CLEDGE: A Hybrid Cloud-Edge Computing Framework over Information Centric Networking

Md Washik Al Azad*, Susmit Shannigrahi†, Nicholas Stergiou*, Francisco R. Ortega‡, Spyridon Mastorakis*
*University of Nebraska at Omaha, †Tennessee Tech University, ‡Colorado State University

2021 IEEE 46th Conference on Local Computer Networks (LCN) | 978-1-6654-1886-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/LCN52139.2021.9525006

***Abstract*—In today's era of Internet of Things (IoT), where massive amounts of data are produced by IoT and other devices, edge computing has emerged as a prominent paradigm for low-latency data processing. However, applications may have diverse latency requirements: certain latency-sensitive processing operations may need to be performed at the edge, while delay-tolerant operations can be performed on the cloud, without occupying the potentially limited edge computing resources. To achieve that, we envision an environment where computing resources are distributed across edge and cloud offerings. In this paper, we present the design of *CLEDGE* (*CL*oud + *EDGE*), an information-centric hybrid cloud-edge framework, aiming to maximize the on-time completion of computational tasks offloaded by applications with diverse latency requirements. The design of *CLEDGE* is motivated by the networking challenges that mixed reality researchers face. Our evaluation demonstrates that *CLEDGE* can complete on-time more than 90% of offloaded tasks with modest overheads.**

***Index Terms*—Hybrid Cloud-Edge Computing, Named Data Networking, Mixed Reality**

I. Introduction

Over the last few years, we have witnessed an explosion of the number of Internet of Things (IoT) devices and the amounts of data that these devices produce. The number of IoT devices is expected to grow further in the future, reaching 75 billion connected IoT devices by 2025 [1]. This calls for pervasive edge computing deployments [2], where computing resources are available at the network edge for the low-latency processing of data generated by IoT and other user devices. However, considering the potentially small-scale deployments of computing resources at the network edge, it is critical that computational tasks offloaded by user devices are executed based on the latency they can tolerate by resources located at a proper distance from the users. For example, delay-sensitive tasks $must$ be executed as close to users as possible, while delay-tolerant tasks can be executed further away (possibly on the cloud), ensuring that: (i) the latency requirements of applications that offload the tasks are met; and (ii) resources are utilized efficiently (*e.g.,* delay-tolerant tasks do not occupy resources close to users that may be critical for the execution of delay-sensitive tasks). Furthermore, when computing resources in a particular edge network are fully utilized, user/application-offloaded tasks should be distributed in an adaptive, swift manner to nearby edge networks or a cloud depending on the latency they can tolerate.

To achieve flexible data processing and distribution of tasks, we envision hybrid computing environments where computing resources will be distributed across several edge networks and cloud offerings. In such environments, computing resources of different access latency and capacities will be available to users in a hierarchical manner. At the network edge, limited resources will be available close to users, while a vast amount of resources will be further away on the cloud at the cost of higher communication latency. As a realization of our vision, in this paper, we present *CLEDGE* (*CL*oud + *EDGE*), an Information-Centric framework for hybrid cloud-edge computing. The *CLEDGE* design is motivated by the networking challenges identified through a survey among researchers in the mixed reality community. *CLEDGE* uses Named Data Networking (NDN) to: (i) realize a two-tier, flexible synchronization process for the exchange of resource utilization information among Edge Nodes (ENs) within the same or different edge networks; and (ii) seamlessly distribute offloaded tasks for execution towards ENs within the same or different edge networks or towards cloud offerings.

Our contributions are the following: (i) we motivate the *CLEDGE* design through a survey conducted among mixed reality researchers in order to better understand their networking challenges and requirements; (ii) we present the design of *CLEDGE*, a hybrid cloud-edge framework over NDN, to tackle the challenges of not only mixed reality applications, but also any application that offloads computational tasks with disparate latency requirements; and (iii) we perform an evaluation study of *CLEDGE* and compare its performance with several baseline approaches. Our evaluation results demonstrate that *CLEDGE* seamlessly integrates edge and cloud computing resources. Specifically, *CLEDGE* achieves on-time task completion rates of at least 90% under both light and heavy load conditions with reasonable overheads, outperforming all baseline approaches by 7-78% in terms of on-time task completion rates.

II. Background and Prior Work

A. Named Data Networking

Named Data Networking (NDN) [3] utilizes application-defined hierarchical naming for data publication and communication. Consumer applications send requests for "named data", called *Interest* packets, which are forwarded towards data producers based on their names. Once a producer receives an Interest, it sends back a *Data packet* that is cryptographically signed by its producer and contains the requested content. To forward Interests towards producers and Data packets back to consumers, NDN forwarders maintain three main data structures: (i) a Forwarding Information Base (FIB), which contains entries of name prefixes along with one or more

outgoing interfaces for Interest forwarding; (ii) a Pending Interest Table (PIT), which stores information about forwarded Interests that have not retrieved data yet; and (iii) a Content Store (CS), which caches retrieved Data packets to satisfy future requests for the same data.

