International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18 (2026) pp. 1-24

https://doi.org/10.70917/ijcisim-2026-1214
Article

Analysis of Delay in Internet of Things Enabled Network: A State Space Control Perspective

Padmaja Mishra [1], Ajay Kumar Yadav [2], Rajesh Kumar Patjoshi [*,1] and Rakhee Panigrahi [3]

[1] Dept. of Electronics and Communication Eng., NIST University, Pallur Hills, Berhampur, Odisha, India; spm.misra@gmail.com
[2] Dept. of Electronics and Communication Eng., C.V. Raman Global University, Bhubaneswar, Odisha, India; ajayyadav@cgu-odisha.ac.in
[3] Dept. of Electrical Eng., Parala Maharaja Engineering College, Berhampur, Odisha, India; rpanigrahi99@gmail.com
* Correspondence author: rajeshpatjoshi1@gmail.com

Abstract: This paper presents a novel control framework designed to address critical challenges of delay and packet loss in networked control systems using edge-cloud integration. The proposed solution utilizes a hybrid approach combining a predictive state-space proportional (PSSP) controller and an adaptive state-space proportional-integral (ASSPI) controller. The predictive controller uses a Luenberger observer to estimate system states, enabling it to proactively compensate for network-induced delays and ensure real-time stability. The adaptive PI controller, a core component of the framework, employs a machine learning algorithm based on Soft Actor-Critic (SAC) with a Long Short-Term Memory (LSTM) network to predict future system states and dynamically adjust control parameters. This intelligent adaptation is further enhanced by integrating Recursive Least Squares (RLS) and an Unscented Kalman Filter (UKF) with an adaptive pole placement technique, providing robust, real-time parameter identification and accurate state estimation. The combined approach demonstrates superior performance in mitigating the effects of variable network latency and data loss, offering a reliable control strategy for latency-sensitive applications. The results show that this integrated compensator is highly effective, paving the way for more resilient and high-performance edge-cloud control systems.

Keywords: Edge-cloud, Adaptive Pole placement, State-space, SAC, Delay compensator, RLS, Packet loss

1. Introduction

In modern industrial and cyber-physical systems, networked control systems (NCS) have become essential due to their adaptability and scalability. These systems, which depend on data exchange over shared communication networks, face critical challenges such as network-induced delays and packet loss. These factors can significantly impair performance, undermine stability, and, in extreme cases, cause system failure. As a result, developing resilient control strategies to address these communication flaws is a significant research focus. Moreover, network performance is essential for the reliable functioning of internet-based services and applications. Metrics including network delay (latency) and packet loss, directly influence both user experience and overall service quality [1]. However, communication and control are closely linked in networked control systems (NCS), and the design of one can significantly impact the performance of the other. The co-design of control and communication scheduling in NCSs has received considerable attention. This paper reviews recent progress in this area. This article first established a basic framework for the co-design problem, then examine key results and methodologies

from the literature. This article also provides analysis and discussion of various scheduling schemes, including static, dynamic, and random scheduling [2]. However, a wide range of methods has been proposed to address network-induced delays. Techniques such as model-based delay compensation have shown promise, utilizing predictive algorithms to anticipate future system states and apply corrective actions in advance [3], [4]. Furthermore, state estimation is essential for managing systems with incomplete data. Observers, such as the Luenberger observer, are used to reconstruct system states from available measurements, which supports predictive control when data is lost [5], [6].

Rapid advancement of Internet of Things (IoT) has made significant progress in multiple sectors, including industrial automation, healthcare, and smart cities, by providing seamless connectivity and intelligent control among various IoT devices. Under these circumstances, Networked Control Systems (NCS) become an essential technology by integrating network communication and control processes in a single platform to achieve adequate control among multiple IoT devices. However, the integration of IoT with NCS presents significant challenges specifically "network-induced delays", which can severely impact the efficiency and reliability of NCS. Moreover, delays in the IoT network can originate from various sources, including network congestion, processing delays, and data transmission latency [7]. Failure to address these delays compromises control effectiveness, leading to reduced accuracy, instability, and potential malfunctions. Minimising such delays is essential for IoT-enabled networked control systems to function as intended. Traditional IoT communication methods often degrade performance due to excessive data volumes that overwhelm network resources, resulting in congestion and transmission delays. Additionally, frequent transmissions elevate the energy consumption of IoT devices. This survey examines strategies for efficient and collaborative communication within the IoT–edge–cloud continuum. It explores techniques to minimize data generation and optimize communication protocols, thereby alleviating network traffic. However, this paper lacks quantitative benchmarking and in-depth analysis for delay-sensitive control applications. It also underemphasizes device constraints, security trade-offs, and emerging paradigms like semantic communication [8].

Traditional control methods often require an accurate mathematical model of the system. In contrast, model-free reinforcement learning can control a system without a model, but does not guarantee stability. This paper addresses how to leverage model-free, data-driven reinforcement learning while ensuring system stability and strong performance [9]. Decentralized queue control with delay shifting in edge–IoT applies reinforcement learning (RL) to reduce delay and packet loss under dynamic traffic is discussed in this article [10]. Each edge node acts as an agent, managing queues and offloading decisions based on local states. Delay shifting redistributes latency away from critical flows, while RL reward functions penalize delay and buffer overflow. This enables adaptive queue management that lowers tail delay, prevents congestion-induced packet loss, and improves scalability compared to static or centralized methods. Furthermore, this article proposes a new edge computing-based framework for IoT data processing and scheduling using deep reinforcement learning. The system architecture features distributed IoT data access, real-time processing, and an intelligent scheduler based on Deep Q Networks (DQN). Experimental results demonstrate that, compared to traditional scheduling methods, the framework reduces average task completion time by 20% and increases resource utilization by 15%. This integration of edge computing and deep reinforcement learning offers a flexible and efficient platform for low-latency IoT applications [11]. However, integrating learning-based controllers with model-based compensators in edge-cloud NCS continues to present significant technical challenges [12]-[16]. Furthermore, delays in environments, known as delayed MDPs (Markov decision processes), might cause the agent to not perceive the state immediately (observation delay, ΔT_{si}), lag in deciding on an action (inference delay, ΔT_{si}), and delay in the action's consequence (execution delay, ΔT_{si}). The entire learning process is hampered by these delays, which compel the agent to base decisions on outmoded knowledge and refrain from taking prompt action. Different forms of delays can be treated equally because, fortunately, only the total delay $\Delta_{T_d} = \Delta T_{si} + \Delta T_{id} + \Delta T_{id}$ matters for decision-making [17].

Accordingly, the Soft Actor–Critic (SAC) algorithm is a reinforcement learning method that develops robust, entropy-regularized policies. While SAC does not explicitly account for system delays, its stochastic action selection and off-policy updates can reduce the impact of sensor-to-controller and controller-to-actuator delays. As a result, SAC demonstrates improved performance and robustness in Edge-IoT systems relative to fixed or adaptive controllers. Moreover, SAC is a stochastic policy-based DRL technique that adds entropy to improve the resilience of policy optimization. Several automatic control system applications have confirmed the SAC algorithm's efficacy. In reinforcement learning applications, action and observation delays are a prevalent problem, especially in networked and remote-control systems. A recent development of Soft Actor-Critic (SAC), the Delay-Correcting Actor-Critic (DCAC) partially resamples trajectory fragments for off-policy multi-step value estimation, hence improving performance in delayed situations. Our research into SAC-integrated distributed PI control with UKF and RLS in edge–cloud IoT systems is further motivated by the necessity for

delay-aware RL frameworks [18]. Moreover, this article is introduced as an intelligent routing algorithm, SAG, that integrates graph neural networks for topology learning, attention mechanisms for quality of service prioritization, and soft actor-critic methods for adaptive decision-making in CIoT edge networks. This approach directly enhances *delay, packet loss, and throughput metrics*. However, SAG encounters challenges related to scalability, training complexity, and practical deployment [19].

Nonetheless, in the controller–actuator channel, the uncertainty of network delay compels the controller to generate an extended control sequence to ensure reliability. However, this practice significantly increases the volume of transmitted data, thereby imposing an additional burden on the network. However, to reduce the amount of transmitted data, a long short-term memory (LSTM) neural network is first adopted to predict the network delay of the packet to be sent is illustrated in this article. Then the controller uses the predicted network delay to determine the prediction horizon [20]. Moreover, using real-world data gathered from various sources in an Internet of Things (IoT)-enabled public transportation agency, this research addressed the establishment of a hybrid deep learning architecture for long-term train delay prediction. While CPSs (Cyber-Physical Systems) employs anticipated values with a nominal schedule to detect primary and secondary delays based on the delay causes, run-time delay, and dwell time delay, LSTM handles long-term prediction tasks [21]. A routing method utilizing *link correlation mining* is introduced to enable *energy-efficient and reliable routing decisions* in wireless edge-IoT networks, resulting in improved delay, packet delivery, and energy consumption is described in this article **[22].**

Moreover, in this study, packet repetition techniques for erasure channels with memory and a large feedback delay are studied. The issue is initially presented as a communications situation where a source seeks to send a single message packet to a destination with the least amount of transmission count and delay possible. The sender must choose whether to send again after receiving delayed acknowledgment feedback about prior attempts at each time step. An agent tries to choose the best transmission policy based on delayed input in this problem, which is then framed as an episodic reinforcement learning task. In order to help the agent, a data-driven channel estimator models channel memory and forecasts erasure probability in subsequent time periods without requiring prior channel knowledge [23]. Furthermore, this paper introduces the split inference with no retransmissions (SI-NR) method, which maintains high prediction accuracy despite packet loss and without the need for retransmissions. The central approach involves training the machine learning model to emulate packet loss through a dropout technique that randomly omits outputs from hidden units in a deep neural network layer. Consequently, the SI-NR system demonstrates robustness to packet loss. Experimental results indicate that SI-NR achieves accurate predictions without retransmission, even when the packet loss rate reaches 60% [24].

However, time delay and packet dropout should be considered when evaluating a control strategy for the NCS that requires enhanced real-time features [25]. The state-space proportional (P) controller offers simplicity but results in steady-state error and is susceptible to issues caused by delay and packet loss. In contrast, the state-space proportional-integral (PI) controller eliminates steady-state error; however, it can cause overshoot, increase resource utilization, and heighten vulnerability to packet loss in Edge-IoT environments. For PMLSMs, the study suggests a Virtual Position Predictive Controller in conjunction with a System Delay Observer. In comparison to conventional controllers, this method improves tracking precision, stability, and resilience by compensating for both fixed and variable delays [26]. A learning-based adaptive optimum control approach for linear time-delay systems is described in this study. It ensures stability while adjusting for time delays, learns optimal controllers online without requiring complete system models, and employs a policy iteration (PI) framework with a Lyapunov–Krasovskii functional as the value function [27]. However, compared to traditional SSP under limited or partially known delays, the Predictive SSP controller in cloud-based setups uses prediction to offset communication delays, allowing for faster response times and superior performance. Its sensitivity to imprecise delay estimate is its drawback, which reduces its effectiveness in the presence of highly fluctuating cloud-induced jitter. The Adaptive SSPI controller, on the other hand, eliminates steady-state error and can manage dynamic and unpredictable delays by combining adaptive estimation and integral action. Nevertheless, because continuous adaptation and integral compensation add more delay to the cloud environment, this robustness comes at the expense of increased computing load and slower responsiveness. Moreover, suggests a cloud-edge collaborative containerized workflow in conjunction with a disturbance observer (DOB) for data-driven predictive control (DPC). uses DOB to account for uncertainty and divides work to help cut down on computation time [28]. This article presents a multi-tiered controller and a policy for deciding whether to execute control in the cloud or on the edge in order to handle trade-offs including stability, latency, and resource usage [29]. furthermore, in order to achieve low latency and great adaptability, this study describes how a cloud–edge hybrid architecture enables UGVs to track pathways precisely by utilizing the cloud for continuous parameter tuning and running MPC in real time at the edge [30].

