Delay Mitigation in Offloaded Cloud Controllers in Industrial IoT

Saad Mubeen, *Senior Member, IEEE,* Pavlos Nikolaidis, Alma Didic, Hongyu Pei-Breivold *Member, IEEE*, Kristian Sandström *Member, IEEE* and Moris Behnam, *Member, IEEE*

***Abstract*—This paper investigates the interplay of cloud computing, fog computing and Internet of Things (IoT) in control applications targeting the automation industry. In this context, a prototype is developed to explore the use of IoT devices that communicate with a cloud-based controller, i.e., the controller is offloaded to cloud or fog. Several experiments are performed to investigate the consequences of having a cloud server between the end device and the controller. The experiments are performed while considering arbitrary jitter and delays, i.e., they can be smaller, equal or greater than the sampling period. The paper also applies mitigation mechanisms to deal with the delays and jitter that are caused by the networks when the controller is offloaded to the fog or cloud.**

***Index Terms*—Industrial IoT, Fog computing, cloud computing, industrial automation systems.**

I. Introduction

Cloud computing [1] and Internet of Things (IoT) [2], [3] are two notable concepts that have evolved significantly over the past few years. Cloud computing is an operational scheme that provides network-based services such as computational power, storage and networking to users within many industrial and application domains. It offers a pool of virtualized computing resources at various levels, covering infrastructure, platforms or software delivered to users as on-demand services from the cloud. In this way, cloud computing is changing the services consumption and delivery platform as well as the way businesses and users interact with IT resources.

IoT extends the cloud computing concept beyond computing and communication to include everything, i.e., also the physical devices. Industrial IoT uses sensors, machine-to-machine (M2M) collaboration and various technologies to gather and analyze data from physical and virtual world for optimized operations and providing services. In IoT, devices are connected through a network. They share data, information and even resources to accomplish their goal or increase total system intelligence. Accordingly, cloud computing and IoT can provide services to consumers and businesses, allowing organizations to become more agile and flexible in pursuing new revenue streams and new business models.

Despite numerous advantages of cloud computing, there are some limitations such as high delays that render cloud computing unfavorable to the industrial control systems that have low-delay requirements. In order to overcome the drawbacks of cloud computing, a local cloud computing architecture called the *fog computing* has been introduced recently [4]. According to Cisco, fog computing extends the cloud computing away from the cloud computing data centers and towards the edge of the network [5].

A representation of the three-tier fog computing architecture proposed by Cisco is shown in Fig. 1. The first tier of the architecture gets data from the local embedded devices and sensors. This tier is used for the M2M interactions and supports *real-time systems* with stringent timing requirements. A real-time system is required to provide its logically correct response within the time that is mandated by the specified timing requirement(s). The M2M interaction is the key aspect to increase the intelligence of the "things" [6], [7]. The response times provided by the first tier are in the order of milliseconds to sub-seconds. The first tier can also filter or pre-process the vast amount of data before it gets sent to the higher tiers as shown in Fig. 1. In this way the higher tiers only get the selected information and can easily communicate with multiple first-tier fog nodes. The second and third tiers support the human-to-machine (H2M) interactions. These tiers can also filter data before sending them to the cloud for further processing. The second tier provides response times in the range from seconds to minutes. Whereas the third tier provides response times from minutes up to days. Note that the higher the tier, the wider the coverage range. Cloud is responsible for the global coverage which is used for intelligent business analytics based on long periods.

A. Problem Statement

There are several works that investigate the application of cloud computing infrastructures in the industrial automation systems, e.g. [8], [9]. In majority of the existing solutions, clouds have been used for monitoring industrial processes and for creating short- and long-term reports for different characteristics of the shop floor. Although cloud computing seems adequate to monitor industrial processes, it cannot be used to control industrial machines because of unpredictable wide-area network (WAN) delays [10]. Therefore, utilizing local resourceful servers, local clouds, can alleviate unpredictable WAN delays. While local clouds seem to offer an attractive solution, it is expensive to acquire and maintain such infrastructure.

Recent advantages in IoT can be leveraged to solve the aforementioned cost problem. Accordingly, costly industrial

S. Mubeen, M. Behnam, P. Nikolaidis, A. Didic are with Mälardalen University, Västerås, Sweden, e-mail: {saad.mubeen, moris.behnam, pns13002, adc13001}@mdh.se.

H. Pei-Breivold is with ABB Corporate Research, Västerås, Sweden, e-mail: hongyu.pei-breivold@se.abb.com.

K. Sandström is with the Swedish Institute of Computer Science (SICS), Västerås, Sweden, kristian.sandstrom@sics.se.

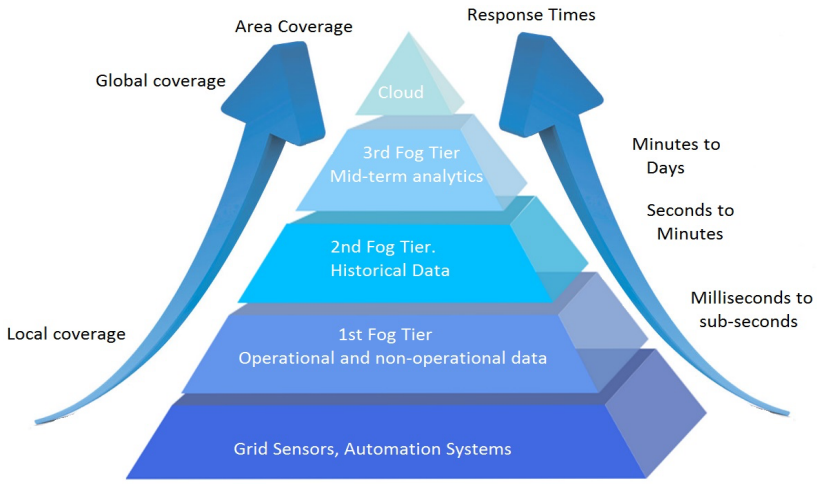


Fig. 1: A representation of the three-tier fog architecture by Cisco.

controllers can be replaced by cheaper components that can sense the process and send data to the local cloud. Since the devices in the context of IoT are not very resourceful, they cannot handle intensive tasks. Hence, a local powerful cloud infrastructure is needed. A main advantage of devices in the IoT concept is that they can enter or leave the network dynamically without influencing the rest of the system. As a result, observability of the system can be increased or decreased on demand. In addition, the centralized controller running on the cloud can perform optimizations or changes in the process that can instantly affect the entire system.

IoT, cloud computing and fog computing can be combined to form a new operational scheme that can benefit the industrial applications in terms of scalability, flexibility and cost effectiveness. In this scheme, the IoT devices communicate with the controller located on the local cloud server, providing real-time control as a service. One of the main challenges that is faced when a networked controller is included in the closed-loop control system is how delays affect the performance of the system. Another challenge is how to mitigate or even compensate these effects on the end device.

A typical closed control loop is shown in Fig. 2. It consists of a controller that controls a physical process through sensors and actuators. The controller is usually located close to the actual process. Fig. 3 depicts the closed loop control when the controller is offloaded to the network, e.g., to a fog node or a cloud. In this case there are network delays in the control loop that should be accounted in the response times of the systems. These delays can have serious effects on the systems with real-time requirements such as making them unpredictable and unstable. A detailed investigation is needed to show the feasibility of offloaded controllers to the fog versus cloud in the delay-sensitive industrial automation systems.

The first tier of the fog architecture, shown in Fig. 1, can be used for closed loop control between the industrial machines and the fog nodes. The second and third tiers can be used to process data to monitor the machines and prevent future hardware failures. While the latter has been investigated in the existing works, this paper focuses on the closed loops with offloaded controllers to the fog as well as to the cloud.

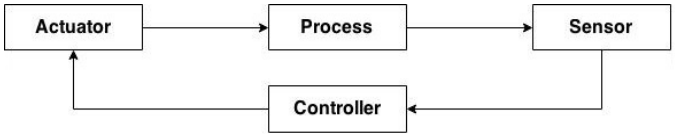


Fig. 2: Closed control loop.

