electronics

MDPI

Article

Sustainable IoT Sensing Applications Development through GraphQL-Based Abstraction Layer

Raees Khan [1,*] and Adnan Noor Mian [2,3]

1 Department of Computer Science, FAST—National University of Computer and Emerging Sciences, Lahore 54000, Pakistan
2 Department of Computer Science, Information Technology University (ITU)-Punjab, Lahore 54000, Pakistan; adnan.noor@itu.edu.pk
3 Computer Laboratory, Department of Computer Science and Technology, University of Cambridge, Cambridge CB3 0FD, UK
* Correspondence: raees.khan@nu.edu.pk

Received: 17 January 2020; Accepted: 21 March 2020; Published: 28 March 2020

Abstract: Internet of Things (IoT) networks are mostly comprised of power-constrained devices, therefore the most important consideration in designing IoT applications, based on sensor networks is energy efficiency. Minor improvement in energy conservation methods can lead to a significant increase in the lifetime of IoT devices and overall network. To achieve efficient utilisation of energy, different solutions are proposed such as duty cycling optimization, design changes at the MAC layer, etc. In this paper, we propose a new approach to overcome this challenge in cloud-based IoT sensing applications, based on integration of an abstraction layer with constrained application mechanism. To achieve energy conservation and efficient data management in IoT sensing applications, we incorporate modules of efficient web framework with cloud services, in order to minimize the number of round trips for data delivery and graph-based data representation. Our study is the first attempt in the literature, to the best of our knowledge, which introduces the potential of this integration for achieving the aforementioned objectives in the target applications. We implemented the proposed interfacing of abstraction layer in constrained applications, to develop a testbed using Z1 IoT motes, Contiki OS and GraphQL web framework with Google cloud services. Experimental comparisons against baseline REST architecture approach show that our proposed approach achieved significant reductions in data delivery delay and energy consumption (minimum 51.53% and 52.88%, respectively) in IoT applications involving sensor network.

Keywords: IoT; GraphQL; abstraction layer; sensing applications; WSN

1. Introduction

Innovative sensor modules, next-generation wireless networks and modern cloud services are the enabling tools, headed for the realisation of the large scale Internet of Things (IoT) applications [1]. IoT devices having sensing abilities, enable us to interact with the environment to collect data and send them to the cloud through communication networks. The number of such sensing modules that have to be connected to the Internet is increasing at rapid speed and is expected to be 500 billion by 2030 [2].

Traditional sensor network standards and technologies are not expected to sufficiently furnish the connectivity for abundant sensing and interaction with the physical world, thus undermining the development of innovative IoT applications on large scale. The rapid growth of modern technologies in communication, computation and sensing platforms is enabling a wide range of IoT sensing applications providing services in an overly connected way [3]. The energy demands of IoT networks are increasing because of the steadily multiplying number of participating sensor devices and a wide range of modern applications [4]. According to an estimation, 12.3 billion mobile devices are expected to be connected

by 2022, which also includes the IoT devices using machine-to-machine communication [5]. Therefore, for these battery operated IoT platforms, efficient energy utilisation is a crucial factor for an extended network life-time, without replacement of the devices or recharging of the batteries.

Data communication is the most energy consuming operation for constrained sensor modules and it can become a bottleneck, especially in IoT sensing applications that generate continuous or plenty of data. Examples of such applications include dense IoT networks and sensor networks with multimedia applications. Such applications use Internet-connected wireless sensor networks with dense deployments of devices in the field or send multimedia (audio, video and images) data from the physical environment. A report by Cisco on visual networking says that during 2015–2016, video monitoring through Internet connectivity increased by 72%, with data sending per month increasing from 516 Petabyte (PB) to 883 PB [6]. These IoT sensing networks provide very useful services in applications such as disaster management, industrial automation, military surveillance, monitoring services in home, agriculture and wildlife, elderly as well as general healthcare and safety. Figure 1 shows a layered architecture of the IoT-based system with typical scenarios for sensing applications.

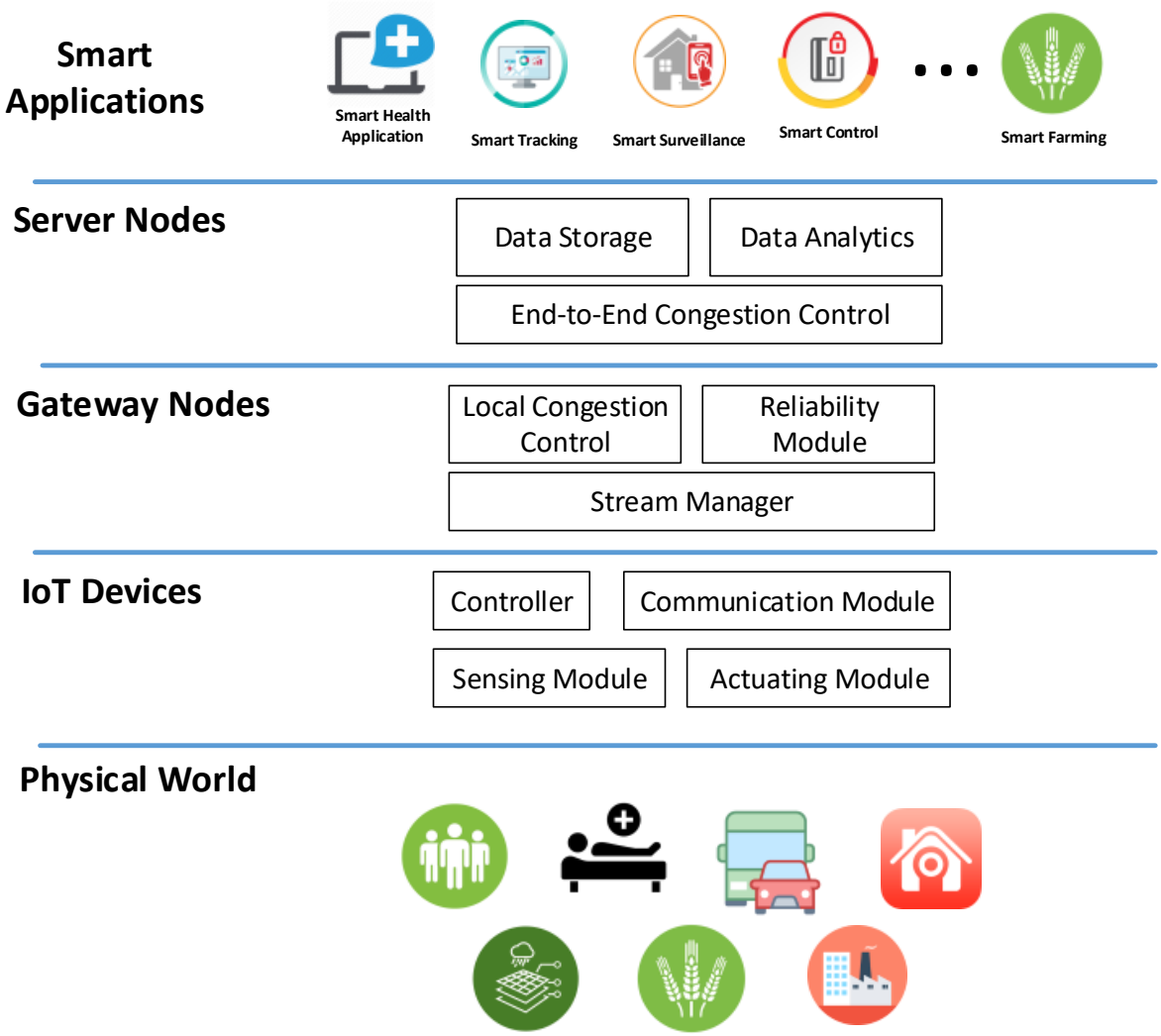


Figure 1. Layered architecture of IoT-based system with typical application scenarios for sensor networks.

Despite having significant use cases of sensor network-based IoT applications, there are many technical challenges in the practical realisation of their deployments on large scale. These challenges mostly narrow down towards the constrained nature of IoT devices in terms of energy, causing transmission delays and information fidelity. Energy is considered the scarcest resource in Wireless

Sensor Networks (WSN) and many different approaches can be found in the literature for efficient utilisation of node energy, to maximise individual nodes' as well as overall network lifetime.

Wireless sensor nodes are usually deployed on a large scale having complex network topology and participating devices mostly are battery operated. This is because large scale wired infrastructure for continuous power supply and connectivity to WSN nodes is not a viable option [7,8]. Similarly, delivery of multimedia content (audio/video) in sensor networks, presents new distinct design challenges, and for which specialised Wireless Multimedia Sensor Networks (WMSN) are designed. Energy conservation is even more challenging in WMSN, because data are complex and large in size, whereas nodes are battery powered mostly [8,9]. Excessive traffic may be injected into the network in such application cases, which is not sustainable in constrained IoT networks. As a consequence, the network can become congested, which results in high latency and packet losses. This becomes a massive source of energy consumption/wastage, especially due to lost packets which were transmitted and partially traveled on their multi-hop path. IoT devices have to consume additional energy if those packets may need re-transmissions for information fidelity in reliable application or due to critical nature of the payload.

The commonly used approach in the literature for energy management is radio duty cycling, that is enabling the IoT devices to turn off the transceiver during idle periods, having no data transmitting or receiving [10]. However, in IoT sensing applications such as those discussed above, data communication is to be continuous and/or with many packets [7]. Therefore, duty cycling opportunities are expected to be very minimal, making this approach less functional and not helpful for energy saving. In the light of duty cycling being less effective in such IoT applications, controlling the excessive traffic generation becomes a key factor for achieving significant energy conservation. Traffic management of the network can be done through controlling data sending rate and number of round trips in the form of number of packets. Thus, efficient data management becomes crucial for longer operating times of IoT devices as well as the lifetime of the network.

The channel contention is also increased when the data flows towards the neighbourhood of the gateway node. This issue can be better seen in the example scenario presented in Figure 2. In such circumstances, the nodes which are close to the gateway node are particularly under severe pressure. In commonly used sensor network architectures, this kind of pressure is because of the extra contention near a single sink, due to multi-path data forwarding to it. This phenomenon is also commonly referred to as the energy-wave problem, as shown in Figure 2. Nodes shown in red colour are under heavy pressure and their residual energy can quickly deplete due to excessive forwarding on behalf of other nodes. Therefore, these nodes are critical in the IoT sensing applications and solutions like proposed in this paper are needed to prolong their lifetime by avoiding unnecessary forwarding, as otherwise the network would soon be disconnected.

The contributions of the study presented in this paper for efficient energy consumption and data management in IoT sensing applications are as follow:

i. We proposed a scheme based on interfacing efficient web framework (GraphQL) and cloud services with constrained application protocol.
ii. We implemented a testbed involving real IoT devices for evaluation of the proposed approach in the target applications.

