# Resilient Cloud Control System: Dynamic Frequency Adaptation via Q-learning

Akbarian, Fatemeh; Tärneberg, William; Fitzgerald, Emma; Kihl, Maria

*Published in:*
27th Conference on Innovation in Clouds, Internet and Networks (ICIN)

*DOI:*
10.1109/ICIN60470.2024.10494429

2024

*Document Version:*
Publisher's PDF, also known as Version of record

# Resilient Cloud Control System: Dynamic Frequency Adaptation via Q-learning

Fatemeh Akbarian[1], William Tärneberg[1], Emma Fitzgerald[1] and Maria Kihl[1]
[1]*Department of Electrical and Information Technology, Lund University, Sweden*

2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN) | 979-8-3503-9376-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICIN60470.2024.10494429

***Abstract*—Traditional control systems face challenges in managing high data loads and computing power, prompting the evolution of Cloud Control Systems (CCS)—a fusion of Networked Control Systems (NCS) and cloud computing. Despite offering manifold advantages, CCS encounters hurdles in navigating the dynamic cloud environment characterized by fluctuating workloads, rendering static frequency settings inefficient. Moreover, the optimal utilization of cloud resources poses a pivotal challenge within CCS operations. To address these, the paper proposes a resilient CCS by adapting system frequency dynamically. Leveraging Q-learning, the approach measures Round Trip Time (RTT) and system output errors, dynamically adjusting the system's frequency to minimize control costs, optimize performance within the dynamic cloud environment, and achieve resource frugality, minimizing resource usage. Through real testbed experiments, this paper evaluates and analyzes the effectiveness of the proposed method, aiming to establish an adaptive and efficient control framework aligned with evolving cloud dynamics.**

***Index Terms*—Cloud, Resiliency, Q-Learning, Frequency adaption**

## I. Introduction

Traditional control systems, limited to closed environments, lacked remote monitoring and adjustment capabilities. With the evolution of industries and the emergence of Networked Control Systems (NCS), communication technologies enabled remote control and monitoring. NCSs leverage networks to manage complex systems from a distance, enhancing flexibility and efficiency.

Nonetheless, there is still a demand to address the challenges faced by control systems in managing high data loads and computing power. To tackle these challenges, a new concept called Cloud Control System (CCS) has been developed by leveraging the benefits of NCS and cloud computing technology. CCS presents a novel approach by shifting the core processing unit to cloud servers, facilitating massive parallel computation and efficient data storage for high loads [1]. The move to CCS reduces system size and offers energy-saving potential, providing an innovative solution to enhance efficiency and extend operational lifespans.

Although Cloud Control Systems offer numerous advantages, they also present new challenges that make industries hesitant to embrace cloud technology. Therefore, it is crucial to find solutions for these issues, which serve as impediments to fully harnessing the advantages of cloud technology. Our primary objective is to establish a resilient cloud control system capable of overcoming these challenges. Security is the first challenge introduced regarding the cloud control system since using the cloud and the network between the cloud and physical domain can create an access point for the attacker to intrude into the system and establish attacks. In our previous works [2], we proposed a solution to make an attack-resilient cloud control system such that the system can be kept stable in the presence of an attack with high efficiency.

Another significant concern arising from cloud integration is the potential for increased delays. Relocating the controller to the cloud means it will be physically distant from the domain it controls, resulting in longer data transmission times. In addressing this challenge, our previous work [3] established a framework for delay-resilient cloud control systems.

In this paper, we present an approach to enhance the resilience and efficiency of cloud control systems by emphasizing dynamic parameter adaptation. Unlike the conventional method of employing static parameters throughout, our proposal focuses on adjusting system parameters based on the current conditions. The initial impetus for this adaptation stems from the inherently dynamic nature of cloud environments, marked by continuously changing conditions and varying workloads [4], [5]. To enhance performance and resource utilization, it is essential that our control systems seamlessly adjust to this dynamic nature. Our research aims to align cloud systems with the evolving environment for peak efficiency.

Furthermore, in today's cloud computing landscape, the efficient allocation and utilization of resources with a frugal approach stands as a paramount concern [6]. Adhering to a static parameter configuration throughout carries the risk of resource wastage and performance bottlenecks. Therefore, by dynamically adjusting system parameters, we aim to achieve resource frugality, minimizing resource usage while maximizing system performance, aligning with the prudent resource management principles integral to modern cloud computing.

Another fundamental motivation for our approach emerges from traditional practices in parameter setting. Historically, many systems have relied on parameters established through trial and error or based on system operators' experience and knowledge of the system's physics, often neglecting

This paper was supported by the Celtic-Next project IMMINENCE funded by Vinnova, and the SSF project SEC4FACTORY under grant no. SSF RIT17-0032. The authors are part of the Excellence Center at Linköping-Lund on Information Technology (ELLIIT) strategic research area, and the Nordic University Hub on Industrial IoT (HI2OT) funded by NordForsk. Maria Kihl is partially funded by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

environmental considerations [7]. However, this conventional approach often falls short of ensuring the optimal parameter configuration. Additionally, when the same system is employed in different environments, it may not exhibit the same level of performance as previously anticipated. Consequently, we propose a new method for system parameters adjustment that strives to accomplish two pivotal goals: first, to ascertain the optimal parameter settings, and second, to automatically adapt these settings in response to changes in the environment and current conditions.

