enriched in computational capabilities and at the same time price has become more affordable. These devices have the potential to be exploited as IoT gateways and have already been demonstrated in earlier research [10,11]. Emergence of these devices paves the way for computing at the edge of

* Corresponding author.
E-mail addresses: hasibur@dsv.su.se (H. Rahman), rahim@dsv.su.se (R. Rahmani).
Peer review under responsibility of King Saud University.

connected things, for example, raspberry pi (Raspberry Pi 3 is a credit card-sized computer with 1.2 GHz quad-core CPU, 1 GB RAM, built-in support for BLE and Wireless LAN, and priced at US $35 [12]) can be employed as an IoT gateway for smart home, mHealth, smart farming, factory automation, shipping, etc. Most of earlier research more or less agree on the typical three-layer IoT architecture as depicted in Fig. 1 [3–5,13]. It shows IoT application where things are connected to a gateway locally and gateway then collects and forwards data to the cloud for further processing which is also shown in the industry for example by AWS IoT [14]. This brings high latency and bandwidth requirements. However, IoT necessitates latency, i.e. response time as low as possible. When IoT was first coined by Kevin Ashton back in 1999, it was proposed in the context of supply chain management [15]. Due to the technological advancements and progressions in the research within IoT have evolved its vision and transformed the way Internet-enabled things are being utilized. IoT vision has been expanded to many other application domains. Such expansion of scopes drives IoT on the verge of experiencing a paradigm shift towards enabling Internet of Everything (IoE) [16]. The focus of which is the integration of people, things, services, context information as seen in Fig. 2. To counter this paradigm shift, new approaches are mandated to research into. Most of the IoT applications require real-time and quick decision making; hence, edge computing proposal is gaining growing attention from researchers recently [9,17–20]. At the same time, cloud computing cannot be ignored since edge computing can only offer solutions for limited data locally [18]- this data is also known as small data [21]-and a future IoT solution need to cater for both small data locally and big data globally [18]. Hence, an IoT solution is mandated to offer both edge and cloud solution. Therefore, this paper proposes a novel IoT solution offering both edge and cloud computing; thus, enabling a Future Internet of Things Controller (FITC).

Properties of future IoT, i.e., IoE, correspond to a cycle as seen from Fig. 2. This cycle can be compared with the vision of autonomic computing loop [16]. The autonomic loop executes self-X algorithms in order to make an autonomous entity. Execution of self-X algorithms depend on the policies embedded into a controller by a human-administrator. Whenever a policy needs to be modified or added or removed, human-administrator usually is consulted. This can also be seen from AWS IoT's rule-engines where rules (policies) are added whenever required and further in XpertRule [22]. However, in the IoT scenario where each controller might control thousands of things, updating rules or policies should be done automatically (as to reduce time and complexity) preferably by learning from the past experiences. Recently big players in the industry like IBM, Amazon, and Microsoft have started to tie up machine learning with IoT. This tying up is expected to offer unprecedented impact on IoT. In future, IoT and Artificial Intelligence (AI) -thereby, machine learning- will be inseparable. The reason is that up until now IoT only focused on collecting and sharing raw-data, it did not focus on providing insight to the raw-data. Existing approaches for edge intelligence are heavily reliant on predefined rules and are time consuming where in order to define new rules a person is reliant on cloud-based intelligence. Therefore, to counter the shortcomings of the current and previous approaches, this paper proposes to provide intelligence at the edge where IoT controller would learn from the past experiences. The proposal proposes an IoT controller to provide three important operations: Decision making, Action, and Prediction (DAP) for the connected things at the edge. The idea is to provide low-level intelligence to the small data at the edge before providing high-level intelligence for the so-called big data at the cloud; thus, enabling distributed intelligence. The need for two-level intelligence was also highlighted earlier in [4,18]. Providing intelligence at edge implies reaping value from the collected raw-data by the gateway, i.e. an IoT controller (this paper regards a device as gateway which only collects and forwards raw-data, and in addition to this when a device provides the portrayed low-level intelligence is regarded as *controller*). However, raw-data collected from IoT applications do not actually provide any usefulness unless insight is harvested [23]. To reap value from the raw-data, the data need to put into context to provide intelligence for converting into knowledge [3,23]. Currently, only rules are employed to provide edge-intelligence. However, as the number of connected things escalate the required number of rules also rockets. Depending only on rules could break the intelligence if new or uncertain events occur. Furthermore, pre-defined rules' domination would only provide pre-assumed intelligence. In order to provide further intelligence to improve the performance of an IoT controller mandates the controller to learn from the experiences by employing machine learning algorithms. In light of the above, the followings can be considered as vital challenges to provide edge-intelligence: contextualization of the raw-data; minimizing dependency only on rules for executing tasks; improving performance of tasks through experiences, i.e. learning; predicting an outcome in the event of uncertainties; efficient routing of the data; self-organization of the things, i.e. sensors and actuators;
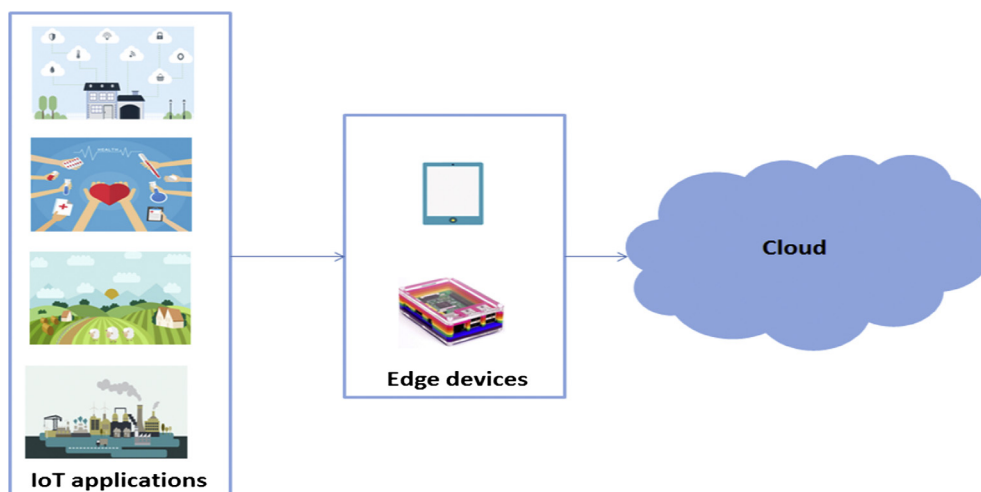

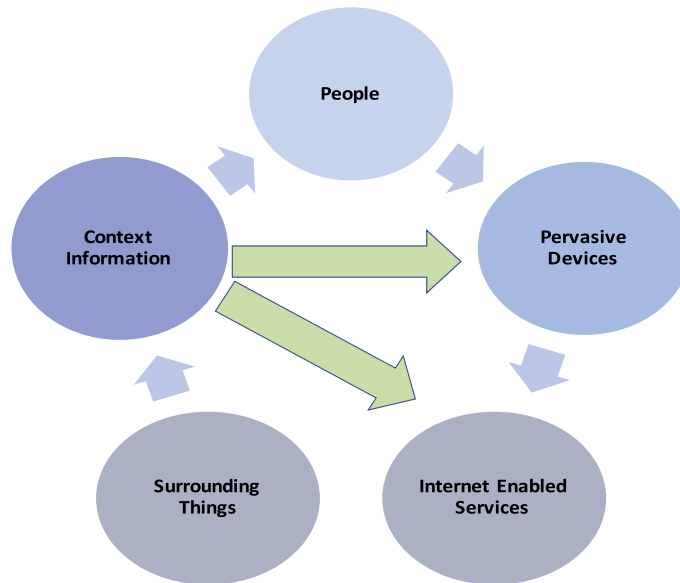**Fig. 1.** A typical simplified three-layer IoT architecture.

Fig. 2. Properties of Internet-of-Everything.

etc. With regard to such challenges, this paper further proposes a novel approach of employing belief-network (also popularly known as Bayesian network) to reduce dependency on the rules and learning algorithm similar to the reinforcement learning for IoT edge controller. Usefulness of such belief-network, i.e. probabilistic approach to learn from experiences was earlier discussed in [23].

The paper does not propose to get rid of employing cloud-intelligence altogether, rather it proposes the novel idea of reaping value from both edge and cloud, thus enabling distributed intelligence (see Fig. 3). This implies providing *intelligence of things* by reaping the *information of things* closer to the devices. In order to demonstrate the feasibility of the proposed approach, two IoT applications namely SmartHome and SmartFarming have been used for references and further, SmartHome application was exploited for verifying the algorithms for providing intelligence at the edge. In order to execute the DAP; the controller necessitates having algorithms that can help in executing these operations. The paper proposes and develops new algorithms with regard to providing intelligence at the edge. Further, the advantage of belief-network over rule-based is investigated followed by results of prediction for simulated SmartHome data.

