# Optimal Allocation of Computation and Communication in an IoT Network

ABHIMANYU CHOPRA, George Mason University, USA
HAKAN AYDIN, George Mason University, USA
SETAREH RAFATIRAD, George Mason University, USA
HOUMAN HOMAYOUN, George Mason University, USA

Abstract— Internet of things (IoT) is being developed for a wide range of applications from home automation and personal fitness, to smart cities. With the extensive growth in adaptation of IoT devices, comes the uncoordinated and substandard designs aimed at promptly making products available to the end consumer. This substandard approach restricts the growth of IoT in the near future and necessitates studies to understand requirements for an efficient design. A particular area where IoT applications have grown significantly is the surveillance and monitoring. Applications of IoT in this domain are relying on distributed sensors, each equipped with a battery, capable of collecting images, processing images and communicating the raw or processed data to the nearest node until it reaches the base station for decision making. In such an IoT network where processing can be distributed over the network, the important research question is how much of data each node should process and how much it should communicate for a given objective. This work answers this question and provides a deeper understanding of energy and delay trade-offs in an IoT network with three different target metrics.

CCS Concepts: • **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

Additional Key Words and Phrases: Wireless sensor networks, media access control, multi-channel, radio interference, time synchronization

**ACM Reference Format:**
Abhimanyu Chopra, Hakan Aydin, Setareh Rafatirad, and Houman Homayoun. 2017. Optimal Allocation of Computation and Communication in an IoT Network. *ACM Trans. Des. Autom. Electron. Syst.* 0, 0, Article 0 ( 2017), 23 pages. https://doi.org/0000001.0000001

## 1 INTRODUCTION

Internet of Things(IoT) is a system of devices, which has the ability to sense, compute and take action on the data such as communicating the data or moving an actuator, just to name a few. The data can range from sound, image to temperature, based on geographic location and system application. Consumer-end products in IoT domain are quickly moving from traditional to network-enabled devices like Apple Watch and Fitbit. These devices have the ability to sense the user's heart rate, track number of steps and generate various reminders including those corresponding to water consumption, walking, competing with friends, among several others. Moreover, they consolidate data between different devices and provide the user with an overall view. Devices such

Authors' addresses: Abhimanyu Chopra, George Mason University, Fairfax, VA, USA, achopra6@gmu.edu; Hakan Aydin, George Mason University, Fairfax, VA, USA, aydin@cs.gmu.edu; Setareh Rafatirad, George Mason University, Fairfax, VA, USA, srafatir@gmu.edu; Houman Homayoun, George Mason University, Fairfax, VA, USA, hhomayou@gmu.edu.

1084-4309/2017/0-ART0 $15.00
https://doi.org/0000001.0000001

as Nest have the ability to sense and control the temperature, lighting, air quality in the consumer's house and alert the user in case of abnormal conditions, or take steps to modify conditions based on pre-defined configurations. IoT has found its applications in numerous fields. One example is the deployment of IoT enabled-devices for object detection in large areas. For instance in forest, IoT devices are utilized to track animals, and near the international border they are used to check alien infiltration. In a smart city's ecosystem, IoT devices help monitor traffic or even pinpoint crimes. In all of these examples of IoT enabled ecosystems, IoT devices are interconnected and should operate in an energy-efficient manner in a resource constrained environment to enhance network lifetime. The proliferation of IoT enabled devices in our homes, work space and every day life results in large amounts of data, which needs to be captured, computed, communicated and appropriately reacted upon in an energy-efficient and timely manner [Sayadi et al. 2018a,b]. There are several parameters influencing the trade-off between energy-efficiency and performance [Makrani et al. 2017; Sayadi et al. 2018c] in an IoT network, including the technology of computing platform of IoT devices (e.g. Intel Edison, Raspberry Pi, Arduino), the wireless technology interconnecting these devices (BLE, ZigBee, WiFi, Cellular), and the application they are running for data processing (Image processing, DSP).

The battery capacity of an IoT device is usually very small. In fact, it is sometimes harvested using solar panels. Unlike other embedded systems [Makrani et al. 2014; Sayadi et al. 2014], IoT devices can not afford to recharge their batteries frequently and are required to work for days or months on a single charge. This requires energy-efficient solutions to enhance their lifetime. An IoT network is also constrained by timing requirements when deployed for real-time applications, enunciating energy-efficient solutions while meeting performance requirements. As a result, energy-efficiency and performance are required to be simultaneously optimized in such IoT networks.

New platforms [Makrani and Homayoun 2017b; Makrani et al. 2018], algorithms and network protocols have been studied in the IoT space to minimize energy consumption. Lazarescu et. al. [Lazarescu 2013] have presented a functional implementation of a complete wireless sensor network platform that can be used for a range of long-term environmental monitoring IoT applications. Serra et al. [Serra et al. 2014] introduced an energy-scheduling method that minimizes energy consumption cost based on the current energy price and user comfort constraints. Edalat et al. [Edalat et al. 2009] and, Yu and Prasanna [Yu and Prasanna 2005] have proposed energy-aware allocation of tasks to increase network lifetime but allocated the task as a single unit rather than distributing the computation over the network. Dynamic voltage scaling [Krishnan 2010] [Pouwelse et al. 2001] has been proposed [Jayakumar et al. 2014] [Zhu et al. 2015] to be used in IoT devices where the CPU works at lower frequencies to decrease the power consumption by regulating the voltage supplied to the system or shutting down inactive modules. Rodoplu et al. [Rodoplu and Meng 1999] build a minimum energy communication network using a position based algorithm aiming to construct a topology consisting of lowest energy paths to transmit from any wireless sensor in a network to the sink node using the concept of relay transmission.

Many works [Baker and Ephremides 1981] [Ephremides et al. 1987] [Gallager et al. 1979] [Singh et al. 1998] [Meng and Rodoplu 1998] [Ettus 1998] [Shepard 1995] have taken the optimal routing approach to minimize the communication energy cost. The approach is to minimize the energy consumed to reach the destination or sink node, which would minimize the energy consumed per packet or task. If all traffic is routed through the same path, batteries of such nodes would run out quickly while that of other nodes would remain intact.

Optimal task/resource allocation has been extensively studied for wireless sensor networks. In [Pilloni and Atzori 2011] node's residual energy is taken into account and a centralized task allocation algorithm is proposed to improve the network lifetime. The task execution time or delay is taken into account in [Zhu et al. 2007] and the same problem is analyzed. In [Pilloni et al. 2012] the authors provide a framework that distributes tasks to different nodes in a network by means of a distributed optimization algorithm based on gossip communication to increase network lifetime.

In this work, we study a common case IoT network where nodes are interconnected, collecting data, and in a cooperative manner processing and communicating the data through the network to a base station for decision making. Such common IoT network can be found in smart cities for surveillance or in environmental applications for monitoring the airborne quality, water quality, radiation, and many other environment indicators. We study IoT networks with various optimization objectives and constraint metrics to address energy-efficiency and performance requirements [Makrani and Homayoun 2017a] by dividing and efficiently allocating computation of workload data over the entire IoT network [Sayadi et al. 2017]. The important research question that is raised in such IoT network is, how much data each node should communicate and how much data should be locally computed for a given optimization goal. The allocation is optimized by analyzing the immediate platform-specific parameters such as computation energy, transmit energy, receive energy, available energy in each node and distance from the base station as well as reduction (or compression) of data based on the running application as it travels through the network and is processed by various nodes. In this paper, this problem is formulated into a LP (Linear Programming) and is solved using Scip Optimization Suite [Gamrath et al. 2016] and Symphony [Ralphs and Güzelsoy 2005] from Coin-OR. This approach not only minimizes the overall energy consumption of the network, but also provides a balanced performance-energy solution with increased network lifetime. The results provide network designer with information to create an efficient IoT network based on application-specific requirements.

This paper in brief makes the following new contributions:

-We study whether in an IoT network a task computation should be entirely done on the source node, or distributed across several nodes as it travels towards the base station.

-We study the computation vs communication problem in IoT networks with three different objectives of minimizing overall energy, maximizing available energy and meeting delay deadline. We define energy and delay weight as two terms to solve the optimization problem.

-We study how different IoT sensor parameters (computing and communication) affect the computation allocation on each node.

In brief, the paper makes the following observations:

-For minimizing energy dissipation objective, all data should be processed on the first node in a homogeneous network or the node/s with the least communication and computation cost in a heterogeneous network.

-If a node has least energy battery level, the data should be computed before it reaches that node. This will cause the computation cost at that node to become zero, and also minimizes the communication overhead.

-In presence of multiple instances of the same platform in the network, the earlier instance closer to the source is always efficient in terms of communication and computation cost.

-We provide an allocation strategy for delay deadline objective, which can be followed for optimal allocation without the overhead of the solver.

## 2 SYSTEM MODEL & PROBLEM DEFINITION

We consider a system with a set of n sensor nodes, S = $\{S_i : i = 0, 1, 2, ..., n\}$, where $S_0$ is the source or generator of data and $S_n$ is the sink and the only node connected to a base station. Each node is powered with a small battery is only capable of communicating with its immediate neighbors, i.e. $S_i$ can only communicate with $S_{i+1}$ and $S_{i-1}$. The task of the sensor nodes is to compute and communicate data generated at $S_0$ to the base station. The task incurs an energy and delay cost on each sensor node.