The primary factors contributing to the dynamic nature of the cloud environment are fluctuating delays, which are often linked to varying workloads within the cloud. Moreover, various factors like setpoint changes, disturbances, and noise fluctuations induce environmental variations. One of the critical system parameters susceptible to the dynamic nature of the cloud is frequency. Frequency directly impacts the workload within the cloud. When discretizing a control system to mimic its continuous counterpart closely, the preference is for a minimal sampling period. However, this preference leads to higher frequency, resulting in increased data transmission and potential network congestion. Moreover, it places a greater computational load on the cloud controller, further amplifying the cloud's workload. Therefore, in this paper, we aim to dynamically adjust the frequency based on real-time conditions, ensuring a higher frequency only when necessitated, such as during the presence of critical information.

A longer delay than the sampling period can result in the control signals received by the plant not aligning with the most recent plant output being sampled. Over time, this can elevate the effective control delay, potentially leading to system instability. Therefore, it is prudent to factor in the current delay when selecting the appropriate frequency. Authors in [8] and [9] present an adaptive sampling scheme designed to maintain a control delay shorter than the sampling period during steady-state operation. This scheme also relies on the system's maximum tolerable delay at a specific sampling interval to ensure a smooth transition between consecutive sampling periods, promoting system stability. In [10], an adaptive sampling scheme has been introduced for wireless sensor networks, aiming to notably diminish communications and consequently lower energy consumption. Most of these efforts have primarily centered around general-purpose signal processing and telecommunication systems, devoid of involvement in control applications.

Authors in [11], proposed a flexible time-triggered sampling approach for wireless control systems that dynamically adjusts sampling periods using smart sensors and feedback control technology. In [12], a frequency control approach is introduced for cloud control systems employing a PID (Proportional-Integral-Derivative) controller. In this study, the PID controller adjusts the sampling period to attain an acceptable rate of request failures, referred to as the miss ratio. The miss ratio signifies the percentage of requests (control signals) not received within the expected sampling period. So, most of these papers prioritize success rates by adjusting frequencies to ensure consistent control signal reception within the expected sampling time.

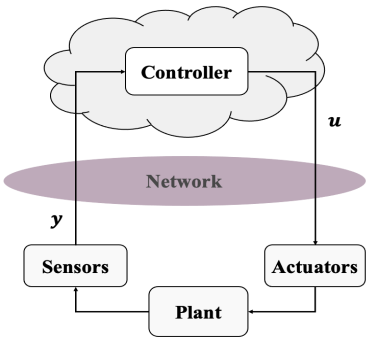


Fig. 1: Cloud Control System (CCS)

In this paper, we propose a frequency adaptation method for cloud control systems employing Q-learning. In contrast to other available approaches that rely on delay or miss ratio for frequency adjustments, our method derives information about the current environment's condition by measuring Round Trip Time (RTT), which includes network delay and execution time, and also by quantifying errors in the system's output arising from setpoint changes, noise, or disturbances. Additionally, we employ the Q-learning method to determine frequencies, which represents a novel approach compared to existing methodologies in this field. This method adjusts frequency to mitigate delay effects, boosting overall system efficiency and optimizing cloud resource use. It minimizes control costs by dynamically setting the frequency to suit the evolving cloud environment.

In the rest of this paper, we first introduce our targeted system which is the cloud control system. Then, we delve into our proposed solution, covering methodologies and techniques. Next, We detail the experiments conducted on an actual testbed, analyze the outcomes, and draw conclusions.

## II. Cloud Control Systems

Our targeted system in this paper is Cloud Control Systems (CCS), illustrated in Fig. 1. The system consists of a plant controlled by a controller implemented in the cloud. The measurement signal, denoted as $y$, is acquired by sensors and transmitted to the controller in the cloud. The controller employs this signal to compute the control signal $u$, and sends it to the actuators to apply on the plant [13], [14].

As discussed in Section I, regarding the dynamic nature of the cloud, it is not efficient to have a constant frequency. The cloud environment is characterized by varying workloads resulting from concurrent applications, leading to unpredictable delays in executing control signals. The chosen control frequency directly defines the sampling period $h$ that is used for discretizing the plant's model and designing a controller accordingly, as well as defining the time allotted for each control loop. As depicted in Fig. 2(a), the plant undergoes periodic sampling at intervals of $h$. During each sampling period, it transmits measurement signals $y$, and the

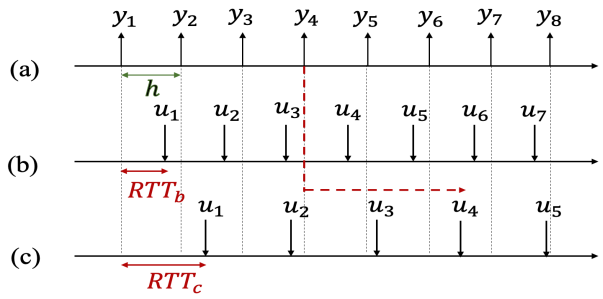


Fig. 2: Control signal availability in different amount of delay: (a) measurement sampling at h sampling period, (b) $RTT \leq h$, (c) $RTT > h$

corresponding control signals $u$ must be received within this interval before the next set of measurements is taken. In other words, the RTT should ideally be shorter than the sampling period, ensuring a timely data exchange.