2. Related work

Making sense of IoT data, that is, to reap value from the IoT data is the current topic and challenge of IoT research. IoT started with the vision of connecting any physical objects to the Internet and collecting data from these physical objects. Therefore, the earlier proposals [5–8,24] of analysing and taking decisions at the cloud made sense since initial objective was to collect data from the physical objects about real world by employing things rather than relying only on human-entered data on the Internet. The objective was simple: to collect data and let computers or higher-level devices take decisions by analysing data. Today the objective lies in harvesting value from the collected data. However, cloud centric solutions fail to provide low-latency which is one of the IoT requirements [9,13,18,19]. IoT further requires computing to be done as closer to the things as possible. Both cloud and edge computing provide advantages that are useful and solve many challenges in IoT paradigm. For example, cloud computing can offer large storage, complex processing [3,11]; analytics and visualization tools [3,11,24]; anywhere access [3], contextualization/personalization [25]; publish/subscribe [7]; and edge computing can offer faster processing [9,10]; better sustainability and energy efficiency [10,13,20]; distributed computing [18]; data pre-processing and filtering [13,19]; mobility support [18]; heterogeneity and interoperability [18]; computing closer to the actual things [13,18], etc. Besides these, edge computing reduces burden on cloud computing in terms of data storage, bandwidth, geographic coverage, analytical dependency, communication overhead, etc. [8,18,19]. Earlier papers showed advantages of edge computing over cloud computing in IoT in terms of: efficiency in energy, data filtering, providing notifications in a SmartHealth application [13];
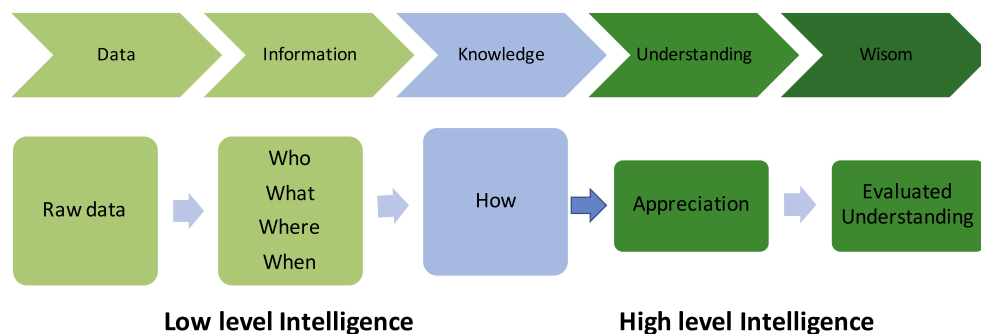

Fig. 3. Distributed (two-level) intelligence for IoT.

latency improvement [17]; synchronization and upload delay, communication overhead improvement [19]; better dynamic scaling and communication cost via mobile fog computing [26]; but none of the papers focused on providing intelligence for decision making and actions, and learning for prediction based on collected raw-data. To counter the challenge of intelligence at the edge, a parallel work in [13] demonstrated reinforcing gateways with system intelligence specific to the e-health scenario. However, that work did not focus on providing intelligence and learning based on collected data which this paper has demonstrated. Some of the solutions such as AWS IoT [14] and XpertRule [22] provide intelligence by employing predefined rules. Both solutions allow new rules to be defined aided by human. A person can define new rules based on analysed data from cloud-centric intelligence by employing AI techniques. Analysed data could be a classification or prediction problem, extract meaningful information from the raw-data [27], etc. However, these approaches are heavily reliant on predefined rules and are time consuming where in order to define new rules a person is reliant on analysed cloud-based intelligence. Moreover, rules fail to scale well with the increase in connected things. Furthermore, to overcome pre-assumed intelligence and in order to provide further intelligence to improve the performance, IoT controller mandates to learn from the experiences by employing machine learning algorithms. Belief-network can be used to learn from experiences as earlier discussed in [23]. However, most of the machine learning algorithms take time to learn, and additionally it requires training data meaning data needs to be readily available prior to applying the learning techniques [23]. Belief-network is gaining attention lately in AI research for prediction, learning, decision-making, etc. It is conceptually simple yet emerging as principal approach to learn through experiences [23]. On the other hand, reinforcement learning allows an agent (decision-maker and learner) to learn by giving reward for each action perceived from the environment [28]. This kind of learning can be applied to both low-level and high-level decision making and prior data is not needed for learning which makes it a competent technique for the IoT realm. Even though a lot of research has been conducted within IoT; research towards assisting IoT with distributed intelligence, employing belief-network in the edge computing to overcome dependency on rules, and learning have not yet been conducted.

3. Proposed approach

Internet-of-Things (IoT) is a network of things where things can be anything from a physical object to a virtual object. The primary objective of IoT is to connect anything anytime and anywhere. However, once things get connected via any path, challenge is to reap value from things' data, i.e. information. That corresponds to catering the *information-of-things*. Earlier many argued that things' capability of providing this information in the IoT domain makes the things smart; however, recently others argue that this capability only makes the things connected- not necessarily smart. In order to make things smart or make sense of information-of-things, providing intelligence is a prerequisite. Thus, it necessitates to provide *intelligence-of-things*. As inefficiency of cloud computing leads to edge computing; therefore, it is necessary to look into other solutions to counter data closer to the things which some researchers have started calling *small data*. Question is "How can a solution be employed that can provide intelligence by utilizing the *small data*?". This also corresponds to the earlier vision of two level intelligence in IoT [4]. Fig. 3 shows a resemblance of information-knowledge pyramid hierarchy. The hierarchy shows relation of the raw-data to ultimately providing wisdom based on the raw-data. In the middle of hierarchy lies the knowledge.

Knowledge also corresponds to the question of *how*. Therefore, the proposal in this paper would look into answering some of the questions related to this "how" the goal of which is to alleviate the intelligence-of-things. The IoT edge controller specifically would look at answering the questions based on the raw-data collected from the things at the edge. The edge controller, apart from answering the "how" questions, would also control other actions such as collecting raw-data from the connected things and converting the collected raw-data into information by answering the questions of *who*, *what*, *where*, *when*, etc. prior to converting into knowledge. The later action is also known as contextualization, i.e. each connected thing is contextualized by answering those questions. Based on this context information (ConIn), controller would provide knowledge to the connected things. For instance, in a SmartFarming scenario, the controller would collect raw-data from sensing devices, for example, about water condition and controller would first contextualize the raw-data into ConIn, and based on the knowledge about the ConIn, controller would take actions; an action could be determining when and how long to sprinkle water. Context information is further processed to execute the DAP. This is what edge controller has been proposed as to provide the low-level intelligence. Now based on the knowledge hierarchy, from raw-data to wisdom, below each of the steps corresponding to low-level intelligence is discussed.

3.1. Distributed intelligence

Raw-data: Raw-data is usually being in the form of symbols or signs in the context of knowledge hierarchy. In an IoT scenario, raw-data is any data that is collected from a thing, e.g. sensor or actuator. Raw-data is usually collected and fed forward by a gateway. There are many protocols that can be exploited by IoT to collect raw-data from the things. Some of the promising protocols are: MQTT, CoAP, DCXP, etc. [6,7]. The things can further be controlled via employing self-organizing algorithms such as presented earlier in [16]. This part of connecting and collecting raw-data is beyond the scope of this particular work. The next two steps are relevant to this particular paper, i.e. information and knowledge.

Information: As seen from Fig. 3, information corresponds to answering few of the fundamental questions. This also refers to contextualizing a thing. The collected data is fed into the contextualization algorithm (see Algorithm 1) which by answering the questions like: *who*, *what*, *where*, *when*, etc. provides more meaning to the raw-data. Here, *who* refers to thing's identity, *what* refers to the actual data, *where* refers to its origin, and *when* refers to its time of occurrence. There could be more context information added at this step for example its relation to other things and/or context information.

Knowledge: This step then provides further low-level intelligence based on the contextualized data. Although the figure shows knowledge as part of the low-level intelligence; in essence, knowledge would be distributed to both low-level and high-level intelligence. The reason being that the proposed IoT controller at the edge would run on resource-constrained devices which might not be able to provide all the required knowledge. High-level knowledge would come from more advanced computational devices. More insight to the data can be provided at the cloud by employing data mining techniques.

