Evaluation of Burst Failure Robustness of Control Systems in the Fog

Nils Vreman
Department of Automatic Control, Lund University, Sweden
nils.vreman@control.lth.se

Claudio Mandrioli
Department of Automatic Control, Lund University, Sweden
claudio.mandrioli@control.lth.se

Abstract

This paper investigates the robustness of control systems when a controller is run in a Fog environment. Control systems in the Fog are introduced and a discussion regarding relevant faults is presented. A preliminary investigation of the robustness properties of a MinSeg case study is presented and commented. The discussion is then used to outline future lines of research.

2012 ACM Subject Classification Computer systems organization → Sensor networks

Keywords and phrases Networked Control Systems, Stability Analysis, Control over Internet, Fault Tolerance

Digital Object Identifier 10.4230/OASIcs.Fog-IoT.2020.8

Funding This work was supported by the ELLIIT Strategic Research Area; by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation; and by the ADMORPH project.
Nils Vreman: ELLIIT, ADMORF
Claudio Mandrioli: WASP

Acknowledgements The authors would like to thank their supervisors Anton Cervin and Martina Maggio for the interesting insights given during the writing of this paper.

1 Introduction

In recent years there has been a trend to move towards decentralized and distributed software infrastructure in industry [1]. This interest rises from the advantages of decoupling the control of the plant from the physical location and allowing for easier software update patches, lower maintenance costs, easier integration and optimization of individual components, among multiple others. This shift in paradigms is based on the emerging 5G-network with low latency, higher bandwidth, lower cost, and more reliable communication channels [4, 9]. However, neither the 5G-network nor the Fog that follows, are exempt from faults [5,12]. The wireless and distributed nature of the Fog introduces different faults from the ones addressed in classical control theory [13].

In real-time control systems, where reliability and timing constraints are of utmost importance, the faults need to be analyzed thoroughly [10]. Recently, [11] showed that it is possible for a control system to run in a Fog setting. The authors managed to control a plant in real-time whilst migrating a controller from the near vicinity of the plant to two datacenters, located at vastly different places. This result shows that it is possible to use the Fog as a platform for distributed control but at the same time poses questions on its general limitations and applicability. There is therefore need to develop methodologies for the study of reliability and robustness of a control system in the Fog environment.

2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020).
Editors: Anton Cervin and Yang Yang; Article No. 8; pp. 8:1–8:8
OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The general set-up, where controller and plant are connected through a network, has been studied under the name of networked control systems [7]. We propose an extension of the work on networked control systems based on types and patterns of faults, characteristic to the Fog environment. In this environment, faults can happen in the computation, actuation, or sensing nodes as well as in the transmission between these nodes. Faults may appear in the computation node due to overloads generated by other running tasks. Communication faults might instead appear due to signal disturbances or channel overloads created by other IoT-devices. In both cases the faults are likely to appear in **bursts**, meaning that the fault appears at an arbitrary point in time and persists for some time before disappearing.

Control systems seem to guarantee an intrinsic robustness toward these faults but its boundaries are yet to be explored. We perform a preliminary investigation of the topic by discussing the possible faults as well as simulating their possible outcomes. Namely, the main contributions of this paper are:

(i) a discussion on the specific faults introduced by a distributed Fog environment,
(ii) an investigation on the effect of bursts of faults on a control system,
(iii) a simulation based stability region for a particular case study.

The rest of the paper is structured as follows: Section 2 introduces control systems in the Fog and discusses the potential faults that can appear. Section 3 presents and discusses the case study of a MinSeg. Finally Section 4 summarizes the paper and outlines future research directions.

2 System Model

2.1 Control Systems

The purpose of a control system is to make a process behave according to some given requirements. Control systems are generally composed of a physical process and a controller. The controller is implemented as a software that receives measurements from the process, elaborates them, and decides accordingly how to steer the actuators of the process. Within this work we study the specific (but still common) case in which the controller implementation is split in two different components: a state observer and a control law. This is often needed due to the states of the system differing from what we can measure. A representation of these components and their interaction is shown in the block diagram of Figure 1.

For capturing the behaviour of the process, the most common class of models used are the so-called time-invariant *state-space models* [2]. State-space models take the form

$$\begin{aligned} \dot{x} &= f(x, u), \\ y &= g(x, u) \end{aligned} \tag{1}$$

where u is a vector of actuation variables decided by the control law (also called the *inputs* of the process), y is a vector of measured variables (also called the *outputs* of the process), and x is a vector of *states* of the process. Therefore, the first equation describes how the system evolves given the current state and input, whilst the second equation instead describes how the measurements are connected to the actual state of the system.