The rest of the paper is organised as follows. Section 2 is about related work. Section 3 presents possible architectures for cloud-based IoT sensing applications considered in this research work. The proposed approach and its implemented components are presented in Section 4. Setting up of the experimental testbed is explained in Section 5. Results and discussions are included in Section 6. The energy-wave problem alternate resolution is in Section 7. Section 8 concludes the work.

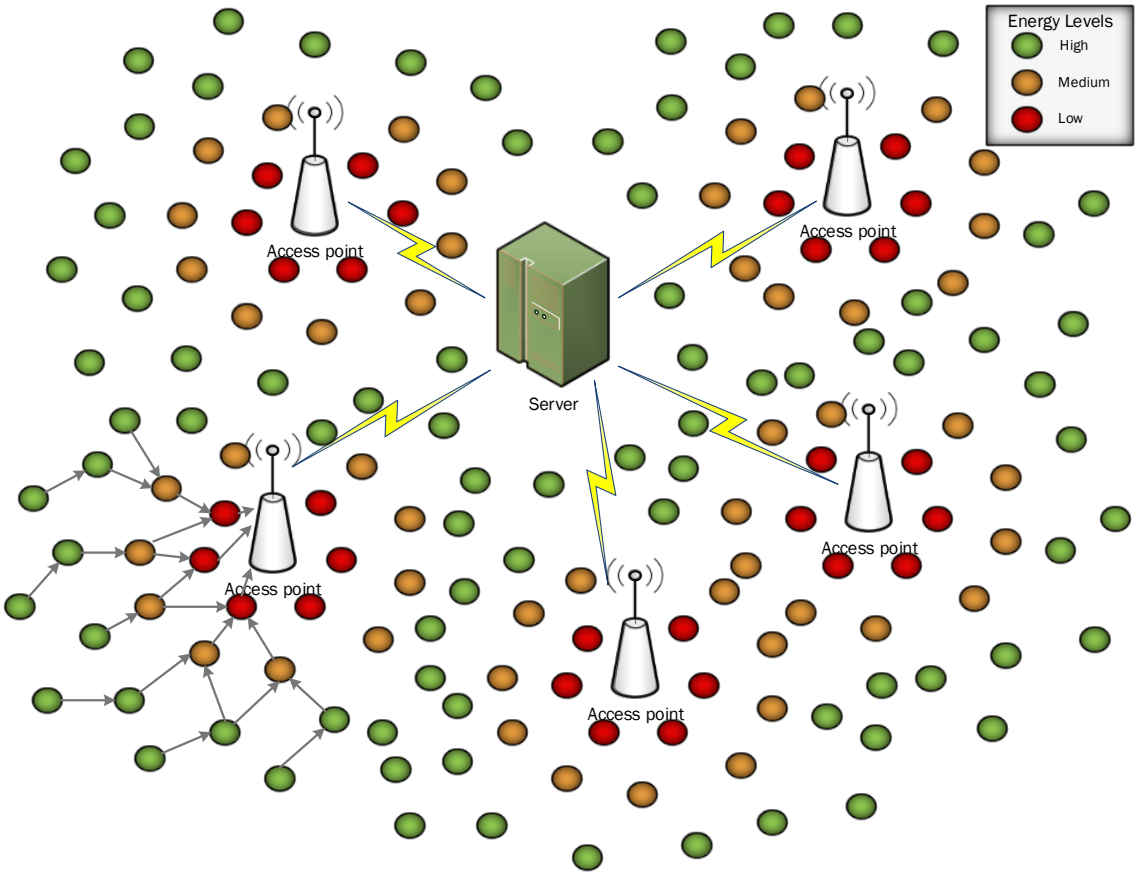


Figure 2. The energy-wave problem in sensor network architecture.

2. Related Work

As the replacement of traditional web services based on REST architecture, Facebook released GraphQL [11] and Falcor [12] was introduced by Netflix. Such solutions mainly target decreasing bandwidth utilisation through implementing reduction in traffic generation involving connected nature data. We base our work in this paper on incorporating these approaches for sensing application scenarios in Contiki-based cloud IoT networks. In the literature, applying such an abstraction layer for IoT applications has limited existence related to work in this paper, as we are not using the traditional enhancement scheme of duty cycling which is found in many of the optimisations for energy usage.

Hartig and Pérez analysed GraphQL to understand properties of the language and also gave details of semantics and complexity of its use [13,14]. They provided a solution to handle large size of GraphQL response for Internet scenario and obtained computation as well as delay improvements, to show it is helpful for more robust web interfaces. Vázquez-Ingelmo et al. [15] looked into complex data-driven ecosystems which need high levels of interoperability and REST APIs are used as common approach to achieve it. They leveraged benefits of GraphQL protocol and presented a case study achieving benefits using GraphQL as main API in different parts of the ecosystem. Taelman et al. [16] presented GraphQL-LD, a linked data querying in linked open data cloud for application development, which is pushing for linked data consumption.

Sabharwal et al. [17] stated that usage of energy can be decreased by utilising the data communication reducing schemes through power-aware applications. Our proposed approach in this work is also reducing the data traffic by utilising an abstraction layer for combining the queries to a single endpoint for multiple resources. Han et al. [18] emphasised reducing active slots in sensor devices and their focus is on optimising the duty cycling mechanism. Piyare and his co-authors [19] discussed hardware and networking details in low power wake-up radios. Our research is different as

we do not utilise the duty cycling scheme, as it is discussed in above section to be not fulfilling in the case of our target high data rate applications. We achieve the power conservation and delay minimising through incorporating the application layer for data transaction optimisation. Ram D. Sriram [20] introduced a gateway as a service for transforming data from application layer protocol, for enhancing the performance. We also use the same protocol CoAP at the application level in our work but integrated it alongside the abstraction layer of graph-based querying and server layers. Translation like this between application layer protocols for IoT implementation has also been introduced in some other research works (e.g., [21–23]). These works mainly emphasise the mapping of IoT protocol CoAP and Internet communication of HTTP, using a service such as Firebase. Similarly, F. Nogatz et al. [24] worked on comparing GraphQL-based cloud service and traditional REST scheme, which gives a helpful insight for transforming these implementations to one another for mapping purpose. In [25], the work introduced REST Query Language (RQL) to minimise the number of data transactions in mobile applications.

The work presented in this paper is an extension of our previous study given in [26]. Previously, we used GraphQL for duty cycling-based sensor networks to optimise active time slots through round trips saving. The study presented in this paper, aims at achieving energy efficiency through communication optimisation using GraphQL layer, in active IoT applications involving high data sending. Several application layer frameworks and protocols are developed for IoT-based applications, e.g., IoTivity, CoAP, REST etc. We integrated an abstraction layer based on GraphQL that works on top of application CoAP layer, as a wrapper to optimise communication in IoT sensing applications. In the experimental study presented in this paper, we implemented proposed GraphQL-based approach and compared it with solutions based on REST architecture, which is the prevalent technology for this purpose. Significant performance gain was observed when the results of the proposed scheme were compared with this base-line scheme. We are quite hopeful that a similar performance gain can be achieved in other cases using the proposed scheme.

3. Application Models for Cloud-based IoT

3.1. REST Architecture and Cloud-based IoT

REST (Representational State Transfer) is a web architectural style, which is built around Uniform Resource Identifier (URI), HyperText Markup Language (HTML) and HyperText Transfer Protocol (HTTP). It defines some constraints for web services and properties. It presents principles for server to expose resources to its clients. However, in the modern age of agile technologies, REST APIs are lacking the flexibility needed for the clients with dynamic requirements.

REST architecture is bound with HTTP protocol for data transport and it requires a separate API endpoint for each server resource. For requirement of flexibility and maintainability, APIs may need re-writing, when the domain data structure, components or users requirements are changed, because endpoint definition in advance for every data request cannot be provided. An application has to address multiple resources, if the data requested are of the complex nature from multiple entities involving different types. Thus, it results in multiple round trips to collect the data.

REST being prevalent web architectural style is still used for accessing variety of data and even for hierarchical nature data. However, for cloud-based IoT networks, it becomes draining because of the limitations of the resources of involved constrained devices. It may become more worrying in case of applications needing continuous large amount of data sending. Such type of applications need communication efficiency, with less number of round trip times for data collection. Hence, IoT sensing applications desire new solutions for cloud-based scenarios.

Figure 3 shows a scenario to the illustrate inefficiency of REST architecture in IoT sensing applications submitting data to the cloud. Here, we have an IoT device attached to a smart car containing two sensors, which is posting data to a cloud server. Each sensor needs to send its data to a separate resource at server through its exposed endpoint. By using the REST architecture, multiple requests are needed to submit data to each resource. Many examples of more complex scenarios can be easily found in real-world IoT applications, e.g., a smart car having multiple sensors such as speed sensor,

acceleration sensor, location sensor, engine and fuel status sensors. Here, for the sake of clarity in the example scenario, only two sensors are being considered, i.e., location sensor and speed sensor. Data collected from these two sensors need to be sent to a cloud server where each sensor node has a separate resource defined for its data, with a unique REST endpoint, i.e., /Car/Loc and /Car/Speed. Each sensor datum will require a separate POST request and, therefore, in a smart city having thousands of smart cars and other IoT devices, we can easily imagine the explosion in the network traffic. As a result, IoT applications can suffer network congestion, resulting in packet losses and energy wastage.

To avoid such situations and improve the lifetime of the sensor nodes, as well as of the overall IoT network, a suitable approach is used to decrease the number of data transactions communicated by the application. One way to achieve this minimisation can be using data fusion at IoT node level or through in-network data aggregation, but the constrained nature of IoT devices poses challenges for adopting such complex data reducing algorithms. The other solution to control network traffic in cloud-based IoT networks can be reducing the number of round trips needed for data posting from sensor nodes, and we adopt this approach in our work.