Understanding & Wisdom: These two operations are meant for operations like extracting meaningful information from data. These data mining challenges are beyond the scope of this paper and can be dealt at the higher level. Since understanding deals with appreciation of the data and wisdom is for evaluating the data which are analytical; and require all the previous steps and synthesizing knowledge from earlier knowledge and information. Wisdom on the other hand requires human intervention as this is the last

and highest level, so it extends the previous steps to infer new knowledge or understanding. A data scientist or a researcher perhaps would infer this new knowledge or understanding. Or in future deep learning based AI solution might be able to provide the required wisdom.

To understand the above steps, let's consider an IoT application in a SmartHome scenario where things such as sensors and actuators are installed for home automation. The sensors and actuators are controlled via an IoT edge controller. The controller collects data from the things and executes DAP. The following example shows how a temperature sensor reading can influence actions such as lighting, heating and making breakfast.

3.1.1. Example 1

Raw-data

Example raw-data from a sensor and actuators
Temperature sensor: 2000
Lighting actuator: 0
Heating: 1
Breakfast (coffee-maker & toaster): 0

Information

Contextualized Temperature Sensor
Who: temperaturesensor1.bedroom@smarthome1
What: 2000/100 = 20 °C (warm)
When: 07:00 AM (morning)
Where: home
Which: heating, lighting
Contextualized Lighting (Actuator)
Who: lighting1.bedroom@smarthome1
What: 0
When: 07:30 AM
Where: home
Which: coffee-maker, toaster
Contextualized Coffee-maker (Actuator)
Who: coffeemaker.kitchen@smarthome1
What: 0
When: 07:30 AM
Where: home
Which: toaster

Knowledge
Above shows example of raw-data and context information from sensor and actuators in a SmartHome scenario. The context information also shows which things are related to each thing. Based on the available context information following knowledge, i.e. DAP can be used. Although there are existing SmartHome platforms that are built upon IFTTT (IF This Then That), e.g. SmartThings [29], CNET SmartHome [30], etc. SmartThings' IFTTT does not consider logical association between things, and often requires user intervention to trigger a task. Moreover, the intelligence is not provided at the edge rather the gateway is only used for connecting things, and for collecting and forwarding data. Temperature sensor at bedroom reading gives 20 °C (warm). Now following this reading, when the temperature at bedroom is above 20 °C the controller can make decision to turn off the heating and turn on the lighting. However, before triggering the action, the controller needs to take into account other related context information for example time of the occurrence of this temperature reading. If this temperature reading occurred at morning (around 07:00 AM), only then turn off heating and turn on the lighting (turning on the light could also depend on the motion sensor, i.e. if it detects any physical activity then turn on the light). This time also refers to the

weekdays of the week, during weekend the time might be set to a later time. The controller also has the knowledge that when this SmartHome user wakes up, she usually takes her breakfast 30 min after waking up. Following turning on the bedroom light, motion sensor detects movement inside the bedroom (confirming that she is now awake) and make decisions on preparing breakfast, e.g. toasts and coffee.

Now another scenario could be that the user returns home on weekdays around 18:00 and the controller knows that if the temperature is below 15 °C (cold) then user turns the heating on when she is at home or about to reach home. To know user's current location, controller consults with her mobile application on the SmartPhone. When the user is 15 min away or 10 km away, controller decides to turn the heating on. The controller knows she eats her supper around 19:30, so it turns the oven on at a desired temperature at 18:45 every day. Now considering a non-regular day when the user joins dinner at a friend's house or outside, the controller makes a prediction on whether or not the oven should be turned on. Here, this knowledge is not embedded into the controller. Therefore, controller learns not to take such usual actions when user is not around.

3.1.2. Example 2

Raw-data

Example raw-data from sensors and actuators
Light: 40
Temp: 18
Nutrition: 26
Moisture: 43
Soil moisture: 30

Information

Contextualized Nutrition Sensor
Who: nutritionsensor1.gateway2@adamsfarm
What: 26%
When: 03:00 PM
Where: Zone 2, Row 3, Column 4
Which: Fertilizer
Contextualized Soil Moisture Sensor
Who: soilsensor.gateway5@adamsfarm
What: 30%
When: 05:00 PM
Where: Zone 4
Which: Sprinklers

Knowledge
The example here shows a probable scenario in farming where sensors and actuators are used for automating tasks. Few of the such tasks are monitoring water, soil, temperature, nutrition and operating fertilizer machine, sprinklers, lighting, etc. For example, Edyn [31], an example of connected farming, enables healthy farming by allowing instant access to one's remote farm. While Edyn alerts its users when to water, fertilize, control light, etc. which is useful but user might be not connected to receive alerts and might risk of not being able to execute such tasks required for healthy plant-growth; therefore, it would have been fitting in the era of connected world to let the IoT controller decide when to execute a task at the farm. For example, a farm could employ different sensors for collecting data as shown above and the controller would control DAP execution if and when to trigger a task. The task could be to start water sprinkle if soil moisture is below the threshold (a value which is specific to application and implementation) but before starting the task, controller would check weather fore-

cast and based on the weather information it would decide how much, and if, water needs to be poured. Similarly, when the nutrition level drops below a desired value, the controller would try to execute the task of providing fertilizer; but fertilizing can also depend on the plant growth meaning if the plants are ready to harvest then there is no need to fertilize the plants.

3.2. Belief-network

To achieve the aforementioned knowledge execution, i.e. steps shown in example 1 and 2, the paper proposes the belief-network concept instead of only rules. More specifically, each of the context information (ConIn) would be assigned with a prior-belief. In Bayesian or belief network, prior-belief is always defined even before collecting data [23]. Following the collection of data, new belief is obtained. And based on the belief-calculation, a task would be executed. Bayesian machine learning belief calculation follows two simple rules of sum and product rule [23]. Based on this, the paper adapts the formula as outlined in Eq. (1). The equation shows how to calculate the probability of taking an action by the IoT edge controller. An action is usually taken based on belief of ConIn that affect the action. Therefore, the controller would first determine which of the ConIn belief are required to take an action. Following this, the controller would calculate the probability according to:

$$Pr(A) = \sum_{i=1}^{n} (\text{ß}_{CI} * \text{ß}_{CIV}) \tag{1}$$

Here, ß is the belief and Pr is the probability, ß_{CI} and ß_{CIV} stands for belief of ConIn and ConIn value respectively. For example, CI is the temperature sensor and CIV is the value of temperature sensor (e.g. cold). Once Pr is calculated then it is checked with a threshold value (e.g. a pre-defined value of 75% which can later be changed or learned), $\varnothing$, when $\varnothing$ exceeds Pr, new belief is established, as formulated in Eq. (2).

$$\text{ß}_{new} = (Pr > \varnothing) \tag{2}$$

Figs. 4 and 5 show such belief-networks for SmartHome and SmartFarming respectively where it illustrates which sensors and actuators affect which actuators. However, the prior-belief may not always be optimal since this is pre-defined; therefore, the paper proposed to exploit the reinforcement learning technique to learn belief through experiences which can then be used to update the prior-belief. This learning implies that whenever an

event occurs, it is given a reward (a numerical value). Fig. 6 illustrates how reinforcement learning-like approach is proposed for an IoT controller. The controller observes actions about a certain IoT application (e.g. SmartFarming) by interacting with the application, and increases frequency for each action in a context state (set of ConIn). An action here could be collecting ConIn for sensors or performing an actuation. Increasing frequency is comparable with giving reward as in reinforcement learning. Frequency is also increased for each specific ConIn at the time of different associated ConIn, e.g. frequency of cold temperature at different time of the day for different user locations. And when the frequency exceeds the threshold level, prior-belief is altered with new belief as formulated in Eq. (2). Furthermore, when one or more context value is missing from the set of ConIn, the missing value is predicted by analysing the learned experiences. This learning algorithm gives advantage over other machine learning algorithms with respect to the fact that no prior data is required and it learns from its experiences perceived from the environment.

4. Model

This section describes modelling of the proposed approach of this paper. The section first describes the workflow of the model and follows with describing the algorithms to achieve the goals set in this paper.

