applied sciences

MDPI

Article

Comparative Analysis of 5G Mobile Communication Network Architectures

Woosik Lee [1], Eun Suk Suh [2,*], Woo Young Kwak [2] and Hoon Han [2]

[1] Korea Telecom, Seongnam-si13606, Korea; woosik.lee@kt.com
[2] Graduate School of Engineering Practice, Institute of Engineering Research, Seoul National University, Seoul 08826, Korea; wooykwak@snu.ac.kr (W.Y.K.); hahnhoon@snu.ac.kr (H.H.)
* Correspondence: essuh@snu.ac.kr; Tel.: +82-2-880-7175

Received: 6 March 2020; Accepted: 2 April 2020; Published: 4 April 2020

Abstract: Mobile communication technology is evolving from 4G to 5G. Compared to previous generations, 5G has the capability to implement latency-critical services, such as autonomous driving, real-time AI on handheld devices and remote drone control. Multi-access Edge Computing is one of the key technologies of 5G in guaranteeing ultra-low latency aimed to support latency critical services by distributing centralized computing resources to networks edges closer to users. However, due to its high granularity of computing resources, Multi-access Edge Computing has an architectural vulnerability in that it can lead to the overloading of regional computing resources, a phenomenon called regional traffic explosion. This paper proposes an improved communication architecture called Hybrid Cloud Computing, which combines the advantages of both Centralized Cloud Computing and Multi-access Edge Computing. The performance of the proposed network architecture is evaluated by utilizing a discrete-event simulation model. Finally, the results, advantages, and disadvantages of various network architectures are discussed.

Keywords: 5G mobile communication network; multi-access edge computing; centralized cloud computing; hybrid cloud computing

1. Introduction

With the emergence of the smartphone, the amount of data traffic has been growing exponentially since 2008. Based on user demand and the introduction of new technologies, mobile communication technology is currently evolving from fourth generation (4G) to fifth generation (5G). The 5G mobile communication technology (5G herein) can be characterized by three major features: enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC), and Ultra Reliable Low Latency Communication (URLLC). The main concept of the eMBB is to extend frequency resources to a millimeter Wave (mmWave) above 6 GHz. mMTC is defined as a network capacity to simultaneously accommodate millions of devices in an area of 1 km^2. Finally, URLLC is a feature for guaranteeing end-to-end latency within 10 ms.

Among the main features of 5G, URLLC can be recognized as the most distinguished feature that differentiates 5G from previous generations of mobile communication technologies. Prior to the discussion of the URLLC, the mobile network architecture needs to be defined. A mobile network can be sub-divided into radio access network and core network. The radio access network is the network link assigned on radio frequency resources and the core network is the network link assigned to wire-line resources. For 5G standard, significant efforts have been made to reduce the latency of radio access networks, including the use of mini-slots and shortened Transmission Time Interval [1–5]. However, reducing the core network latency is more difficult because of the physical distance limitation (~200 km/ms).

To address the challenges in reducing core network latency, a new network architecture called Multi-access Edge Computing (MEC) architecture has been proposed [6–8]. The main concept of MEC is to reduce the physical distance between user entity (UE) and application server. Network functions are virtualized on clouds, in which application servers are collocated; MEC is an architecture that allocates computing resources closer to UE. Figure 1 shows the current 4G architecture and the new MEC architecture. The 4G architecture shown in this paper will be referred to as the Centralized Cloud Computing (CCC) architecture [9,10], as tasks are processed at the central cloud.

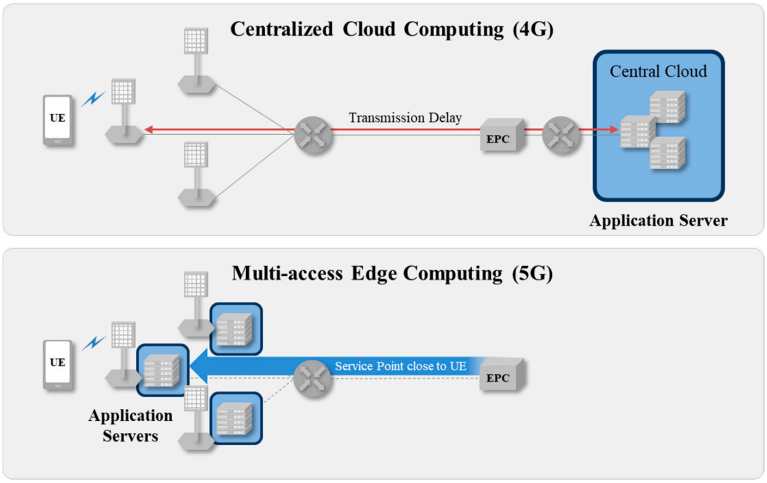

Figure 1. Centralized Cloud Computing (CCC) and Multi-access Edge Computing (MEC) architectures.

In the CCC architecture, the application server is located at the central cloud, causing a delay in the core network if the physical distance between the central cloud and the UE is long. The MEC architecture can reduce transmission delay by placing application server and computing resource (edge cloud) near the target region. However, a network based on the MEC architecture can experience significant delay under certain disruptive situations. Figure 2 shows two of these situations and illustrates the different responses of the CCC and MEC architectures.

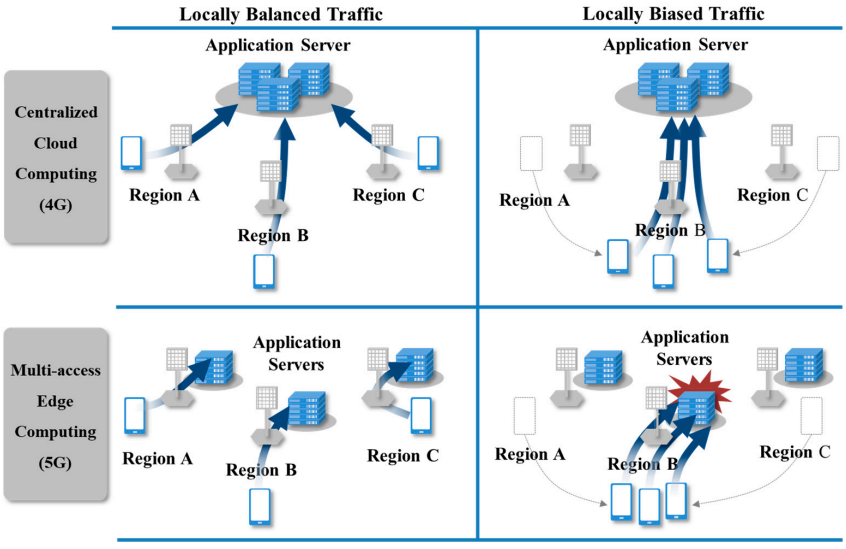

Figure 2. CCC and MEC architecture responses to different data traffic situations.

The left side of the figure shows both the CCC and MEC-based networks operating under normal data traffic conditions. All the data traffic in the CCC-based network is transmitted to the central application server, where the central cloud is located. In the MEC-based network, the regional data

traffic is transmitted to respective regional application server to be processed at the edge cloud. The right side of the figure depicts a situation where a data traffic spike occurs in a particular region. Events where large crowds gather, such as the New Year's Eve celebration in Times Square, can cause unusually high data traffic in a specific region. In the current CCC-based network, no noticeable delay occurs as the data traffic is handled by the central cloud, which has sufficient computing resources. However, the MEC-based network has a high probability of experiencing significant delay in that specific region. This is due to the fact that the computing resources allocated to an application server for that particular region may not be sufficient to handle the congested data traffic in the allocated time. Although MEC-based networks can significantly decrease delay time through computer resource distribution and forward deployment of application servers, they are vulnerable to regional data traffic spikes, if insufficient computing resources are allocated in that region.

In summary, the CCC and MEC architectures have their advantages and disadvantages. The CCC architecture can utilize centralized computing resources for all users' traffic, thus being more robust to regional data traffic spike. However, the transmission delay in the CCC architecture is directly proportional to the physical distance between feature elements, resulting in greater time delay for data traffic generated far away from the central cloud. Conversely, the MEC architecture, by locating feature elements in close proximity, can reduce transmission delay. However, it is vulnerable to regional data traffic spikes due to the limited computing resources allocated to each application server.