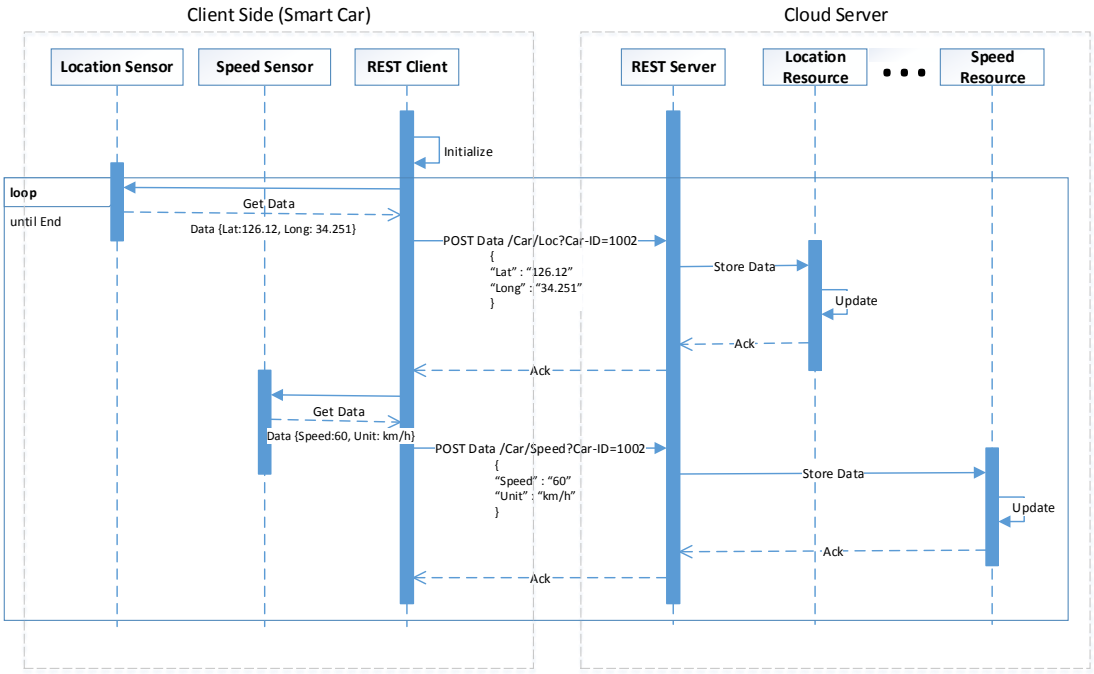


Figure 3. Smart car example scenario with multiple requests from IoT device to POST data to multiple end-points using REST architecture.

3.2. GraphQL Framework and its Feasibility

GraphQL employs message exchange similar to any classical request–response system, but it provides a single API endpoint on server for communication. GraphQL query language is used for building client applications and it aims to combine all the data requests into one, resulting in reduced number of requests for data retrieval/delivery. This approach can result in improvement of IoT network performance by reducing the number of round trips for data exchange. It is also efficient due to the reduction in size of data transactions, as clients can state their requests for particular fields of needed data for their working.

GraphQL, developed in 2012, was internally used for three years by Facebook, but it was open-sourced with a specification release in October 2015 [11]. Its latest release was released on 10 June 2018, and the most recent working draft on 20 June 2019. It is increasingly adopted since its release and each day it handles hundreds of billions of API calls by Facebook applications' data fetching.

Implementation of its modules in popular programming languages and tools for its interaction with existing back-end technologies are available.

GraphQL enables flexible queries on linked (No SQL graph-based) data and its single endpoint approach is backed by a structured hierarchical system for combining data of various sources. It does the representation of data as set of nodes which is able to have nesting and relations with other nodes. Its query is a set of hierarchical data fields requested and only those which are required are returned to the client, resulting in reducing the size of data. Data requests are to be sent to a unique endpoint for the unification, which means it requires a single URI to present all available data via its single API. Figure 4 presents same example IoT scenario of a smart car with two sensors to illustrate the working of the GraphQL-based client–server system. We can observe that, unlike REST architecture, data from multiple sensors can be posted to the server in a single transaction. This way its reduction in the number of round trips makes efficient use of available resources of the IoT network.

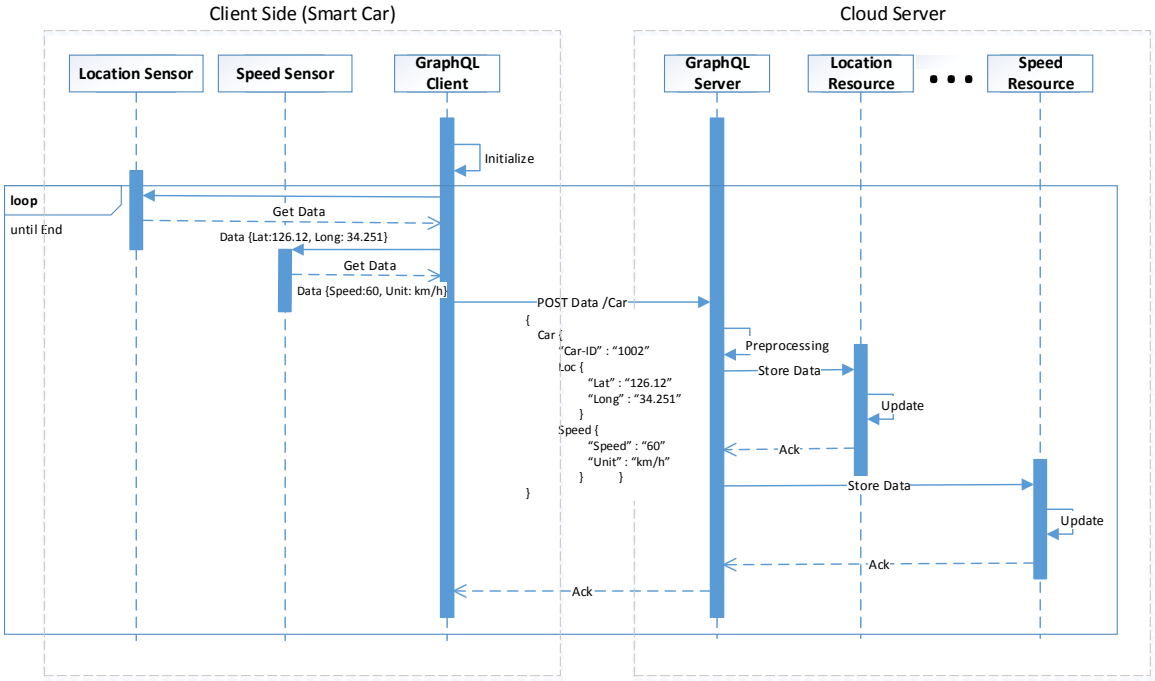


Figure 4. Smart car example scenario with IoT device POSTing data to multiple end-points in single request using GraphQL architecture.

By analysing its efficiency features suitable for IoT sensing applications, our proposal for integration of this web framework with IoT protocols and cloud services appears to be a viable solution.

4. Proposed Approach

Energy is the most scare resource in IoT devices as they are mainly powered by dry batteries and, according to a study [27], around 60% of the node energy is consumed by the communication sub-system. Due to a shared wireless channel, the chances of collision in data transmission and packet losses arise when data transmission rate or number of active data sending nodes are increasing. A suitable approach to address this challenge is to reduce the number of transmissions at individual node level, which is beneficial in two ways: (a) energy consumption is reduced by the fraction of transmissions reduced; and (b) reduced number of transmissions results in reduction of collision in data transmission, thus avoiding energy wastage in packets re-transmission. Our proposed scheme addresses the energy efficiency challenge.

As discussed in the previous section, GraphQL framework has suitable features which can be used in sensing applications and it has no implementation dependency, which means it can be integrated with

constrained IoT libraries. We expected it to furnish efficient energy and bandwidth utilisation as well as reduction in communication delays in IoT applications. Therefore, we were motivated to propose an approach based on integration of its abstraction layer, cloud services and constrained application protocol for IoT. In this paper, we interface a GraphQL abstraction layer with IoT CoAP protocol both on client and server modules, and integrate in cloud services for efficient energy utilisation and data management. We implemented a testbed based on our proposed incorporation of web/cloud technologies for IoT applications, to evaluate on different scenarios posing various data injection challenges.

The proposed integration of graph-based web framework and constrained application request/response structure is targeted for both scalar and multimedia sensing data. The experimental study presented in this paper focused on the high injection of sensing traffic that is not sustainable by the sensor network. Traffic minimisation is also possible by using local data fusion at IoT node level to reduce the data into some scalar values. However, such techniques can fail where sensing data are of different types and to be delivered to different end-points. The proposed interfacing of GraphQL modules with CoAP protocol has the features to optimise network traffic of diverse data, serving the respective data repositories and the clients. GraphQL framework adds hierarchical data communication possibility with data having multiple relationships, otherwise needing separate trip times for delivery in REST style. IoT networks with large deployments and frequent data communication even having scalar sensor data can become congested and need lightweight schemes such as the one presented here.

The proposed approach for wireless sensor networks-based IoT applications involves cloud services and maintaining client–server architecture, where implementation is running client side on sensor nodes and cloud services are providing residence for the server side. We integrate the GraphQL abstraction module on our client side in the form of GraphQL query language and we incorporate its default module named as Runtime on the server side hosted in Google cloud functions. This implemented Runtime provides data exchange according to the configuration by clients where they can mention the filters, operations and structure of the received data in the aggregated queries. Figure 5 presents the block components of the proposed scheme integrating cloud services, web framework, and interfacing with IoT CoAP protocol, showing their positions on client/server side of the system.

In our proposed methodology, GraphQL abstraction layer is used to represent IoT data in graphs and queries notion, which makes the paradigm scalable and flexible. If we need modification of data objects, it is done as the modification in graphs, making evolution and changes very simplified. The graph query module on client side of the implementation provides flexibility by allowing IoT clients to design any specific query. Therefore, server does not need implementing endpoints for all possible data requests. The client sensor layer can specify the data exchange requirement in a declarative manner and it achieves only the required data transaction, nothing extra is bundled in the object.

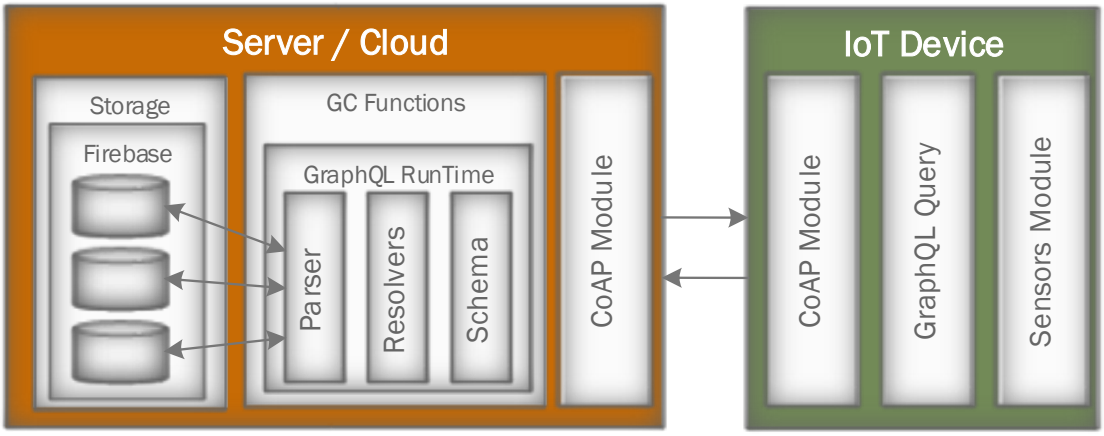


Figure 5. Components of the integrated framework on the client and server sides.

The integration of GraphQL and cloud services results in the reduction of round trips for CoAP application, when IoT devices have to deliver data to the cloud storage. For example, if an IoT device has to post a data sample having one-to-many association, the classical REST way of communicating these data will need one endpoint for the parent entity and one for the child entity, and it will use two request–response round trips to send the data. In our proposed implementation, these two entities are unified in a single query on client side and delivered in a single round trip to the server hosted on cloud functions. Thus, this approach can result result in saving many round trips of communication for large complex data having many complex structures, thus saving energy and time for submitting the data.