4.1. Workflow

The paper proposes to provide distributed intelligence to counter the influx of context information in the IoT domain by providing intelligence both at the edge and at the cloud as illustrated in Fig. 3. Edge-intelligence implies that intelligence based on raw-data collected by the IoT controller from the things would be provided as fast as possible. The first task of the proposed approach is to contextualize the collected raw-data. This is done by answering four fundamental questions of knowledge-hierarchy. Algorithm 1 deals with this, i.e. contextualization. Hence, the first task of the model is to check if a raw-data has been contextualized or not; if the data is already contextualized i.e. the questions are answered, the ConIn would be forwarded to determine a task after updating existing set of ConIn and frequency of each ConIn. Algorithm 2 demonstrates the determine task part, it then decides whether to activate an action based on the available thing's ConIn (TConIn) or forwards the ConIn to learn experiences. Algorithm 3 illustrates
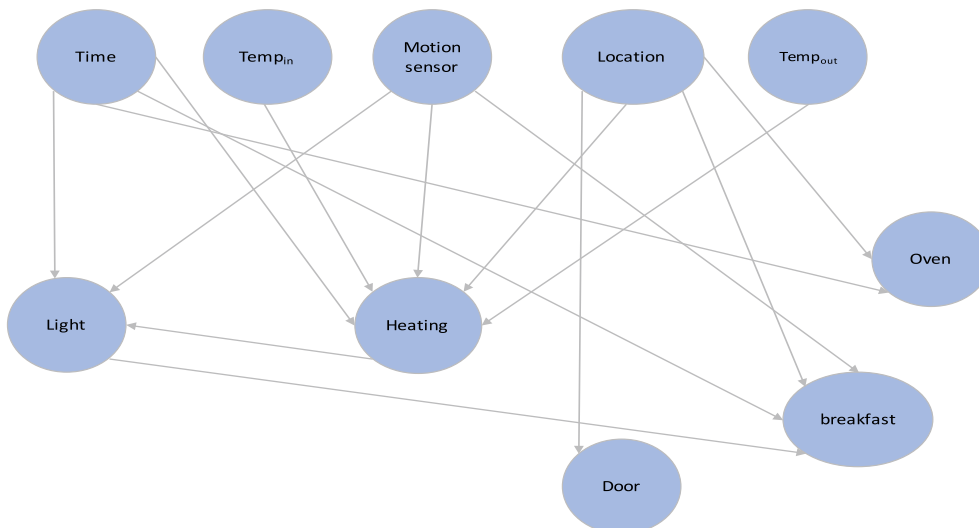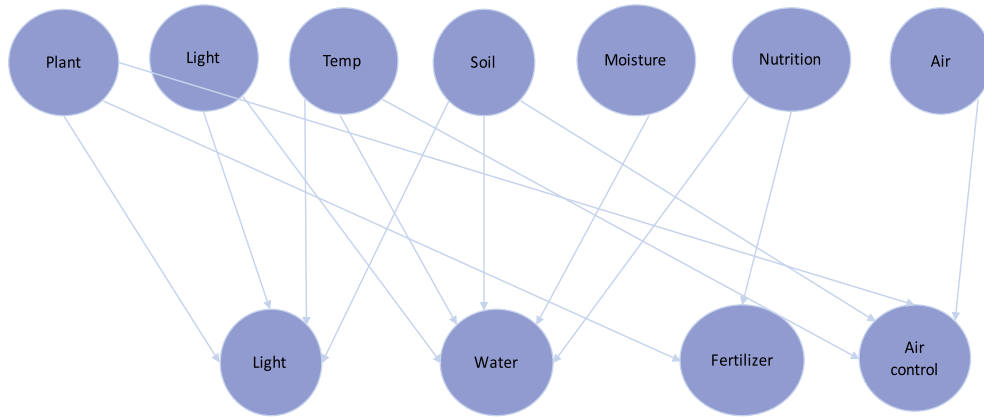

Fig. 4. Example Belief-Network for SmartHome.

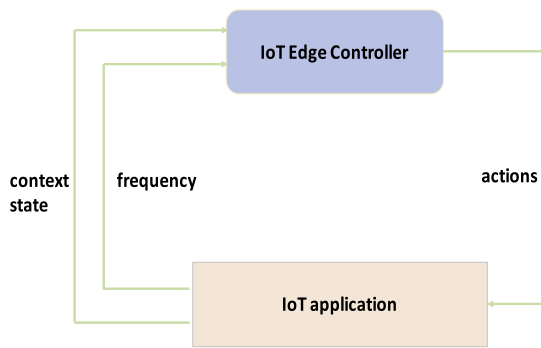**Fig. 5.** Example Belief-Network for SmartFarming.

Fig. 6. Learning experiences based on reinforcement learning.

learning experiences which are then forwarded for learning and/or prediction. Learning and/or prediction usually take place in scenarios when one or more ConIn is missing. In such cases, first associated ConIn (AConIn) is obtained which is along with the available ConIn then decides whether to predict or learn an action.

Algorithm 4 and 5 take care of learning and prediction respectively. Fig. 7 further demonstrates the workflow of the proposed approach to alleviate *intelligence of things* by harvesting *information of things* at the edge.

4.2. Algorithms

4.2.1. Contextualization

This, contextualization, algorithm deals with answering questions to provide more meaning to the raw-data from a connected thing. This also corresponds to the second step of the knowledge hierarchy as shown in Fig. 3. The algorithm takes the sensed raw-data as input and outputs the contextualized data. It also takes care of calculating frequency of each ConIn, frequency of each ConIn at given ConIn, and AConIn. These frequencies and AConIn are then exploited by other algorithms for carrying out the DAP. At this stage, ConIn are mapped to some other context information that it may affect, e.g. some actuation and occurrence of the ConIn is also saved for future uses, e.g. finding frequency.
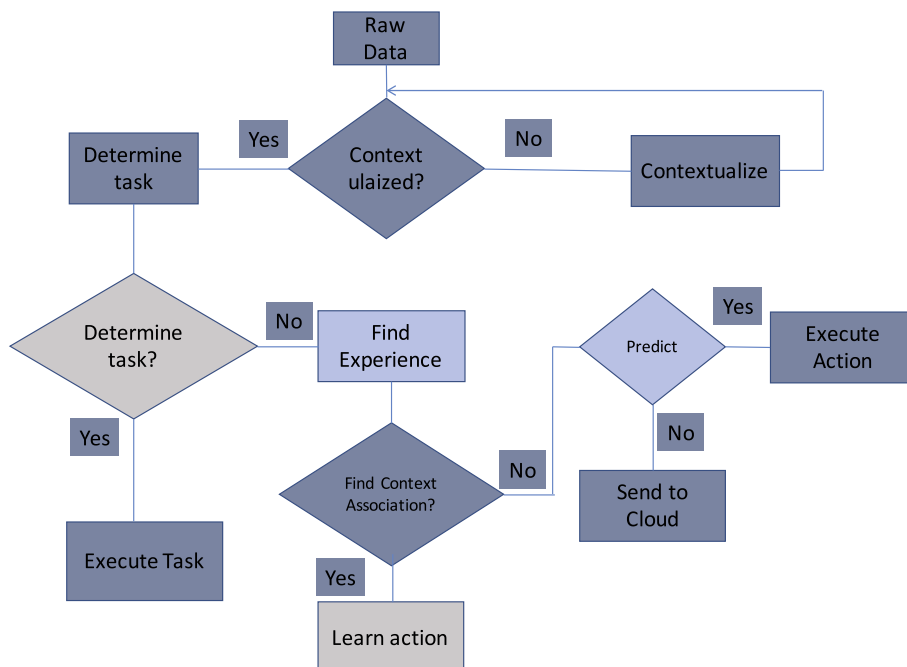

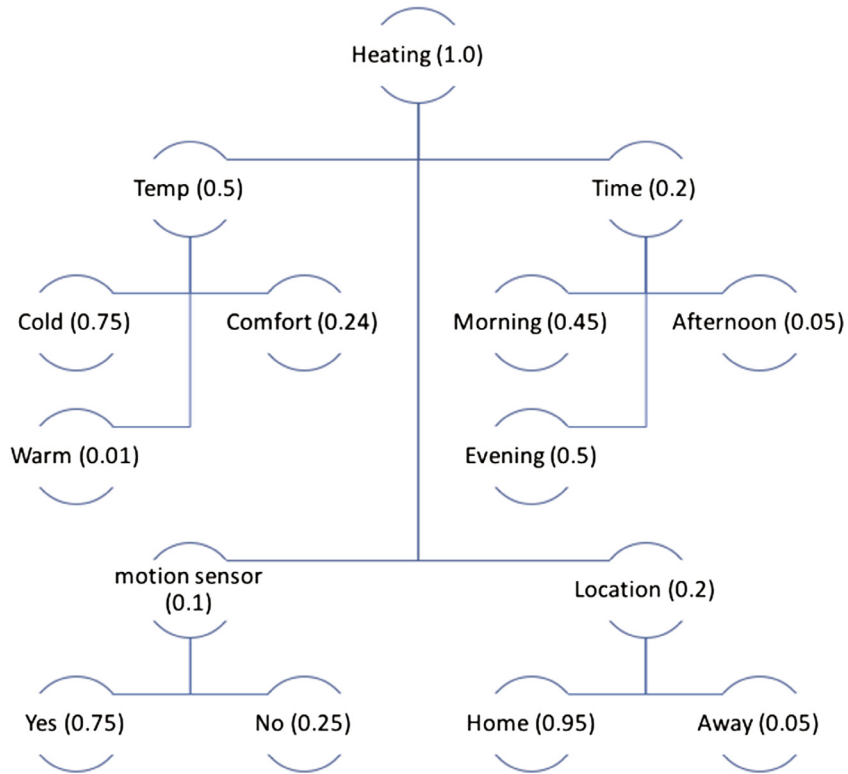**Fig. 7.** Workflow of the proposed approach.