The purpose of the observer is to estimate x given the known input u and measured output y. The most common state observer is called the Luenberger observer [2]. This observer is based on running a simulation of the process according to Equation (1) and correcting it given the discrepancy between the expected measurement from the simulation and the true measurement. In this way, the information contained in the model is merged with the measurements of the process. By calling the estimated state $\hat{x}$ and expected output $\hat{y}$ the Luenberger observer is implemented with the following equation:

$$\begin{aligned} \hat{y} =& g(\hat{x}, u), \\ \dot{\hat{x}} =& f(\hat{x}, u) + K(y - \hat{y}) \end{aligned} \tag{2}$$

where the constant K (also called the *observer gain*) can be chosen using different techniques (e.g. pole placement or Kalman filtering) [2]. Intuitively a high value of K represents that the observer will rely more on the measurements and a low value will make the observer rely more on the simulated model of the process.

The estimated state $\hat{x}$ is then used by the controller, together with the desired behaviour r to compute the control action u. The most common class of controllers are linear controllers which can be written as

$$u = L\left(r - \hat{x}\right), \tag{3}$$

where L is called the *control gain* and can be designed using standard techniques from control theory – e.g. pole-placement or LQG control [2].

The mentioned design methodologies provide formal guarantees on the stability and performance of the control system. Stability is the most fundamental property required in a control system and states that the system variables x, $\hat{x}$, u, and y over time will not diverge but instead converge to some finite value. Intuitively these guarantees depend on the accuracy of the available model. In a control system, the capability of satisfying the requirements in presence of modelling errors, disturbances, and component faults is called **robustness**. In this work we analyze the robustness to the Fog faults in terms of stability guarantees. Specifically, in the Fog context, we want to evaluate how tolerant a control system is to faults.

2.2 Control Systems in the Fog

In traditional control systems, the control software is executed by an embedded device that is attached to the physical process. Leveraging the emerging Fog network, it becomes natural to move the controller into the Fog. This allows the control system to access higher computational power, ease software updates, synchronize with other IoT-devices as well as alter the hardware during runtime.

When the controller is run in the Fog, it interacts with the plant through wireless channels, whilst sharing the computational power with other processes. Therefore, both the communication channel and the computation platform are subject to the interference of other IoT-devices. These phenomena introduce new and specific disturbances different from the ones seen in traditional control systems. Evaluating the performance of control systems in their presence is critical for the safe and optimal implementation of a control system in the Fog.

In Figure 1, where the block diagram of the control system is shown, the dashed boxes highlight that the controller (both state observer and control law) is executed in the Fog and that the process is placed in a different physical location. This shows that the desired behaviour r, the control actuation u, and the measurement y are the signals traveling on wireless communication channels.[1]

[1] Different set-ups could be considered, for example the state observer and the control law could be run in different IoT nodes. In this work we limit our study to this case being it the closest to the traditional set-up.

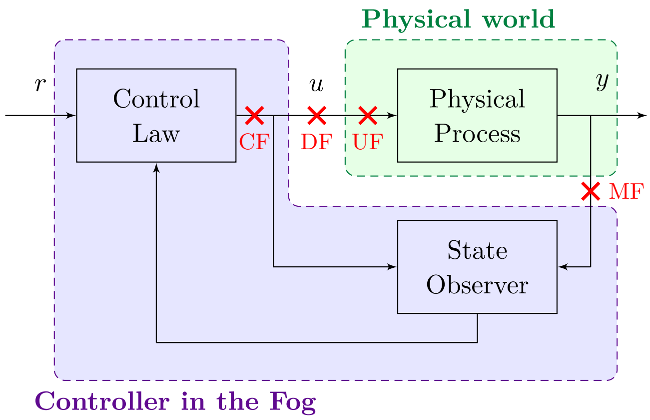

Figure 1 A control system in the Fog context. The purple dashed block illustrates all the components located in the Fog, whilst the green dashed block represents the physical process we are trying to control. Transmission channels are represented by black arrows and red crosses are used to indicate a transmission fault.

2.3 Fog Faults and Control Systems

Given the Fog-IoT set-up described in the previous section, faults could be introduced when communicating the signals r, u, and y as well as when executing the controller. Among these we exclude faults in the communication of r, since this signal is set by an external operator and not affected by the control system.