B. Cloud and Edge Computing Research

The community has explored cloud computing approaches over NDN, often in combination with other next-generation networking technologies, such as Software Defined Networking and Network Function Virtualization [4]. Named Function Networking (NFN) [5] attempted to utilize lambda expressions to formulate computations and distribute them for execution to computing resources. NFaaS extended the NFN design by placing computing functions in the network and executing them through lightweight virtual machines [6]. RICE augmented the capabilities offered by both NFN and NFaaS to enable consumer authentication and input parameter passing [7]. Amadeo *et al.* [8], Krol *et al.* [9], and Mastorakis *et al.* [10] have explored edge computing frameworks in NDN. Such frameworks can support the execution of programs/tasks in diverse environments, including the edge of the network.

Other approaches have proposed hybrid computing models based on Software Defined Networking (SDN) [11], however, the SDN controller becomes a single point of failure, while approaches to replicate/distribute the controller may result in considerable overheads [12]. Analytical modeling studies of hybrid computing systems have also been conducted [13]. In general, traditional IP-based solutions require complex configuration and maintenance of the communication infrastructure. On the contrary, NDN communication is name based. Nodes and networks can be added or removed transparently to the users without additional configuration or maintenance.

None of the prior works has focused on effectively distributing computational tasks over NDN to available computing resources scattered across edge networks and the cloud with the objective of satisfying the maximum possible number of tasks, each with its own completion deadline. The initial aspiration for *CLEDGE* started from understanding the requirements of a specific target community (*i.e.,* the mixed reality community). In the course of designing *CLEDGE*, we realized that such requirements apply not only to mixed reality applications but, in general, to real- or near real-time applications. To this end, *CLEDGE* can accommodate the requirements of diverse application use-cases, such as smart homes, public safety, industrial control systems, and IoT applications.

III. The Use-Case of Mixed Reality

To better understand the needs of the mixed reality community and align the design of *CLEDGE* with our primary use-case, we conducted a community survey among mixed reality researchers (N=27)[14]. Figure 1a presents the requirements of the mixed reality community in terms of networking. 46% of the participants pointed out that latency is the most critical factor. A "reliable" and consistent latency may be also needed, while having low latency at the beginning of communication followed by higher latency later on is problematic. 38% of the participants pointed out the minimization of packet loss as their primary requirement. However, multiple respondents indicated that for the use cases where latency is important, the reduction of packet loss cannot come at the cost of increased latency. 15% mentioned that guaranteed bandwidth is the most important requirement for their applications.

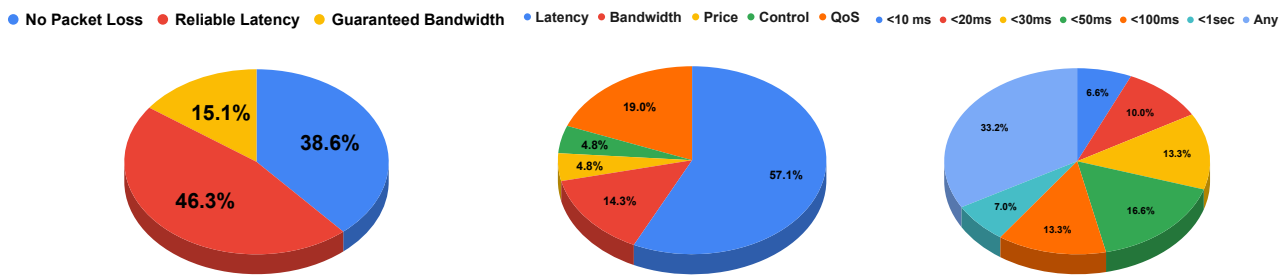
Figure 1b shows the problem areas. 57% of the responses indicated that the network struggles to meet their latency requirements. They mentioned several reasons for higher latency than what their applications can tolerate: high latency towards remote cloud offerings, inconsistent latency over production networks, and more. 19% of the participants suggested that they are concerned about the Quality of Service (QoS), such as low packet loss, guaranteed deadlines for the delivery of processing results, and low jitter. Only 14% of the responses noted a lack of bandwidth as a cause for concern.

Figure 1c illustrates the latency breakdown, as desired by the community. About 46% of the participants require latency below 50ms, while about 60% require latency below 100ms. The participants indicated that achieving sub-100ms latency with cloud computing is unlikely, thus they often deploy and manage computing resources in their local networks. While low latency requirements have been reported previously [15], [16], the surprising finding in this survey was the breadth of use-cases, and how diverse the latency requirements are. Some applications indeed need ultra-low latency (less than 10-30ms). However, other applications (e.g., 2D Augmented Reality) can tolerate up to 100ms or up to a second of latency. The diverse latency requirements of mixed reality applications cannot be accommodated through exclusive edge or cloud offerings. For example, applications that can tolerate up to 50ms of latency are unlikely to operate properly with cloud offerings, where Round-Trip Times (RTTs) may vary from 50ms to 300ms [17].

To address these issues, we propose *CLEDGE*, a hybrid cloud-edge environment that shows promise to fulfil the diverse latency requirements of mixed reality applications. In *CLEDGE*, hierarchically distributed computing resources may be located at different distances from the users: (i) in edge networks either one hop (accessed through direct links, such as LTE/5G) or 2-3 hops away from users; and (ii) on remote clouds. *CLEDGE* enables the execution of computational tasks with diverse latency requirements by finding the appropriate execution locations at the edge or on the cloud based on the latency that the tasks can tolerate.

