Digital Object Identifier 10.1109/ACCESS.2021.3099942

edgeRouting: Using Compute Nodes in Proximity to Route IoT Data

Vasileios Karagiannis[1], **Stefan Schulte**[2]
[1]Distributed Systems Group, TU Wien, Vienna, Austria (e-mail: v.karagiannis@dsg.tuwien.ac.at)
[2]Christian Doppler Laboratory Blockchain Technologies for the Internet of Things, TU Wien, Vienna, Austria (e-mail: s.schulte@dsg.tuwien.ac.at)

Corresponding author: Vasileios Karagiannis (e-mail: v.karagiannis@dsg.tuwien.ac.at).

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation. The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

ABSTRACT Due to the proliferation of edge computing, cloud providers have started offering compute nodes at the edge of the network in addition to traditional compute nodes in data centers. So far, various systems have been proposed for processing Internet of Things (IoT) data on both edge and cloud compute nodes in order to reduce the communication latency. However, such systems do not typically consider that the network bandwidth between an edge node and a cloud node can be orders of magnitude higher than the bandwidth between an IoT device and a cloud node. As a result, the IoT data are commonly sent selectively to either edge or cloud nodes disregarding alternative network paths through edge nodes, which may have higher network bandwidth, and lower communication latency.
To avoid this, in this paper we analyze the latency of sending data to edge and cloud compute nodes of cloud providers. Based on this analysis, we propose edgeRouting which routes the data through the closest edge compute node. By doing that, edgeRouting exploits both the low propagation delay of nodes at the edge, and the high bandwidth among edge and cloud compute nodes of cloud providers. To evaluate our approach, we perform experiments on a real-world setup with nearby and remote compute nodes of a cloud provider, and we show that edgeRouting reduces the communication latency by up to 55% compared to alternative methods.

INDEX TERMS Communication latency, Edge computing, Fog computing, Internet of Things, Propagation delay, Queuing delay, Transmission delay

I. INTRODUCTION

Due to the advent of the Internet of Things (IoT), a huge amount of data generated by sensor devices are sent to the cloud for processing [1]. Since remote cloud compute nodes alone may not be able to process these data in a timely manner, edge computing has been proposed for reducing the communication latency [2]. To achieve this, edge computing relies on the utilization of distributed compute nodes which aim at processing the IoT data in the proximity of the IoT devices [3].

Driven by the potential market demand for edge computing, cloud providers extend their network of data centers with additional sites which provide computational resources at the edge of the network. For instance, Google utilizes *Edge Points of Presence* in various regions around the world, and Microsoft uses *Edge Zones* in population centers which are far away from cloud infrastructure [4], [5]. To exploit such resources in an efficient manner, various approaches have been proposed for sending data to compute nodes in proximity [2]. Typically, this includes a placement algorithm which considers network-related metrics such as end-to-end propagation delay and bandwidth, to decide which one of the available compute nodes at the edge and the cloud should process the data [6], [7]. Based on the decisions of this algorithm, the data are sent to a suitable compute node for processing.

Even though such approaches may reduce the communication latency compared to using only remote cloud nodes [8], considering end-to-end metrics might hide that different parts of the underlying network may have very different capacities. For instance, when sending data from an IoT device to an edge or cloud compute node of a cloud provider, the bandwidth of the transmission is limited by the Internet provider (e.g., AT&T or Telefonica), and is commonly a few Megabits per second (Mbps) [9], [10]. However, when sending data between compute nodes, cloud providers do not impose such low limits, thereby allowing transmissions with bandwidth that can be orders of magnitude higher [11], [12]. Nevertheless, this bandwidth is not utilized when the data are sent directly from an IoT device to the compute node that performs the processing. Therefore, considering end-to-end metrics from the IoT devices to the available compute nodes may hide alternative network paths which can potentially reduce the communication latency even more.

To counter this problem, we analyze the different delays that contribute to the latency of sending data from IoT devices to edge and cloud compute nodes offered by cloud providers. Based on this analysis, we propose *edgeRouting* which is a routing approach for reducing the communication latency. According to edgeRouting, when sending data from an IoT device to a suitable compute node of a cloud provider, these data are routed through the closest available edge compute node. This way, the data are sent to the edge node using the connection of an Internet provider (i.e., with potentially limited bandwidth), and from there, the data are routed again to the suitable compute node with higher bandwidth offered by a cloud provider. Thus, this approach manages to leverage both the low propagation delay of edge compute nodes, and the high bandwidth among edge and cloud compute nodes of cloud providers, in order to achieve low latency.

Our contributions include the analysis of the latency of sending IoT data to distributed compute nodes at the edge and the cloud, and the design of a routing approach which sends the data in a manner that reduces this latency. Furthermore, we implement edgeRouting along with alternative routing approaches, and we conduct a comparative evaluation considering IoT use cases. According to our results, the proposed edgeRouting reduces the communication latency by up to 55%.

The rest of this paper abides by the following structure: Section II provides a discussion of related work, and Section III describes our system model. Afterwards in Section IV, we analyze the latency of sending IoT data to distributed compute nodes, and we propose an approach for reducing this latency. Subsequently in Section V, we perform experiments in order to compare the proposed approach to alternative methods, and we present our findings. Finally, Section VI concludes this paper, and proposes promising directions for future work.

II. RELATED WORK

We identify related work in approaches from the literature which route IoT data to edge and cloud compute nodes. To present such approaches in a comprehensive manner, we divide them into three categories, namely direct routing, multihop routing, and overlay routing. Specifically, direct routing is presented in Section II-A, multihop routing is discussed in Section II-B, and overlay routing is discussed in Section II-C. Finally, in Section II-D we provide a discussion of the existing approaches compared to the proposed edgeRouting.

A. DIRECT ROUTING APPROACHES

Most of the related approaches from the literature present a placement algorithm that distributes applications on the available compute nodes, and then route the data from the data source to a suitable compute node directly. For this reason, we refer to such approaches as *direct routing*. For example, Samanta et al. [13] design a framework for executing applications in a system with edge and cloud compute nodes, according to an auction-based placement algorithm. Each compute node in this system is assumed to be addressable from every access point in the network, so that data can be routed directly to the compute node which hosts a specific application. Bellavista et al. [14] present a middleware for fog computing. In this work, each IoT device utilizes a local agent which finds a suitable compute node considering various application constraints, and sends data directly to that node. Yao and Ansari [15] propose a system whereby the IoT data are sent directly to suitable compute nodes which use a scheduler for assigning the necessary applications on virtual machines based on an optimization problem. Mansouri and Wong [16] design a quality of experience optimization framework in which the IoT devices send data directly to either a nearby or a remote compute node. Du et al. [17] create a decision maker component which selects a user device, a nearby compute node or a remote node for deploying an application. Then, a user device communicates with the selected node directly.

B. MULTIHOP ROUTING APPROACHES

In addition to direct routing, there are other approaches which also implement a placement algorithm, but route the data on a multihop path from the data source to a suitable compute node. We refer to these as *multihop routing*. In such approaches, the data travel on a path of various compute nodes, until a suitable node accepts the data for processing. For instance, Martinez et al. [18] design a computing infrastructure in which the data are routed through various compute nodes based on optimization logic that considers transmission delay and network congestion. Ascigil et al. [19] propose a placement approach whereby the applications are distributed on edge and cloud compute nodes, and the data are routed on a multihop path along these nodes, until accepted for processing. Mortazavi et al. [20] present the design and implementation of a system that routes the data on a path of network compute nodes that exist between the edge and the cloud. Tong et al. [21] discuss a hierarchical structure of edge and cloud compute nodes for serving workloads from mobile users. In this approach, the peak workloads are routed on a path towards the cloud in order to serve a larger number of users. Okay and Ozdemir [22] propose an approach based on software-defined networks, for processing IoT data on nearby and remote compute nodes. In this approach, there are various distributed controllers which manage the local traffic (and lower the latency) while the data are routed through various nodes to a suitable compute node for processing.

C. OVERLAY ROUTING APPROACHES

There is also related work on approaches that do not stem from the edge computing literature, but do aim at improving the communication between sender and destination by routing the data through intermediate nodes. For example, Brennan and Rabinovich [23] discuss the potential benefits of overlay routing compared to native IP routing, and propose a multipath TCP approach which creates routing detours that improve throughput. Lee et al. [24] investigate the utilization of bandwidth as the main metric for creating overlay paths which are shown to have lower latency than native Internet routing. Lumezanu et al. [25] present a mechanism for finding mutually beneficial overlay detours for peer nodes, so that the communication latency is reduced. Gummadi and Madhyastha [26] explore overlay detours for creating resilient routing paths that are less likely to fail than native Internet paths. Cai et al. [27] conduct an experimental study of the communication between clouds, and show improved throughput when using detours.

D. DISCUSSION

To sum up the discussion of related approaches from the literature, most current edge computing works route the data either directly or on a multihop path, to a suitable compute node for processing. Notably, both of these routing approaches typically consider end-to-end communication, and disregard alternative network paths with detours that may lower the communication latency due to providing higher bandwidth.

The overlay detour approaches on the other hand, which consider alternative network paths, manage to show benefits compared to native Internet routing. However, most of these approaches have not been designed in the context of edge computing, and therefore, do not take into account edge compute nodes. In addition, most overlay detour approaches stem from the field of overlay networks consisting of peers, and suffer from privacy concerns related to sharing data among peer nodes [25].

In this paper, we propose using a detour when routing IoT data to edge and cloud compute nodes. We consider such environments as an ideal case for applying detours because: *i*) The IoT data are not routed through other (peer) users, but rather through the available edge compute nodes which can actually improve privacy [28], [29]. *ii*) The difference in the capacity of paths with detours can be so large (e.g., due to the different bandwidth limits between Internet and cloud providers), that utilizing such paths may lead to significant benefits (as shown in Section V).

III. SYSTEM MODEL

In this section, we present the utilized system model for processing IoT data using edge and cloud compute nodes. For this, first we discuss the architecture of the target systems

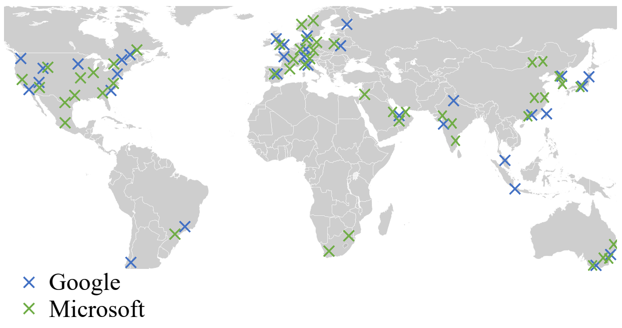


FIGURE 1: Regions around the world with computational resources offered by two cloud providers.

in Section III-A. Afterwards in Section III-B, we discuss the typical behavior of various system parameters that affect the communication latency in such systems.

A. SYSTEM ARCHITECTURE

Our system includes distributed compute nodes in data centers and at the edge of the network, which can be used for processing data from IoT devices [30]. Such nodes can be provisioned on demand by cloud providers. For example, Fig. 1 shows the regions with (existing or announced) computational resources from two cloud providers, namely Google and Microsoft [31], [32]. Notably, these regions, i.e., from two cloud providers alone, may provide compute nodes in the proximity of IoT devices in various areas around the world (e.g., in Europe, India, eastern Asia, Australia, and the US). Thus, based on Fig. 1, and by considering that other commercial cloud providers may support additional regions, we assume that compute nodes from cloud providers can be provisioned both far away, and close to IoT devices [33].