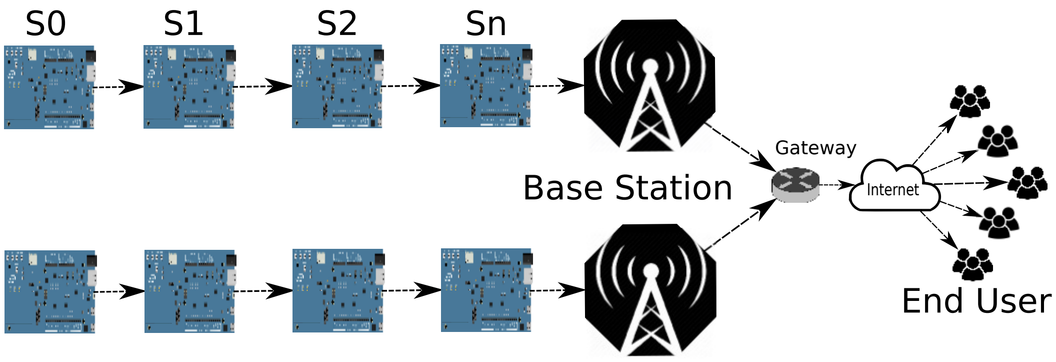


Fig. 1. IoT Network

The energy cost at each node $N_i$, has two components as defined in Table 1, namely the computation cost and the communication cost. At $S_0$, the receive cost is zero as it does not receive data from any other node.

Table 1. Components of energy cost at a node

| **Energy cost component** | **Description** |
|---|---|
| Computation | Computation cost per bit ($P_i$) * Bits computed ($x_i$) |
| Communication | Transmission cost + Receive cost |
| Transmission | Transmission cost per bit ($T_i$) * Bits transmitted |
| Receive | Receive cost per bit ($R_i$) * Bits received |

Table 2. Components of delay at a node

| **Delay component** | **Description** |
|---|---|
| Computation | Computation delay per bit ($D_{Pi}$) * Bits computed ($x_i$) |
| Transmission | Bits transmitted*[1/Transmission rate ($D_{Ti}$)] |

The delay cost $D_i$ for a node is defined by it's two components, computation delay and transmission delay in Table 2. When a node transmits data, the next node receives the data simultaneously. Thus, we do not take into account any data receive delay. The platform specific parameters defined for our network are listed in Table 3.

When a node computes part of the total data size $\alpha$, the size of the processed data reduces based on the compression factor. $C_i$ is defined as 1 - compression factor. For instance, if 100 bits are computed on a node with $C_i$ of 0.1, the size of the computed data is only 10 bits. The node then transmits all the computed data along with uncomputed data to its immediate neighbor towards the sink node. If uncomputed data reaches the sink node, all of it must be computed before being

Table 3. Platform parameters & their description

| Parameter | Description | Unit |
|---|---|---|
| $N_i$ | Energy cost incurred at node i | J |
| $D_i$ | Delay cost incurred at node i | s |
| $x_i$ | Data allocated to be computed at node i | bits |
| $P_i$ | Computation cost per bit at node i | J/bit |
| $T_i$ | Transmission cost per bit at node i | J/bit |
| $R_i$ | Receive cost per bit at node i | J/bit |
| $C_i$ | (1 - Compression factor) at node i | - |
| $E_i$ | Energy level at node i | J |
| $D_{Pi}$ | Delay due to computation per bit at node i | s/bit |
| $D_{Ti}$ | Transmission rate at node i | bit/s |
| $\alpha$ | Data size | bit |
| $\delta$ | Delay deadline | s |

Table 4. Problem formulation for minimizing overall energy consumption

| MOEC problem constraints | | | |
|---|---|---|---|
| Minimize | $\sum_{i=0}^{n}$ | $N_i$ | (a) |
| Subject to | $\sum_{i=0}^{n}$ | $x_i = \alpha$ | (b) |

communicated to the base station. Such an IoT network is commonly used for surveillance, traffic control, and environmental monitoring, where sensors are collecting and cooperatively processing and transmitting the data to the base for decision making.

We study this type of common IoT network to find out the optimal assignment of computation vs communication work with various objectives such as meeting a deadline, or energy-efficiency.

## 3 MODELING & FORMULATION

For optimization purposes, we define our model for various metrics and formulate the problem into its objective and constraints.

### 3.1 Minimize Overall Energy Consumption (MOEC)

Minimize Overall Energy (MOEC) objective is to minimize the overall energy consumption over the entire network, thus the sum of energy consumption throughout all nodes in the network should be the least.

Table 4 shows the parameters employed for the linear programming formulation. Expression (a) is the objective of the problem. $\sum_{i=0}^{n} N_i$ is the sum of the energy consumed on each node in the system, which takes into account the energy used to compute and communicate data to the next node. It is defined as:

$$N_i = \overbrace{P_i x_i}^{Computation\ Energy\ Cost} + T_i\overbrace{(\alpha - \sum_{j=0}^{i} x_j + \sum_{j=0}^{i} C_j x_j)}^{Transmit\ Energy\ Cost} + \overbrace{R_i(\alpha - \sum_{j=0}^{i-1} x_j + \sum_{j=0}^{i-1} C_j x_j)}^{Receive\ Energy\ Cost} \tag{1}$$

When simplified in terms of $x_i$, $\sum_{i=0}^{n} \quad N_i$ can be represented as,

$$\sum_{i=0}^{n} A_i x_i + \beta \tag{2}$$

where, $A_i$ is the weight/coefficient of $x_i$ towards the total energy consumption and,

$$A_i = P_i + \{\sum_{j=i}^{n-1} T_j(C_i - 1)\} + T_n * C_i + \sum_{j=i+1}^{n} R_j(C_i - 1) \tag{3}$$

and constant $\beta$ is,

$$\beta = \alpha * (\sum_{j=0}^{n-1} T_j + \sum_{j=1}^{n} R_j) \tag{4}$$

It should be noted that $A_i$ is defined as a node's energy weight and takes communication as well as computation cost into account.

Expression (b) in Table 4 is the constraint, where $x_i$ is the amount of data assigned to be computed on node i and their sum should be equal to the data size $\alpha$, to make sure that all data is computed within the network.

## 3.2 Maximize Minimum Energy Left on Any Node (MME)

The MME's objective is to maximize the minimum energy left on any node. This objective is correlated with the network lifetime. The network is considered to be defunct when any node's energy depth reaches zero. This problem is formulated in two steps as shown Table 5, where we calculate a number of optimal solutions in step 1 and reduce the search area further in step 2.

We consider a sensor node $S_m$ having the least energy level $E_m$ among all the nodes in the network. To maximize $E_m$, we formulate the objective such that the solution consumes the least energy on $S_m$ to complete the task i.e., Minimize $N_m$, expression (c). It is essential to check if the objective unnecessarily penalize other nodes in the network and hence expression (d) restricts the energy consumption on all other nodes such that their energy level do not fall below $E_m$. Expression (e) makes sure that all data is computed within the network. It should be noted that we derive many optimal solutions in step 1, each having different overall energy consumption. An additional constraint (i.e., g) derived from step 1 is utilized in step 2 to minimize the energy consumption on node $N_m$ and subsequently minimize the overall energy consumption to find the solution for our bi-objective problem, from the pool of optimal solutions.

Table 5. Maximize minimum energy left on any node

| MME problem constraints | | | |
|---|---|---|---|
| Step 1 | Minimize | $N_m$ | (c) |
| | | energy at node m is minimum | |
| | Subject to | $\forall\, i \in \{0, .., n\}, i \neq m :\; E_i - N_i \geq E_m$ | (d) |
| | | $\sum_{i=0}^{n} \quad x_i = \alpha$ | (e) |
| Step 2 | Minimize | $\sum_{i=0}^{n} \quad N_i$ | (f) |
| | Subject to | $N_m \leq$ Optimal solution from step 1 | (g) |
| | | $\forall\, i \in \{0, .., n\} :\; E_i - N_i \geq E_m$ | (h) |
| | | $\sum_{i=0}^{n} \quad x_i = \alpha$ | (i) |

Table 6. Minimize energy with deadline constraint

| DD problem constraints | | | |
|---|---|---|---|
| Minimize | $\sum_{i=0}^{n}$ | $N_i$ | (j) |
| Subject to | $\sum_{i=0}^{n}$ | $D_i \leq \delta$ | (k) |
| | $\sum_{i=0}^{n}$ | $x_i = \alpha$ | (l) |

## 3.3 Delay Deadline(DD)

The Delay Deadline(DD) objective is defined to observe the effect of computation allocation on energy and delay for real-time applications in an IoT network. The objective of this metric is to minimize the overall energy consumption while meeting the performance or deadline requirement of the task. The deadline $\delta$ in expression (k) may be a user defined constraint or a quality of service assurance by the network which restricts the end to end delay in the network.

To observe the delay associated with the network, we study various sources of delay on a node. As discussed in section 2, the delay on each node is defined to be the sum of the computation and transmission delay. To simplify the problem and avoid complexity, we assume queuing delay to be zero. Moreover, since the distances between two neighbor nodes is small in an IoT network, the propagation delay is assumed to be zero.

