Computer Communications 194 (2022) 213–225

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Edge computing vs centralized cloud: Impact of communication latency on the energy consumption of LTE terminal nodes

Chiara Caiazza [a,b], Silvia Giordano [c], Valerio Luconi [d], Alessio Vecchio [a,*]

[a] Dip. di Ing. dell'Informazione, Università di Pisa, Largo L. Lazzarino 1, 56122 Pisa, Italy
[b] University of Florence, Italy
[c] Dip. di Tecnologie Innovative, SUPSI, Polo Universitario Lugano–Campus Est, Via la Santa 1, CH-6962 Lugano-Viganello, Switzerland
[d] Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Via G. Moruzzi, 1, 56124 Pisa, Italy

ARTICLE INFO

Keywords:
Edge computing
Energy saving
IoT communication

ABSTRACT

Edge computing brings several advantages, such as reduced latency, increased bandwidth, and improved locality of traffic. One aspect that is not sufficiently understood is to what extent the different communication latency experienced in the edge-cloud continuum impacts on the energy consumption of clients. We studied the energy consumption of a request–response communication scheme when an LTE node communicates with edge-based or cloud-based servers. Results show that the reduced latency of edge servers bring significant benefits in terms of energy consumption. Experiments also show how the energy savings brought by edge computing are influenced by the prevalent direction of data transfer (upload vs download), load of the server, and daytime/nighttime operation.

1. Introduction

With edge computing, processing and storage capabilities are moved from centralized data centers to the periphery of the network, so that they can be close to the end users. The aim of edge computing is to provide low latency and high bandwidth that can be exploited to improve existing applications or to build new ones. The migration of cloud services to the edge can bring benefits in several domains. The reduced latency improves the performance of applications such as online gaming [1], augmented reality [2], and connected vehicles [3]. Real-time video analytics [4] and other traffic-intensive applications can benefit from higher bandwidth. Additionally, the edge computing paradigm allows keeping the traffic local to the access network without the need to transit through the public Internet. This may reduce the load on the core network, especially when bandwidth-intensive applications are considered. In this paper, we use the *edge computing* term according to its broadest meaning: an architectural solution where computing and storage resources are in proximity of the end users, without necessarily assuming that such resources must be co-located with specific elements of the network architecture, such as eNodeB or RAN infrastructure ([5] describes some deployment options). The rationale is that we are interested in evaluating edge computing as an architectural design solution, without specific implementation and/or deployment details.

Both edge and cloud solutions can be beneficial for applications running on smart things [6]. Many smart devices collect data from the environment, examples include the air quality sensors in a smart city [7], intelligent cameras [8], and e-health monitoring appliances [9]. Produced information is transmitted to the server side, where it is aggregated, analyzed, and stored. The server side is typically implemented as a set of cloud services, to cope with fluctuations in computing and storage demand [10–12]. In the last years, also the adoption of the edge computing in combination with smart devices received significant attention [13–15]. Pushing computing, storage, and intelligence closer to the sources of information demonstrated its benefits in several domains, such as manufacturing [16], wearable cognitive assistance [17], and smart transportation [18]. Investigation focused on several aspects of the edge vs cloud relationship, such as service deployment, image and stream processing, and task placement [19–23].

The advantages of edge computing have almost always been quantified in terms of improved response times. An aspect that did not receive enough attention is the impact of edge computing on the energy consumption of client devices, which is definitely important as such devices are typically battery-operated. In this paper, we answer the following research question: how much energy is saved on client devices when using edge computing compared to cloud computing during communication? The study focuses on a request–response protocol, as this is the way of operating of the majority of applications that interact with the cloud (and hence also with the edge).

* Corresponding author.
E-mail addresses: chiara.caiazza@unifi.it (C. Caiazza), silvia.giordano@supsi.ch (S. Giordano), valerio.luconi@iit.cnr.it (V. Luconi), alessio.vecchio@unipi.it (A. Vecchio).

https://doi.org/10.1016/j.comcom.2022.07.026
Received 2 March 2022; Accepted 16 July 2022
Available online 23 July 2022
0140-3664/© 2022 Elsevier B.V. All rights reserved.

Table 1
The list of symbols used.

T_I	The application period (the interval between a request and the next one)
T_{TX}	The time needed to transmit the data to the server
T_{RX}	The time needed to receive the data from the server
T_{ELAB}	The elaboration time of the server
T_W	The time between the end of transmission and the beginning of reception
T_Q	The time elapsed between the end of T_{RX} and the end of the current application period
E_I	The energy consumption during T_I, i.e. during a single request–response cycle
E_{TX}	The energy consumption during T_{TX}
E_{RX}	The energy consumption during T_{RX}
E_W	The energy consumption during T_W
E_Q	The energy consumption during T_Q
T_C	The time spent in CR before moving to SHORT DRX in the absence of packets sent or received
T_S	The time spent in SHORT DRX before moving to LONG DRX in the absence of packets sent or received
T_L	The time spent in LONG DRX before moving to IDLE in the absence of packets sent or received
T_{PROM}	The time needed to promote the interface from IDLE to CR
P_C	The power consumption during T_C
P_S	The power consumption during T_S
P_L	The power consumption during T_L
P_{PROM}	The power consumption during T_{PROM}

The client, which issues the requests, is supposed to be energy-constrained and wirelessly connected, as it happens for many smart and/or mobile devices. The server is supposed to be placed either in the edge or in the cloud, with significant differences in terms of communication latency. The communication between the two application components (client and server) is assumed to be based on LTE, as this is the technology currently used for a significant fraction of machine-to-machine communication. To answer the above research question, this work brings the following technical contributions:

- An analytical model that describes the energy consumption of a request–response protocol is provided. The model, differently from existing literature, allows application designers to understand how the various phases of the protocol, such as transmitting the data or waiting for a reply, affect the overall energy consumption when communicating with an edge server or a cloud one.
- The energy needed on client devices, when communicating with edge- and cloud-based servers, is studied when varying the most important parameters of operations. Results show that edge computing, most of the time, can reduce the energy consumption of client devices. Our results concern a request–response communication scheme, whereas existing literature on the topic mostly covers offloading of computation.
- The analytical model is fed with network traces collected in the real world to incorporate in the evaluation the complexities of the TCP protocol. Results show that edge computing can bring a significant reduction of the energy consumed on the client side, up to 55%, when the request–response communication scheme operates on top of a connection-oriented protocol. Experiments also show how the energy savings brought by edge computing can be affected by the prevalent direction of data transfer (upload vs download), load of the server, and daytime/nighttime operation.

The rest of this paper is organized as follows. The scenario of operation and the communication scheme are presented in Section 2, while the model of the LTE interface is given in Section 3. The energy consumption of the considered communication scheme is computed in Section 4, while results for connectionless and connection-oriented communication are shown in Sections 5 and 6, respectively. Section 7 discusses the main findings and the limitations of this study. Section 8 summarizes the related work. Finally, Section 9 concludes this work. To conclude, Table 1 summarizes the symbols used in the rest of this paper.

2. Scenario of operation and method

We assume a scenario of operation like the one depicted in Fig. 1. The scenario comprises a typical network architecture where an access network provides connectivity to the terminal nodes, and an edge network hosts the edge servers and connects them to the base stations (BSs). The edge network is connected, directly or indirectly, to the public Internet from which remote cloud servers can be reached. Terminal nodes communicate with application components that can be executed either on edge servers or cloud servers. Depending on the placement of the application, various advantages can be achieved. When applications are running on edge servers, communication is generally characterized by reduced latency and higher bandwidth. On the contrary, the use of a remote cloud brings benefits in terms of available computing power and ease of operation. For simplicity, we consider a scenario with just one edge server and one cloud server. Since our aim is to study the energy consumption solely on the client side, and how this energy consumption depends on the latency, a scenario including multiple edge or cloud servers would add unnecessary complexity to the problem, as, from our perspective, different edge or cloud servers are only characterized by different latency towards the client.

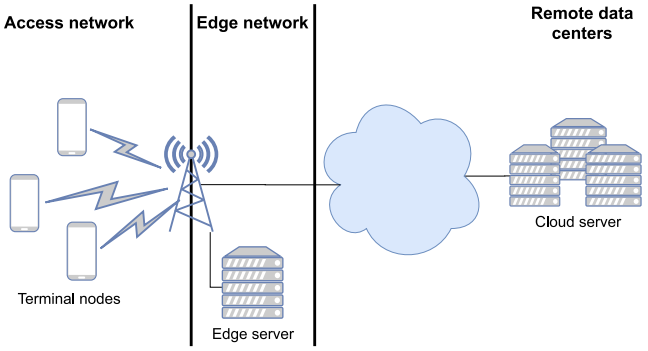


Fig. 1. A modern network architecture including an access network, an edge network, and remote data centers.

In this scenario, we consider a client–server application, represented in Fig. 2. The client (C) runs on terminal nodes, while the server (S) can run either on edge or on cloud machines. The application operates according to a request–response model, whose period is equal to T_I. Such periodic behavior can be found in many classes of applications, e.g., a smart camera that periodically sends the collected data to a central repository or a node that checks for incoming messages or updated information according to a polling-based approach. A Machine Type Communication (MTC) traffic pattern that is of periodic nature is indicated in [24] as one of the two recurrent schemes. The other one is the event-based communication pattern, which is however outside the scope of this paper. At the beginning of each period, C sends a request to the server. This operation requires a time T_{TX} that depends

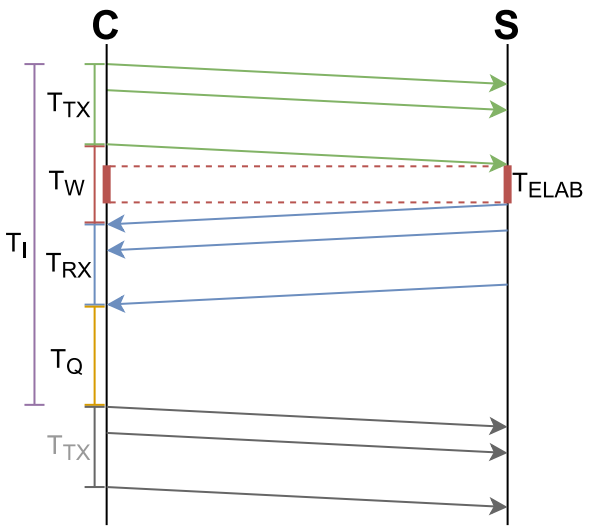


Fig. 2. The interaction between the client (C) and the server (S) during the period T_I.

