Computation Offloading and Resource Allocation for Low-power IoT Edge Devices

Farzad Samie[1] , Vasileios Tsoutsouras[2] , Lars Bauer[1] , Sotirios Xydis[2] , Dimitrios Soudris[2] , Jörg Henkel[1]
[1] *Chair for Embedded Systems (CES), Karlsruhe Institute of Technology (KIT), Germany*
[2] *Microprocessors and Digital Systems Laboratory, ECE, National Technical University of Athens, Greece*

***Abstract*—With the proliferation of portable and mobile IoT devices and their increasing processing capability, we witness that the edge of network is moving to the IoT gateways and smart devices. To avoid Big Data issues (e.g. high latency of cloud based IoT), the processing of the captured data is starting from the IoT edge node. However, the available processing capabilities and energy resources are still limited and do not allow to fully process the data on-board. It calls for offloading some portions of computation to the gateway or servers. Due to the limited bandwidth of the IoT gateways, choosing the offloading levels of connected devices and allocating bandwidth to them is a challenging problem.**

This paper proposes a technique for managing computation offloading in a local IoT network under bandwidth constraints. The existing bandwidth allocation and computation offloading management techniques underutilize the gateway's resources (e.g. bandwidth) due to the fragmentation issue. This issue stems from the discrete coarse-grained choices (i.e. offloading levels) on the IoT end nodes. Our proposed technique addresses this issue, and utilizes the available resources of the gateway effectively. The experimental results show on average 1 hour (up to 1.5 hour) improvement in battery life of edge devices. The utilization of gateway's bandwidth increased by 40%. [1]

***Keywords*-Internet of Things, IoT, Edge Computing, Resource Allocation, Computation Offloading**

I. Introduction

Recent advances in technologies of sensors, wireless communication and embedded processors have enabled the design of small-size low-power and low cost devices that can be networked or connected to the Internet. These are the key components of the emerging paradigm of Internet-of-things (IoT) [1, 2]. IoT is covering an ever increasing range of applications, such as healthcare monitoring, smart home, smart building, smart city, etc.

One of the challenges in IoT is to process and analyze a huge amount of data from heterogeneous devices. The massive number of IoT devices will lead to a rapid explosion of the scale of collected data. This challenge has two aspects: 1) Big Data [3], and 2) diverse application requirements of IoT [4]. Handling all these collected data with central cloud servers is inefficient, and even sometime is unfeasible, because of:

- the limitation of computing, communication, and storage resources,
- the overall energy and cost,

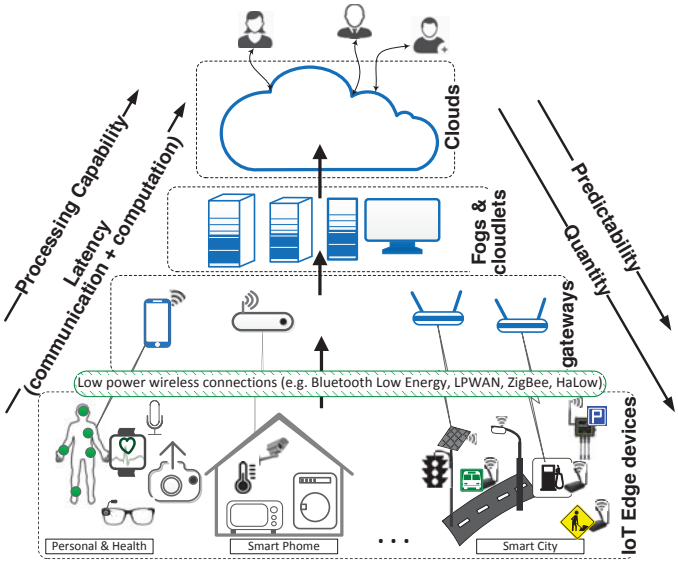


Figure 1: Computation layers in IoT systems and their properties [1]

- and unreliable latency [5, 6].

To deal with these issues, the task of processing the data is pushed to the network edges introducing concepts of Fog computing, cloudlet, and Mobile Edge Computing (MEC) [6, 7, 8, 9]. According to a report by IDC Futurescape, around 40% of IoT-generated data will be processed, stored, and acted upon close to the edge of network [10]. Edge computing (EC) enables analysis of information processing at the source of the data which sometimes is also referred to as in-network or on-board processing.