The web framework integrated in the proposed implementation provides a unified abstraction layer having an internal schema and resolver/parser for storage repositories. It also allows the change in technologies of front and back-end for varieties of applications. In the following subsections, we further discuss the integrated client-side graph-based data query module used to send data requests of the application and Runtime module of the server side hosted at cloud functions, which entertains these requests. We show the level of proposed scheme implementation in Figure 6.

In some cases, data processing can be a major source of energy consumption for IoT nodes when local processing is done to reduce the data into some scalar values. In such cases, our proposed strategy will be limited, as we are targeting to achieve energy efficiency through communication management. Thus, the utility of the proposed scheme is in proportion to the energy consumed in communication. Local processing can be used to reduce communication traffic and, in that case, those processing algorithms will be more power consuming modules [9]. However, the constrained nature of devices is again a hurdle in application of complex data processing modules. Even after using some local processing techniques, network traffic still can be have ample volume, thus our technique of communication optimisation remains a useful proposal.

We discuss the details of block components of the proposed scheme in the subsequent sections.

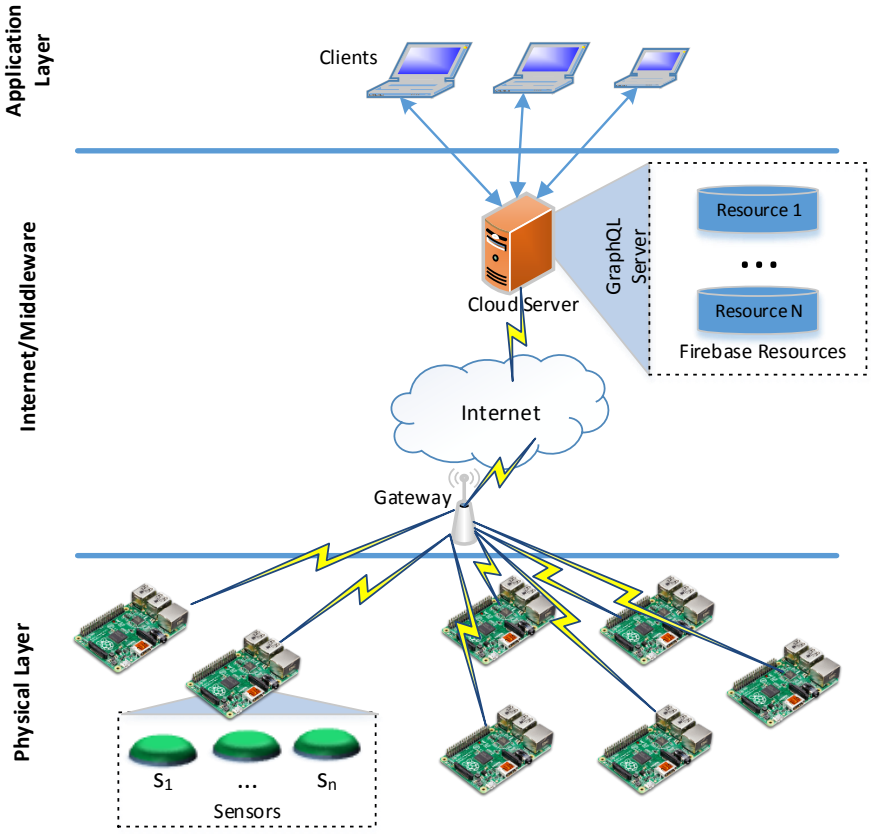


Figure 6. Layered setup of the proposed architecture for supporting IoT sensing applications.

4.1. Server-Side Implementation

Base libraries for server-side environment for fulfilling graph-based data queries provide reference specification. We adopted these libraries as base for server-side implementation of the interfacing in proposed scheme. The integrated Runtime server makes a unified abstraction layer at the REST level to contain internals of the system.

The framework Runtime server hosted at cloud functions does not have data model definition. It implicitly uses a logical model-based over storage layer of the server side, with a hierarchical graph-based view. Thus, the incorporated server layer makes underlying data representable in a directed labeled multi-graph having nodes and labels. This Runtime is not dependent on the data back-end and other underlying modules in the architecture. Client applications of different IoT devices can send requests to single endpoint of the server. Client requests must be sent to it in a clearly defined query formed as plain text, structuring it hierarchically in a unified entity involving all date associations. The server module integrated with CoAP layer executes the received query and sends back a JSON object, having entities/relations from multiple data sources of the storage service.

4.2. Client-Side Interfacing

The query language module integration is the main enabling component of the client side in our proposal. This module integrated with CoAP layer is used to send data exchange messages to the respective server abstraction hosted on the cloud services. As stated above, format of the queries is similar to JSON objects, but these requests are supported by the server layer in the cloud. At the Runtime server integrated in our proposed approach, the general structure of possible queries is defined prior in the application.

In our proposed setup, the GraphQL server module implementation provides a unified interface in cloud-side system for data storage service interactions. Its back-end revolvers have specifications of the data fields which are available for client queries. POST request of CoAP layer is employed for query sending containing object as the payload. The client side uses query structure to dictate the response object construction, making the proposed approach a client specific scheme.

The schema module in the server side contains the set of fields to define types of involved data objects. The server environment is able to provide result filtering, because query fields are similar to program functions, able to take arguments and return the required values only. The size of the result object is the combination of sizes of requested fields, unlike the classical architecture which extracts all the fields of an interface causing over-fetching of data. This over-fetching control feature provides further efficiency on client side in constrained devices-based IoT sensing applications.

4.3. Scheme Formulation

Cloud computing, fog computing [28] and edge computing [29] are different computational models being explored in the paradigm of IoT. Cloud computing focuses on centralised management, whereas the other two involve computation near the source devices. These technologies are vital for the realisation of IoT. We can use the cloud computing for the delegation of computationally intensive processing and centralised management of IoT. It can facilitate implementing and executing different software applications across diverse devices involved in the IoT network. This suits the incorporation of abstraction layer in our proposed approach. Implementation of GraphQL-based proposal is workable as it is an application layer framework and we can use it with any back-end frameworks meeting its protocol specification. Another point of its implementation feasibility is that GraphQL is not dependent on any specific transport protocol for client–server communication because of keeping its data of queries and responses in the payload.

In typical IoT networks, sensor nodes are generating data at some specific rate or when some particular event is detected. Data are usually sent to the neighbouring gateway node where necessary processing can be carried out. Gateway nodes then forward the data to the central server (sink) node for further analysis. Figure 7 shows the typical scenario of IoT network where multiple source nodes are connected to the central sink node through intermediate gateway nodes. Let's consider S_i be the ith

source generating data of total size d_i with a particular data rate r_i. Every source node S_i is connected to a gateway node G_i where GraphQL client is deployed to perform necessary aggregation and then data are forwarded to the sink node.

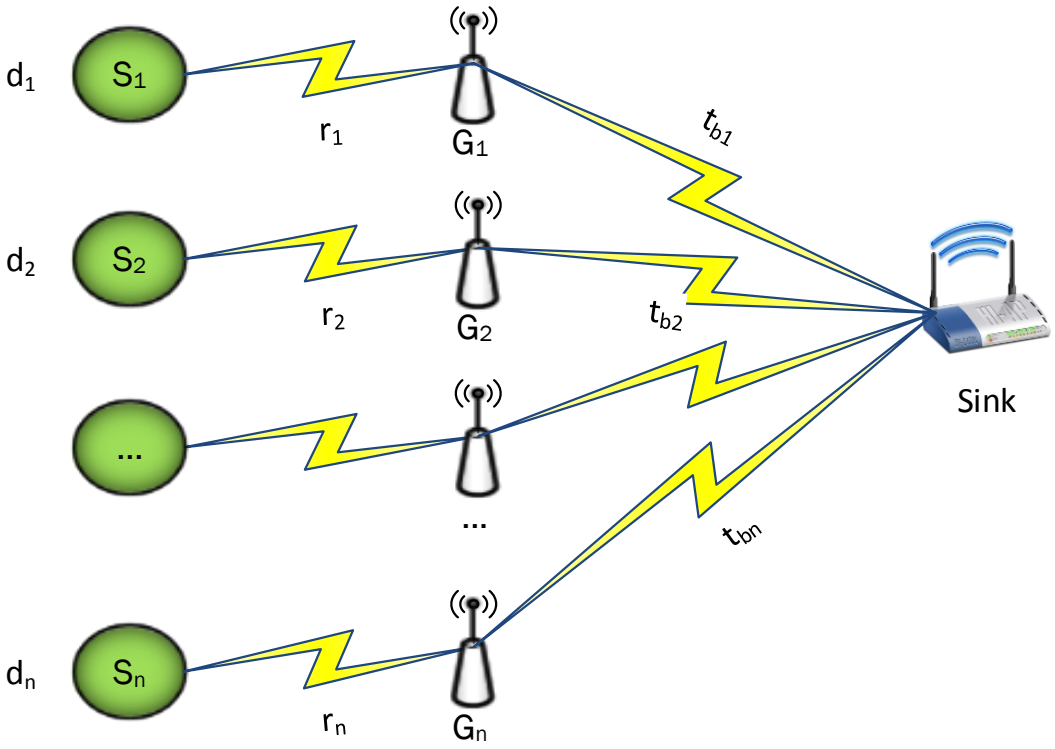


Figure 7. Typical scenario of IoT network where multiple source nodes are connected to the central sink node through intermediate gateway nodes.

Data aggregation at each gateway node depends upon the data receiving rate from the connected source nodes. Afterwards, data transmission time t_bi for a single packet by each gateway node depends on the aggregated data size b_i at respective gateway node. Brief descriptions of the various symbols used in the formulation are given in Table 1.

Data transmission time t_bi for a single packet by each gateway node can be computed as below

$$t_{bi} = RTS + CTS + T_x(b_i) + Ack \tag{1}$$

For total data d_i generated by source node S_i with data sending rate r_i, minimum transmission time $minTx_i$ is given by

$$minTx_i = \frac{d_i}{p_i \times r_i} \tag{2}$$

The actual total transmission time to send out the application data amount from the sensor node to sink node is the sum of the time spend in transmission, receiving, idle and sleep states [30]. However, Equation (2) provides a simplified model for minimum transmission time (theoretical) from sensor node to its next-hop gateway node assuming the node is not facing any interference or contention. For instance, a node has to send 1000 Bytes of data (including packet headers) to its gateway node and the data sending rate is 5 packets per second where packets size is 100 Bytes. Then, the minimum transmission time $minTx_i$ for this node becomes 2 s, computed using Equation (2).

Table 1. Description of the symbols used in the formulation.