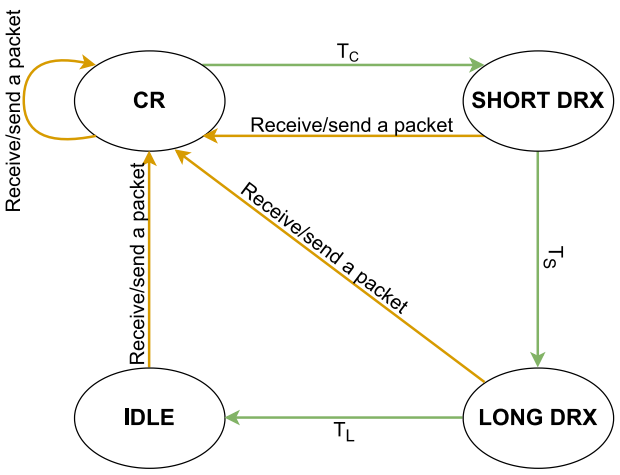


Fig. 3. The finite state machine of an LTE module.

on multiple factors: the amount of data to be transferred, the bandwidth of the wireless segment, and the communication latency. We assume that S is always idle, thus it is always waiting for incoming requests from C, and that the received request is immediately processed in a fixed amount of time T_{ELAB} (in Section 6.4, the impact of an increasing load on the server will be considered). S then sends back the response to C, e.g., a confirmation message that the request has been correctly processed. From the client's perspective, T_W is the interval between the end of the transmission of the last byte of the request and the start of the reception of the first byte of the response. The reception of the response has a duration of T_{RX}. Finally, C remains idle for a time T_Q before starting a new request–response cycle.

2.1. Method

To estimate the energy consumption of the request–response communication scheme, a power model of the LTE interface of client devices is needed. The power model defines the operational states of the interface, the possible transitions between such states, and the power needed in every state. The transition between the operational states is driven by transmissions and reception of packets, or their lack of.

The model is used to study the energy consumption when communication takes place on top of a connection-less protocol or on top of a connection-oriented one. In the first case, the study is purely analytical as the communication pattern is simple and an analytical approach allows us to understand the impact of network conditions (namely the delay), and operational parameters on the different phases of the communication pattern. In the second case, the complexities of transport-level protocols (such as TCP) do not allow a purely analytical approach. Thus, the energy consumption is estimated by feeding the power model of the interface with data extracted from real world experiments.

3. Power model of an LTE interface

The power model of the LTE interface we consider is the one described in [25], both in terms of required power and duration of transitions (the technique was originally proposed in [26]). The interface operates as a Finite State Machine with four states: CR (Continuous Reception), SHORT DRX (Short Discontinuous Reception), LONG DRX (Long Discontinuous Reception), and IDLE. The possible transitions between states are shown in Fig. 3. Overall, the LTE interface stays in the more energy-hungry state CR when it has to send and receive packets, and moves to the most energy efficient state IDLE when there is no traffic. DRX states are intermediate states used for transitioning in less energy-hungry states gradually, as specified in the following.

In detail, every time the interface has to send or receive a packet, it enters the CR state, which is the state with the highest power consumption. Once the transmission is completed, the LTE interface remains in CR for a time T_C, waiting for new packets to be sent or received. If no packet is sent or received during T_C, the interface moves into the SHORT DRX state. In this state, the interface alternates sleep phases and short wake-up periods. The sleeping periods reduce the amount of power consumed by the interface, but during those periods, the interface is unable to receive incoming data. Wake-up periods are used to check for possibly incoming packets. Since the interface is not always ready to receive incoming transmissions, an additional delay may affect the reception of packets. If no packet is sent or received for a time equal to T_S, the interface transitions into the LONG DRX state. Similarly to SHORT DRX, in LONG DRX the LTE interface alternates sleeping periods with wake-up ones. However, in LONG DRX the sleeping periods are longer, trading increased delay during reception of packets for improved energy efficiency. Finally, in the absence of transmissions and reception of packets, after a time T_L, the LTE module moves into the IDLE state. In this state, the interface sleeps most of the time, requiring the smallest amount of energy. Only when a new packet needs to be sent or received the interface returns into the CR state. However, the transition from the IDLE state to the CR state is not immediate, as a "promotion" period is needed to bring the interface into higher operational states. Note that the promotion period is not needed for the transitions from the SHORT DRX and the LONG DRX states to the CR state.

3.1. Parameters of operation

As previously stated, the operational parameters of the LTE interface that we use are the ones provided in [25]. The considered interface has T_C, T_S, and T_L equal to 200 ms, 400 ms, and 11 000 ms, respectively. The power consumed during the sending and the receiving phases depends on the quality of the Reference Signal Received Power (RSRP). We consider a power consumption P_{TX} of 1200 mW for the sending phase and a power consumption P_{RX} of 1000 mW for the receiving phase. In CR, outside any transmission period, a power consumption P_C of 1000 mW is considered. In SHORT DRX, the interface consumes 788 mW during a wake-up time, the duration of a wake-up time is 41 ms, and the interface wakes up with a period of 100 ms. In LONG DRX, a wake-up time consumes 788 mW for 45 ms and the interface wakes up with a period of 320 ms. During the sleeping phases, the

Table 2
The operational parameters of the LTE interface.

Power consumption	Value (mW)
P_{TX}	1200
P_{RX}	1000
P_C	1000
P_S	359.07
P_L	163.23
P_I	14.25
P_{PROM}	1200
Transition times	**Duration (ms)**
T_C	200
T_S	400
T_L	11 000
T_{PROM}	200

energy consumption is equal to 61 mW (for both SHORT and LONG DRX). The wake-up time in IDLE requires 570 mW with a duration of 32 ms, and the period is 1280 ms. The sleep phase in the IDLE state has negligible energy consumption. For simplicity, in SHORT DRX, LONG DRX, and IDLE we calculated the mean power consumption hereafter indicated as P_S, P_L, and P_I. Their values are equal to 359.07 mW, 163.23 mW, and 14.25 mW, respectively. Finally, the LTE promotion requires an average power P_{PROM} equal to 1200 mW, and its duration T_{PROM} is equal to 200 ms. The parameters of the LTE interface are summarized in Table 2.

4. Evaluating the consumption of a request–response communication scheme

The energy E_I needed by the LTE interface during a single iteration of the request–response protocol can be computed as

$$E_I = E_{TX} + E_W + E_{RX} + E_Q + E_{PROM_{TX}} + E_{PROM_{RX}}$$

where E_{TX} is the energy needed to send the data to the server, E_W is the energy spent to wait for a response from the server, E_{RX} is the energy needed to receive the response, E_Q is the energy spent to wait for the beginning of the next iteration, and $E_{PROM_{TX}}$ and $E_{PROM_{RX}}$ are the energy required to bring the LTE interface into CR before T_{TX} and T_{RX} respectively. The energy E_{TX} can be computed as

$$E_{TX} = T_{TX} \cdot P_{TX}$$

Similarly E_{RX} can be computed as

$$E_{RX} = T_{RX} \cdot P_{RX}$$

Note that during T_{TX} and T_{RX}, the interface remains in CR since packets are continuously sent and received (e.g. outgoing TCP segments and incoming ACKs). Instead, when the interface receives no packets, e.g. during T_W and T_Q, it may move towards the states characterized by lower energy consumption.

At the beginning of T_W, the LTE module is in CR, and it will remain in this state for a time T_C before moving to SHORT DRX. This means that the interface will never move from CR if T_W is smaller than T_C. In this case, the energy E_W is equal to

$$E_W = T_W \cdot P_C$$

Instead, if T_W is greater than T_C the interface stays in CR for T_C, and then it enters in SHORT DRX for the residual time. If the residual time is smaller than T_S the interface remains in SHORT DRX and the energy consumption can be computed as

$$E_W = T_C \cdot P_C + (T_W - T_C) \cdot P_S$$

If T_W is larger, the interface remains in SHORT DRX for T_S, and then it goes into LONG DRX for the residual time. If the residual time is shorter than T_L, then E_W can be expressed as

$$E_W = T_C \cdot P_C + T_S \cdot P_S + (T_W - T_C - T_S) \cdot P_L$$

Finally, when T_W is even larger, the interface stays in LONG DRX for T_L and then goes into the IDLE state for the remaining time, and in this case E_W can be expressed as

$$E_W = T_C \cdot P_C + T_S \cdot P_S + T_L \cdot P_L + (T_W - T_C - T_S - T_L) \cdot P_I$$

To summarize, E_W can be expressed as follows:

$$E_W = \begin{cases} T_W \cdot P_C & \text{if } T_W \le T_C \\ T_C \cdot P_C + (T_W - T_C) \cdot P_S & \text{if } T_C < T_W \le (T_C + T_S) \\ T_C \cdot P_C + T_S \cdot P_S + (T_W - T_C - T_S) \cdot P_L & \text{if } (T_C + T_S) < T_W \le (T_C + T_S + T_L) \\ T_C \cdot P_C + T_S \cdot P_S + T_L \cdot P_L + (T_W - T_C - T_S - T_L) \cdot P_I & \text{if } (T_C + T_S + T_L) < T_W \end{cases}$$

T_Q can be expressed as the difference between T_I, the period of operation of the application, and the time needed for a request–response phase:

$$T_Q = T_I - T_{TX} - T_{RX} - T_W - T_{PROM_{TX}} - T_{PROM_{RX}}$$

where $T_{PROM_{TX}}$ and $T_{PROM_{RX}}$ are the time required to bring back the LTE interface into the sending/receiving states, if the LTE interface reaches the IDLE state. Since no packets are sent or received in T_Q, the energy spent during this phase can be computed similarly to E_W. Thus E_Q can be expressed as:

$$E_Q = \begin{cases} T_Q \cdot P_C & \text{if } T_Q \le T_C \\ T_C \cdot P_C + (T_Q - T_C) \cdot P_S & \text{if } T_C < T_Q \le (T_C + T_S) \\ T_C \cdot P_C + T_S \cdot P_S + (T_Q - T_C - T_S) \cdot P_L & \text{if } (T_C + T_S) < T_Q \le (T_C + T_S + T_L) \\ T_C \cdot P_C + T_S \cdot P_S + T_L \cdot P_L + (T_Q - T_C - T_S - T_L) \cdot P_I & \text{if } (T_C + T_S + T_L) < T_Q \end{cases}$$

Since the LTE promotion power P_{PROM} is the same for transmission and reception, as well as the promotion duration T_{PROM} we can state that:

$$E_{PROM_{TX}} = E_{PROM_{RX}} = E_{PROM} = T_{PROM} \cdot P_{PROM}$$

It has to be noted that $E_{PROM_{TX}}$ is consumed only if the LTE interface enters the IDLE state during T_Q, thus if $T_Q > (T_C + T_S + T_L)$. Similarly, $E_{PROM_{RX}}$ is consumed only if the LTE interface enters the IDLE state during T_W, thus if $T_W > (T_C + T_S + T_L)$.