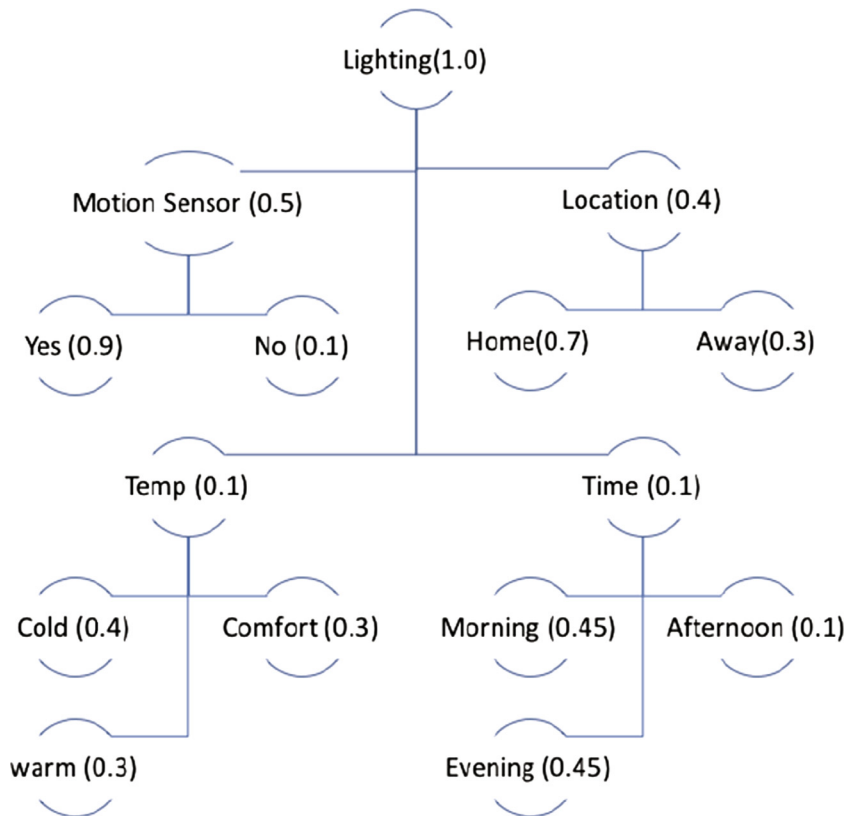**Fig. 8.** Probability, i.e. belief distribution for turning on heating.

Fig. 9. Probability, i.e. belief distribution for turning on a light.

4.2.2. Determine task

This algorithm is where based on ConIn a decision is made to execute a task or experience is obtained in case of missing ConIn. As the paper proposes employing belief-network for decision making; the algorithm first looks up to the prior-belief of each ConIn if set of thing's ConIn (TConIn) is not missing. The prior-belief of each ConIn is predefined; this can be defined at the cloud based on some data mining or other machine learning algorithms. At the initial stages of installing the IoT controller for a SmartHome, SmartFarming, e-Health, etc. applications, user could be involved to set the prior-belief based on individual preferences, i.e. each IoT controller could be personalized and personalization could be saved and initialized at the edge. Later on, the prior-belief can be altered by the controller based on experiences – through learning. As for this paper, prior-belief has been predefined which has been distributed among simulated ConIn (see Figs. 8 and 9). After retrieving prior-belief for each of the TConIn, probability for each task is calculated using Bayesian Theorem as shown in Section 3 and in [23]; and actions related to TConIn are executed when the calculated probability exceeds the pre-defined threshold value. Exploring belief-network at the controller is deemed efficient compared to using only rules (see Section 5.2 for more). Next, when one or more ConIn is missing from TConIn, the algorithm first fetches its AConIn. Once the AConIn is found, then TConIn and AConIn are fed into find experience algorithm for further analysis or prediction.

Algorithm 1 Contextualization

---
1. Initialize Connection/communication to collect data
2. **while** there is sensed data **do**
3.   **if** sensed data is not contextualized **then**
4.   what: raw-data
5.   where: origin of the raw-data
6.   when: time of occurrence
7.   who: originator of the raw-data
8.   **else if** sensed data is contextualized **then**
9.   find its frequency
10.   frequency at the given ConIn
11.   what other ConIn it is related to, e.g. some actuation
12.   update ConIn
13.   **end if**
14.   determineTask(TConIn);
15.   addToAConIn();
16.   addToTConIn();
17. **end while**

Algorithm 2 Determine Task

---
1. Fetch TConIn
2. **for** all TConIn **do**
3.   **while** TConIn is not empty **do**
4.   Look up the prior-belief for each ConIn
5.   Calculate probability of tasks related to TConIn (Eq. (1))
6.   **if** probability of task > threshold **then**
7.   Execute task(s), i.e. decision
8.   **end if**
9.   **end while**
10.   **while** TConIn is not empty **do**
11.   findAConIn(TConIn)
12.   predict (TConIn);
13.   **end while**
14.   findExperience(TConIn, AConIn);
15. **end for**

4.2.3. Find experience

This algorithm finds experiences of a thing in the IoT domain. It takes TConIn and AConIn as input and returns thing's experiences. This algorithm checks each of the ConIn and stores frequency for each tuple of ConIn (e.g. {what, when}, {what, when, where}, {what, when, where, who}). This frequency based on the experience later is used to find belief, learning and prediction - thereby, improve performance of tasks. For example, when one or more ConIn is missing, this tuple of context information's frequency is used to predict the missing ConIn. Further, the frequency based on experience is used to learn action for each set of ConIn and later prior-belief can be altered.

Once the edge controller has learned the experiences by counting frequency of each ConIn tuple and total frequency of each set of ConIn, this algorithm can learn from the experiences to take an action. This is similar to reinforcement learning where an agent learns from its environment which gives reward (a numerical value) for each action. Similarly, this algorithm also increases frequency for every action, and saves occurrence of every ConIn tuple as illustrated in Fig. 6. This reward or frequency is then used to calculate belief of an action. Once the algorithm has calculated belief of certain ConIn, it checks with a threshold value (the choice of value could be determined at the cloud by human-expertise, by exploring machine learning technique, by the user during personalization) and when check returns true, it helps the controller to learn actions related to this ConIn. However, if belief fails to meet the desired threshold value, it can then decide whether to send to cloud for high-level action or no action is required to trigger. Algorithm 4 describes this.

4.2.4. Prediction

This algorithm is employed to predict missing values in a set of ConIn. This algorithm takes TConIn as input with missing values, and finds other ConIn (OConIn) for the available ConIn. Following this, the algorithm finds frequency for each OConIn. It then checks if the frequency is the highest among OConIn frequencies; when the frequency is the highest frequency it identifies as the most probably missing ConIn. For example, if the algorithm is given a tuple such as {when, who, where}, the OConIn for this tuple would be what; now assuming that what has the four values {very cold, cold, comfort, warm}, the algorithm would fetch probability (from the OConIn frequencies) for each of what value. And if cold has the maximum probability, e.g. 45%, the algorithm would predict cold as the missing ConIn.

Algorithm 3 Find Experiences

---
1. Fetch TConIn
2. **for** each what (sensed data) **do**
3.   **for** each when (time) **do**
4.   add to tuple
5.   increase frequency
6.   **for** each where (location) **do**
7.   add to tuple
8.   increase frequency
9.   **for** each who (originator) **do**
10.   add to tuple
11.   increase frequency
12.   find AConIn
13.   add to tuple (TConIn, AConIn) to learn experiences
14.   learnBelief(TConIn, AConIn);
15.   **end for**
16.   **end for**
17.   **end for**
18. **end for**

Algorithm 4 Learn Belief

1. Fetch TConIn, AConIn
2. **for** i = size of TConIn **do**
3.   **for** j = size of AConIn **do**
4.   **if** TConIn(i) is associated with AConIn(j) **then**
5.   fetch frequency of (TConIn(i), AConIn(j))
6.   calculate probability of (TConIn(i), AConIn(j))
7.   **if** probability > threshold (Eq. (2)) **then**
8.   new belief is obtained
9.   **end if**
10.   **else if**
11.   forward for higher level action (to cloud)
12.   **end if**
13.   **end for**
14. **end for**

Algorithm 5 Prediction

1. **for** each available tuple of ConIn **do**
2.   find OConIn for this tuple
3.   **for** each available tuple of ConIn **do**
4.   fetch frequency
5.   **if** frequency of OConIn equals highest frequency **then**
6.   ConIn predicted
7.   **end if**
8.   **end for**
9. **end for**

5. Results & discussions

This section demonstrates the evaluation of the concept mentioned in the earlier sections. This paper has employed simulations as experiments [32,33] for verifying feasibility of the proposed approach. Simulations generate new data about empirical systems [32] which is also a kind of experiment and allow to carry scientific inquiry of a model with less-cost and more quickly [33]. The model has been developed in Java and the developed model has been tested on a raspberry pi 2 (Model B) [12] to observe performances since the paper mentioned it as a suitable candidate for edge controller. By consulting earlier research [11], data about a SmartHome has been generated which is randomized in each simulation. These simulated data are later fed into the developed model to generate new data to verify the feasibility of the model. First of the objectives is investigation of employing belief-network instead of only rules where it is shown how such belief-network can help in reducing rule domination and obtaining faster response time. Subsequently, the results of applying belief-network and reinforcement learning to predict or learn actions have been portrayed. Each simulation has been run for 100 times except for prediction part which has been run few hundred times; and mean values for each simulation are reported. Evaluation of the algorithms can be seen in the following sub-sections.