If the combined delay $D$ in our CCS—encompassing both cloud execution and network transmission—results in an RTT shorter than $h$, as depicted in Fig. 2(b), the control signals aligned with the sampled plant outputs are delivered regularly at intervals of $h$. This occurs despite the existing delay, ensuring periodic updates to the plant with new control signals. However, if $RTT > h$ as depicted in Fig. 2(c), the plant is updated every $RTT$ instead of the expected $h$ intervals, despite the measurement signal is still sampled at $h$. This discrepancy arises because the received control signals do not correspond to the most recent plant output being sampled. For example, in Fig. 2(c), $u_1$ is received in the second sampling period after sending $y_2$ while we expect receiving $u_2$. Also, this delay results in an exponentially growing effective control delay from sampling to its corresponding control signal, potentially leading to instability over time. This instability can be attributed to the effective sampling period differing from the one used in controller design, thus deviating from the intended controller functionality. Consequently, the plant is updated every $RTT_c$, while the sampling occurs every $h$.

In Section I, we discussed another vital factor influencing frequency adaptation: its direct impact on the workload imposed by the controller on the cloud system. A higher frequency results in more frequent data accumulation from measurements, closely resembling continuous data streams. However, this also means sending data to the cloud more frequently, increasing the number of times the cloud must execute control signals. With a constant frequency setup, the system consistently transmits data without considering new information. Hence, frequency adjustment becomes pivotal based on conditions. If measurement signals remain relatively consistent, indicating minimal new information, a lower frequency suffices. Conversely, substantial fluctuations in measurement signals suggest a wealth of new data, demanding a higher chosen frequency. Therefore, it becomes essential to adapt the frequency based on the prevailing conditions.

## III. Proposed Solution

Q-learning is a fundamental reinforcement learning technique used to solve sequential decision-making problems [15]. It is particularly well-suited for scenarios where an agent interacts with an environment, taking actions to maximize cumulative rewards over time. In the context of cloud control systems and frequency adaptation, Q-learning can be employed to make intelligent decisions about adjusting system frequency to achieve desired performance goals. Q-learning relies on several key components: states, actions, rewards, and Q-table, which we'll delve into in the following Sections.

### A. States

The first key component in the Q-learning method is states. The states are feedback from our system and represent the system's current situation. Hence, regarding the fact that our goal is the resiliency of our cloud control system in different environments, we should use the parameters that can define the different environments (conditions) as the states in our Q-learning method. So, a state could include information about the current workload, resource utilization, and other relevant variables. In this paper, varying delay in the environment is very important for us, and we want to have a resilient cloud control system to delay. Hence, one of the states can be RTT in the system that is calculated as follows:

$$RTT = T_{up} + T_{exe} + T_{down} \tag{1}$$

where $T_{up}$ is the time to send the measurement signal from the plant to the controller, $T_{down}$ is the time for transmitting a control signal from the controller to the plant, and $T_{exe}$ is the execution time for computing the control signal in the cloud. So, based on this, after sending the measurement signal, we should wait to receive the corresponding control signal to calculate RTT. Then, the newest RTT we have access to is related to the latest control signal we have received.

The other factor that we want to adjust the parameters of the cloud control system according to is the output error that can happen due to the setpoint change, disturbance, noise, etc. For instance, in normal condition when there is no disturbance, noise and the setpoint is constant, it is reasonable to have a lower frequency since there is no new information to update the system. However, when there is a setpoint change, there is more new information that the system should be updated about, and we probably need to have lower sampling time and higher frequency. For considering such information, we will consider error as the other state in our Q-Learning algorithm and it is calculated as follows:

$$e_k = x_k - setpoint_k \tag{2}$$

where $x$ is the state variable of the control systems.

### B. Action Space

Actions are the choices available to the agent in a given state. Regarding our purpose in this paper, which is adjusting the system's frequency based on the current condition, our action is the frequency that should be used in the current control loop.

Not all frequencies are suitable for every system; each system typically operates within its own acceptable range of frequencies. We determine the acceptable frequency range for the system based on [8], and each of the frequencies in this range is defined as an action for the Q-Learning. The choice of frequency plays a crucial role in system performance. In an ideal scenario without considering delay and limitations in cloud resources, higher frequencies are preferred for optimal performance. However, practical constraints, such as the controller's implementation capabilities and control delay, can limit how high the frequency can be. As frequency decreases, system performance generally deteriorates. We can set a threshold for allowable performance degradation. A common metric used is Integrated Accumulated Error (IAE) to define the acceptable frequency range for a system. By conducting experiments with different frequencies, starting from the maximum value "$f_{max}$" determined by plant controller characteristics and incrementally decreasing frequency, the IAE is measured over a fixed duration. Based on the allowable IAE range, an appropriate frequency range is chosen. In this paper, considering the testbed that is explained in Section IV-A, the range of the frequencies that are considered as action space is: $f \in \{10 \text{ Hz}, 20 \text{ Hz}, 30 \text{ Hz}, 40 \text{ Hz}, 50 \text{ Hz}\}$.

A frequency from this set is chosen by the Q-Learning and is sent to the controller. Then the controller should be redesigned based on that and generate the corresponding control signal to send to the plant. Also, the sampling period for that loop is set based on this chosen frequency.