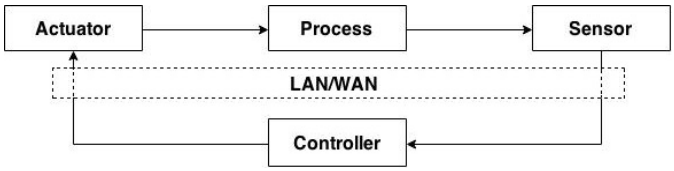


Fig. 3: Closed loop with networked controller.

The devices in the context of IoT are not very resourceful to support computation-intensive tasks. Using cloud computing, control can be provided as a service allowing the execution of computation-intensive algorithms in the cloud. However, according to [10], cloud computing cannot be used to control the automation industrial machines, mainly, due to unpredictable network delays. In order to overcome the limitation of cloud computing, this paper advocates the effectiveness of local cloud computing infrastructure, especially fog computing for the delay-sensitive industrial automation systems.

B. Paper Contributions

In order to address the challenges discussed in the previous subsection, we develop a control system application prototype by exploiting the principles of IoT, cloud computing and fog computing. Using the prototype, we perform a number of experiments to investigate the impact of local and wide area networked controller on the closed loop control. In order to do the performance evaluation, we consider arbitrary jitter and delays, i.e., they can be smaller, equal or greater than the sampling periods. Additionally, we apply two delay mitigation mechanisms for the end device. These mechanisms do not

use any internal information from the controller, in fact these mechanisms rely only on the received data.

It should be noted that the goal of this paper is not to invent new techniques for interplay of IoT, cloud computing and fog computing in the industrial automation, but to investigate and show the feasibility of existing techniques in this area. The contributions in this paper include a comparative evaluation of various scenarios in the above-mentioned context including (1) local controller, (2) controller offloaded to the cloud and (3) controller offloaded to the fog. The contributions also include the application of delay mitigation techniques, i.e., prediction methods have been implemented in the end devices, whereas the adaptive PI algorithm is implemented in the cloud-based controller. The techniques, prototype and experiments that are presented in this paper are applied in the industrial settings within the domain of the automation control systems, provided by one of our industrial partners. The proof of concept provided in this paper serves as the foundation for the advanced cloud-based controllers which will be eventually used in the automation industry, e.g., robotic arm and collaborative machines.

C. Paper Layout

The rest of the paper is organized as follows. Section II discusses the related work. Section III presents the prototype architecture. Section IV presents the experimental evaluation. Section V provides the outlook and discussion. Section VI concludes the paper and discusses the future work.

II. Related Work

In the past few years, IoT has gained wide popularity in many domains such as smart cities, smart grid, commodities tracking and monitoring, logistics, security, transportation, health monitoring, home automation, environmental and agricultural applications, just to name a few [11], [12], [13], [14]. Recent studies predict that IoT will bring around 26 billion devices to the connected world by 2020 [15]. According to [10], the application of IoT in the industrial automation is addressed to a very small extent. IoT solutions for the industrial automation are still evolving.

Cloud computing has recently drawn the attention of the automation industry [10], [16]. There are several works that investigate the usage of cloud computing in the industrial automation systems. In this regard, preliminary studies and initial experiences of utilizing cloud computing in the industrial automation are discussed in [17], [18]. Langman and Meyer [8] highlight the increase in the interest of the automation industry in applying the concepts of cloud computing to the process control. They propose a new service-oriented architecture, called the web-oriented automation system that employs the principles of cloud computing in the industrial automation. Vogel-Heuser et al. [19] propose a two-layered architecture based on cloud computing for the industrial automation. In this architecture the upper layer provides the process control and management services. Whereas the lower layer provides the field bus communication and control services. According to [18], the work in [19] lacks a concrete solution to implement the proposed architecture in the industrial automation.

There are several works that utilize cloud computing for the purpose of monitoring data and managing control processes in the industrial automation [20], [21]. According to [8] and [17], the majority of existing approaches and solutions in the context of cloud computing in the industrial automation focus on the higher levels than the field-level. Utilization of cloud computing in control loops in the industrial automation has received very less attention. In [22] Chen et al. propose robot as a service in the industrial automation. In [23] Givehchi et al. investigate the use of a virtualized Programmable Logic Controller (PLC). They show that the virtualized PLC can provide similar behaviour to a hardware PLC, but it's performance decreases considerably with an increase in the sampling frequency. The issues concerning the real-time aspects and timing predictability in the industrial automation systems employing the principles of cloud computing need a detailed investigation.

Cisco has recently provided the vision of fog computing [4], [24]. Since its introduction, the applications of fog computing have been explored in many domains including wireless sensor and actuator systems, health data management, connected vehicles and smart grid [5], [25]. Gazis et al. [26] discuss various challenges that are faced when fog computing is used in the context of industrial IoT. Sarkar et al. [26] develop a mathematical model of fog computing to investigate its applicability in IoT. None of these works target the delay-sensitive industrial automation control systems which is the main focus of this paper.

Stojmenovic [7] explores the M2M networks in fog computing. Madsen et al. [6] explore the reliability aspect of fog computing. Stojmenovic and Sheng [27] discuss security concerns, particularly the man-in-the-middle attacks in fog computing. Hong et al. [28] address the mobility aspect in fog computing by introducing the mobile fog model. In [29], fog computing is used for one of the most computation-intensive tasks, namely the brain-state classification, where it achieves low response times to enable augmented brain-computer interaction. Yi et al. [25] compare the delays in fog and cloud by means of a synthetic experiment, however a full-fledged industrial implementation is left for the future work. On the other hand, investigation of delay effects due to offloaded controllers to fog and cloud in the industrial automation is targeted in this paper. Hong et al. [30] propose a heuristic for dynamic deployment of programs to the end devices and fog nodes. In this regard, they propose a deployment platform and perform experiments. However, they do not investigate the effect of delays and delay mitigation techniques in fog and cloud. Furthermore, there is no industrial case study that evaluates the fog combined with cloud (fog-cloud) architecture; and how a system can benefit from a smart combination of these two platforms.

Offloading is used to make the decision which task should be offloaded to meet the system requirements. Computational offloading algorithms focus on meeting the timing constraints of their tasks or increasing throughput and improving quality of service . There are several offloading techniques such

as [31], [32], [33], [34], [35], [36] that can be used complementary to our work. We investigate the control-as-a-service scenario where control is offered as a cloud service. We explore the advantages and disadvantages of executing the service in cloud, fog and locally. Our offloading goal is not to increase performance but to offer a centralized control for the industrial automation systems.

The problem of Networked Control Systems (NCS) has been addressed in 2001 in [37]. Plenty of work exists regarding this problem such as [38], [39]. In NCS the controller is tuned in order to compensate network delays or data dropout. In comparison, in this paper we offload the local tuned controller to the fog and cloud. We investigate the effects of offloaded controller on the response times and jitter of the industrial automation systems. Furthermore, we utilize the mitigation mechanisms that are aimed to mitigate the extra delays introduced by the network due to offloading the controller to the fog and cloud.

In summary, cloud infrastructures are used for monitoring industrial machines, improving business intelligence and reporting. Moreover, cloud computing is used in high-level procedures and not in low-level procedures that require faster and predictable responses by the systems. In order to overcome these problems, fog computing is utilized. However, this technology is yet to mature.

III. Proposed System Architecture

A. Prototype Architechure

A prototype is built in order to create a representation of the real case. The setup, shown in Fig. 4, consists of an end device connected to a switch through which it can reach a local server (fog node) or a remote one (cloud node). Since the setup is realised in a local environment, a delay emulator is used for simulating the distance between the device and the cloud node.

The fog and cloud nodes are set up as the Ubuntu servers. For emulating the connection over WAN with the cloud, a software solution called the Wide Area Network Emulator (WANEM)[1] is used. WANEM is based on the Linux kernel and it utilizes NetEm[2], among other, functionalities in a simple graphical user interface provided on a live Knoppix[3] disk. WANEM enables specifying typical network problems, such as delays, jitter, packet loss, packet corruption, connection loss, etc. It also enables specifying the bandwidth, setting a correlation percentage for each characteristic and a few distributions for the delays (normal, pareto, paretonormal) for simulating various realistic internet conditions. All communication routed through WANEM is affected by the set parameters. The end device is an Arduino Uno[4] running a small program that involves sensing and actuating. It sends the sensed value to the server (fog or cloud) where a control value is calculated. The control value is then sent back to the Arduino to actuate.