EC not only reduces the huge workload of central computing servers (e.g. clouds), but also decreases the latency of data processing which includes the network latency for sending/receiving the required data plus the response time for performing the task on the cloud server. Figure 1 shows the hierarchical layers of computation in an IoT system [1]. As we move to the higher levels (i.e. from edge devices to the cloud servers), the processing capability increases. However, the latency would increase due to two factors: 1) network delay and 2) more workload on the servers. Therefore, the predictability for the real-time properties would decrease.

With the proliferation of portable and mobile devices, and increasing processing capabilities of endpoints in IoT,

we envision that the edge of the network is moving to the gateways and smart devices [11, 12]. We are witnessing a trend towards processing the captured data at the IoT edge nodes, even when data acquisition rate is relatively high, such as in healthcare monitoring applications which involve biomedical signals [13, 14], or structural health monitoring applications [15].

Many IoT devices are battery-operated or have limited energy sources due to either their portability requirements, or the lower cost of installation, deployment and maintenance. This limitation imposes the following two constraints on the edge devices: 1) processing capability, 2) communication bandwidth. Due to the first constraint, some of the IoT applications are not able to autonomously process the whole collected data at the edge devices. Hence, a portion of the processing task must be *offloaded* to the more powerful layers (e.g. gateways, Fogs or cloudlets) [14, 13]. However, to offload the computation, the raw data or partially-processed data must be transmitted to the gateway, and this is where the bandwidth constraint comes into play.

Bandwidth Constraint: Limited energy resources on IoT edge nodes mandate low-power wireless technologies. One of the main limitations of low power wireless technologies is their low data rate and throughput. Despite the recent and rapid enhancements in these wireless technologies and protocols, the throughput is still low. Table I summarizes the throughput of the most popular low-power IoT wireless technologies including Bluetooth low energy (BLE), ANT+, ZigBee, low power WiFi (also known as HaLow), Low Power Wide Area (LPWAN) technologies which include LoRa, SigFox, etc. [16]. It is worth mentioning that the throughput may decrease further in case of interference with other surrounding wireless radios [1].

Table I: Throughput of IoT wireless technologies [17, 18]

Wireless	BLE	ANT+	ZigBee	HaLow	LoRa	SigFox
Throughput [Kbps]	270	20	120	150	50	<10

IoT envisions a model in which IoT edge nodes are connected to the Internet through gateways, as illustrated in Figure 1. A gateway (i) enables seamless integration of low-power wireless networks of IoT edge nodes with other networks (e.g. cellular network, LAN, etc. [19]) and (ii) provides local data processing service at the edge of the network [20] in the EC paradigm. Although the gateway might exploit a high-bandwidth connection to the Internet (i.e. cellular or WiFi), its interface with IoT edge nodes is still a low-power wireless connection such as LoRa, BLE or ZigBee, which have a low bandwidth (see Figure 1 and Table I).

The limited bandwidth of the gateway is usually shared among multiple IoT edge nodes, which introduces the bandwidth allocation as a challenge in IoT, especially when video, sound, or some bio-medical signals are involved. The bandwidth demand of IoT edge nodes can be reduced by novel compression techniques [14, 21] or service quality adaptation [22]. However, the most effective approach would be processing the data partially on the IoT devices and offload the rest of computation to the gateway or a more resourceful device (e.g. Fogs, cloudlets, could servers).

Although there are several approaches for computation offloading and bandwidth allocation on the gateway in the domain of IoT and MEC [22, 23], they have not addressed the *fragmentation issue*: Since the offloading levels on the IoT edge nodes are discrete and coarse-grained, the resources on the gateway (e.g. bandwidth) will be underutilized. The reason is that the remaining amount of resources may not be enough for the IoT devices to change their offloading level, which would result in unused resources on the gateway. We refer to this phenomena as fragmentation issue.

In this paper we study the problem of computation offloading (as an essential part of edge computing) from IoT edge nodes under communication bandwidth constraints. We address the fragmentation issue by presenting a technique that allows IoT edge nodes to utilize the available resources on the gateway to their full potential, even if the offloading levels are coarse-grained. Depending on the desired optimization goal, this technique makes the system more efficient.