### C. Reward Design

In the Q-learning method, a reward function should be designed such that quantifies the system's performance based on the chosen objectives. This reward function guides the learning process by providing feedback to the agent. This paper aims to increase the system's efficiency, which can be measured by investigating the control cost. Hence, in this Q-learning problem, the reward is considered $-J$ where $J$ is the control cost function. By considering this reward, we will have a higher reward for a lower cost. As it is explained in Section IV-A, a Model Predictive Controller (MPC) is considered in our testbed. Hence, by considering this controller, the control cost can be calculated as $J$ in equation (4) in section IV-A2. Following (4), in MPC, the control signal is determined to minimize the cost function while accounting for constraints. Our Q-learning solution is applicable to any type of CCS, and the definition of control cost and reward can vary depending on the specific controller used in CCS.

### D. Q-Table and Q-Values

In Q-learning, Q-table plays a pivotal role in the decision-making process. It is a data structure that keeps track of the learned values for each state-action pair, reflecting the agent's knowledge of the expected cumulative rewards. The Q-table is a 2D array showing states in rows and actions in columns. Each cell holds a Q-value, estimating the expected cumulative reward by taking action 'a' in state 's' following the optimal policy. These values update throughout the learning process.

As explained in Section III-A, RTT and error are considered as states, but since these are continuous variables, we need to discretize them to create a Q-Table. For this, we consider $RTT \in [0 \text{ ms}, 250 \text{ ms}]$, and $error \in [0, 1]$ m, and then we break these ranges into 20 different discrete values for each of these two states. Also, discrete actions are needed to create the Q-table, and as explained before in Section III-B, we can define five different actions based on different frequencies. Hence, our Q-table's size will be $20 \times 20 \times 5$.

The Q-table is initialized with random values at the beginning of the learning process. Then, during the learning process, Q-values are updated based on the rewards received and the learned estimates of future rewards. This learning process is done in two parts. The first part is called exploration, which involves trying out different actions randomly, even if they are not currently considered optimal, to gather information about their effectiveness. Exploration is crucial for discovering potentially better actions and improving the agent's knowledge of the environment. The second part, called exploitation, involves choosing actions that the agent believes are currently optimal based on existing knowledge and have the highest Q-value. Exploitation aims to maximize short-term rewards by sticking to known good actions. Over time, these updates lead to the Q-table converging to more accurate estimates of optimal action values, and after the learning process, the final Q-table will be used for our intent.

## IV. Experiments

In this section, we detail the testbed used for our proposed method, followed by evaluation from three distinct aspects.

### A. Real Testbed

To assess the effectiveness of our proposed frequency adaptation method, we implement it on a real testbed, detailing each component in the subsequent Sections.

*1) Plant:* In our experiment, we consider a ball and beam system as a plant consisting of a long beam where a ball moves back and forth. This inherently unstable system causes the ball to swing and potentially fall off the beam. To ensure stability, the controller adjusts the beam's angle using an electric motor, aiming to maintain the ball at a specified position on the beam. The beam is 1.1 meters long, and the allowable ball position ranges from $-0.55$ meters to $0.55$ meters. This system involves three measured signals: ball position ($x_1$), ball speed ($x_2$), and beam angle ($x_3$). The continuous-time model for this process is outlined below:

$$\dot{x}(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -\frac{5g}{7} \\ 0 & 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0.44 \end{bmatrix} u(t) \qquad (3)$$

where $g = 9.80665$ is the gravity of Earth [16]. We discretize this continuous time model with a sampling time of $h$ for designing the controller.

*2) Controller:* We design a Model Predictive Controller (MPC) to ensure stability and regulate the ball's position in the ball and beam system [17]. As mentioned earlier, the continuous model of the plant described in (3) needs to be discretized using the sampling period $h$ to design the controller. When the frequency is not constant, $h$ becomes variable as well. Consequently, the controller must be reconfigured in each control loop to correspond to the discretized system's model based on the specific sampling period $h$ of that loop. At each sampling instant $k$, the control action $u$ is determined by solving an open-loop optimal control problem with a finite horizon ($N$). This problem utilizes the current state of the plant as the initial state, as follows:

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}} \quad & J = \sum_{i=k}^{k+N-1} x_i^T Q x_i + u_i^T R u_i + x_{k+N}^T P x_{k+N} \\ \text{subject to} \quad & x_{i+1} = A x_i + B u_i \\ & G \begin{bmatrix} x_i \\ u_i \end{bmatrix} \leq g, \quad H \begin{bmatrix} x_i \\ u_i \end{bmatrix} = h, \quad x_{n+k} \in \mathcal{T} \end{aligned} \tag{4}$$

where $J$ is control cost function, $Q$, $R$ and $P$ are cost matrices, $A$ and $B$ define the discrete model of the system, $x$ is the state vector, $u$ is the control signal, and the constraints of the system are defined by the matrices and vectors $G$, $g$, $H$ and $h$. We deployed this controller in a Kubernetes cluster which will be described in the following Section. In this approach, MPC in each loop generates a sequence of $N$ control signals, with the first signal applied to the current loop and the subsequent signals intended for future loops. In this paper, we use the predicted control signals to mitigate delays. When a specific control signal is delayed, the system uses the most recent received control signal sequence. By considering the delay duration and the sampling period of the latest received signal sequence, the system determines which element in that sequence corresponds to the current loop.

*3) Kubernetes Cluster:* The testbed includes a seven-node Kubernetes cluster functioning as the edge cloud. Kubernetes (K8S) is an open-source platform renowned for managing containerized workloads and services, offering declarative configuration and seamless automation capabilities [18]. The cluster incorporates both an Nginx ingress [19] and a Prometheus operator [20]. The Nginx ingress is exposed through the K8S NodePort paradigm. The Kubernetes cluster serves as the edge cloud to implement the controller.