In this study, we propose a new hybrid network architecture, appropriately named Hybrid Cloud Computing (HCC) architecture, which combines the advantages of both CCC and MEC network architectures while minimizing their risks. This newly proposed architecture, along with the standard CCC and MEC architectures, is analyzed in terms of the data traffic processing capability using a discrete event-based simulation model. The comparative analysis results for the three network architectures are presented in this paper. The following are the contributions of this study: (1) Establishing a simulation model of a communication network architecture that incorporates an offloading decision algorithm considering the queuing status in the system, which is a major factor influencing the latency in actual network environment, and (2) Assessing the feasibility of various communication network architectures using this model, including the newly proposed HCC architecture, with actual field data to reflect real-life situations.

2. Previous Works

Mobile cloud computing technology is the foundation of the MEC architecture. The technology attempts to overcome the limited computing resources of the mobile terminals of subscribers and to reduce power consumption by offloading the workload to a cloud located in a remote location. Research related to this area include work by Kemp et al. [11], who constructed a prototype of and image processing application that incorporated an offloading methodology, where the terminal workload is offloaded to a server. Cuervo et al. [10] proposed a mobile cloud computing related framework named MAUI. Using the proposed framework, they defined a system to classify the offloading code, client-server connection, and offloading decision function based on optimization using linear programming and profiling function to improve the decision-making accuracy. Chun et al. [9] proposed the CloneCloud framework, which enables more effective offloading through virtualization-based synchronization of program execution environment between the user terminal and server. More recent work focuses on establishing the application offloading process for augmented reality through mobile virtual reality [12].

The offloading methodology is another closely related topic that has received a lot of attention. Research on this subject is divided into two different areas, namely the single layer offloading methodology and multiple layer offloading methodology. The former focuses on developing methods to improve the efficiency of decision making for offloading between user terminal and edge cloud, while the latter focuses on solving edge cloud overload problem from a more macroscopic (system level) perspective.

There are two different approaches to the single layer offloading methodology. The first approach involves the optimization of computing resources. Taking this perspective, Huang et al. [13] used the Lyapunov optimization technique to solve the offloading decision problem for the purpose of minimizing terminal power consumption, subject to a dependency between the off-loadable modules. Recent work to implement this approach was published by Chen and Hao [14], who proposed a method to minimize the delay time through efficient allocation of the computing resources required for offloading in the virtualized network environment. Additionally, Hao et al. [15] proposed a new concept called task caching, which solved the terminal power consumption minimization problem through mixed integer optimization, while considering MEC, storage resources, and execution time.

The second approach to the single layer offloading methodology attempts to coordinate and optimize both computing and communication resources. In the work by Liu et al. [16], the authors modeled the terminal task buffer queuing state, task processing state, and offloading transmission state based on a Markov decision process and proposed a solution to the delay time optimization problem using a one-dimensional searching algorithm. Hong et al. [17] proposed a method to obtain an optimal trade-off condition between delay time and power consumption using the finite-state Markov chain model. In recent work by You et al. [18], the authors proposed an algorithm to solve the computing and communication resource optimization problems using an offloading decision model, which is divided into computing resource model based on time division access and communication resource model based on orthogonal frequency division access.

The multiple layer offloading methodology focuses on solving the MEC architecture overload from a more macroscopic perspective. Srinivasan and Agrawal [19] proposed a new architecture called the mobile-central office re-architected as a data center (M-CORD) to address the MEC architecture's overload problem. The M-CORD is a network architecture that utilizes the service orchestration platform called XOS to link the distributed and virtualized base stations to the core network, while allocating centrally located computing resources to each individual edge cloud. Through utilization of this architecture, sharing of computing resource between individual edge clouds is possible, and the edge cloud overload problem can be addressed. Lin et al. [20] defined the MEC architecture as a multiple layer system consisting of terminal edge cloud, and core layers. They proposed a methodology for optimizing computing resources and minimizing delay time through best allocation of computing resources and traffic to nodes in each layer. Kiani and Ansari [21] divided an MEC architecture into three different layers and implemented an auction-based optimization algorithm to obtain minimum cost for the network.

Recently, new methodologies have been proposed for optimizing offloading tasks between clouds. Table 1 lists the recent works in this area.

Table 1. Published works on offloading optimization between clouds.

Previous Research	Multi-Tiered	Offloading	Optimization Objectives	Optimization Methodology	Queuing Status Sharing
Hou et al. [22]	No	Horizontal	Latency	Heuristic	No
Ren et al. [8]	Yes	Vertical	Latency	Convex	No
Ahn et al. [23]	Yes	Vertical	Cost	Heuristic	No
Zhao et al. [24]	Yes	Vertical	Utilization	Convex	No
Zhang et al. [25]	Yes	Vertical	Profit	Convex	No
Zhang et al. [26]	Yes	Vertical	Profit	Convex	No
Ruan et al. [27]	Yes	Vertical	Latency	Heuristic	No
Thai et al. [28]	Yes	Vertical and Horizontal	Cost	Approximation	No

Hou et al. [22] proposed a model for horizontal offloading between autonomous vehicles and road infrastructure, and optimized for latency using a heuristic algorithm. Ren et al. [8] assumed a vertical offloading model for edge and cloud, and optimized for latency using convex optimization. Ahn et al. [23] proposed a cooperation model between edge and cloud for video analytics based on Internet of Things (IoT). They defined the optimization as a mixed integer problem and applied the heuristic algorithm to optimize the cost required for cloud operation. Zhao et al. [24] employed a vehicle network comprising edge and cloud, and tried to optimize cloud utilization using convex

optimization. Zhang et al. [25,26] used whole-sale and buy-back models between edge and cloud to share computing resources, and optimized the profit from the edge's perspective. Ruan et al. [27] assumed an energy management infrastructure as a cloud model comprising three tiers, and optimized for latency using joint optimization between Stackelberg and Lyapunov-based pricing and energy demand. Thai et al. [28] proposed vertical and horizontal offloading models between edge to cloud and edge to edge, and used an approximation algorithm to minimize the cost.

Surveying previous works revealed several research gaps that can be addressed. Several papers proposed new frameworks for offloading between user terminals and the cloud and they established foundations for future MEC architecture related research. However, the limitation of their works is that their main direction was focused on the design of the system's functional elements, and they did not propose ways to optimize offloading under various operating conditions. Moreover, the single layer offloading methodology is another research area on which the majority of research has been published. However, the single layer offloading methodology exhibits limitations in addressing the MEC architecture's issues of computing resource shortage, work overload and quality degradation resulting from local traffic spike. The studies on multiple layer offloading methodology focus optimal allocation of computer resources and traffic to multiple terminals, edge clouds, and cloud nodes at the upper layer. Numerous works published in this area were focused on solving the high complexity optimization problem at the center. Although many scholars have studies offloading optimization, as listed in Table 1, no study has incorporated queuing status sharing between clouds in the offloading decision making framework, which has a significant impact on the performance in an actual cloud network operating environment. In this research, by incorporating queuing status sharing between clouds into the simulation model, we estimate realistic latency values to make offloading decision. This is used to assess the newly proposed HCC architecture.

3. Proposed Architectures

3.1. Overview

In this study, a new communication architecture that can address the architectural problems encountered by current CCC and MEC architectures, is proposed. Figure 3 illustrates the three architectures analyzed in this study: (1) CCC architecture, which is the current 4G architecture; (2) newly proposed HCC architecture; (3) MEC architecture, which is the architecture proposed for 5G systems.

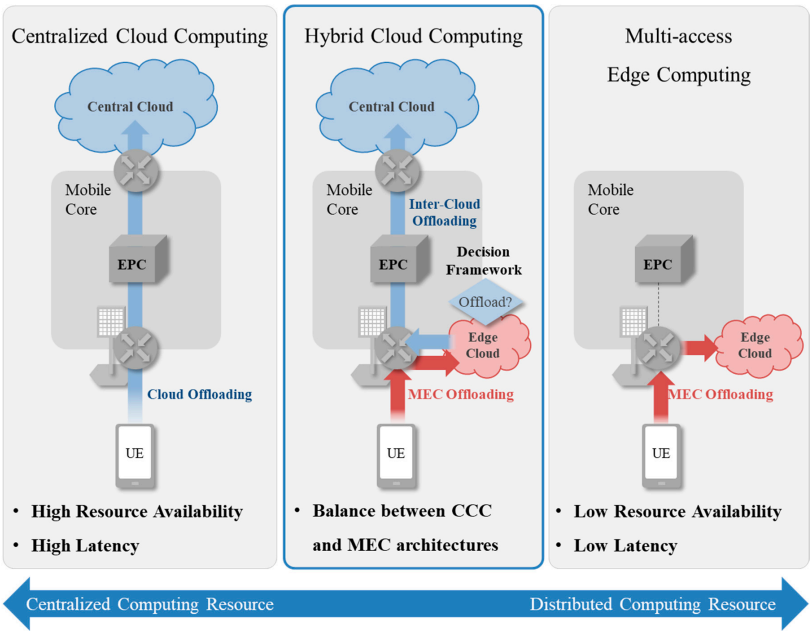

Figure 3. Three network communication architectures proposed for the analysis.