Moreover, **Mohajeri** *et al.* [31] proposed a discrete-time modelling approach for NCSs that jointly considers *time-varying delays and random packet dropouts*. By augmenting the state-space with delayed states and modeling dropouts stochastically, their method provides a **unified framework** for stability and performance analysis. This is highly relevant to IoT and Edge–Cloud systems, where such uncertainties are common and can also represent attack-induced disruptions. This paper explains how to design controllers for networked control systems that remain stable and perform well even when the network causes delays and loses packets, using LMIs as the main design tool [32]. This article focuses on cyber-security in CPS, showing methods to detect malicious attacks and maintain system stability using fault-tolerant control strategies against simultaneous replay and false-data injection attacks [33].

In practice, a robust controller with an observer or estimator and a time delay compensator can be designed to achieve stable output control in an Edge-Cloud environment utilizing the Adaptive Pole placement technique. A well-designed hybrid edge-cloud system aims to ensure system stability by strategically placing the system's poles (which represent system response characteristics such as stability and speed, and should ideally be located inside the unit circle in control theory) by leveraging the low latency of the edge for time-sensitive tasks and the power of the cloud for more complex, non-real-time computations. In edge-cloud systems, maintaining stable output is critical for IoT applications. The system's control loops must provide dependable, consistent responses to real-world sensors and devices. Unstable behaviour can result from high latency in cloud-based processing or network congestion. This instability can lead to erratic device functioning, equipment damage, or safety risks. A hybrid architecture that utilizes low-latency edge processing is often preferred to maintain system stability.

This article investigates the design of a robust controller for an IoT environment, leveraging an adaptive pole placement technique to mitigate network-induced time delays and other challenges inherent in edge-cloud integration. The proposal focuses on developing a dependable controller that minimizes critical delay values by dynamically adjusting the system's poles to compensate for the varying latency between the edge and the cloud. To further enhance performance and optimize the system's architecture, a reinforcement learning algorithm is employed. This algorithm enables the system to intelligently decide which tasks should be processed at the low-latency edge and which should be offloaded to the powerful cloud, thereby autonomously reducing the total network-induced time delay and improving the overall performance of the networked control system (NCS). The followings are summary of our principal contributions:

(1) This paper discussed how in an edge computing environment, a Predictive state space Proportional controller, a Luenberger observer, and a delay compensator are integrated to proactively mitigate network delays and packet loss, ensuring a stable and reliable control system for real-time applications.

(2) This framework also makes use of an LSTM network, an Adaptive State-space PI controller, and the Soft Actor-Critic (SAC) reinforcement learning algorithm. The SAC, which automatically learns to modify the controller's parameters to provide steady performance and ideal resource management in a dynamic cloud environment, is informed by historical cloud data processed by the LSTM. In this integrated system, RLS and UKF manage the "perception" (i.e., precisely estimating the system's present state and parameters), while SAC and LSTM handle the "intelligence" of the control loop (i.e., learning the optimal control strategy).

(3) The controller's predictive nature gives it significant time delay resistance. The delay compensator is an additional layer. In the presence of time-varying or unknown delays, it improves the control approach and makes the system more resilient and stable. In this work, a Smith Predictor delay compensator is added. This guarantees the predictive controller's continued efficacy in the face of real-world complexity.

(4) Both the Predictive State space Proportional controller and the Adaptive State space Proportional Integral controller in this article utilizes an Adaptive pole placement technique. The Adaptive pole placement method in a Predictive State space Proportional controller configuration provides the Predictive controller with an accurate and continuously updated plant model, which constitutes an indirect adaptive control approach. The integration of edge-cloud architecture introduces new challenges and opportunities, such as enabling global optimization and learning, supporting low-latency control at the edge, facilitating computational offloading to the cloud, and addressing network delays and unpredictability. Adaptive State space PI controller, on the other hand, improves robust gain scheduling, lowers computing burden, improves the plant model, and refines the adaptive algorithms themselves. This makes it possible to learn and refine the control technique over time.

While several studies have been conducted to lessen the time delay caused by networks, research on the Edge-Cloud hybrid Internet of Things is underway. The structure of this paper is organized as follows. The system model and problem formulation are elaborated in Section II. Section III examines the fundamental concept and design methodology of the predictive and adaptive State-Space controllers.

Simulation results and analysis are presented in Section IV, and the final remarks are elaborated in Section V.

2. System Model and Problem Formulation

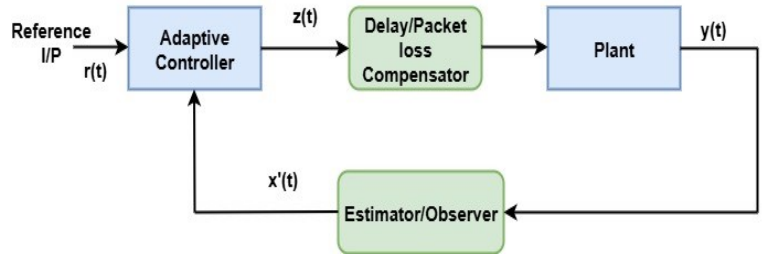

Figure 1. General structure of an Adaptive Control

This diagram shows an adaptive control system in a networked environment. The controller adjusts based on the estimated system state. A compensator manages delays and packet losses, ensuring reliable control despite network imperfections. The estimator closes the feedback loop and enables the system to robustly track the reference input r(t).

However, Adaptive control in edge–cloud systems dynamically responds to delays and packet loss. This differs from fixed controllers. Edge nodes provide rapid local adaptation. The cloud handles long-term learning and coordination. This approach enhances transient response, reduces steady-state error, conserves bandwidth, and supports scalability for IoT devices. This delivers robust, efficient, and reliable performance for critical applications such as automation, healthcare, and smart infrastructure.

The system process model depicted in Figure 1, designs a typical state-space network controller system in an Edge-cloud integrated environment utilizing the Adaptive Pole placement technique. The state-space method-based physical system, which is mainly like a non-linear time-variant one and is characterized by:

$$\begin{aligned} \dot{x}(t) &= f\left(x(t_i), z(t_i), t_i\right) + \varpi(t_i) \\ p(t) &= h\left(x(t_i), z(t_i), t_i\right) + \upsilon(t_i) \end{aligned} \tag{1}$$

where $x(t) \in \Re^n$ is a state vector (actuator/sensor states)), $z(t_i) \in \Re^m$, control input (from edge/cloud controller), $p(t) \in \Re^p$ measured outputs (fed back through IoT sensors), $f(\cdot) h(\cdot)$ is nonlinear, time-varying functions and $\varpi(t_i), \upsilon(t_i)$ is process and measurement disturbances (noise, packet loss, delay).

As we're not only modeling the physical plant but also the networked control structure**.** If we want to characterize a nonlinear, time-variant physical system under edge–cloud integration using the state-space method**,** the formulation typically looks like this:

At the Edge (for fast response, and limited resources), local controller compensates short-term variations.

$$z_e(t_i) = g_e\left(x(t_i), \hat{x}_e(t_i), t_i\right) \tag{2}$$

and, *at the Cloud* (for slower, high-computation, global optimization):

$$z_c(t_i) = g_c\left(x(t_i - \tau_i), \hat{x}_c(t_i), t_i\right) \tag{3}$$

where τ_i is the delay due to communication and scheduling.

Now, the actual control input applied to the system is a fusion of edge and cloud decisions:

$$\dot{x}(t) = \alpha(t_i) z_e(t_i) + \left(1 - \alpha(t_i)\right) z_c(t_i - \tau_i) + \varpi(t_i) \tag{4}$$

Where, $\alpha(t_i)$ balances *edge priority vs. cloud support*.

Finally, the Edge-Cloud integrated non-linear time-variant system can be written as,

$$\begin{aligned} \dot{x}(t) &= f\left(x(t_i), \alpha(t_i) z_e(t_i) + (1-\alpha(t_i)) z_c(t_i - \tau_i), t_i\right) + \varpi(t_i) \\ p(t) &= h\left(x(t_i), t_i\right) + \upsilon(t_i) \end{aligned} \tag{5}$$

Figure 2 illustrates an edge–cloud adaptive control framework designed to compensate for delay and packet loss. At the edge, a delay compensator, predictive SSP, and Luenberger observer provide real-time state estimation and rapid plant control. In the cloud, RLS and UKF improve parameter identification and state estimation, while adaptive SSPI uses SAC and LSTM (Long Short-Term Memory) for learning-based controller optimization. This hybrid approach integrates fast local response with intelligent cloud adaptation to enhance robustness, efficiency, and reliability in IoT systems.

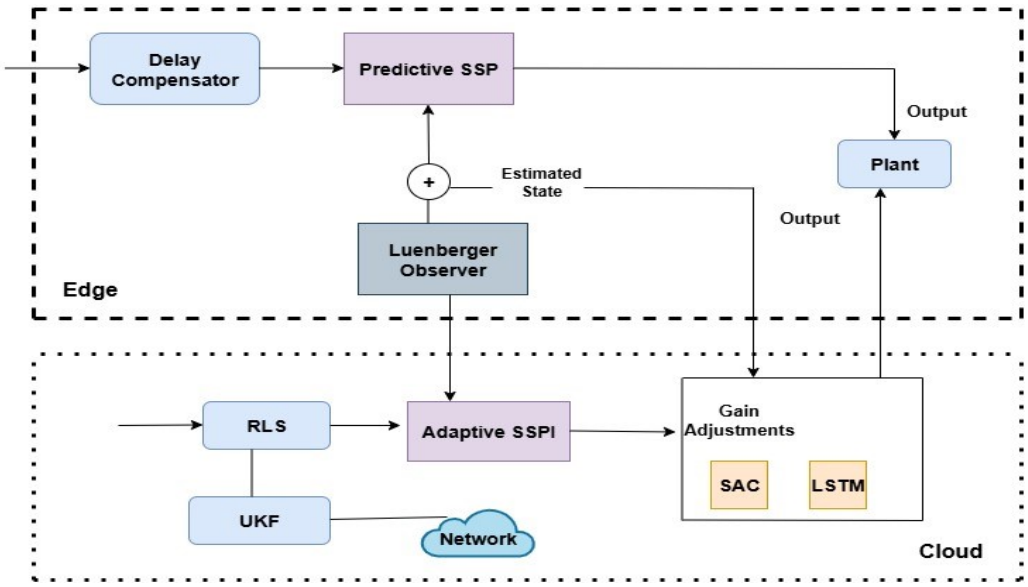

Figure 2. Proposed Hybrid Edge-Cloud Control System for Delay Compensation and Adaptive Gain Tuning

3. Design Methodology of State-Space Predictive and Adaptive Controllers

3.1. Predictive State space Proportional Controller Design for Improvement of Network Delay Performance

In the rapidly evolving landscape of modern edge-cloud integrated networked control systems, communication delays and packet loss are critical obstacles to achieving optimal stability and performance. Traditional Proportional and fixed-gain controllers are inadequate for handling time-varying delays, often leading to undesirable oscillations, poor transient response, and, in some cases, instability. To overcome these challenges, we present the innovative Predictive State-Space Proportional (PSS-P) controller. This cutting-edge solution leverages a Luenberger observer for precise state estimation, employs a Smith predictor to effectively counteract the effects of input delays, and features an adaptive pole-placement mechanism that dynamically tunes the proportional gain matrix in real-time. By integrating these advanced techniques, our framework significantly enhances delay tolerance and delivers robust performance. Below, we outline the mathematical formulation that reinforces this revolutionary design.