IV. System Model & Assumptions

We define an edge network as an autonomous network of Edge Nodes (ENs), EN_1, EN_2, ..., EN_n, that offer a set of services (*e.g.,* object recognition, face detection) to users. The ENs are server-class nodes with computing and storage resources. We assume that ENs can be accessed though direct links (*e.g.,* LTE, 5G, WiFi) or links of 2-3 network hops, and that user devices (*e.g.,* mobile phones, AR headsets) are associated with an edge network within their communication range. Applications running on user devices offload computational



(a) Networking requirements identified by mixed reality researchers

(b) Networking challenges identified by mixed reality researchers

(c) Tolerable end-to-end latency as identified by mixed reality researchers

Fig. 1: Mixed reality community survey results

tasks to ENs in the edge network they are associated with by specifying the services to be invoked along with input data. The ENs execute these tasks and return the results to the users.

We assume that multiple edge networks may be available, each administrated by the same or different entities (*e.g.,* service providers). Given that edge computing resources may be limited at any given time, we utilize an adaptive distribution scheme for offloaded tasks. When a user offloads a task, this task will be distributed for execution to available computing resources at an appropriate distance from the user based on the latency that the task can tolerate. Furthermore, when a user offloads a task but no resources are available in an edge network x (*i.e.,* the ENs of x are fully utilized), x can further offload (distribute) the task to a nearby edge network y with available resources. When resources are not available at the edge (*i.e.,* neighboring edge networks do not have adequate resources available), resources on a cloud may be utilized.

Task naming and composition: Following approaches proposed in prior work [7], [12], we represent computational tasks as Interests with the following name format: "`/<service-name>/<input-hash>`". An example of a task name is illustrated in Figure 4a. The first name component of a task specifies the service to be invoked, while the second one refers to a hash of the task input data, which distinguishes tasks for the same service but with different input data. Each task is associated with a deadline by which the edge (or the cloud) needs to execute the task and return the results back to the user (*i.e.,* the delay that the user application can tolerate until it receives the task execution results). This deadline will be attached to the parameters of an Interest [18] in order to leverage in-network caching for tasks with the same input.

Input data of small sizes can be directly attached to the parameters [18] of an Interest [12]. As illustrated in Figure 2, additional Interest-Data packet exchanges may be employed to pass input data of larger sizes (*e.g.,* high-resolution images or video frames) [7]. In such cases, the EN will utilize the forwarding hint of the user device [19] to request the input data from the device. The EN will also send to the device an estimated Time To Completion (TTC) for the offloaded task and a thunk [20], which is a name that will allow the device to reach the particular EN that executes the task after TTC has expired and retrieve the task execution results. A thunk may consist of a concatenation of the name prefix of the EN executing the task and a hash that represents the internal state of execution and identifies the execution of a specific task.

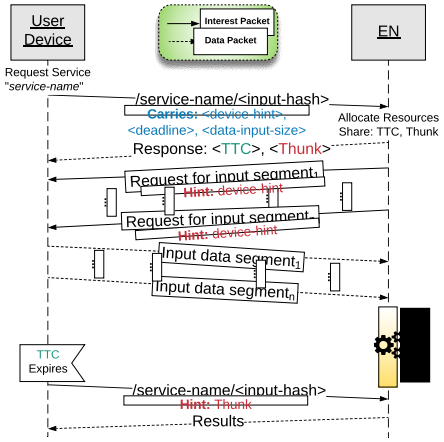


Fig. 2: Example of the task offloading process.

V. Design

A. Design Overview

We present the design of *CLEDGE* through a running example (Figure 3), where multiple edge networks and a remote cloud are available to execute tasks offloaded by users. Edge networks are interconnected through a network of Edge Gateways (EGs). Specifically, one of the ENs in each edge network is designated as an EG. *CLEDGE* features a two-tier synchronization process: the first tier takes place within an edge network and involves the ENs and the EG in this network, while the second tier involves synchronization among the EGs of different edge networks. Through this process, we enable: (i) ENs to be aware of the up-to-date availability of computing resources at all other ENs in their edge network; and (ii) EGs to be aware of the up-to-date availability of computing resources across edge networks. In addition to resource availability, ENs will be able to estimate the RTT to other ENs in the same edge network, while EGs will be able to estimate the RTT to other edge networks.

Our synchronization process is not bound to a specific NDN synchronization protocol, but is able to employ existing NDN synchronization protocols [21], involving two tiers of synchronization to scale up the overall process and mitigate the resulting overhead as the number of edge networks and ENs increases. At the same time, the different synchronization

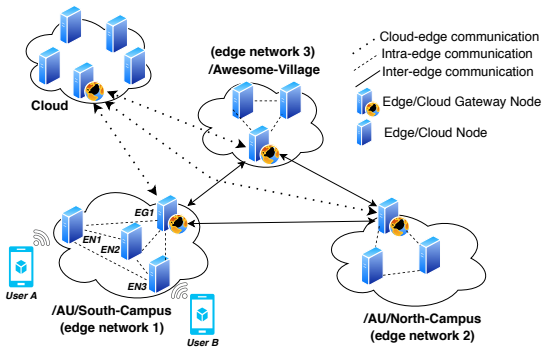
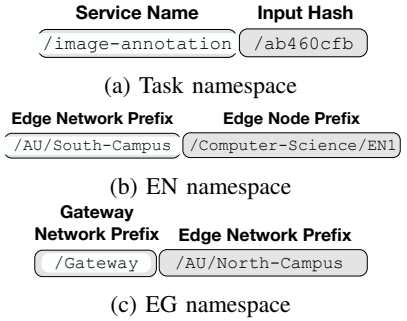