3.2. CCC Architecture

The CCC architecture is the current architecture used in 4G systems. In this architecture, the service request of the user passes through the local level base station and backhaul network and is then delivered to core functions, such as a serving gateway or a public data network gateway. The request then goes through the internet before finally arriving at the central cloud to complete the service process. In the actual mobile communication network, the physical transmission distance for a related service could be several hundred kilometers, causing high latency. However, the CCC architecture features the benefit of centralized pooled computing resources as shown in Figure 2.

3.3. HCC Architecture

To incorporate the advantages of both CCC and MEC architectures, a new network architecture, named HCC architecture, is proposed in this study. As illustrated in Figure 3, the HCC architecture has computing resource at the central cloud, and at application servers (edge clouds) for the regions they service. The normal volumes of local data traffic are handled by the application server and edge cloud for that region, resulting in short delay time. However, when a data traffic spike occurs and the edge cloud at the local application server is tied up with previous tasks, the extra data traffic is offloaded to the computing resource located at the central cloud, rather than waiting in queue at the local application server. The local application server makes an offloading decision for every task to achieve lower latency at that moment. To estimate the latency, real time statistical data is gathered at the respective application server, including the status of computing resources and process queue for the edge and central clouds. This can alleviate potential data traffic spike problem experienced by a completely distributed MEC architecture. The HCC architecture provides a more balanced approach to handling data traffic compared to CCC architecture, in which every task must be processed at the central cloud regardless, and MEC architecture, in which every task must be processed at local edge clouds. The HCC architecture proposed in this study is based on the MEC standard reference architecture established by the European Telecommunication Standard Institute (ETSI) [6]. The architecture and associated offloading decision-making framework can be realized in the form of a mobile edge service in the standard reference architecture. Moreover, the information between clouds, required for the offloading decision-making framework, is exchanged through an Mp3 interface in the standard reference architecture. The proposed HCC architecture is a high-level design based on technologies that satisfy standard requirements. Further, the functional elements of the system proposed in this paper are mapped to lower modules in the standard reference architecture, and therefore, can be easily implemented in a real network environment.

3.4. MEC Architecture

In a 5G system based on network virtualization, the logical functions to process the user traffic at the base station and core instrument are integrated into a user plane function and is distributed to serviced regions. Furthermore, considering the service-based architecture, which aims to fuse the network and application, the 5G system supports a communication path between the user plane function and the application. The MEC architecture takes advantage of this feature by placing an application function, which was located on the central cloud originally, to a local cloud at individual application servers and connects it with the user plane function. The MEC architecture reduces the delay time significantly by moving the application function closer to the UE. However, due to the distributed allocation of computing resources, the available computing resource per location is limited, making the architecture vulnerable to regional traffic spikes as shown in Figure 2.

4. Simulation Model

4.1. Simulation Model Overview

To assess the capabilities of the different architectures, a simulation model of the network architecture was constructed. Figure 4 shows the communication network modeled on the left, and the overall structure of the corresponding simulation model on the right.

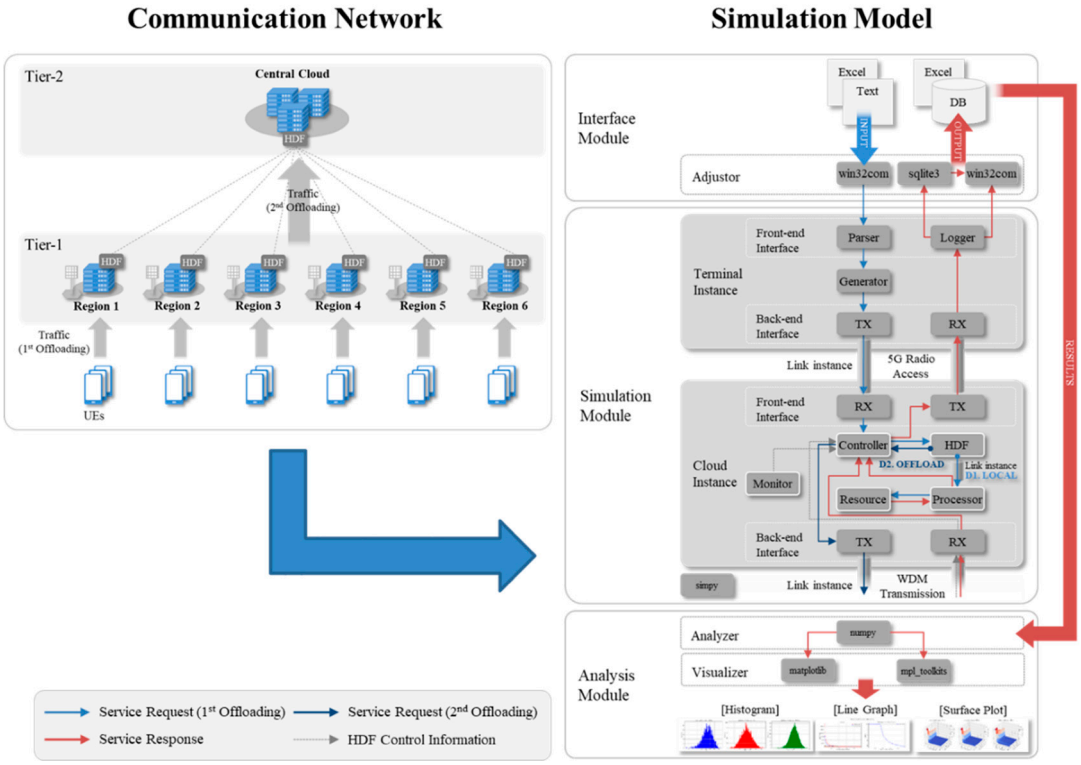

Figure 4. Network architecture simulation model.

The communication network shown on the left in the figure has a multi-tiered structure. The central cloud is located in Tier-2, while six edge clouds are located in Tier-1. The central cloud is connected to the edge cloud in a star topology structure. The various UEs communicate continuously with the edge clouds. The edge clouds and the central cloud are connected through an IP network based on optical transmission technology. The network simulation model was constructed so that it represented the CCC, HCC, or MEC architectures depending on the computing resource allocation between the central cloud and edge clouds. Thus allocating 100% of the computing resources to the central cloud represented the CCC architecture, while allocating 100% of computing resources to the edge clouds represented the MEC architecture. The HCC architecture was represented by any architecture in which the computing resources were allocated to both the central and edge clouds.

The simulation model was developed in the Microsoft Windows 10 (64 bit) environment on a laptop computer equipped with Intel Core i5-6200U processor with 8 GB of RAM. The main module used to develop simulation model was Python-based SimPy (version 3.0.10), which is a discrete-event simulation framework. The additional modules used were NumPy for data analysis and Matplotlib for graphic output.

As shown on the right side of Figure 4, the simulation model is divided into interface, simulation, and analysis modules. The interface module permits the user to input simulation parameter values, collect simulation results, and produce status information. The simulation module receives information from the interface module and simulates the communication network response.

The simulation module can be decomposed into a cloud module and a link module. The cloud module can be realized as an edge cloud or a central cloud, where the central cloud can be placed in a higher tier. Moreover, for each individual cloud, an arbitrary amount of computing resources can be allocated, and the offloading decision-making framework and policy to execute it can be inputted. The link module is used to realize network links between clouds, and by adjusting the physical parameters, such as the latency characteristics and variance, it can simulate wired and wireless links.

The analysis module receives the simulation results from the interface module and visualizes it appropriately to suit the objective of the analysis. By displaying the results in different types of graphs, it allows efficient analysis and interpretation of the simulation results.

There are two newly designed modules incorporated into the network architecture simulation model to specifically simulate and optimize the HCC architecture. The first module is the HCC offloading decision module, and the second one is the HCC computing resource allocation module.