Step-1: *Linearize the nonlinear time-varying plant around the current operating point and model the network input delay* τ_i .

$$\begin{aligned} \dot{x}(t) &= A(t_i) x(t_i) + \mathrm{B}(t_i) z(t_i - \tau t_i) + \varpi(t_i) \\ p(t) &= C(t_i) x(t_i) + \upsilon(t_i) \end{aligned} \tag{6}$$

where $x(t) \in \Re^n$, $z(t_i) \in \Re^m$, $p(t) \in \Re^p$ and $\varpi(t_i), \upsilon(t_i)$ is process and measurement disturbances.

Step-2: *Keep a nominal delay-free model (used by Smith predictor / observer):*

$$\begin{aligned} \dot{x}_{mi}(t) &= A_{mi}(t_i)x_{mi}(t_i) + B_m(t_i)z(t_i) \\ p_{mi}(t) &= C_{mi}(t_i)x_{mi}(t_i) \end{aligned} \tag{7}$$

Step-3: Design a Luenberger observer for the delay-free model

$$\dot{\hat{x}}(t) = A_{mi}(t_i)\hat{x}(t_i) + B_{mi}(t_i)z(t_i) + L_e(t_i)\left(p(t) - C_{mi}(t_i)\hat{x}(t_i)\right) \tag{8}$$

Where, $L_e \in \Re^{n \times p}$ is chosen so that $A_{mi} - L_e C_{mi}$ is stable (i.e., observer poles faster than controller poles).

Step-4: *Smith predictor / state prediction for input delay*

To compensate input delay τ_i , predict the plant state forward by τ_i using the delay-free model. With the estimated current state $\hat{x}(t_i)$ the τ_i –ahead predicted state (assuming $z(t)$ is held constant during $[t_i, t_i + \tau_i)$ or using known future command history) is:

$$\hat{x}(t_i + \tau_i) = \varphi(\tau_i)\hat{x}t_i + \Gamma(t_i)z(t_i) \tag{9}$$

where (for time-invariant A_{mi} or locally frozen $A_{mi}(t_i)$ over $[t_i, t_i + \tau_i)$.

$$\varphi(\tau_i) = e^{A_{mi}\tau_i}, \Gamma(\tau_i) = \int_0^{\tau} e^{A_{mi}s} B_{mi} ds \tag{10}$$

The predicted state constitutes the core of the Smith predictor, as it provides an estimate of the system state at the moment when the control action is implemented.

Step-5: *Predictive State-space Proportional Control Law:*

Use a proportional gain on the predicted state:

$$z(t_i) = -\mathrm{K}(t_i)\hat{x}(t_i + \tau_i) \tag{11}$$

So, the control acts on the predicted (delay-compensated) state. Substituting the prediction:

$$z(\mathrm{t}_i) = -\mathrm{K}(t_i)\left(\varphi(\tau_i)\hat{x}(\tau_i) + \Gamma(\tau_i)z(t_i)\right) \tag{12}$$

Solving for $z(\mathrm{t}_i)$ if $\Gamma(\tau_i)$ appears on both sides. For single-step hold (assume $\Gamma(\tau_i)$ $z(\mathrm{t}_i)$ small or invertible term:

Rearrange the equation:

$$z(t_i) = -\left(\mathrm{I} + \mathrm{K}(t_i)\Gamma(\tau_i)\right)^{-1} \mathrm{K}(t_i)\varphi(\tau_i)\hat{x}(t_i) \tag{13}$$

thus, if invertible:

$$\left(\mathrm{I} + \mathrm{K}(t_i)\Gamma(\tau_i)\right)z(t_i) = -\mathrm{K}(t_i)\varphi(\tau_i)\hat{x} \tag{14}$$

Where Eq. 14 is called Predictive State-space Proportional Control Law.

Where, $\hat{x}(t_i)$: estimated state (from Luenberger observer)

τ_i : is the network delay

$\varphi(\tau_i) = e^{A_{mi}\tau_i}$, state transition matrix over delay period,

$\Gamma(\tau_i) = \int_0^{\tau} e^{A_{mi}s} B_{mi} ds$, input contribution over delay

$\mathrm{K}(t_i)$: adaptive feedback gain matrix (updated using pole placement)

This algebraic correction accounts for the fact the predicted state included the present $z(\mathrm{t}_i)$.

Step-6: *Adaptive Pole-placement for* $\mathrm{K}(t_i)$ *(Estimate than place)*

i. Estimate $A(t_i)$, $\mathrm{B}(t_i)$ online using RLS

- Stack parameters $\vartheta(t_i) - vec\left(\left[A(t_i), \mathrm{B}(t_i)\right]\right)$
- Use standard discrete RLS to update $\vartheta(t_i)$ from a discretized counterpart

ii. Compute the nominal discrete-time pair $\left(\hat{A}, \hat{B}\right)$ and desired pole set $P_{des}(t_i)$ (reflecting required delay performance).

iii. Solve for $\mathrm{K}(t_i)$ by Pole placement, $\mathrm{K}(t_i) = place\left(\hat{A}(t_i), \hat{\mathrm{B}}(t_i), P_{des}(t_i)\right)$

where $place(\cdot)$ is any pole-placement routine that returns a state-feedback gain.

This approach recomputes K periodically (or whenever model change exceeds threshold).

Step-7: *Discrete Time Sampled Implementation for Edge*

Let sample period is T_{si} . Discrete model with input delay of d samples:

$$\begin{aligned} x_{ki+1} &= A_d x_{ki} + B_d z_{ki-d} + \varpi_{ki} \\ p_{ki} &= C_d x_{ki} + \upsilon_{ki} \end{aligned} \tag{15}$$

Prediction by d steps:

$$\hat{x}_{ki+d} = \varphi_d^{(d)} \hat{x}_{ki} + \sum_{i=0}^{d-1} \varphi_d^{(i)} B_d z_{ki+i-1} \tag{16}$$

Where, $\varphi_d = A_d$ and $\varphi_d^{(i)} = A_d^i$, then discrete controller: $z_{ki} = \mathrm{K}_{ki} \hat{x}_{ki+d}$

3.2. Adaptive Proportional Integral State-space Controller Design for Improvement of Network Delay Performance

The proposed controller utilizes an Adaptive Proportional–Integral State-Space (APIS-S) framework to address network-induced delay and packet loss in cloud-based control environments. The framework augments the plant model with an integral state to eliminate steady-state error and applies adaptive state-feedback based on real-time system identification and prediction. A dual-estimation strategy is implemented, where a Recursive Least Squares (RLS) algorithm rapidly updates linear model parameters for pole placement, and an Unscented Kalman Filter (UKF) estimates nonlinear state trajectories and slowly varying parameters. To manage time-varying communication delays, a Long Short-Term Memory (LSTM) network predicts future delay patterns, which are incorporated into a Smith predictor structure for delay compensation. Concurrently, a Soft Actor–Critic (SAC) reinforcement learning agent adaptively adjusts target pole locations and feedback gains to balance performance and robustness under uncertain network conditions. This integrated adaptive pole-placement mechanism generates proportional–integral feedback gains that evolve online in response to both plant dynamics and network status. Collectively, these components enable delay-aware prediction, state estimation, adaptive stabilization, and resilience to packet loss, thereby improving transient response, reducing steady-state error, and enhancing overall stability in cloud-based edge–IoT control systems.

Below, this article outlines the mathematical formulation that strengthens this revolutionary design.

Step-1: Discrete augmented plant (sample time T_{si})

Augment state with integral of output error:

$$\nu_{ki} = \nu_{ki-1} + \mathrm{T}_{si}\left(p_{ki-1} - p_{ref,ki-1}\right),$$
$$\aleph_k = \begin{bmatrix} x_{ki} \\ \nu_{ki} \end{bmatrix} \tag{17}$$

Linearized, delayed discrete model (delay d_{ki} samples):

$$\aleph_{ki+1} = A_{a,ki}\aleph_{ki} + B_{a,ki} z_{ki-d_{ki}} + \varpi_{ki},$$
$$p_{ki} = C_{a,ki}\aleph_{ki} + \upsilon_{ki} \tag{18}$$

Where, $A_{a,ki} = \begin{bmatrix} A_{ki} & 0 \\ C_{ki} & \mathrm{I} \end{bmatrix}, B_{a,ki} = \begin{bmatrix} B_{ki} \\ 0 \end{bmatrix}, C_{a,ki} = \begin{bmatrix} C_{ki} & 0 \end{bmatrix}$

Step-2: *Using Estimators and Predictors*
i. UKF(state +slow parameters)

Augmented UKF state $\chi_{ki} = \begin{bmatrix} \aleph_{ki}^T & \vartheta_{ki}^T \end{bmatrix}^T$

where, ϑ_{ki}^T slow plant parameters.