[1]http://wanem.sourceforge.net/.

[2]http://www.linuxfoundation.org/collaborate/workgroups/networking/netem.

[3]http://knoppix.net/.

[4]http://www.arduino.cc/en/Main/ArduinoBoardUno.

The control loop consists of a photo resistor and an LED. The controller can change the brightness of the LED until it reaches the desired value (set to 700 in the experiments). It should be noted that the photo resistor and LED are considered as a sensor and actuator in the prototype for the proof of concept. The photo resistor and LED can be generally replaced by any other complex sensor and actuator respectively.

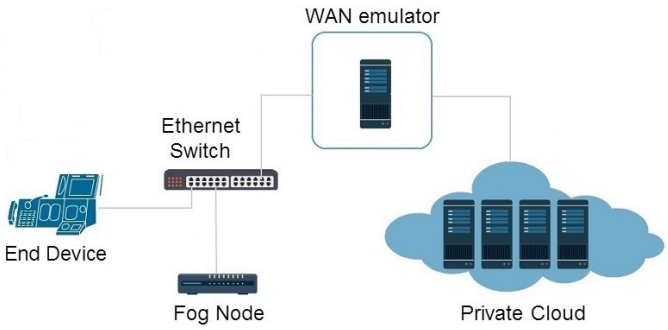


Fig. 4: Architecture overview.

For the device-server communication, two different approaches are used. The first one, called polling, is listed in Algorithm 1. It involves the Arduino waiting for, or *polling*, the server to establish the connection. In this case, after sampling the sensor value and sending it to the server, the device does nothing until the response from the server is received and could potentially wait forever for it. This approach could be used in a scenario where the sensor and actuator are placed on the same device. This approach is also suitable in the scenario where sensor and actuator are located on separate devices.

Algorithm 1 Polling

```
1:  procedure POLLING CONTROLLER
2:      begin:
3:      sensorValue ← readSensor();
4:      sendMsgToServer(sensorValue);
5:      loop:
6:      if resultAvailable() then
7:          result ← readResult();
8:          applyValue(result);
9:          goto begin;
10:     end if
11:     goto loop;
12: end procedure
```

The second approach, called non-polling, is listed in Algorithm 2. In this case the device sends the sensed value and checks if a reply is available. If there is no message, the device does not wait for the server to reply, hence it is *not polling* the server. Instead, the device repeats its cycle, sending another value and checking for a received message again. On the server side this means that multiple messages with the same value will be received. This might lead the controller to think that the calculated values are taking no effect and thus respond more aggressively, which is not a desired behaviour. This approach suits a case where the sensor and actuator are located on separate devices.

Algorithm 2 Non-polling

```
1: procedure NON-POLLING CONTROLLER
2:     begin:
3:     sensorValue ← readSensor();
4:     sendMsgToServer(sensorValue);
5:     if resultAvailable() then
6:         result ← readResult();
7:         applyValue(result);
8:     end if
9:     goto begin;
10: end procedure
```

The proposed local cloud architecture prototype is consistent with the requirements from our industrial partners in the automation domain. The number of end devices in the final system on the shop floor is expected to be in the order of a few thousands. It should be noted that the local cloud infrastructure complemented by fog computing will be used for not only monitoring, but also for controlling the industrial automation systems consisting of several thousand end devices. That is, the controller will be offloaded to the fog node(s) in the case of delay-sensitive applications. Whereas, the controller will be offloaded to the cloud in the case of the applications that are relatively less sensitive to network delays. In addition, the cloud will be used for monitoring these applications.

B. Delay Mitigation

In order to mitigate delays caused by the networked controller, having some simple mechanism on the device side is desirable. Typically, model-based controllers are used for control loops with time delays (also referred to as dead time) in which the sensing needs to be done at a certain time after the actuation. Note that the controller has to wait for the dead time before it can get any feedback from the process regarding the recent control value. The controller then contains a model of the process it is controlling and is able to provide adequate response based on the behaviour of the model. However, these controllers are not intended for variable delays and do not fit into the idea of the desired simple mechanism. In our case the control is achieved using a PI controller. Since the delays affect the integral term of the controller, an adaptive PI controller [40] would be sufficient to deal with the delays. The adaptive controller can be tuned to adapt to the delays and is simple enough for our purposes. Statistical prediction methods are also used for predicting the delays [41] or, in dynamic offloading, to asses the current state of network parameters (such as bandwidth or loads on the server end) to avoid measuring them too often which increases overhead [42]. This is another method that is considered in our experiments, where the predictions could be used to predict a missed value from the controller. Due to the nature of the algorithms we use (polling and non-polling) and the considered mitigation methods, the adaptive PI controller suits the non-polling approach while the prediction methods are used on the polling approach.

In order to improve the polling method, a timeout is added and a prediction method is used in case no message is received from the server before the timeout. When the timeout occurs, the predicted control values are applied at the device. This timeout is set to the average round trip time (RTT) for the device-server communication, which is 18 milliseconds. Two prediction methods are used: exponential moving average and double exponential smoothing model. Both of these prediction methods make use of all the data collected, without having to store or manage a large number of variables or the need to collect more than two values to start off. The exponential moving average [43], also called exponential smoothing, calculates a weighted average of the previous data values x_t, as presented in (1), where A is a value between 0 and 1. The predicted value for the next cycle, F_{t+1}, is the value calculated in the current cycle (2).

$$s_t = Ax_t + (1 - A)s_{t-1} \tag{1}$$

$$F_{t+1} = s_t \tag{2}$$

The double exponential smoothing average model is calculated by the "Holt model" forecast [44]. It calculates two terms in each cycle, as seen in (3) and (4), and has two smoothing factors A and B, both of which take values between 0 and 1. The second term b_t represents the change in the slope, or the trend. This method calculates F_{t+m}, the predicted value of x_{t+m} at time $t + m$, $m > 0$, by using (5)

$$s_t = Ax_t + (1 - A)(s_{t-1} + b_{t-1}) \tag{3}$$

$$b_t = B(s_t - s_{t-1}) + (1 - B)b_{t-1} \tag{4}$$

$$F_{t+m} = s_t + mb_t \tag{5}$$

In case of the non-polling approach, due to delays, the controller receives the same value multiple times before a change is registered. Note that this situation cannot occur in the polling approach because the fresh sensor values are sent from the end device to the controller only when the control signals corresponding to the previous sensor value have been received from the controller. When the controller receives multiple messages with the same value, it affects the integral factor of the PI controller. In this case an adaptive PI controller is suitable to mitigate the affects of the delays on the controller. A smoothing factor, a, is calculated based on the sampling period and the round trip time, as shown in (6) to lessen the influence of these changes caused by delays.

$$a = \frac{sampling\ period}{sampling\ period + RTT} \tag{6}$$

This smoothing factor is then applied to the received values before they are used by the actuator.

IV. Experimental Evaluation

We performed a large number of experiments on the prototype discussed in the previous section [45]. In this section we

explore the effects of delay and *jitter* on the system's response, due to offloaded controller to the fog or cloud. Jitter refers to the variable delay between 0 and its specified value. The parameters of interest in the system's response are *overshoot* and *settling time*. Overshoot is defined as the difference between maximum and targeted values of the response. Whereas, the settling time refers to the amount of time that elapses from a change in the input to the corresponding stabilized response. In the following subsections, first we consider the system's response without using delay mitigation mechanisms. Then we consider the effect of delay mitigation mechanisms on the response of the system under high level of delays and jitter. The sampling period is set to 14 ms in all the experiments.

A. Case 1: Response Without Mitigation Mechanisms

This case is further divided into two cases. In the first case, we consider the delays and jitter do not exceed the sampling period. Whereas in the second case, we investigate the effects of delays and jitter that exceed the sampling period. Both these cases are investigated using the polling and non-polling approaches.