4.2. HCC Offloading Decision Module

The HCC offloading decision module is specifically designed for the HCC architecture. The role of the HCC offloading decision module is to execute offloading decisions for individual tasks entering the edge cloud to minimize end-to-end latency from user perspectives. Since the CCC and MEC architectures do not need to make task offloading decisions, the HCC offloading decision module is utilized when the network model is simulating the HCC architecture. When a specific task enters the edge cloud, it compares the time it takes to process the task directly at the edge cloud (t_{EC}) to the time it takes to process by offloading it to the central cloud (t_{CC}). Here, t_{EC} can be expressed as

$$t_{EC} = t_{EC}^{queue} + t_{EC}^{process} \tag{1}$$

where t_{EC}^{queue} is delay time in the queue and $t_{EC}^{process}$ is the actual task processing time. Additionally, the task processing time through the central cloud, t_{CC}, can be expressed as

$$t_{CC} = t_{CC}^{queue} + t_{CC}^{process} + 2t_{EC}^{transport} \tag{2}$$

where t_{CC}^{queue} is delay time in the queue, $t_{CC}^{process}$ is the actual task processing time, and $2t_{EC}^{transport}$ is the round trip time for the task from the edge cloud to central cloud and back.

Equations (1) and (2) are further expanded by introducing new parameters: w, R_{EC}, and R_{CC}. Parameter w is the actual work load of an individual task. R_{RC} and R_{CC} are the computing resources for the edge cloud and the central cloud, respectively, expressed in CPU cycle frequency in units of megahertz (MHz). With these parameters, the time required to process a specific task with work load w for the edge cloud and the central cloud can be expressed, respectively, as follows:

$$t_{EC} = t_{EC}^{queue} + \frac{w}{R_{EC}} \tag{3}$$

$$t_{CC} = t_{CC}^{queue} + \frac{w}{R_{CC}} + 2t_{EC}^{transport} \tag{4}$$

In addition to the parameters defined, the decision variable x need to be defined. If we set $x = 0$, then it is assumed that $t_{EC} \leq t_{CC}$ and the task is processed at the edge cloud. If we set $x = 1$ then the task is offloaded to the central cloud ($t_{EC} > t_{CC}$). The overall latency (t_{HC}), due to the task offloading decision, can be expressed as

$$t_{HC} = (1 - x)t_{EC} + xt_{CC} \tag{5}$$

4.3. HCC Architecture Computing Resource Allocation Module

The second simulation module is used to determine optimal computing resource allocation between individual edge clouds and the central cloud within the HCC architecture. The module is constructed on the basis of queueing theory.

Assume that λ is the arrival rate for tasks into the HCC network, and μ is the service rate. Then, by using the Little's Law, the queuing time (t^{queue}) for each edge cloud or central cloud can be expressed as

$$t^{queue} = \frac{\lambda/\mu}{\mu - \lambda} \tag{6}$$

Here, if we assume a simple network in which the edge cloud and central cloud are connected in a 1:1 manner, then the queuing time for the edge cloud and the central cloud are expressed, respectively, as follows:

$$t_{EC}^{queue} = \frac{\lambda_{EC}/R_{EC}}{R_{EC} - \lambda_{EC}} \tag{7}$$

$$t_{CC}^{queue} = \frac{\lambda_{CC}/R_{CC}}{R_{CC} - \lambda_{CC}} \tag{8}$$

The arrival rate for edge cloud (λ_{EC}) and central cloud (λ_{CC}) can alternatively be expressed as the ratio of tasks offloaded from edge cloud to the central cloud (p), which can then be stated as $\lambda_{EC} = \lambda(1-p)$ and $\lambda_{CC} = \lambda p$.

To prevent queuing time divergence for a specific edge cloud or central cloud, the HCC offloading decision must be made to minimize the difference, Δ, between t_{EC}^{queue} and t_{CC}^{queue}, which can then be expressed as

$$\Delta = \left| \frac{\lambda(1-p)/R_{EC}}{R_{EC} - \lambda(1-p)} - \frac{\lambda_p/R_{CC}}{R_{CC} - \lambda_p} \right| \tag{9}$$

Taking the assumption one step further, a more realistic network structure is considered, where edge clouds and central cloud are connected in N:1 star topology configuration. Here, Δ can be expressed as

$$\Delta = \left| \frac{1}{N} \sum_{i=1}^{N} \frac{\lambda_i(1-p_i)/R_{EC,i}}{R_{EC,i} - \lambda_i(1-p_i)} - \frac{\sum_{i=1}^{N} \lambda_{p,i}/R_{CC}}{R_{CC} - \sum_{i=1}^{N} \lambda_p} \right| \tag{10}$$

In the simulation model, $R_{EC,i}$ and R_{CC} are set as the design variables, and t_{HC} and Δ are measured through the simulation. Additionally, variable λ is represented by the data traffic and μ is represented by the computing resource.

5. Simulation Assumptions and Architecture Robustness Assessment

5.1. Simulation Assumptions

Assumptions for the CCC, HCC, and MEC architecture simulations were made for fair comparative analysis. They are listed below.

Network structure: As shown in Figure 4, the network consists of one central cloud facility, connected to six edge clouds in cluster that is in a star shaped topology. For the simulation, it was assumed that the distance between the city where edge clouds are located and the central cloud facility was 200 km for all three architectures. For the radio access network path, it was assumed that the latency for all three architectures are same.

Edge cloud connectivity: For the simulation case study, one of the key simplifying assumption was the implementation of the simplified network policy, which assumes that network will only use dedicated resource allocated to the region. Another assumption made was that each edge cloud interact with central cloud only, not with other edge clouds. Although the standard from ETSI takes into account that edge clouds can connect to nearby edge clouds for mobile application purposes, the case study

presented in this study primarily focuses on computational offloading from edge clouds to the central cloud, which is not related to the task transfer from an edge cloud to another edge cloud, thus justifying the assumption made.

Latency in radio access network and core network: For communication network, latency occurs in radio access network and core network paths. In the simulation, it was assumed that all three architectures had equal amount of latency for both paths for fair comparison. For the radio access network path, the assumed latency for the simulation was set at 3 ms. For the core network path, the amount of latency was set so it corresponds to 200 km distance equivalent latency between the central cloud and edge clouds for all architectures.

Measurement units: For the simulation, the task is defined as a job that a mobile device requests to the cloud, and the workload is defined as the amount of computing resource required to process the task. The cloud computing resource is expressed as the CPU cycle frequency in units of MHz. The workload, which is the amount of CPU cycles required to complete a given task, is expressed in 1000 cycle. The processing time, which is the time required to complete a given task, can be obtained by dividing the task workload by the available computing resource. It is expressed in units of ms. Data traffic, which is the number of tasks generated per time unit, is expressed as tasks per second.

Application and key performance index for network architectures: For the case study, the network was used to transmit data to and from personal mobile communication devices. The performance measure used for this architectural analysis was the percentage of tasks that exceeded end-to-end processing time of 10 ms. This is based on the latency guideline proposed by Lee et al. [29]. The threshold for out-of-specification percentage was set to 5% after consulting Korea Telecom's internal subject matter experts regarding the out-of-specification deviation for this particular application.

5.2. Architecture Robustness Assessment

Using the simulation model constructed, CCC, HCC, and MEC architectures are simulated. For the initial architecture assessment, we tested the robustness of each architecture with respect to the increase in task data traffic.

For the robustness assessment, the total computer resource for each architecture was set at 60,000 MHz. For CCC architecture, all computer resources are placed in the central cloud. For MEC architecture, six edge clouds were assigned 10,000 MHz each. For the HCC architecture, the trend analysis was performed using the HCC architecture computing resource allocation module introduced in the previous section. Figure 5 shows the analysis results in the form of a log-scale. The x-axis represents the percentage of computing resources allocated to the central cloud, and the y-axis represents the Δ value in Equation (10) for the HCC architecture with $N = 6$.

For this particular HCC architecture, the trend analysis revealed that the optimum central cloud to edge clouds distribution ratio was 25:75, meaning 25% of the total computing resource (15,000 MHz) was allocated to the central cloud and 75% of the resources (45,000 MHz) were allocated in equal amounts to six edge clouds. It should be noted that the trend analysis for determining the optimal resource allocation ratio between the central cloud and edge clouds need to be conducted for different situations, since the optimal ratio may be different for each specific situation.