UKF predict/update (symbolically):

$$\hat{\chi}_{ki|ki-1} = UKF\_\mathrm{pr}edict\left(\hat{\chi}_{ki-1|ki-1}, z_{ki-d_{ki-1}}\right),$$
$$\hat{\chi}_{ki|ki} = UKF\_update\left(\hat{\chi}_{ki|ki-1}, p_{ki}\right) \tag{19}$$

Extract estimates, $\hat{\aleph}_{ki}, \hat{\vartheta}_{ki} \leftarrow \hat{\chi}_{ki|ki}$

ii. RLS (for fast linear identification)

Form regression $\varphi_{ki} = \chi_{ki} = \begin{bmatrix} \aleph_{ki}^T & z_{ki-d_{ki}}^T \end{bmatrix}^T$. For each row of X_{ki+1}:

$$\hat{\Theta}_{ki} = \hat{\Theta}_{ki-1} + L_{ki}\left(\mathrm{X}_{ki+1} - \hat{\Theta}_{ki-1}^T \varphi_{ki}\right),$$
$$L_{ki} = \mathrm{P}_{ki-1}\varphi_{ki}\left(\lambda + \varphi_{ki}^T \mathrm{P}_{ki-1}\varphi_{ki}\right)^{-1}, \mathrm{P}_{ki} = \lambda^{-1}\left(\mathrm{I} - L_{ki}\varphi_{ki}^T\right)\mathrm{P}_{ki-1} \tag{20}$$

Recover linearized matrices $A_{a,ki}, B_{a,ki}$ from $\hat{\Theta}_{ki}$.

iii. LSTM delay predictor

From recent network features H_{ki}:

$$\hat{\tau}_{ki+1} = LSTM\left(H_{ki}; \varpi\right)$$
$$d_{ki+1} = round\left(\hat{\tau}_{ki+1} \Big/ \mathrm{T}_{si}\right) \tag{21}$$

Step-3: *Adaptive pole selection (SAC) & gain computation*
i. SAC produces high-level action (pole adjustments or gain delta)

$$\text{SAC state: } s_{ki} = \left[\hat{\mathrm{X}}_{ki}, \hat{\vartheta}_{ki}, \hat{\tau}_{ki}, QoS_{ki}\right] \tag{22}$$

For example: $a_{ki} = \left[\Delta_{p_k}, \Delta_a^{SAC}\right]$ where, Δ_{p_k} are pole shifts.

Policy sampling: $a_{ki} \sim \Pi_{\varphi}\left(\cdot \middle| s_{ki}\right)$

ii. Desired adaptive poles

Base desired discrete poles: P^{base}.*SAC* modifies,

$$P_{ki} = P^{base} + \Delta_{p_k} \tag{23}$$

Enforce spectral constraints (projection) so, $|\lambda| \prec \partial_{\max} \prec 1$

iii. Adaptive pole-placement gain

Compute state-feedback PI gain by pole placement on the RLS/UKF linear model:

$$\mathrm{K}_{a,ki}^{place} = place\left(\hat{A}_{a,ki}, \hat{B}_{a,ki}, P_{ki}\right) \tag{24}$$

Combine with SAC gain increment:

$$\mathrm{K}_{a,ki} = proj_k\left(\mathrm{K}_{a,ki}^{place} + \Delta\mathrm{K}_{a}^{SAC}\right) \tag{25}$$

where $proj_k(\cdot)$ bounds gain to a safe convex set k.

Step-4: *Smith-style delay compensation (prediction to application time)*

Smith-style delay compensation, a model-based predictor to "look ahead" by the network delay and compute the control action *as if the delay were not there*.

Predict augmented state d steps ahead with identified model $\hat{A}_{a,ki}, \hat{B}_{a,ki}$:

$$\hat{\mathrm{X}}_{ki+d} = \hat{A}_{a,ki}^{d}\hat{\mathrm{X}}_{ki} + \sum_{i=0}^{d-1}\hat{A}_{a,ki}^{i}\hat{B}_{a,ki}z_{ki-d+i} \tag{26}$$

Define $\varphi_{d,ki} = \hat{A}_{a,ki}^{d}$ and $\Gamma_{d,ki} = \sum_{i=0}^{d-1}\hat{A}_{a,ki}^{i}\hat{B}_{a,ki}$

Step-5: *Final Control Law ((delay-compensated PI state-feedback)*

To ensure robustness during online adaptation, the proposed APIS-S controller incorporates both projection and update rules. The gain matrix is projected onto a bounded convex set to prevent excessively large values. The invertibility of the matrix $(\mathrm{I}+\mathrm{K}\Gamma)$ is maintained by introducing a regularization term. Recursive least squares (RLS) updates are stabilized using a forgetting factor λ in the range $\lambda \in (0.98,1)$. The unscented Kalman filter (UKF) is controlled by covariance thresholds to prevent divergence in the presence of packet loss. The soft actor-critic (SAC) agent's actions are constrained by a projection operator, ensuring that the adapted poles remain within the unit circle. Together, these mechanisms ensure that parameter updates remain bounded and prevent instability during abrupt fluctuations in delay.

Now, the exact corrected (accounts for Γ dependence if control appears in prediction) can be written as:

$$\begin{aligned}&\left(\mathrm{I}+\mathrm{K}_{a,ki}\Gamma_{d,ki}\right)z_{ki} = -\mathrm{K}_{a,ki}\varphi_{d,ki}\hat{\mathrm{X}}_{ki}\\ &\Rightarrow z_{ki} = -\left(\mathrm{I}+\mathrm{K}_{a,ki}\Gamma_{d,ki}\right)^{-1}\mathrm{K}_{a,ki}\varphi_{d,ki}\hat{\mathrm{X}}_{ki}\end{aligned} \tag{27}$$

3.3. Time Delay Compensator

Networked and cloud-based control systems inherently experience time delays resulting from communication, computation, and scheduling constraints, which can degrade system stability and performance. Time delay compensators, including the Smith Predictor and its contemporary variants, are implemented to predict and counteract the adverse effects of these delays on system dynamics.

Smith predictors have been extended for wireless networked control systems through the integration of delay estimators into the predictor structure [34]. These methods enhance robustness to random wireless delays relative to classical Smith predictors. However, they rely exclusively on estimation-based

compensation, which does not adequately address rapid delay variations or the need for system adaptation because adaptive pole placement and learning-based predictors are not incorporated. Classical Smith Predictors operate under the assumption of constant delays, which limits their effectiveness in networked environments characterized by packetization and jitter. To overcome this challenge, [35] introduces a conceptual modification of Smith Predictors for networked control systems that integrates estimators and compensators to manage variable delays and packet loss. However, classical Smith Predictors are applicable only to stable plant scenarios. A recent modification [36] extends this approach to unstable linear systems with input delay and offers exponential stability guarantees. However, this method does not address time-varying or stochastic network delays and lacks adaptive estimation or learning mechanisms. In contrast, the proposed controller combines Smith-style delay compensation with state-space adaptation using recursive least squares (RLS) and unscented Kalman filter (UKF) algorithms, as well as reinforcement learning techniques such as soft actor-critic (SAC) and long short-term memory (LSTM) networks, to achieve robust performance in cloud-edge environments.

If $p(t)$ is the actual output, and $\hat{p}(t)$ is the model based prediction,

$$\hat{p}(t) = G(s)z(t) \tag{28}$$

$$p(t) = G(s)z(t - \mathrm{T}_{di}) \tag{29}$$

The Smith Predictor reconstructs a pseudo-delay-free signal**:**

$$p_{sp}(t) = \hat{p}(t) + \left(p(t) - \hat{p}(t - \mathrm{T}_{di}) \right) \tag{30}$$

Here,

$\hat{p}(t)$: Predicted output without delay

$p_{sp}(t)$: Correction term accounting for modeling error.

$p(t) - \hat{p}(t - \mathrm{T}_{di})$: Correction term accounting for modeling error.

The Smith Predictor doesn't remove the delay physically instead, it recreates the system's output as if the delay weren't there**,** allowing the controller act in real time.

3.4. Packet loss Compensation in Controller design

With the aim to guarantee the boundedness of system states over a limited interval, Lyapunov–Krasovskii functionals and LMI-based conditions were created in [37], which examined finite-time stability analysis for networked control with packet dropout and delay. Although these approaches work well for theoretical assurances, they don't directly address dynamic adaptability in cloud/edge systems.

Analysing packet loss in Wireless Network Control Systems is essential because such losses significantly degrade network performance. Wireless networks are particularly vulnerable to packet loss caused by interference, fading, and signal attenuation. This paper proposes a compensation model for packet loss utilizing the Kalman filter. The packet loss process is modelled using the Gilbert-Elliot framework and compared with a Proportional-Integral-Derivative (PID) controller. Experimental results demonstrate that the application of Kalman filters enhances process performance during data transmission losses. The multidisciplinary nature of wireless networked control systems (WNCS) necessitates the use of simulations to support both system development and the evaluation of controller performance. Control and management within WNCS facilitate the creation of diverse control architectures and improve the estimation of system parameters. The adoption of advanced methodologies in network control systems has contributed to the rapid advancement of WNCS. In particular, the implementation of filtering techniques has attracted significant attention for enhancing system performance [38]. Markov jump linear models are frequently used in classical optimal control strategies for networked systems with packet loss and delay. For instance, using Lyapunov-based LMIs to ensure mean-square stability, [39] constructed an ideal LQR-type controller and expressed delay and dropout as Markov processes. These methods, however, are not flexible enough to handle stochastic or time-varying cloud settings and rely on known transition probabilities. On the other hand, our approach addresses

packet loss and delay in edge-cloud systems by combining Smith-style prediction, adaptive estimating (RLS, UKF), and learning-based tuning (SAC, LSTM).

If a measurement p_{ki} is lost at time ki, the estimator uses prediction;

- UKF/Luenberger: skip update and propagate $\mathrm{X}_{ki+1|ki} = \hat{A}_{a,ki}\hat{\mathrm{X}}_{ki} + \hat{B}_{a,ki}z_{ki-d}$, covariance P is inflated to reflect uncertainty.
- RLS: skip parameter update (or use last update).
- SAC/LSTM: LSTM indicates higher loss probability → SAC can choose conservative pole shifts (slower poles) via p_{ki} to improve robustness.

Methodically, one models packet loss by a Bernoulli process $\gamma_{ki} \in \{0,1\}$ where $\gamma_{ki} = 1$ when measurement arrive. Then observer update Eq (8) becomes conditional:

$$\dot{\hat{x}}(t) = A_{mi}(t_i)\hat{x}(t_i) + B_{mi}(t_i)z(t_i) + \gamma_{ki}L_e(t_i)\left(p(t) - C_{mi}(t_i)\hat{x}(t_i)\right) \tag{31}$$

Let p denotes the packet loss probability, and $\gamma_{ki} \in \{0,1\}$ be the packet reception indicator at time ki (where, $\gamma_{ki} = 1$), The expected control input of the two controllers under packet loss is given by:

for Predictor SSP:

$$\Xi\left[z_{ki}^{P}\right] = (1-p)z_{ki}^{nom} + p\hat{z}_{ki}, \tag{32}$$

where $\hat{z}_{ki}$ is the predictor output generated using UKF, LSTM, SAC, or Leubenger methods.

For Adaptive SSPI:

$$\Xi\left[z_{ki}^{A}\right] = (1-p)z_{ki}^{nom} + pz_{ki-1}^{A}, \tag{33}$$

with Recursive Least Squares (RLS) providing online parameter adaptation.

4. Results Analysis

Delay and packet loss are two of the most important variables that affect control performance and system stability in Edge–Cloud integrated IoT systems. While packet loss compromises data integrity and can lead to an accumulation of errors in feedback loops, delay adds latency to communication between sensors and controllers and between controllers and actuators, influencing overshoot and settling time. Therefore, this paper rigorously evaluates the suggested controllers' performance under various delay and packet loss levels in order to verify their resilience. The following sections present these results and compare the behavior of Adaptive SSPI and Predictive SSP in these conditions in MATLAB/Simulink environment.