1) When delay and jitter are smaller than or equal to the sampling period: In this case, the prototype is tested with delays and jitter that are comparable to the sampling period. The effects of delays and jitter on the response of the system under polling scheme are shown in Fig. 5a and Fig. 5b respectively. Similarly, the effects of delays and jitter on the response of the system under non-polling scheme are shown in Fig. 5c and Fig. 5d respectively. The curve identified with "Local" in Fig. 5a and Fig. 5c represents the response of the system with a local controller, i.e., without offloading the controller to the fog or cloud. The response of the system due to direct communication with the server is labeled as "fog" in Fig. 5a and Fig. 5c. In the case of offloaded controller to the fog node, the delay is less than half a millisecond. The network delays in the rest of the responses in Fig. 5a and Fig. 5c are 1 ms and 10 ms.

When polling scheme is used as shown in Fig. 5a the disturbance in the response curve is not big, especially for the fog and 1 millisecond communication delay. As expected, the network delays that are less than the sampling period can be tolerated without any significant degradation in the performance of the control loop. That is, there is a very small overshoot in all the four curves. The settling time is also small and very similar except for the response corresponding to 10 ms delay. Therefore it can be concluded, that offloading the controller to fog or cloud servers does not cause any significant degradation of the response under polling scheme as long as the network delays are smaller than the sampling period.

Similarly, when non-polling scheme is used as shown in Fig. 5c there is no significant effect on the system's response as long as delays are kept below the sampling period. However, the degradation in the performance becomes severe when messages are queued in the network. As we can notice in Fig. 5c, when the delay is 10 ms the device sends two messages to the controller with the same value before it receives a response from the controller. As a result, the "I" part of the PI controller is doubled, thereby causing bigger changes to the actuator which downgrades the systems' performance. That is, when the system receives the second message, it computes the same control value as that of the previous message. This, in turn, forces the controller to compute a larger control value in order to decrease the error because the previous value does not affect the system. Obviously, such interpretation from the controller is wrong since the first computed value is not yet applied.

We investigate how jitter affects the performance of the controller in Fig. 5b and Fig. 5d. The effect of four different values of jitter on the response of the system is shown in Fig. 5b for the polling approach. When the jitter is smaller than the sampling period, it does not significantly affect the performance of the controller. However, the jitter increases the overshoot but the system still manages to stabilize itself under the settling time of 120 ms in all the cases. In the case of non-polling approach, the effect of four different values of jitter on the response of the system is shown in Fig. 5d. The results in this case are similar to those in the case of the polling approach.

2) When delay and jitter are larger than the sampling period: In this subsection, we investigate the effect on the response of the system due to delays and jitter which are considerably larger than the sampling period. Once again, we consider both polling and non-polling approaches. In this set of experiments, the values of the delays and jitter are increased until the system starts to oscillate. In Fig. 6a, the polling approach is used and the system becomes unstable when the delay exceeds 50 ms. The oscillation for the delay more than 50 ms exceeds the limit of the steady state error therefore the system is considered unstable. The settling time for 50 ms delay is less than 600 ms.

In Fig. 6c, the non-polling approach is used. It can be noticed that the system tolerates smaller delays compared to the polling approach. The system eventually stabilizes when 15 ms delay is applied, however the settling time is much higher as compared to the polling approach. The system becomes unstable for 23 ms delay. Moreover, the systems' response downgrades significantly from 15 ms delay. The degradation of system's performance, for this approach, starts when the sensor sends consecutive messages before the actuator receives a message. That happens when delays exceed the sampling period.

In Fig. 6b and Fig. 6d, the effect of jitter in the control loop is depicted for polling and non-polling approach respectively. As it can be noticed, the overshoot increases as the jitter increases, but the system manages to reach the steady state after some time. Jitter does not affect the system in the same way that delays do. Because of the jitter, the controller can receive some values before the sampling period is exceeded. Also the controller can receive some values that exceed the sampling period by a small amount. Since jitter is a variable delay, it may affect the control loop in a different way on every run.

3) Discussion on settling time and overshoot in polling and non-poling schemes: The comparison between the effects of delay and jitter on the controller shows that both affect the settling time since it takes more time to get enough responses

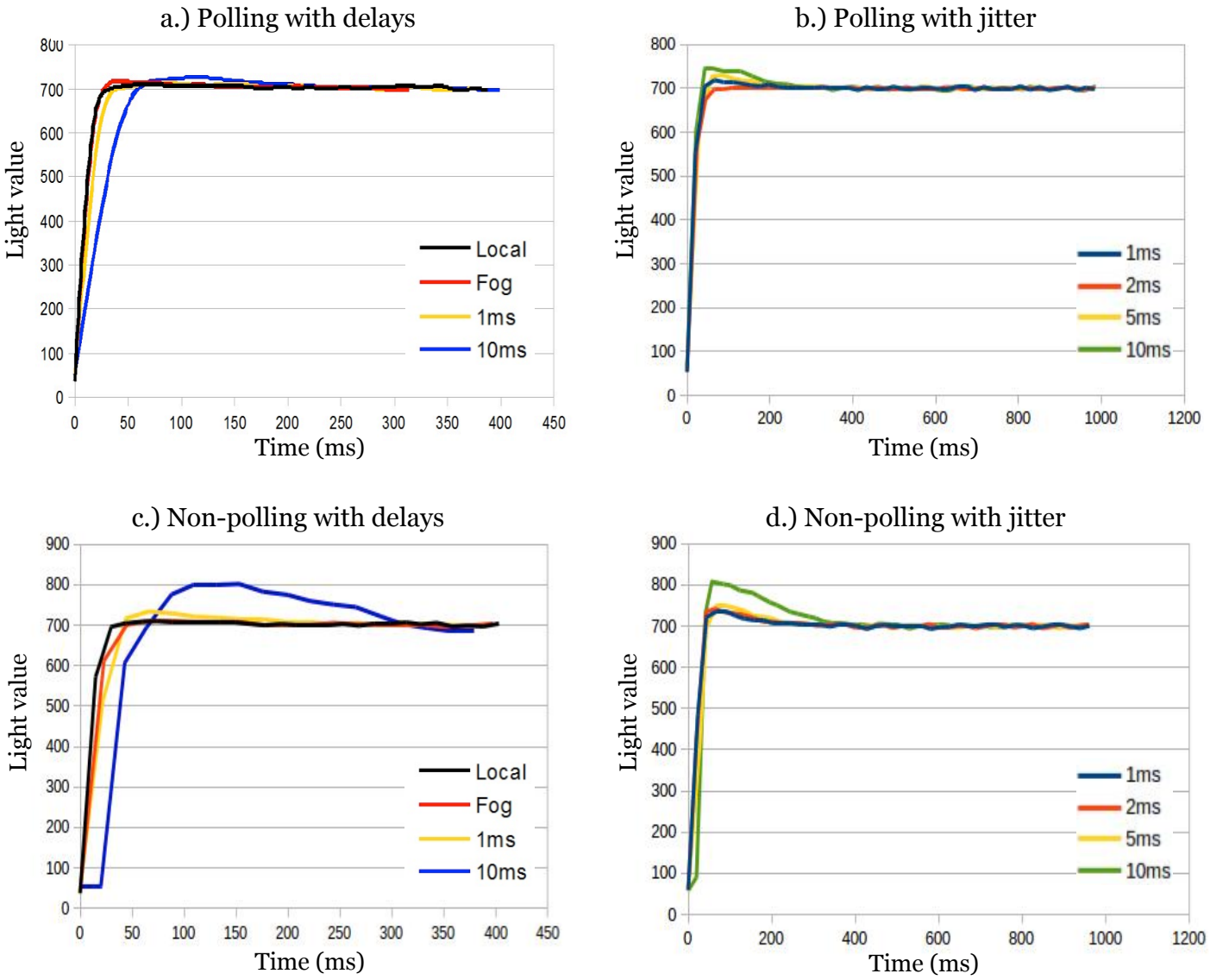


Fig. 5: System response under delay and jitter smaller than or equal to sampling period, both polling (a and b) and non-polling (c and d) approaches.

to stabilize the system. Both the delay and jitter also affect the overshoot of the controller. Additionally, the polling approach tolerates delays better than the non-polling approach. The overshoot is plotted against the Round Trip Time for polling and non-polling approaches in Fig. 7 and Fig. 8 respectively. In these figures, "Local" represents the case when the controller is not offloaded to fog or cloud. Hence the network delay in this case is considered as zero. The label "Fog" represents the case when the controller is offloaded to the fog server. In this case the delay is very small (equal to half a millisecond). The overshoot for the non-polling approach increases faster as compared to the polling approach. Similarly, the settling time is plotted against RTT for polling and non-polling schemes in Fig. 9 and Fig. 10 respectively. It can be seen that the polling scheme results in a smaller settling time compared to the non-polling scheme. Moreover, from all the four graphs we can verify that offloading the controller to fog nodes does not affect the system. Consequently, the local servers can be included in the control loop without performance degradation.