From a different perspective, T_W can be rewritten as

$$T_W = T_{ELAB} + RTT$$

where RTT is the Round Trip Time (RTT) between the client and the server. The RTT value is going to be significantly smaller when the server is executed on the edge compared to when the server is executed in a possibly distant cloud infrastructure. Thus, the energy spent in T_W depends on the RTT. Similarly, also T_Q depends on the RTT, but in this case according to an opposite relationship: the larger the RTT, the smaller T_Q (as mentioned, we assume that the application operates according to a fixed period T_I).

5. Evaluation of connectionless communication

We compared the energy consumption E_I when the client interacts with both a server running on an edge node (hereafter E_I^E) and a server running in the cloud (hereafter E_I^C). In accordance with the scheme depicted in Fig. 2, we consider a simple application where the client uses UDP datagrams to exchange data with a server periodically. Since

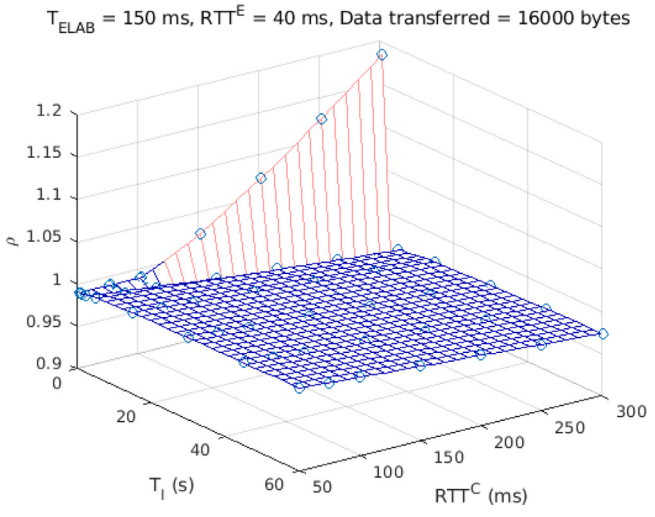


Fig. 4. The ratio (ρ) between E_I^E and E_I^C for different values of RTT^C when varying the period of operation T_I. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 3
E_W, E_Q, and E_I values for both edge and cloud configurations in a connectionless communication scenario, for T_I values of 750 ms and 1000 ms, and RTT^c values ranging from 50 ms to 300 ms. The values have been computed considering T_{ELAB} equals to 150 ms, RTT^e equals to 40 ms, and an amount of data transferred equal to 16 000 B for both B_{TX} and B_{RX}. The configuration with the lower energy consumption is underlined.

T_I (ms)	RTT^C (ms)	E_W^E (mJ)	E_W^C (mJ)	E_Q^E (mJ)	E_Q^C (mJ)	E_I^E (mJ)	E_I^C (mJ)
750	50	190.0	200.0	225.9	222.3	729.5	735.9
750	75	190.0	209.0	225.9	213.3	729.5	735.9
750	100	190.0	218.0	225.9	204.3	729.5	735.9
750	150	190.0	235.9	225.9	162.0	729.5	711.5
750	200	190.0	253.9	225.9	112.0	729.5	679.5
750	250	190.0	271.8	225.9	62.0	729.5	647.4
750	300	190.0	289.8	225.9	12.0	729.5	615.4
1000	50	190.0	200.0	315.6	312.0	819.2	825.6
1000	75	190.0	209.0	315.6	303.1	819.2	825.6
1000	100	190.0	218.0	315.6	294.1	819.2	825.6
1000	150	190.0	235.9	315.6	276.1	819.2	825.6
1000	200	190.0	253.9	315.6	258.2	819.2	825.6
1000	250	190.0	271.8	315.6	240.2	819.2	825.6
1000	300	190.0	289.8	315.6	222.3	819.2	825.6

UDP lacks any rate control mechanisms (such as congestion and flow control mechanisms), the time required to send data depends only on the number of bytes transferred and the interface bitrate. Then, T_{TX} can be computed as

$$T_{TX} = \frac{8 \cdot B_{TX}}{bitrate_{uplink}}$$

where B_{TX} is the number of bytes sent by the client interface within a single period, including the overhead introduced by the underlying network stack levels. We compute T_{TX} and T_{RX} considering an uplink bitrate of 1 Mbps and a downlink bitrate of 0.8 Mbps, respectively. With such low bandwidth bitrates, it is reasonable to assume that the bottleneck is located on the wireless segment of the path between C and S. As a consequence, we assume that packets received during T_{RX} are received back-to-back on the client interface. Then, T_{RX} can be computed as:

$$T_{RX} = \frac{8 \cdot B_{RX}}{bitrate_{downlink}}$$

where B_{RX} is the number of bytes received on the client interface within a single iteration, including the overhead introduced by the underlying network stack levels.

Results are presented by analyzing the ratio $\rho = E_I^E / E_I^C$. When ρ is smaller than 1, the edge-based solution provides benefits in terms of energy consumption. The opposite occurs when ρ is larger than 1. In our analysis, we assume a fixed value for the RTT towards the edge server, RTT^E, equal to 40 ms (this value is derived from a set of measurements described in [27]). In the following sections, we analyze ρ when varying the RTT towards the cloud RTT^C, the period of operation T_I, the amount of sent/received data, and the elaboration time T_{ELAB}, in different combinations.

5.1. Varying the period of operation

First of all, we evaluated the impact of the period of operation of the application on the terminal node. We assume a fixed amount of sent/received data of 16 000 B and a fixed elaboration time T_{ELAB} equal to 150 ms. Fig. 4 shows ρ when varying the application period (T_I), and the RTT towards the cloud server (RTT^C). The red region of the 3D surface corresponds to $\rho > 1$ values. This indicates that, for the combinations of T_I and RTT^C comprised in the red region, the interaction with a remote cloud requires less energy than the edge-based solution. Conversely, the blue region of the curve corresponds to the space of parameters where the edge-based approach is less energy-demanding. As can be observed, when T_I is very small, the amount of energy needed to communicate with a cloud server is lower than the one needed to communicate with the edge server, especially when high RTT^C values are considered. Instead, when T_I becomes larger, the edge configuration becomes more energy efficient than the other one. Let us first understand why, for very small values of T_I, the cloud-based solution is more energy efficient. It has to be noted that E_{TX} and E_{RX} are the same in both edge and cloud configurations. This happens because these values only depend from the transmission and reception bitrate, which we assumed fixed, and the amount of data transferred, which we also assumed fixed. Thus, the only sources of differences between the edge and the cloud configurations are E_W and E_Q, which are influenced by the RTT. The T_W for the edge configuration (hereafter T_W^E) is equal to 190 ms: 150 ms for T_{ELAB}, and 40 ms for the RTT towards the edge (hereafter RTT^E). With this duration, the LTE interface remains in the CR state for the entire duration of T_W^E, as we recall that the LTE interface transitions from the CR state to the SHORT DRX state after 200 ms without sending or receiving a packet. In the cloud configuration instead, RTT^C is higher than RTT^E, thus, for fixed T_I values, T_W^C is higher than T_W^E and T_Q^C is smaller than T_Q^E, especially for high RTT^C values. Let ΔRTT be the difference between the cloud and the edge RTT:

$$\Delta RTT = RTT^C - RTT^E$$

Consequently

$$T_W^C = T_W^E + \Delta RTT$$
$$T_Q^C = T_Q^E - \Delta RTT$$

Differently from the edge configuration, in the cloud configuration, during T_W^C, the LTE interface stays in the CR state for 10 ms more, then it transitions to the SHORT DRX state. This means that, in the cloud configuration, the LTE interface spends most of the additional ΔRTT of T_W^C in a state characterized by relatively low power consumption. Thus the additional amount of energy consumed in T_W^C, with respect to the energy consumed in T_W^E is not so high. For small values of T_I and high RTT^C, T_Q^C lasts only few milliseconds, which, even if LTE interface spends them in the CR state, result in a low amount of spent energy. Instead, in the edge configuration, the LTE interface spends most of the additional ΔRTT of T_Q^E in the CR state, with a high power consumption. This means that the additional energy spent by the cloud configuration in T_W^C is lower than the additional energy spent by the edge configuration in T_Q^E. As an example, Table 3 shows the values

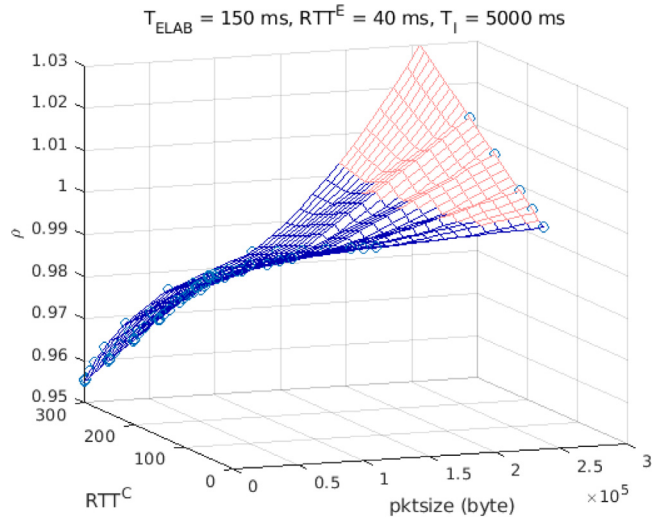


Fig. 5. The ratio (ρ) between E_I^E and E_I^C for different values of RTT^C when varying the amount of transferred data.