In such environments, we consider that application providers deploy their applications on compute nodes in order to process data from IoT devices. These applications can be, e.g., for detecting anomalies in the energy consumption (such as gas leaks) using smart meters, for monitoring the appearance of house visitors using smart doorbells, etc. [34]. For such cases, utilizing a single compute node with enough computational resources to process the data from all the IoT devices (i.e., a centralized cloud node) may result in high communication latency for data coming from regions outside the proximity of this node [6]. Utilizing multiple compute nodes in the proximity of all the IoT devices, each one with enough resources to process all the data, lowers the latency, but may increase the monetary cost due to maintaining many nodes with high computational resources. Thus, utilizing few compute nodes with high computational resources, and many compute nodes with lower resources but distributed around the regions of the IoT devices, can provide a balance between delay and cost [35].

Therefore, in this work we target environments with a system architecture that includes multiple geographically distributed edge and cloud compute nodes and IoT devices, as shown in Fig. 2. The IoT devices in such systems represent devices with sensors, which act as data sources. The edge and cloud compute nodes can be created on demand in any region which is supported by a cloud provider (e.g., Google, Microsoft, Amazon, Oracle, IBM, Alibaba, etc). Edge compute nodes offered by cloud providers are similar to cloud compute nodes in that they are both able to provide the same services [5]. However, since edge compute nodes reside in the proximity of the data sources, the communication latency of sending data from IoT devices to these nodes is lower [36].

The flow of the data in this system starts from the IoT devices which generate data using the sensors. These data are sent to a compute node over a logical link that is facilitated by an Internet provider (e.g., using cable or 4G Internet), as shown in Fig. 2. Notably, when the data are sent from an IoT device to a compute node, restrictions imposed by the Internet provider may apply. For instance, Internet providers commonly limit the upload bandwidth to a few Mbps [9], [10]. However, when the data are sent between compute nodes of cloud providers, such low limits do not apply, because cloud providers allow ingress/egress bandwidth with a limit that is orders of magnitude higher [11], [12].

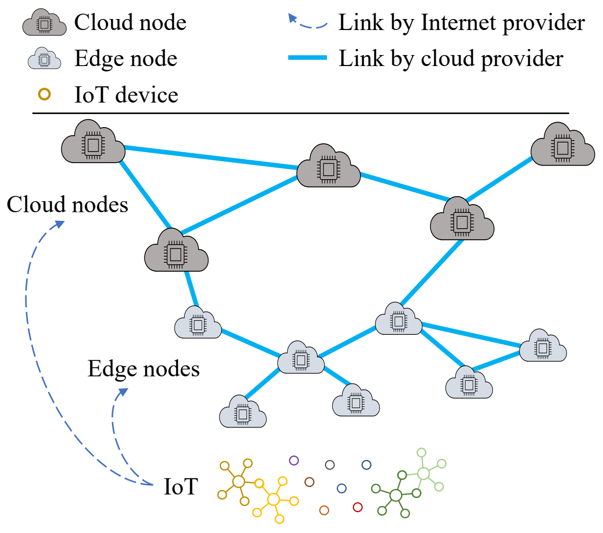


FIGURE 2: The target system architecture with distributed edge and cloud compute nodes, and IoT devices. The links show the communication between compute nodes (facilitated by cloud providers), and between IoT devices and compute nodes (facilitated by Internet providers).

B. COMMUNICATION LATENCY

The latency of sending data to a compute node can be considered as the sum of the following delays: propagation delay noted as Pro (calculated as $linkDistance/dataSpeed$), transmission delay noted as Tra (calculated as $dataSize/bandwidth$), queuing delay noted as Que (amount of time the data wait in queues, e.g., of network devices), and processing delay (time needed to process the headers) [37], [38]. The processing delay is usually very low (several microseconds), and may therefore be considered negligible [37]. These delays apply when sending a packet over a link, but for simplicity, we consider that similar delays apply when sending IoT data between nodes. Thus, all the factors that contribute to the communication latency are considered by taking into account the aforementioned delays. Notably, each one of these delays consists of all the time periods that are required for sending the data between various nodes in the underlying network, until the data reach the destination compute node. This means that, e.g., the Pro of sending data from an IoT device to a cloud compute node, includes all the propagation delays of the logical link that connects sender and destination. The same applies for Tra, and Que.

Such delays apply regardless of the utilized communication protocol, e.g., TCP, UDP, HTTP, etc. [39]. Each protocol may have a different message structure which can affect the discussed delays, e.g., an HTTP message may be larger than a UDP message with the same payload, which affects the Tra. Similarly, the specific characteristics of the Internet connection, and underlying network, e.g., using cable, or 4G Internet, can affect these delays. However, when the same communication protocol is used over the same Internet connection, we consider that measuring the communication latency in this manner provides a good approximation for comparing different routing approaches in systems with edge and cloud compute nodes.

The nodes in our model are represented by a set $s = \{n_1, n_2, \ldots, n_N\}$ which includes n_1 being the IoT device, n_2 being the edge compute node in closest proximity of n_1, and all the other compute nodes of the system. Thus, to refer to, e.g., the Pro between an IoT device and the compute node in closest proximity, we use Pro_{n_1,n_2} (similarly: Tra_{n_1,n_2}, Que_{n_1,n_2}, etc.). n_1 may send data to any of the other nodes of s for processing. The compute node that processes the data is denoted as n_X. To make our system model more comprehensible, we summarize all the utilized notation in Table 1.

As discussed in Section III-A, this work targets systems with edge and cloud compute nodes offered by cloud providers, which are used for processing IoT data over the Internet. Based on the literature of similar systems, we deduce the following formulations regarding the typical behavior of the communication latency when sending data to nearby and

TABLE 1: The utilized notation.

Symbol	Description
$s = \{n_1, n_2, \ldots, n_N\}$	Set including all the nodes of the system.
n_1	The IoT device (i.e., a data source).
n_2	Edge compute node closest to n_1.
n_X	Compute node that processes the data.
Pro_{n_i,n_j}	Propagation delay from n_i to n_j.
Tra_{n_i,n_j}	Transmission delay from n_i to n_j.
Que_{n_i,n_j}	Queuing delay from n_i to n_j.
$DLat_{n_1,n_X}$	Direct routing latency from n_1 to n_X.
$MLat_{n_1,n_X}$	Multihop routing latency from n_1 to n_X.
$ELat_{n_1,n_X}$	edgeRouting latency from n_1 to n_X.

remote compute nodes [40], [41]:

- Pro is defined as $linkDistance/dataSpeed$, which means that this delay is affected by the speed that the data travel on the network links, and the traveled distance between sender and destination. Since the speed can be considered a constant (which is usually around $2 \cdot 10^8$ meters per second based on the signal speed on copper cables) [42], Pro depends on the distance of the network path that connects sender and destination. Based on this definition, we can also say that since the denominator is a very high constant number, Pro approaches zero, when the numerator becomes small. Thus, when n_1 is very close to n_2, Pro_{n_1,n_2} approaches a very small number which does not contribute considerably to the communication latency, and may therefore be considered insignificant [40]. Notably, this applies to Pro_{n_1,n_2}, but not the whole communication latency between n_1 and n_2, which may be higher. Hence:

$$Pro_{n_1,n_2} \approx 0 \tag{1}$$

- Tra is defined as $dataSize/bandwidth$, which means that for the same data size, this delay depends on the available bandwidth. We can also say that when the bandwidth becomes abundant, Tra approximates zero. Since edge compute nodes of cloud providers belong to the same network of the provider's data centers, the bandwidth limit is the same for both edge and cloud compute nodes [5]. This limit may vary according to each cloud provider and the current network load, but is usually very high. For example, Google allows seven Gigabits per second (Gbps) of egress bandwidth to external IPs, while the ingress bandwidth can be more or even unlimited [11]. Microsoft also allows unlimited ingress/egress bandwidth [43]. Notably, even though the bandwidth may be abundant, a compute node needs to have sufficient processing resources (e.g., CPU and memory) to process the ingress/egress data. In addition, the utilized bandwidth may be subject to a pricing model. When such restrictions do not pose significant concerns, the compute nodes of cloud providers are able to utilize as much network bandwidth as they need. Therefore, Tra does not contribute considerably to the communication latency, and can be considered insignificant, when the sender and the destination are both compute nodes which communicate with very high bandwidth [44]. Hence:

$$\forall\, n_i, n_j \in s - \{n_1\}, \quad Tra_{n_i,n_j} \approx 0 \tag{2}$$

- The queuing delay between compute nodes of cloud providers can also be considered insignificant in the calculation of the communication latency. This may happen when there is abundant bandwidth, as previously discussed, that makes it unlikely for the data to have to wait in queues [41]. Hence, Que does not contribute considerably to the communication latency when sender and destination are compute nodes of cloud providers, i.e.:

$$\forall\, n_i, n_j \in s - \{n_1\}, \quad Que_{n_i,n_j} \approx 0 \tag{3}$$

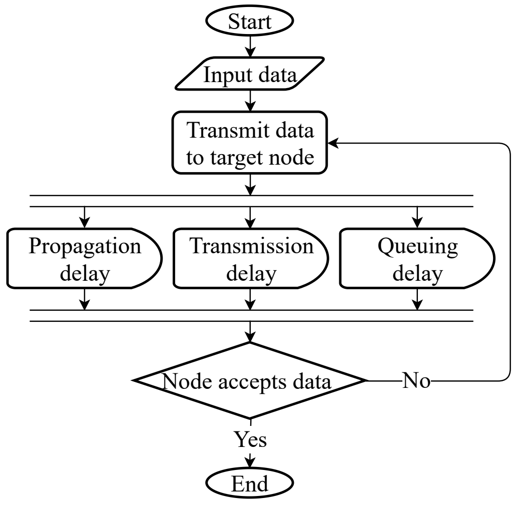


FIGURE 3: Flowchart with the communication latency of sending data to a compute node.

We use Equations (1), (2), and (3), for comparing the communication latency of different routing approaches in Section IV. After that, we revisit them in Section V-D, and we discuss these equations from a practical point of view based on our experiments, also considering cases when these approximations deviate from zero.

The overall communication delay of executing an application on a compute node of a cloud provider always includes the delay of sending the data from the sender to the destination node, i.e., the upload latency. In case the application produces a response that needs to be sent back to the sender, the overall communication delay also includes the download latency. For simplicity, in this work we focus on reducing the communication latency of sending the data, i.e., the upload latency, because this latency constitutes the minimum delay to start executing an application. Nevertheless, in our evaluation in Section V-C, we also discuss how the download latency is affected by the different routing approaches.

To further elaborate on how the communication latency of sending data to distributed compute nodes is modeled in this paper, we depict a flowchart in Fig. 3. Initially, the input data are generated at n_1 and sent to a target node $n \in s - \{n_1\}$. This transmission is subject to propagation, transmission, and queuing delays, until the data arrive at the target node. Upon arrival, this node decides to either accept the data for processing, or to forward the data to a new target node. The former results in the end of the transmission which means that there is no more communication latency. The latter results in another transmission which is repeated until the data are accepted by a node. Thus, the communication latency of sending data to distributed compute nodes is considered as the period from the time the data are sent from n_1 until the data are accepted by a node.

IV. SENDING IOT DATA TO COMPUTE NODES

In this section, we examine the latency of sending IoT data to edge and cloud compute nodes, based on different routing approaches. Specifically, Sections IV-A and IV-B analyze the latency of common routing approaches from the literature (discussed in Section II). Afterwards in Section IV-C, we take into account this analysis in order to design a routing approach which lowers the latency by utilizing the underlying network resources more efficiently.

A. DIRECT ROUTING

As discussed in Section II, direct routing is used widely for sending IoT data to edge and cloud compute nodes. According to this approach, the IoT devices are configured to send the data directly to the compute nodes which perform the required processing [13]. In the flowchart of Fig. 3, this means that when using direct routing the data are accepted by the target node after one transmission. Thus, the target node accepts the data the first time, and the loop that starts when a node does not accept the data (in Fig. 3) is not triggered. To facilitate this, a placement algorithm is utilized for placing the applications on suitable compute nodes based on various resource-related (e.g., CPU and RAM) and/or network-related (e.g., transmission and propagation delay) metrics [45]. After the applications are placed, the data are commonly routed from an IoT device (through a local gateway [46]) to a suitable compute node directly, using the connection of an Internet provider, as shown in Fig. 4a.