Paper structure: in Section II, we briefly review related work and background. Section III presents the problem formulation. After that, we provide a detailed presentation of our novel solution in Section IV. Experimental results and evaluations are presented and discussed in Section V , while we conclude the paper in Section VI.

II. Related Work

In [23], a decentralized game theoretic approach for computation offloading is presented. The wireless channel is considered as the scarce and shared resource. In this approach the decision is between fully offloading the computation or fully processing on-board. The proposed approach cannot address the fragmentation issue.

Samie et al. [22] have proposed a service quality and computation offloading technique for the resource-constraint IoT edge nodes and gateways. However, in their proposed solution, the gateway might be underutilized due to fragmentation.

A joint optimization of bandwidth and computational resources for computation offloading in a dense deployment scenario is presented in [24], which considers the presence of radio interference. The proposed solution is however unaware of fragmentation issues.

Several other research works have been conducted to address the problems of edge computing [25], but none of them has addressed the fragmentation issue in shared resources.

III. System Model

We consider a local network of N IoT edge nodes $\mathcal{I} = \{I_1, \cdots, I_N\}$ connected to a gateway. IoT device I_d is described by several parameters including:

$$I_d = \Big(Q_d, R_d\Big), \qquad d = 1, \cdots, N \tag{1}$$

- Q_d denotes the number of different *computation offloading levels* that the IoT device offers. Some of the applications can be partitioned into several pipeline stages, each of which processes the data received from the previous stage and feeds the results to the next stage (if any). As an example, consider an IoT-based heart monitoring device. In its first stage, it can perform some pre-processing on the data, e.g. applying a filter on it. In the second stage, it can apply a transform operation, e.g. Digital Wavelet Transform (DWT). And in the last stage, it extracts the crucial complexes and features [22].
  Even if the application does not support such a pipeline structure, it has at least two computation offloading levels: (i) Level 1 submits the raw data without on-board processing, (ii) Level 2 indicates 'no computation offloading', thus fully processes the data and only transmits the results.
- R_d denotes the set of possible *transmission data rates* of device d for the offloading levels. They depend on the input data rate and data resolution of fixed value, as well as the computation offloading strategy of the device.
  The transmission data rate r_{d_i}, which is associated with the offloading level i, corresponds to the share of captured data that is offloaded plus the (intermediate) results from the partial on-board processed data.

$$R_d = \big\{r_{d_i} \mid i \in [1, Q_d]\big\} \tag{2}$$

The gateway connects these devices to the Internet. It receives data from IoT edge devices, then either processes it and transmits the final results to the Internet or directly relays the data to another processing unit (e.g. cloudlets, Fogs, cloud servers [20, 1, 6]). R_G is the total available bandwidth of the gateway to receive data from IoT devices. The effect of environment and surrounding devices (e.g. external interference) on the transmission can be modeled in R_G, R_d and other parameters (e.g. energy consumption). However, this is beyond the scope of this work.

Generally, the IoT local network (i.e. gateway and edge nodes) has an objective and optimization goal depending on the application. It introduces an optimization problem such as minimizing the overall energy consumption, maximizing the overall service quality [22], maximizing the battery life of edge nodes, etc. subject to resource constraints such as bandwidth, processing power, etc. [22].

These problems can be formulated as Eq. (3) to (4). The offloading level i for each IoT device d must be determined at run-time, such that the constraints are fulfilled (Eq. (4)) and the desired objective is maximized/minimized (Eq. (3)).

$$\text{Optimization goal:} \quad \begin{matrix}\text{maximize (objective)}\\ \text{or}\\ \text{minimize (objective)}\end{matrix} \tag{3}$$

$$\text{Constraints:} \quad \begin{matrix}\sum_{\forall d} r_{d_i} \le R_G\\ \text{other contraints (if any)}\end{matrix} \tag{4}$$

These problems belong to the class of *generalized assignment problems*. Particularly, they can be formulated as a multiple choice knapsack problem (MCKP), where the gateway corresponds to the 'knapsack', the bandwidth or other constraints correspond to the 'weight' or 'volume', and the offloading levels of each IoT edge node correspond to the class of items from which one and only one item must be picked. Even though MCKP is known to be NP-hard, pseudo-polynomial solutions exist. One of the shortcomings of the solutions to the current setup is the underutilized resources of the gateway due to the fragmentation issue (see Section I).