Fig. 3: *CLEDGE* running example. Three edge networks are available with a connection to a remote cloud. One edge network is located on the south campus of the Awesome University (AU), one on the north campus of AU, and one in the awesome village where AU is located.

tiers enable a flexible design, where the synchronization frequency of the first tier is decoupled from the synchronization frequency of the second tier. Moreover, the synchronization frequency of ENs in an edge network can be modified independently of the frequency of other edge networks, effectively adapting to the ongoing operational conditions. For instance, more frequent synchronization can be selected when user traffic and resource utilization change rapidly, while less frequent synchronization can be selected when conditions are stable. We evaluate the impact of the synchronization frequency on the performance and overhead of *CLEDGE* in Section VI.

CLEDGE adaptively distributes offloaded tasks for execution with the objective of maximizing the task satisfaction rate (*i.e.,* the execution results will be returned to users by the deadline of each task). Overall, tasks will be executed as further from (or closer to) the users as their deadlines allow. To meet this goal, *CLEDGE* ensures that delay-sensitive tasks will find available resources at ENs close to users, while tasks that can tolerate additional delay will be executed at ENs further away from users or even on the cloud. For example, once a user offloads a delay-sensitive task to an EN in edge network 1 (Figure 3), the EN tries to execute the task if it has available resources. If this EN has no available resources, it offloads the task to the closest (in terms of RTT) EN with enough resources to satisfy the deadline. If no ENs or the EG in edge network 1 have available resources and the task cannot tolerate execution on the cloud, the task is forwarded through the network of EGs to another edge network with available resources (*e.g.,* edge network 2 or 3 in Figure 3). In the case of a delay-tolerant task, the task is distributed to the furthest EN within an edge network (or even the cloud) that can satisfy the task deadline.

To achieve adaptive and accurate distribution of tasks based on the latency they can tolerate, ENs need to be aware of the network delay to available computing resources as well as the time that these resources may need to execute the tasks. To accomplish that, *CLEDGE* establishes profiles for each service offered in an edge network. These profiles help ENs to estimate how much time each service needs to execute a task. ENs in an edge network exchange such profile



(a) Task namespace

(b) EN namespace

(c) EG namespace

Fig. 4: Namespace design

information (statistics) through the synchronization process within their network, while, optionally, these statistics can also be exchanged among edge networks through the EGs.

B. Namespace Design

Figure 4 illustrates the namespace design in *CLEDGE*. As we mentioned in Section IV, we represent tasks as Interests that adhere to the following naming format: "`/<service-name>/<input-hash>`". The first name component of a task specifies the service to be invoked and the second one refers to a hash of the task input data. These Interests also carry the task completion deadline in their parameters [18], so that tasks for the same input can take advantage of NDN in-network caching. In Figure 4a, we illustrate the name of a task that invokes an annotation service for a certain image as an input.

Each EN has a name under the name prefix of the edge network it belongs to. For example, in Figure 4b, we present the name of an EN on the south campus of the Awesome University (AU) and, specifically, in the Computer Science department. Furthermore, each EG will have a name prefix for communication with other edge networks. For example, for the execution of tasks offloaded on the south campus of AU by computing resources on the north part of the campus, the tasks will be distributed from the EG of the edge network on the south campus towards the EG of the edge network on the north campus through the name presented in Figure 4c.

C. Distribution of Tasks Within an Edge Network

Synchronization process: The first tier of synchronization in *CLEDGE* involves the ENs and the EG within an edge network. This process takes place, so that the ENs and the EG can maintain up-to-date information needed for the adaptive and accurate distribution of offloaded tasks. Such information may include the latest utilization of the computing resources of each EN and the latest statistics about the execution of each service that will be used for the creation and maintenance of the service profiles. During this process, each EN can also measure the RTT to every other EN. Note that an alternative to synchronization may be an event driven model where *CLEDGE* finds an appropriate location of execution as tasks arrive. However, the additional lookup and task placement time may cause problems for latency sensitive tasks.

Building service profiles: Each computational task invokes a certain service. Over time, ENs execute offloaded tasks and

build execution profiles for each invoked service by collecting statistics during the execution of each task. The statistics to be collected can be determined by the network administrators, but metrics of interest may include execution times of tasks (*e.g.,* average values, mean values, 95th percentile, or distributions) per service. To take into account the heterogeneous hardware specifications that ENs may have, service profiles can further include the execution times and the utilization of resources (*e.g.,* required CPU cores, RAM) for different hardware setups. During the synchronization process, ENs share with each other updates on the service profiles they have built. Through such profiles, ENs can estimate the computing resources needed for the execution of tasks for a service and how long the execution of tasks for a service might take.

Task distribution process: Once a task is offloaded from a user application to an EN, the EN estimates whether the task should be executed right away or it should be distributed to an EN further away from the user (lazy task execution). To estimate how far or close to the user a task should be executed, ENs consider the following factors: (i) the task deadline; (ii) the RTT towards other ENs in this edge network; (iii) an estimate of how long the task execution might take based on the profile of the invoked service; and (iv) the availability of computing resources on ENs across the edge network.