The task data traffic experienced by individual region was varied from 1800 tasks per second to 10,800 tasks per second. For each architecture, the percentage of tasks with out-of-specification percentage was measured as the function of task traffic. Figure 6 shows the results of the robustness assessment for all three architectures under increasing data traffic.

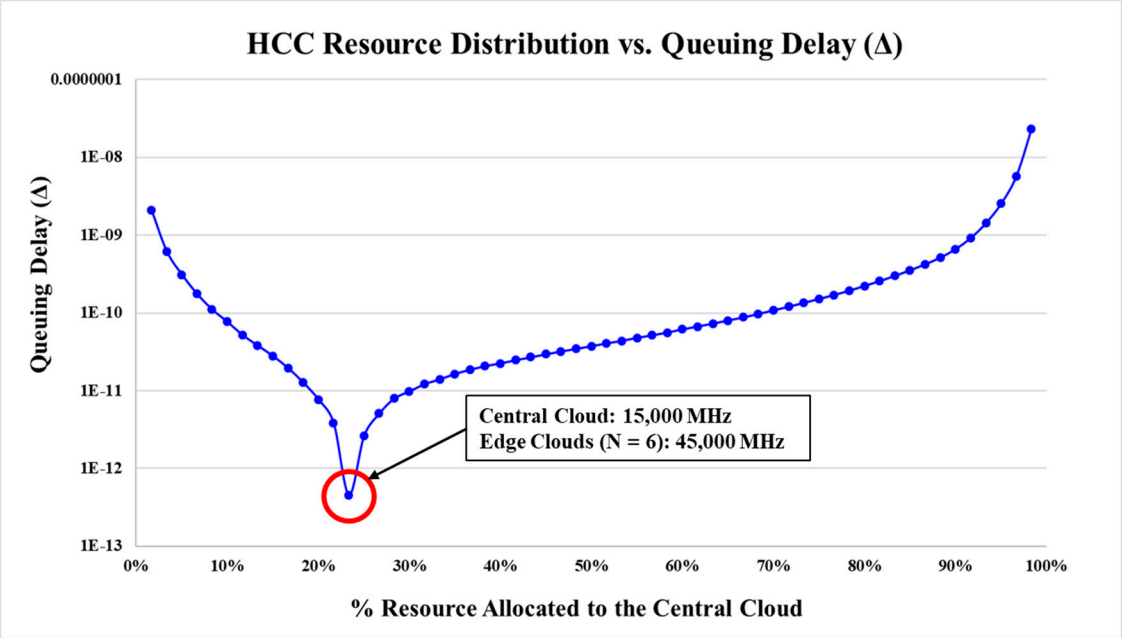

Figure 5. Trend analysis results for computing resource allocation in the Hybrid Cloud Computing (HCC) architecture.

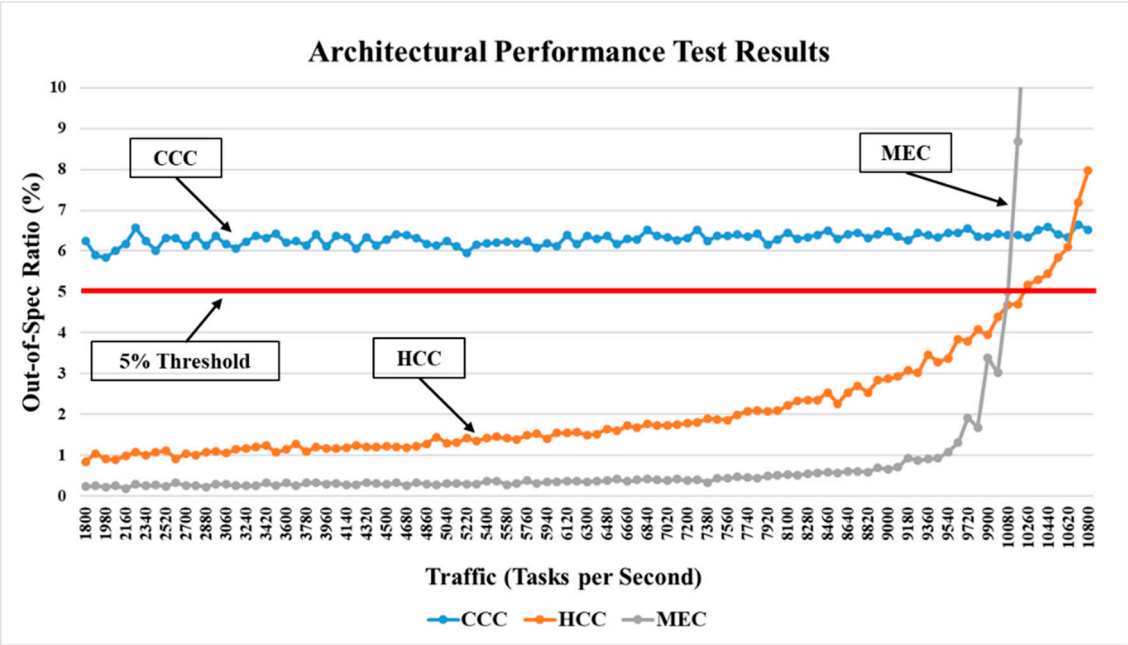

Figure 6. Robustness assessment for CCC, HCC, and MEC architectures.

The results show that the CCC architecture does not always satisfy the 5% out-of-specification ratio; nevertheless, it is also most robust among the three architectures. The HCC architecture shows more sensitivity to the data traffic increase than the CCC architecture. However, for most data traffic ranges simulated, the HCC architecture processed data traffics well within the specification. The MEC architecture had the best performance in terms of the percentage of out-of-specification ratio. However, as the data traffic reaches more than 9000 tasks per second, the MEC architecture's performance deteriorates rapidly, which shows that there is an upper limit to the data traffic range in which MEC is robust. This initial assessment revealed key characteristics of each architecture.

6. Simulation Using Actual Field Data

6.1. Simulated Scenarios

Using the network simulation model developed in this work, the CCC, MEC, and HCC architectures were modeled and their robustness to task traffic increase was assessed. To evaluate their capabilities in a realistic situation, two different daily data traffic scenarios were generated. The first scenario simulated normal daily traffic patterns for six regions of the same city in Korea, while the second scenario assumed data traffic spike in certain regions of the city. Figures 7 and 8 show the data traffic patterns for the two scenarios.

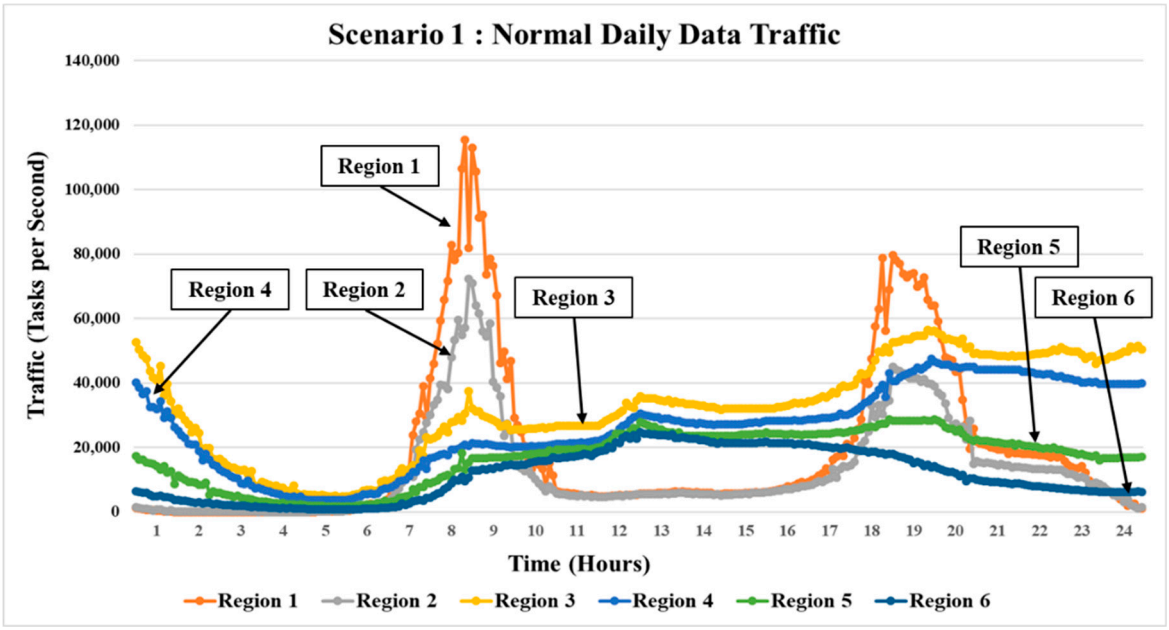

Figure 7. Twenty-four hour data traffic patterns for Scenario 1: normal daily traffic conditions.