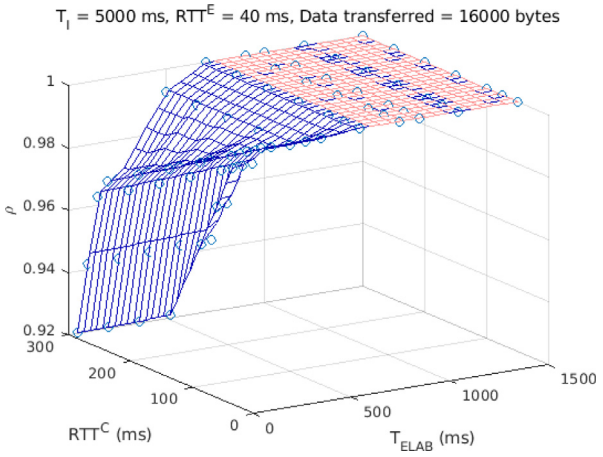


Fig. 6. The ratio (ρ) between E_I^E and E_I^C for different values of RTT^C when varying elaboration time T_{ELAB}.

of E_W, E_Q, and E_I for both the edge and cloud configuration, when varying RTT^C, for two possible values of T_I, 750 ms and 1000 ms, for illustrating the source of differences. For simplicity, for the edge and the cloud configurations we added to all the parameters the superscripts "E" and "C", respectively. This notation will be kept for the rest of the paper. As can be observed, when T_I is equal to 750 ms, for higher RTT^C values, E_Q^C drops down to 12.0 mJ, and E_I^C becomes smaller than E_I^E. Let us now analyze why, for larger T_I values, the edge-based configuration is more energy efficient (the blue area in Fig. 4). When T_I increases there is no impact on T_W, thus E_W remains smaller for the edge configuration. Conversely, in both edge and cloud configurations, T_Q and E_Q increase as the period of operation gets larger. When T_I is equal to 1 s, in the cloud configuration, the residual time T_Q^C is barely sufficient to let the LTE interface enter the SHORT DRX state. This means that for larger T_I the energy spent in T_Q becomes very close for the two configurations and E_W is the component that makes the difference.

5.2. Varying the amount of sent/received data

We now evaluate how the amount of data transferred during requests and responses impacts on the energy needed for the two configurations. Fig. 5 shows the value of ρ when the period of operation is fixed (T_I = 5000 ms), the elaboration time is fixed (T_{ELAB} = 150 ms), the amount of data sent/received during each request/response varies from few bytes to 256 kB, and the cloud RTT, RTT^C, varies from 50 ms to 300 ms. When the amount of transferred data is small, the edge-based configuration is always more energy efficient, especially when RTT^C is high. The reasons for this behavior are similar to the ones described in Section 5.1. Since the RTT^C is higher, also T_W^C is higher. A fraction of the additional time ΔRTT is spent in the CR state, with high power consumption. When the amount of data transferred is small, T_{TX} and T_{RX} are small for both edge and cloud, while T_Q^E and T_Q^C are big enough to let the interface enter the LONG DRX state for both the configurations. This means that the parameter that most influences the energy consumption is T_W, which strongly depends on the RTT. Since the RTT^C is higher, the energy consumption is higher in cloud configurations.

On the contrary, when a higher amount of data is sent/received, T_{TX} and T_{RX} are larger and, as a consequence, T_Q^E and T_Q^C become smaller. When T_Q^E and T_Q^C become so small that the LTE interface is unable to move to the lower energy states (e.g., LONG DRX), then the global consumption becomes more favorable to the cloud-based configuration.

5.3. Varying the elaboration time

We now evaluate the impact of the elaboration time of the server. The value of ρ for different values of T_{ELAB} and RTT^C is shown in Fig. 6. For this analysis, we set T_I = 5000 ms and an amount of sent/received data of 16 000 B. T_{ELAB} is varied between 0 ms and 1500 ms, and RTT^C between 50 ms and 300 ms. The figure shows that the edge configuration is always convenient for small T_{ELAB} values. This happens because small T_{ELAB} values in the edge configuration produce small T_W^E values, which the LTE interface spends in the CR state, but for a small amount of time. In the cloud configuration instead, especially when high RTT^C are considered, T_W^C is much longer than T_W^E, thus the amount of time spent in CR is higher. With T_I = 5000 ms, both T_Q^E and T_Q^C are long enough to make the LTE interface stay in CR for the entire time, and move to less energy demanding states. When T_{ELAB} is small, the total amount of time spent in CR in the cloud configuration is higher than in the edge configuration, and as a consequence also the energy spent is higher.

When T_{ELAB} increases, both T_W^E and T_W^C increase accordingly. For a given T_{ELAB} on, in the edge configuration the LTE interface is able to move into the SHORT DRX state or even in the LONG DRX state during T_W. This decreases the benefits obtained during T_W^E by the edge-based solution. In fact, as can be seen, from a given T_{ELAB} on, choosing either an edge or a cloud configuration does not make any difference from an energy consumption perspective.

5.4. Varying the elaboration time and the application period

Finally, we evaluate the impact of varying the elaboration time T_{ELAB} and the period T_I, while keeping unchanged the cloud RTT (i.e. RTT^C) to 170 ms, and the amount of transmitted data to 16 000 B (Fig. 7). The cloud-based configuration brings benefits only when T_I is really small. For almost all the other configurations, there is almost no benefit in either the edge or the cloud solutions, except for extremely low values of T_{ELAB}. In this latter case, when the server is running in the edge, the energy consumption is lower. This happens because a short T_{ELAB} can reduce the time spent in CR during T_W for both edge and cloud configurations. When the server is in the cloud, in the T_W^C interval, the LTE interface spends a relevant part of ΔRTT in CR, with significantly higher power consumption. Instead, when T_{ELAB} becomes larger than 160 ms, the consumption during T_W for both configurations becomes similar. In this scenario, the difference in energy consumption between the two configurations originates from T_Q. For small T_I values, the cloud-based solution is more energy efficient, as the residual time T_Q^C is smaller than the corresponding time of the edge-based solution, thus less energy demanding.

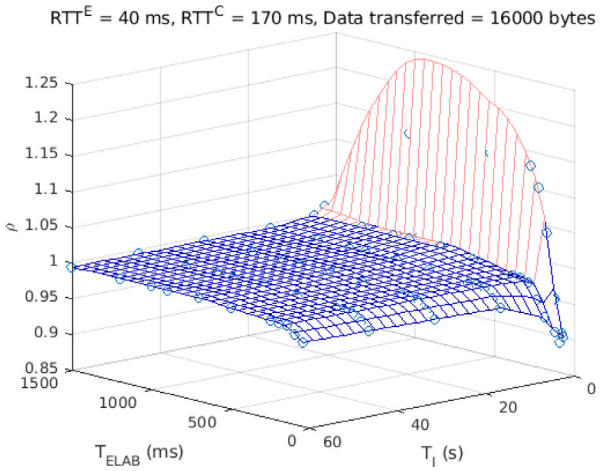


Fig. 7. The ratio (ρ) between E_I^E and E_I^C when varying the elaboration time T_{ELAB} and the period of operation T_I.

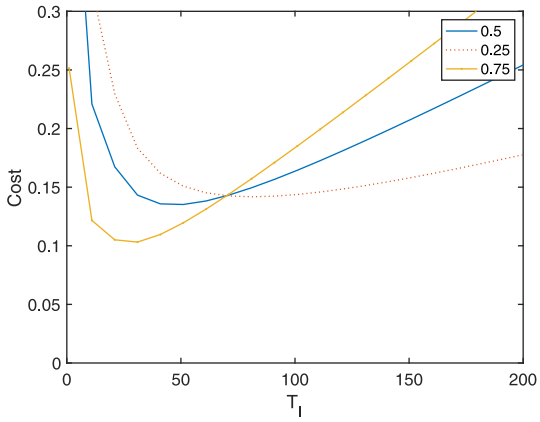


Fig. 8. Communication costs.

5.5. Impact of communication latency on communication costs: an example

Let us suppose that a terminal node produces a given amount of data X every hour. Let us also suppose that communication costs can be defined to take into account not only the energy expenditure of the terminal node, but also the delay that the data incur waiting to be transferred to the server side. Communication cost C could be defined as

$$C = \alpha * \frac{E}{E_{MAX}} + (1 - \alpha) * \frac{D}{D_{MAX}}$$

where E is the total energy needed to transfer the data, and D is the time the data wait on the client side. The latency D is basically the T_I period considered so far. α is a coefficient that can be used to weigh the two components depending on the user's preference for saving energy or obtaining a reduced delay. E_{MAX} is the maximum total energy needed for transferring the data and D_{MAX} is the maximum possible delay. Such maximum values depend on the considered space of parameters and are used to normalize the two cost components before assigning them a weight. Since the total amount of data X in an hour must be always the same, the amount of data that must be included in a single request–response cycle is equal to $X/(3600\ \text{s}/T_I)$. The reply is supposed to be just 1 byte, to confirm the reception of data.

Fig. 8 shows how C changes when varying T_I. The latency between the terminal node and the server is supposed to be equal to 50 ms, and X is equal to 10e6 B. The three curves correspond to values of α equal to 0.25, 0.5, and 0.75. The minimum cost is obtained when the T_I period is approximately equal to 30, 60, and 80 s, depending on the weight given to the two components.

6. Evaluation of connection-oriented communication

In this section, we consider the performance of a TCP-based application. When TCP is involved, the maximum throughput achievable at the steady-state is limited by multiple factors. For example, the flow control mechanism guarantees that the amount of data sent cannot exceed the receiving windows. If a device with poor receiving capabilities is considered, this mechanism can affect the throughput. Instead, if the buffer at the receiver side is sufficient to handle incoming traffic, the steady-state throughput will be determined by the congestion control mechanism.