For example, in Figure 3, let us assume that user A offloads a task in edge network 1 that reaches EN1 for execution. EN1 will initially use factors (i), (ii), and (iii) to determine whether the task needs to be executed right away or it can be executed by an EN in this network further away from the user (*i.e.,* EN2 and EG1). To avoid resource exhaustion, we avoid distributing offloaded tasks between ENs that are reachable by users through a direct link (*e.g.,* WiFi, LTE/5G), such as EN1 and EN3 in Figure 3, unless there are no other computing resources available in an edge network. Assuming that the task can tolerate to be executed by both EN2 and EG1, but EG1 does not have available resources, EN1 will attach EN2's prefix (*e.g.,* "`/AU/South-Campus/Computer-Science/EN2`" following the namespace design of Figure 4) as the task's forwarding hint, so that the task is forwarded towards EN2. If both EN2 and EG1 have adequate resources, EG1 will be preferred, since it makes computing resources closer to users available for tasks that may not tolerate the network delay towards EG1. In either case, EN1's resources will stay available for tasks that cannot tolerate to be distributed to other ENs for execution.

D. Task Distribution Across Edge Networks and Cloud

Gateway synchronization process: The second tier of synchronization happens among the EGs. Each EG is responsible for synchronizing with other EGs on behalf of the ENs in its own edge network. This process takes place, so that each EG is aware of which edge networks have available resources and estimate how far these resources might be, so that tasks can be distributed from one edge network to another for execution when computing resources are occupied within a network.

During the gateway synchronization process, the information to be synchronized among the EGs can be determined by the edge network administrators. Information of interest may include: (i) the RTT towards the closest EN in each EG's network that has available resources (0 if the EG itself has available resources); and (ii) the utilization of the resources of this EN (or the EG itself). EGs may also optionally exchange aggregated statistics about services invoked within their networks to increase the accuracy of the established service profiles. Through the message exchanges for synchronization, the EGs can measure the RTT to each other.

Task distribution across edge networks: In contrast to the cloud, which may offer an abundance of computing resources, edge networks typically offer limited resources. When an edge network does not have available resources to execute newly offloaded tasks, such tasks will be forwarded to the EG of the network. The EG will determine based on the task deadline, the availability of resources in other edge networks, and the RTT towards these resources, whether the task can tolerate the delay for execution by another edge network or the cloud. Subsequently, a task will be forwarded to the cloud (preferable if the task can tolerate the delay given the abundance of cloud resources) or another edge network with the goal of meeting the task deadline. As a result, tasks can be executed among edge networks based on their resource availability.

For distribution of tasks across edge networks, we utilize forwarding hints. Specifically, the sending EG attaches the name prefix of the destination edge network (or cloud) as the forwarding hint of the original task. The name prefixes of edge networks for communication through the gateways follow the namespace design of Figure 4c. For example, in Figure 3, let us assume that through the synchronization process within edge network 1, ENs are aware that no resources are available in this edge network. In this case, a newly offloaded task will be forwarded by ENs to EG1. EG1 will decide whether this task can tolerate to be executed on the cloud or it needs to be distributed to another edge network for execution. Assuming that EG1 decides to distribute the task to edge network 2, EG1 will attach the prefix "`/Gateway/AU/North-Campus`" as the forwarding hint of the original task.

VI. Evaluation

In this section, we evaluate *CLEDGE* through a simulation study. Our goal is to evaluate: (i) whether *CLEDGE* can successfully meet the completion deadlines of tasks with diverse latency requirements; (ii) the overhead associated with *CLEDGE*; and (iii) whether *CLEDGE* can offer reliable latency to applications for the completion of their tasks.

A. Evaluation Setup

We have implemented *CLEDGE* in ndnSIM [22]. Figure 5 shows the topology and Table I shows the simulation parameters. Each edge network mirrors the topology and setup of edge network 1 in Figure 5. To determine the one-way network latency from users to the cloud, we ran 1000 pings from various locations in the US to Amazon Web Services (AWS) servers in regions around the world using a web tool (https://www.cloudping.info) Our measurements showed that the latency to AWS servers in different US regions varies from

TABLE I: Simulation parameters.

Parameter	Value(s)
Number of edge networks	5
Number of users	100
Number of services	50
Network Stack	NDN directly on top of the MAC layer (IEEE 802.11n for wireless and IEEE 802.3 for wired connections)
Number of tasks that ENs and EGs can execute simultaneously	8
Total number of offloaded tasks per simulation run	500,000
Total simulation runs	10
Task execution times	40%-60% of task deadline (randomly selected)

40ms to 80ms, therefore, we selected the latency between each EG and the cloud to be 50ms (about 60ms from users to the cloud). Each EG is 5 hops away from the cloud, while each user is 8 hops away from the cloud. The latency and hop count values follow values reported in recent studies [23] and cloud computing trends [24].

Each user in our topology randomly selects one of the offered services. Following the conclusions of our survey (Section III), the services are selected from one of the following categories: (i) delay-sensitive services invoked by tasks with deadlines between 10ms and 50ms; (ii) "`regular type`" services invoked by tasks with deadlines between 50ms and 100ms; and (iii) delay-tolerant services invoked by tasks with deadlines between 100ms and 1000ms. Each service is associated with a deadline selected based on its category. For example, a delay-sensitive service s_1 will be associated with a deadline randomly selected between 10ms and 50ms and the tasks that invoke s_1 will have the associated deadline. We experimented with two load profiles: (i) *light load:* each user offloads 2 to 8 tasks per second for a total of about 500 tasks per second; and (ii) *heavy load:* each user offloads 10 to 30 tasks per second for a total of about 2,000 tasks per second. We also implemented a mechanism for two-tier synchronization, where each tier synchronizes periodically and independently of the other. In Section VI-B, we present the average results collected over 10 runs for a total of 500,000 tasks per run.