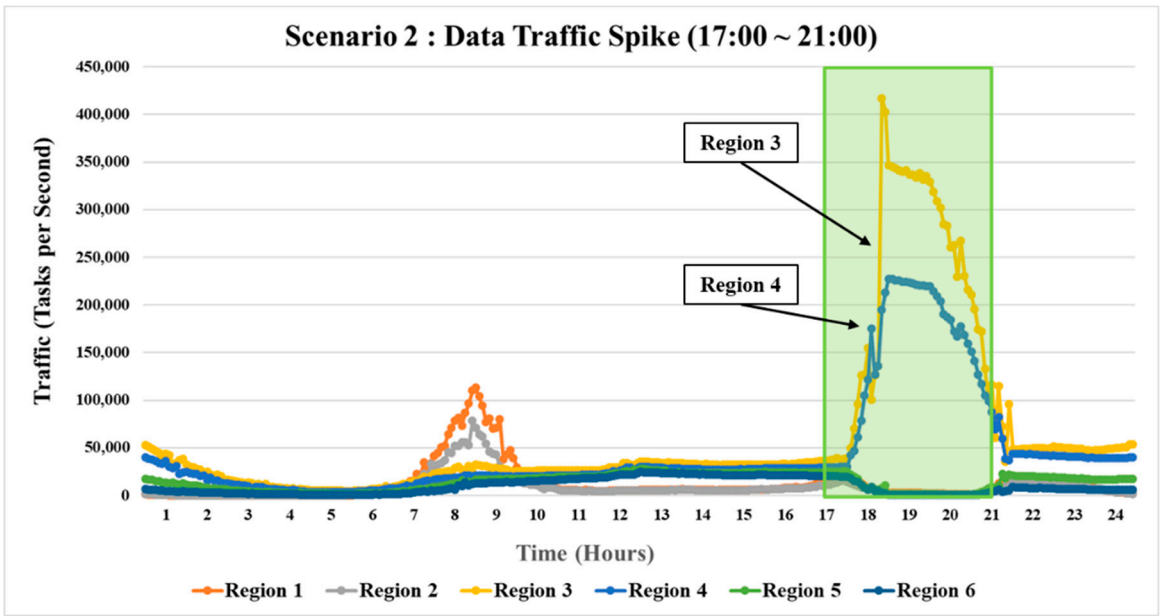

Figure 8. Twenty-four hour data traffic patterns for Scenario 2: regional data traffic spike.

Scenario 1 (normal daily data traffic): In this scenario, six data traffic patterns shown in Figure 7 were used as inputs into six base stations. All the data traffic patterns shown are actual normal daily data traffic patterns acquired during the same 24-h period from the six different regions. Data for each region are gathered at regular intervals, and are grouped by an hour segment to determine the amount of data traffic per hour. In the figure, data traffic pattern for each region are labeled for clarification purposes. Unfortunately, further detail regarding these regions were not available due to confidentiality issues. Each data traffic pattern shows a different peak traffic hour. This is attributed to the type of region where data was acquired. Certain regions were residential areas, with data traffic peaks before and after work hours. Some were commercial areas, with data traffic peaks after normal work hours. Other regions were business districts, with peak traffic hours occurring during normal business hours. Using these data, responses for the three network architectures were obtained through the simulation.

Scenario 2 (regional data traffic spike): In this scenario, a crowd gathering event in two of the six regions in the city was simulated. The six data traffic patterns in Figure 8 shows how the city's data traffic pattern was disrupted. During the evening hours (17:00~21:00 h), the data traffic for four regions were decreased, while the other two regions were loaded with data traffic from the four regions. As with Scenario 1, the responses for the three network architectures were obtained through the simulation using these inputs.

6.2. Simulation Assumptions

Total amount of computing resources for the clouds: For the simulation, all three network architectures were assigned the same amount of total computing resources. This was done to assess proposed architectures under same set of conditions for fair comparison. The total amount of computing resources was determined as follows. Initially, the amount of computing resources necessary to handle the data traffic during the peak time was determined for each of the six regions. Next, the total amount of computing resources was set to the sum of the individual resources determined for each region, which totaled to 52,200 MHz. Therefore, for all three network architectures, the total amount of computing resources was set to 52,200 MHz. Table 2 shows computing resource allocation for each network architecture.

Table 2. Computing resource allocation for three architectures.

	Computing Resource Allocation (MHz)		
Regions	**CCC**	**HCC**	**MEC**
Central Cloud	52,200	7830	-
Region 1	-	8271	9730
Region 2	-	7735	9100
Region 3	-	7353	8650
Region 4	-	7072	8320
Region 5	-	7013	8250
Region 6	-	6928	8150
Total Resources	52,200	52,200	52,200

Computing resource allocation for CCC architecture: For the CCC architecture, the entire 52,200 MHz of computing resources were allocated to the central cloud.

Computing resource allocation for HCC architecture: For the HCC architecture, the computing resources were allocated to both the central cloud and edge clouds. The key issue was determining the allocation ratio between the central cloud and edge clouds. This was done by performing a trend analysis using the simulation model. Computing resources were distributed to six edge clouds, and the amount allocated to each edge cloud was the same as that of the MEC architecture. For each edge cloud, the computing resources were then decreased in steps of 1% from the original resource amount,

and allocated to the central cloud. The simulation was then conducted to assess the performance of the HCC architecture. The results of the trend analysis revealed that the optimal allocation ratio between the central cloud and edge clouds for this particular task traffic was 15% allocation to the central cloud and 85% allocation to edge clouds. Therefore, out of 52,200 MHz of computing resources, approximately 7830 MHz was allocated to the central cloud, and the rest of the resources were allocated to the six edge clouds in their respective regions, as listed in Table 2. One thing to note is that for HCC, the amount of computing resource allocated to the central cloud is 7830 MHz, which is not too much different from other edge clouds allocation amounts. In real situation, there would be more amount of resource allocated to the central cloud. However, for this case study, since all three network architectures compared must have same amount of computing resources, this was determined to be reasonable.

Computing resource allocation for MEC architecture: For the MEC architecture, the entire amount of computing resources was allocated to the edge clouds located in the six base stations. The resource allocated to each edge cloud corresponded to the computing resources required to handle peak task traffic for Scenario 1.

7. Simulation Results and Discussion

7.1. Simulation Results

The response of the three network architectures to data traffic patterns were expressed as the percentage of tasks that exceeded a processing time of 10 ms during the 24-h period. Figures 9 and 10 show the simulation results for the three architectures for Scenarios 1 and 2. The response of each architecture is shown with appropriate labels.

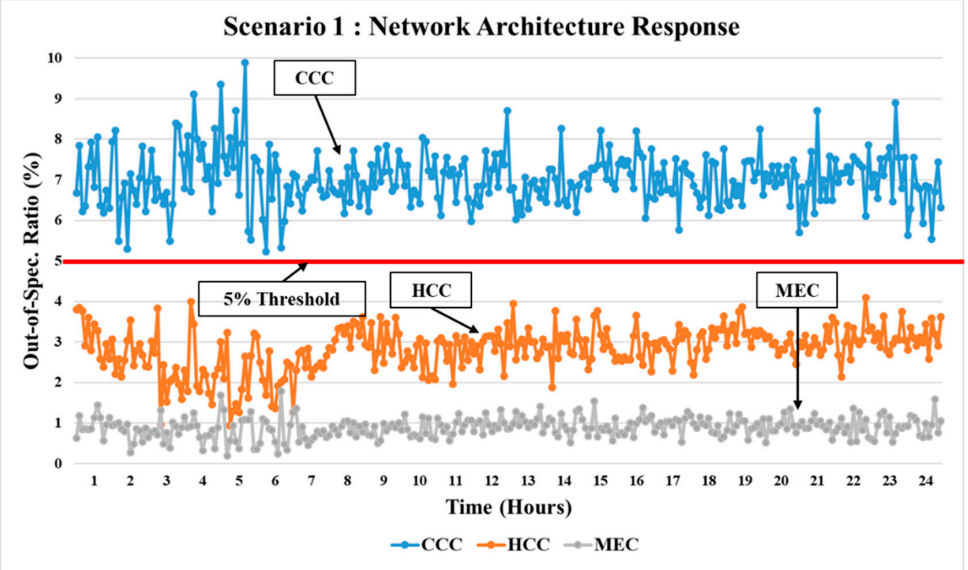

Figure 9. Individual network architecture's response to data traffic shown in Scenario 1.