5.1. Simulation scenario

In order to verify the feasibility of the proposed approach in this paper, the algorithms are verified by simulated ConIn values. Simulation scenario for each of the DAP is presented in this sub-section where decision is made based on the prior-belief of each ConIn. In the following example of a SmartHome scenario, assuming that

Table 1
Advantages of Belief-Network over rule-based.

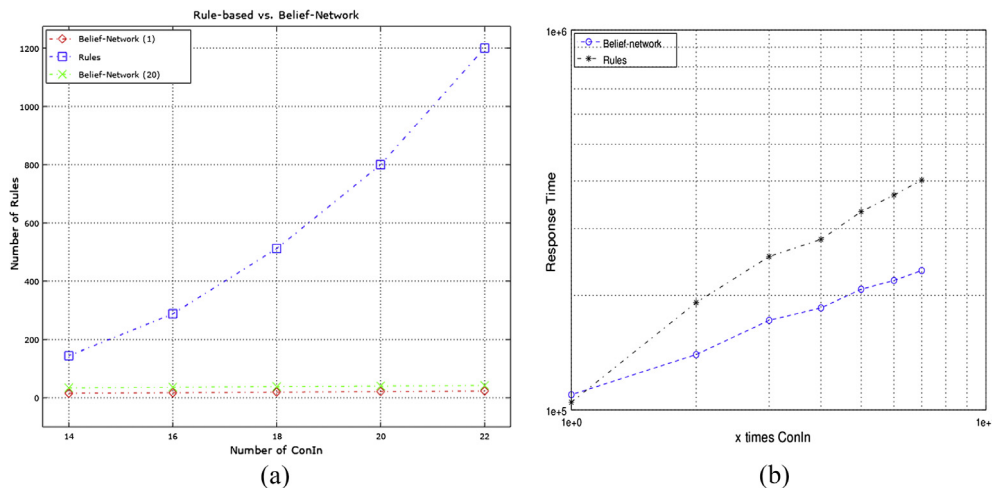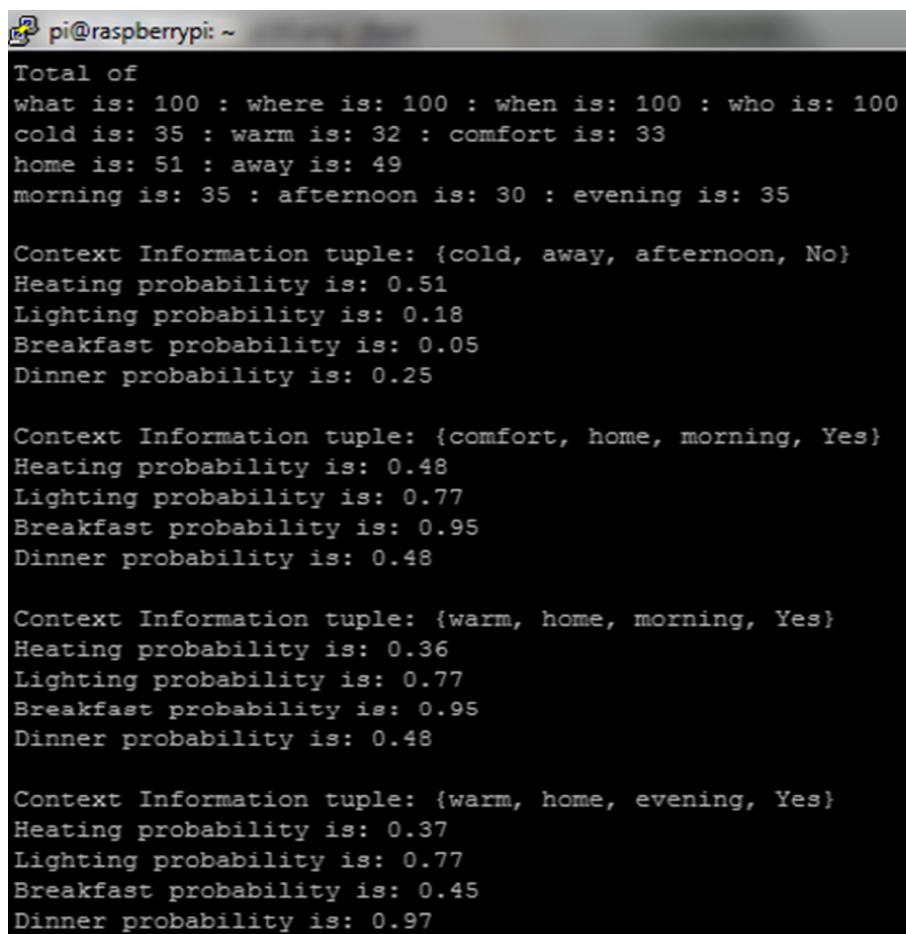| What	When	Motion	Location	Heating	Lighting	Total ConIn	Increase in ConIn	Total rules	% increase in rules
3	3	2	2	2	2	14	-	144	-
4	3	2	3	2	2	16	2	288	100
4	4	2	4	2	2	18	4	512	255
5	5	2	4	2	2	20	6	800	456
6	5	2	5	2	2	22	8	1200	733

Fig. 10. (a) Rules increase; (b) response time.

```
if temp is cold
    tempPBHeating = temperaturePBHeating * coldPBHeating;
else if temp is comfort
    tempPBHeating = temperaturePBHeating * comfortPBHeating;
else if temp is warm
    tempPBHeating = temperaturePBHeating * warmPBHeating;
end if
if time is morning
    timePBHeating = timeOfOccurencePBHeating * morningPBHeating;
else if time is afternoon
    timePBHeating = timeOfOccurencePBHeating * afternoonPBHeating;
else if time is evening
    timePBHeating = timeOfOccurencePBHeating * eveningPBHeating;
end if
if location is home
    locationPBHeating = currentLocationPBHeating * homePBHeating;
else if location is away
    locationPBHeating = currentLocationPBHeating * awayPBHeating;
end if
if mSensor is yes
    mSensorPBHeating = motionSensorPBHeating * mSensorYesPBHeating;
else if mSensor is no
    mSensorPBHeating = motionSensorPBHeating * mSensorNoPBHeating;
end if
heatingProb = tempPBHeating + timePBHeating + locationPBHeating + mSensorPBHeating;
//Rule for taking action based on the above prior beliefs
if heatingProb is greater than threshold (e.g. 70%)
    activateHeating();
end if
```

Fig. 11. Pseudo-code for turning on heating in belief-network.

Fig. 12. Simulated ConIn examples and probability for actuations.

there are four actuators, e.g. heating, lighting, making breakfast and dinner (could be more for example: door, robot cleaner, window blinds control, etc.), probability for each of the actuation is 1.0 (since probability cannot be over 1.0) which is then distributed among the ConIn that affect the actuation. For instance, activation of heating is dependent mainly on temperature, user's current location, time and motion sensor. Of these 4 ConIn, temperature affects heating mostly followed by user's location, time and motion sensor. Therefore, temperature is given 0.5 of probability followed by 0.2 to both location and time, and 0.1 to motion sensor. Motion sensor gets low probability because user might be at home but no physical activity (e.g. sleeping) is detected. However, temperature can be further divided into many sub-categories (e.g. cold, comfort, warm; or low, medium, high) and each of these sub-categories would then be given prior-belief. Obviously cold or low temperature affects heating more than any other ConIn, so this ConIn would get higher belief compared to others. Figs. 8 and 9 show the prior-belief distribution for heating and lighting actuation respectively. However, the prior-belief that is distributed among the ConIn may not be optimal, optimal belief distribution may be done by employing other type of learning techniques at the cloud as part of high-level intelligence or during personalization as mentioned earlier.