For different TCP congestion control strategies (e.g., Reno, Vegas, CUBIC), the steady-state throughput has been shown to be always dependent on the RTT [28–31], even if in different forms. The models proposed in the literature are however valid under precise assumptions and contain parameters (e.g., the packet loss probability) whose values can change over time and can only be collected through network measurements. For this reason, to overcome the limitations of an analytical analysis of TCP flows, we conducted a set of experiments aimed at obtaining the values of T_{TX}, T_{RX}, T_W, and T_Q in a real LTE environment. These values are then included in our model to calculate ρ for connection-oriented communications.

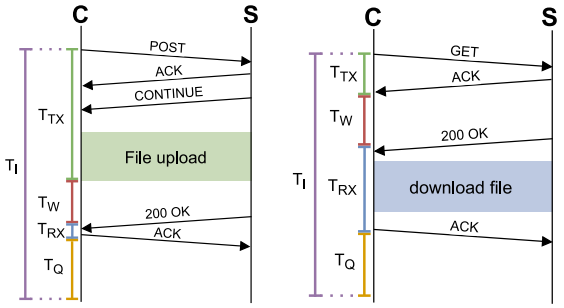
6.1. Application models

According to the scheme of Fig. 2, we consider two applications that periodically exchange data on an existing TCP connection. The first application consists of a client which sends a relatively large amount of data to a server and receives small responses. For example, this behavior can be representative of a smart camera that uploads footage data on a server and receives a confirmation message. The second application implements the opposite scenario, where a client sends small requests and receives larger responses. This second application can be representative of a device that periodically polls a centralized server and receives some updates, as many smart objects or mobile applications do. In practice, these behaviors are mapped on an HTTP POST request and an HTTP GET request, respectively (Fig. 9). In the first application, shown in Fig. 9(a), we consider T_{TX} as the time that goes from the start of the HTTP POST request to the reception of the last ACK from the server. T_{RX} is instead the time that goes from the reception of the HTTP response from the server to the sending of the ACK from the client. In the second application, shown in Fig. 9(b), we consider T_{TX} as the time from the start of the HTTP GET request to the reception of the ACK from the server. T_{RX} is instead the time required to receive the response from the server, which also includes the file transfer, and send the ACK back to the server.

For both applications, we still consider T_W as the time elapsed between the end of T_{TX} and the start of T_{RX}, and T_Q as the residual time required to complete the current operational period T_I.

6.2. Experimental setup

To obtain the values of T_{TX}, T_{RX}, T_W, and T_Q of both edge and cloud configurations in a realistic environment, we conducted a set of experiments using the setup of Fig. 10. We used a Raspberry Pi to host the client application. The Raspberry Pi was chosen to include in the experiments the possible effects caused by a device with limited computational capabilities, as these devices are the subject of this study. The Raspberry Pi is connected to an Android smartphone, which provides LTE connectivity through a USB connection. The client application sends HTTP requests towards an NGINX server hosted at the University of Pisa for both edge and cloud experiments. We observed



(a) Uploading a file to the server. (b) Downloading a file from the server.

Fig. 9. Connection-oriented application models.

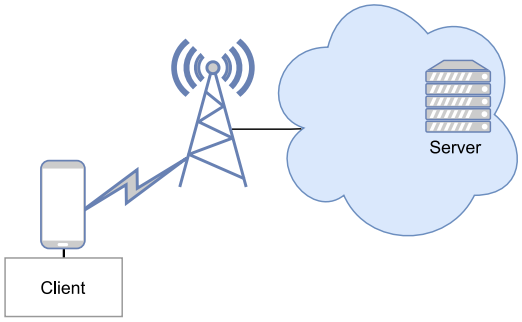


Fig. 10. Experimental setup for the connection-oriented communication evaluation.

approximately 75 ms of average RTT, measured with ping, between the client and the server. Then, to mimic a cloud environment, we used *tc* to add a 100 ms delay on the server interface for both incoming and outgoing traffic. This means that $\Delta RTT = 200$ ms was added for the cloud configuration.

Algorithm 1: The client algorithm executed on the RaspberryPi during the experimental phase.

```
for i ∈ {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} do
    for f ∈ {2, 4, 6, 8, 10, 12} MB do
        for s ∈ [server_EDGE, server_CLOUD] do
            Start the tshark session;
            Send an HTTP POST request to upload on server s a
              file of size f;
            Send an HTTP GET request to download from server
              s a file of size f;
            Stop the tshark session and store the cap results;
        end for
    end for
end for
```

The experiments have been conducted as shown in the pseudo code of Algorithm 1. At the beginning, the client starts a sniffing session between the client and the server using *tshark*. At this point, the client uses *curl* to send to the edge server two HTTP requests. The former is a POST request that implements the behavior of the first application, while the latter is an HTTP GET request that implements the behavior of the second application. Finally, the client stops the sniffing session, storing the collected trace inside a *cap* file for further processing. Then, the procedure is repeated using the cloud server as a target and keeping the size of the uploaded/downloaded file unchanged. Finally, once both edge and cloud metrics have been collected the whole procedure is repeated considering a larger file.

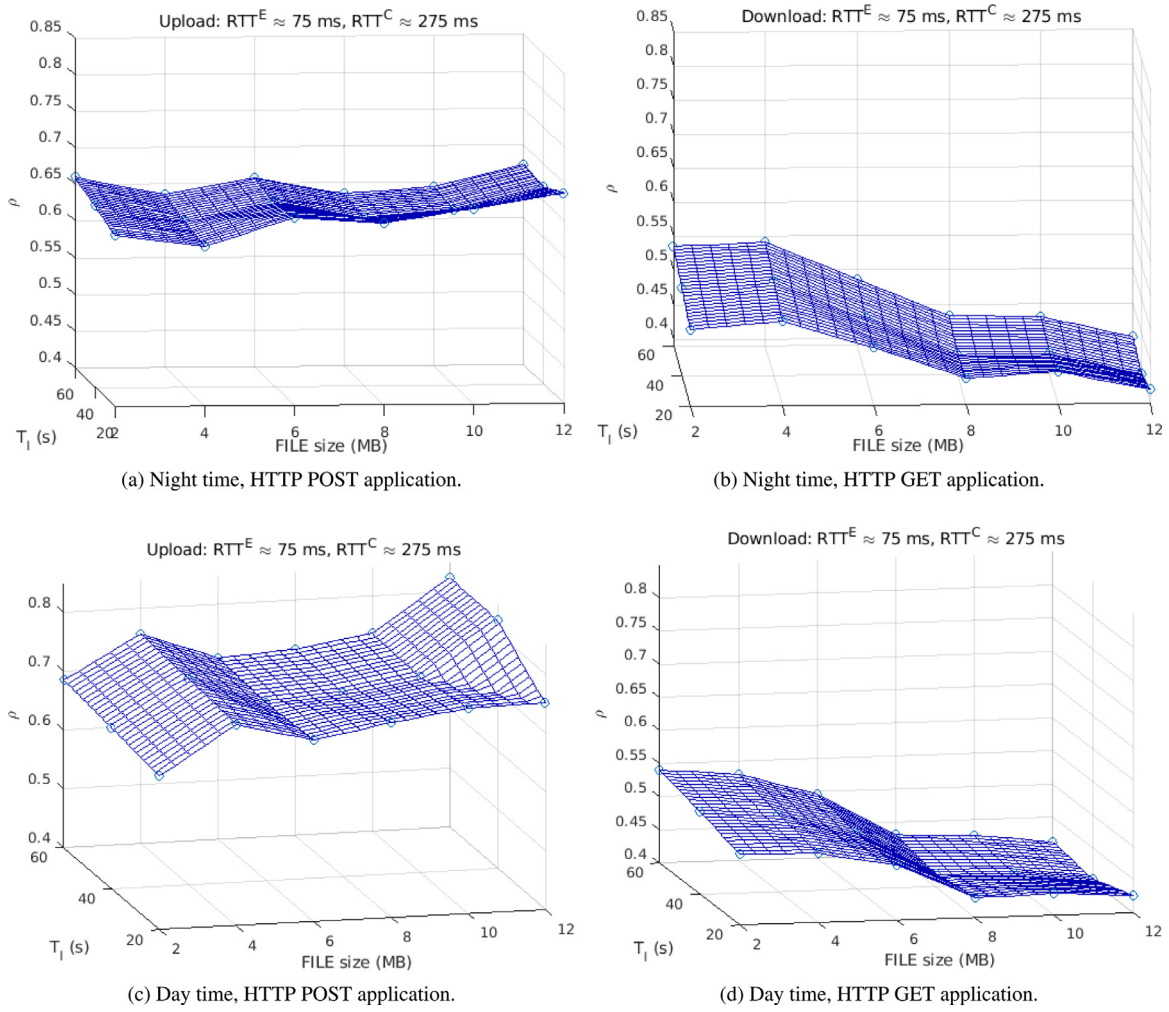
We conducted two experiments. The first experiment was run overnight to collect T_{TX}, T_{RX}, and T_W values in a scenario where the cell is lightly loaded, while the second one was run during the day to collect metrics in the presence of a higher network load. Finally, the values of T_{TX}, T_{RX}, and T_W are extracted from the *cap* file, and T_Q is computed as the difference between T_I and the sum of the other values. These values are then used to compute the corresponding energy values, as explained in Section 4. It must be noted that each HTTP request of the experimental phase was performed independently from the others using curl on different TCP connections, while the models considered for the two applications assume to operate on existing connections. In addition, we did not consider the packets exchanged to open and close the TCP connections.

6.3. Results

For each T_I and each file size, we define E_I^E and E_I^C for a connection-oriented communication as the sum of the energy consumed over the 10 repetitions of each experiment involving the edge and the cloud server, respectively. We consider T_I values from 20 s to 60 s. The ratio ρ between E_I^E and E_I^C is depicted in Fig. 11 for the two applications, and night and day scenarios.