Symbol	Description
S_i	ith source node
d_i	Total data generated by source S_i
r_i	Data generation rate by source S_i
p_i	Packet size at each source node
G_i	ith gateway node
b_i	Aggregated data size at G_i
t_{bi}	Data transmission time by gateway node G_i
$minTx_i$	Minimum transmission time for data generated by source node S_i
$maxTx_i$	Maximum transmission time for data generated by source node S_i
A_i	Total number of aggregated packets at gateway G_i
$\phi_{i,k}$	Scheduling interval for kth packet at gateway G_i
α	Starting time of an interval
β	Ending time of an interval
θ_{G_i}	Throughput of gateway G_i node

Total number of aggregated packets A_i at gateway G_i becomes

$$A_i = \frac{d_i}{b_i} \tag{3}$$

As all gateway nodes are directly connected with the sink node, at any given time instant, only one gateway node can forward the packets. Let $\phi_{i,k} = [\alpha, \beta]$ be time interval at which kth packet at gateway G_i node is scheduled where α and β are the start and end time of this particular interval. Given above formulation, maximum transmission time $maxTx_i$ for total data d_i generated by source node S_i can be computed as

$$maxTx_i = \phi_{i,A_i}(\beta) \tag{4}$$

whereas throughput θ_{G_i} of gateway G_i node becomes

$$\theta_{G_i} = \frac{d_i}{\phi_{i,A_i}(\beta) - \phi_{i,1}(\alpha)} \tag{5}$$

Overall network throughput θ_G can be formulated as below

$$\theta_G = \frac{\sum_{i=1}^{n} d_i}{max(\phi_{i,A_i}(\beta)) - min(\phi_{i,1}(\alpha))} \tag{6}$$

The goal is to maximise overall throughput while minimising the transmission time at individual gateway nodes. Thus, objective function can be formulated as below

$$\frac{\theta_G}{max(\phi_{i,A_i}(\beta) - \phi_{i,1}(\alpha))} \tag{7}$$

Constraints:

$$0 < minTx_i \leq maxTx_i \quad \text{for all } i = 1...n \tag{8}$$

$$\bigcap(\phi_{i,k}, \phi_{j,l}) = \emptyset \quad \text{for all } i = 1...n \text{ and } i \neq j \tag{9}$$

$$\|\phi_{i,k}\| = t_{bi} \quad \text{for all } i = 1...n \tag{10}$$

5. Experimental Testbed Setup

In detailing our experimental testbed, we first discuss our server-side setup of Runtime with cloud services. For our setup implementation, we used the state-of-the-art cloud services of Google Platform infrastructure. Google has introduced Firebase storage services for IoT data and which gained popularity as NoSQL cloud storage [31]. We evaluated our GraphQL-based approach in IoT applications through experiments using real IoT devices (Z1) and Firebase appears to be an effective cloud service storage for our designed testbed for experiments. For our experimentation of server-side module, we also used cloud functions for hosting. The Runtime environment was hosted via Google Cloud Functions and it connects the server-side Firebase service with the client-side GraphQL queries.

NoSQL-based databases used as cloud services and its types, which store data in an aggregated form offering single view data retrieval, are called aggregate-oriented NoSQL databases. The category of graph-based NoSQL databases provide multiple views of data. This ability comes from storing data as graph nodes and putting relationships as labels on the edges of the graph, but this category takes advantage of a standard language for data querying. Firebase is an aggregate-oriented NoSQL database for storage in the cloud environment, in which data are discrete, thus lacking support for relationships, in comparison to graph databases. Therefore, our implementation needs a parser for using Firebase with GraphQL-based implementation to store high connected data in discrete data storage. Thus, we used cloud functions of Google to host the required parser, which translates GraphQL queries to the underlying storage. The specification's Runtime implements this parser for storing highly connected data, received from IoT nodes, in the discrete data cloud storage of Firebase.

As far as choosing GraphQL query language on client side IoT devices for our implementation is concerned, GraphQL has introduced a new way of clients querying an API. There are also some other query languages available for graph databases, but they mostly have dependencies on other software. Therefore, their acceptance as standard in the domain could not be possible. The GraphQL query language has an inherent design of API interaction. As Firebase database can also allow interaction through APIs, we chose it in our implementation with API facing GraphQL.

The experimental setup/testbed used in this research comprised of real IoT devices Z1 manufactured by Zolertia, having low power MSP430F2617 micro-controller, with 16 MHz RISC CPU, 8 KB RAM and 92 KB internal memory. Deployment was done in real indoor environment of university environment with walls, people's mobility, WiFi interference, etc. Moreover, connectivity and traffic scenarios were changed in many ways to check different possible real world situations. Figure 8 shows a snapshot of a typical scenario of our working setup.

For our IoT devices, we used Contiki OS with its default implementation and used IEEE 802.15.4 standard protocol on lower layers of the stack. Data transmission was limited to 250 kbps and 2.4 GHz radio band was used as a medium for channels. We used RPL of the Contiki stack as the routing layer protocol and radio duty cycling was set to off, to make a case study of continuous sensing IoT scenarios. At the MAC layer, we enabled re-transmissions for one-hop packet reliability and the border router using RPL DODAG worked as a gateway for connection with the cloud services. The messages used for sensor values were of POST nature used to deliver data to destination end-point repositories with typical size of 96 bytes under Contiki OS layered stack. Energy values were measured using Powertrace [32], a network level power profiling model for low power wireless networks. It uses a linear power model to estimate the power as the sum of all active power states and energy of the system is derived from the time spent in each power state. For example, CPU in sleep and active modes are separate power states and radio transceiver in transmission or listening modes adds separate power usage to the energy model. Table 2 shows the configuration of IoT devices and the server system used in this set of experiments.

Figure 8. Indoor experimental setup with real Z1 motes to evaluate the proposed approach for IoT sensing applications.

The proposed approach uses communication optimisation through traffic minimisation and thus helps in relieving the bottleneck issue possible due to hierarchical setup. This problem can be better explained through an example scenario, as presented in Figure 2. The nodes near sink are critical and solutions such as those proposed in this paper are needed to prolong its lifetime by avoiding unnecessary forwarding as otherwise the network will soon be disconnected.

Figure 9 shows the experimental topology scenarios used in our study for performance evaluation of the proposed approach. In Scenario 1, we have a single IoT device (real Z1 mote) with multiple sensors attached to it, and its data are posted to separate resources on the cloud server. In Scenario 2, we increase the number of IoT device (Z1 motes) to study the impact of increased traffic/contention around the gateway node. In Scenario 3, some of the IoT devices are not in direct communication range of the gateway node and this scenario is to study the impact of multi-hop communication on the performance comparison of the proposed approach against baseline REST schemes.

In these experiments, a CoAP server residing at cloud maintained three data resources/services and an IoT device must submit data to one or more of these repositories. In cloud-based IoT applications, these data resources can be assumed as a server providing different services to the clients, who subscribed to these resources, with each of the resources getting data from separate sensors of the field IoT devices. In these experiments, real IoT devices Z1 are used as the senders and as the border router attached to the gateway. These IoT motes use client–server mechanism of IoT CoAP protocol and its Er-rest implementation in Contiki OS to achieve data submission to the resources at the cloud system.

In our experiments, we considered scenarios of putting more traffic injection into the system using typical sensors available on our Z1 devices with Contiki OS layered stack sending adequate message sizes. We want show high traffic scenarios with multiple sensors and data destinations using multiple motes. We addressed the issue of high data traffic and its management in general, but mainly non-scalar sensor data can be the reason of network congestion and a good candidate case of the proposed approach.

Table 2. Configuration details of Server and IoT devices used in experimental setup.

Component	IoT Device	Server
Hardware	Zolertia Z1 MCU: MSP430F2617	Google Cloud Services
	Radio: CC2420	
	RAM: 8 kB	
	ROM: 92 kB	
Operating System	Contiki OS	Ubuntu
Development Tool	Cooja, Eclipse	Python
Libraries and Frameworks	GraphQL client, Query Language	Firebase, GraphQL Runtime

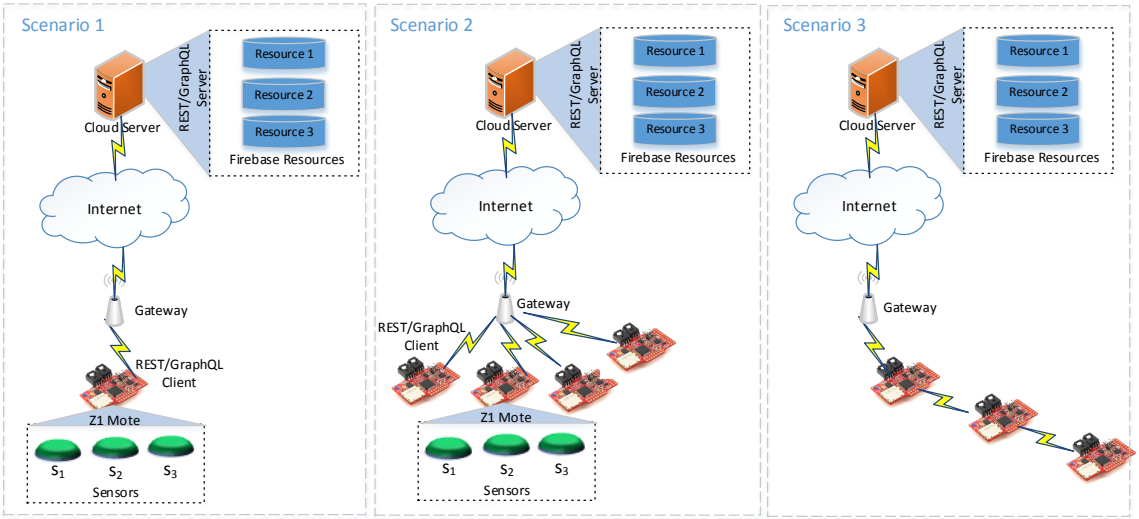


Figure 9. Experimental scenarios for performance analysis of the proposed scheme.

6. Results and Discussion

Our proposed approach is helpful in prolonging operational lifetime of IoT devices and overall network by achieving energy efficiency through communication optimisation. The proposed approach is effective in two cases: (a) current IoT networks with sparse deployments; and (b) future IoT networks with expected dense deployments. Experimental results presented in this paper were collected in a testbed environment with 4 IoT devices contained in a single site and significant (51–52%) improvement in achieving energy efficiency was observed. These results gives us confidence to further explore and evaluate proposed idea in a bigger experimental setup with a higher number of IoT devices.

In this section, we discuss the results obtained from all three experimental scenarios. Comparisons include three approaches: (i) rest-2R needing two round trip (REST-based approach subscribed to two data resources); (ii) REST-based sending to three resources (rest-3R needing three round trips sending each for separate resource endpoint); and (iii) our proposed approach based on abstraction layer to send data to three cloud-based data resources.