5.2. Belief-network vs. rule-based solution

As the number of connected things increase in IoT so does the volume of context information. This high influx of ConIn would require a large number of rules to execute a certain task. For instance, in a SmartHome scenario as shown earlier in example 1 (Section 3.1.1), if we want to apply rules for events related to a temperature sensor at bedroom, where it is assumed that sensed temperature values are grouped in different ranges (e.g. cold, comfortable, warm (3)), and after contextualizing the temperature values with AConIn (*location*: home, away (2); *time*: morning, afternoon, evening (3); *motion*: yes,

no (2)) would require ($3 \times 2 \times 3 \times 2 =$) 36 rules to activate any action comparing every conditions based on these ConIn. Now if there are 3 more ConIn are added, the required rules would become 108 (= $3 \times 3 \times 4 \times 3$) - a whooping 200% increase in the rules. This gives an idea about how depending only on the rules for intelligence would be impractical for thousands, or even more, of things for a particular controller. Table 1 and Fig. 10(a) further demonstrate this dependency on the rules.

From Fig. 10(a) and Table 1, it is seen that a total of 1200 rules might be required for a total of 22 ConIn, whereas for ConIn of 14 the rule shrinks to 144. This is roughly 733% increase in the number of rules requirement only for 8 ConIn increase. The requirement of number of rules for every condition to be met follows the product rule of thing's ConIn. This can be obtained by the following formula:

$$N_r = \prod_{i=1}^{n} N_{CI_i} \tag{3}$$

N_r = Total number of rules, N_{CI} = Number of ConIn, n = total number of context state

Now if belief-network, i.e., prior-belief, is employed instead of rules for each of the ConIn as shown in earlier section, the requirement for rules decrease drastically. The total number of rules needed by assigning prior-belief for each ConIn is equivalent to the total of number of ConIn plus one rule for taking each action for combined belief as formulated in Eq. (4). Fig. 10(a) shows two results of belief-network with 1 and 20 actions to be taken, this suggests that even with increase in number of actions, rules for belief-network remain lower. Fig. 11 illustrates pseudo-code of rules for employing belief-network for taking an action. The pseudo-code also signifies the fact that the more ConIn are there, the more advantageous employing belief-network becomes.

$$N_r = \sum_{i=1}^{n} N_{CI_i} + N_A \tag{4}$$

```
pi@raspberrypi ~
ConIn tuple before prediction is: {home, bedroom, morning}
ConIn tuple: {cold, home, bedroom, morning} and Frequency: 997
ConIn tuple: {comfort, home, bedroom, morning} and Frequency: 675
ConIn tuple: {warm, home, bedroom, morning} and Frequency: 756
Probability for cold is: 0.41
Probability for comfort is: 0.28
Probability for warm is: 0.31
The predicted ConIn is: cold
ConIn tuple after prediction is: {cold, home, bedroom, morning}
Heating probability is: 0.73
Lighting probability is: 0.77
Breakfast probability is: 0.95
Dinner probability is: 0.48

ConIn tuple before prediction is: {away, living, afternoon}
ConIn tuple: {warm, away, living, afternoon} and Frequency: 831
ConIn tuple: {comfort, away, living, afternoon} and Frequency: 798
ConIn tuple: {cold, away, living, afternoon} and Frequency: 768
Probability for warm is: 0.35
Probability for comfort is: 0.33
Probability for cold is: 0.32
The predicted ConIn is: warm
ConIn tuple after prediction is: {warm, away, living, afternoon}
Heating probability is: 0.14
Lighting probability is: 0.18
Breakfast probability is: 0.05
Dinner probability is: 0.25
```

Fig. 13. Results of prediction for missing value (1).

N_A refers to the number of actions to be taken for N_{CI}.

Besides, belief-network demonstrates faster response time compared to only rule-based approach as illustrated in Fig. 10(b). The figure was plotted in logarithmic scale, and 'x' times of same ConIn was used to measure the response time (in nanosecond) on raspberry pi for both belief-network and rule-based approach. The result shows that belief-network and rule-based approach required almost same response time for the first set of ConIn when number of rules for both of these were almost same; however, as the number of ConIn increase belief-network required low response time compared to only rule-based approach. This response time is one of the significant performance metrics in edge computing and the proposed approach has demonstrated faster response time.

The example further signifies the advantages belief-network provides over only rule-based solution. The controller only needed 11 rules for 10 ConIn to turning on heating, whereas rule-based would have to define a rule for every condition for each ConIn which would make number of rules to be 36. However, belief-network requires to provide prior-belief which rule-based does not necessarily need to. This prior-belief distribution among ConIn can further be learned or predicted by utilizing concept similar to reinforcement learning as shown in the following sub-section.

5.3. Learning and prediction

The example shown above, i.e. prior-belief for each ConIn, works when all the ConIn are available. However, in a situation when one or more ConIn is unavailable, the IoT controller would be unable to make a decision since one or more of the beliefs to execute a certain task are missing. In such cases, the IoT controller would have to predict through learned experiences to take actions. Learning in this paper has been limited to experiences only where the IoT controller would learn frequency of each ConIn and ConIn tuple. However, learning an action based on ConIn and AConIn is also possible which has been left for future work. Fig. 12 shows results of running the model of the proposed approach assuming that no ConIn value is missing. This figure illustrates how collected raw-data is being contextualized (TConIn) and four such examples are shown. In the 3rd example, TConIn is {warm, home, morning, yes} which is interpreted as "temperature is warm in the morning at home and the user is awake (motion sensor detected a movement)". Probability for each actuation is calculated using Eq. (1) and presented in the figure. Heating probability is 0.36 for the 3rd ConIn tuple which indicates that heating does not need to be activated, even though user is at home but temperature is warm; lighting probability is high at 0.77 which indicates light should be turned on. The actuations that would be activated based on the following TConIn examples are: lighting and breakfast (2nd & 3rd TConIn); lighting and dinner (4th TConIn).

Fig. 13 shows results for prediction, after running few hundred more simulations, when a ConIn is missing. Only temperature value missing has been considered for proof-of-concept of the algorithms. For the example shown in Fig. 12, frequency and experience (algorithm 1 and 3) for each particular ConIn are stored which are subsequently used to predict the missing values. The first TConIn tuple in Fig. 13 shows that available ConIn tuple: {home, bedroom, morning}, here bedroom refers to the 'who' that is the identification of the context originator. The probability for missing temperature value is 0.41, 0.28 and 0.31 respectively for

Fig. 14. Results of prediction for missing value (2).

cold, comfort, and warm. Therefore, the prediction algorithm predicts "cold" as the most probable missing temperature for this particular ConIn tuple. Afterwards, these predicted values are used with the available ConIn and actions are taken against this ConIn tuple. For the first ConIn tuple, heating, lighting and breakfast actions are taken which seem an accurate predicted estimation since user is at home in the morning. Thus, turning on these actions are the most probable actions. The algorithm through the experiences learns to improve its tasks. However, the probability for missing value illustrated in Fig. 13 appear very close, the reasons being the randomness of the three-simulated value of temperature where after few hundred simulations each value might appear equally. Fig. 14 further shows results of prediction algorithm with five simulated value of temperature. The probability of each value in this particular case seems more spread. In either case, the prediction algorithm deems successful in predicting the most probable missing ConIn value and thereby improving the controller's performance in executing tasks. Furthermore, Figs. 13 and 14 also suggest that algorithms are successful in learning the frequencies, e.g. frequency for each ConIn, ConIn tuple at given ConIn.

6. Conclusions

Internet of Things, the next technological revolution in the current world, applications have so far been relied on cloud-based solutions for computations, analytics, etc. While the advantages offered by cloud-based solutions cannot be disregarded; the need for computations, analytics closer to the things, i.e. at the edge is gaining growing attention recently. Edge analytics are dominated by rules which are computationally expensive and fail to scale well with the ever-mounting number of things in the IoT paradigm which mandate to look into alternative solution such as AI based solutions. AI based solutions can also cater for uncertain and new events. The paper has presented an AI based distributed-intelligence assisted Future Internet of Things Controller (FITC) that utilizes both edge and cloud based intelligence. More specifically, edge controller is employed to provide low-level intelligence and cloud controller would provide high-level intelligence.