Thus, The delay on each node is formulated as follows:

$$D_i = \overbrace{D_{Pi} x_i}^{Computation\ Delay} \overbrace{+ (\alpha - \sum_{j=0}^{i} x_j + \sum_{j=0}^{i} C_j x_j)/D_{Ti}}^{Transmission\ Delay} \tag{5}$$

$\sum_{i=0}^{n} D_i$ can be simplified in terms of $x_i$ and be represented as,

$$\sum_{i=0}^{n} B_i x_i + \lambda \tag{6}$$

where, $B_i$ is the weight/coefficient of $x_i$ towards the total delay and,

$$B_i = D_{Pi} + \{\sum_{j=i}^{n-1} (C_i - 1)/D_{Tj}\} + C_i/D_{Tn} \tag{7}$$

and constant $\lambda$ is,

$$\lambda = \alpha * (\sum_{j=0}^{n-1} T_j) \tag{8}$$

It should be noted that $B_i$ is defined as a node's delay weight and it takes communication as well as computation delay cost into account.

## 4 EXPERIMENTAL SETUP

A simulator is developed in C-language, which uses platform and network parameters and synthesizes the problem into a solver specific format based on the required objective. We use two solvers to obtain the optimal solution to the problem generated by the simulator, Symphony [Ralphs and Güzelsoy 2005] and Scip optimization suite [Gamrath et al. 2016]. We perform image feature detection using Harris Corner and Canny Edge detection from the OpenCV Suite [Bradski 2000] on NVIDIA Jetson TK1, Intel Galileo and Intel Edison boards to obtain real world values for platform parameters. These are low power embedded platform which are designed for a wide range of low-end to high-end IoT applications processing. The power consumption of all three platforms is measured using picoScope digital oscilloscope and the timing is measured using the CPU timing functions available under the GNU C library. Result for these experiments is shown in Table 7. The communication cost is measured at 0.00525uW/bit for Wi-Fi when transmitting a 40 Mbps User Datagram Protocol (UDP) payload, 0.153 uW/bit for BLE and 185.9 uW/bit for Zigbee in our experiments [Smith 2011].

Table 7. Real value of platform parameters

| Algorithm / Platform | **Harris Corner** | | **Canny Edge** | |
|---|---|---|---|---|
| | J/bit | s/bit | J/bit | s/bit |
| Nvidia Jetson | 3.87E-008 | 1.16E-008 | 2.02E-008 | 6.21E-009 |
| Nvidia Jetson Low Power | 2.47E-008 | 2.36E-008 | 1.01E-008 | 1.31E-008 |
| Intel Galileo | 7.89E-008 | 4.32E-007 | 2.58E-008 | 1.30E-007 |
| Intel Edison | 1.91E-008 | 4.71E-008 | 6.63E-009 | 2.80E-008 |

## 5 OBSERVATIONS

### 5.1 Minimize Overall Energy Consumption (MOEC)

For MOEC objective, the results show that all computation work should be assigned to the node having the least energy weight ($A_i$), to reduce the total power consumption. For a homogeneous network, this is always the first node. It is important to note that the node weight is defined as $A_i$, taking communication as well as computation cost into account. Following is a short proof of why $A_i$ of the first node in a homogeneous network is the smallest:

$$Computation\ cost\ on\ node = x_i * P_i \tag{9}$$

Since the computation factor $P_i$ is the same on every node, we are free to allocate computation on any node with no change to the total computation cost in the network. However,

$$\begin{aligned} Comm^n\ cost\ on\ node = Receive\_data\_size * R_i + \\ Transmit\_data\_size * T_i \end{aligned} \tag{10}$$

The communication cost on a node is proportional to the size of data it receives from and transmits to its neighbor. The size of processed data reduces by some factor based on the compression factor $C_i$. If all data is processed on the first node, all other nodes are only responsible for communicating a smaller size of data incurring the lowest possible cost on that node.

It is observed that, an increase in the computation cost ($P_i$) results in an increase in the weight/coefficient of the node. An increase in transmission cost $T_i$, receive cost $R_i$ and compression factor $C_i$ negatively affects the weight.[Table 8].

Table 8. Proportionality of platform parameters on node's energy weight ($A_i$)

| **Platform Parameter** | **Proportionality** |
|---|---|
| Computation cost per bit | Proportional |
| Transmission cost per bit | Inversely Proportional |
| Receive cost per bit | Inversely Proportional |
| Compression Factor | Inversely Proportional |

Based on the network configurations and platform specifications, in special cases, the lowest node weights could correspond to two or more nodes. In these cases, we can distribute the computation work on the nodes with equal energy weights in any bias and minimize the total energy consumption of the system.

To show that with minimizing the total power consumption as our objective, carrying out the entire computation on one node is always the best solution, we use a counter argument.

Proof. The overall energy consumption of the system when simplified in terms of $x_i$ is given by:

$$\sum_{i=0}^{n} A_i x_i + C$$

Let us assume that two nodes (i.e. Y and Z) have large coefficients and node M has the least coefficient in the above equation. We assume that the overall energy consumption of the system is least when work is assigned to node Y and node Z. Thus,

$$A_Y x_Y + A_Z x_Z + C \leq A_M x_M + C \quad ; \quad x_Y + x_Z = x_M$$

But,

$$A_M \leq A_Y \quad | \quad A_M \leq A_Z$$

Multiplying both sides of the two equations with $x_Y$ and $x_Z$ respectively, we get:

$$A_M x_Y \leq A_Y x_Y \quad | \quad A_M x_Z \leq A_Z x_Z$$

Table 9. Task allocation strategy for MME objective

| | |
|---|---|
| Step 1 | Search node $S_{min}$ with minimum energy $E_{min}$ |
| Step 2 | Find least energy weight node $S_{Amin0}$ earlier in the sequence than $S_{min}$ |
| Step 3 | Allocate maximum percentage of data $x_{S_{Amin0}}$, s.t. available energy remain above $E_{min}$ after processing. |
| Step 4 | If $\sum_{i=0}^{n} x_{S_{Li}} < \alpha$ |
| | Find next least weight node $S_{Li+1}$ |
| | Go to Step 3 |
| | Else, |
| Step 5 | Finish. |

Adding both of the equations, replacing $x_Y + x_Z$ with $x_M$ and adding constant $C$ on both sides,

$$A_M x_M + C \leq A_Y x_Y + A_Z x_Z + C$$

which contradicts our initial assumption. $\square$

Thus carrying out all computation on node M would be the most efficient solution given the MOEC objective.

## 5.2 Maximize Minimum Energy Left on Any Node (MME)

We observe that when the battery life-time constraint is added to our optimization objective, the best solution is to distribute the computation work among several nodes. This task distribution leads to increased overall energy consumption of the network but allows a longer network lifetime.

For MME,the first step is to identify the node with minimum energy $S_{min}$ in the network. The allocation is then done such that all data is computed before reaching the node with the least energy level. The allocation will be done at the node with least energy weight. This minimizes the energy consumption on the node with minimum energy $S_{min}$ by reducing the communication overhead because the size of the workload reduces after computation on previous node. In case none of the previous node can be used for allocation due to energy level restriction, the node $S_{min}$ or the ones after can be used based on their energy weights. Table 9 shows this allocation strategy. It is noted that when the cost of computation is small, noticeably less than the cost of communication, the rate at which the battery of the first node depletes is becoming small such that the data computation may never be offloaded to the next node.

To better understand the distribution of computation in a heterogeneous network we provide a simple example. The energy weights of the first ten nodes of a twenty five node network are shown in Figure 2. To account for heterogeneity, we divide the network into two group of nodes i.e. from node 0 to node 4 (Type I) and from node 5 to node 24 (Type II). Type I nodes have a lower computation and communication energy cost (weight) than type II and initially have equal energy available to them at 10800 J(Fully charged AA batteries). Node 5 is an exception with the least available energy in the network at some random value, 10400 J. In Figure 3, the y-axis represent the available energy on each node and the x-axis represent each nodes in the network. MME 0 is