For the proof-of-concept, we consider a problem in this work with the optimization goal to maximize the battery life among devices (Eq. (5)), and its constraint is the gateway's bandwidth (Eq. (6)). *It should be emphasized that our proposed solution is not restricted to this problem formulation, and it is applicable to any problem that can be modeled as Eq.* (3) *and* (4). For our considered problem, we need to use these parameters on the IoT edge nodes:

- E_d is the remaining energy in the battery of device d.
- p_{d_i} is the overall power consumption of device d operating at offloading level i. It includes the power consumption for on-board processing as well as data transmission (at the rate r_{d_i}).
- b_{d_i} denotes the estimated battery lifetime of IoT device d at offloading level i. This parameter depends on (i) its remaining energy and (ii) its total power consumption rate: $b_{d_i} = \frac{E_d}{p_{d_i}}$.

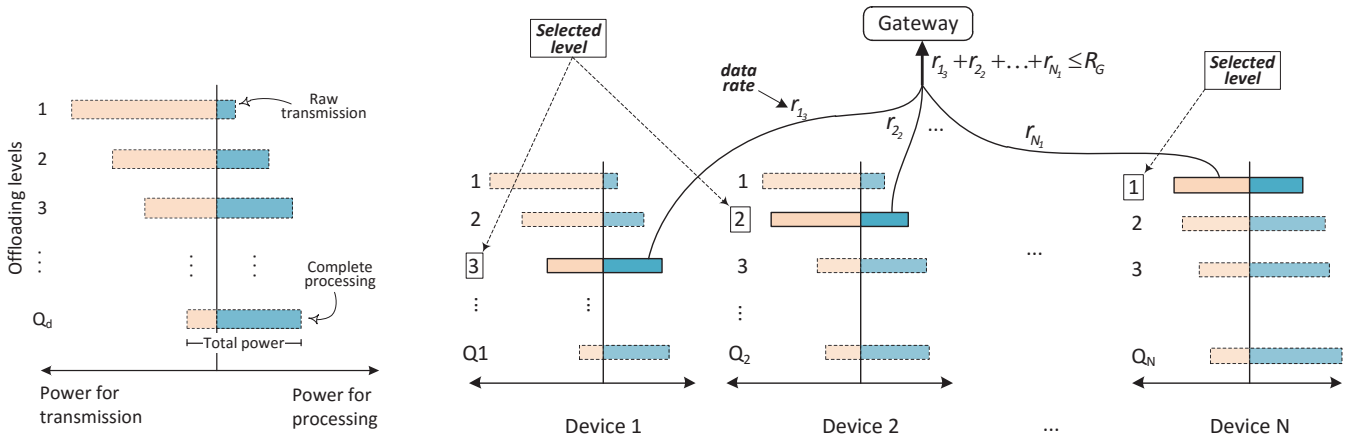
$$\text{Optimization goal:} \quad \text{maximize}(\min_{\forall d} b_{d_i}) \tag{5}$$

$$\text{Bandwidth constraint:} \quad \sum_{\forall d} r_{d_i} \le R_G \tag{6}$$

By selecting the computation offloading level on each IoT edge node, not only its bandwidth demand but also its power consumption is changed and consequently, its expected battery life will change. Figure 2 shows the problem model that we consider in this work. Figure 2a shows different offloading levels of one IoT edge node (level 1 being to offload all data, level Q_d being to process all data on-board), while Figure 2b illustrates the bandwidth allocation and offloading management with respect to the gateway's constraint.

IV. Our Proposed Solution

In order to address the fragmentation problem and underutilization of gateway's resources, we suggest a solution to extend the on-board processing model. It's worth to note that this solution does not need any modification or extension in



(a) Offloading levels of an IoT device and their power consumption

(b) Problem model: choosing offloading level for each device, meeting the bandwidth constraint of the gateway

Figure 2: The offloading levels of a device, and the problem model of the local network

the pipeline structure of the software application. As mentioned, the fragmentation issue stems from the discreteness of choices. This issue is more severe when the number of offloading levels is small and the difference between data transmission rate of offloading levels is high (e.g. when the IoT application only supports two offloading levels: raw data transmission and fully process the data).