Thus, assuming that data are sent from an IoT device n_1 to a suitable compute node n_X with $n_X \in s - \{n_1\}$, the communication latency using direct routing $DLat_{n_1,n_X}$ can be defined as the sum of the propagation, transmission, and queuing delay, i.e.:

$$DLat_{n_1,n_X} = Pro_{n_1,n_X} + Tra_{n_1,n_X} + Que_{n_1,n_X} \quad (4)$$

Pro_{n_1,n_X} depends on the speed that the data travel on the network links, and the distance between n_1 and n_X. Since the speed can be considered a constant (as discussed in Section III-B), this delay is affected by the distance of the network path that connects n_1 and n_X. This path is usually selected based on a gateway protocol that implements a best path algorithm [47].

Tra_{n_1,n_X} depends on the upload bandwidth limit of the Internet provider, and the data size. The data size is a variable which depends on the specific application. For example, applications that perform analytics based on sensor measurements may utilize smaller data sizes than image processing applications.

Que_{n_1,n_X} depends on the time that the data wait in queues, e.g., the buffers of the network devices that perform the routing. Notably, if the buffers are full, e.g., due to congestion (which may occur in IoT environments), packets are dropped and need to be retransmitted. This also increases the communication latency [48].

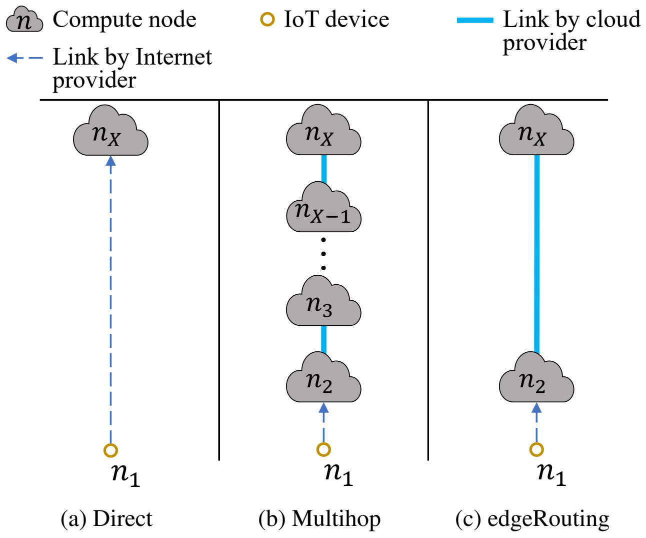


FIGURE 4: Routing approaches for sending data to edge and cloud compute nodes. The links show the provider that facilitates the communication (either cloud provider or Internet provider).

Therefore, while the propagation delay is not affected by the bandwidth limit of the Internet provider, the transmission delay can increase when the network bandwidth is low. Moreover, low bandwidth may result in the data waiting in queues longer, which might lead to congestion, dropped packets and retransmissions [48]. Thus, the bandwidth of the Internet provider plays a critical role in the communication latency of direct routing.

B. MULTIHOP ROUTING

Similar to direct routing, in multihop routing there is also a placement algorithm which distributes the applications on the compute nodes (as discussed in Section II-B). However, the flow of the data is different. In multihop routing, the data are sent from an IoT device n_1 to an edge compute node n_2. If this node hosts the required application, and has adequate available computational resources to perform the required processing, the data are accepted. Otherwise, the data are rerouted to the next node in proximity [21]. Similarly, that node either accepts or reroutes the data, and the process repeats. This process is bound to finish, because usually the last node on a multihop path is a cloud compute node which is assumed to always be able to perform the required processing [36].

In the flowchart of Fig. 3, when using multihop routing the data are accepted by the target node after one or more transmissions (one transmission if $n_X = n_2$, more otherwise). This means that the loop that starts when a node does not accept the data (in Fig. 3) may be triggered multiple times, thereby initiating more transmissions and inducing more delay. These transmissions are facilitated by different providers, and may therefore have different bandwidth limits. As shown in Fig 4b, the data are sent from the IoT device to the compute node in proximity using the connection of an Internet provider. Assuming that this node is not able to perform the required processing, the data are routed again to another compute node. This time however, since the data are routed from a compute node, the transmission is facilitated by a cloud provider, i.e., with higher bandwidth.

Thus, when sending data from an IoT device n_1 to a suitable compute node n_X using multihop routing, the communication latency $MLat_{n_1,n_X}$ consists of the propagation, transmission, and queuing delays, between each sender and destination until the data reach n_X, i.e.:

$$\begin{aligned} MLat_{n_1,n_X} = Pro_{n_1,n_2} + Tra_{n_1,n_2} + Que_{n_1,n_2} + \\ Pro_{n_2,n_3} + Tra_{n_2,n_3} + Que_{n_2,n_3} + \ldots + \\ Pro_{n_{X-1},n_X} + Tra_{n_{X-1},n_X} + Que_{n_{X-1},n_X} \end{aligned} \tag{5}$$

The propagation delay Pro_{n_1,n_X} in multihop routing is the sum of the propagation delay of each transmission, i.e., $Pro_{n_1,n_2} + Pro_{n_2,n_3} + \ldots + Pro_{n_{X-1},n_X}$. Thus, the routing path from n_1 to n_X includes detours (i.e., $n_2, n_3, \ldots, n_{X-1}$), and is not based on a best path algorithm (unless $n_2, n_3, \ldots, n_{X-1}$ exist within the best path from n_1 to n_X). Therefore, the traveled distance of the data in multihop routing, is expected to be longer than a direct transmission from n_1 to n_X, which means that the propagation delay is increased.

The transmission delay Tra_{n_1,n_X} consists of the transmission delay from n_1 to n_2, and the transmission delay between compute nodes of cloud providers. The former can be increased by the bandwidth limit imposed by the Internet provider. The latter however, can be considered rather insignificant based on Equation (2).

The queuing delay Que_{n_1,n_X} consists of the queuing delay from n_1 to n_2, and the queuing delays between compute nodes. Que_{n_1,n_2} can be affected by low bandwidth and congestion, but this is unlikely because nodes that reside close to each other do not typically suffer from low bandwidth [36], [49], although the limit of the Internet provider still applies. The queuing delays between compute nodes are expected to be insignificant based on Equation (3).

C. EDGEROUTING

Based on the analysis of the latency in Sections IV-A and IV-B, we note that while direct routing has low propagation delay, the transmission and queuing delays may increase due to the bandwidth limit of the Internet provider. Multihop routing on the other hand, may have low transmission and queuing delays due to the high bandwidth between compute nodes, but the propagation delay may increase due to the detours. Thus, in order to leverage both the low propagation delay of nodes in proximity, and the high bandwidth between compute nodes, we propose edgeRouting.

According to edgeRouting, the data are routed from an IoT device n_1 to the edge compute node in closest proximity n_2, and from there, the data are routed directly to the suitable compute node n_X for processing, as shown in Fig. 4c. In the flowchart of Fig. 3, when using edgeRouting the data are accepted by the target node after one or two transmissions (one transmission if $n_X = n_2$, two otherwise). This means that the loop that starts when a node does not accept the data (in Fig. 3) is either not executed (when $n_X = n_2$), or executed one time (when $n_X \neq n_2$) which triggers a second transmission directly to n_X. The first transmission is routed by the Internet provider, and the second by the cloud provider. This way, during the first routing the data are sent with low propagation delay, and during the second routing, the data are sent with low transmission and queuing delays due to the high bandwidth.

Notably, direct routing commonly preconditions that there is a placement algorithm which distributes the applications on the compute nodes, and that the IoT devices (or the gateways) know the address of n_X in order to initiate the transmission of the data (as discussed in Section IV-A). Similarly, multihop routing usually preconditions that there is a placement algorithm, and that the IoT devices know the address of n_2 for initiating the transmission (as discussed in Section IV-B). edgeRouting preconditions a placement algorithm, that the address of n_2 is known, and that the address of n_X is known. This enables sending the data to n_2 along with the address of n_X which is used by n_2 for the second routing.

The communication latency in edgeRouting $ELat_{n_1,n_X}$ consists of the propagation, queuing, and transmission delays to send the data from n_1 to n_2, and from n_2 to n_X, i.e.:

$$\begin{aligned} ELat_{n_1,n_X} = Pro_{n_1,n_2} + Tra_{n_1,n_2} + Que_{n_1,n_2} + \\ Pro_{n_2,n_X} + Tra_{n_2,n_X} + Que_{n_2,n_X} \end{aligned} \tag{6}$$

The propagation delay Pro_{n_1,n_2} can be insignificant based on Equation (1). Pro_{n_2,n_X} depends on the distance between n_2 and n_X. The transmission delay includes the Tra_{n_1,n_2} which can be increased by the bandwidth limit of the Internet provider, and Tra_{n_2,n_X} which can be insignificant based on Equation (2). The queuing delay contains Que_{n_1,n_2} which is unlikely to be affected by congestion (as discussed in Section IV-B), and Que_{n_2,n_X} which can be insignificant based on Equation (3). Regarding the comparison of edgeRouting with direct and multihop routing, we deduce that edgeRouting is able to provide similar or lower communication latency based on Lemmata 1 and 2:

Lemma 1. *Equation (6) $\lesssim$ Equation (4).*

Proof: When the compute node in closest proximity performs the required processing, i.e., $n_X = n_2$, for Equation (6) applies that: $Pro_{n_2,n_X} + Tra_{n_2,n_X} + Que_{n_2,n_X} = 0$ because this part of the equation represents the communication latency to send the data from n_2 to n_2, which does not incur any latency. Thus, Equation (6) $= Pro_{n_1,n_2} + Tra_{n_1,n_2} + Que_{n_1,n_2} + 0 =$ Equation (4).

When $n_X \neq n_2$, the propagation delay of edgeRouting according to Equation (6), i.e., $Pro_{n_1,n_2} + Pro_{n_2,n_X}$, is approximately equal to Pro_{n_2,n_X}, because Pro_{n_1,n_2} can be considered insignificant based on Equation (1). Pro_{n_2,n_X} can be considered approximately equal to the propagation delay of direct routing from Equation (4), i.e., Pro_{n_1,n_X}, because n_1 and n_2 reside in proximity, which means that the distance from n_1 to n_X is similar to the distance from n_2 to n_X. It is possible that these two transmissions take different paths to n_X, which may have slightly different path distances. However, the very high data speed of the transmissions (discussed in Section III-B) makes Pro resilient to small distance changes. Thus, even in such cases Pro does not change significantly. Consequently, since $Pro_{n_2,n_X} \approx Pro_{n_1,n_X}$, and $Pro_{n_1,n_2} \approx 0$, the propagation delay of edgeRouting is approximately equal to the propagation delay of direct routing, i.e.:

$$Pro_{n_1,n_2} + Pro_{n_2,n_X} \approx Pro_{n_1,n_X} \tag{7}$$

The transmission delay of direct routing from Equation (4), i.e., Tra_{n_1,n_X}, depends on the bandwidth limit of the Internet provider. Since the bandwidth may be shared among many users, the effective bandwidth of Tra_{n_1,n_X} can be lower than the limit, based on the overall network traffic which depends on the current network load [50]. This phenomenon can cause low bandwidth in Tra_{n_1,n_X}, especially when n_X is a remote cloud compute node [9]. Tra_{n_1,n_2} is also facilitated by the Internet provider, and can be affected by the current network load too. However, since n_1 and n_2 reside close to each other, Tra_{n_1,n_2} is unlikely to suffer from low bandwidth [36], [49]. Thus, we consider that $Tra_{n_1,n_2} \leq Tra_{n_1,n_X}$. The equality stands when the network is not/equally loaded in both transmission delays, which means that similar bandwidth is utilized. The inequality applies when the network is loaded due to n_X being a remote node, and Tra_{n_1,n_X} has less bandwidth than Tra_{n_1,n_2}. The case that the network is loaded only at the edge, i.e., between n_1 and n_2, without being loaded between n_1 and n_X is unlikely because based on our system model, n_X can either equal n_2 or be farther away. The transmission delay of edgeRouting according to Equation (6) includes Tra_{n_1,n_2} and Tra_{n_2,n_X}. Tra_{n_2,n_X} can be considered insignificant based on Equation (2). Thus, $Tra_{n_2,n_X} \approx 0$ and $Tra_{n_1,n_2} \leq Tra_{n_1,n_X}$, which means that the transmission delay of edgeRouting is similar or lower than the transmission delay of direct routing, i.e.:

$$Tra_{n_1,n_2} + Tra_{n_2,n_X} \lesssim Tra_{n_1,n_X} \tag{8}$$

The queuing delay of edgeRouting according to Equation (6) consists of Que_{n_1,n_2} which is unlikely to suffer from congestion, and Que_{n_2,n_X} which can be considered insignificant (both are discussed in Section IV-B). The queuing delay of direct routing from Equation (4), i.e., Que_{n_1,n_X}, may be increased due to congestion in the routing path from n_1 to n_X (as discussed in Section IV-A). Thus, we assume that $Que_{n_1,n_2} \leq Que_{n_1,n_X}$. The equality applies if there is no congestion, whereas the inequality applies if there is congestion in the path from n_1 to n_X. Therefore, since $Que_{n_2,n_X} \approx 0$, and $Que_{n_1,n_2} \leq Que_{n_1,n_X}$, the queuing delay of edgeRouting is similar or lower than the queuing delay of direct routing, i.e.:

$$Que_{n_1,n_2} + Que_{n_2,n_X} \lesssim Que_{n_1,n_X} \tag{9}$$

By adding Inequalities (7), (8), and (9), applies that:

$$\begin{aligned} &Pro_{n_1,n_2} + Pro_{n_2,n_X} + Tra_{n_1,n_2} + Tra_{n_2,n_X} + \\ &Que_{n_1,n_2} + Que_{n_2,n_X} \lesssim Pro_{n_1,n_X} + Tra_{n_1,n_X} + \\ &Que_{n_1,n_X} \Leftrightarrow \\ &Equation\ (6) \lesssim Equation\ (4) \end{aligned} \tag{10}$$

Hence, we note that the communication latency of edgeRouting is expected to be lower than direct routing because the transmission and queuing delays are likely to be lower, while the propagation delay is likely to be similar. This applies because edgeRouting enables the IoT devices to avoid direct communication with remote compute nodes, thereby avoiding network paths with potential congestion and low bandwidth.

Lemma 2. *Equation (6) $\lesssim$ Equation (5).*

Proof: When the compute node in closest proximity performs the required processing, i.e., $n_X = n_2$, Equation (6) $= Pro_{n_1,n_2} + Tra_{n_1,n_2} + Que_{n_1,n_2} =$ Equation (5). Similarly, when $n_X = n_3$, Equation (6) $= Pro_{n_1,n_2} + Pro_{n_2,n_3} + Tra_{n_1,n_2} + Tra_{n_2,n_3} + Que_{n_1,n_2} + Que_{n_2,n_3} =$ Equation (5).

When $n_X \neq n_2$ and $n_X \neq n_3$, both the propagation delay of edgeRouting according to Equation (6), and the propagation delay of multihop routing according to Equation (5) include Pro_{n_1,n_2}. Apart from that, edgeRouting also includes Pro_{n_2,n_X}, whereas multihop routing includes $Pro_{n_2,n_3} + \ldots + Pro_{n_{X-1},n_X}$. For these delays applies that the former is lower or equal, i.e., that $Pro_{n_2,n_X} \leq Pro_{n_2,n_3} + \ldots + Pro_{n_{X-1},n_X}$. The equality applies if $n_3, \ldots, n_{X-1}$ exist on the path from n_2 to n_X, i.e., the same path that is followed in Pro_{n_2,n_X}. The inequality applies when $n_3, \ldots, n_{X-1}$ are detours, because detours increase the distance from n_2 to n_X and consequently, the propagation delay (as discussed in Section IV-B). Thus, by adding Pro_{n_1,n_2} to both sides of the aforementioned inequality, applies that the propagation delay of edgeRouting is equal or lower than the propagation delay of multihop routing, i.e.:

$$Pro_{n_1,n_2} + Pro_{n_2,n_X} \leq Pro_{n_1,n_2} + \ldots + Pro_{n_{X-1},n_X} \tag{11}$$

The transmission delay of edgeRouting according to Equation (6) includes Tra_{n_1,n_2} which is also included in the transmission delay of multihop routing according to Equation (5). Except for that, edgeRouting includes Tra_{n_2,n_X}, while multihop routing includes $Tra_{n_2,n_3} + \ldots + Tra_{n_{X-1},n_X}$. These are all transmission delays between compute nodes, which can be considered insignificant based on Equation (2). Thus, $Tra_{n_2,n_X} \approx Tra_{n_2,n_3} + \ldots + Tra_{n_{X-1},n_X}$. By adding Tra_{n_1,n_2} to both sides, applies that the transmission delay of edgeRouting is similar to the transmission delay of multihop routing, i.e.:

$$Tra_{n_1,n_2} + Tra_{n_2,n_X} \approx Tra_{n_1,n_2} + \ldots + Tra_{n_{X-1},n_X} \tag{12}$$

The queuing delay of edgeRouting in Equation (6), and the queuing delay of multihop routing in Equation (5) both include queuing delays between compute nodes, which can be considered insignificant based on Equation (3). Thus, $Que_{n_2,n_X} \approx Que_{n_2,n_3} + \ldots + Que_{n_{X-1},n_X}$. By adding Que_{n_1,n_2} to both sides, applies that the queuing delay of edgeRouting is similar to the queuing delay of multihop routing, i.e.:

$$Que_{n_1,n_2} + Que_{n_2,n_X} \approx Que_{n_1,n_2} + \ldots + Que_{n_{X-1},n_X} \tag{13}$$

By adding Inequalities (11), (12), and (13), applies that:

$$\begin{gathered} Pro_{n_1,n_2} + Pro_{n_2,n_X} + Tra_{n_1,n_2} + Tra_{n_2,n_X} + \\ Que_{n_1,n_2} + Que_{n_2,n_X} \lesssim Pro_{n_1,n_2} + \ldots + \\ Pro_{n_{X-1},n_X} + Tra_{n_1,n_2} + \ldots + Tra_{n_{X-1},n_X} + \\ Que_{n_1,n_2} + \ldots + Que_{n_{X-1},n_X} \Leftrightarrow \\ Equation\ (6) \lesssim Equation\ (5) \end{gathered} \tag{14}$$

Hence, we note that the communication latency of edgeRouting is expected to be lower than multihop routing because the propagation delay is likely to be lower, while the transmission and queuing delays are likely to be similar. This applies because the routing path of multihop routing may include detours which increase the traveled distance of the data, and consequently the communication latency.

V. EVALUATION

To evaluate edgeRouting, we implement a prototype for each routing approach, i.e., the proposed edgeRouting, and the direct and multihop routing. The latter two represent alternative approaches (as discussed in Section II) which we use as baselines. Furthermore, we run experiments and we measure the incurred communication latency. To present our results in intelligibly, first we describe the details of the evaluation environment (in Section V-A), and then we discuss the actual quantitative results (in Section V-B). Subsequently, we provide a discussion of the examined approaches based on other aspects than the communication latency (in Section V-C). Finally, we elaborate on weaknesses and limitations of the work at hand (in Section V-D).

A. EVALUATION ENVIRONMENT

1) Evaluation Setup

To create a system as shown in Fig. 2, we employ a Raspberry Pi 4 as the IoT device, and distributed compute nodes provisioned using the Google Cloud Platform (GCP). We position the IoT device in central Europe (in Vienna, Austria), and we provision compute nodes in the currently available regions of the GCP in the broader area, i.e., Zurich, Frankfurt, Belgium, Netherlands, and London. We consider this to be an appropriate setup for examining the communication latency of sending IoT data to distributed compute nodes, because there are already many experimental IoT and smart city applications running in this area (e.g., in France, Spain, Serbia, Germany, and the United Kingdom) [51].

Notably, even though edge compute nodes have been announced by cloud providers (e.g., the Edge Zones by Microsoft), their availability is still limited [5]. For this reason, we emulate the edge compute node n_2 using the compute node in closest proximity to n_1 (i.e., the compute node in Zurich). This way, our emulated edge compute node can be reached with the lowest propagation delay, while providing high bandwidth when sending data to other compute nodes (the same as the edge compute nodes discussed in our system model in Section III). However, our edge compute node is not very close to the IoT device, e.g., in the same city, but rather in a nearby city which is approximately 600 kilometers away. This means that the propagation delay between n_1 and n_2 can be about 3 ms, based on the propagation delay definition discussed in Section III-B. In fact, this delay might be slightly higher because the network distance between n_1 and n_2 can be a bit larger than the physical distance. Consequently, the propagation delay using the emulated edge compute node might not be extremely small as discussed in Section III-B, but it is still small enough not to affect the presented results significantly. This is discussed further in Section V-C in which we analyze the error that occurs due to emulating the edge compute node. In brief, this error is estimated to be trivial for this evaluation.

Regarding the underlying network, the bandwidth between GCP compute nodes can be up to 7 Gbps because we use the external IPs of the compute nodes [11]. Even though internal IPs enable higher bandwidth, we use the external IPs in order to make our setup representative of scenarios with compute nodes from different cloud providers, i.e., when the use of internal IPs is not possible. For the bandwidth between the IoT device and the compute nodes (i.e., from the Internet provider) we use a standard cable connection with upload bandwidth of up to 6 Mbps (and download bandwidth of up to 50 Mbps). Additionally, since 4G Internet has become typical for IoT devices [52], we also use a standard 4G Internet connection with upload bandwidth of up to 7.17 Mbps (and download bandwidth of up to 48.97 Mbps).

2) Evaluation Baselines

For the implementation of the multihop routing approach, the data are sent on a path from the IoT device n_1 to n_X, i.e., $n_1, n_2, \ldots, n_{X-1}, n_X$. The compute nodes on this path are ordered based on network proximity. This means that n_2 is the closest compute node to n_1, n_3 is the closest to n_2, etc. To measure proximity, we use round trip times which consistently result in the same order of nodes. Notably, the order of the compute nodes based on network proximity is the same as based on physical proximity, i.e., n_1 is in Vienna, n_2 is in Zurich, n_3 is in Frankfurt, n_4 is in Belgium, n_5 is in the Netherlands, and n_6 is in London. To achieve this order, n_1

is configured to send the data to n_2, n_2 to n_3, etc., until n_X. For the implementation of the direct routing approach, the IoT device n_1 is configured to send the data directly to n_X. edgeRouting is implemented as discussed in Section IV-C.

The suitable compute node n_X is commonly selected using a placement algorithm, as discussed in Sections I and II. However, in order to gather information from routing data to each compute node of the system, we run separate experiments with each node being selected as n_X. To implement the routing approaches, we use prototypes developed in Java 11, which we deploy both on the IoT device (for sending the data), and on the compute nodes (for rerouting or accepting the data for processing). All the nodes use the Debian 10 operating system (the Raspberry Pi uses the Debian 10-based Raspbian).