First, we can notice that the value of ρ is always smaller than 1. This means that the edge configuration is always convenient as it has smaller energy consumption. This happens because, as stated above, the throughput of a TCP connection depends on the RTT. Thus, the throughput is higher, and increases faster, when smaller RTT are involved, i.e., when the server is placed on the edge node. This implies a shorter time to upload (or download) a file, and, as a consequence, a shorter amount of time spent in the CR state, the most energy consuming state, by the client LTE interface. Therefore, the energy consumption of the edge configuration is smaller. As a confirmation, in Fig. 12 we show the transmission times for uploading and downloading a file in both edge and cloud, and night and day scenarios. We can observe that the transmission times of the edge configuration are always lower than those of the cloud configuration. However, the difference between the transmission times is not always the same. For example, for the HTTP POST application, in both night time and day time, the transmission time obtained in the edge scenario grows with approximately the same slope as the transmission time obtained in the cloud scenario. For the HTTP GET application instead, the transmission times in the cloud scenario grow much faster than in the edge scenario. This reflects also in the different shapes of surfaces of Fig. 11. For example, for the HTTP POST application, in the night time, the surface is basically flat (Fig. 11(a)). In the day time instead, ρ grows as the file size grows, especially for high T_I times (Fig. 11(c)). This happens because the transmission times of the edge and the cloud scenarios in the day time grow with the exact same slope, while in the night time the edge transmission times grow a bit slower. For the HTTP GET application instead, the trend is similar in both night time and day time (Figs. 11(b) and 11(d)). The differences between the HTTP POST and the HTTP GET applications can be due to the different transmission technologies for downlink and uplink in LTE.

It has to be noted that the values of ρ in all scenarios of Fig. 11 show a trend that differs from the one obtained when a UDP-based application was considered. For example, for each RTT^C considered, Fig. 5 shows that the values of ρ increase as the amount of exchanged data increases. In the connection-oriented model this happens only for the HTTP POST application in the day time (Fig. 11(c)), and still the trend is not the same as in Fig. 5. This can happen for mainly two reasons. Firstly, the connection-oriented experiments use the HTTP protocol implemented by existing tools that may introduce additional overhead on both the server and the Raspberry Pi. Secondly, the TCP protocol provides several mechanisms to adapt the application throughput to the status of both the end-points and the network, while the UDP protocol lacks such mechanisms. As a consequence, the values of T_{TX} and T_{RX}



(a) Night time, HTTP POST application.

(b) Night time, HTTP GET application.

(c) Day time, HTTP POST application.

(d) Day time, HTTP GET application.

Fig. 11. The ratio (ρ) between the E_I^E and E_I^C, when varying the application period T_I and the size of the transferred file.

computed for the connectionless model are independent from the server location, and the differences in the amount of energy consumed are dominated only by small differences in T_W and T_Q. Instead, when a connection-oriented communication is considered, the throughput is affected by the RTT, which in turn depends on the server distance.

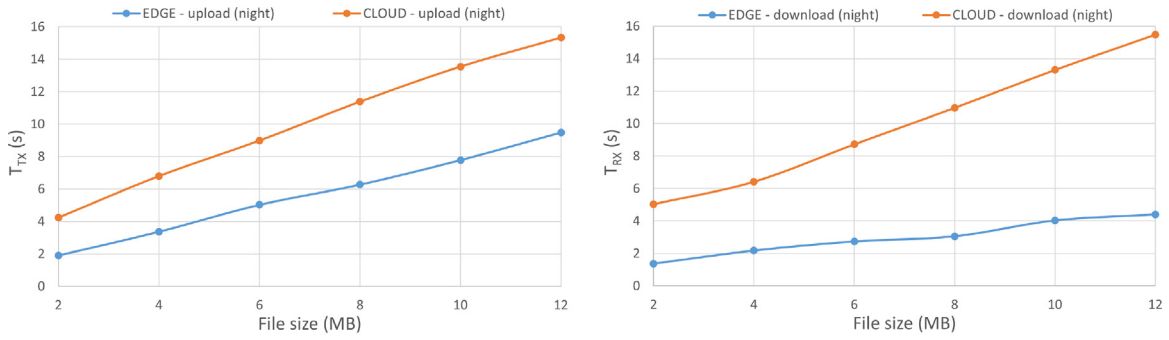
6.4. Evaluating the impact of multiple clients

In this section, we evaluate how the performance of an LTE device running a connection-oriented application can be affected by the presence of other devices interacting with the same server. To this purpose, we performed a set of experiments using the setup of Fig. 13. The specific goal was to collect the T_{TX}, T_{RX}, and T_W values between the monitored client while the server, at the same time, communicates with other clients. The monitored client is the one described in the previous sections. The server side is implemented on top of the *Express* framework and it was hosted at the University of Pisa. The client periodically sends HTTP GET requests toward the Express server, asking for a resource of fixed size (100 kB). An artificial workload was generated using *autocannon* [32]. The latter is a tool used to stress-test Web applications. Autocannon was executed on the same network where the Express server was located, to enable the execution of large numbers of requests. An increasing number of concurrent connections was used to simulate the presence of larger pools of clients. For each concurrent connection, autocannon executed GET requests as fast as possible. Finally, to exert a non-negligible computational load on the server the response was a randomly generated string.

Algorithm 2: The procedure followed during the experiments.

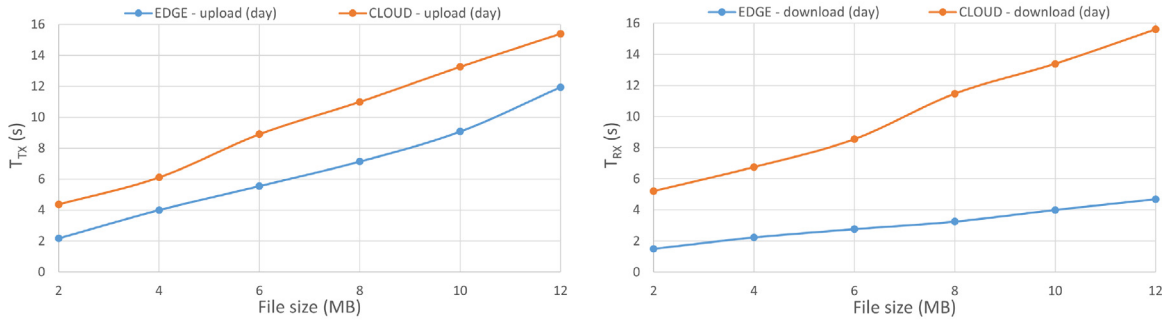
```
for i ∈ {1, 2, 3, 4, 5, 6} do
    for c ∈ {0, 10, 50, 100, 150, 200} do
        Configure the cross-traffic generator;
        for s ∈ [server_EDGE, server_CLOUD] do
            Start the tshark session;
            Send an HTTP GET request to download a resource
              from server s;
            Stop the tshark session and store the cap results;
        end for
    end for
end for
```

The experiments were performed according to the pseudo-code of Algorithm 2. At the beginning of each experiment, the workload generator is activated to send HTTP GET requests using c concurrent connections. Then, the client starts a tshark session and sends an HTTP GET request to download a resource from the server. The collected trace is stored for later processing. The last step is carried out for both the edge and the cloud configuration. Finally, the workload generator is



(a) Night time, HTTP POST application.

(b) Night time, HTTP GET application.



(c) Day time, HTTP POST application.

(d) Day time, HTTP GET application.

Fig. 12. The mean time required to transmit data upon 10 repetitions.

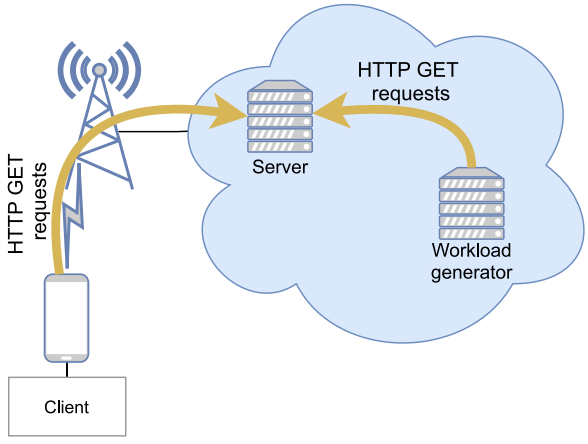


Fig. 13. The setup used to collect T_{TX}, T_W, and T_{RX} values when increasing the load on the server.

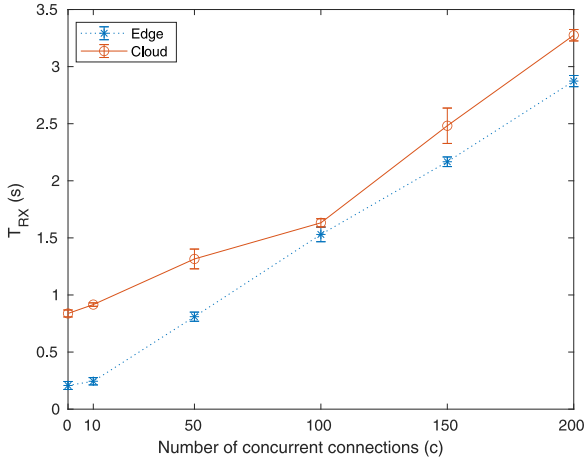


Fig. 14. The mean time required to receive 100 kB considering an increasing server workload and a ΔRTT of 200 ms. The mean values have been computed upon 6 repetitions. The figure shows the 95% confidence intervals.

re-configured to use a higher amount of concurrent connections and the whole procedure is repeated.

Fig. 14 shows the time needed to download 100 kB when varying the server workload. In particular, the curves in Fig. 14 are the average values obtained with 6 repetitions. As can be seen, the maximum difference between edge and cloud performance can be appreciated when there is no HTTP request sent by the cross-traffic generator. Conversely, edge and cloud performance are more similar when a high number of concurrent connections are considered. It should be noted that a high number of concurrent requests may lead to saturation of the links in the proximity of the server. Furthermore, since each response is generated randomly, a high number of requests inflict a high computational load on the server. Hence, this result could indicate that communication latency tends to become less relevant when server resources are running low.