A. Offloading Levels: discrete to continuous transform

The key idea is to regularly switch between different offloading levels at IoT devices such that it appears to the gateway as if the IoT device would operate at an intermediate (practically not existing) offloading level.

Property 1: Let us consider two offloading levels i and j with data transmission rates of r_{d_i} and r_{d_j} where $r_{d_i} \leq r_{d_j}$. For any r'_d, $r_{d_i} \leq r'_d \leq r_{d_j}$, the device d can operate in a way such that its data transmission rate equals r'_d from gateway's and high-level (e.g. cloud) perspective.

Proof: Consider a time frame of T. If the device operates at the offloading level j for t_1 time, then changes the level and operates at the offloading level i for $(T-t_1)$ time, then the average transmission rate, and power consumption are respectively:

$$r'_d = \frac{t_1 \times r_{d_j} + (T-t_1) \times r_{d_i}}{T} \quad p'_d = \frac{t_1 \times p_{d_j} + (T-t_1) \times p_{d_i}}{T} \tag{7}$$

By choosing the proportion of t_1 and $(T-t_1)$, the IoT device can imitate any intermediate data rate r'_d, or intermediate offloading level with (r'_d, p'_d). ■

The detailed scheme for the IoT device is as follows: It starts operating at j-th level (which has higher transmission rate compared to i-th level). It keeps working at this configuration for t_1 time. However, it transmits its data to the gateway at the rate of r'_d ($r'_d \leq r_{d_j}$). It buffers the rest of produced data (i.e. $r_{d_j} - r'_d$) on the memory. Then after t_1, it switches to the i-th level whose data-generation rate is r_{d_i} ($r_{d_i} \leq r'_d$). It still transmits the data to the gateway at the rate of r', which consists of previously buffered data and newly generated data. Therefore, while the device is only operating on i-th and j-th offloading levels, the gateway sees the device operating at an intermediate configuration mode with (r'_d, p'_d).

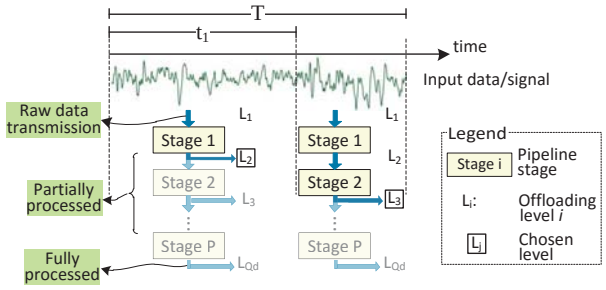


Figure 3: Switching between offloading levels 2 and 3

Example 1: Figure 3 illustrates an example for our proposed mechanism. Let us assume the IoT device is switching between these two offloading levels: (i) Level 2 indicates data transmission at the rate of 0.8 Kbps, (ii) Level 3 indicates data transmission at the rate of 0.3 Kbps. By spending 0.6 s at level 1 and 0.4 s at level 2, the device can deliver data rate at $(0.8\ \text{[Kbps]} \times 0.6 + 0.3\ \text{[Kbps]} \times 0.4)/(0.6+0.4) = 0.6$ [Kbps].

Given r'_d as the desired data transmission rate of device d, the device calculates the proportion of t_1 and $(T-t_1)$ as follows:

$$\frac{t_1}{T-t_1} = \frac{r'_d - r_{d_i}}{r_{d_j} - r'_d} \tag{8}$$

where r_{d_i} and r_{d_j} are the closest data transmission rates among offloading levels such that $r_{d_i} < r'_d < r_{d_j}$.