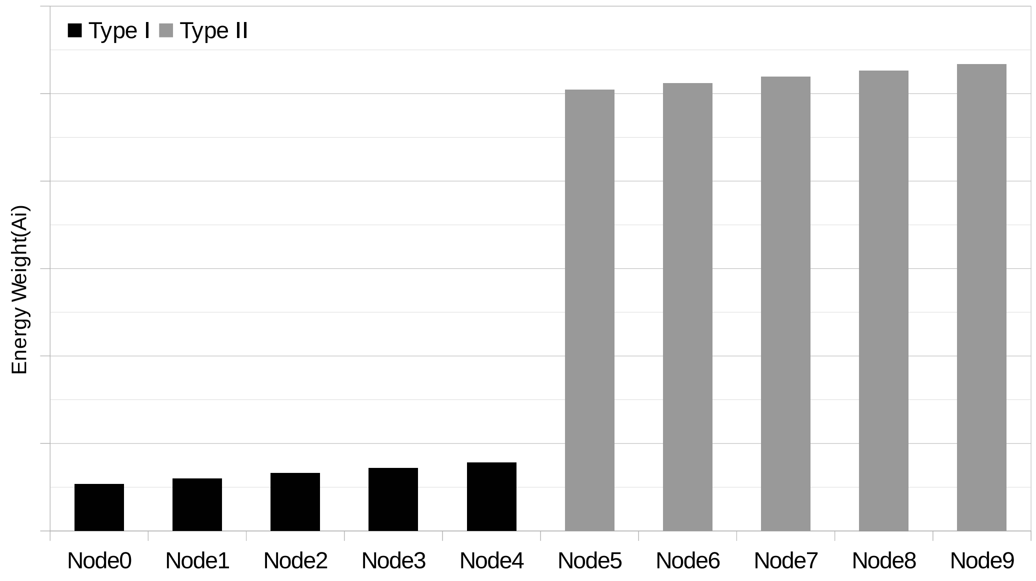


Fig. 2. Comparison of available energy on nodes

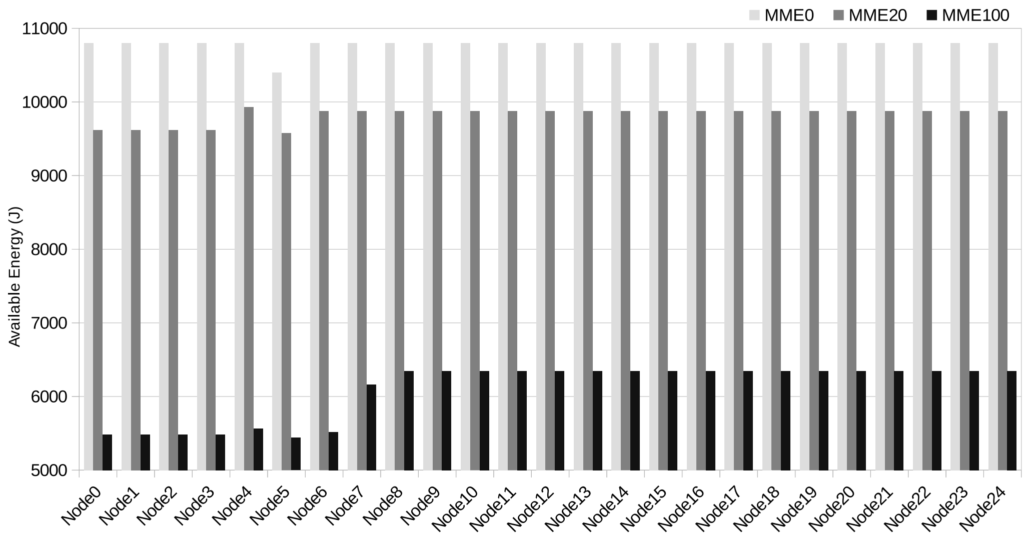


Fig. 3. Comparison of available energy on nodes

the initial state of the network. MME 20 represents the energy available in the battery of each node after twenty tasks are computed on the network and MME 100 is the state of the batteries after one hundred tasks have been computed on the network. Initially, 100% of data is computed on the first node, since it has the least energy weight. After a few tasks, the available energy difference between node 0 and node 5 is not large enough to keep computing all the data. Hence, part of the data would now be computed on node 1, as it is the next appropriate node with least energy weight.

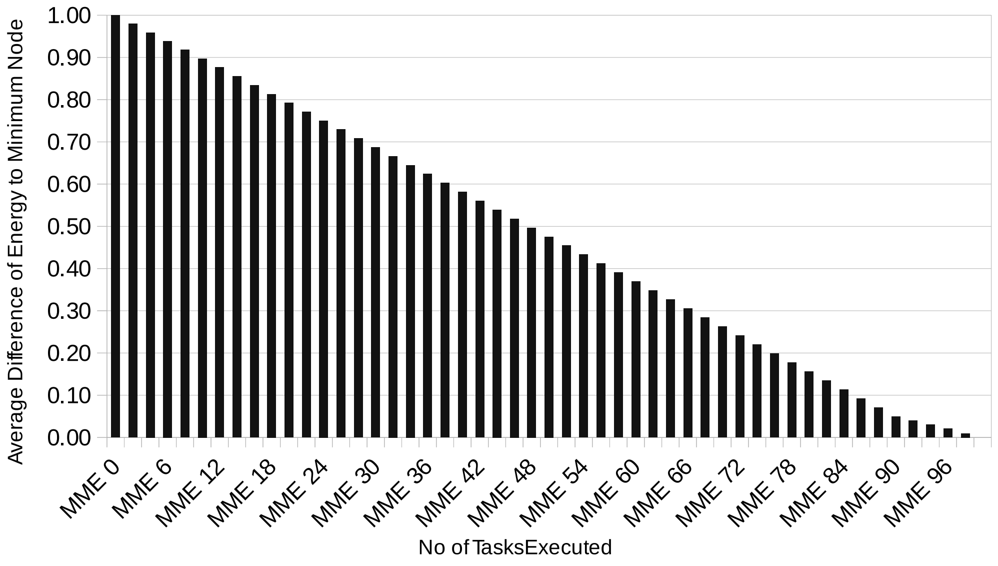


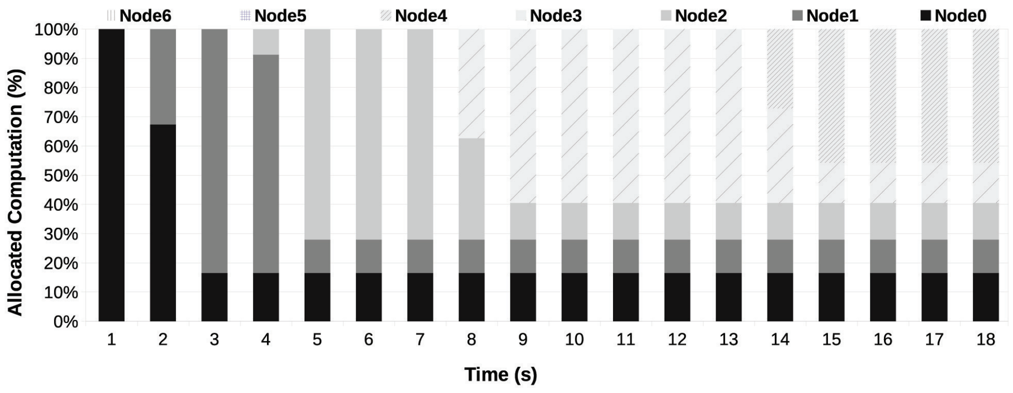
Fig. 4. Variation in energy levels of all nodes across the network

This trend keeps repeating and after 20 tasks(MME 20) node 0 to node 3 have to transfer some computation to node 4. Since, node 4 was only communicating data till now, it has sufficient energy to take some computation work. As node 4 keeps computing more data their energy levels drop to the same level. Computation distribution allows the energy level on all nodes in the network to deplete together rather than one or a group of nodes being completely depleted. This trend is further discussed in detail while discussing Figure 5 and Figure 6.

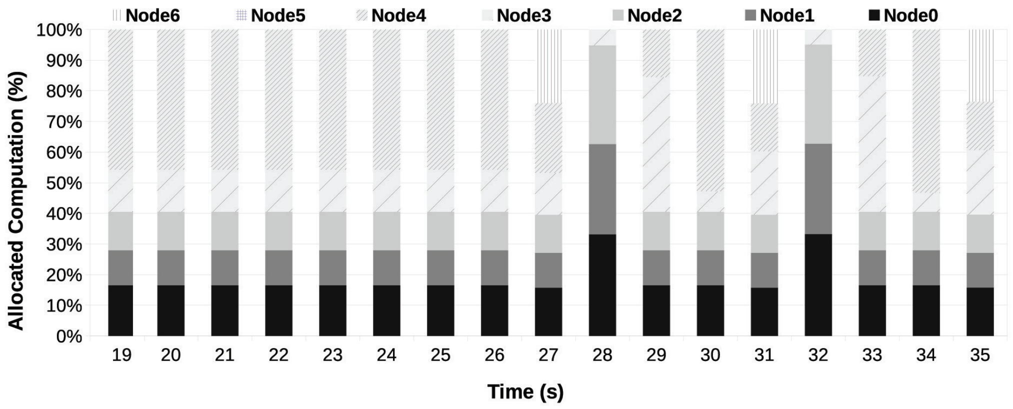
We notice that the depletion of energy on nodes is balanced by the solution and the average variation in energy levels of all nodes in the network is reducing as more tasks are processed. The solution attempts to avoid reducing the energy of the battery on the node(s) with minimum energy. This trend can be seen in Figure 4.

We explain the allocation by using two figures, one which shows us the energy levels at different nodes after completion of each task and the other showing allocations per task.

Figure 5 shows assignment of computation work during the simulation time. Each column in the graph shows the allocation of computation suggested by the simulator. The y-axis is the percentage of work allocated on a node and the x-axis is the simulated time. Every time interval (1 second) a new task is injected into the network, computed at optimal locations and communicated to the base station. As expected, initially, the first node or the node with the least weight gets majority of the allocation to reduce the energy consumption and gradually the work is offloaded to the nodes with the least weight from the pool of available nodes to preserve the network's life-time. Figure 6, 7, 8 show the available energy on each node during simulation time. Figure 6 shows the overall trend from time zero to the end of simulation and Figure 7 and Figure 8 are for shorter time intervals for better presentation of trends and results. In Figure 7, the point where transfer of data to the next node for computation increases can be identified as the time where the difference in available energy on the node with majority of data allocation is negligible to the energy level of the node with least available energy in the network, i.e. Node 5 in this case. At this point, the node cannot keep consuming the same amount of energy and computation load needs to be offloaded to



(a) Time 0s - 18s



(b) Time 19s - 35s

Fig. 5. Assignment of computation work during the simulation

the next appropriate node to prevent its energy depleting below the least allowable level in the network, i.e. energy level of Node 5.