Finally, Fig. 15 show the ratio ρ computed considering different values for the application period T_I and different levels of concurrent connections. As can be seen, ρ values are always lower than one. This indicates that the edge configuration is always the most convenient. Then, we can see that the ρ values are coherent with the T_{RX} values of Fig. 14. In fact, the lowest values of ρ can be appreciated when c is equal to zero. For such a value of c, the values of T_{RX} is significantly different for the two configurations. Conversely, the highest ρ values can be appreciated when c is equal to 100, i.e. when the values of T_{RX} collected for the two configurations are close. However, this result should not surprise since the interface is in CR during T_{RX}. Hence, differences in the time needed to receive the data have a relevant impact on the overall power consumption.

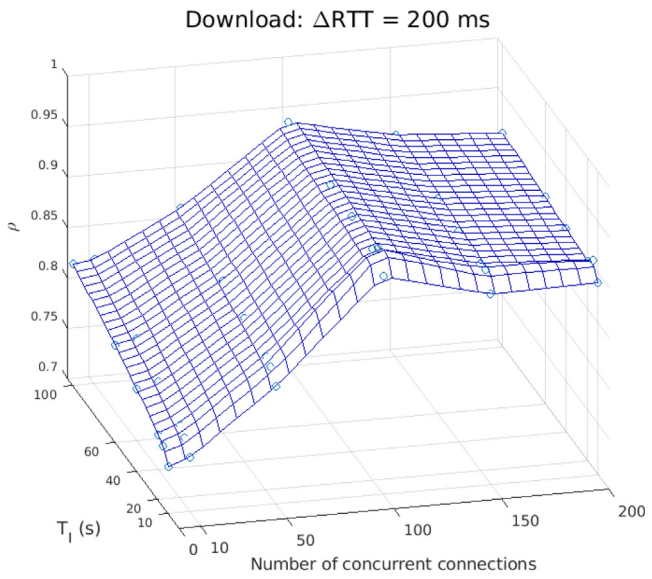


Fig. 15. The ratio (ρ) between the E_I^F and E_I^C, when varying the application period T_I and the number of concurrent connections.

7. Discussion

The results obtained show that, almost always, edge-based solutions allow saving energy on LTE terminal nodes. In particular, results highlight under what network conditions benefits are obtained compared to solutions that rely on a remote cloud. Reducing the energy consumption on terminal devices is a fundamental step for obtaining greener communications and a more sustainable Internet. In addition, in some domains such as Industry 4.0 and Smart Cities, the number of devices belonging to a single organization can be in the order of hundreds or thousands. More efficient communication does not only bring reduced operational costs, but also easier management of battery operated devices. In some cases, the designer just has to push the application server to the edge to achieve some benefits, in other cases some tuning of application operational parameters can be useful (as shown in Section 5.5).

Results have been obtained under a precise set of assumptions, in particular the request–response scheme, the periodic activity, and, in some cases, the use of HTTP as the application level protocol. While these assumptions can be frequently met for the considered scenario concerning smart devices, it is also possible that some applications could be characterized by a highly irregular communication pattern. These aspects have not been included in our model to keep reasonable the level of complexity and understand the main dynamics of the phenomenon. The obtained results pave the way for future work aimed at evaluating the impact of less regular communication patterns – such as when communication is event-driven – and application-level protocols not based on HTTP.

8. Related work

Our work, as mentioned, uses the energy model of the LTE interface defined in [25], which in turn was based on [26]. In particular, in [26] the authors presented a method to infer the operational parameters of an LTE interface: by sending to the device packets with opportunely calibrated intervals and observing the time to respond, the transition times and the duty cycle of DRX modes were estimated. Both the model and the parameters were validated using an hardware power monitor [33], obtaining an error rate below 6%. In [25] the LTE power model was refined and used to estimate the energy expenditure of users in the wild. The goal of the study was to understand how the energy of a smartphone is depleted by its components: CPU, GPU, Wi-Fi, and the cellular interface. Data derived from many users showed that 24.4% of the overall energy was spent on cellular communications. The study did not consider the energy benefits brought by edge computing when compared to centralized cloud, nor evaluated the impact of latency on energy consumption. However, the significant amount of energy spent by a smartphone for cellular communication proves the need for our work.

Other studies focused on finding the optimal DRX parameters, to balance the power-saving constraints with latency requirements. In [34], the problem was faced by using Markov chains, whereas in [35,36] semi-Markov processes were used. Machine learning was used in [37] to predict the optimal sleeping periods of an interface. All these works highlight the importance of reducing the energy needed by an LTE interface, but according to an orthogonal approach compared to our study. Our analysis highlights that the adoption of a network architecture based on edge computing is able to reduce the energy demands of client nodes. The benefits, in terms of energy, brought by edge computing were marginally evaluated in the past, and such benefits can be *added* to the ones obtained by calibrating the LTE interface parameters. In other words, while the above literature studies how the parameters of operation of an LTE interface can be tuned, we evaluated the impact of external elements, in particular the placement of the server the mobile node communicates with.

Some works focus on edge computing and energy consumption. The majority of studies at the intersection of the two topics are related to computation offloading, i.e. moving CPU-intensive tasks from mobile devices to the edge. In fact, smart devices are characterized by limited computational resources and delegating tasks to an edge server can be helpful in many domains, such as authentication [38], e-health [39], and workflow applications [40]. In addition, smart devices are typically battery-powered, thus, offloading part of the computation may bring benefits in terms of required energy. In [41] a technique based on cooperative task offloading and caching in a MEC environment is presented. The technique minimizes the energy consumption while guaranteeing proper security levels. The performance of a three-tier fog network for IoT applications has been studied in [42], considering power consumption, service latency, and CO_2 emission metrics. The overall power consumption has been split into the power consumed to forward packets, perform computation, store data, and migrate applications in the cloud. Task offloading mechanisms for hierarchical edge computing networks have been analyzed in [43], and [44] where the offloading strategy is optimized to take into account both the energy costs and the task execution latency. In hierarchical edge computing networks there are different layers of edge servers, more or less powerful in terms of computational power, and more or less distant from the end users. Other works on the optimal energy-aware offloading schemes in a MEC environment can been found in [45,46]. However, all the above studies are based on the idea of moving part of the computation onto edge nodes. We, on the contrary, do not assume any transfer of tasks from terminal nodes to edge servers. Our study makes clear which benefits can be obtained when communicating with servers at different distances in the latency space. This information could be useful to include a refined model of communication costs, in terms of energy, in many of the studies concerning task offloading. The reader is also forwarded to [47] for a survey about energy aware edge computing, where approaches at different levels are summarized, from specific hardware designs to middleware.

An experimental evaluation of the benefits of offloading computation to edge- and cloud-based servers is presented in [17]. Different applications characterized by different levels of computing requirements were tested. The results show that, in many cases, an edge server can reduce both energy consumption and response time compared to keeping the computation local to the device or using a more distant server in the cloud. Also such work focuses on computation offloading. Our work on the contrary does not assume any offloading of computation. Not only our results are related to pure communication, but also

provide an indication of *why* the edge (or the cloud) provides benefits in terms of needed energy. For the results provided in Sections 5 and 6, we highlighted how energy is spent during the different phases of communication. This, in turn, provides a conceptual framework that allows the application designers to make their applications more energy efficient.

9. Conclusions

The introduction of edge nodes close to the terminal devices provides several advantages such as reduced latency, higher bandwidth, reduced traffic in the core network, and distributed workload among edge and cloud servers. Further, deploying application components on edge nodes may diminish the energy consumption of terminal devices, which generally are battery-powered.

We assessed the energy consumption of a terminal node operating in an edge computing infrastructure. Starting from a FSM-based model of the LTE interface, we computed the energy consumption on the terminal nodes, considering a simple client–server application. We computed energy consumption in a connectionless communication scenario analytically using the FSM, and the energy consumption in a connection-oriented communication scenario by introducing real-world measurements into the model. The results show that the edge server is always convenient in a connection-oriented scenario. This result can be mainly ascribed to the dependency of the TCP throughput from the RTT between the client and the server. When the client interacts with a remote cloud server, the throughput grows slower as it is influenced by the higher RTT experienced between the two end-points. As a result, a longer amount of time is required to send (or receive) the data. This forces the client interface to stay longer in the CR state, which is characterized by high power consumption. On the other hand, the power consumption in a connectionless communication scenario is less affected by the location of the server. The edge-based solution is still generally more favorable, but in some cases the cloud-based one proved to be a better choice. As connectionless communication protocols, such as UDP, lack of any flow and congestion control mechanisms, the time required to send (or receive) some data is independent of the server location. Differences can be found only in idle periods, where no data is transmitted, and the interface may be in a state different than CR with lower energy consumption.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is partially funded by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence). The views expressed are solely those of the authors.

References

[1] X. Zhang, H. Chen, Y. Zhao, Z. Ma, Y. Xu, H. Huang, H. Yin, D.O. Wu, Improving cloud gaming experience through mobile edge computing, IEEE Wirel. Commun. 26 (4) (2019) 178–183.

[2] T. Braud, F.H. Bijarbooneh, D. Chatzopoulos, P. Hui, Future networking challenges: The case of mobile augmented reality, in: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 1796–1807.

[3] F. Giust, V. Sciancalepore, D. Sabella, M.C. Filippou, S. Mangiante, W. Featherstone, D. Munaretto, Multi-access edge computing: The driver behind the wheel of 5G-connected cars, IEEE Commun. Stand. Mag. 2 (3) (2018) 66–73.

[4] J. Wang, Z. Feng, Z. Chen, S. George, M. Bala, P. Pillai, S.-W. Yang, M. Satyanarayanan, Bandwidth-efficient live video analytics for drones via edge computing, in: 2018 IEEE/ACM Symposium on Edge Computing (SEC), 2018, pp. 159–173.

[5] F. Giust, G. Verin, K. Antevski, J. Chou, Y. Fang, W. Featherstone, F. Fontes, D. Frydman, A. Li, A. Manzalini, et al., MEC deployments in 4G and evolution towards 5G, ETSI White Paper 24 (2018) (2018) 1–24.

[6] R. Srinivasa, N.K.S. Naidu, S. Maheshwari, C. Bharathi, A.R. Hemanth Kumar, Minimizing latency for 5G multimedia and V2X applications using mobile edge computing, in: 2019 2nd International Conference on Intelligent Communication and Computational Techniques (ICCT), 2019, pp. 213–217.