Buffer Size: The size of required memory to buffer the data depends on 1) the time spent in higher offloading level (i.e. t_1), and 2) its surpassed data. It can be calculated as follows by substituting the value of r'_d from Eq. (7):

$$F = t_1 \times (r_{d_j} - r'_d) = \frac{t_1.(T-t_1)}{T} \times (r_{d_j} - r_{d_i}) \tag{9}$$

Given a fixed time interval T, the maximum size of buffer

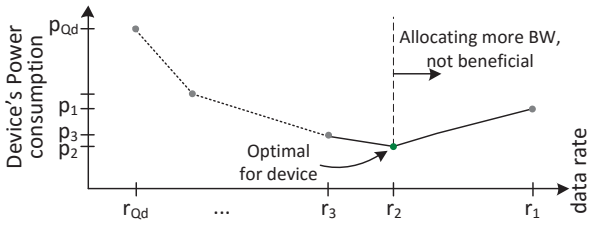


Figure 4: Piecewise-linear function to select the continuous allocated bandwidth of an IoT edge node

happens when the desired data transmission rate is $r'_d = \frac{1}{2}(r_{d_i} + r_{d_j})$ which corresponds to $t_1 = \frac{T}{2}$. In this case, the buffer size equals $F = \frac{T}{4} \times (r_{d_j} - r_{d_i})$.

B. Bandwidth Allocation & Offloading Management

Although the required information about offloading options, power consumption and remaining energy is distributed across the IoT devices, the final decision on bandwidth allocation is made on the gateway. Each device sends the following parameter values to the gateway and updates them once they are changed: E_d and $\{(r_{d_i}, p_{d_i})\}_{1 \le i \le Q_d}$. In most application scenarios (if not all of them), the r_{d_i} parameter is a fixed value as it depends on the application's pipeline structure, which is also fixed. However p_{d_i} may change over time due to several reasons: the effect of interference on the data transmission cost is one of them. Finally, E_d changes due to energy consumption or battery recharge.

Once the gateway receives these parameters from the devices, it forms a piecewise-linear continuous function using (r, p) pairs for each device, as shown in Figure 4. For each device, d, the gateway selects a transmission rate r'_d in the range $[r_{Q_d}\ r_1]$. It corresponds to a power consumption p'_d, which is found using the piecewise-linear function (see Figure 4). The selection of r'_d is toward maximizing the battery lifetime. A simple implementation for finding the optimal allocation is inspired by the *gradient projection method*, used in optimization algorithms. It starts from an initial solution and converges to the optimum solution after a limited number of iterations.

The iterative bandwidth allocation algorithm, executed on the gateway, is presented in Algorithm 1. In the initial solution, the gateway gives each device its minimum bandwidth demand (i.e. $\min(r_{d_i}), 1 \le i \le Q_d$), as shown in Line 1. Then it calculates the battery life of each device under this configuration (i.e. $b_{d_i} = E_i/p_{d_i}$). The gateway also calculates its remaining bandwidth, $R' = R_G - \sum_{\forall d}(r_{d_i})$ (Line 3). The remaining bandwidth must be allocated to the devices, step by step, while prioritizing the devices with lower battery life.

At each iteration, the gateway finds the devices with the lowest and the second lowest battery life (let us denote them respectively by d^*, d^+, as shown in Lines 5-6). Then gateway checks if the battery life of d^* can be extended by allocating more bandwidth to it (Line 7). If not, gateway excludes this device at next iterations (Line 15). Otherwise, it calculates the power consumption for d^* which makes its battery life equal to battery life of d^+ (i.e. $p^* = E^*/b^+$, see Line 9). It allocates the bandwidth which would result to closest power consumption to p^* (Line 10). In cases that b^* and b^+ are close, the gateway increases the allocated bandwidth of d^* by Δ, which represents the minimum amount of bandwidth that can be given to or taken away from an IoT device (Lines 13-14). It iteratively keeps repeating this procedure until the remaining bandwidth, R' is totally consumed.

Algorithm 1: Iterative BW Allocation on the Gateway

```
   Procedure Initialize ( )
1      for all device:   allocate minimum bandwidth
2      for all device:   calculate b_{d_i}
3      Calculate R' = R_G − ∑_{∀d}(allocted BW)  // the remaining BW
   end
   Procedure Iterate ( )
4      while R' > 0 do    // gateway still has BW to allocate
5          d* ← device with minimum battery
6          d+ ← device with second minimum battery
7          if d* benefits from more BW then
8              if batt. life of d*  != batt. life of d+ then
9                  p* = E*/b+
10                 r'_d ← corresponding BW to p*
11                 R' ← R' − r'_d
12
               else
13                 Increase allocated BW of d* by Δ
14                 R' ← R' − Δ
               end
           else
15             exclude it in the next iteration
           end
       end
   end
```

V. Evaluation and Results

To demonstrate the effectiveness of our proposed technique, we conducted some experiments based on a personal health monitoring case-study. It includes real world measurements on an actual IoT node and trace driven network simulation to investigate the behavior of the whole system.