Thus, after the initial allocation, alternate nodes are allocated with the majority of the computation, and the initial nodes are allocated a constant computation load such that their available energy stays above the least allowable level in the network. As the energy level on node 4 reaches close to node 5, we observe a slightly different behavior. Node 5 is skipped for computation allocation and computation is moved to node 6. This causes the energy depletion rate of node 5 to increase since a larger data size is communicated to the next node(Processed Data + Unprocessed data). Subsequently, when next time a new task is injected into the network, the gap between the energy level of node 0 - node 4 to that of minimum battery node increases such that more data is allocated back to these nodes. This pattern occurs recursively till the energy level at node 6 depletes to that of node 5. Next, node 7 is allocated some data and the pattern repeats from node 0 to node 7. This pattern is clearly shown in Figure 8 and Figure 6.

An interesting observation from Figure 6 is regarding the network life-time which can be inferred from the slope plotted between the available energy and time. The slope represents the depletion

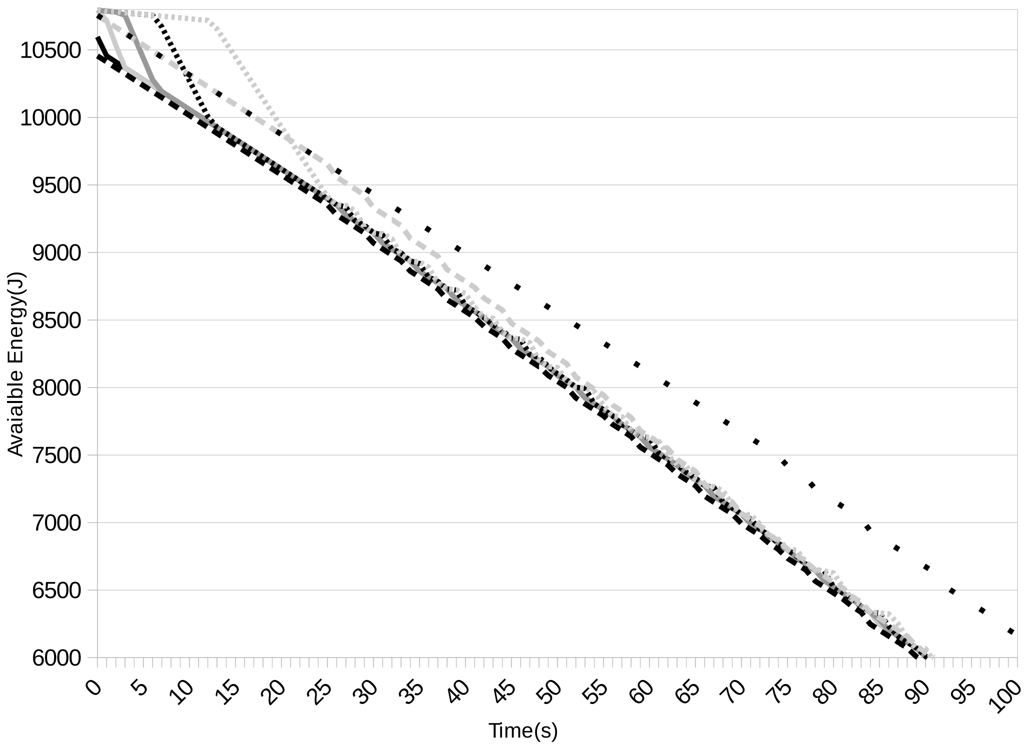


Fig. 6. Available energy on nodes during MME simulation [Time 0s-100s]

rate of available energy on a node. The slope for each node is proportional to the computation and communication cost of the node and can be used to understand whether the computation will be offloaded to the next efficient node with the least weight. If the slope of the node with minimum energy is smaller (negative) than the most efficient node, the computation will not be offloaded. The network lifetime increases as the slope positively increases towards zero and decreases when it moves in the other direction.

## 5.3 Delay Deadline (DD)

For the delay deadline optimization objective, we study the effect of deadline requirement on the computation allocation. For a homogeneous network, the first node is the most efficient node in terms of energy cost and delay cost. If all data is processed on the first node, the overhead of communication is minimized, as data size gets reduced by some compression $C_i$. As we compute data away from the source node, the delay increases and reaches its maximum when the entire data is computed on the last node. It should be noted that the node's delay weight is proportional to the computational delay per bit and inversely proportional to the transmission rate and the compression factor (See Table 10).

We also study a heterogeneous system made of two types of nodes placed alternatively in groups. We look at the $A_i$ and $B_i$ (Introduced in section 3.3.1 and 3. 3.3), the energy weight and the delay weight of each node to understand allocation. In Figure 9 when comparing identical type of groups, we notice that all nodes of a group closer to the source are always more efficient in terms of energy (first node of the group being the most efficient) than a group placed later in the network. The

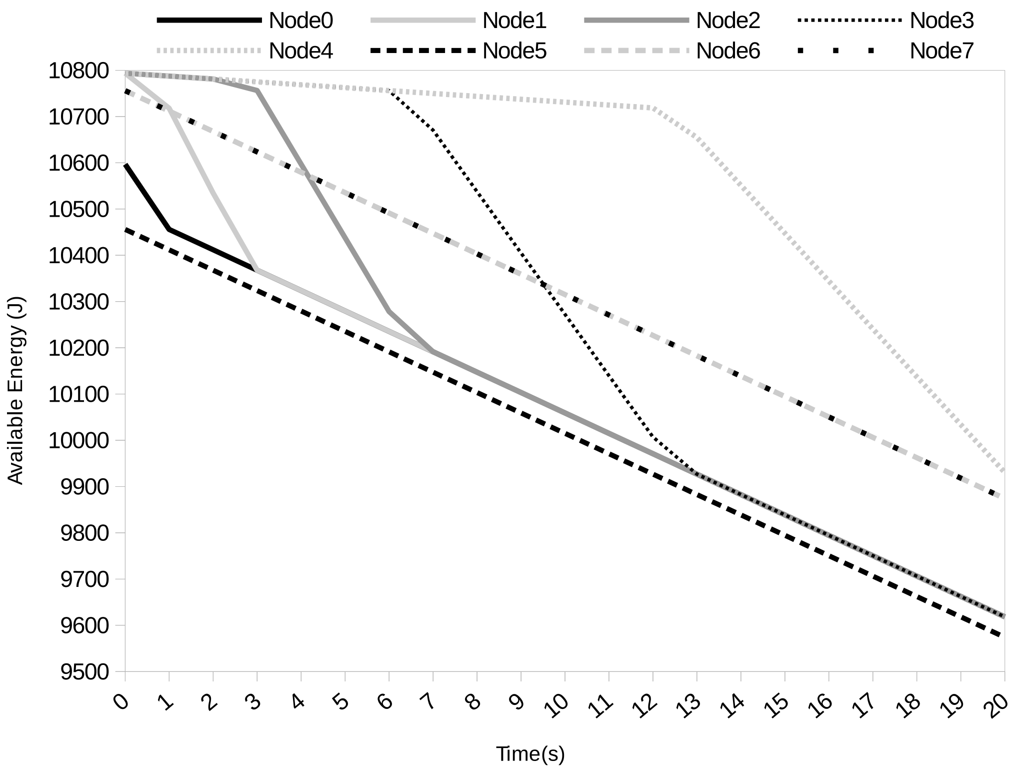


Fig. 7. Available energy on nodes during MME simulation [Time 0s-20s]

Table 10. Proportionality of platform parameters on delay weight ($B_i$)

| **Platform Parameter** | **Proportionality** |
|---|---|
| Computational delay per bit | Proportional |
| Transmission rate | Inversely Proportional |
| Compression Factor | Inversely Proportional |

same trend can be seen in Figure 10 for delay weights for each node. These observations can be used to select candidate nodes for computation allocation in a network.