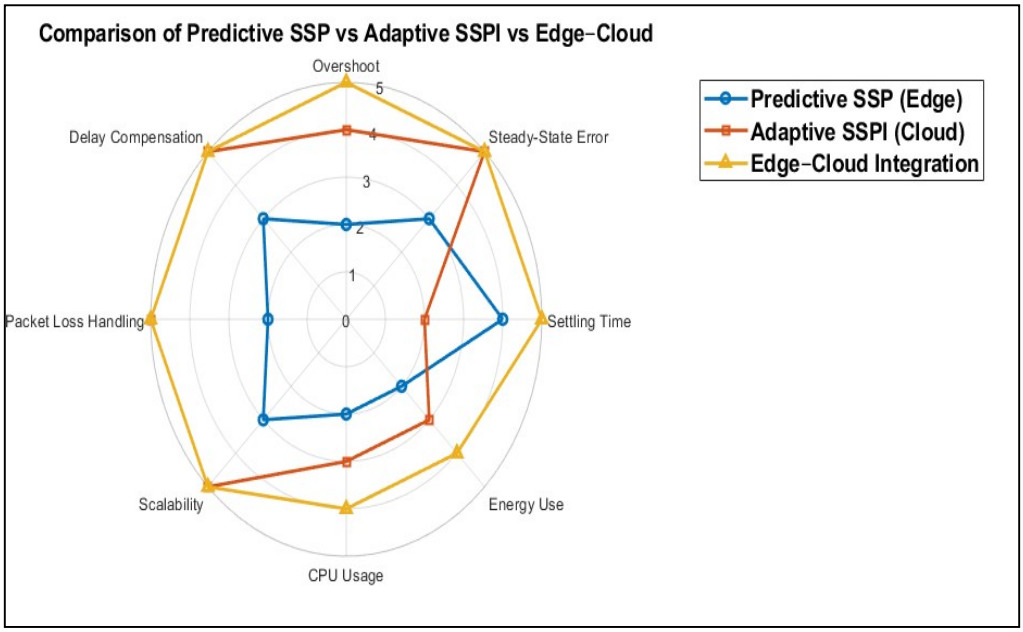

Figure 3. Comparative Performance of Edge–Cloud Integrated vs. Edge-Only and Cloud-Only Approaches under Network Constraints of Adaptive SSPI Control and Predictive SSP control

The radar chart compares the performance of three control strategies across multiple criteria, as shown in Figure 3. These criteria include overshoot, steady-state error, settling time, energy use, CPU usage, scalability, packet loss handling, and delay compensation. The outer polygon demonstrates that the edge–cloud integrated adaptive controller with reinforcement learning and estimation techniques achieves superior performance in almost all metrics. It excels particularly in handling delay, packet loss, and scalability. The intermediate polygon represents the edge-only adaptive approach. This approach performs moderately well, especially in scalability and steady-state accuracy, but lags in energy efficiency and settling time. The innermost polygon corresponds to the cloud-only or conventional controller, which shows limited capability across all dimensions. Overall, the results highlight that combining predictive and adaptive techniques within an edge–cloud framework yields the most robust and efficient control performance for IoT-based NCS.

Table 1. illustrates the *Cloud-only controller* struggles with delay, overshoot, and scalability, but has low edge CPU demand. The *Edge-only controller* improves responsiveness and accuracy but is limited in scalability and optimization. The *Edge–Cloud Integrated Adaptive controller* achieves the *greatest steadiness*, low overshoot, minimal steady-state error, short settling time, optimized CPU and energy usage, and excellent scalability with strong packet loss and delay compensation.

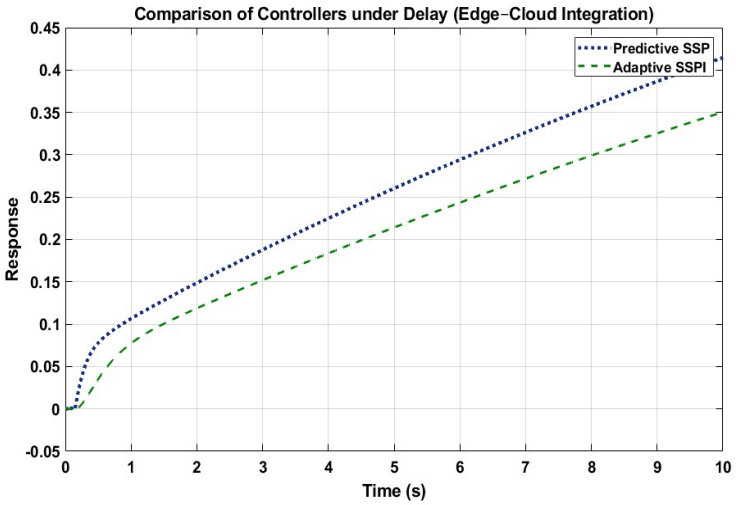

Figure 4. Comparative Transient Response of S-C and C-A Delay between Predictive SSP and Adaptive SSPI

Figure 4 shows how the transient response of predictive SSP and adaptive SSPI controllers is affected by sensor-to-controller and controller-to-actuator delays. Because of its prediction method, the Predictive SSP shows a quicker initial climb, which helps to offset network-induced delays. Its performance, however, is heavily reliant on prediction accuracy and could grow unstable as delays rise. The Adaptive SSPI, on the other hand, reacts more slowly but keeps smoother, more stable dynamics, demonstrating its resilience to changes in delay. These findings point to a crucial trade-off: While adaptive SSPI emphasizes robustness at the expense of slower response, predictive SSP compensates for delays but is computationally intensive.

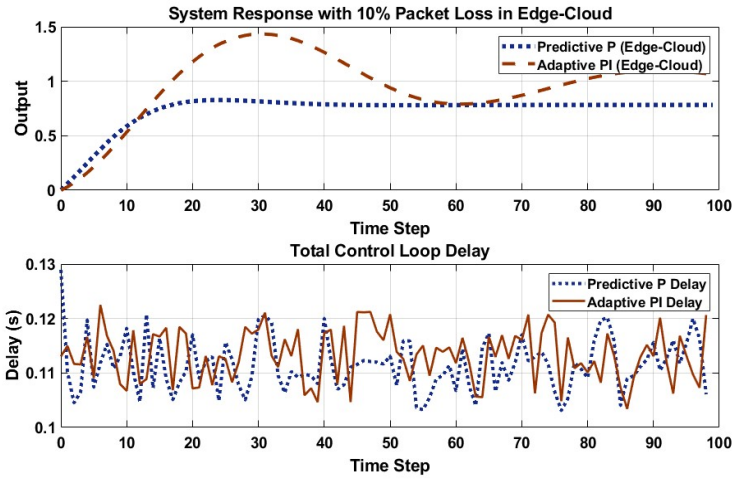

Figure 5. System Response with 10% Packet Loss in Edge-Cloud" and "Total Control Loop Delay

Figure 5 compares two control systems under a 10% packet loss rate: Predictive P (Edge-Cloud) and Adaptive PI (Edge-Cloud). Both experience similar delays due to packet loss. However, the Predictive P controller remains more robust and stable, maintaining consistent performance. In contrast, the Adaptive PI controller shows reduced stability and is more prone to instability. These results highlight the benefits of predictive control strategies in networks with significant packet loss.

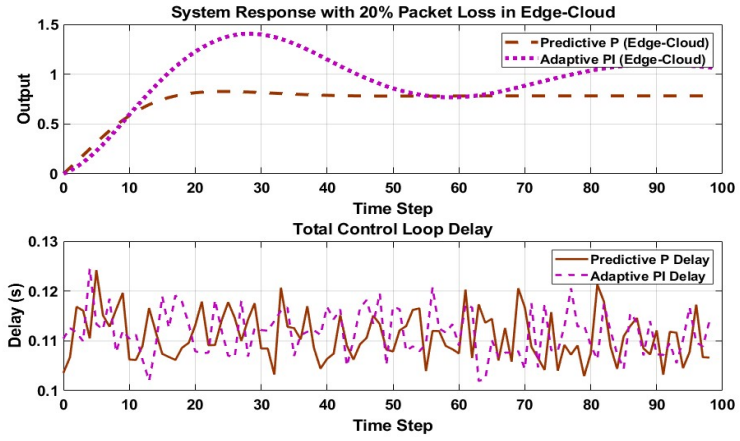

Figure 6. System Response with 20% Packet Loss in Edge-Cloud" and "Total Control Loop Delay

Figure 6 strongly establishes the trade-offs between different control strategies in unreliable network conditions under 20% packet loss rate. The **Predictive P** controller excels in high packet loss situations. It can predict future states and retain working even if packets drop. This makes it robust for edge-cloud applications with unreliable connectivity. In contrast, the **Adaptive PI** controller, although designed to manage some network variations, performs poorly under high packet loss. Its dependence on a continuous, reliable data stream for the integral component makes it unsuitable for this environment. This analysis indicates that for cyber-physical systems operating over lossy networks, a predictive control approach is more effective than traditional feedback-based methods.

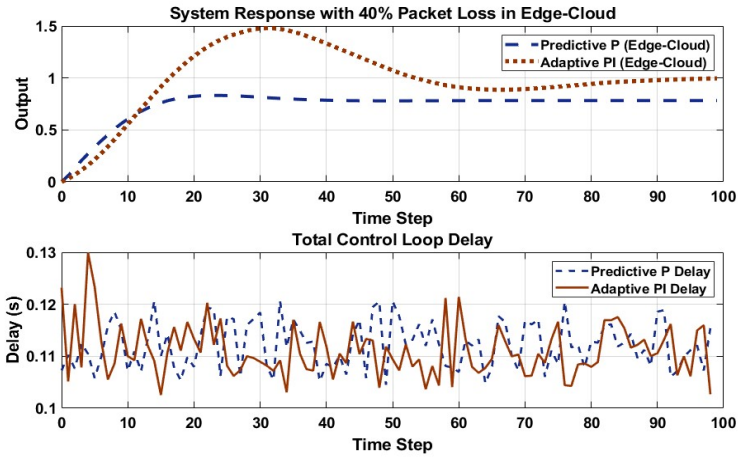

Figure 7. System Response with 40% Packet Loss in Edge-Cloud" and "Total Control Loop Delay.

Even with a 40% packet loss rate, the Predictive P controller output shows consistent performance, settling at about 0.8. On the other hand, the latency is extremely erratic and varies greatly. The Adaptive PI controller, on the other hand, performs erratically in the same circumstances. The output becomes unpredictable, the system loses control, and it reacts unpredictably to significant data loss. Like other high-loss situations, the delay is inconsistent and exhibits notable spikes as depicted in Figure 7.

Table 2, enlightens that Predictive SSP succeeds under high packet loss (40%) when predictors are accurate, yet demands significant computation and strong edge–cloud connectivity. In sharp contrast, Adaptive SSPI is lighter and more robust in low-to-moderate losses with reduced communication overhead, but falters under high losses as errors accumulate.