To implement the Q-learning method on this testbed, during the learning phase, each learning episode involves running the control system for 100 seconds. We operate the control system, utilizing square waves for both setpoint and delay, where the intervals and amplitudes of these waves are randomly chosen to encompass all feasible states. The cumulative rewards across all control loops during this 100 seconds are calculated. During the initial one-third of episodes, exploration occurs, involving the random selection of frequencies for each control loop. Subsequently, in the remaining episodes, exploitation takes place, where frequencies are chosen based on the highest Q-value obtained through the learning process. The learning process continues until the reward converges.

### B. Evaluation

In order to evaluate our proposed method, we establish three different experiments that are stated in the following.

*1) Average Behavior of the System:* In the initial phase of our evaluation, we contrast the system's average behavior using our proposed method against scenarios employing constant frequencies from the acceptable frequency range $\{10\ \text{Hz}, 20\ \text{Hz}, 30\ \text{Hz}, 40\ \text{Hz}, 50\ \text{Hz}\}$. Also, we assess the workload imposed by each case on the cloud, aiming to appraise the effective utilization of cloud resources.

For this purpose, we explore the performance of the cloud control system under normal conditions without introducing additional delays. The experiments entail running the system six times, each time for 100 ms. The initial run involves employing frequency adaptation through our proposed methodology, while the subsequent five runs maintain constant frequencies within an acceptable range, gradually decreasing from high to low settings. In order to conduct a comprehensive assessment of control performance, we consider a constant setpoint but we add some strong disturbances that enter the system as follows:

$$x_{k+1} = A x_k + B u_k + \begin{bmatrix} 0 & w_k & 0 \end{bmatrix}^T \tag{5}$$

i.e., adjusting the speed of the ball. Here, $A$ and $B$ are the state space matrices discretized using very small sampling time ($h = 5$ ms) closely resembling the continuous system. Hence, choosing a higher frequency for the controller allows quicker detection and response to disturbances. The sequence of disturbances is,

$$w = \begin{bmatrix} 0 & \dots & w_0 & 0 & \dots & w_1 & 0 & \dots \end{bmatrix} \tag{6}$$

where the values of $w_i$ are drawn from a normal distribution $\mathcal{N}\left(\mu_w, \sigma_w^2\right)$ which $\sigma_w^2 = 0.001$ and $\mu_w$ is randomly selected from the interval $[0.003, 0.01]$, and these disturbances are applied at random time to the system for variable durations ranging from one second to four seconds. As a performance metric to compare the error in each case, we will measure the total cost for 100 ms using $\sum_{k=0}^{N} J_k$ that in each loop $J$ is calculated using the cost function in (4). Additionally, we assess the maximum error during 100 ms, and Integral Absolute Error (IAE). However, due to disparate sampling times across experiments, to be able to compare the IAE of different 6 experiments, unifying the IAE comparisons necessitates revising IAE calculation by considering sampling time as follows :

$$IAE = \sum_{k=0}^{N} |y_k - s_k| \Delta t_k \tag{7}$$

where $y_k$ represents the positional signal, while $s_k$ denotes the setpoint for the ball's position along the beam. The term $\Delta t_k$ signifies the corresponding variable sampling time, specifically varying within our proposed solution while remaining constant across other cases. Additionally, to measure the efficiency of cloud resource utilization, we measure the total

execution time in the cloud over the entire 100 ms, providing insight into the extent of cloud utilization.

*2) Challenging the System:* In the second part of the evaluation, we challenge the cloud control system by considering a square wave setpoint that is changing randomly and also applying a delay higher than the normal condition encountered in the last part. This involves two key steps: We establish the tolerable range of delay that the system, without our proposed method, can endure without destabilizing. We incrementally increase the delay, observing how the system reacts. This experiment is carried out under two conditions: first, with a constant frequency representing the traditional approach, and second, using our proposed frequency adaptation method. By comparing the system's performance in these scenarios, we assess the effectiveness of our proposed method in mitigating the effects of elevated delays.

*3) Dynamic Workload Evaluation:* In the final phase of our experiment, we aim to assess the cloud control system's performance under varying workloads within the cloud environment. To accomplish this, we implement the controller on a Kubernetes cluster that is used as the edge cloud in our experiment. By deploying pods designed for heavy computational tasks, we intentionally induce a load within the cluster for a specified duration. Our methodology involves executing a Python script dedicated to identifying prime numbers within a vast numerical range, necessitating substantial computational resources. We adjust the cloud load based on the number of pod replicas we deploy.

In our experimental setup, we execute the proposed frequency adaptation mechanism and contrast its performance against experiments conducted by using a constant frequency $f = 20$ Hz. We exclusively compare our method with the constant frequency of $f = 20$ Hz, which stands as the prevailing frequency utilized in the testbed, as documented in [2], [3], [16]. This comparative analysis allows us to evaluate the efficacy of our proposed control in handling dynamic workload variations within the cloud infrastructure. The MPC controller employs online optimization, as represented by equation (4), to generate the control signal. For a fixed horizon time $N_t$, the execution time increases with higher frequencies due to an increase in the number of iterations within the (4). Specifically, the number of iterations $N$ is determined by the equation $N = \frac{N_t}{h}$, where $h = \frac{1}{f}$ represents the sampling period corresponding to the frequency $f$. $N_t$ represents a continuous variable denoting the duration of time, while $N$ corresponds to its discrete counterpart determined based on the chosen sampling period. Augmenting the workload within the cloud environment has the potential to elevate response times, thereby prolonging the reception duration for the control signal. This delay can significantly influence the performance of the control system. Therefore, we examine the efficacy of our proposed frequency adaptation method to assess its responsiveness under varying workload conditions.