In Figure 9, node 10 has the least weight towards energy consumption and is the best candidate to start our allocation strategy with. All subsequent nodes within the homogeneous group and of other identical groups in the network are not considered candidates as they will have a higher weight towards energy consumption as well as delay. All data can be allocated to be processed on the candidate node and checked for deadline adherence. This is the most efficient node of the network and network designers should make sure that most of their workload deadlines can be met with this node itself. If the solution does not meet the deadline, we search for the next candidate to offload the data. It should be noted that if the node with the smallest weight in terms of energy consumption and delay is the same and does not meet the deadline, it is impossible to find any other

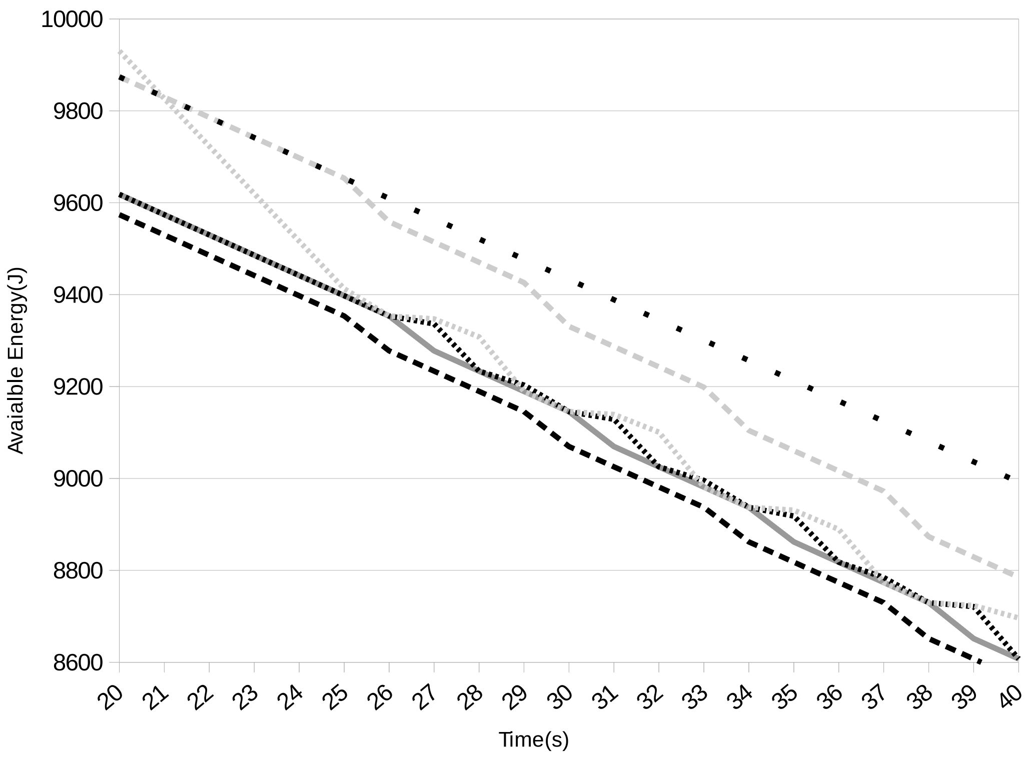


Fig. 8. Available energy on nodes during MME simulation [Time 20s-40s]

computation allocation that adheres to the deadline. The next candidate is node 0 in this example, as this is the node with the least energy weight from the left over nodes with an improved delay weight. Allocation of data is now iteratively increased on this new candidate and reduced on the previous one until the solution meets the required deadline. The optimal solution for computation allocation will always have maximally two nodes sharing computation.

The complete allocation strategy is listed in Table 11 and another example is used to explain it.

We study another example of a heterogeneous network with energy and delay weights of each node as shown in Figure 11 and Figure 12 respectively. This example shows how the allocation algorithm operates. Following the algorithm, we first find the node with the least and second least energy weight in the network. Allocation on these nodes incur the lowest energy cost and become a good starting point. In this example these nodes are node 16(least energy weight) and node 8(second least energy weight). The next step is to allocate all data on node 16 and check if this solution adheres to the required deadline. If the deadline is met, we have the optimal solution for the defined deadline. If the deadline cannot be met with this node and a node with a smaller delay weight is not present in the network, the deadline cannot be met (step 5). In step 6, we find all appropriate candidates for transfer of allocation to meet the deadline, which was already discussed earlier in this section. The candidates in this example are node 8 and node 0 as they have much smaller delay weight $B_i$ than node 16. Here node 9 to node 15 and node 1 to node 7 are not candidates for transfer of allocation as the first node in a homogeneous group has the least energy and delay weight, i.e. node 0 and node 8. With extensive simulations we found out that the optimal solution includes two or less nodes. Hence, in step 7, we find out the allocation distribution on (node 16, node 8) and (node 16, node 0), such that the deadline is met. We compare the total energy consumption between the two allocation combination and choose the one that uses the minimum energy. It should be noted that if there exists an allocation such that the deadline is met with the top two