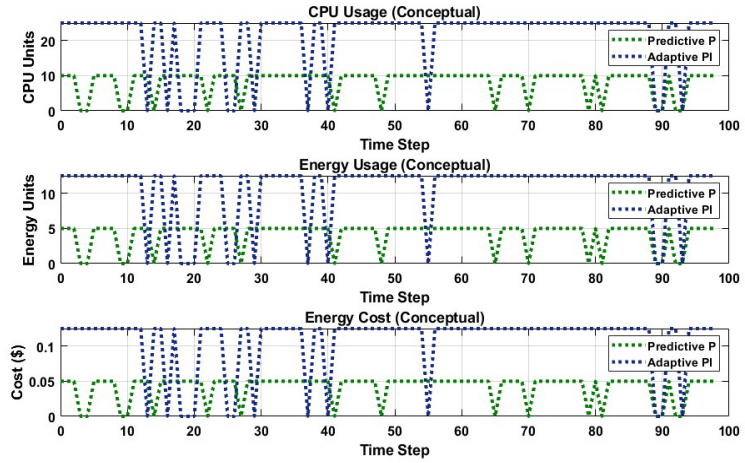

Figure 8. Performance Comparison of Predictive SSP and adaptive SSPI controller using only Cloud using CPU Usage, Energy Cost and Energy Efficiency

The Figure 8 which consists of three conceptual time-series plots, presents the performance, energy usage, and cost associated with two CPU provisioning strategies in a discrete-time environment. The **Predictive P** strategy demonstrates consistently *low and stable* CPU usage, maintaining approximately *10 CPU Units* with occasional reductions to 0. This pattern indicates a conservative or baseline allocation, likely intended for steady, low-to-medium workloads. In contrast, the Adaptive PI strategy exhibits highly variable and aggressive behaviour, frequently alternating between **20 CPU Units** (the maximum capacity depicted) and **0 CPU Units**. This behaviour reflects an on-demand system that rapidly scales to maximum capacity when workload is detected and reduces to zero when idle. *Predictive P also maintains low and stable energy usage (approximately 5* Energy Units) and cost **(***approximately $0.05), prioritizing* stability and a lower overall baseline cost, though it may risk under-provisioning during periods of high demand. Conversely, Adaptive PI results in high and **volatile energy usage** (often peaking at 12 Energy Units) and cost (often peaking at $0.12), prioritizing immediate and maximum resource allocation to address peak demands, but leading to *significantly higher instantaneous energy consumption and cost.*

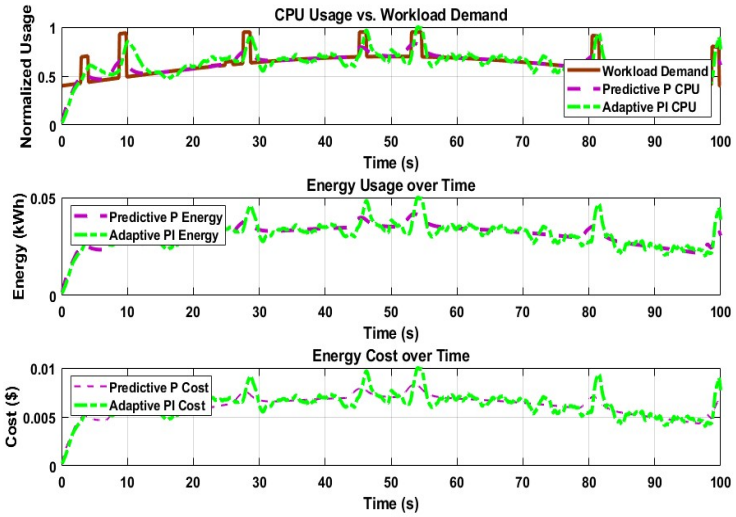

Figure 9. Performance Comparison of Predictive SSP and adaptive SSPI controller using Edge-Cloud integrated for CPU Usage, Energy Cost and Energy Efficiency

The Figure 9 illustrates the performance, energy consumption, and cost associated with two *CPU control strategies* in response to varying workload d**emand**. Both the *Predictive P CPU and Adaptive PI CPU* strategies closely track workload demand over a 100-second period, with each exhibiting peak that correspond to demand spikes. This indicates that both strategies are generally effective at resource provisioning. The Adaptive PI CPU line is smoother and demonstrates more consistent tracking of workload demand compared to the **Predictive P CPU** line, which exhibits higher peaks and more aggressive responses in certain intervals, such as at t=10s and t=30s. The *Adaptive PI strategy results in lower energy consumption (kWh) and cost ($) than the Predictive P strategy, as evidenced by the Adaptive PI line remaining below* the **Predictive P** line in both the energy and cost plots. These results indicate that the *Adaptive PI controller is more energy-efficient and cost-effective*, achieving comparable CPU usage tracking with reduced energy consumption and lower cost.

As shown in Table 3, Predictive proportional (P) control consumes greater CPU resources, energy, and incurs higher costs as a result of instability and significant fluctuations. In contrast, adaptive proportional-integral (PI) control offers improved workload tracking, reduced CPU usage, lower energy consumption, and decreased operational costs. These characteristics enhance its scalability and efficiency in Edge–Cloud environments.

While both Predictive SSP and Adaptive SSPI offer distinct advantages in Edge–Cloud integration, each also has important limitations. Predictive SSP relies on computationally intensive predictors such as UKF, LSTM, or SAC. These may exceed the processing capacity of lightweight edge devices. Furthermore, its performance is highly sensitive to model accuracy and synchronization between the edge and cloud. Any mismatch in predictor training or extra network latency can degrade control quality. The frequent communication required for prediction also imposes considerable bandwidth overhead. This constrains scalability in large-scale IoT deployments.

In contrast, Adaptive SSPI is computationally lightweight and less dependent on cloud resources, but its performance deteriorates significantly under high packet loss conditions (>30–40%), where integral action may accumulate errors and threaten stability. Additionally, its transient response is slower than predictive methods, limiting its effectiveness in highly dynamic environments. Owing to its reliance on replaying past values and adjusting gains via RLS, it also lacks the forward-looking predictive capability necessary to compensate for long communication gaps. Thus, although Adaptive SSPI provides a more resource-efficient and robust option under moderate network stress, its scalability under extreme loss and jitter remains restricted.

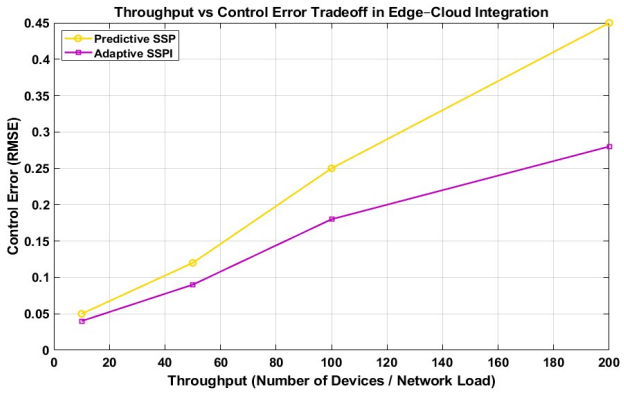

Figure 10. Tradeoff between throughput (network load) and control error (RMSE) for Predictive SSP and Adaptive SSPI in an Edge–Cloud integrated system

This Figure 10 illustrates the relationship between throughput, measured as network load, and control error, quantified by root mean square error (RMSE), for two strategies within an Edge–Cloud integrated system. The X-axis indicates the number of devices or the traffic load in the Edge–Cloud system, with 'Edge' denoting local devices and 'Cloud' denoting centralized computing resources. The Y-axis represents control accuracy, quantified by error. The Figure illustrates that increasing the number of devices results in network congestion and higher control errors. Predictive SSP results in higher error rates, whereas Adaptive SSPI maintains more consistent control performance and achieves lower root mean square error (RMSE), which quantifies the difference between predicted and actual values. Consequently, Adaptive SSPI is better suited for large-scale Edge–Cloud integration.

As depicted in Table 4, Adaptive SSPI demonstrates greater scalability compared to Predictive SSP, as it sustains lower control error with increasing network load, measured by throughput or number of devices.

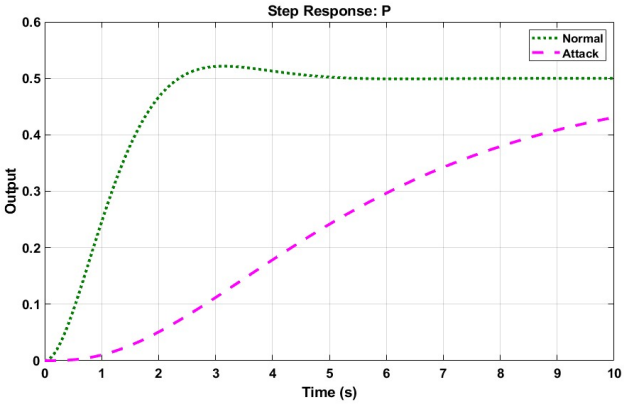

Figure 11. Step Response Performance of State-space Proportional Controller under Normal and Attack Conditions

The step reaction graph as depicted in Figure 11 makes it evident how drastically the attack affected the system's functionality. Under typical circumstances, the system responds in a desirable, rapid manner, rising to its steady-state value of roughly 0.5 in a matter of seconds with little overshoot. The Attack scenario, on the other hand, drastically deteriorates the dynamics, resulting in a noticeably slower rise in output. The system is still far from settling at t=10 seconds, indicating a significant increase in both the rising and settling times. This slow behaviour indicates that the attack successfully increased the system's time constant or created a significant delay, rendering the control loop extremely unresponsive and falling short of crucial performance standards.

The simulation graph as revealed in Figure 12 shows that the performance of the PI control system is significantly deteriorated by the attack. In contrast to the Normal response, which is quick and has no steady-state error, the Attack response slows down and rises uncontrollably, deviating significantly from the intended setpoint of 1.0 during the observed period. This indicates that the attack has changed the system dynamics in a substantial way, possibly by sending false signals into the actuator or sensor channels, which resulted in a loss of effective control.

When the system is operating in "Attack" mode as opposed to "Normal" mode, both controllers show a notable decline in control system performance. The consequent poor performance is very consistent with the impacts of packet dropout or delay in a Networked Control System (NCS), even though the simulation figures themselves do not specifically identify packet dropout as the cause.