It is foreseen that the combination of IoT and AI is inseparable, and edge controller by utilizing AI techniques closer to the things would allow necessary edge analytics. Belief-network enables to make decisions based on prior-belief and it further can learn new belief based on probability. The proposed IoT edge controller further exploits capability similar to the reinforcement learning to learn experiences which can then be used for learning probability and making prediction. To the best of our knowledge, this paper is the first to propose such novel idea of employing belief-network and reinforcement learning at the edge of IoT. The feasibility of the proposed approach has been backed up by simulated SmartHome scenario on a raspberry pi as a proof-of-concept. Furthermore, mathematical formulas for obtaining probability, new belief and rules for belief-network have been presented in this paper. Results conveyed here suggest that employing belief-network considerably reduces requirement of rules and enables faster response time for the edge controller; it also allows learning new belief based on experiences. Making prediction is one of the important capabilities that was missing from earlier edge solutions which this paper has successfully achieved to make prediction of the missing ConIn. The paper has also presented algorithms to achieve the aforementioned capabilities. Further, an algorithm to provide meaning to the raw-data inspired by the information-knowledge hierarchy has also been presented. Given the evidences reported in this paper, the approach can be used in many IoT applications such as SmartHome, SmartFarming, SmartHealth, Waste Management, etc. which can be explored in the future.

Moreover, to complement the work presented in this paper, few of the work that can be done in future are as follows: further verifying the FITC performance by incorporating a cloud controller along with the presented edge controller; a distributed edge controller for large scale IoT applications such as traffic/environment management, SmartAgriculture, etc.; learning was limited to experiences and finding new belief in this paper and this could be extended to learning other actions such as when to sense data, activate an actuation, when to send data to cloud, etc.; providing multi-modal Context-Aware reasoNing (CAN) in order to employ edge controller for various IoT applications. It would also be interesting to investigate dynamic behaviour of the controller.

References

[1] H. Sundmaeker, P. Guillemin, P. Friess, S. Woelffle, Vision and Challenges for Realising the Internet of Things Technical Report, European Commission Information Society and Media, 2010.

[2] M. Ali Feki, F. Kawsar, M. Boussard, L. Trappeniers, The Internet of things: the next technological revolution, Computer 46 (2) (2013) 24–25, http://dx.doi.org/10.1109/MC.2013.63.

[3] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (IoT): a vision, architectural elements, and future directions, Future Gener. Comput. Syst. 29 (7) (2013) 1645–1660.

[4] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Context aware computing for the Internet of Things: a survey, IEEE Commun. Surv. Tutorials 16 (1) (2014) 414–454, First Quarter.

[5] OpenIoT: Open Source Solution for the Internet of Things into the Cloud, January 2012. <http://open-platforms.eu/library/openiot-the-open-source-internet-of-things/> (last accessed: January 2017).

[6] H. Rahman, R. Rahmani, T. Kanter, Enabling scalable publish/subscribe for logical-clustering in crowdsourcing via mediasense, in: IEEE Science and Information (SAI) Conference 2014, August 27–29, 2014, London, UK, 2014.

[7] A. Antonić, M. Marjanović, K. Pripužić, I.P. Žarko, A mobile crowd sensing ecosystem enabled by CUPUS: Cloud-based publish/subscribe middleware for the Internet of Things, Futur. Gener. Comput. Syst. 1 (2015) 1–16, http://dx.doi.org/10.1016/j.future.2015.08.005.

[8] A. Botta, W. de Donato, V. Persico, A. Pescapé, On the integration of cloud computing and internet of things, in: 2014 International Conference on Future Internet of Things and Cloud, Barcelona, 2014, pp. 23–30. http://dx.doi.org/10.1109/FiCloud. 2014.14.

[9] Cisco White Paper, Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are <https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf> (last accessed: January 2017).

[10] C. Perera, P.P. Jayaraman, A. Zaslavsky, P. Christen, D. Georgakopoulos, MOSDEN: an internet of things middleware for resource constrained mobile devices, in: 2014 47th Hawaii International Conference on System Sciences, Waikoloa, HI, 2014, pp. 1053–1062. http://dx.doi.org/10.1109/HICSS.2014.137.

[11] H. Rahman, R. Rahmani, T. Kanter, M. Persson, S. Amundin, Reasoning Service enabling SmartHome Automation at the Edge of Context Networks, New Advances in Information Systems and Technologies, vol. 444, Springer International Publishing, 2016, pp. 777–786, http://dx.doi.org/10.1007/978-3-319-31232-3_73.

[12] Raspberry Pi. <https://www.raspberrypi.org/> (last accessed: January 2017).

[13] A.M. Rahmani, T.N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, P. Liljeberg, Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: a fog computing approach, Future Gener. Comput. Syst. (2017).

[14] Amazon AWS IoT. <https://aws.amazon.com/iot/> (last accessed: January 2017).

[15] K. Ashton, That "Internet of Things" thing, RFiD J. (2009).

[16] H. Rahman, T. Kanter, R. Rahmani, Supporting self-organization with logical-clustering towards autonomic management of Internet-of-things, Int. J. Adv. Comput. Sci. Appl. (IJACSA) 6 (2) (2015), http://dx.doi.org/10.14569/IJACSA.2015.060204.

[17] A.M. Haubenwaller, K. Vandikas, Computations on the edge in the internet of things, in: Procedia Computer Science, The 6th International Conference on Ambient Systems, Networks and Technologies (ANT 2015), vol. 52, 2015, pp. 29–34.

[18] F. Bonomi, R. Milito, J. Zhu, 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 (MCC '12), ACM, New York, NY, USA, 2012, pp. 13–16.

[19] M. Aazam, E.N. Huh, Fog computing and smart gateway based communication for cloud of things, in: 2014 International Conference on Future Internet of Things and Cloud, Barcelona, 2014, pp. 464–470.

[20] S. Sarkar, S. Chatterjee, S. Misra, Assessment of the suitability of fog computing in the context of internet of things, IEEE Trans. Cloud Comput. 99 (2015) 1, http://dx.doi.org/10.1109/TCC.2015.2485206.

[21] M. Kavis, Forget Big Data – Small Data is Driving The Internet of Things. <http://www.forbes.com/sites/mikekavis/2015/02/25/forget-big-data-small-

data-is-driving-the-internet-of-things/#736c8b98661b> (last accessed: January 2017).

[22] XpertRule Software Ltd. <http://xpertrule.com/> (last accessed: January 2017).

[23] Z. Ghahramani, Probabilistic machine learning and artificial intelligence, Nature 7553 (2015) 452–459.

[24] P.P. Jayaraman, C. Perera, D. Georgakopoulos, S. Dustdar, D. Thakker, R. Ranjan, Analytics-as-a-service in a multi-cloud environment through semantically-enabled hierarchical data processing, Softw. Pract. Exp. (2016).

[25] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P.P. Jayaraman, A. Zaslavsky, I.P. Žarko, et al., in: Interoperability and Open-Source Solutions for the Internet of Things, Springer, 2015, pp. 13–25.

[26] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: a programming model for large-scale applications on the internet of things, in: Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC '13), ACM, New York, NY, USA, 2013, pp. 15–20.

[27] P. Barnaghi, F. Ganz, C. Henson, A. Sheth, Computing perception from sensor data, Sensors, IEEE, Taipei, 2012, pp. 1–4, http://dx.doi.org/10.1109/ICSENS.2012.6411505.

[28] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, The MIT Press Cambridge, Massachusetts, London, England. <https://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html> (last accessed: January 2017).

[29] SmartThings: Smart Home, Intelligent Living. <https://www.smartthings.com/> (last accessed: January 2017).

[30] CNET Smart Home – CNET. <http://www.cnet.com/smart-home/> (last accessed: September 2016).

[31] EDYN Smart Gardeing. <https://edyn.com/> (last accessed: January 2017).

[32] A. Barberousse, S. Franceschelli, C. Imbert, Computer simulations as experiments, Synthese 169 (3) (2009) 557–574.

[33] E.C. Parke, Experiments, simulations, and epistemic privilege, Philos. Sci. 81 (4) (2014) 516–536.

Hasibur Rahman earned his Philosophy of Licentiate degree in Computer and Systems Sciences in 2015 from Stockholm University, Sweden where he is currently working towards his final doctoral degree. Prior to this, he received his M.Sc. degree in Computer Engineering from Mid Sweden University in 2013. Immediate after he joined Stockholm University as a Research Assistant where he started his current research in Internet of Things. He has previously conducted research in Wireless Communications -CDMA, Single Frequency Networks, Mobile TV, DVB-T/H, etc. His current research includes autonomic management of IoT, Context Information management, Intelligent SmartCity applications, Context-Aware Reasoning services, etc.

Rahim Rahmani earned a Ph.D. in communications in heterogeneous networks and he is an Associate Professor of Computer Science at the Department of Computer and System Sciences at Stockholm University, where his research focuses on Collaborative ubiquitous services and context-aware mobile communication and service architectures and self-organizing application infrastructures. He is a member of the editorial board of International Journal of Wireless Networking and Communications.