Setup: We consider the Intel Quark SoC as the core of our IoT devices, which has been proposed and used for wearable IoT devices. The gateway is assumed to exploit an ARM Cortex-M3 processor. We assume BLE as the wireless connection between gateway and IoT devices. Power consumption and throughput values of data transmission are according to [17]. We choose a rechargeable Lithium-Ion coin cell battery for IoT devices with nominal capacity of 420 mAh. Moreover, we consider a realistic discharge model for the battery as [26] to evaluate the remaining energy at runtime (i.e. E_d).

Application Scenario: We consider a health monitoring application for ECG analysis. The captured signal from heart activity is processed to detect abnormality or arrhythmia. The analysis flow contains four main stages: (i) Filtering, (ii) Segmentation & heart beat detection, (iii) Feature extraction,

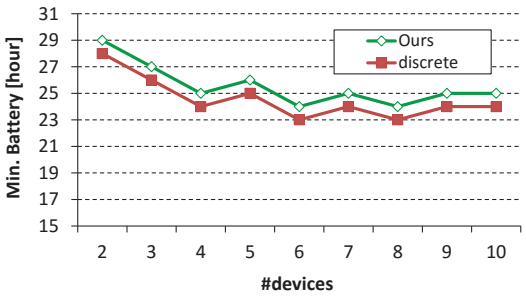


Figure 5: Minimum battery life of IoT edge nodes with our technique compared to the discrete configurations

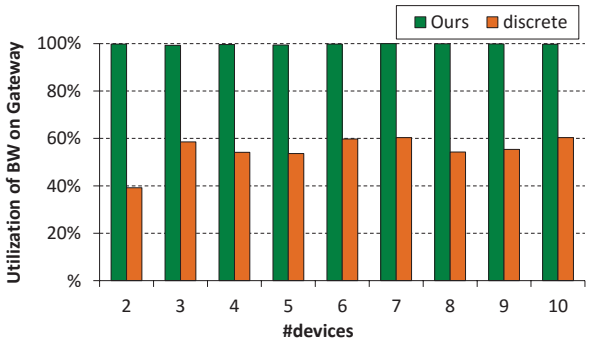


Figure 6: Utilization of gateway's bandwidth in our technique compared to the discrete configurations

and (iv) Diagnosis & classification. We use real-world ECG data recordings from MIT-BIH Arrhythmia Database [27] which also provides annotations for the signals.

Results We implement our technique in a local IoT network with varying number of edge nodes (i.e. from 2 to 10). We investigate the minimum battery life as well as the utilization of bandwidth on gateway. Figure 5 shows the minimum battery life of our technique compared to a technique with discrete configuration. Our technique achieves on average 1 hour (up to 1.5 hour) improvement in minimum battery lifetime which is attributed to complete use of all available bandwidth of the Gateway in contrast to the fragmented utilization of the discrete resource allocation scheme (Figure 6).

VI. Conclusion

With the increasing processing capabilities of IoT edge nodes, and proliferation of portable devices, we have witnessed that computation is pushed to the smart devices and gateways. This paper studies the computation offloading and bandwidth allocation in local networks of IoT edge nodes. We present a novel technique that allows the devices to utilized the limited resources (e.g. gateway's bandwidth) fully and efficiently. Experimental results for a health monitoring case study show up to 40% improvement in utilization of gateway's bandwidth. We achieve up to 1.5 hour improvement in battery life of IoT edge devices.

References

[1] F. Samie and L. Bauer and J. Henkel. IoT Technologies for Embedded Computing: A Survey. In *CODES+ISSS*, 2016.

[2] L. Atzori and A. Iera and G. Morabito. The Internet of Things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] I. Lee and K. Lee. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, 2015.