We compare *CLEDGE* to the following baseline approaches: (i) *cloud-only:* tasks are exclusively offloaded onto the cloud for execution; (ii) *edge-only:* tasks are offloaded to ENs for execution. If an EN does not have available resources, it buffers incoming tasks for later execution in a first come first served manner once resources become available; (iii) *cloud-edge:* tasks are initially offloaded to ENs for execution. If an EN does not have available resources, then the tasks are sent to the cloud for execution; and (iv) *adaptive cloud-edge:* tasks are offloaded to ENs for execution. If an EN does not have available resources, tasks are distributed to the closest (in terms of RTT) EN. The main difference with *CLEDGE* is that adaptive cloud-edge does not consider how much latency a task can tolerate when decisions about the task distribution are made. If no available resources exist within an edge network, tasks will be forwarded through an EG to the closest (in terms of RTT) edge network with available resources. If none of the edge networks has available resources, tasks will be sent to the cloud. Our evaluation metrics include the following:

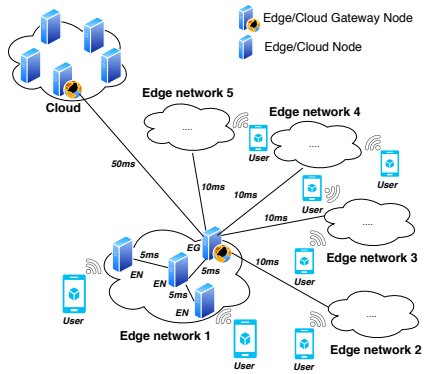


Fig. 5: Evaluation topology. Note that we experimented with varying link delays and numbers of edge networks, services, and users, concluding that the results of these experiments follow the same trend as the results we present in Section VI-B.

- **Task satisfaction rate:** The percent of tasks that are completed on-time (*i.e.,* the tasks are executed and their results are returned to users by their associated deadlines).
- **Normalized overhead:** The volume of traffic generated for the completion of offloaded tasks normalized by the total size of tasks. For *CLEDGE*, the overhead includes the generated traffic for the two-tier synchronization mechanism and the distribution of tasks for execution within an edge network, across edge networks, or towards the cloud. For all other approaches, the overhead includes the traffic for the distribution of tasks for execution by the edge or cloud.
- **Task completion latency:** The average latency for the completion of offloaded tasks.
- **Reliability of task completion latency:** The standard deviation of the completion times of tasks for a certain service from the deadline associated with this service.

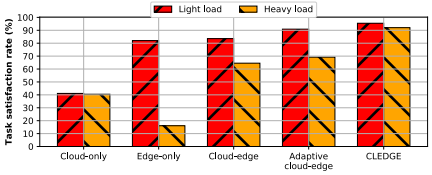
B. Evaluation Results

Task satisfaction rate and overhead: In Figure 6, we present results for the task satisfaction rate and normalized overhead of *CLEDGE* compared to other approaches. Our results indicate that the execution of tasks on the cloud (cloud-only approach) results in low task satisfaction rates and significant overhead, since tasks are forwarded far away from users. Execution of tasks only by the ENs that receive them from users (edge-only approach) results in low overhead under light and heavy loads, low satisfaction rates under heavy loads (the resources of ENs are always fully utilized), and relatively high satisfaction rates for low loads (the resources of ENs are in general available). These results signify the need for a hybrid cloud-edge computing model, since exclusive execution of tasks by the cloud or edge cannot lead to satisfactory on-time completion rates under both low and high loads. Furthermore, cloud-edge and adaptive cloud-edge result in reasonable overhead (between the range of the cloud-only and edge-only overhead), while they achieve relatively high satisfaction rates for light loads and reasonable satisfaction rates for heavy loads.

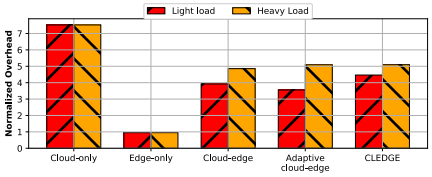
CLEDGE is able to satisfy 95% and 92% of the offloaded tasks for light and heavy loads respectively, achieving 7-78% higher satisfaction rates than the compared approaches.

Specifically, *CLEDGE* satisfies 13% and 7% more tasks than cloud-edge and adaptive cloud-edge respectively for light loads, while, for heavy loads, *CLEDGE* satisfies 28% and 24% more tasks than cloud-edge and adaptive cloud-edge respectively. For light loads, the ENs that directly receive offloaded tasks from users in general have available computing resources to execute these tasks. However, under heavy loads, the computing resources of ENs are in general fully utilized, therefore, *CLEDGE* can successfully distribute tasks based on their deadlines to available resources within the same or different edge networks or onto the cloud.