3) Evaluation Experiments

Since the proposed edgeRouting aims at reducing the communication latency, we conduct various experiments, and we measure the induced latency for each routing approach. Considering that the communication latency can also be affected by the data size (as discussed in Section III-B), we perform experiments with different payloads in order to acquire results that can be extrapolated to a wider range of use cases. To represent small data sizes, we consider sensor measurements of few bytes, while for larger data sizes we examine images of approximately 430 kilobytes (kB). Both of these data sizes are selected to represent actual IoT applications, namely: a smart home application, and an IoT image processing application.

For the small data sizes, we present the results of using as payload the measurements of a smart gas sensor (in Section V-B1). The actual values of these measurements are acquired from a publicly available dataset that is provided by the Loughborough University, and is part of the data gathered in the context of the REFIT project which monitored various smart homes in the United Kingdom [53]. Since these values have been collected periodically every 30 minutes, we configure our IoT device to produce periodic messages with the same frequency. Each message includes a gas sensor measurement as payload, and is sent to a compute node n_X for processing. This configuration can be representative of a smart energy use case (e.g., for detecting anomalies such as leaks and malfunctions), and also for a variety of other IoT use cases that include sending periodic sensor measurements to a compute node [54].

For the large data sizes, we experiment with images that can be representative of various IoT use cases that perform image processing (in Section V-B2). For example, a smart doorbell that sends a notification with the image of the visitor to the house owner, or opens the door automatically if the visitor is also a resident [34]. The size of about 430 kB is selected for being a common size of images intended for face recognition in environments with edge and cloud compute nodes [55]. We also perform experiments with face recognition tasks being executed in compute nodes using the OpenCV library [56], which aid in interpreting the communication latency results better. This is discussed in Section V-C.

Since measurements of experiments conducted using the Internet can vary over time due to potential changes in the network load, we run the three examined routing approaches at the same time. Thus, each generated sensor measurement is sent to n_X three times based on each routing approach (sequentially). We do this in order to collect results which represent the examined approaches under the same network conditions, thereby making the results of each approach comparable to the others.

In the experiments, we measure the time needed to send the data from the IoT device n_1 to n_X. This time period represents the communication latency from n_1 to n_X including all delays, and potential additional rerouting by compute nodes on path (i.e., detours). Since we use HTTP requests to send the data, the communication latency also includes parsing of the HTTP headers and making the data available to n_X, but excludes any other processing by n_X.

B. EVALUATION RESULTS

For this evaluation, we run 2,000 experiments for each routing approach, in order to acquire results that capture the general behavior of the system for each setting. Notably, the latency measurements of all the examined approaches include outliers. This is expected in experiments conducted over the Internet, because the resources of the underlying network may be shared among many users which affect the network load. The outliers represent isolated occurrences of maximum values which may affect the average latency, despite being rather rare. For this reason, we consider the median as a more representative statistical measure of the results, since it is not affected (as much) by the outliers. To visualize our results in a meaningful manner, in the following we present box plots which include all the latency values of our experiments, and we interpret these values based on the median, and the difference in the interquartile range.

1) Sensor Measurements as Payload

In Figs. 5 and 6, we plot the distribution of the latency values when sending sensor measurements via cable and 4G Internet. These figures show the latency when each compute node of the system is selected as n_X. Specifically, Fig. 5a shows the latency of direct routing over cable, Fig. 5b shows the latency of multihop routing over cable, and Fig. 5c shows the latency of edgeRouting over cable. Similarly, Figs. 6a, 6b, and 6c shows the latency of the examined routing approaches over 4G.

When using the same Internet connection, the latency of n_2 is always very similar, and the latency of n_3 is very similar between multihop routing and edgeRouting. This is expected due to the proofs of Lemmata 1 and 2 which say that n_2 has the same latency in all the examined routing approaches, and n_3 has the same latency in multihop routing and edgeRouting.

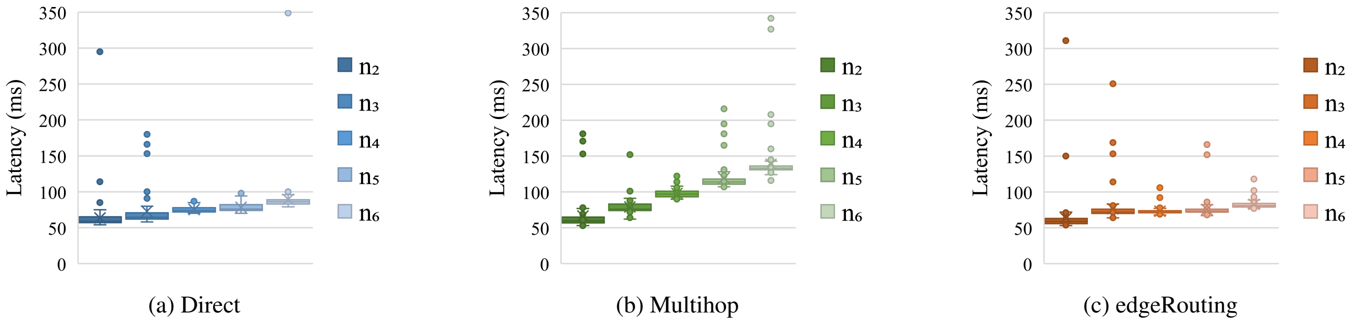


FIGURE 5: Latency of sending sensor measurements to distributed compute nodes via cable Internet for each routing approach.

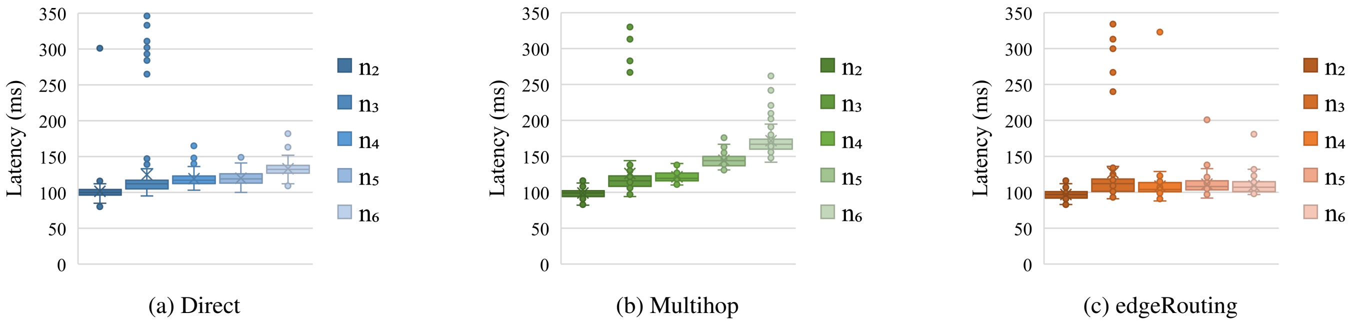


FIGURE 6: Latency of sending sensor measurements to distributed compute nodes via 4G Internet for each routing approach.

For direct routing, we note that based on the interquartile ranges of Figs. 5a, and 6a, the latency has a tendency to increase when n_X resides farther away from n_1. This is expected since the propagation delay increases due to the longer distance between n_1 and n_X. Multihop routing also exhibits this behavior based on the interquartile ranges of Figs. 5b, and 6b, although the rate of the increase is higher. This happens because the propagation delay of multihop routing includes detours which are not present in direct routing. In edgeRouting as shown in Figs. 5c, and 6c, this increase is not as evident. There is a clear increase in the interquartile ranges from n_2 to n_3, but after that the latency stabilizes with a slight increase for n_6 in Fig. 5c. This happens because even though the propagation delay increases, this increase is countered by a decrease in the transmission and queuing delays due to the transmissions between compute nodes.

To visualize the trend of each routing approach in a more obvious way, Fig. 7 shows only the median values of latency when using cable Internet (from Fig. 5). Similarly, in Fig. 8, we show only the median values of latency when using 4G Internet (from Fig. 6). In Fig. 7, we note that all approaches start with a very similar median at n_2 (about 59 ms). At n_3, we observe that the median of direct routing (64 ms) is increased only slightly, while multihop and edgeRouting are a bit higher (about 74 ms). This happens because multihop and edgeRouting follow the same path to n_3 which includes a detour from n_2. After that, the median of multihop routing keeps increasing until n_6 (133 ms), while the median values of direct routing and edgeRouting are very similar, and do not increase significantly until n_6 (the edgeRouting median is 81 ms, and the direct routing median is 86 ms). Thus, we conclude that when sending sensor measurements over cable Internet, edgeRouting is similar or better than multihop

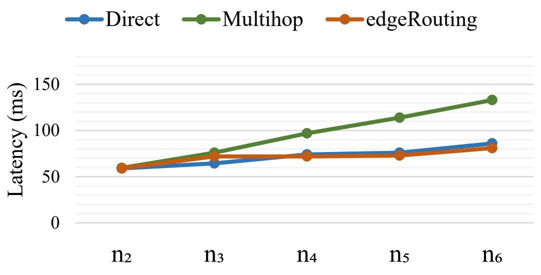


FIGURE 7: Median values of latency from Fig. 5 for each routing approach.

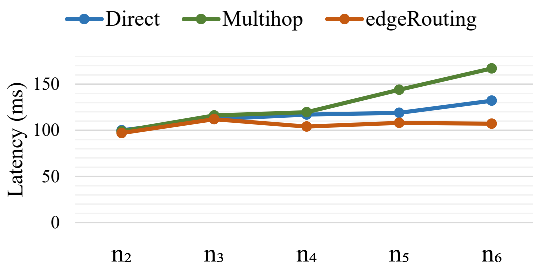
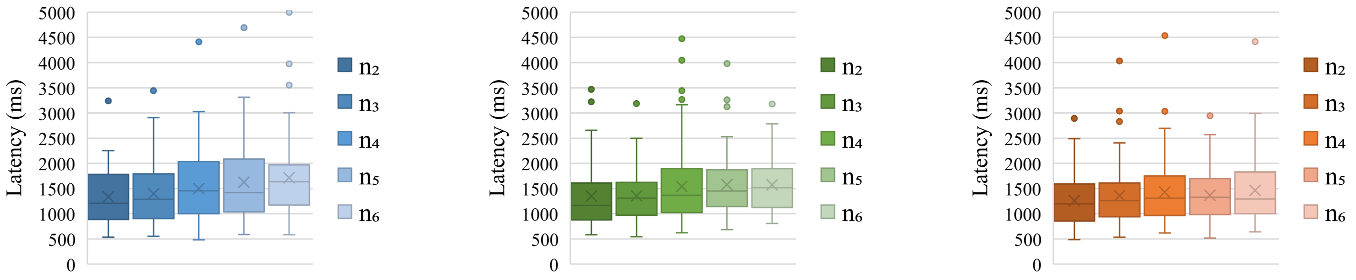
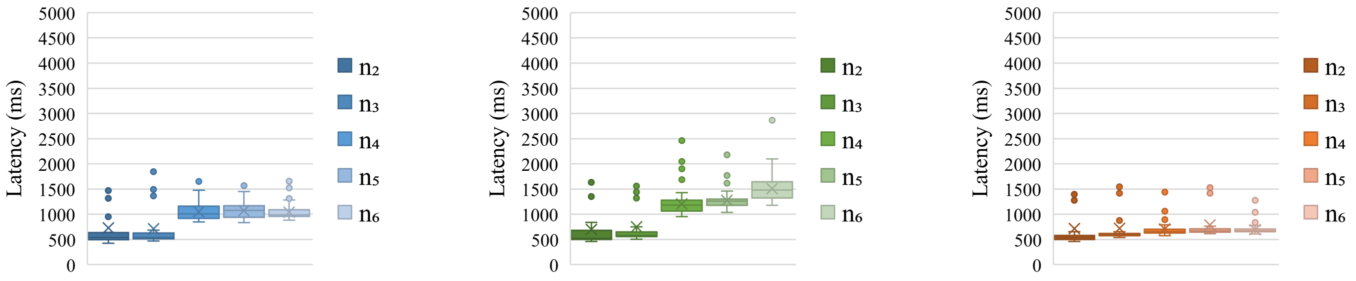


FIGURE 8: Median values of latency from Fig. 6 for each routing approach.