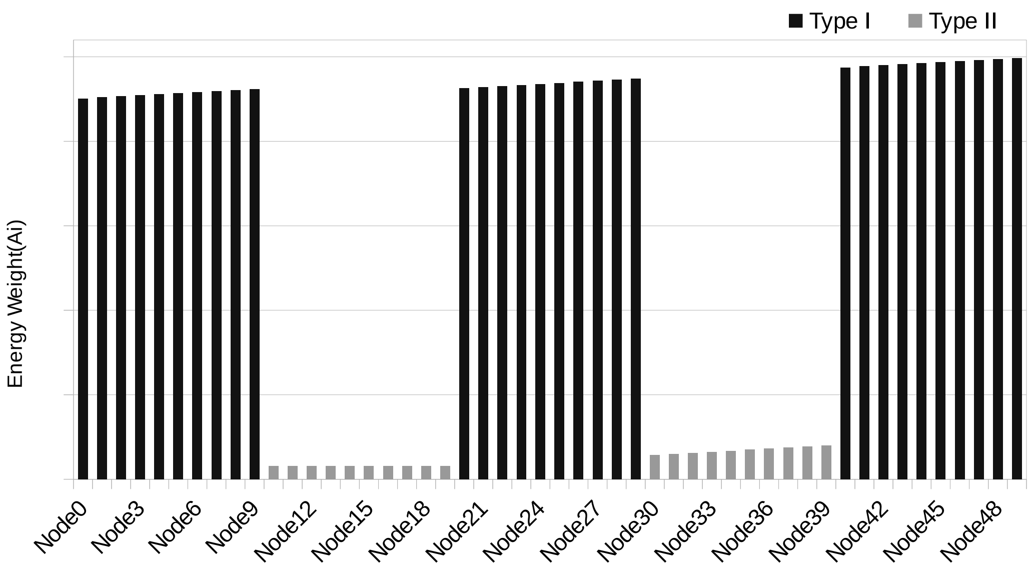


Fig. 9. Energy weight ($A_i$) for each node in the network

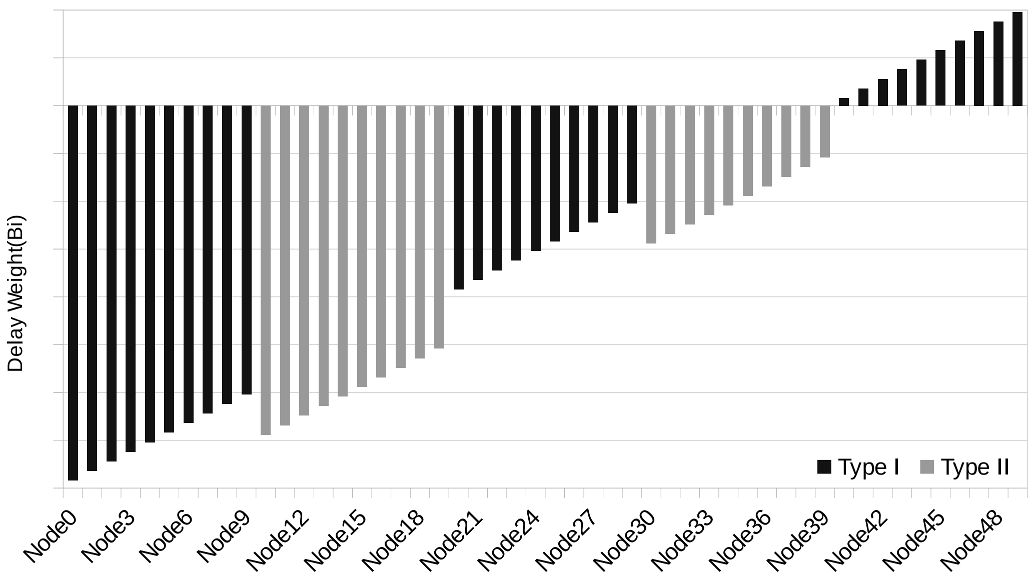


Fig. 10. Delay weight ($B_i$) for each node in the network

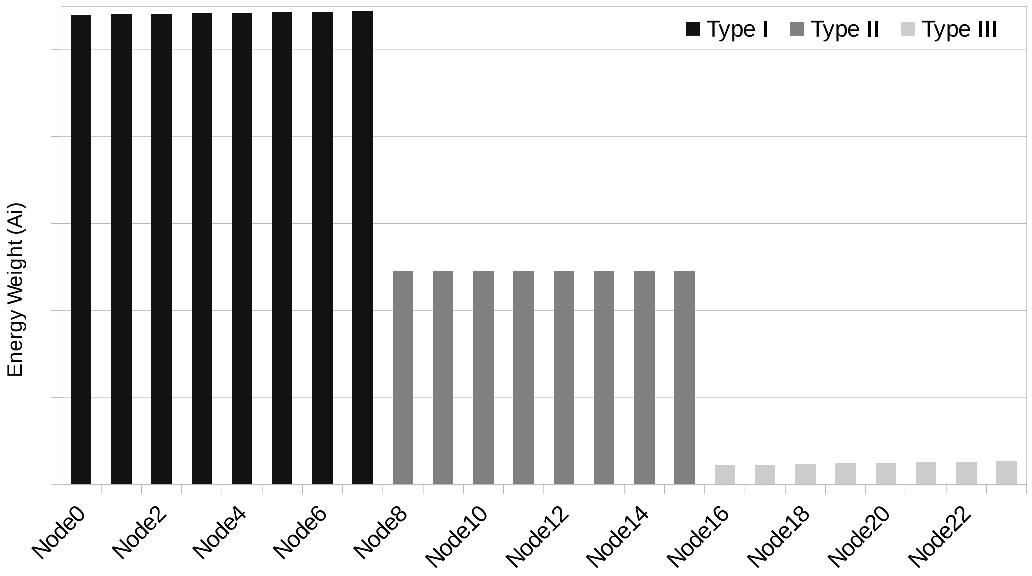


Fig. 11. Delay weight ($A_i$) for each node in the sample network

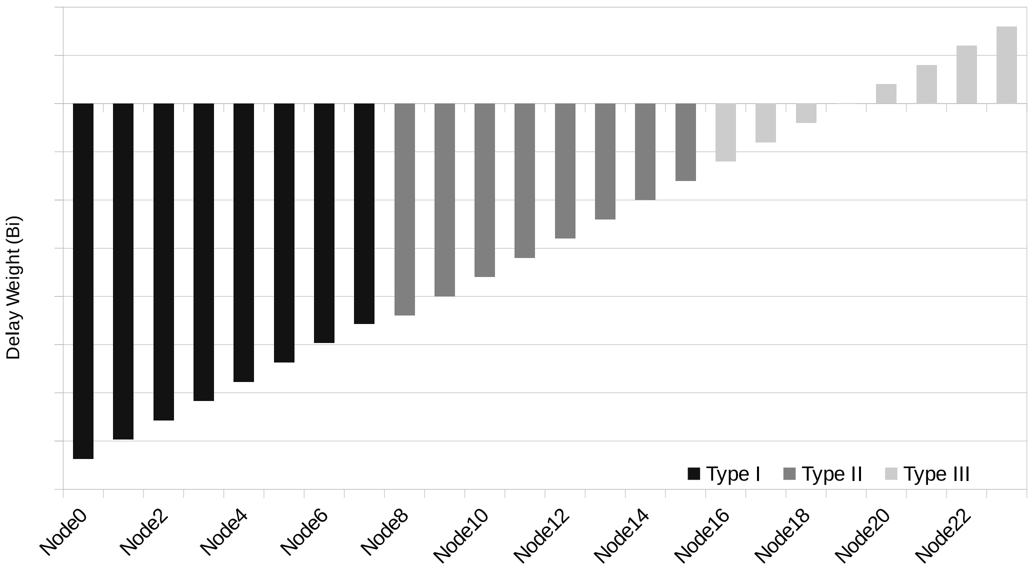


Fig. 12. Delay weight ($B_i$) for each node in the sample network

candidates, which are node 16 and node 8 in this example, the optimal solution can be found with a combination of the best candidate (node 16) and other candidates(node 8, node 0), i.e. (node 16, node 8) and (node 16, node 0). If not, we have to explore other possible combinations, like (Node8, Node0) in this example and compare the total energy consumption between all of them.

Table 11. Task allocation strategy in a heterogeneous network - DD objective

| | |
|---|---|
| Step 1 | Find node $S_m$ with least energy weight $A_m$ |
| Step 2 | Find node $S_n$ with least energy weight s.t. |
| | $A_n \geq A_m$ and $B_n < B_m$ |
| Step 3 | Allocate 100% data to be computed on $S_m$ |
| Step 4 | Check if solution meets deadline |
| | If True, finish. |
| | Else, |
| Step 5 | Check if $B_m$ is least in the system |
| | If True, Deadline cannot be met. |
| | Else, |
| Step 6 | Find candidates for transfer of allocation |
| | i.e. nodes where, $A_i \geq A_{i-1}$ and $B_i < B_{i-1}$ |
| Step 7 | For all candidates $S_i$, find $(x_i, x_{mi})$ such that, |
| | $B_i x_i + B_m x_{mi} \leq \delta$, where $x_i + x_{mi} = \alpha$ |
| Step 8 | If there exists no solution for $B_n x_n + B_m x_m \leq \delta$ |
| | Store $Z \longleftarrow Min(\forall i \in candidates : A_i x_i + A_m x_m)$ |
| | $S_m = S_n$, Go to Step 2 |
| | Else, |
| Step 9 | $Min(\forall i \in candidates : A_i x_i + A_m x_m, Z)$, finish. |

We now show that for the deadline delay objective two or less than two nodes will only provide an optimal solution. For this proof we use a counter argument.

Proof. Let us assume we have two solutions that adhere to the deadline $\delta$, one where two nodes participate and the other where an extra node participate in the solution. Let us name these nodes, $S_D, S_E, S_F$ with energy weights $A_D, A_E, A_F$.
We have the following assumptions:

$$A_D < A_E < A_F \tag{11}$$

$$\overbrace{x_D + x_E + x_F}^{3\,node\,solution} = \overbrace{x_G \quad + \quad x_H}^{2\,node\,solution} = \alpha \tag{12}$$

We study three cases and assume that energy consumption is less when allocation is done on three nodes compared to two nodes or less. We then prove that our assumption is incorrect.

Case I : When deadline is met with $S_D, S_E$

Case I.A: When moving from a two to three node solution some part of $x_H$ is offloaded to the third node for computation

$$x_G = x_D \quad \& \quad x_H = x_E + x_F \tag{13}$$

Comparing energy consumption between the two solutions i.e. $(S_D, S_E, S_F)$ and $(S_D, S_E)$:

$$A_D x_D + A_E x_E + A_F x_F + \beta < A_D x_G + A_E x_H + \beta$$

Using equation 13 and removing equal terms on both sides, we get

$$A_F < A_E$$

Which contradicts our initial assumption in equation 11.
Case I.B: When moving from a two to three node solution some part of $x_G$ is offloaded to the third node for computation

$$x_H = x_E \quad \& \quad x_G = x_D + x_F \tag{14}$$

Comparing energy consumption between the two solutions

$$A_D x_D + A_E x_E + A_F x_F + \beta < A_D x_G + A_E x_H + \beta$$

Using equation 14 and removing equal terms on both sides, we get

$$A_F < A_D$$

Which contradicts our initial assumption in equation 11.
Case I.C: When moving from a two to three node solution some part of $x_G$ and $x_H$ is offloaded to the third node for computation

$$x_G > x_D \quad \& \quad x_H > x_E \tag{15}$$

Comparing energy consumption between the two solutions

$$A_D x_D + A_E x_E + A_F x_F + \beta < A_D x_G + A_E x_H + \beta$$

Using equation 12, we get

$$A_D x_D + A_E x_E + A_F x_F < A_D x_G + A_E (x_D + x_E + x_F - x_G)$$

Removing equal terms and moving remaining terms to LHS.

$$x_F(A_F - A_E) + (x_G - x_D)(A_E - A_D) < 0$$

But,

$$x_F(A_F - A_E) > 0, since \quad A_F > A_E \quad \& \quad x_F > 0$$
$$(A_E - A_D) > 0, since \quad A_E > A_D$$
$$(x_G - x_D) > 0, since \quad x_G > x_D$$

which contradicts our initial assumption.
Case II : When deadline can be met with $S_E, S_F$, but not $S_D, S_E$
In this case if the deadline can be met with $S_D, S_E, S_F$, it can also be met with $S_D, S_F$.

$$i.e. \quad B_D > B_E > B_F \tag{16}$$

Comparing energy consumption of the two solutions, i.e. $(S_D, S_E, S_F)$ and $(S_E, S_F)$, we assume the three node solution consumes less energy than a two node solution.

$$A_D x_D + A_E x_E + A_F x_F + \beta \le A_E x_G + A_F x_H + \beta$$

Simplifying the equation and moving all terms to LHS

$$x_D(A_D - A_E) + (x_H - x_F)(A_E - A_F) \le 0 \tag{17}$$

In the above equation,

$$x_D > 0$$
$$(A_D - A_E) < 0$$
$$(A_E - A_F) < 0$$

Hence, for the equation 17 to hold true, we have two cases

$$(x_H - x_F) \geq 0 \quad or \quad (x_H - x_F) < 0$$

We now prove that in case of $(x_H - x_F) \geq 0$ the deadline cannot be met, and for $(x_H - x_F) < 0$, $S_E, S_F$ provide better energy consumption.
Case II.A: $(x_H - x_F) \geq 0$
Comparing delays between the two solution i.e. $(S_D, S_E, S_F)$ and $(S_E, S_F)$,

$$B_D x_D + B_E x_E + B_F x_F + \lambda \leq B_E x_G + B_F x_H + \lambda$$

Using equation 12 and simplifying,

$$x_D(B_D - B_E) + (x_H - x_F)(B_E - B_F) \leq 0 \tag{18}$$

But,

$$\begin{aligned} x_D &> 0 \\ (B_D - B_E) &> 0 \\ (B_E - B_F) &> 0 \\ (x_H - x_F) &\geq 0 \end{aligned}$$

which contradicts equation 18.
Case II.B: $(x_H - x_F) < 0$ or $(x_F - x_H) > 0$
Comparing energy consumption between the two solutions, i.e. $(S_D, S_E, S_F)$ and $(S_D, S_F)$:

$$A_D x_D + A_E x_E + A_F x_F + \beta \leq A_D x_G + A_F x_H + \beta$$

Simplifying the equation and moving all terms to LHS

$$x_E(A_E - A_D) + (x_F - x_H)(A_F - x_D) \leq 0 \tag{19}$$

But,

$$\begin{aligned} x_E &> 0 \\ (A_E - A_D) &> 0 \\ (A_F - x_D) &> 0 \\ (x_F - x_H) &> 0 \end{aligned}$$

which contradicts equation 19.