[4] B. Zhang, N. Mor, J. Kolb, et al. The cloud is not enough: saving IoT from the cloud. In *USENIX Conference on Hot Topics in Cloud Computing*, pp. 21–21, 2015.

[5] R. Want and B. N. Schilit and S. Jenson. Enabling the internet of things. *IEEE Computer*, 48(1):28–35, 2015.

[6] O. Salman and I. Elhajj and A. Kayssi et al. Edge computing enabling the Internet of Things. In *WF-IoT*, pp. 603–608, 2015.

[7] A. Ahmed and E. Ahmed. A survey on mobile edge computing. In *Int'l Conference on Intelligent Systems and Control (ISCO)*, 2016.

[8] N. Fernando and S. W. Loke and W. Rahayu. Mobile cloud computing: A survey. *Future Generation Computer Systems*, 29(1):84–106, 2013.

[9] W. Shi, J. Cao, and Q. Zhang, et al. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 2016.

[10] C. Raphael. Why edge computing is crucial for the IoT. Online: http://www.rtinsights.com/why-edge-computing-and-analytics-is-crucial-for-the-iot/, 2016.

[11] T. Zachariah, N. Klugman, B. Campbell, et al. The Internet of Things has a gateway problem. In *Mobile Computing Systems and Applications (HotMobile)*, pp. 27–32, 2015.

[12] L. Catarinucci, D. De Donno, L. Mainetti, et al. An IoT-aware architecture for smart healthcare systems. *IEEE Internet of Things Journal*, 2012.

[13] R. Braojos, I. Beretta, and J. Constantin, et al. A wireless body sensor network for activity monitoring with low transmission overhead. In *IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 265–272, 2014.

[14] D. Bortolotti, M. Mangia, A. Bartolini, et al. Energy-aware bio-signal compressed sensing reconstruction on the WBSN-gateway. *IEEE Transactions on Emerging Topics in Computing*, 2016.

[15] *Sensors to Support the IoT for Infrastructure Monitoring: Technology and Applications for Smart Transport/Smart Buildings*, 2015.

[16] M. S. Mahmoud and A. A. Mohamad et al. A study of efficient power consumption wireless communication techniques/modules for internet of things (IoT) applications. *Advances in Internet of Things*, 2016.

[17] P. Smith. Comparing low-power wireless technologies. Tech Zone, Digikey Online Magazine, Digi-Key Corporation, 2011.

[18] Multitech. Introduction to LoRa. Online: http://www.multitech.net/developer/software/lora/introduction-to-lora/.

[19] Q. Zhu, R. Wang, and Q. Chen, et al. IoT gateway: Bridging wireless sensor networks into internet of things. In *Int'l Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 347–352, 2010.

[20] T. N. Gia, M. J. A.-M. Rahmani, and T. Westerlund, et al. Fog computing in healthcare Internet-of-Things: A case study on ECG feature extraction. In *Int'l Conf. on Computer and Information Technology (CIT)*, pp. 356–363, 2015.

[21] F. Samie and L. Bauer and J. Henkel. An approximate compressor for wearable biomedical healthcare monitoring systems. In *CODES+ISSS*, pp. 133–142, 2015.

[22] F. Samie, V. Tsoutsouras, S. Xydis, et al. Distributed QoS Management for Internet of Things under Resource Constraints. In *CODES+ISSS*, 2016.

[23] X. Chen. Decentralized computation offloading game for mobile cloud computing. *IEEE Transactions on Parallel and Distributed Systems*, 26(4):974–983, 2015.

[24] S. Sardellitti and G. Scutari and S. Barbarossa. Joint optimization of radio and computational resources for multicell mobile-edge computing. *IEEE Transactions on Signal and Information Processing over Networks*, 1(2):89–103, 2015.

[25] Y. Mao and J. Zhang. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal of Solid-State Circuits*, 51(3):712–723, 2016.

[26] C. Zhu and X. Li and L. Song et al. Development of a theoretically based thermal model for lithium ion battery pack. *Journal of Power Sources*, 223:155–164, 2013.

[27] A. L. Goldberger, L. A. Amaral, L. Glass, et al. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23), 2000.