B. Case 2: Response with Delay Mitigation Mechanisms

This case is further divided into two cases. In the first case, we use adaptive PI controller for mitigating delays due to offloaded controller to the cloud. Whereas in the second case, we apply prediction/forecasting methods to mitigate the delays.

1) Mitigating delays using Adaptive PI Control: In this case, we investigate the effect of delay mitigation mechanism using the adaptive PI controller on the response of the system. We show the results using the non-polling scheme since the adaptive PI control inherently suits more to this scheme. In the first experiment, a delay of 20 ms is considered as shown in Fig. 11a. The smoothing factor is set to 0.31 according to the formula in (6). Compared to the behavior of the controller

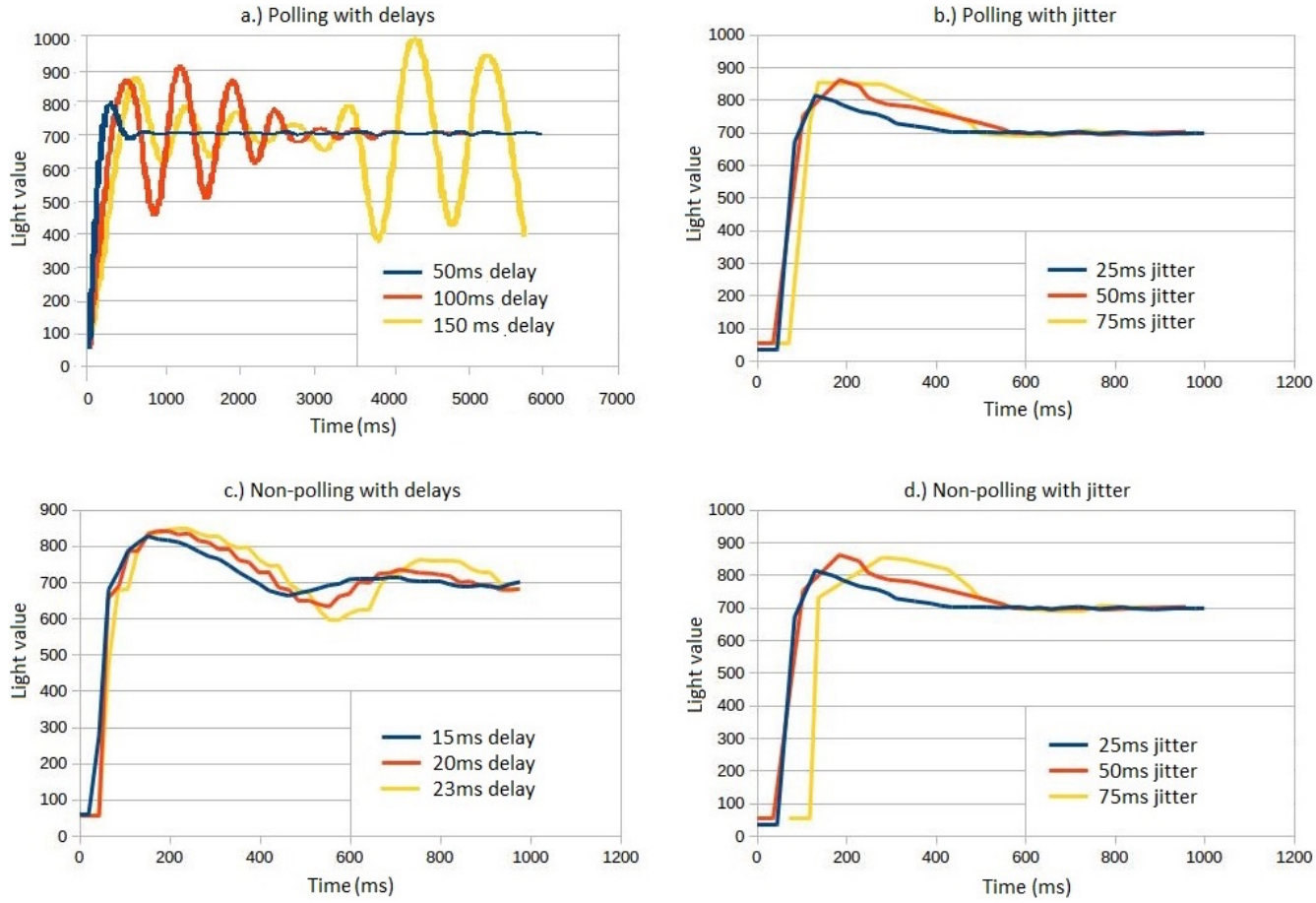


Fig. 6: System response under delay and jitter greater than sampling period, both polling (a and b) and non-polling (c and d) approaches.

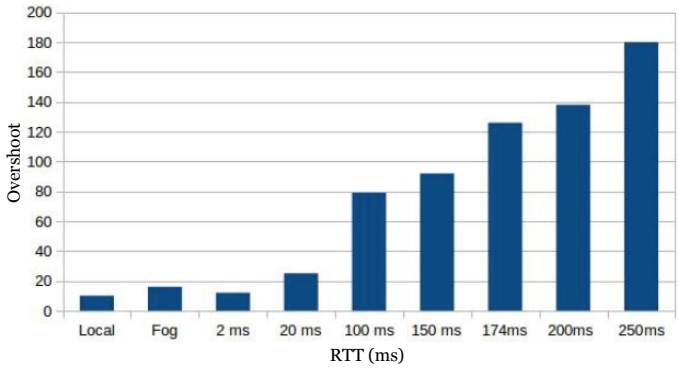


Fig. 7: Overshoot in various scenarios using the polling approach.

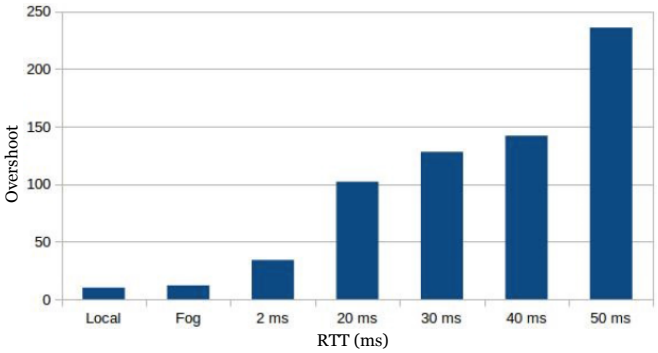


Fig. 8: Overshoot in various scenarios using the non-polling approach.

before the smoothing factor is applied, overshoot has been decreased from 142 to 56 and settling time from 1415 ms to 601 ms. In the second experiment, a delay of 25 ms is applied. The system is unstable without delay mitigation mechanism as shown in Fig. 11b. In this experiment the smoothing factor is set to 0.26. When the smoothing factor is applied the system manages to stabilize after 686 ms with an overshoot of 59.

In the next two experiments, the jitter is 25 ms and 75 ms as shown in Fig. 11c and Fig. 11d respectively. The smoothing factor is computed and RTT is set to the maximum possible value. The smoothing factors are set to 0.26 and 0.1 respectively. In the case of 25 ms jitter, the overshoot has decreased from 134 to 0 and the settling time from 384 to 199. When the jitter is set to 75 ms the system becomes unstable. Utilizing the smoothing factor the system manages to stabilize after 1264 ms with an overshoot of 62.