The results obtained for Scenario 1 demonstrated each architecture's response to normal daily traffic. For the CCC architecture, the percentage of tasks that required more than 10 ms of processing time was always greater than the 5% specification threshold. For the HCC architecture, the percentage of tasks that required more than 10 ms was less than the threshold limit at all times. The MEC architecture performed the best, with the lowest percentage of tasks (~1%) that required more than 10 ms.

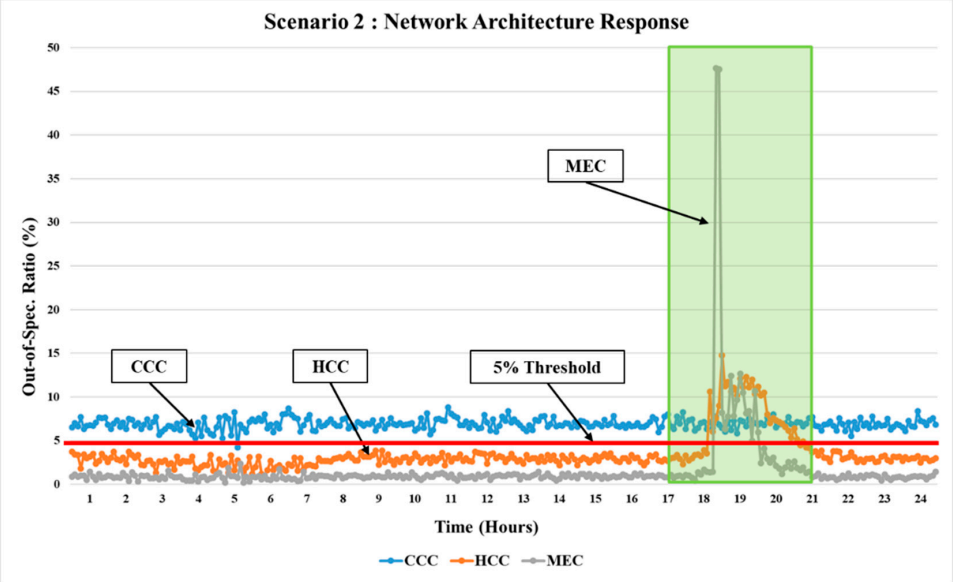

Figure 10. Individual network architecture's response to data traffic shown in Scenario 2.

Scenario 2, which simulated crowd gathering at two regions, demonstrated each network architecture's behavior when the data spike occurred. The CCC architecture demonstrated robust response as expected. This was because, as long as the total amount of data traffic at a given time was the same, the processing time for data traffic remained largely unchanged. However, the percentage of delayed tasks was above the threshold, which is similar to the results of Scenario 1. Conversely, the MEC architecture was significantly sensitive to the data traffic spike. As shown in the Scenario 2 results in Figure 10, the percentage of delayed tasks during the data traffic spike rose to nearly 50%, demonstrating that the architecture is vulnerable to such disturbances. Finally, the HCC architecture's response also demonstrated its sensitivity to the data traffic spike, but in a considerably more inhibited manner than the MEC architecture. The peak percentage of delayed tasks was approximately 15%, which was significantly lower than that of MEC architecture.

The results of the simulation-based analysis demonstrated the advantages and disadvantages for three network architectures. The CCC architecture was robust to data traffic spikes. However, due to the physical distance between the central cloud and base stations, the number of delayed tasks was always above the permitted threshold. The MEC architecture performed reasonably well under normal condition. However, it was considerably vulnerable if a data traffic spike occurred. The HCC architecture offers a compromise between the CCC and MEC architectures, as it meets the specification threshold for delayed task, but still impedes the detrimental effect of data traffic spikes.

7.2. Discussion

Simulation results provided valuable insights for newly proposed HCC architecture in terms of how they behave under different situations, and how these insights gained be used when the HCC architecture is actually implemented. More discussions on simulation results, issues for improvement, and limitations are presented.

In the simulation environment and conditions assumed in the case study, HCC architecture processed tasks well within the allowed specification under the normal condition, although it was slower than processing times by the MEC architecture. This has to do with the restriction of the total resource assigned for allocation between the central cloud and edge clouds of the HCC architecture. For fair comparative analysis of three network architectures, the total computing resource was fixed at 52,200 MHz for each architecture. Since the total computing resource amount was the sum of computing resource required to process peak hour data traffic for each of six regions, task processing time by the MEC architecture was the best among three architectures compared. HCC architecture,

on the other hand, was forced to allocate some of these regional computing resources to the central cloud due to the restriction imposed by simulation assumption. This resulted in somewhat higher average processing time than the MEC architecture, due to some tasks forced to be offloaded to the central cloud during busy hours.

Another reason for higher processing time has to do with HCC offloading decision module located in each application server. When the HCC offloading decision module makes a decision for processing the task at the edge cloud or offload to the central cloud, it checks the queue at the central cloud at that instant. However, since all six application servers in the simulation model are making this decision, only checking the status at the central cloud but not the other application servers, this might generate more unexpected delay at the central cloud than expected, due to the random arrival of tasks from several application servers.

Insights gained from the simulation model and case study can serve as valuable guidelines for the actual HCC architecture development and deployment. The first way to further reduce HCC architecture's task processing time is to allocate the computing resource required to process the peak hour traffic to each application server, and allocate additional computing resource to the central cloud. This, in other words, means that the HCC architecture with amount of computing resources equal to that of MEC architecture at each cloud, and then add extra computing resource at the central cloud. With this implementation option, the performance of HCC architecture under normal data traffic condition should be equal to that of the MEC architecture. During the data traffic spike, as shown in Scenario 2 of the case study, further reduction of task processing delay than results shown in Figure 10 can be expected. However, the marginal overhead required for the HCC architecture over the MEC architecture is the extra investment for the central cloud, along with connective infrastructure between the central clouds and edge clouds.

The second approach to implement the HCC architecture in the field is to differentiate offloading paths for latency critical tasks and non-critical tasks. In reality, latency critical tasks, such as tasks from autonomous vehicles, and non-critical tasks, such as web-browsing tasks from mobile devices, arrive in mixed batches. The offloading decision module determines the priority of the task, and makes decision to either process it locally or to offload to the central cloud. In this case, the computing resource required for each edge cloud would correspond to the amount of latency critical tasks for peak hour, not the total amount of tasks. This may result in less amount of computing resources required than the first option previously mentioned. Although these two implementation options for the HCC architecture require more marginal overhead compared to the MEC architecture, they offer noticeable advantages over the MEC architecture.

8. Conclusions and Future Work

In this study, three communication network architectures were analyzed for their response capability under normal and disruptive data traffic conditions. The current CCC architecture, the MEC architecture, and the newly proposed HCC architecture were modeled using a Python-based simulation software. The behaviors of these architectures were assessed using actual data traffic patterns obtained from the field. The results were used to quantify the advantages and disadvantages of each architecture, and they demonstrated the potential benefit of the new HCC architecture. Insights gained from the case study were discussed, along with how these insights can be used to benefit the actual deployment of the HCC architecture in the field.

There is scope to further analyze the HCC architecture. The architecture can be assessed in terms of other key performance indices, such as the latency, cost, utilization, and profit, which are critical performance criteria for 5G network architectures. Additionally, for the HCC architecture, determining the optimal allocation between the central and edge clouds is an important research topic. What are some other realistic data traffic scenarios that need to be considered? How can computing resources be optimally allocated among the central cloud and edge clouds based on possible scenarios? Is there a "golden allocation ratio" that can include the majority of possible scenarios? Is there any variation

of the HCC architecture that can further enhance its performance? Will there be any performance improvement by integrating machine learning algorithms into offloading decision framework? All of these questions present valuable opportunities for future research.