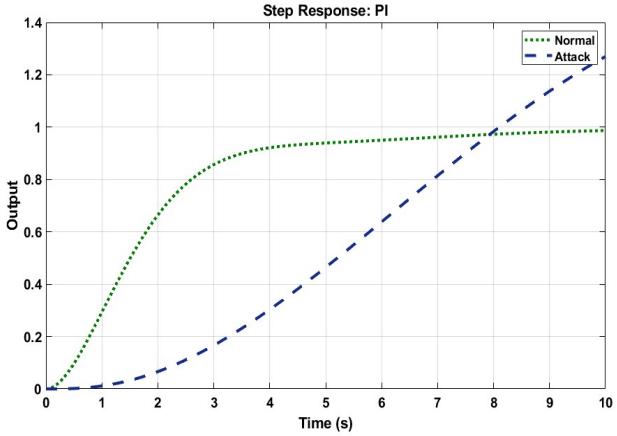

Figure 12. Step Response Performance of State-space Proportional Integral Controller under Normal and Attack Conditions

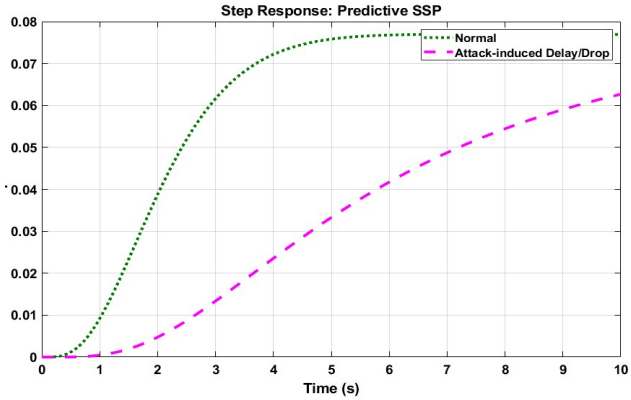

Figure 13. Step Response Performance of Predictive State-space Proportional Controller under Normal and Attack Conditions

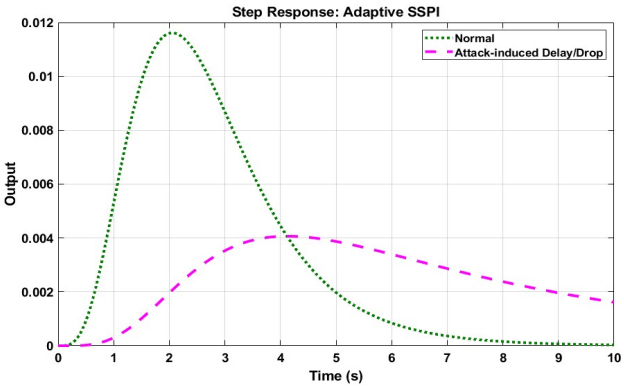

Figure 14. Step Response Performance of Adaptive State-space Proportional Integral Controller under Normal and Attack Conditions

The Figure 13. presents the step response of a Predictive State-Space Proportional (SSP) controller under both normal and attack-induced delay or drop conditions. The curve, representing normal operation, displays a rapid rise and smooth convergence to a steady state, which indicates stable performance. In contrast, the curve, corresponding to attack-induced conditions, shows a slower rise and

a lower final output, reflecting performance degradation caused by communication delays or packet drops. This comparison illustrates that attacks increase rise time, reduce steady-state accuracy, and impair overall controller effectiveness.

The Figure 14 presents the step response of an Adaptive State-Space Proportional-Integral (SSPI) controller under both normal and attack-induced delay or packet drop conditions. During normal operation, the controller exhibits a rapid response, achieving a pronounced peak at approximately 2 seconds, followed by a smooth decay. This behaviour demonstrates the controller's adaptive and responsive characteristics. In contrast, under attack-induced conditions, the response is considerably slower, with a diminished peak and extended settling time, indicating reduced performance and compromised stability. This comparison demonstrates that although the Adaptive SSPI controller performs effectively under normal circumstances, attack-induced delays or packet drops substantially impair its adaptive capabilities, resulting in a slower rise time, decreased output, and prolonged decay.

However, packet loss has a detrimental effect on both controllers in the Edge-Cloud integrated example, but because the Adaptive SSPI depends on prompt and accurate feedback, its performance deteriorates more than the other controller. The Predictive SSP, on the other hand, maintains some predictive power in spite of packet loss.

The comparison highlights trade-offs among SSP, SSPI, Predictive SSP, and Adaptive SSPI under normal and attack conditions in Table 5. In normal operation, SSP offers the fastest rise time but suffers from steady-state error, while SSPI removes error at the cost of slower cloud-based response. Predictive SSP improves SSP by compensating delay but cannot fully eliminate errors, whereas Adaptive SSPI balances speed and accuracy with near-zero error through adaptive tuning.

Under attack, SSP becomes unstable with high error, and SSPI slows down and remains vulnerable. Predictive SSP provides partial robustness by mitigating packet drops but leaves residual error. Adaptive SSPI is the most resilient, dynamically handling delay, jitter, and packet loss while maintaining near-zero error.

Overall, cloud-based SSP/SSPI are fragile, while predictive and adaptive edge-cloud approaches are more robust, with Adaptive SSPI achieving the best stability, accuracy, and resilience.

Table 1. Performance comparison analysis of Edge–Cloud integrated Adaptive Control vs. Edge-Only and Cloud-Only approaches under network constraints

Performance Metric	Cloud-only Controller	Edge-only Controller	Edge-cloud Integrated Adaptive Controller
Overshoot	High	Medium	Low
Steady-state error	High	Low	Very low
Settling Time	Long	Medium	Short
Energy Use	High	Medium	Low
CPU Usage	Low	Medium	Optimized
Scalability	Poor	Good	Excellent
Packet loss Handling	Weak	Moderate	Strong
Delay Compensation	Poor	Moderate	Excellent

Table 2. Comparison of predictive SSP and adaptive SSPI controllers under Packet Loss in Edge–Cloud integration

Performance metric	**Predictive SSP**	**Adaptive SSPI**
Compensation strategy	Replace lost packet with predicted $\hat{z}_{ki}$ computed at edge/cloud	Hold/Replay last control z_{ki-1} and adapt controller gains via RLS
Mathematical action	$z_{ki} = (\gamma_{ki}) z_{ki}^{nom} + (1-\gamma_{ki}) \hat{z}_{ki}$	$z_{ki} = (\gamma_{ki}) z_{ki}^{nom} + (1-\gamma_{ki}) \hat{z}_{ki} - 1$, RLS updates paramters
Estimator(s) used	UKF / LSTM / SAC / Leubenger (prediction model resides on edge/cloud).	RLS for parameter estimation; optional lightweight predictor on edge
Performance @ 10% loss	Negligible degradation	Small degradation
Performance @ 20% loss	Reasonable degradation	Perceptible degradation
Performance @ 40% loss	Best among two, if predictors (LSTM/UKF/SAC/Leubenger) are well-trained and edge/cloud latency is small. Otherwise performance drops	*Severe degradation*; integral action may accumulate error; instability risk if packet loss correlates with delays.
Transient response	Better	Worse
Steady-state error	Low if predictor is accurate: $\hat{z}_{ki} \approx z_{ki}^{nom}$	Low due to integral action, but convergence slower under frequent loss.
Robustness to model mismatch	Sensitive. Robustness improves with adaptive predictors (SAC/LSTM)	More robust to model mismatch for small losses, but performance suffers as loss increases.
Computation & communication	Higher compute (predictors) and more edge–cloud interaction; needs timely synchronization.	Lower compute per step; RLS is lightweight; fewer cloud predictions required.
Network load (edge-cloud)	Higher	Lower

Table 3. Comparing predictive SSP and adaptive SSPI **i**n terms of CPU usage, energy consumption, and cost**.**

Performance metric	**Predictive SSP**	**Adaptive SSPI**	**Observation**
CPU Usage (Cloud-only)	High fluctuations, spikes up to 20+ units.	Lower and more stable, about 10 units	Adaptive PI uses fewer CPU resources, better stability.
CPU Usage vs Workload (Edge-cloud)	Tracks workload but overshoots in some intervals.	Closely tracks workload demand with less deviation	Adaptive PI matches workload more efficiently.
Energy Usage (Cloud-only)	Peaks often around 10 units, unstable.	Stable around 5 units, fewer peaks.	Adaptive PI reduces energy consumption.
Energy Usage over Time (Edge-cloud integration)	Slightly higher, with more spikes under workload changes.	Slightly lower, smoother under workload demand.	Adaptive PI is more energy-efficient.
Energy cost (Cloud-only)	Higher fluctuations in cost, unstable.	Lower and steadier cost	Adaptive PI provides cost savings
Energy cost over Time (Edge-cloud integration)	More variable, slightly higher at peaks	Lower and smoother, follows workload demand efficiently.	Adaptive PI has economic advantage.
Scalability & Robustness	Degrades under fluctuating workloads, unstable CPU and energy demand.	Stable even under high fluctuations, adaptive behavior ensures robustness	Adaptive PI is more scalable and sustainable.

Table 4. Comparison Analysis between Predictive SSP and Adaptive SSPI in terms of Scalability for Edge–Cloud integration

Features	Predictive SSP	Adaptive SSPI
Scalability with Number of Devices	Control error grows rapidly with increasing devices	Maintains lower control error even as devices scale up
Error Growth Trend	Steep increase in RMSE as throughput rises	Gradual increase in RMSE with throughput.
Performance at Low Load (10–50 devices)	Acceptable, comparable to Adaptive SSPI.	Similar to Predictive SSP, stable control
Performance at Medium Load (~100 devices)	Control error increases significantly, system stability starts degrading	Control error increases moderately, stability remains acceptable
Performance at High Load (~200 devices)	High error (≈0.45 RMSE), scalability bottleneck.	Lower error (≈0.28 RMSE), robust under high load.
Suitability for Large-Scale Edge–Cloud Systems	Less suitable	Highly suitable

Table 5. Comparison Analysis between SSP, SSPI, Predictive SSP, Predictive SSP and Adaptive SSPI in terms of Attack vs Normal Condition

Aspect	SSP	SSPI	Predictive SSP	Adaptive SSPI
Rise time on Normal response	Fast (but delayed by cloud)	Slower (cloud + integral)	Fast (prediction reduces delay impact)	Moderate (adaptive tuning slows initially)
Rise time on Under attack	Appears fast but unstable under attack	Slow + affected by cloud attacks	Faster, prediction mitigates some attack delay	Stable, adjusts dynamically, but slightly slower
Normal response on steady state	Non-zero	≈ 0	Non-zero (improved slightly by prediction)	≈ 0
Under attak steady state	Large, increases under attack	Lower than SSP but still grows due to cloud delay & packet loss	Lower than SSP, still > 0	≈ 0 even under attack (adaptive compensation)
Overall effect on normal response	Quick but inaccurate	Accurate but slower	Quicker, partially better accuracy	Balanced, stable, accurate
Overall effect under attack condition	Highly degraded, oscillatory	More robust, but cloud delay worsens under attack	Robust to mild attacks, reduced SSE, faster than cloud SSPI	Most resilient: tolerates delay, jitter, packet drops, keeps SSE ≈ 0

5. Conclusion

In an edge–cloud integrated IoT system, this study compares Adaptive SSPI and Predictive SSP controllers under various delay and packet loss conditions. The results indicate that Predictive SSP achieves superior short-term performance when predictive models are well trained; however, it incurs higher computational costs, is sensitive to model inaccuracies, and requires greater edge–cloud communication resources. In contrast, Adaptive SSPI offers a lightweight implementation, lower network load, and resilience under moderate packet loss; however, it exhibits slower transient response and performance degradation under high-loss scenarios. These findings suggest no universal controller solution—Predictive SSP is preferred in resource-rich, high-loss environments, while Adaptive SSPI suits resource-limited, moderate-loss settings. This analysis provides practical insights for selecting controllers in diverse IoT environments.