[7] G.B. Fioccola, R. Sommese, I. Tufano, R. Canonico, G. Ventre, Polluino: An efficient cloud-based management of IoT devices for air quality monitoring, in: 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a Better Tomorrow (RTSI), 2016, pp. 1–6, http://dx.doi.org/10.1109/RTSI.2016.7740617.

[8] M. Masud, G. Muhammad, H. Alhumyani, S.S. Alshamrani, O. Cheikhrouhou, S. Ibrahim, M.S. Hossain, Deep learning-based intelligent face recognition in IoT-cloud environment, Comput. Commun. 152 (2020) 215–222, http://dx.doi.org/10.1016/j.comcom.2020.01.050, URL https://www.sciencedirect.com/science/article/pii/S0140366419312988.

[9] Z. Yang, Q. Zhou, L. Lei, K. Zheng, W. Xiang, An IoT-cloud based wearable ECG monitoring system for smart healthcare, J. Med. Syst. 40 (12) (2016) 1–11.

[10] P.P. Ray, A survey of IoT cloud platforms, Future Comput. Inf. J. 1 (1) (2016) 35–46, http://dx.doi.org/10.1016/j.fcij.2017.02.001, URL https://www.sciencedirect.com/science/article/pii/S2314728816300149.

[11] H.-L. Truong, S. Dustdar, Principles for engineering IoT cloud systems, IEEE Cloud Comput. 2 (2) (2015) 68–76, http://dx.doi.org/10.1109/MCC.2015.23.

[12] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: a survey, Future Gener. Comput. Syst. 56 (2016) 684–700.

[13] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, D. Soudris, J. Henkel, Computation offloading and resource allocation for low-power IoT edge devices, in: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), IEEE, 2016, pp. 7–12.

[14] R. Morabito, V. Cozzolino, A.Y. Ding, N. Beijar, J. Ott, Consolidate IoT edge computing with lightweight virtualization, IEEE Netw. 32 (1) (2018) 102–111.

[15] W. Shi, S. Dustdar, The promise of edge computing, Computer 49 (5) (2016) 78–81, http://dx.doi.org/10.1109/MC.2016.145.

[16] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, Q. Zhang, Edge computing in IoT-based manufacturing, IEEE Commun. Mag. 56 (9) (2018) 103–109, http://dx.doi.org/10.1109/MCOM.2018.1701231.

[17] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, M. Satyanarayanan, Quantifying the impact of edge computing on mobile applications, in: Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems, in: APSys '16, Association for Computing Machinery, New York, NY, USA, 2016.

[18] B. Qi, L. Kang, S. Banerjee, A vehicle-based edge computing platform for transit and human mobility analytics, in: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, in: SEC '17, Association for Computing Machinery, New York, NY, USA, 2017, http://dx.doi.org/10.1145/3132211.3134446, URL https://doi.org/10.1145/3132211.3134446.

[19] C. Savaglio, G. Campisano, G. di Fatta, G. Fortino, IoT services deployment over edge vs cloud systems: a simulation-based analysis, in: IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2019, pp. 554–559, http://dx.doi.org/10.1109/INFCOMW.2019.8845305.

[20] A. Heredia, G. Barros-Gavilanes, Edge vs. cloud computing: where to do image processing for surveillance? in: J. Dijk (Ed.), Artificial Intelligence and Machine Learning in Defense Applications, Vol. 11169, SPIE, International Society for Optics and Photonics, 2019, pp. 20–32, http://dx.doi.org/10.1117/12.2533380.

[21] P. Silva, A. Costan, G. Antoniu, Investigating edge vs. Cloud computing trade-offs for stream processing, in: 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 469–474, http://dx.doi.org/10.1109/BigData47090.2019.9006139.

[22] P. Nikolaou, Y. Sazeides, A. Lampropoulos, D. Guilhot, A. Bartoli, G. Papadimitriou, A. Chatzidimitriou, D. Gizopoulos, K. Tovletoglou, L. Mukhanov, G. Karakonstantis, On the evaluation of the total-cost-of-ownership trade-offs in edge vs cloud deployments: A wireless-denial-of-service case study, IEEE Trans. Sustain. Comput. (2019) 1, http://dx.doi.org/10.1109/TSUSC.2019.2894018.

[23] A. Luckow, K. Rattan, S. Jha, Exploring task placement for edge-to-cloud applications using emulation, in: 2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC), 2021, pp. 79–83, http://dx.doi.org/10.1109/ICFEC51620.2021.00019.

[24] N. Nikaein, M. Laner, K. Zhou, P. Svoboda, D. Drajic, M. Popovic, S. Krco, Simple traffic modeling framework for machine type communication, in: ISWCS 2013; the Tenth International Symposium on Wireless Communication Systems, 2013, pp. 1–5.

[25] X. Chen, N. Ding, A. Jindal, Y.C. Hu, M. Gupta, R. Vannithamby, Smartphone energy drain in the wild: Analysis and implications, SIGMETRICS Perform. Eval. Rev. 43 (1) (2015) 151–164.

[26] J. Huang, F. Qian, A. Gerber, Z.M. Mao, S. Sen, O. Spatscheck, A close examination of performance and power characteristics of 4G lte networks, in: Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, in: MobiSys '12, Association for Computing Machinery, New York, NY, USA, 2012, pp. 225–238.

[27] C. Caiazza, C. Cicconetti, V. Luconi, A. Vecchio, Measurement-driven design and runtime optimization in edge computing: Methodology and tools, Comput. Netw. 194 (2021) 108140.
[28] M. Mathis, J. Semke, J. Mahdavi, T. Ott, The macroscopic behavior of the TCP congestion avoidance algorithm, SIGCOMM Comput. Commun. Rev. 27 (3) (1997) 67–82.
[29] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: A simple model and its empirical validation, SIGCOMM Comput. Commun. Rev. 28 (4) (1998) 303–314.
[30] N. Cardwell, S. Savage, T. Anderson, Modeling the performance of short TCP connections, Techical Report, Citeseer, 1998.
[31] W. Bao, V.W.S. Wong, V.C.M. Leung, A model for steady state throughput of TCP CUBIC, in: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, 2010, pp. 1–6.
[32] Autocannon: An HTTP/1.1 benchmarking tool, 2022, https://github.com/mcollina/autocannon Accessed: 2022-02-18.
[33] Monsoon power monitor, 2022, https://www.msoon.com/high-voltage-power-monitor Accessed: 2022-02-18.
[34] C. Tseng, H. Wang, F. Kuo, K. Ting, H. Chen, G. Chen, Delay and power consumption in LTE/LTE-A DRX mechanism with mixed short and long cycles, IEEE Trans. Veh. Technol. 65 (3) (2016) 1721–1734.
[35] L. Zhou, H. Xu, H. Tian, Y. Gao, L. Du, L. Chen, Performance analysis of power saving mechanism with adjustable drx cycles in 3GPP LTE, in: 2008 IEEE 68th Vehicular Technology Conference, 2008, pp. 1–5.
[36] Y. Mehmood, L. Zhang, A. Förster, Power consumption modeling of discontinuous reception for cellular machine type communications, Sensors 19 (3) (2019).
[37] P. Brand, J. Falk, J. Ah Sue, J. Brendel, R. Hasholzner, J. Teich, Adaptive predictive power management for mobile LTE devices, IEEE Trans. Mob. Comput. (2020) 1.
[38] P. Hao, X. Wang, Integrating PHY security into NDN-IoT networks by exploiting MEC: Authentication efficiency, robustness, and accuracy enhancement, IEEE Trans. Signal Inf. Process. Netw. 5 (4) (2019) 792–806.
[39] P.P. Ray, D. Dash, D. De, Edge computing for Internet of Things: A survey, e-healthcare case study and future direction, J. Netw. Comput. Appl. 140 (2019) 1–22, http://dx.doi.org/10.1016/j.jnca.2019.05.005, URL https://www.sciencedirect.com/science/article/pii/S1084804519301651.
[40] K. Peng, M. Zhu, Y. Zhang, L. Liu, J. Zhang, V.C.M. Leung, L. Zheng, An energy- and cost-aware computation offloading method for workflow applications in mobile edge computing, EURASIP J. Wireless Commun. Networking 2019 (1) (2019) 207, http://dx.doi.org/10.1186/s13638-019-1526-x.
[41] M.I.A. Zahed, I. Ahmad, D. Habibi, Q.V. Phung, Green and secure computation offloading for cache-enabled IoT networks, IEEE Access 8 (2020) 63840–63855.
[42] S. Sarkar, S. Chatterjee, S. Misra, Assessment of the suitability of fog computing in the context of internet of things, IEEE Trans. Cloud Comput. 6 (01) (2018) 46–59.
[43] Y. Pei, Z. Peng, Z. Wang, H. Wang, Energy-efficient mobile edge computing: Three-tier computing under heterogeneous networks, Wirel. Commun. Mob. Comput. 2020 (2020).
[44] J. Zhang, X. Hu, Z. Ning, E.C.. Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks, IEEE Internet Things J. 5 (4) (2018) 2633–2645.
[45] H. Mazouzi, K. Boussetta, N. Achir, Maximizing mobiles energy saving through tasks optimal offloading placement in two-tier cloud: A theoretical and an experimental study, Comput. Commun. 144 (2019) 132–148, http://dx.doi.org/10.1016/j.comcom.2019.05.017, URL https://www.sciencedirect.com/science/article/pii/S0140366419301720.
[46] X. Li, L. Zhao, K. Yu, M. Aloqaily, Y. Jararweh, A cooperative resource allocation model for IoT applications in mobile edge computing, Comput. Commun. 173 (2021) 183–191, http://dx.doi.org/10.1016/j.comcom.2021.04.005, URL https://www.sciencedirect.com/science/article/pii/S0140366421001419.
[47] C. Jiang, T. Fan, H. Gao, W. Shi, L. Liu, C. Cérin, J. Wan, Energy aware edge computing: A survey, Comput. Commun. 151 (2020) 556–580, http://dx.doi.org/10.1016/j.comcom.2020.01.004, URL https://www.sciencedirect.com/science/article/pii/S014036641930831X.