Hence, we prove that allocation on two nodes will always consume less energy for a delay requirement than a three node solution. We can similarly prove that a three node solution is always better than a four node solution and so on. □

## 5.4 Solver Overhead

We assume that the optimization solver is run on the source node every time a new task is injected into the network. The source node is aware of each node's computation and communication cost. This information needs to be updated to the source node initially by all nodes and again in case the node is replaced or if their was a significant change in the battery level without the involvement of computation and communication cost. The cost of updating these parameters is negligible over the lifetime of the network. The solver has a very small overhead in comparison to the task studied in this work. For a small task(standard definition image frame of size 922Kb), if we compute 100% data on a Nvidia Jetson node with BLE($D_{T_i} = 1.00E - 006$), the computation delay is 0.046s and communication delay at 70% compression rate is 2.26s, bringing the total delay of the operation to 2.30s. This implies that in a homogeneous system (all nodes are Nvidia Jetson with BLE), assuming that all data is computed on the first node which is the most efficient solution, the total delay is 2.3s for the first node plus an additional 2.26s per node for every other node the data needs to go through to reach to the base station. In comparison, the average solver run time is 2ms. For a 10 node homogenous network, the solver delay overhead is 0.008%. For larger tasks, the solver run-time remains the same leading to an even lower solver overhead. The energy consumption of the solver is 0.06J on average for any data size whereas for a 10 node homogeneous network the total energy consumption to compute and communicate date to the base station is 0.46J. It should be noted that the solver in its current state has many I/O operations to log intermittent results for this study which when removed will further lower the energy and delay overhead.

## 6 FUTURE WORK AND CONCLUDING REMARKS

Energy constraints and performance requirements are real issues for IoT networks. The allocation of computation in an IoT network is a challenging problem given the diversity of network, optimization goal, computation and communication overhead and type of task to be performed. This work is an effort to thoroughly study and analyze task computation and communication allocations in IoT networks given three important optimization metrics, namely energy-efficiency, available energy maximization and delay-deadline optimization. The metrics were formulated into an ILP(Integer Linear Programming) problem and the results show that when energy efficiency is the only objective, it makes sense to process all data on a single node. In the presence of a deadline constraint, the results show that it is beneficial to distribute the computation across the network. The observations can be used by network designers to understand the energy delay trade off within the network and appropriately design the network such that the most energy efficient node or least energy weight node can be identified and used to assure quality of service with low energy consumption. The formulated allocation algorithm can be used to quickly find an optimal solution without requiring to run complex algorithms, lowering the overhead. This work can be further used in interpreting the optimal allocation for a completely connected network. After an optimal path is identified, computation allocation can be distributed to make the system more efficient. We conclude that with an energy efficiency metric, for a completely connected network the optimal solution would be to allocate computation on one node i.e. the node with the least energy weight.

## REFERENCES

Dennis Baker and Anthony Ephremides. 1981. The architectural organization of a mobile radio network via a distributed algorithm. *IEEE Transactions on communications* 29, 11 (1981), 1694–1701.

G. Bradski. 2000. OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

Neda Edalat et al. 2009. A price-based adaptive task allocation for wireless sensor network. In *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*. IEEE, 888–893.

Anthony Ephremides, Jeffrey E Wieselthier, and Dennis J Baker. 1987. A design concept for reliable mobile radio networks with frequency hopping signaling. *Proc. IEEE* 75, 1 (1987), 56–73.

Matthew Ettus. 1998. System capacity, latency, and power consumption in multihop-routed SS-CDMA wireless networks. In *Radio and Wireless Conference, 1998. RAWCON 98. 1998 IEEE*. IEEE, 55–58.

Robert G Gallager et al. 1979. *A distributed algorithm for minimum weight spanning trees.* Citeseer.

Gerald Gamrath et al. 2016. *The SCIP Optimization Suite 3.2.* Technical Report 15-60. ZIB, Takustr.7, 14195 Berlin.

Hrishikesh Jayakumar et al. 2014. Powering the internet of things. In *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM, 375–380.

Musaravakkam Samaram Krishnan. 2010. Dynamic voltage scaling. (June 15 2010). US Patent 7,739,531.

Mihai T Lazarescu. 2013. Design of a WSN platform for long-term environmental monitoring for IoT applications. *IEEE Journal on emerging and selected topics in circuits and systems* 3, 1 (2013), 45–54.

Hosein Mohammadi Makrani and Houman Homayoun. 2017a. Memory requirements of Hadoop, Spark, and MPI based big data applications on commodity server class architectures. In *IEEE International Symposium on Workload Characterization (IISWC)*. 112–113.

Hosein Mohammadi Makrani and Houman Homayoun. 2017b. MeNa: A Memory Navigator for Modern Hardware in a Scale-out Environment. In *IEEE International Symposium on Workload Characterization (IISWC)*. 2–11.

Hosein Mohammadi Makrani, Amir Mahdi Hosseini Monazzah, Hamed Farbeh, and Seyed Ghassem Miremadi. 2014. Evaluation of software-based fault-tolerant techniques on embedded OSâĂŹs components. In *International Conference on*

*Dependability (DEPEND)*. 51–57.

Hosein Mohammadi Makrani, Setareh Rafatirad, Amir Houmansadr, and Houman Homayoun. 2018. Main-Memory Requirements of Big Data Applications on Commodity Server Platform. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE.

Hosein Mohammadi Makrani, Shahab Tabatabaei, Setareh Rafatirad, and Houman Homayoun. 2017. Understanding the role of memory subsystem on performance and energy-efficiency of Hadoop applications. In *Eight International Green and Sustainable Computing Conference (IGSC)*. 1–6.

Teresa H Meng and Volkan Rodoplu. 1998. Distributed network protocols for wireless communication. In *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, Vol. 4. IEEE, 600–603.

Virginia Pilloni and Luigi Atzori. 2011. Deployment of distributed applications in wireless sensor networks. *Sensors* 11, 8 (2011), 7395–7419.

Virginia Pilloni, Mauro Franceschelli, Luigi Atzori, and Alessandro Giua. 2012. A decentralized lifetime maximization algorithm for distributed applications in wireless sensor networks. In *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 1372–1377.

Johan Pouwelse, Koen Langendoen, and Henk Sips. 2001. Dynamic voltage scaling on a low-power microprocessor. In *Proceedings of the 7th annual international conference on Mobile computing and networking*. ACM, 251–259.

Ted K Ralphs and Menal Güzelsoy. 2005. The SYMPHONY callable library for mixed integer programming. In *The next wave in computing, optimization, and decision technologies*. Springer, 61–76.

Volkan Rodoplu and Teresa H Meng. 1999. Minimum energy mobile wireless networks. *IEEE Journal on selected areas in communications* 17, 8 (1999), 1333–1344.

Hossein Sayadi, Hamed Farbeh, Amir Mahdi Hosseini Monazzah, and Seyed Ghassem Miremadi. 2014. A data recomputation approach for reliability improvement of scratchpad memory in embedded systems. In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2014 IEEE International Symposium on*. IEEE, 228–233.

Hossein Sayadi, Amir Houmansadr, Setareh Rafatirad, Houman Homayoun, et al. 2018a. Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*. ACM, 212–215.

Hossein Sayadi, Nisarg Patel, Avesta Sasan, and Houman Homayoun. 2017. Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures. In *Computer Design (ICCD), 2017 IEEE International Conference on*. IEEE, 129–136.

Hossein Sayadi, Nisarg Patel, Avesta Sasan, Setareh Rafatirad, Houman Homayoun, et al. 2018b. Ensemble learning for effective run-time hardware-based malware detection: A comprehensive analysis and classification. In *Proceedings of the 55th Annual Design Automation Conference*. ACM, 1.

Hossein Sayadi, Divya Pathak, Ioannis Savidis, and Houman Homayoun. 2018c. Power conversion efficiency-aware mapping of multithreaded applications on heterogeneous architectures: A comprehensive parameter tuning. In *Design Automation Conference (ASP-DAC), 2018 23rd Asia and South Pacific*. IEEE, 70–75.

Jordi Serra et al. 2014. Smart HVAC control in IoT: Energy consumption minimization with user comfort constraints. *The Scientific World Journal* 2014 (2014).

Timothy Jason Shepard. 1995. *Decentralized channel management in scalable multihop spread- spectrum packet radio networks*. Ph.D. Dissertation. Massachusetts Institute of Technology.

Suresh Singh, Mike Woo, and Cauligi S Raghavendra. 1998. Power-aware routing in mobile ad hoc networks. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, 181–190.

Phil Smith. 2011. Comparing Low-Power Wireless Technologies. https://www.digikey.com. (2011).

Yang Yu and Viktor K Prasanna. 2005. Energy-balanced task allocation for collaborative processing in wireless sensor networks. *Mobile Networks and Applications* 10, 1-2 (2005), 115–131.

Chunsheng Zhu, Victor CM Leung, Lei Shu, and Edith C-H Ngai. 2015. Green Internet of Things for smart world. *IEEE Access* 3 (2015), 2151–2162.

Jinghua Zhu, Jianzhong Li, and Hong Gao. 2007. Tasks allocation for real-time applications in heterogeneous sensor networks for energy minimization. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, Vol. 2. IEEE, 20–25.