2) Mitigating delays using prediction methods: In this case, we investigate the effect of delay mitigation mechanism, using

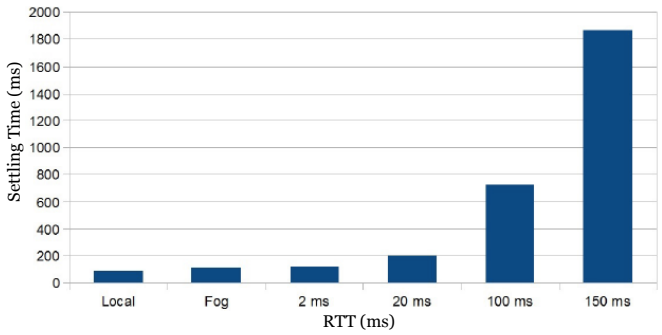


Fig. 9: Settling time in various scenarios using the polling approach.

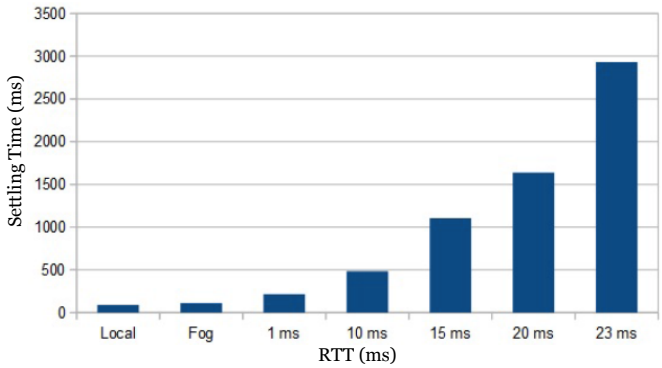


Fig. 10: Settling time in various scenarios using the non-polling approach.

the prediction mechanisms discussed in Section III-B, on the response of the system. Fig. 12a shows that the system is under the delay of 75 ms. In this case, the two prediction methods are compared with respect to the response of the polling method when executed with the same delay without delay mitigation. While the exponential moving average has a smoother transition, the double exponential smoothing has a faster settling time. Both methods, however perform better than the original system.

Similar results are gained when a 100 ms delay is applied, as shown in Fig. 12b. We can notice that the responses of both methods have disturbances in the beginning. This is because a smaller number of messages are available in the message buffer at the beginning as compared to later times. This means, the system has to rely on the prediction methods for input. However, the prediction methods depend on the values from the server to improve their accuracy. This is less of a problem once the system becomes more stable, since changes in the slope are smaller and therefore it is less likely to predict a value that stands out significantly.

As for the jitter, the system performs much better as compared to the delay. Fig. 12c and Fig. 12d show the prediction methods performing with 25 ms and 75 ms of jitter, respectively. We can see that in both the cases, there is barely any overshoot. Moreover, the curves are much smoother in the beginning as compared to the results with constant delays. This is due to variations in the delay. The system performs better when the system uses less predicted values in the beginning. In the case of jitter there is barely any noticeable difference between the two prediction methods, although the exponential moving average settles the response slightly faster.

V. Outlook and Discussion

The controller is clearly affected by the longer delays and jitter. However, the system manages to stabilize when the delays are smaller than the sampling period for both polling and non-polling approaches. Introducing a local server in the control loop adds a small delay of 2-3 ms and could be considered. However, the setup should be tested with increased load on the server to see how it would affect the response. The adaptive PI controller is an intuitive solution. It manages to stabilize the response of the system, and offers a great improvement to the non-polling approach in all cases. However, it has no proof for the robustness of the algorithm. It should be examined on a more complex system or have a mathematical proof in order to prove its robustness. The prediction methods perform better under jitter than under delays. This is because both methods depend on updating their calculations, meaning they depend on getting correct values from the server. In the case of delays, most predictions happen in the start while the control values change and before the system reaches its stable point. As the delays increase, the system relies on more consecutive predicted values, worsening the response of the system. In the case of jitter, the total amount of predictions made is roughly the same as for the delays, but the predictions are more distributed over time rather than concentrated in the start, and thus the response is much better. Even though the delay mitigation mechanisms manage to improve the responses under delays and jitter, none of the results are comparable to the local control with no delays. The system examined here is a relatively simple case. However, the results are reasonable considering the ratio of the sampling period of the process and the delays applied. Most of the research regarding hosting controllers on a cloud consider a more comfortable difference between the sampling period and the network delay. Whereas, we considered arbitrary delays and jitter between the end device and the offloaded controller.

One important conclusion that can be drawn from the experimental evaluation is that there is a strong need for the development of a smart resource management technique. Such a technique should be able to dynamically (at run-time) decide whether to offload the controller to a remote server or not. In the case of offloading requirements, it should be able to decide whether to offload the controller to a fog server, private cloud or public cloud. The decisions should be made on the basis of several aspects such as admission control information, application-specific heuristics, and information about the global resources from the cloud. One way to implement such technique is to develop a multi-level resource manager. That is, it can be implemented at various levels including the public cloud, private cloud, fog or even locally in the “thing”. However, the level of intelligence and resource management capabilities will be different at different levels. Together with our industrial partners, we are currently investigating all these aspects.

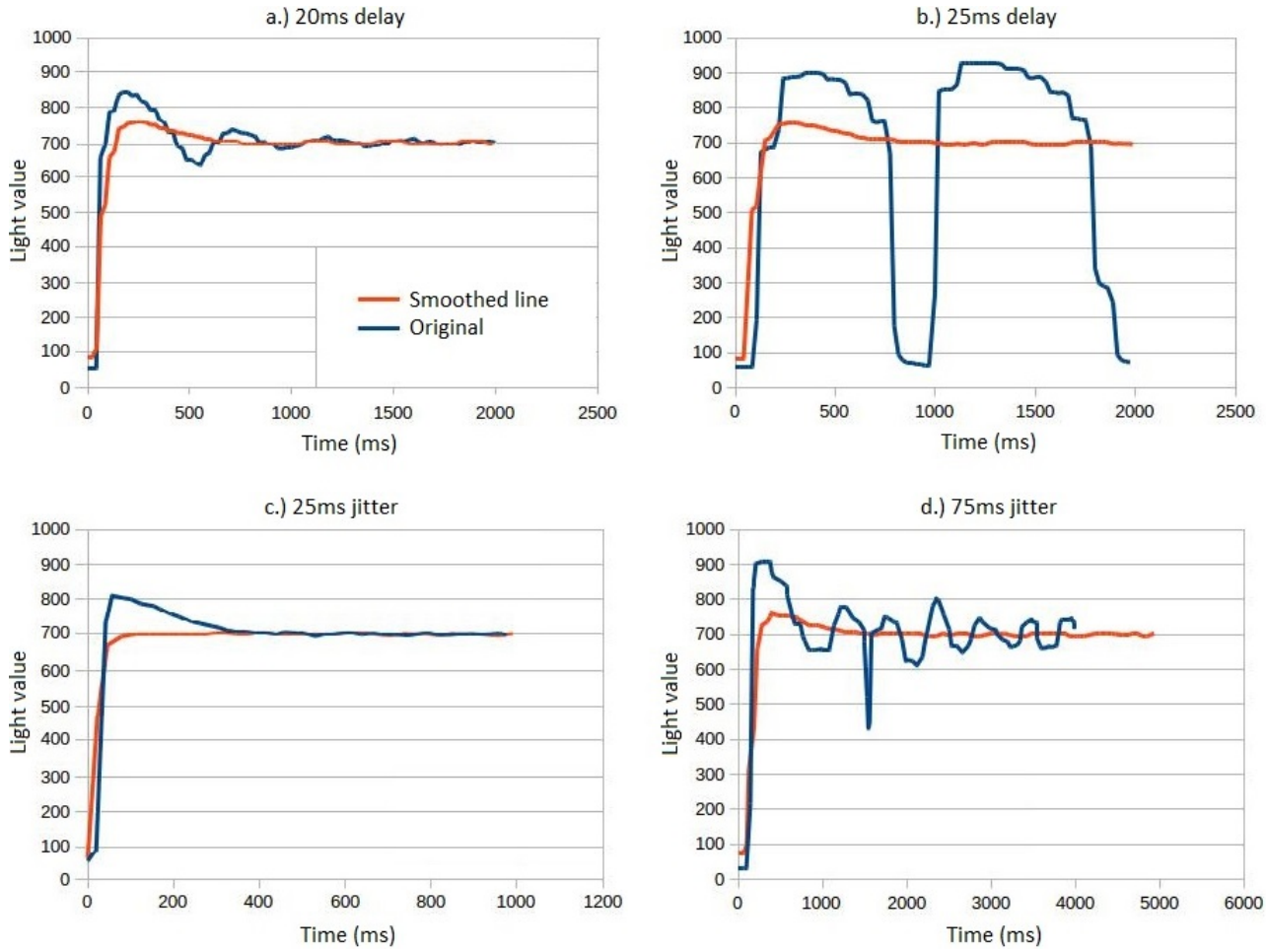


Fig. 11: System response using adaptive PI control under different delays (a and b) and jitter (c and d).