Our work in future expansions can be enhanced by edge AI-driven solutions that integrate predictive and adaptive control inside a unified framework. Federated learning and lightweight reinforcement learning agents could be employed at the edge to optimize controller parameters in dynamic network conditions and reduce reliance on cloud synchronization. Additionally, multi-agent collaboration and AI-driven resource allocation can further improve scalability, robustness, and fault tolerance in large-scale Edge–Cloud IoT systems. Such techniques will enable more intelligent, robust, and energy-efficient control architectures. Additionally, to further lower communication overhead and

enhance delay and packet loss compensation, we can incorporate event-triggered control into Predictive SSP and Adaptive SSPI controllers.

References

1. Chaves, L. S., Callegari, J. M. S., Araujo, L. S., & Brandao, D. I. (2025). Impact of Latency and Packet Error on Communication in Centralized Microgrid Control: Modeling and Guidelines. *IEEE Access*.
2. Lu, Z., & Guo, G. (2023). Control and communication scheduling co-design for networked control systems: A survey. *International Journal of Systems Science*, *54*(1), 189-203.
3. Tian, G. S., Xia, F., & Tian, Y. C. (2012). Predictive compensation for variable network delays and packet losses in networked control systems. *Computers & Chemical Engineering*, *39*, 152-162.
4. Tian, Z. D., Gao, X. W., Gong, B. L., & Shi, T. (2015). Time-delay compensation method for networked control system based on time-delay prediction and implicit PIGPC. *International Journal of Automation and Computing*, *12*(6), 648-656.
5. Stanojevic, K., Steinberger, M., & Horn, M. (2024, December). State Estimation in Networked Control Systems with Time-Varying Delays: A Simple yet Powerful Observer Framework. In *2024 IEEE 63rd Conference on Decision and Control (CDC)* (pp. 6916-6921). IEEE.
6. Mahmoud, M. S., Memon, A. M., & Shi, P. (2014). Observer-based fault-tolerant control for a class of nonlinear networked control systems. *International Journal of Control*, *87*(8), 1707-1715.
7. Halldorsson, R. M., Dushku, E., & Dragoni, N. (2021). ARCADIS: asynchronous remote control-flow attestation of distributed IoT services. *IEEE Access*, *9*, 144880-144894.
8. Kreković, D., Krivić, P., Žarko, I. P., Kušek, M., & Le-Phuoc, D. (2025). Reducing communication overhead in the IoT-edge-cloud continuum: A survey on protocols and data reduction strategies. *Internet of things*, 101553.
9. Han, M., Zhang, L., Wang, J., & Pan, W. (2020). Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*, *5*(4), 6217-6224.
10. Kovtun, V. (2025). Decentralized queue control with delay shifting in edge-IoT using reinforcement learning. *Scientific Reports*, *15*(1), 30845.
11. Jiang, Y., Wang, Z., & Jin, Z. (2023). IoT data processing and scheduling based on deep reinforcement learning. *International Journal of Computers Communications & Control*, *18*(6).
12. Xu, J., Wan, W., Pan, L., Sun, W., & Liu, Y. (2024, February). The fusion of deep reinforcement learning and edge computing for real-time monitoring and control optimization in iot environments. In *2024 3rd International Conference on Energy and Power Engineering, Control Engineering (EPECE)* (pp. 193-196). IEEE.
13. Zubair Islam, M., Shahzad, Ali, R., Haider, A., & Kim, H. S. (2022). Reinforcement learning-aided edge intelligence framework for delay-sensitive industrial applications. *Sensors*, *22*(20), 8001.
14. Nieto, G., De la Iglesia, I., Lopez-Novoa, U., & Perfecto, C. (2024). Deep Reinforcement Learning techniques for dynamic task offloading in the 5G edge-cloud continuum. *Journal of Cloud Computing*, *13*(1), 94.
15. Hazra, A., Tummala, V. M. R., Mazumdar, N., Sah, D. K., & Adhikari, M. (2024). Deep reinforcement learning in edge networks: Challenges and future directions. *Physical Communication*, *66*, 102460.
16. Tian, Z., & Wang, Y. (2022). Predictive control compensation for networked control system with time-delay. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, *236*(1), 107-124.
17. Wang, W., Han, D., Luo, X., & Li, D. (2023, October). Addressing signal delay in deep reinforcement learning. In *The Twelfth International Conference on Learning Representations*.

18. Bouteiller, Y., Ramstedt, S., Beltrame, G., Pal, C., & Binas, J. (2020, April). Reinforcement learning with random delays. In *International conference on learning representations*.
19. Wang, Z., Gong, T., Huang, S. H., & Yi, B. (2025). Graph Neural Network with Soft Actor-Critic and Attention based Large Model for Intelligent Edge Routing in Consumer Internet of Things. *IEEE Transactions on Consumer Electronics*.
20. Chen, T., Wang, L., Liu, Z., & Su, H. (2022, May). Predictive control of network control systems with unknown transmission delay via LSTM network. In *2022 13th Asian Control Conference (ASCC)* (pp. 1060-1065). IEEE.
21. Wu, J., Du, B., Wu, Q., Shen, J., Zhou, L., Cai, C., ... & Zhou, Q. (2021). A hybrid LSTM-CPS approach for long-term prediction of train delays in multivariate time series. *Future transportation*, *1*(3), 765-776.
22. Zhou, X., Yang, X., Ma, J., & Wang, K. I. K. (2021). Energy-efficient smart routing based on link correlation mining for wireless edge computing in IoT. *IEEE Internet of Things Journal*, *9*(16), 14988-14997.
23. Facenda, G. K., Khisti, A., Tan, W. T., & Apostolopoulos, J. (2023). Deep reinforcement learning for latency-sensitive communication with adaptive redundant retransmissions. *IEEE Transactions on Communications*, *71*(5), 2771-2783.
24. Itahara, S., Nishio, T., & Yamamoto, K. (2021, December). Packet-loss-tolerant split inference for delay-sensitive deep learning in lossy wireless networks. In *2021 IEEE Global Communications Conference (GLOBECOM)* (pp. 1-6). IEEE.
25. Wang, Z., & Fukushima, S. (2020). Control strategy for networked control systems with time delay and packet dropout using linear matrix inequalities. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), 42.
26. Wang, R., Su, C., Du, Q., Xiong, Q., Zhuang, Y., Zhang, X., & Guo, X. (2024). Virtual position predictive control with system delay observer to improve PMLSM position tracking accuracy. *IET Electric Power Applications*, *18*(11), 1605-1615.
27. Cui, L., Pang, B., & Jiang, Z. P. (2023). Learning-based adaptive optimal control of linear time-delay systems: A policy iteration approach. *IEEE Transactions on Automatic Control*, *69*(1), 629-636.
28. Gao, R., Li, Q., Dai, L., Zhan, Y., & Xia, Y. (2023). Workflow-based fast data-driven predictive control with disturbance observer in cloud-edge collaborative architecture. *IEEE Transactions on Automation Science and Engineering*, *21*(3), 2816-2840
29. Gu, X., Pezzutto, M., Schenato, L., & Dey, S. (2025). Optimal Control Selection over the Edge-Cloud Continuum. *arXiv preprint arXiv:2503.07349*.
30. Yang, X., & Ni, J. (2023). A cloud-edge combined control system with MPC parameter optimization for path tracking of unmanned ground vehicle. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, *237*(1), 48-60.
31. Mohajeri, K., Madadi, A., Tavassoli, B., & Rahiman, W. (2023). Discrete-time modelling methodology of networked control systems under packet delay and dropout. *IET Cyber-Physical Systems: Theory & Applications*, *8*(3), 131-148.
32. Wang, Z., & Fukushima, S. (2020). Control strategy for networked control systems with time delay and packet dropout using linear matrix inequalities. *EURASIP Journal on Wireless Communications and Networking*, *2020*(1), 42.
33. Baroumand, S., Zaman, A., & Mihaylova, L. (2023). Attack detection and fault-tolerant control of interconnected cyber-physical systems against simultaneous replayed time-delay and false-data injection attacks. *IET Control Theory & Applications*, *17*(5), 527-541.

34. Gamal, M., Sadek, N., Rizk, M. R., & Abou-elSaoud, A. K. (2016). Delay compensation using Smith predictor for wireless network control system. *Alexandria Engineering Journal*, *55*(2), 1421-1428.
35. Steinberger, M., & Horn, M. (2020). From classical to networked control: Retrofitting the concept of smith predictors. *arXiv preprint arXiv:2010.05486*.
36. Pyrkin, A., & Kalinin, K. (2025). Modified Smith predictor for unstable linear systems. *arXiv preprint arXiv:2507.22243*.
37. Sun, Y., Sun, Y., & Yang, C. (2021). Finite-Time Control of Networked Control Systems with Time Delay and Packet Dropout. *Journal of Control Science and Engineering*, *2021*(1), 3093865.
38. Silva, C. A. G. D., & Santos, E. L. D. (2023). A compensation model for packet loss using Kalman filter in wireless network control systems. *Energies*, *16*(8), 3329.
39. Wang, H., Liu, T., Li, Z., & Liang, X. (2024). Optimal control for networked control system with Markovian packet loss and delay. *Asian Journal of Control*, *26*(6), 3162-3178.

Author Biographies

Padmaja Mishra received her B.Tech (AMIETE) degree in Electronics and Telecommunication Engineering from IETE, NEW Delhi. She received her M.Tech degree in Electronics and Communication engineering from National Institute of Science and Technology, Berhampur. She has received her Ph.D degree from C.V. Raman Global University, Bhubaneswar. Her interests include Network control, IoT, Embedded system design, Machine learning, Artificial Intelligence, Network Security and Wireless sensor networks.

Ajay Kumar Yadav received his B.Tech degree in Electronics and Communication Engineering from Dr. AITH Kanpur. He received his M.Tech and Ph.D from MNNIT Allahabad. Currently he is working as an Assistant Professor at C.V. Raman Global University, Bhubaneswar. His interests include Communication Engineering, Machine learning, Network Control.

Rakhee Panigrahi received the B.Tech degree in electrical engineering from the Veer Surendra Sai University of Technology Burla, Odisha India, in 2002, and the M.Tech and Ph.D. degrees in electrical engineering from the National Institute of Technology, Rourkela, India, in 2009 and 2015, respectively. She is currently an Asst. Professor in the Department of Electrical Engineering, PMEC, Berhampur. Her research interests include VLSI design, estimation techniques with application to power quality, Renewable energy and estimation technique in microgrid islanding detection.

Rajesh Kumar Patjoshi received the B.Tech degree in Electronics and Telecommunication engineering from the Amaravati University, India and the M.Tech and Ph.D. degrees in electronics and communication engineering from the National Institute of Technology, Rourkela, India, in 2010 and 2015, respectively. He is currently an Head and Associate Professor in the Department of Electronics and Communication Engineering at NIST University, Berhampur, Odisha. Her research interests include VLSI Design, Artificial Intelligence, Machine learning, Renewable Energy Systems, Network Control systems for IoT, Microgrids, Embedded Systems, Low-Power and RF VLSI Design. Email:rajeshpatjoshi1@gmail.com