(a) Direct (b) Multihop (c) edgeRouting

FIGURE 9: Latency of sending images to distributed compute nodes via cable Internet for each routing approach.



(a) Direct (b) Multihop (c) edgeRouting

FIGURE 10: Latency of sending images to distributed compute nodes via 4G Internet for each routing approach.

routing with latency reductions that reach up to 39% based on the median values, while direct routing and edgeRouting perform similarly.

In Fig. 8, we can observe that all approaches start with a very similar median at n_2 (about 99 ms). At n_3, we observe that all approaches increase similarly (about 114 ms) due to the increased propagation delay. After that, each approach follows a different trend.

Direct routing increases monotonically but only slightly, with a more defined increase at n_6 (132 ms). This happens due to the increased propagation delay which does not cause considerable changes because the nodes do not reside extremely far from each other. Multihop routing also increases monotonically until n_6 (167 ms), but with a higher rate because the propagation delay includes detours. edgeRouting does not follow this trend, but rather stabilizes until n_6 (107 ms). This happens because the decrease in transmission and queuing delays (due to employing transmissions between compute nodes) is enough to hide the increase of the propagation delay. In multihop routing, there is also similar decrease in transmission and queuing delays, but this not noticeable due to the increased propagation delay of the detours. Thus, we conclude that when sending sensor measurements over 4G Internet, edgeRouting is similar or better than the baselines. The latency reductions reach up to 19% compared to direct routing, and up to 36% compared to multihop routing.

2) Images as Payload

In Figs. 9 and 10, we show the distribution of the latency results when sending images using the cable and the 4G Internet connections. To show the trend of these results, we show the median values alone in Figs. 11 and 12. Figs. 9 and 10 show the latency for each compute node of the system being

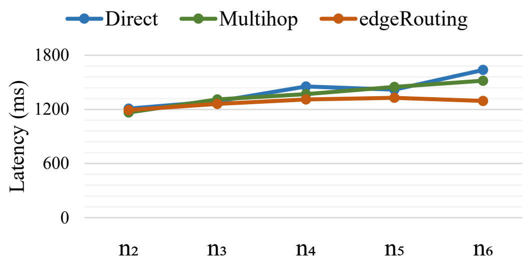


FIGURE 11: Median values of latency from Fig. 9 for each routing approach.

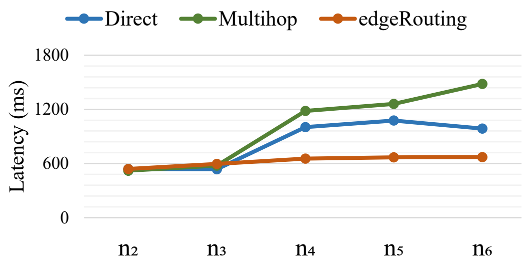


FIGURE 12: Median values of latency from Fig. 10 for each routing approach.

selected as n_X. Figs. 9a, 9b, and 9c show the latency over cable for direct routing, multihop routing, and edgeRouting, respectively. Similarly, Figs. 10a, 10b, and 10c show the latency over 4G for direct routing, multihop routing, and edgeRouting, respectively. Presumably, the latency of n_2 for each routing approach when using the same Internet connection is very similar, and the latency of n_3 is similar between multihop routing and edgeRouting (as discussed in Lemmata 1 and 2).

Based on the interquartile ranges of the latency over cable in Fig. 9, we note that all routing approaches have a very slight tendency to increase. This tendency stems from the increased propagation delay when n_X is farther away. The reason that this increase is very small is that the propagation delay does not contribute significantly to the latency because the routed data are images (i.e., rather large). Thus, the latency is affected more by the transmission and queuing delays. In the case of edgeRouting, the propagation delay is also increased, but this is not very obvious due to the low transmission and queuing delays of the transmissions between compute nodes. Multihop routing also has low transmission and queuing delays but the latency is increased because of the detours.

When using 4G as shown in Fig. 10, the baselines exhibit an increase which is now more defined, while edgeRouting is rather stable. The reason that the increase is more defined, is that the latency is much lower when using 4G. Thus, the same increase in propagation delay has more potential to affect the overall latency. However, this does not affect edgeRouting significantly, because the increased propagation delay is again hidden by the low transmission and queuing delays.

Interestingly, when sending images over cable the results have much more variance than over 4G, which can be observed by looking at the interquartile ranges between Figs. 9 and 10. We believe that the prime contributing factor for this phenomenon is the higher upload bandwidth limit of the 4G network which allows larger volumes of data to be transferred at once, thereby reducing the variance. Another factor could be a potential difference in the network load between cable and 4G. However, since this is not observed when sending sensor measurements (between Figs. 5 and 6), it is more likely the former.

In Fig. 11 which shows the median values of latency over cable (from Fig. 9), we note that the three approaches are very similar at n_2 (about 1,190 ms), and n_3 (about 1,290 ms). After that, direct routing increases the most until n_6 (1,639 ms). Multihop routing also increases but slightly less until n_6 (1,518 ms). edgeRouting remains rather stable until n_6 (1,293 ms). Therefore, when sending images over cable Internet, edgeRouting is similar or better than the baselines with latency reductions that reach up to 21% compared to direct routing, and up to 15% compared to multihop routing.

In Fig. 12 which shows the median values of latency over 4G (from Fig. 10), we observe that all approaches start with a very similar median at n_2 (about 530 ms), and n_3 (about 570 ms). After that, direct and multihop routing increase until n_6 (988 ms and 1,482 ms, respectively). edgeRouting on the other hand increases only slightly until n_6 (671 ms). Hence, when sending images over 4G, edgeRouting is similar or better than the baselines, and the latency reduction is up to 32% compared to direct routing, and up to 55% compared to multihop routing.

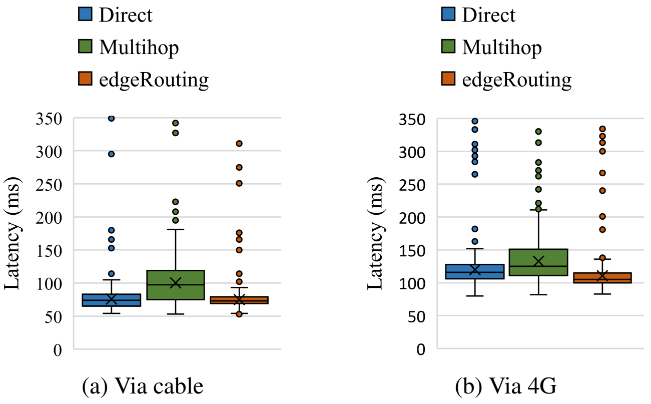


FIGURE 13: Latency of sending sensor measurements to distributed compute nodes for each routing approach.

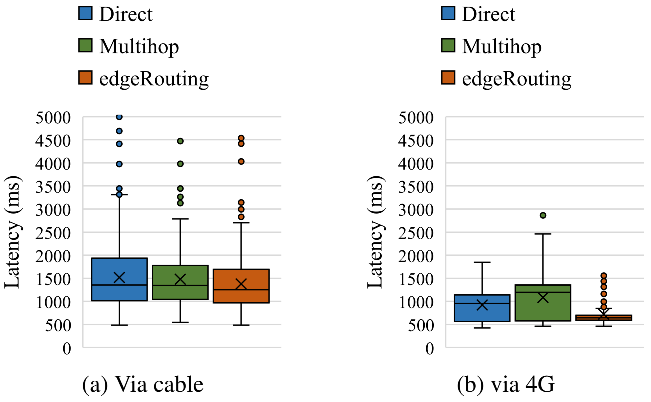


FIGURE 14: Latency of sending images to distributed compute nodes for each routing approach.

3) Overview of the Results

To provide an overview of our results, Fig. 13a shows the distribution of all the latency values of sending sensor measurements over cable. This means that each box plot includes the latency from all compute nodes being used as n_X. Similarly, Fig. 13b shows the latency of sending sensor measurments over 4G, Fig. 14a shows the latency of images over cable, and Fig. 14b shows the latency of images over 4G. The exact values of average, median, and standard deviation of these box plots are shown in Tables 2 and 3.

In Fig. 13a, we note that the median of edgeRouting (73 ms) using a cable connection is very similar to direct routing (74 ms), but approximately 26% less than multihop routing (98 ms). In addition, we note that the upper quartile of edgeRouting (79 ms) is well below the median of multihop routing. This shows that 75% of the edgeRouting values are lower than 50% of the multihop routing values. Fig. 13a also shows that, excluding outliers, multihop routing has the highest latency. This happens because the transmission and queuing delays which are affected by the data size do not contribute a lot to the latency, since the data size is small. Thus, the latency is affected more by the propagation delay. Multihop routing has the highest propagation delay because the data travels to n_X through detours, as discussed in Section IV-B. Direct routing and edgeRouting have similar propagation delays (as discussed in Section IV-C) which is why their latency is also similar.

The results when using 4G shown in Fig. 13b, exhibit the same behavior as cable (in Fig. 13a) with regard to the interquartile range of multihop and edgeRouting. However, the upper quartile of edgeRouting (115 ms) is now lower than the median of direct routing (116 ms). This indicates that 75% of the values of edgeRouting are also lower than 50% of the values of direct routing. This stems from the transmission and queuing delays which are affected by the bandwidth, and are lower in edgeRouting due to the higher bandwidth between compute nodes. The median of edgeRouting (105 ms) is approximately 9% less than direct routing (116 ms), and 16% less than multihop routing (125 ms).

The latency when using 4G (in Fig. 13b) is, overall, higher than when using cable (in Fig. 13a), even though the 4G connection has a higher upload bandwidth limit. This shows that the bandwidth of the Internet provider is not the only factor that affects the incurred latency of sending data to compute nodes. For this reason, routing approaches such as edgeRouting may be able to reduce the communication latency despite the utilized Internet connection. The reason that 4G has higher latency in this case, is likely due to potential higher propagation delay of the paths over the 4G network (since small data sizes are not affected significantly by the transmission delay).

In Fig. 14a which shows the latency of images over cable, the median of edgeRouting (1,252 ms) using a cable connection is approximately 8% less than direct routing (1,356 ms), and approximately 7% less than multihop routing (1,343 ms). Notably, the time reduction of edgeRouting when sending images over cable (about 100 ms reduction in the median compared to direct and multihop routing), is much higher than when sending sensor measurements (1 ms compared to direct routing, and 25 ms compared to multihop routing). The reason for this is that images are larger files which are more likely to be affected by bandwidth limitations. Thus, since edgeRouting leverages the high bandwidth between compute nodes, images are able to be sent faster. In Fig. 14b which shows the results of images over 4G, we note that the median of edgeRouting (644 ms) is approximately 32% less than direct routing (954 ms), and approximately 46% less than multihop routing (1,195 ms). Excluding outliers, the maximum value of edgeRouting (846 ms) is less than the median of both direct routing, and multihop routing. This suggests that all the values of edgeRouting (apart from the outliers) are less than 50% of the values of the baselines.

TABLE 2: Average, median, and standard deviation of the latency values in Fig. 13.

	Cable			4G		
	Direct	Multi.	edgeR.	Direct	Multi.	edgeR.
Average	76	100	75	120	133	111
St. dev.	21	33	21	28	33	30
Median	74	98	73	116	125	105

TABLE 3: Average, median, and standard deviation of the latency values Fig. 14.