6.1. Scenario-1

In this scenario, we have single IoT device (Z1 mote) with multiple sensors attached to it, and the data are posted to separate resources on the cloud server. Here, we show delay and energy consumption results on increasing transaction loads (in form of data sending interval) in Figure 10. The results were collected with varying packet sending interval from 2 to 0.5 s, with a step size of −0.5. With an increase in data sending rate, we observe increase in the delay values. REST-based scheme with three resources subscribed suffers greatly and around 100 ms delay increase is observed when the data sending interval

is reduced from 2 to 0.5 s. However, the proposed scheme results in the lowest delay for all the cases. Furthermore, we observe the increase in the delay for it is marginal with increase in data sending rate, which shows that the proposed approach is more robust and resilient, and hence more suitable for IoT sensing applications.

Energy consumption results also exhibit a similar pattern. A significant increase in energy consumption is observed for the REST-based schemes, on all the data sending intervals and further increases when the data sending rate is increasing. The proposed approach outperforms the baseline REST-based schemes and achieves delivery of the same data using less than half energy cost. Packet Delivery Ratio (PDR) and throughput results for this scenario are not meaningful (hence, not reported), as there is no notable packet loss or throughput changes for this one device-based evaluation scenario. Spread of the values has been shown by error bars for both the metrics and we can see better performance confidence for proposed GraphQL-based approach.

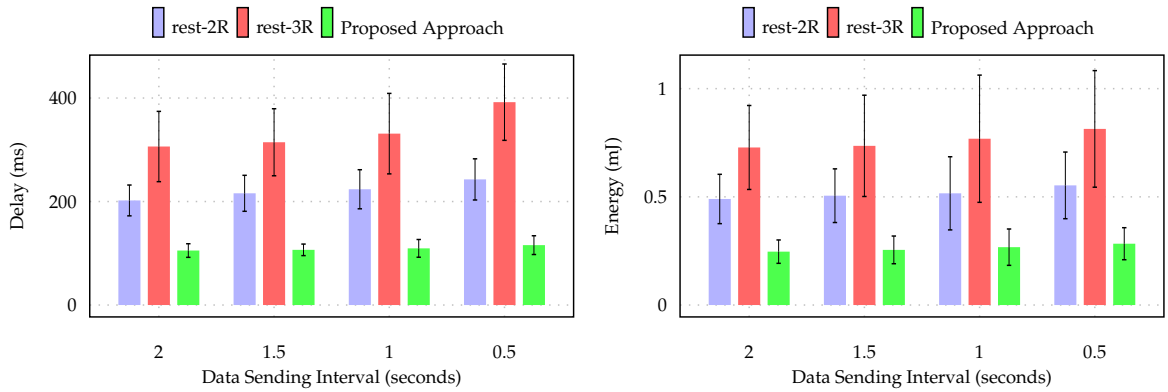


Figure 10. Scenario 1: Average delay and energy consumption comparison of proposed approach against REST based communication to two and three data resources.

6.2. Scenario-2

This scenario has device configurations as in Scenario 1, but we varied the number of IoT client devices (number of Z1 motes ranging 1–4) in a dumbbell-shaped network topology, having all the devices/senders in direct one-hop connectivity to the border router/cloud gateway. Here, offered/data load increases in the form of multiple active senders ranging from one to four devices. Comparisons was done among: (i) rest-2R (nodes registering data to two resources with separate endpoints); (ii) rest-3R (sender nodes submitting data to three separate resources at the cloud with CoAP server); and (iii) using proposed framework implementation for sending data to three cloud resources. Here, we also show results for PDR and system throughput (in terms of number of successful data transaction) in Figure 11, because all the sender devices are in the same contention area to the border node in the used topology. For both values, the proposed scheme shows better performance in comparison to REST architecture.

With the increase in offered data load, increase in the delay is minimal and values are lowest for the proposed scheme in comparison in all the cases. Both the REST architecture schemes with two and three resources subscription have started to suffer when three active devices are communicating simultaneously. Similarly, the least used and the marginal increase in energy consumption is observed in the proposed scheme with increase in offered load, but the REST-based approach are behind and their energy consumption increase is linearly going up when multiple devices submit their data to the cloud data repositories. These schemes also suffer in terms of packet losses on high load and rest-3R scheme shows significant wastage of energy in terms of data loss on load of four active devices. The throughput result also shows some achievement in the case of the proposed framework.

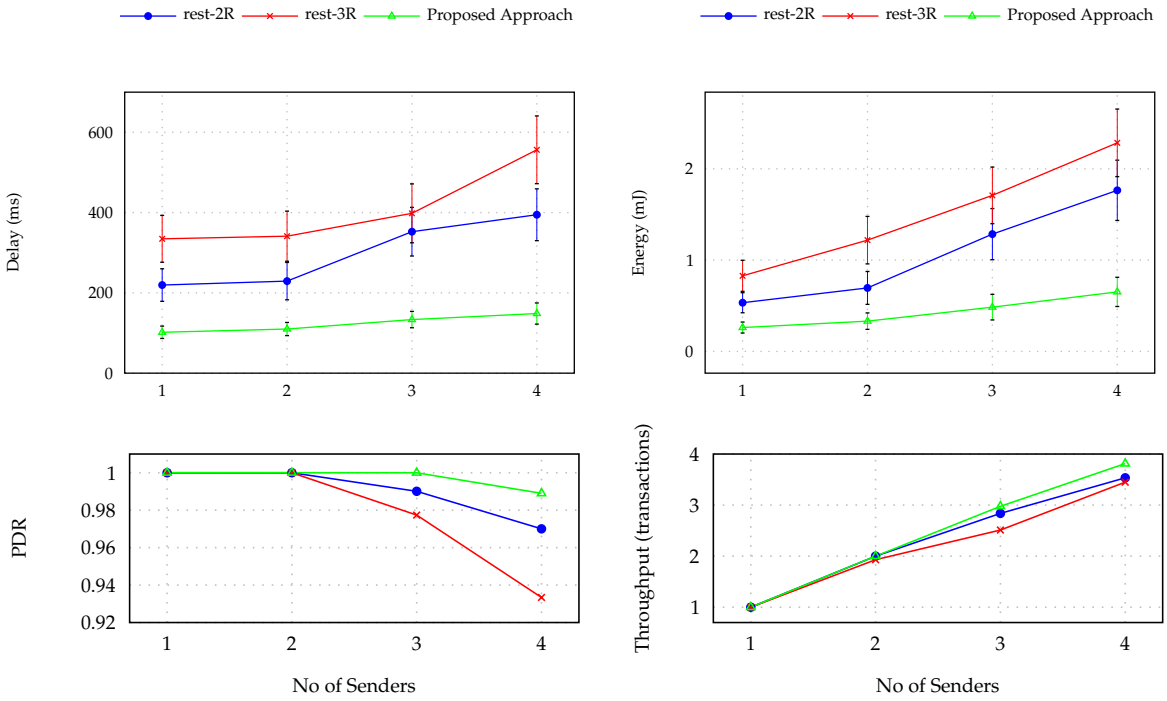


Figure 11. Scenario 2: Delay, energy consumption, packet delivery ratio (PDR), and throughput on different Loads.

The proposed approach outperforms state of the art REST-based schemes in this scenario too and achieves energy-efficient, in-time delivery with a minimum loss of data packets. In the graphs of delay and energy, we can see the the proposed approach also shows better spread of the values with confidence, in comparison to other two schemes.

6.3. Scenario-3

In this scenario, we tested the performance of our proposed system implementation for hop-by-hop communication in IoT sensing application. We used four Z1 nodes in these experiments, with one mote acting as border router/gateway and three motes as set of hop-by-hop sending devices. We collected results for four metrics: delay, energy, PDR, and throughput. Our results, as shown in Figure 12, clearly establish that proposed scheme is much better in energy consumption as well as all the other three metrics. In hop-by-hop communication with increasing distance in terms of hops, energy consumption and delay increase for the proposed approach is normal. However, REST-based approaches, especially 3R-rest communication, face disastrous output in the maximum three hop case. Similarly, packet loss for rest-2R and rest-3R are higher when the number of hops is increased but the proposed architecture maintains its PDR as well as throughput results. It significantly outperforms the REST schemes on all the metrics.

6.4. Results Summary

In these experiments, we observed a 46.27% increase in delay using REST architecture, when the number of subscribed resources (end-points) is increased from two to three. On three subscribed resources (rest-3R), REST results in an average delay of 481.48 ms, whereas the proposed approach could achieve an average delay of 159.54 ms, i.e., a significant 66.86% reduction in delay. Our experimental results reveal that the average amount of energy consumption increases from 1.04 to 1.53 mJ when the number of subscribed resources (end-points) is increased from two to three using REST-based architecture. However, average energy consumption for the proposed scheme was recorded as 67.97 mJ, i.e., 52.88% and 67.97% reduction as compared to the baseline REST architecture with two and three end-points, respectively.

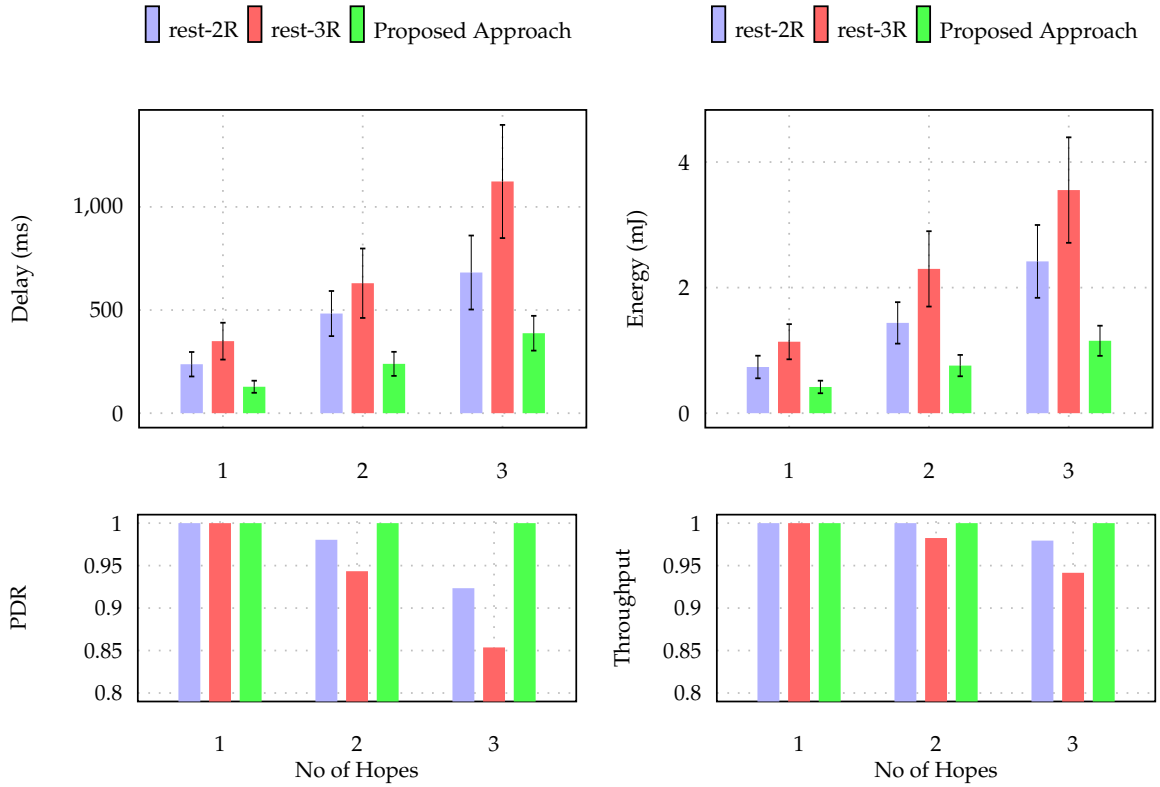


Figure 12. Scenario 3: Average delay, energy consumption, PDR, and throughput comparisons with increasing number of sender hopes.