Author Contributions: Conceptualization, W.L. and E.S.S.; Methodology, W.L., E.S.S, W.Y.K. and H.H.; Software, W.L.; Validation, W.L., E.S.S., W.Y.K. and H.H.; Formal analysis, W.L.; Investigation, W.L. and E.S.S.; Resources, E.S.S.; Data curation, W.L.; Writing—original draft preparation, W.L. and E.S.S.; Writing—review and editing, W.L. and E.S.S.; Visualization, W.L. and E.S.S.; Supervision, E.S.S., W.Y.K. and H.H.; Project administration, E.S.S.; Funding acquisition, E.S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2016R1D1A1A09916273).

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

Nomenclature

4G	Fourth Generation
5G	Fifth Generation
AI	Artificial Intelligence
CCC	Centralized Cloud Computing
eMBB	Enhanced Mobile Broadband
ESTI	European Telecommunication Standard Institute
HCC	Hybrid Cloud Computing
IoT	Internet of Things
MEC	Multi-access Edge Computing
mMTC	Massive Machine Type Communication
mmWave	Millimeter Wave
ms	Millisecond
URLLC	Ultra Reliable Low Latency Communication
UE	User Entity

References

1. Bag, T.; Garg, S.; Shaik, Z.; Mitschele-Thiel, A. Multi-Numerology Based Resource Allocation for Reducing Average Scheduling Latencies for 5G NR Wireless Networks. In Proceedings of the 2019 European Conference on Networks and Communications (EuCNC), Valencia, Spain, 18–21 June 2019; pp. 597–602.
2. Li, J.; Sahlin, H.; Wikstrom, G. Uplink PHY Design with Shortened TTI for Latency Reduction. In Proceedings of the 2017 IEEE Wireless Communications and Networking Conference (WCNC), San Francisco, CA, USA, 19–22 March 2017; pp. 1–5.
3. Wang, L.; Gao, Y.; Dong, Y.; Li, Z.; Zhang, Z.; Zhang, X.; Liu, X. Study on Flexible TTI Scheduling for LAA Systems. In Proceedings of the 2018 15th International Symposium on Wireless Communication Systems (ISWCS), Lisbon, Portugal, 28–31 August 2018; pp. 1–5.
4. Xiaotong, S.; Nan, H.; Naizheng, Z. Study on system latency reduction based on Shorten TTI. In Proceedings of the 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, China, 6–10 November 2016; pp. 1293–1297.
5. Zhang, X. Latency reduction with short processing time and short TTI length. In Proceedings of the 2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Xiamen, China, 6–9 November 2017; pp. 545–549.
6. *Multi-access Edge Computing (MEC)*; Framework and Reference Architecture, ETSI GS MEC 003 V2.1.1; ETSI Industry Specification Group (ISG): Sophia-Antipolis, France, 2019.
7. Montero, R.S.; Rojas, E.; Carrillo, A.A.; Llorente, I.M. Extending the Cloud to the Network Edge. *Computer* **2017**, *50*, 91–95. [CrossRef]
8. Ren, J.; Yu, G.; He, Y.; Li, G.Y.; Li, Y. Collaborative Cloud and Edge Computing for Latency Minimization. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5031–5044. [CrossRef]
9. Chun, B.-G.; Ihm, S.; Maniatis, P.; Naik, M.; Patti, A. CloneCloud: Elastic execution between mobile device and cloud. In Proceedings of the Sixth Conference on Computer Systems, Salzburg, Austria, 10–13 April 2011; pp. 301–314.

10. Cuervo, E.; Balasubramanian, A.; Cho D-k Wolman, A.; Saroiu, S.; Chandra, R.; Bahl, P. MAUI: Making smartphones last longer with code offload. In Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, San Francisco, CA, USA, 15–18 June 2010; pp. 49–62.
11. Kemp, R.; Palmer, N.; Kielmann, T.; Seinstra, F.; Drost, N.; Maassen, J.; Bal, H. eyeDentify: Multimedia Cyber Foraging from a Smartphone. In Proceedings of the 2009 11th IEEE International Symposium on Multimedia, San Diego, CA, USA, 14–16 December 2009; pp. 392–399.
12. Wei, X.; Wang, S.; Zhou, A.; Xu, J.; Su, S.; Kumar, S.; Yang, F. MVR: An Architecture for Computation Offloading in Mobile Edge Computing. In Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, USA, 25–30 June 2017; pp. 232–235.
13. Huang, N.; Wang, P.; Niyato, D. A Dynamic Offloading Algorithm for Mobile Computing. *IEEE Trans. Wirel. Commun.* **2012**, *11*, 1991–1995. [CrossRef]
14. Chen, M.; Hao, Y. Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 587–597. [CrossRef]
15. Hao, Y.; Chen, M.; Hu, L.; Hossain, M.S.; Ghoneim, A. Energy Efficient Task Caching and Offloading for Mobile Edge Computing. *IEEE Access* **2018**, *6*, 11365–11373. [CrossRef]
16. Liu, J.; Mao, Y.; Zhang, J.; Letaief, K.B. Delay-optimal computation task scheduling for mobile-edge computing systems. In Proceedings of the 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, 10–15 July 2016; pp. 1451–1455.
17. Hong, S.-T.; Kim, H. QoE-Aware Computation Offloading to Capture Energy-Latency-Pricing Tradeoff in Mobile Clouds. *IEEE Trans. Mob. Comput.* **2019**, *18*, 2174–2189. [CrossRef]
18. You, C.; Huang, K.; Chae, H.; Kim, B.-H. Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 1397–1411. [CrossRef]
19. Srinivasan, K.; Agrawal, N.K. A study on M-CORD based architecture in traffic offloading for 5G-enabled multiaccess edge computing networks. In Proceedings of the 2018 IEEE International Conference on Applied System Invention (ICASI), Chiba, Japan, 13–17 April 2018; pp. 303–307.
20. Lin, Y.-D.; Lai, Y.-C.; Huang, J.-X.; Chien, H.-T.; Huang, C.-X. Three-Tier Capacity and Traffic Allocation for Core, Edges, and Devices for Mobile Edge Computing. *IEEE Trans. Netw. Serv. Manag.* **2018**, *15*, 923–933. [CrossRef]
21. Kiani, A.; Ansari, N. Toward Hierarchical Mobile Edge Computing: An Auction-Based Profit Maximization Approach. *IEEE Internet Things J.* **2017**, *4*, 2082–2091. [CrossRef]
22. Hou, X.; Ren, Z.; Wang, J.; Cheng, W.; Ren, Y.; Chen, K.-C.; Zhang, H. Reliable Computation Offloading for Edge Computing-Enabled Software-Defined IoV. *IEEE Internet Things J.* **2020**, *1*. [CrossRef]
23. Ahn, S.; Lee, J.; Kim, T.Y.; Choi, J.K. A Novel Edge-Cloud Interworking Framework in the Video Analytics of the Internet of Things. *IEEE Commun. Lett.* **2020**, *24*, 178–182. [CrossRef]
24. Zhao, J.; Li, Q.; Gong, Y.; Zhang, K. Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7944–7956. [CrossRef]
25. Zhang, Y.; Lan, X.; Li, Y.; Cai, L.; Pan, J. Efficient Computation Resource Management in Mobile Edge-Cloud Computing. *IEEE Internet Things J.* **2018**, *6*, 3455–3466. [CrossRef]
26. Zhang, Y.; Lan, X.; Ren, J.; Cai, L. Efficient Computing Resource Sharing for Mobile Edge-Cloud Computing Networks. *IEEE/ACM Trans. Netw.* **2020**, 1–14. [CrossRef]
27. Ruan, L.; Yan, Y.; Guo, S.; Wen, F.; Qiu, X. Priority-Based Residential Energy Management With Collaborative Edge and Cloud Computing. *IEEE Trans. Ind. Inform.* **2020**, *16*, 1848–1857. [CrossRef]
28. Thai, M.-T.; Lin, Y.-D.; Lai, Y.-C.; Chien, H.-T. Workload and Capacity Optimization for Cloud-Edge Computing Systems with Vertical and Horizontal Offloading. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 227–238. [CrossRef]
29. Lee, K.; Kim, J.; Park, Y.; Wang, H.; Hong, D. Latency of Cellular-Based V2X: Perspectives on TTI-Proportional Latency and TTI-Independent Latency. *IEEE Access* **2017**, *5*, 15800–15809. [CrossRef]