	Cable			4G		
	Direct	Multi.	edgeR.	Direct	Multi.	edgeR.
Average	1515	1478	1376	920	1085	723
St. dev.	677	587	584	309	419	263
Median	1356	1343	1252	954	1195	644

Notably, the latency of using 4G to send images (shown in Fig. 14b) is, overall, lower than with cable (shown in Fig. 14a). This can be attributed to the higher upload limit of the 4G connection. Furthermore, we note that edgeRouting performs particularly well compared to the baselines when using 4G (as shown in Fig. 14b). We presume that the reason for this is that the 4G network which is primarily used by mobile devices, may be less loaded at the network edge (i.e., between n_1 and n_2) than cable which is used primarily by stationary users that commonly run more bandwidth-consuming applications such as online gaming [57]. Thus, edgeRouting benefits from using a larger part of the available bandwidth between n_1 and n_2. Direct routing on the other hand, may not benefit from this, because the image is sent from n_1 to n_X, and this path might be more loaded. Multihop routing can also benefit from more bandwidth at the edge, but the overall latency is increased due to the detours. This assumption is also supported by Fig. 10 which shows that the latency of direct routing is rather low for nearby nodes n_2 and n_3 suggesting that the network is not particularly loaded. However, after n_3 there is a steep increase indicating potential bandwidth limitations for direct transmissions to remote nodes. Similarly, multihop routing has low latency for nodes n_2 and n_3, but after that the latency increases due to the accumulated propagation delay of the detours.

In our experiments, we also measure the bit rate of the transmissions based on the routed data (including payload and packet headers). These results align with the presented communication latency results, and can be interpreted in the same manner. For this reason, we do not discuss them explicitly in this evaluation. However, in order to provide a different perspective on the overall performance of each routing approach we note the following: When sending sensor measurements over cable and 4G, direct routing has an average of 3.42 kB/s with a standard deviation of 0.97 kB/s. Multihop routing has an average of 2.89 kB/s with a standard deviation of 0.95 kB/s. edgeRouting has an average of 3.54 kB/s with a standard deviation of 0.87 kB/s. When sending images over cable and 4G, direct routing has an average of 431.06 kB/s with a standard deviation of 198.45 kB/s. Multihop routing has an average of 400.36 kB/s with a standard deviation of 189.04 kB/s. edgeRouting has an average of 499.83 kB/s with a standard deviation of 199.23 kB/s. Presumably, the values of sending sensor measurements are much lower than images because the available network bandwidth of both Internet connections is not fully utilized when sending very small data sizes. This reduces the bit rate compared to utilizing more of the available bandwidth, which happens when sending images.

Hence, we conclude that, overall, edgeRouting provides the lowest latency and highest bit rate in our experiments. There are cases that edgeRouting has similar latency to the baselines, but the distribution of the latency values in general, is reduced. The minimum gains for edgeRouting occur when sending sensor measurements over cable Internet, in which case edgeRouting performs similar to direct routing (but better than multihop routing with 26% reduction). The maximum gains occur when sending images over 4G, in which case edgeRouting reduces the latency by 32% and 46% compared to direct and multihop routing, respectively. Importantly, these outcomes comply with the analysis of Section IV-C, which draws similar conclusions (i.e., Lemmata 1 and 2).

C. DISCUSSION

Since the proposed edgeRouting aims at reducing the communication latency of sending IoT data to edge and cloud compute nodes, our evaluation reflects on that by measuring this latency, and by comparing it to the baselines. However, the implementation of each routing approach utilizes a different amount of computational resources, e.g., for rerouting, which can be considered as overhead. For example, direct routing does not require additional computational resources in compute nodes for rerouting data, because the data are sent directly to n_X. Multihop routing on the other hand, may employ many compute nodes for rerouting (i.e., $n_2, \ldots, n_{X-1}$) until the data reach n_X, while edgeRouting employs exactly one (i.e., n_2). To make sure that the overhead does not compromise the operation of the system, in our experiments we monitor the resource utilization of all the compute nodes. This overhead is found to be negligible in all the examined approaches. Specifically, we use relatively small compute nodes (i.e., with 2 vCPU), and the CPU utilization in all the approaches remains constantly less than 3%, with sporadic spikes which do not exceed 7%.

Furthermore, in systems with edge and cloud compute nodes, the goal is usually to achieve low latency of offloading computations. This includes both the communication latency, and the execution delay of the application [58]. The execution delay can vary based on the specific tasks of an application. For example, performing face recognition tasks on an edge node using common image files (as the images we use in this evaluation), requires about 200 ms [55]. In our experiments, using the OpenCV library and pre-trained Haar cascade classifiers, we also reach this number, with a standard deviation of about 30 ms. Notably, this is significantly lower than the average time needed for sending an image to a compute node, as discussed in Section V-B2. Thus, the communication latency can be a prime factor in the overall latency of offloading computations. This further advocates the importance of approaches that reduce the communication latency—such as the proposed edgeRouting. Additionally, the communication latency may include both upload and download latency, if the application produces a result that needs to be sent back (e.g., to an actuator). Even though the download bandwidth of the Internet provider may be higher than the upload bandwidth (which is the case in our setup), the difference compared to the bandwidth between compute nodes is still significant. Thus, we consider that the interpretation of our results, applies to download latency as well.

As discussed in Section V-A1, in our experiments we use a compute node which resides in Zurich, to emulate the edge compute node n_2. Thus, n_2 is not in such close proximity to n_1 to provide close-to-zero propagation delay. Instead, $Pro_{n_1,n_2} \approx 3$ ms, as mentioned in Section V-A1. This may raise the question of how the presented results would differ, had we used an actual edge compute node. If we had used an edge node as n_2, the latency values of n_2 in all our experiments, would be slightly lower due to a 3 ms lower propagation delay. In direct routing, the rest of the latency values (i.e., to all the compute nodes apart from n_2), would not be affected because Pro_{n_1,n_2} is not part of the communication latency $DLat_{n_1,n_X}$ since the data is sent directly to n_X. This also means that for direct routing, the median values of the overall latency in Figs. 13 and 14 would not be affected, because the median is not affected by a change in the lower values of the data. In multihop routing and edgeRouting, Pro_{n_1,n_2} always affects the communication latency $MLat_{n_1,n_X}$ and $ELat_{n_1,n_X}$, as discussed in Section IV. This means that for multihop routing and edgeRouting, all the latency values of our experiments would be slightly lower due to a 3 ms reduction in Pro_{n_1,n_2}. The median values in this case would also be lowered by the same amount as all the other values. However, since the exact values of the medians are between 73 and 1,343 ms (as shown in Tables 2 and 3), a 3 ms reduction accounts for about 0.2–4% of these values. A slight change in such a small percentage of the values does not have the potential to create considerable error. Thus, we consider that the emulated edge compute node does not affect the results significantly, compared to an actual edge compute node.

Based on our analysis in Section IV, edgeRouting manages to provide similar or lower communication latency than the baselines, under the condition that Pro_{n_1,n_2} approximates zero, as shown in Equation (1). However, the presented results show that the proposed routing approach works well even when Pro_{n_1,n_2} slightly deviates from zero. The reason that our approach works well even though n_2 is not an edge compute node, is that n_2 is significantly closer to n_1 than the other available compute nodes. Thus, we note that edgeRouting performs best when n_2 is an edge compute node in very close proximity to n_1, although benefits can also be observed as long as n_2 is significantly closer to n_1 than the other compute nodes of the system.

It is worth mentioning that even though our system model and evaluation target the specific case of IoT devices sending data to distributed compute nodes of cloud providers, edgeRouting may also be applicable to other similar environments. For example, a recent trend is the utilization of various distributed local nodes which all work together to provide an ad hoc cloud [59]. In such environments, there is usually a cloud connector that connects the ad hoc cloud with actual cloud resources [60]. For this exact communication between a node of the ad hoc cloud, and cloud compute nodes of cloud providers, edgeRouting can be used for reducing the communication latency. In this case, the local ad hoc cloud node becomes the data source n_1, while the other available cloud compute nodes are $n_2, \ldots, n_N$. Therefore, our system model and the presented evaluation can also apply for the communication between an ad hoc cloud and compute nodes of cloud providers.

D. WEAKNESSES AND LIMITATIONS

In Section III-B, we formulate Equations (1), (2), and (3), for delays that do not contribute significantly to the communication latency, and approximate zero. While these equations can apply in the target environment, there is no guarantee that they will always hold, especially since the communication over the Internet can be unpredictable [61], [62]. Therefore, we consider this as a potential weakness in our theoretical analysis of the communication latency in systems with edge and cloud compute nodes. In our experiments, we use the standard 7 Gbps bandwidth limit with external IPs from Google, and an edge compute node which resides in a nearby city. These settings do not opt for a very close approximation to zero in Equations (1), (2), and (3), which could be pursued with utilizing an edge compute node in the same city, and higher bandwidth that can be acquired using internal IPs or a pay-to-use plan for increased bandwidth. Despite not aiming for a close approximation to zero, edgeRouting still exhibits the expected behavior with reduced communication latency over the baselines. This shows that even when taking into account the potential weakness in our theoretical analysis due to the unpredictability of the Internet, edgeRouting still provides significant benefits.

A potential limitation of our work can be with regard to the different routing policies of Internet and cloud providers. These providers may implement their own policies for selecting the network path of every transmission, and each policy may affect the communication latency. In our experiments, we use only the default routing of the providers, because different policies are not configurable by users.

Nevertheless, we consider that edgeRouting can exhibit benefits over direct and multihop routing, independently of the routing policy of the providers, for the following reasons: *i*) In direct routing, the Internet provider is not aware of potential nearby compute nodes of cloud providers, that can be used as detours. Thus, the data cannot be sent through a compute node. For this reason, edgeRouting which preconditions an edge compute node that can be used as a detour, holds an advantage over direct routing, regardless of the routing policy of the Internet provider. *ii*) In multihop routing, sending data through various compute nodes can introduce detours which inevitably increase the propagation delay. Therefore, by avoiding these detours, edgeRouting can provide benefits independently of the routing policy of the cloud provider. There is the possibility that only compute nodes which exist on the network path from n_2 to n_X are used as detours. In this case, edgeRouting and multihop routing perform similarly, with multihop routing having slightly higher latency due to the overhead of examining the data on each compute node on the path (as discussed in Section IV-B).

Another way to affect the selection of routing paths between compute nodes, is by utilizing a load balancer. For example, GCP allows the use of different traffic tiers that can be utilized by a load balancer, for routing the incoming traffic of a compute node to different instances which may reside in different areas. Thus, this technique can be useful when scaling a node by creating multiple instances, because the load balancer can distribute the incoming traffic on these instances automatically [63]. Even though load balancers can affect the routing paths, we do not consider them as competing approaches, but rather as supplementary techniques which can be implemented on top of edgeRouting, for scaling purposes.

VI. CONCLUSION

Since current approaches for routing IoT data to edge and cloud compute nodes may not consider routing paths with increased bandwidth due to detours, in this paper we propose edgeRouting. edgeRouting routes the data through the closest edge compute node which is commonly able to communicate with other compute nodes with very high bandwidth, thereby reducing the communication latency. To support this claim, we analyze the factors that contribute to the communication latency in edgeRouting and in alternative approaches, and we show that edgeRouting provides similar or lower latency. Furthermore, we perform an evaluation using nearby and remote compute nodes, and considering real-world use cases. Our results show that edgeRouting reduces the communication latency by up to 55% compared to the alternatives.

Due to the encouraging results which are particularly promising when routing images over a 4G Internet connection, in the future we plan to adapt our approach to consider compute nodes in base stations for potential integration into 5G networks. This can be very useful for applications that operate over cellular networks and require a lot of bandwidth, such as IoT image processing, augmented reality, and video streaming. Exploring such use cases may also reveal further insights which apply when using different file sizes. In addition, it would be interesting to investigate the efficiency of edgeRouting when applied to computing systems which employ only private edge compute nodes, i.e., that do not belong to a cloud provider. This can provide benefits for use cases related to the concept of ad hoc clouds.