## V. Results and Discussion

As detailed in Section IV-B1, the initial phase of the experiment involves assessing the average behavior of the system using our proposed method across two key aspects: the performance of the control system and the efficient utilization of cloud resources in normal condition without introducing additional delays. In the experiment in Fig. 3, the RTT has a normal distribution $\mathcal{N}\left(\mu_r, \sigma_r^2\right)$, which $\sigma_r^2 = 6.9051e-06$ $s^2$ and $\mu_r = 0.0115$ $s$ which indicates a normal condition with no significant delays. However, occasional disturbances are observed, leading to deviations in the ball's position from its designated setpoint and consequently resulting in errors.

The green curves in this figure depict the results obtained through our proposed frequency adaptation method within CCS. This method strategically maintains a high frequency solely when necessary, prioritizing lower frequencies to minimize cloud workload and reduce network traffic. When disturbances occur, causing errors in the ball's position, the Q-learning method responds by opting for a higher frequency. Increased frequency allows the controller to rapidly receive new information and detect errors, enabling quicker reactions for rectification. Consequently, as illustrated in Fig. 3, when disturbances are present, the error in the ball's position using our proposed method is so close to the scenario employing a constant high frequency ($f$=50 Hz), which exhibits minimal error. Despite the error similarity, the total execution time—reflecting cloud resource utilization for control signal generation—is significantly reduced compared to the constant high-frequency case ($f$=50 Hz), as shown in Fig. 4(a), leading to minimized resource usage and enhanced resource frugality.

Examining Fig. 4, the experiment employing a constant $f = 10$ Hz showcases the lowest total execution time and the number of control loops, indicating minimal workload on the cloud infrastructure. However, this scenario exhibits the highest Integral Absolute Error (IAE), "maximum error",