Comparative analysis of packet delivery ratio (PDR) and throughput results show that all schemes achieve satisfactory results. There was no significant impact on PDR using REST architecture with the increase in number of end-points from two to three and this is because fewer nodes (Z1 motes) were used in experimental scenarios. There was very little room for improvement in performance for the proposed approach and only 4.16% and 7.5% improvement was observed in PDR and throughput, respectively.

The proposed approach achieves this performance because it can deliver the same data with a reduced number of round trips, thus saving a significant amount of energy that is highly desirable in IoT networks. This concludes our discussion on experimental results conducted in a controlled testbed environment with a limited number of IoT devices. These results give us the confidence to test the proposed approach in a more congested and challenging setup involving higher number of nodes to further validate its performance.

7. Energy-Wave Problem and High Power Communication

The energy-wave problem can be raised in dense IoT networks where data are communicated to gateway nodes in multi-hop fashion due to communication range limitations of constrained devices. We focus on IoT networks with high data rate above, either due to multimedia sensors or dense deployment with frequent reporting scenarios. Such networks using multi-hop data collection, e.g., in Scenario 3 of our experimentation, face the energy-wave problem. Our proposed approach can help in overcoming the energy-wave problem through traffic management in constrained IoT networks with high data sending rate. We considered the energy-wave problem and tested an added proposal for overcoming it. Our approach was using direct high power communication instead of low power multi-hop data sending. We obtained supporting results for this test case by changing the communication range of involved devices.

IoT devices are constrained in energy supply and communication range. Multi-hop communication is used for long range IoT applications connectivity. Channel contention is increased when the data flow towards the neighbourhood of the gateway node and hierarchical connection may face bottleneck

issues. This phenomenon is also commonly referred as the energy-wave problem, as shown in Figure 2. We propose high power single hop connectivity as efficient alternative to multi-hop data delivery and resolution of energy-wave problem. We discuss supporting results obtained on real experiments based on our implementation of GraphQL-based abstraction layer using our IoT testbed in outdoor environment.

7.1. Hierarchical Connectivity Bottleneck

The Internet of Things (IoT) revolution promises to make "things" including electronic, industry, health devices and home appliances part of the Internet world. Most of the devices in Internet of Things paradigm have to be constrained in energy supply powered by batteries with wireless connectivity. Therefore, energy efficiency is a driving factor regarding design of IoT applications' architecture. Communication over wireless channels is responsible for the large portions of energy consumption of the IoT devices. Therefore, IEEE 802.15.4 wireless communication standard was designed to extend the lifetime of IoT power sources by using low power transmissions. It provides low-rate, short range connectivity between communicating devices. Hop-by-hop communication-based mesh architecture is suggested in long range needing IoT applications [33]. In multi-hop topology, end-to-end communication between a data sender and destination is achieved by using intermediate nodes as forwarders in routing path. This architecture provides the connectivity between ends through multi-hop paths by using low power low range transmissions in hope of energy saving but communications' quality of service is deteriorated [34].

We tested the assumption of communications suffering in hop-by-hop communication for long range data transmissions. We suggest direct one-hop high power transmissions for achieving long range connectivity for better quality of service. We tested our case on real IoT research motes on a basic set of experiments of increasing data rates and found supporting results to our position. For high power direct connectivity, we obtained up to 15 times less average delays and more than four times higher response receiving rate at high data sending rates. Even energy consumption is better conserved in high power case due to in-time high rate data delivery, causing less duty cycling for IoT devices. We obtained the best result of energy saving at one request per second rate and it is six times lower than multi-hop data delivery. Single hop communication causes much less duty cycling for the sender in high power communication case, which is also one of the main reasons for low energy consumption.

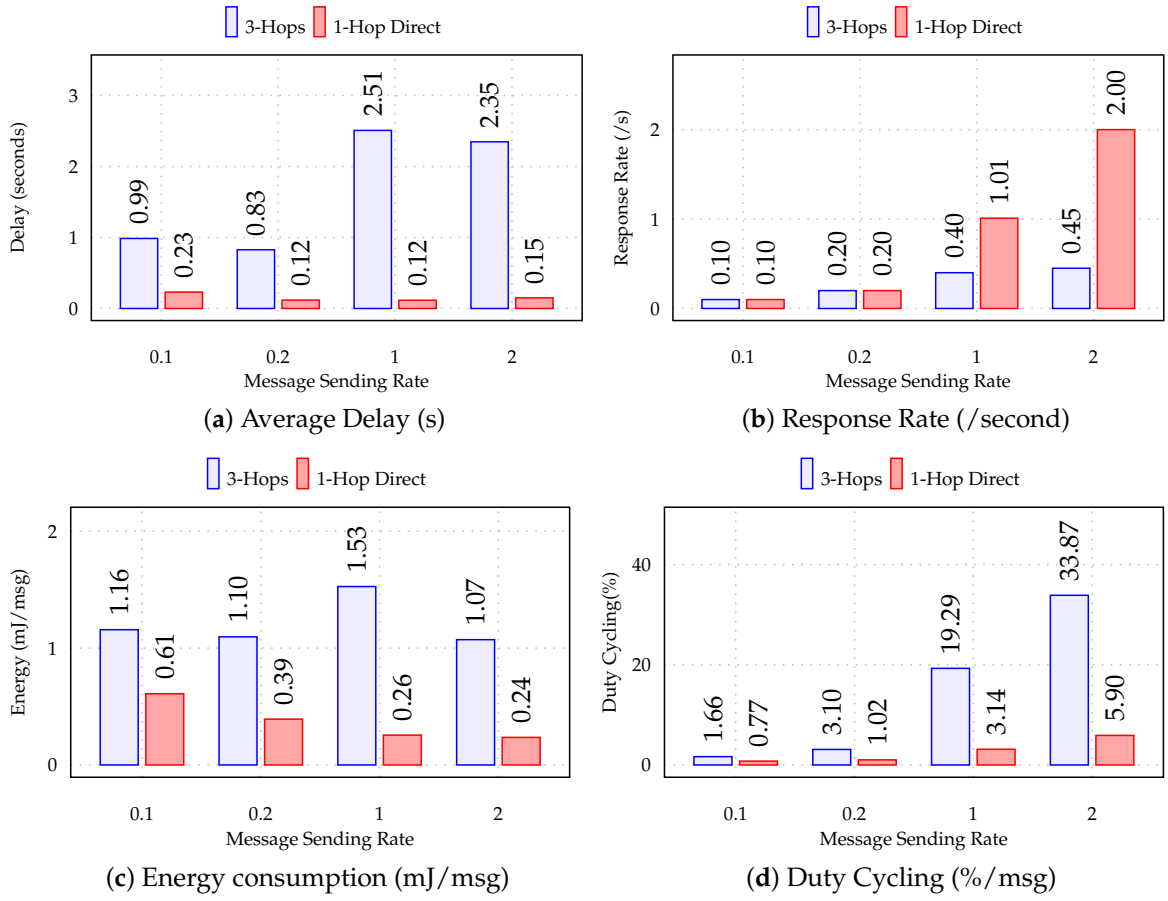
7.2. Communication Setup and Test Findings

We designed experiment scenarios to test our assumption and suggested high power communications. We used four Zolertia Z1 IoT motes with a custom small external antennas to enable short hop ranges in lab environment, running on Contiki OS for IoT. IETF standard ReST-based CoAP protocol is often used in modern IoT applications, thus we used Contiki implementation [35] of this request–response protocol for our IoT communications. In all the trials, there was a sender node providing in-field sensing service in response to periodic requests by a client installed on a server laptop, sent via Z1 mote acting as IoT gateway node. In each experiment trial, the client made 100 requests for sensors' data and variation in frequency of requests massages was done to check effect of different data load on the IoT system.

We used two topologies for connectivity experiments, in which sender and receiver gateway were placed 12 m away in an indoor/home environment. In first scenario, low transmission power equal to -25 dbm (0.0032 mW) was set to restrict direct connectivity. Connectivity for data communication was provided by placing two z1 motes in between at equal distances with same transmission powers (-25 dbm) to act as intermediate forwarders. We made sure that each mote had connectivity range only to next forwarder and communication was really happening via multi-hop path. In the other scenario, we switched off intermediate motes and increased the transmission powers to -15 dbm (0.0316 mW), which was sufficient to provide direct connectivity between in-filed sensing server and gateway node. The intervals between periodic request messages were varied as 10, 5, 1 and 0.5 s providing message sending rates 0.1, 0.2, 1 and 2 per second, respectively. We measured average round trip delays, average energy consumption by sender per request–response message exchange, average

response receiving rate achieved at receiver gateway per second and duty cycling to be done by sender node per request–response exchange.

All the results comparisons in Figure 13 are on the same set of experiments and variations. The only difference is of transmission powers to provide connectivity between sender and receiver. We consider the results in Figure 13 one by one.



(**a**) Average Delay (s) (**b**) Response Rate (/second)

(**c**) Energy consumption (mJ/msg) (**d**) Duty Cycling (%/msg)

Figure 13. Results comparisons with respect to increasing message sending rates.

Delay: Average delay for direct connectivity case Figure 13a is much less on all the request sending rates. The average for all four sending rate experiments is 0.62 s for direct connectivity case, whereas it is 8.5 s for hop-by-hop case. The high power case is five times better at low sending rates and 15 times less on high data rates in comparison to multi-hop communication. Its values are not much affected by more load in shape of high message rates on the system, whereas hop-by-hop architecture suffers by the load and delays are increased on high data rates.

Response Receive Rate: The response receiving rate on increasing request rates in Figure 13b are very encouraging and, on high data rates, direct connectivity scenario delivers all the data in time. The hop-by-hop case suffers severely at high load rates and is unable to deliver the data even in reasonable time. On two requests per second rate, direct transmission delivers at two packets per second, whereas hop-by-hop communication is only able to deliver at 0.45 packets per second. It is 4 times less than request sending rate as well as delivery rate of multi-hop case. On high data sending rates of 1 and 2, to complete the total 100 request/response exchanges, direct connectivity case needed 100 and 50 s, whereas hop-by-hop communication took 253 and 244 s, respectively. This is one of the major reasons for high energy usage for three-hop case because it has to run for 2.5–5 times longer, utilising the extra power.