In terms of overhead, *CLEDGE* achieves reasonable overheads in the range between cloud-only and edge-only. Specifically, *CLEDGE*'s overhead is marginally higher (about 2-4%) than cloud-edge and adaptive cloud-edge for high loads, while, for light loads, *CLEDGE* results in about 11% and 17% higher overheads than cloud-edge and adaptive cloud-edge respectively. This is attributed to the fact that *CLEDGE* aims to provide reliable latency to applications with diverse latency requirements under network and resource loads that may rapidly change. To this end, it ensures that a part of the computing resources of ENs close to users will be available to execute latency-sensitive tasks that may be received in the future. This is achieved at the price of forwarding tasks that can tolerate latency to resources further away from users.



(a) Task satisfaction rate

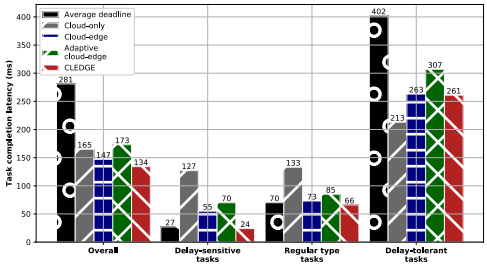


(b) Normalized overhead

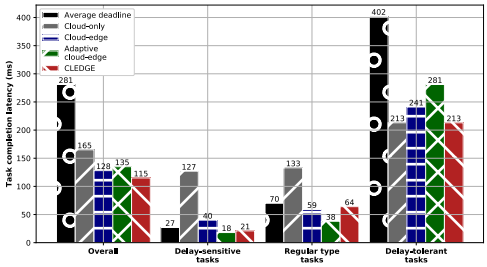
Fig. 6: Task satisfaction rate and overhead results.

Task completion latency: In Figure 7, we present the average completion latency for all tasks and for each service category. Our results demonstrate that *CLEDGE* is the only approach that achieves completion times lower than the deadlines of tasks associated with services of all the different categories. This indicates that *CLEDGE* can successfully distribute tasks for execution to available computing resources based on their deadlines, being able to satisfy categories of tasks/services with diverse latency requirements. *CLEDGE* also achieves the lowest overall task completion latency (*i.e.,* average completion time among tasks of all categories) and meets the latency requirements of task categories that other approaches cannot (*e.g.,* delay-sensitive and regular type tasks under heavy loads).

Reliability of task completion latency: In Figure 8, we show results on the reliability of the task completion latency for *CLEDGE* and adaptive cloud-edge for a sample time interval of 10 seconds during our experiments under heavy load.



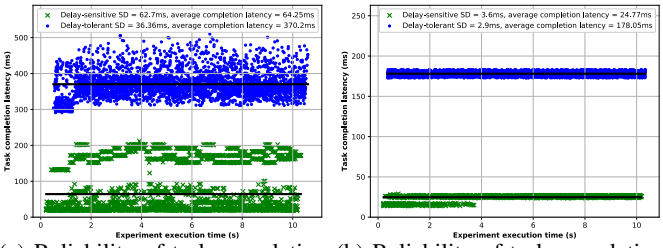
(a) Task completion latency (heavy load)



(b) Task completion latency (light load)

Fig. 7: Average task completion latency. The results for edge-only are significantly high, thus they are omitted to improve readability. The "overall" results include the average of the results of delay-sensitive, regular, and delay-tolerant tasks.

Note that we present results for a selected delay-sensitive and delay-tolerant service. We have verified that these results are representative of the results for all other services of the same nature. Our results indicate that *CLEDGE* achieves consistent latency for both delay-sensitive and delay-tolerant tasks with a minimal standard deviation of 3.6ms and 2.9ms respectively. On the other hand, adaptive cloud-edge results in inconsistent latency with a standard deviation of 62.7ms and 36.36ms for delay-sensitive and delay-tolerant tasks respectively. We verified that all other approaches result in inconsistent latency and follow the same trend as adaptive cloud-edge.



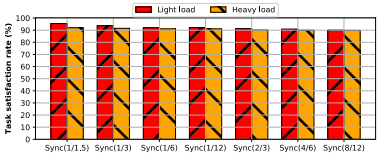
(a) Reliability of task completion latency for adaptive cloud-edge (b) Reliability of task completion latency for *CLEDGE*

Fig. 8: Reliability of task completion latency for a sample time interval (10 seconds) under heavy load. Results for completion latency and Standard Deviation (SD) are presented for the tasks of a delay-sensitive and delay-tolerant service.

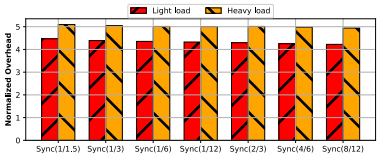
Distribution of executed tasks: Through our evaluation, we were able to identify whether offloaded tasks were executed at the edge or on the cloud. We further identified whether the tasks were executed in the same or a different edge network than the one they were offloaded onto. Our results indicated that *CLEDGE* executed 41-44% of the tasks within

their home edge networks (*i.e.,* the same edge networks the tasks were offloaded onto), 11-15% across different edge networks, and 41-48% on the cloud. This is attributed to the fact that *CLEDGE* distributed delay-sensitive tasks to computing resources within their home edge networks, while tasks were distributed to neighboring edge networks only when their home edge networks did not have available resources and the tasks' deadlines did not allow for execution on the cloud.