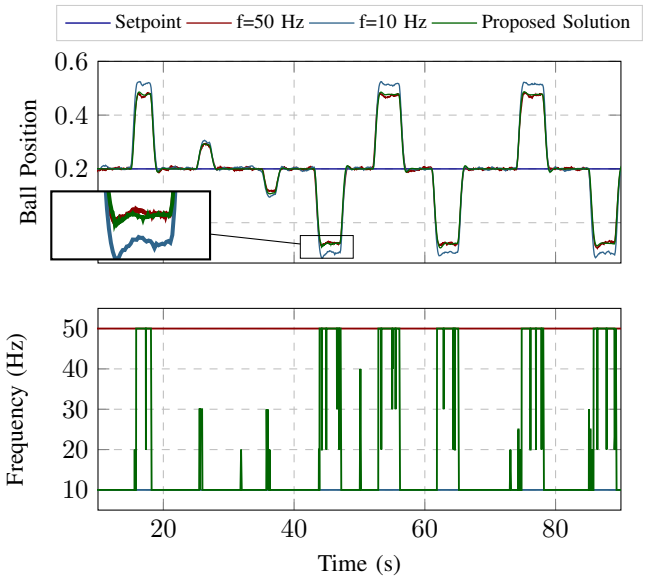


Fig. 3: Examining Average behavior of the system

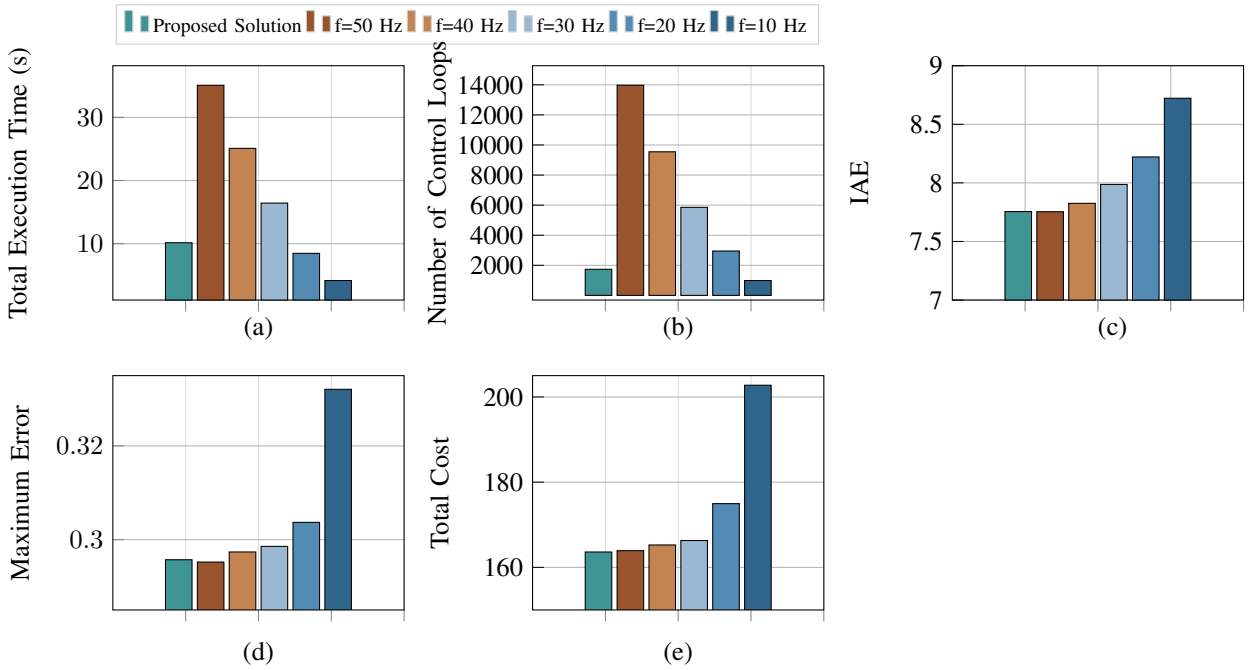


Fig. 4: Comparing the proposed solution with constant frequencies for different performance metrics: (a) Total Execution Time, (b) Number of control loops, (c) IAE, (d) "maximum error", and (e) Total cost.

and total cost. These results stem from reduced accuracy in discretizing the system due to a larger sampling time, leading to delayed error detection and subsequent reaction. Conversely, the experiment with a constant $f = 50$ Hz demonstrates the highest total execution time and the number of control loops, signifying a heavier cloud workload. Yet, this configuration yields the lowest IAE, "maximum error", and total cost due to its higher accuracy in discretizing the system with a smaller sampling time. This allows for quicker error detection and more rapid reactions. Upon examining the green bars in Fig. 4, evaluating our proposed solution, a clear trend emerges: the total execution time and number of control loops are significantly lower compared to the highest frequency setting ($f = 50$ Hz). However, in terms of IAE, "maximum error", and total cost, our proposed solution performs comparably well. This observation highlights the effective control performance achieved by our proposed solution, akin to the highest frequency configuration, while notably reducing cloud resource utilization. For instance, in a 100 ms experiment, the constant $f = 50$ Hz setting demands approximately 35.09 s of execution time, whereas our proposed solution requires only around 10.16 s. Hence, adopting our proposed solution facilitates resource efficiency, striking a balance between minimized resource consumption and optimized system performance.

The second part of our evaluation, as outlined in Section IV-B2, involved challenging the system with delays beyond tolerable limits and setpoint change. We introduced network delay on the downlink between the cloud-based controller and the plant in the form of a random square wave, where both the amplitude and intervals were randomly chosen, such that RTT changes between 100 ms and 210 ms. As setpoint we also consider a square wave with random amplitude and intervals. In Fig. 5, the red curve represents the CCS results without our solution, operating at a constant frequency of 20 Hz. Notably, when the RTT exceeds 150 ms, the system's efficiency deteriorates, causing the ball's position to deviate from the setpoint. Once the RTT surpasses 200 ms, the system becomes unstable, leading to the ball falling off the end of the beam. Conversely, the green curve in Fig. 5 represents the CCS results using our frequency adaptation method. It is evident that the system can survive even when the delay exceeds 200 ms, and it exhibits better efficiency for delays over 150 ms compared to the system using a $f = 20$ Hz.

In our last experiment (Section IV-B3), we thoroughly test how well the system performs with increased cloud workload. Increased workload in the cloud can lead to resource contention, where multiple processes or applications compete for the same resources. This contention can delay the allocation of resources required for processing requests, leading to higher response times. In this experiment, there is a network delay of about 100 ms. Then, we introduce extra applications in the cloud for a period, intentionally causing the cloud's response time to increase. In Fig. 6, the highlighted red area represents the timeframe during which the response time increased, attributable to the escalating workload within the cloud. This leads to a longer execution time required for the cloud to generate the control signal and consequently increases RTT. Employing a constant frequency of $f = 20$ Hz, adversely impacts system performance. As evident in the highlighted red area, the longer response time results in significant deviations of the ball position from its setpoint.