VI. Conclusion and Future Work

In this paper we have investigated the effects of offloading the controller to the fog and cloud. We have built a prototype to create a realistic scenario. Moving the controller to a remote server degrades system's performance. Even small delays affect the control loop when they exceed the sampling period. The forecasting methods that we have investigated, work better with variable delays because the predicted values that are used are more scattered, as opposed to constant values where the predictions are concentrated at the beginning. However, these methods are challenged in the case when consecutive predicted values are needed, especially in the start up of the control system. Whereas, the adaptive PI controller uses an intuitive idea for mitigating delays inside the network. A formal proof for the robustness of such approaches is needed.

Since not much research has been conducted in this area, there are some possibilities left to try out in the future. The polling and non-polling approaches can be expanded into a more sophisticated system. Moreover, loads can be introduced on both ends to simulate a more complex approach. These methods can be tested with a more complicated system. Another interesting future work is to develop a smart resource manager that is capable of making offloading decisions at runtime. Such a manager can be realized at various levels in the IoT infrastructure such as public cloud, private cloud and fog.

Acknowledgement

The work in this paper is supported by the Swedish Foundation for Strategic Research through the project Future factories in the Cloud (FiC), the Swedish Governmental Funding from Strategic Research Area through the project XPRES and the Swedish Knowledge Foundation through the project PreView.

References

[1] B. Sosinsky, Cloud Computing Bible, ISBN 978-0-470- 90356-8, Wiley Publishing, Inc., 2011.

[2] K. Ashton, "*That 'Internet of Things' Thing, in the real world things matter more than ideas*," http://www.rfidjournal.com/articles/view?4986, June 2009, [Online; Accessed 06-February-2015].

[3] IEEE Internet of Things, http://iot.ieee.org/about.html, accessed: January 2017.

[4] F. Bonomi, "Keynote talk: Connected vehicles, the internet of things, and fog computing," in *8th ACM International Workshop on VehiculAr Inter-NETworking (VANET)*, 2011.

[5] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "*Fog computing and its role in the internet of things*," pp. 13–16, 2012.

[6] H. Madsen, G. Albeanu, B. Burtschy, and F. L. Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable fog computing," in *20th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2013, pp. 43–46.

[7] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Australasian Telecommunication Networks and Applications Conference (ATNAC)*, 2014, pp. 117–122.

[8] R. Langmann and L. Meyer, "Automation services from the cloud," in *11th IEEE International Conference on Remote Engineering and Virtual Instrumentation (REV)*, 2014, pp. 256–261.

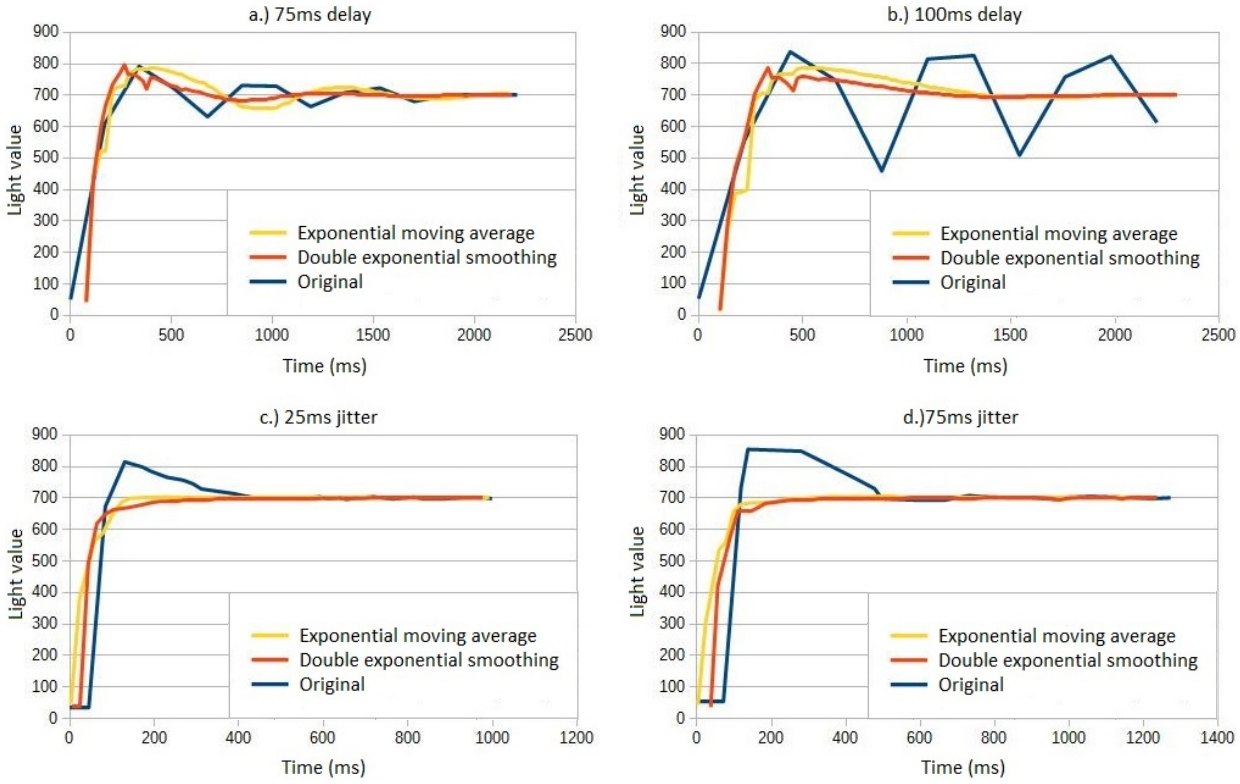


Fig. 12: System response using the prediction methods under delays (a and b) and jitter (c and d).

[9] O. Givehchi and J. Jasperneite, "*Industrial automation services as part of the Cloud: First experiences*," *Proceedings of the Jahreskolloquium Kommunikation in der Automation–KommA, Magdeburg*, 2013.

[10] H. P. Breivold and K. Sandström, "Internet of Things for Industrial Automation Challenges and Technical Solutions," in *8th IEEE International Conference on Internet of Things (iThings 2015)*, Dec. 2015.

[11] A. Serbanati, C. M. Medaglia, and U. B. Ceipidor, *Building blocks of the internet of things: State of the art and beyond*. INTECH Open Access Publisher, 2011.

[12] E. Borgia, The Internet of Things Vision: Key Features, Applications and open Issues, Journal of Computer Communications, 2014.

[13] Internet of Things From Research and Innovation to Market Deployment, Editors: Ovidiu Vermesan, Peter Friess, River Publishers Series in Communication.

[14] J.Gubbi, R.Buyya, S.Marusic, M.Palaniswami, Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions, Journal of Future Generation Computer Systems, 2013.

[15] Gartner, "Gartner says the internet of things installed base will grow to 26 billion units by 2020," December 2013, http://www.gartner.com/newsroom/id/2636073, accessed January 2017.

[16] H. P. Breivold, I. Crnkovic, I. Radosevic, and I. Balatinac, "Architecting for the Cloud: A Systematic Review," in *17th IEEE International Conference on Computational Science and Engineering (CSE2014)*, Dec. 2014.

[17] O. Givehchi, H. Trsek, and J. Jasperneite, "*Cloud computing for industrial automation systems - A comprehensive overview*," in *18th IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*, 2013, pp. 1–4.

[18] O. Givehchi and J. Jasperneite, Industrial Automation Services as part of the Cloud: First Experiences, in: Jahreskolloquium Kommunikation in der Automation - KommA (Communication in the Automation Conference), Magdeburg, 2013.