REFERENCES

[1] T. Pasquier, J. Singh, J. Powles, D. Eyers, M. Seltzer, and J. Bacon, "Data provenance to audit compliance with privacy policy in the internet of things," *Personal and Ubiquitous Computing*, vol. 22, no. 2, pp. 333–344, 2018.

[2] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[3] V. Karagiannis and S. Schulte, "Distributed algorithms based on proximity for self-organizing fog computing systems," *Pervasive and Mobile Computing*, vol. 71, p. 101316, 2021.

[4] "Google cloud network," accessed: April 2021. [Online]. Available: https://cloud.google.com/about/locations#network

[5] "Microsoft edge zones," accessed: April 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/networking/edge-zones-overview#edge-zones

[6] V. Karagiannis and A. Papageorgiou, "Network-integrated edge computing orchestrator for application placement," in *International Conference on Network and Service Management (CNSM)*. IEEE, 2017, pp. 1–5.

[7] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, and Q. Li, "A survey of virtual machine management in edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1482–1499, 2019.

[8] V. Karagiannis, "Compute node communication in the fog: Survey and research challenges," in *Workshop on Fog Computing and the IoT (IoT-Fog)*. ACM, 2019, pp. 36–40.

[9] S. A. Noghabi, L. Cox, S. Agarwal, and G. Ananthanarayanan, "The emerging landscape of edge computing," *Mobile Computing and Communications*, vol. 23, no. 4, pp. 11–20, 2020.

[10] Y.-H. Liu, J. Prince, and S. Wallsten, "Distinguishing bandwidth and latency in households' willingness-to-pay for broadband internet speed," *Information Economics and Policy*, vol. 45, pp. 1–15, 2018.

[11] "Google cloud limits," accessed: April 2021. [Online]. Available: https://cloud.google.com/vpc/docs/quota#per_instance

[12] B. R. Kandukuri, R. P. V., and A. Rakshit, "Cloud security issues," in *International Conference on Services Computing (SCC)*. IEEE, 2009, pp. 517–520.

[13] A. Samanta, L. Jiao, M. Mühlhäuser, and L. Wang, "Incentivizing microservices for online resource sharing in edge clouds," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 420–430.

[14] P. Bellavista, L. Foschini, N. Ghiselli, and A. Reale, "MQTT-based Middleware for Container Support in Fog Computing Environments," in *Symposium on Computers and Communications (ISCC)*. IEEE, 2019, pp. 1–7.

[15] J. Yao and N. Ansari, "Fog resource provisioning in reliability-aware IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8262–8269, 2019.

[16] H. Shah-Mansouri and V. W. Wong, "Hierarchical fog-cloud computing for IoT systems: A computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, 2018.

[17] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[18] I. Martinez, A. Jarray, and A. S. Hafid, "Scalable design and dimensioning of fog-computing infrastructure to support latency sensitive IoT applications," *IEEE Internet of Things Journal*, vol. 7, pp. 5504–5520, 2020.

[19] O. Ascigil, T. K. Phan, A. G. Tasiopoulos, V. Sourlas, I. Psaras, and G. Pavlou, "On uncoordinated service placement in edge-clouds," in *International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2017, pp. 41–48.

[20] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, and E. de Lara, "Cloudpath: A multi-tier cloud computing framework," in *Symposium on Edge Computing*. ACM, 2017, pp. 1–13.

[21] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2016, pp. 1–9.

[22] F. Y. Okay and S. Ozdemir, "Routing in fog-enabled IoT platforms: A survey and an SDN-based solution," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4871–4889, 2018.

[23] S. Brennan and M. Rabinovich, "Improving communication through overlay detours: Pipe dream or actionable insight?" in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 1422–1431.

[24] S.-J. Lee, S. Banerjee, P. Sharma, P. Yalagandula, and S. Basu, "Bandwidth-aware routing in overlay networks," in *International Conference on Computer Communications (INFOCOM)*. IEEE, 2008, pp. 1732–1740.

[25] C. Lumezanu, R. Baden, D. Levin, N. Spring, and B. Bhattacharjee, "Symbiotic relationships in internet routing overlays." in *Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX, 2009, pp. 467–480.

[26] P. K. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall, "Improving the reliability of internet paths with one-hop source routing." in *Symposium on Operating Systems Design and Implementation (OSDI)*, vol. 4. USENIX, 2004, pp. 13–13.

[27] C. X. Cai, F. Le, X. Sun, G. G. Xie, H. Jamjoom, and R. H. Campbell, "Cronets: Cloud-routed overlay networks," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 67–77.

[28] M. Satyanarayanan, G. Klas, M. Silva, and S. Mangiante, "The seminal role of edge-native applications," in *International Conference on Edge Computing (EDGE)*. IEEE, 2019, pp. 33–40.

[29] F. Pallas, P. Raschke, and D. Bermbach, "Fog computing as privacy enabler," *IEEE Internet Computing*, vol. 24, no. 4, pp. 15–21, 2020.

[30] W. Shi, G. Pallis, and Z. Xu, "Edge computing [scanning the issue]," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1474–1481, 2019.

[31] "Google cloud regions," accessed: April 2021. [Online]. Available: https://cloud.google.com/about/locations#regions

[32] "Microsoft cloud regions," accessed: April 2021. [Online]. Available: https://azure.microsoft.com/en-us/global-infrastructure/geographies

[33] S. Ilager, R. Muralidhar, and R. Buyya, "Artificial intelligence (ai)-centric management of resources in modern distributed computing systems," in *Cloud Summit*. IEEE, 2020, pp. 1–10.

[34] C. Perera, C. H. Liu, and S. Jayawardena, "The Emerging Internet of Things Marketplace from an Industrial Perspective: A Survey," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 4, pp. 585–598, 2015.

[35] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "Cost-effective processing for delay-sensitive applications in cloud of things systems," in *International Symposium on Network Computing and Applications (NCA)*. IEEE, 2016, pp. 162–169.

[36] V. Karagiannis and S. Schulte, "Comparison of alternative architectures in fog computing," in *International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2020, pp. 19–28.

[37] Y. Zhong, M. Haenggi, F.-C. Zheng, W. Zhang, T. Q. Quek, and W. Nie, "Toward a tractable delay analysis in ultra-dense networks," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 103–109, 2017.

[38] X. Li and X. Zhang, "A concurrent multi-path transmission of data allocation algorithm based on heterogeneous networks," in *International Conference on Systems and Informatics (ICSAI)*. IEEE, 2016, pp. 487–491.

[39] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things," *Transaction on IoT and Cloud computing*, vol. 3, no. 1, pp. 11–17, 2015.

[40] R. Hussain, M. Amini, A. Kovalenko, Y. Feng, and O. Semiari, "Federated edge computing for disaster management in remote smart oil fields," in *International Conference on High Performance Computing and Communications (HPCC)*. IEEE, 2019, pp. 929–936.

[41] P. Kansal and A. Bose, "Bandwidth and latency requirements for smart transmission grid applications," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1344–1352, 2012.

[42] M. Fadhil, G. Owenson, and M. Adda, "Proximity awareness approach to enhance propagation delay on the bitcoin peer-to-peer network," in *International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2411–2416.

[43] "Microsoft cloud limits," accessed: April 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-network/virtual-machine-network-throughput

[44] X. Ji, J. Huang, M. Chiang, and F. Catthoor, “Downlink OFDM scheduling and resource allocation for delay constraint SVC streaming,” in *International Conference on Communications (ICC)*. IEEE, 2008, pp. 2512–2518.

[45] T. Elgamal, A. Sandur, P. Nguyen, K. Nahrstedt, and G. Agha, “Droplet: Distributed operator placement for IoT applications spanning edge and cloud resources,” in *International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 1–8.

[46] V. Karagiannis, “Building a testbed for the internet of things.” Alexander Technological Educational Institute of Thessaloniki, 2014, pp. 1–92.

[47] M. Z. Ahmed, A. H. AbdallahHashim, O. O. Khalifa, and M. J. Salami, “Border gateway protocol to provide failover in multihoming environment,” *International Journal of Information Technology*, vol. 9, no. 1, pp. 33–39, 2017.

[48] N. Akhtar, M. A. Khan, A. Ullah, and M. Y. Javed, “Congestion avoidance for smart devices by caching information in MANETS and IoT,” *IEEE Access*, vol. 7, pp. 71 459–71 471, 2019.

[49] M. Satyanarayanan, “The emergence of edge computing,” *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[50] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino, “Scalability and performance evaluation of edge cloud systems for latency constrained applications,” in *Symposium on Edge Computing (SEC)*. IEEE, 2018, pp. 286–299.

[51] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, “Internet of Things (IoT): Research, Simulators, and Testbeds,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1637–1647, 2017.

[52] M. M. Alam, H. Malik, M. I. Khan, T. Pardy, A. Kuusik, and Y. Le Moullec, “A survey on the roles of communication technologies in IoT-based personalized healthcare applications,” *IEEE Access*, vol. 6, pp. 36 611–36 631, 2018.

[53] “Refit smart home dataset,” accessed: April 2021. [Online]. Available: http://www.doi.org/10.17028/rd.lboro.2070091.v1

[54] F. J. Dian, R. Vahidnia, and A. Rahmati, “Wearables and the internet of things (IoT), applications, opportunities, and challenges: A survey,” *IEEE Access*, vol. 8, pp. 69 200–69 211, 2020.

[55] N. Muslim and S. Islam, “Face recognition in the edge cloud,” in *International Conference on Imaging, Signal Processing and Communication (ICISPC)*. ACM, 2017, pp. 5–9.

[56] “Opencv,” accessed: April 2021. [Online]. Available: https://opencv.org/

[57] J. Gascon-Samson, J. Kienzle, and B. Kemme, “Dynfilter: Limiting bandwidth of online games using adaptive pub/sub message filtering,” in *International Workshop on Network and Systems Support for Games (NetGames)*. IEEE, 2015, pp. 1–6.

[58] J. Liu and Q. Zhang, “Code-partitioning offloading schemes in mobile edge computing for augmented reality,” *IEEE Access*, vol. 7, pp. 11 222–11 236, 2019.

[59] A. J. Ferrer, J. M. Marquès, and J. Jorba, “Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 6, pp. 1–36, 2019.

[60] A. J. Ferrer, J. M. Marques, and J. Jorba, “Ad-hoc edge cloud: A framework for dynamic creation of edge computing infrastructures,” in *International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2019, pp. 1–7.

[61] E. Nygren, R. K. Sitaraman, and J. Sun, “The akamai network: a platform for high-performance internet applications,” *ACM SIGOPS Operating Systems Review*, vol. 44, no. 3, pp. 2–19, 2010.

[62] P. Zhao, W. Yu, S. Yang, X. Yang, and J. Lin, “On minimizing energy cost in internet-scale systems with dynamic data,” *IEEE Access*, vol. 5, pp. 20 068–20 082, 2017.

[63] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, “Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling,” *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 319–334, 2016.

VASILEIOS KARAGIANNIS is with the Distributed Systems Group of TU Wien, Vienna, Austria. He has a BSc from the Alexander Technological Educational Institute of Thessaloniki, Greece, and a MSc from the University of Patras, Greece. Outcomes from his research on cloud computing, edge computing, and Internet of Things have contributed to the publication of various scientific articles and patents.

STEFAN SCHULTE is Associate Professor and head of the Christian Doppler Laboratory Blockchain Technologies for the Internet of Things at the Faculty of Informatics at TU Wien. His research interests span the areas of data engineering, cloud computing, the Internet of Things, and the application and extension of blockchain technologies. Findings from his research have been published in more than 100 refereed scholarly publications.