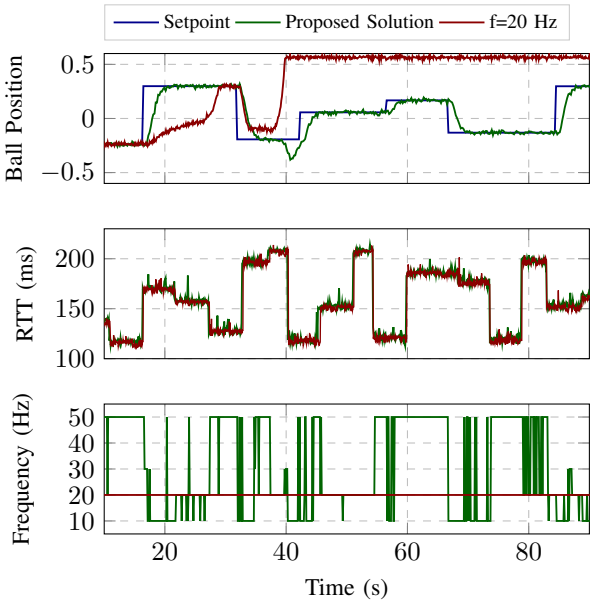


Fig. 5: Challenging the system: delay and setpoint variation

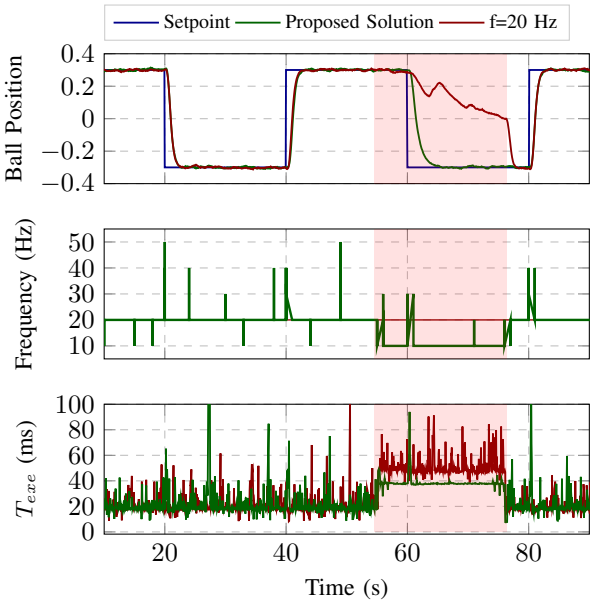


Fig. 6: Testing system resilience: increasing cloud workload

This discrepancy arises from the higher execution time of $f = 20$ Hz. However, our proposed frequency adaptation method responds to the increased response time by lowering the frequency. As detailed in Section IV-B3 and equation (4), lower frequencies result in reduced execution time. Therefore, our method aims to decrease the response time by selecting $f = 10$ Hz, employing a higher frequency only in cases where a setpoint change induces an error. In Fig. 6, with our proposed method, during the highlighted red area where the response time increases, the execution time remains lower compared to the constant $f = 20$ Hz scenario, and the ball follows its setpoint well proves the efficiency of our method.

## VI. Conclusion

Accounting for the dynamic nature of cloud environments, employing a fixed frequency in cloud control systems proves to be inefficient. To address this challenge, this paper introduced a novel frequency adaptation method employing Q-learning. This method dynamically selects the frequency for each control loop based on the current environmental conditions. Our evaluation, conducted on a real testbed, demonstrates that this approach not only improves overall system efficiency but also optimizes cloud resource utilization.

## References

[1] Y. Xia, Y. Zhang, L. Dai, Y. Zhan, and Z. Guo, "A brief survey on recent advances in cloud control systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 7, pp. 3108–3114, 2022.

[2] F. Akbarian, W. Tärneberg, E. Fitzgerald, and M. Kihl, "Attack resilient cloud-based control systems for industry 4.0," *IEEE Access*, vol. 11, pp. 27 865–27 882, 2023.

[3] H. Peng, F. Akbarian, W. Tärneberg, and M. Kihl, "Punctual cloud: Achieving punctuality for time-critical cloud control systems," in *12th IEEE International Conference on Cloud Networking, IEEE CloudNet 2023*, 2023, accepted/In press.

[4] B. Feng, Z. Ding, and C. Jiang, "Fast: A forecasting model with adaptive sliding window and time locality integration for dynamic cloud workloads," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1184–1197, 2022.

[5] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Computing*, vol. 23, no. 4, pp. 2399–2424, 2020.

[6] H. Shen and L. Chen, "A resource-efficient predictive resource provisioning system in cloud systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3886–3900, 2022.

[7] A. M. de Paor and M. O'Malley, "A describing function technique for sampling period selection and controller discretisation," *Transactions of the Institute of Measurement and Control*, vol. 15, no. 4, pp. 207–212, 1993.

[8] L. Samaranayake, M. Leksell, and S. Alahakoon, "Relating sampling period and control delay in distributed control systems," in *EUROCON 2005-The International Conference on" Computer as a Tool"*, vol. 1. IEEE, 2005, pp. 274–277.

[9] L. Samaranayake, "Delay compensation, design and simulation of controllers for distributed control systems," in *First International Conference on Industrial and Information Systems*. IEEE, 2006, pp. 491–496.

[10] R. Willett, A. Martin, and R. Nowak, "Backcasting: adaptive sampling for sensor networks," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, 2004, pp. 124–133.

[11] F. Xia and W. Zhao, "Flexible time-triggered sampling in smart sensor-based wireless control systems," *Sensors*, vol. 7, no. 11, pp. 2548–2564, 2007.

[12] W. Tärneberg, P. Skarin, K.-E. Årzén, and M. Kihl, "Resilient cloud control system: Realizing resilient cloud-based optimal control for cyber-physical systems," *arXiv preprint arXiv:2304.00857*, 2023.

[13] Y. Xia, "Cloud control systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 2, pp. 134–142, 2015.

[14] M. Sajjadi and H. Seifi, "Governor parameter estimation considering upper/lower production limits," in *2019 IEEE Milan PowerTech*. IEEE, 2019, pp. 1–6.

[15] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[16] F. Akbarian, W. Tärneberg, E. Fitzgerald, and M. Kihl, "A cloud-control system equipped with intrusion detection and mitigation," *Electronic Communications of the EASST*, vol. 80, 2021.

[17] P. Skarin, J. Eker, M. Kihl, and K.-E. Årzén, "Cloud-assisted model predictive control," in *2019 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2019, pp. 110–112.

[18] Kubernetes, url: https://kubernetes.io.

[19] Ingress-nginx, url: https://github.com/kubernetes/ingress-nginx.

[20] Prometheus-operator, url: https://github.com/coreos/prometheus-operator.