[19] B. Vogel-Heuser, G. Kegel, K. Bender, and K. Wucherer, "Global information architecture for industrial automation," *Journal of Automation Technology (Automatisierungstechnische Praxis)*, vol. 51, no. 1, pp. 108–115, 2009.

[20] H. Sequeira, P. J. Carreira, T. Goldschmidt, and P. Vorst, "*Energy Cloud: real-time cloud-native Energy Management System to monitor and analyze energy consumption in multiple industrial sites*," in *7th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, 2014.

[21] J. Delsing, F. Rosenqvist, O. Carlsson, A. W. Colombo, and T. Bangemann, "*Migration of industrial process control systems into service oriented architecture*," in *38th Annual IEEE Conference on Industrial Electronics Society (IECON)*. IEEE, 2012, pp. 5786–5792.

[22] Y. Chen, Z. Du, and M. García-Acosta, "Robot as a service in cloud computing," in *5th IEEE International Symposium on Service Oriented System Engineering*, ser. SOSE '10, 2010, pp. 151–158.

[23] O. Givehchi, J. Imtiaz, H. Trsek, and J. Jasperneite, "*Control-as-a-service from the cloud: A case study for using virtualized PLCs*," in *10th IEEE Workshop on Factory Communication Systems*, 2014.

[24] Cisco, Cisco Delivers Vision of Fog Computing to Accelerate Value from Billions of Connected Devices, Tech. Rep., January 2014, avaialble: https://newsroom.cisco.com/press-release-content?articleId=1334100.

[25] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Nov 2015, pp. 73–78.

[26] V. Gazis, A. Leonardi, K. Mathioudakis, K. Sasloglou, P. Kikiras, and R. Sudhaakar, "Components of fog computing in an industrial internet of things context," in *12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)*, June 2015, pp. 1–6.

[27] I. Stojmenovic and W. Sheng, "*The Fog computing paradigm: Scenarios and security issues*," in *Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2014, pp. 1–8.

[28] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwlder, and B. Koldehofe, "*Mobile fog: A programming model for large-scale applications on the internet of things*," in *Proceedings of the 2nd ACM SIGCOMM workshop on Mobile cloud computing*. ACM, 2013, pp. 15–20.

[29] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Mendez, C. E. Chung, Y. T. Wang, T. Mullen, and T. P. Jung, "Augmented brain computer interaction based on fog computing and linked data," in *International Conference on Intelligent Environments (IE)*, 2014, pp. 374–377.

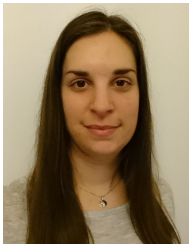
[30] H. J. Hong, P. H. Tsai, and C. H. Hsu, "Dynamic module deployment in a fog computing platform," in *18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Oct 2016, pp. 1–6.

[31] A. Toma and C. Jian-Jia, "Server resource reservations for computation offloading in real-time embedded systems," in *11th IEEE Symposium*

on Embedded Systems for Real-time Multimedia (ESTIMedia), 2013, pp. 31–39.
[32] W. Lingyun and A. Canedo, "Offloading industrial human-machine interaction tasks to mobile devices and the cloud," in *IEEE Conference on Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–4.
[33] Y. Nimmagadda, K. Kumar, L. Yung-Hsiang, and C. S. G. Lee, "Real-time moving object recognition and tracking using computation offloading," in *International Conference on Intelligent Robots and Systems (IROS), IEEE/RSJ*, 2010, pp. 2449–2455.
[34] A. Toma and J.-J. Chen, "*Computation offloading for real-time systems*," pp. 1650–1651, 2013.
[35] L. L. Ferreira, G. Silva, and L. M. Pinho, "Service offloading in adaptive real-time systems," in *16th IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*, 2011, pp. 1–6.
[36] Z. Yuan, L. Hao, J. Lei, and F. Xiaoming, "*To offload or not to offload: An efficient code partition algorithm for mobile cloud computing*," in *1st IEEE International Conference on Cloud Networking (CLOUDNET)*, 2012, pp. 80–86.
[37] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *Control Systems, IEEE*, vol. 21, no. 1, pp. 84–99, 2001.
[38] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proceesings of the IEEE*, vol. 95, no. 1, 2007.
[39] T. C. Yang, "Networked control system: a brief survey," *IEE Proceedings-Control Theory and Applications*, vol. 153, no. 4, 2006.
[40] K. J. Åström and T. Hägglund, *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society; Research Triangle Park, NC 27709, 2006.
[41] X.-L. Zhang and P. Liu, "A new delay jitter smoothing algorithm based on pareto distribution in cyber-physical systems," *Wireless Networks*, pp. 1–11, 2015.
[42] C.-S. Shih, S.-M. Wang, J. Chen, and Y.-H. Wang, "*Workload migration framework for streaming applications on smartphones*," in *20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2014, pp. 1–8.
[43] R. G. Brown, *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.
[44] J. Guerard, *Introduction to financial forecasting in investment analysis*. Springer Science & Business Media, 2013.
[45] P. Nikolaidis and A. Didic, "Real-time control in industrial IoT," Master's thesis, Mälardalen University, Sweden, June 2015.

Saad Mubeen Dr. Mubeen is a Senior Member of IEEE. He is a Senior Lecturer/Assistant Professor at the School of Innovation, Design and Engineering at Mälardalen University, Sweden. He has previously worked in the automotive industry, employed by Arcticus Systems AB, Sweden. He has also worked as a consultant for Volvo Construction Equipment, Sweden. He received his PhD in Computer Science and Engineering from Mälardalen University in 2014. His research interests include model-based development of vehicular embedded systems with a focus on timing models, end-to-end timing analysis and multicore platforms. Saad has co-authored over 90 research publications in international peer-reviewed journals, conferences, workshops and book chapters.

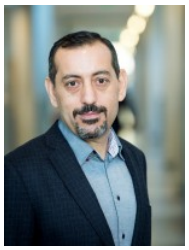
Pavlos Nikolaidis Pavlos has worked as a consultant for ABB Corporate Research, Sweden. He holds a Master degree in computer science from Mälardalen University, Sweden.



Alma Didic Alma has worked as a consultant for ABB Corporate Research, Sweden. She holds a Master degree in computer science from Mälardalen University, Sweden.

Hongyu Pei-Breivold Dr. Pei-Breivold is a Principal Scientist within the Industrial Internet-of-things group at ABB Corporate Research, Sweden. She obtained her PhD degree in Computer Science and Engineering from Mälardalen University in 2011. She is also an adjunct researcher at Mälardalen University. She has published more than 30 peer-reviewed articles in journals, conferences and workshops. She is active in academia as program committee member, track co-chair, and industry-research chair in international conferences. Her main research interests are software evolution, cloud computing, internet-of-things technologies and their applications in industry.

Kristian Sandströlm Dr. Sandström is a senior researcher at Research institutes of Sweden (RISE) SICS. His research interest is within industrial software systems, specifically in internet of things and cloud technologies for industrial systems and furthermore includes software architecture, design, analysis, and implementation of embedded real-time systems with high demands on reliability. Kristian received a PhD from Royal Institute of Technology, KTH, Stockholm, Sweden, in 2002, and holds a position as Adjunct Professor at Mälardalen University. He has held a position as a Senior Lecturer at Mälardalen university and he co-founded and worked at ZealCore Embedded Solutions. He has been subsequently employed as a manager at ENEA R&D and later as a Principal Scientist at ABB, Sweden.



Moris Behnam Dr. Behnam is a Senior Lecturer at the School of Innovation, Design and Engineering at Mälardalen University, Sweden. Currently, he is leading the Networked and Embedded Systems Division at Mälardalen University, Sweden. Moris received his PhD degree in Computer Science and Engineering from Mälardalen University in 2010. His main research interest is on resource virtualization for industrial distributed real-time systems. He has been working on virtualization techniques in both operating system level and communication level using resource reservation techniques. Moris has published over 175 publications in international journals/conferences/workshops. Moris has organized and chaired several international conferences and workshops including VTRES 2013, VTRES 2014 and WFCS 2016.