Energy Consumption: Now, if look at the energy consumption of both the cases Figure 13c, it is more interesting to observe these results. Direct connectivity case even outperforms hop-by-hop communication in energy consumption on all data rates despite using much high transmission power.

The greater is the sending rate of requests, the greater is the difference and benefit per message exchange because, at high rates, data flow as a stream and control packets are fewer. At one request per second, direct connectivity consumes six times lower energy per message than multi-hop communication.

Duty Cycling Faced: Average duty cycling faced per message exchange Figure 13d shows the second main reasons for less energy usage. In hop-by-hop case, sender has to perform more duty cycling on all the sending rates. The highest case is at message rate 2 where it goes to 33.87% in comparison 5.9% by direct communicating sender.

Other reasons for high energy consumption and duty cycling in multi-hop case are packet losses, more control packets and channel contention problems due to more no of communicating devices. Packet losses and resultant retransmissions are obvious from the fact that calculated packet delivery ratio (PDR) dropped up to 0.8 on high data rates, which means 20% extra data communication and hence extra energy usage, duty cycling and total time for data delivery.

On all the metrics, the direct connectivity high power case outperforms the hop-by-hop connectivity despite using about 10 times lower transmission power by sender in multi-hop communication. By analysing these preliminary results, high power transmissions-based direct one hop communication appears to be a promising option for IoT connectivity extending. Application quality of service is expected to improve in comparison to multi-hop communication and even energy consumption is reduced due to in-time, loss free delivery and reduced duty cycling. We have further verification plans for extensive long range outdoor experiments about 100 m distance and testing the case with multiple IoT application protocols.

8. Conclusions

The realisation of large scale IoT sensing applications faces many challenges due to the constrained nature of involved devices and traditional communication protocols. GraphQL is an application layer framework introduced by Facebook to resolve the efficiency challenges of communication in web-based linked data. We leveraged its appropriate features as the abstraction layer and implemented that layer for sensor network-based IoT application scenarios. We studied the performance of our incorporation on real IoT motes (Z1) having Contiki OS, through experiments involving different scenarios, which depict continuous sending IoT sensing applications. Through our empirical studies of the proposed approach, we got the strengths of GraphQL-based abstraction layer validated for our target cases, by achieving a significant improvement in the usage of energy (52–68% reduction) and transaction delays (51–67% reduction). We also considered energy-wave problem faced in hierarchical structure adopted for multi-hop IoT communication and tested a proposal of high power direct communication using our testbed to relive this issue. We obtained improved performance in this case too for our experimental setup. Therefore, we suggest utilisation of modern web software through efficient integration in modern IoT applications. We aim to further strengthen our proposal through more use cases in large IoT networks via different applications of such modern technology. Our implementation and experimental study will help the research community and industry to make knowledgeable decisions and do further exploration in utilising web-based technologies for emerging IoT applications involving large scale implementation.

Author Contributions: R.K. and A.N.M. discussed and confirmed the idea. R.K. carried out the experiments, analysis and wrote the main paper. R.K. and A.N.M. did reviews and editing. A.N.M. was supervising the work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Boccardi, F.; Heath, R.W.; Lozano, A.; Marzetta, T.L.; Popovski, P. Five disruptive technology directions for 5G. *IEEE Commun. Mag.* **2014**, *52*, 74–80. [CrossRef]
2. CISCO White Paper. Internet of Things at a Glance. Technical Report. Available online: https://www.cisco.com/c/dam/en/us/products/collateral/se/Internet-of-things/at-a-glance-c45-731471.pdf (accessed on 14 September 2019).

3. Ahmad, I.; Kumar, T.; Liyanage, M.; Ylianttila, M.; Koskela, T.; Bräysy, T.; Anttonen, A.; Pentikäinen, V.; Soininen, J.P.; Huusko, J. Towards gadget-free Internet services: A roadmap of the Naked world. *Telemat. Inform.* **2018**, *35*, 82–92. [CrossRef]
4. Ejaz, W.; Naeem, M.; Shahid, A.; Anpalagan, A.; Jo, M. Efficient energy management for the Internet of things in smart cities. *IEEE Commun. Mag.* **2017**, *55*, 84–91. [CrossRef]
5. CISCO White Paper. Global Mobile Data Traffic Forecast Update, 2017–2022. Technical Report. Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html/ (accessed on 14 September 2019).
6. CISCO White Paper. Cisco Visual Networking Index: Forecast and Methodology, 2016–2021. Technical Report. Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html (accessed on 14 September 2019).
7. Harjula, E.; Mekonnen, T.; Komu, M.; Porambage, P.; Kauppinen, T.; Kjällman, J.; Ylianttila, M. Energy Efficiency in Wireless Multimedia Sensor Networking: Architecture, Management and Security. In *Greening Video Distribution Networks*; Springer: Cham, Switzerland, 2018; pp. 133–157.
8. Li, S.; Kim, J.G.; Han, D.H.; Lee, K.S. A survey of energy-efficient communication protocols with QoS guarantees in wireless multimedia sensor networks. *Sensors* **2019**, *19*, 199. [CrossRef] [PubMed]
9. Shallari, I.; O'Nils, M. From the Sensor to the Cloud: Intelligence Partitioning for Smart Camera Applications. *Sensors* **2019**, *19*, 5162. [CrossRef] [PubMed]
10. Choudhury, N.; Matam, R.; Mukherjee, M.; Shu, L. Beacon synchronization and duty-cycling in IEEE 802.15. 4 cluster-tree networks: A review. *IEEE Internet Things J.* **2018**, *5*, 1765–1788. [CrossRef]
11. GraphQL, Oct. 2016. Available online: https://graphql.github.io/graphql-spec/ (accessed on 21 October 2019).
12. Husain, J. Keynote: JSON Graph: Reactive REST at Netflix. 2015. Available online: https://dl.acm.org/doi/10.1145/2742580.2742640 (accessed on 18 October 2019).
13. Hartig, O.; Pérez, J. An Initial Analysis of Facebook's GraphQL Language. In Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web (AMW 2017), Montevideo, Uruguay, 7–9 June 2017.
14. Hartig, O.; Pérez, J. Semantics and complexity of GraphQL. In Proceedings of the 2018 World Wide Web Conference. International World Wide Web Conferences Steering Committee, Lyon, France, 23–27 April 2018; pp. 1155–1164.
15. Vázquez-Ingelmo, A.; Cruz-Benito, J.; García-Pe nalvo, F.J. Improving the OEEU's data-driven technological ecosystem's interoperability with GraphQL. In Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality, Cádiz, Spain, 18–20 October 2017; p. 89.
16. Taelman, R.; Vander Sande, M.; Verborgh, R. GraphQL-LD: Linked Data Querying with GraphQL. In Proceedings of the International Semantic Web Conference (P&D/Industry/BlueSky), Monterey, CA, USA, 8–12 October 2018.
17. Sabharwal, M.; Agrawal, A.; Metri, G. Enabling Green IT through Energy-Aware Software. *IT Prof.* **2013**, *15*, 19–27. [CrossRef]
18. Han, K.; Luo, J.; Liu, Y.; Vasilakos, A.V. Algorithm design for data communications in duty-cycled wireless sensor networks: A survey. *IEEE Commun. Mag.* **2013**, *51*, 107–113. [CrossRef]
19. Piyare, R.; Murphy, A.L.; Kiraly, C.; Tosato, P.; Brunelli, D. Ultra low power wake-up radios: A hardware and networking survey. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2117–2157. [CrossRef]
20. Sriram, R.D.; Sheth, A. Internet of Things Perspectives. *IT Prof.* **2015**, *17*, 60–63. [CrossRef]
21. Lerche, C.; Laum, N.; Golatowski, F.; Timmermann, D.; Niedermeier, C. Connecting the web with the web of things: lessons learned from implementing a CoAP-HTTP proxy. In Proceedings of the 2012 IEEE 9th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2012), Las Vegas, NV, USA, 8–11 October 2012; pp. 1–8. [CrossRef]
22. Castellani, A.; Loreto, S.; Rahman, A.; Fossati, T.; Dijk, E. Best Practices for HTTP-CoAP Mapping Implementation. Available online: https://tools.ietf.org/id/draft-castellani-core-advanced-http-mapping-01.html (accessed on 10 November 2019).
23. Leggieri, M.; Hausenblas, M. Interoperability of two RESTful protocols: HTTP and CoAP. In *REST: Advanced Research Topics and Practical Applications*; Springer: New York, NY, USA, 2014; pp. 27–49.
24. Nogatz, F.; Seipel, D. Implementing GraphQL as a query language for deductive databases in SWI-prolog using DCGs, quasi quotations, and dicts. *arXiv* **2017**, arXiv:1701.00626.

25. Rasool, S.; Saleem, A.; Mian, A.N. Poster: RQL: REST Query Language for Converting Firebase to a Mobile Cloud Computing Platform. In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom'17, Snowbird, UT, USA, 16–20 October 2017; ACM: New York, NY, USA, 2017; pp. 567–569. [CrossRef]
26. Rasool, S.; Khan, R.; Mian, A.N. GraphQL and DC-WSN-Based Cloud of Things. *IT Prof.* **2019**, *21*, 59–66. [CrossRef]
27. Nechibvute, A.; Chawanda, A.; Luhanga, P. Piezoelectric energy harvesting devices: an alternative energy source for wireless sensors. *Smart Mater. Res.* **2012**, *2012*. [CrossRef]
28. Vaquero, L.M.; Rodero-Merino, L. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 27–32. [CrossRef]
29. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
30. Krug, S.; O'Nils, M. Modeling and comparison of delay and energy cost of iot data transfers. *IEEE Access* **2019**, *7*, 58654–58675. [CrossRef]
31. Overview of Internet of Things. 2017. Available online: https://cloud.Google.com/solutions/iot-overview#storing (accessed on 2 January 2020).
32. Dunkels, A.; Eriksson, J.; Finne, N.; Tsiftes, N. *Powertrace: Network-Level Power Profiling for Low-Power Wireless Networks*; Swedish Institute of Computer Science, Kista, Sweden, 2011.
33. Shang, W.; Yu, Y.; Droms, R.; Zhang, L. Challenges in IoT Networking via TCP/IP Architecture; Technical Report, NDN Project, Tech. Rep. NDN-0038. Available online: https://named-data.net/wp-content/uploads/2016/02/ndn-0038-1-challenges-iot.pdf (accessed on 29 December 2019).
34. Andreev, S.; Galinina, O.; Pyattaev, A.; Gerasimenko, M.; Tirronen, T.; Torsner, J.; Sachs, J.; Dohler, M.; Koucheryavy, Y. Understanding the IoT connectivity landscape: A contemporary M2M radio technology roadmap. *IEEE Commun. Mag.* **2015**, *53*, 32–40. [CrossRef]
35. Kovatsch, M.; Duquennoy, S.; Dunkels, A. A low-power CoAP for Contiki. In Proceedings of the 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, Valencia, Spain, 17–22 October 2011; pp. 855–860.