Impact of synchronization frequency: In Figures 9a and 9b, we present results on the impact of the frequency of the two-tier synchronization process on the task satisfaction rate and the overhead of *CLEDGE*. Our results indicate that the synchronization frequency does not have a major impact on these metrics under both light and heavy loads. As synchronization becomes less frequent, the satisfaction rate and the overhead decrease by 5%. Since the light and heavy load profiles do not include rapid task offloading rate changes, the results demonstrate that the synchronization period can be relatively long (in the order of several seconds). We further performed experiments where users continuously switched between light and heavy loads, signifying that the synchronization frequency can impact the satisfaction rate when rapid changes of the utilization of the resources happen. In such cases, ENs may send an explicit notification to synchronize with others when they detect a rapid change of their resource utilization.



(a) Synchronization frequency impact on *CLEDGE*'s satisfaction rate



(b) Impact of synchronization frequency on *CLEDGE*'s overhead

Fig. 9: Impact of the two-tier synchronization frequency on the task satisfaction rate and overhead. The notation $Sync(x/y)$ denotes that synchronization takes place within edge networks every x seconds and among EGs every y seconds.

VII. Conclusion and Future Work

In this paper, we presented *CLEDGE*, an NDN framework for hybrid cloud-edge computing. Its design was motivated by the requirements of the mixed reality community for the execution of tasks with diverse deadlines along with reliable response times. Our evaluation indicated that *CLEDGE* can provide adaptive distribution of tasks based on the latency they can tolerate to available computing resources within the same or different edge networks, and towards the cloud.

In our future work, we will investigate the following directions: (i) sophisticated synchronization mechanisms that provide beneficial trade-offs between task satisfaction rates and overheads; (ii) utilize *CLEDGE* to facilitate the operation of a wide range of applications; and (iii) implement a *CLEDGE* prototype, conduct a real-world evaluation study of its design, performance, and scalability.

Acknowledgements

This work is partially supported by National Science Foundation awards CNS-2016714, CNS-2104700, OAC-2019163, and OAC-2019012, the National Institutes of Health (NIGMS/P20GM109090), the Nebraska University Collaboration Initiative, and the Nebraska Tobacco Settlement Biomedical Research Development Funds.

References

[1] Cisco: The future of IoT miniguide: The burgeoning IoT market continues. https://www.cisco.com/c/en/us/solutions/internet-of-things/future-of-iot.html, 2019.

[2] Weisong Shi et al. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[3] Lixia Zhang et al. Named data networking. *ACM SIGCOMM Computer Communication Review*, 44(3):66–73, 2014.

[4] Ravishankar Ravindran et al. Towards software defined icn based edge-cloud services. In *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, pages 227–235. IEEE, 2013.

[5] Manolis Sifalakis et al. An information centric network for computing the distribution of computations. In *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014.

[6] Michał Król et al. NFaaS: named function as a service. In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, 2017.

[7] Michał Król et al. Rice: Remote method invocation in icn. In *Proceedings of the 5th ACM Conference on Information-Centric Networking*, 2018.

[8] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Ndne: Enhancing named data networking to support cloudification at the edge. *IEEE Communications Letters*, 20(11):2264–2267, 2016.

[9] Michał Król et al. Compute first networking: Distributed computing meets icn. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*, pages 67–77, 2019.

[10] Spyridon Mastorakis et al. Towards service discovery and invocation in data-centric edge networks. In *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, pages 1–6. IEEE, 2019.

[11] Y. Liu, Z. Zeng, X. Liu, X. Zhu, and M. Z. A. Bhuiyan. A novel load balancing and low response delay framework for edge-cloud network based on sdn. *IEEE Internet of Things Journal*, 7(7):5922–5933, 2020.

[12] Spyridon Mastorakis et al. Icedge: When edge computing meets information-centric networking. *IEEE Internet of Things Journal*, 2020.

[13] Dumitrel Loghin et al. Towards analyzing the performance of hybrid edge-cloud processing. In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 87–94. IEEE, 2019.

[14] Susmit Shannigrahi, Spyridon Mastorakis, and Francisco R Ortega. Next-generation networking and edge computing for mixed reality real-time interactive systems. In *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2020.

[15] M. S. Elbamby et al. Toward low-latency and ultra-reliable virtual reality. *IEEE Network*, 32(2):78–84, 2018.

[16] Tony Driscoll, Suzanne Farhoud, Sean Nowling, et al. Enabling mobile augmented and virtual reality with 5g networks. Technical report, 2017.

[17] Content Delivery Network and Cloud Performance - Citrix, May 2021.

[18] NDN Team. Ndn interest parameters specification, 2019.

[19] NDN Team. Ndn forwarding hint specification, 2019.

[20] Peter Zilahy Ingerman and ET IRONS. Thunks. a way of compiling procedure statements with some comments on procedure declarations. Technical report, PENNSYLVANIA UNIV PHILADELPHIA, 1960.

[21] Tianxiang Li et al. Distributed dataset synchronization in disruptive networks. In *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pages 428–437. IEEE, 2019.

[22] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. On the evolution of ndnsim: An open-source simulator for ndn experimentation. *ACM SIGCOMM Computer Communication Review*, 47(3):19–33, 2017.

[23] Chanh Nguyen et al. Why cloud applications are not ready for the edge (yet). In *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pages 250–263, 2019.

[24] AWS Architecture Blog: Internet Routing and Traffic Engineering, May 2020. [Online; accessed 29. May 2020].