In this work we discuss the following scenarios (shown in Figure 1 as red-crosses in the control system block diagram):

- **Computation Faults** [**CF**]: the IoT computation node does not manage to execute the controller within the real-time constraints of the control system. Therefore, the actuator does not receive any actuation command and the controller does not update. This could happen, for example, if the node was suddenly overloaded by other computations.
- **Detected Actuation Transmission Faults** [**DF**]: the communication channel fails in transmitting the actuation signal u but the control software detects it and can take counteraction. The actuator does not receive any command and the controller updates accordingly. This could for example happen if the communication channel is kept busy by other IoT devices.
- **Undetected Actuation Transmission Faults** [**UF**]: the communication channel fails in transmitting the actuation signal u and the control software is not aware of this and can therefore not take any counteraction. The actuator does not receive any command and the controller updates normally. This could happen, for example, if the controller get access to the communication channel but the channel itself does not succeed in transmitting the signal.
- **Measurement Transmission Faults** [**MF**]: the communication channel fails in transmitting the measurement signal y. The control software does not receive it and can therefore take counteraction. This could for example happen if the communication channel is either busy or simply fails to transmit the signal.

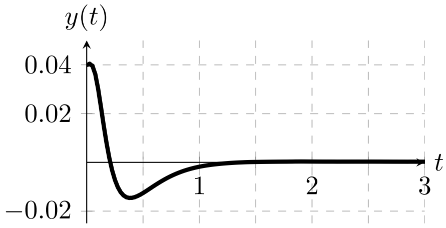

Figure 2 Time series of the MinSeg leaning angle under regular conditions.

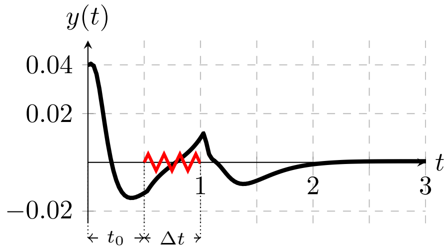

Figure 3 Time series of the MinSeg leaning angle in the presence of a burst of computation faults starting at $t_0 = 0.5$ with a duration of $\Delta t = 0.5$.

2.4 Fault Patterns

In this work we consider fault patterns of the **burst** type, meaning that a given fault happens sporadically and continuously for a relatively short time. The reason for considering this type of faults is that Fog-IoT components are expected to incur temporary overload periods that will result in temporary unavailability.

We define burst faults as a sequence of consecutive component faults that happens at an arbitrary point in time and does not repeat. Therefore, a burst fault of a given type is defined by two quantities: an initial time t_0 and a duration Δt. A graphical representation of a burst fault and the corresponding process is shown in Figure 3, as opposed to the system response in regular conditions shown in Figure 2. In Figure 3, the system experiences a burst of computation faults at time $t_0 = 0.5$ with length $\Delta t = 0.5$. The red zig-zag line (in the x-axis) shows the interval in which the fault occurs and the plot shows the time series of the leaning angle of the MinSeg.

When the actuator does not receive an updated control signal, we implemented a *zeroing* strategy [8]. This means that the variable u is set to zero in the presence of computation faults (CF), detected (DF), and undetected actuation transmission faults (UF). We chose a zeroing strategy over the alternative of re-applying the previous control signal. In fact, in many control systems, after the computation of the control signal, additional steps are performed, for example to transform the control signal between different coordinate spaces (e.g., dynamic coordinate systems). The presence of faults would render this additional computation (and holding the previously computed signal) infeasible, therefore justifying our choice.

3 Results and Discussion

We will in this section evaluate a model of a real control system through simulations. From the simulation results, we discuss the system robustness to the transmission disturbances presented in Section 2. The real-world process that we chose to analyze is a MinSeg [6] controlled via bluetooth technology. Due to the fact that the MinSeg is inherently unstable, with fast dynamics, it is a relevant process for our experiments. The performance degradation of the control system is therefore clearly exposed when computation and transmission errors are introduced.

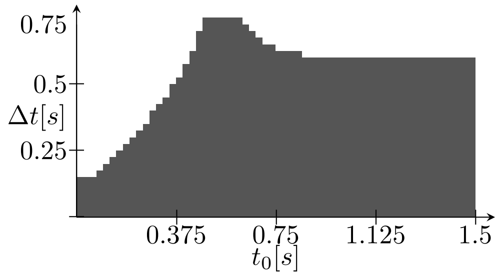

Figure 4 Stability region for simulations using computation faults [CF].

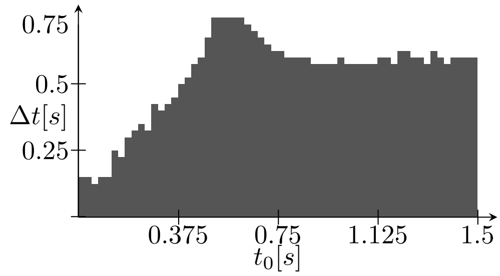

Figure 5 Stability region for simulations using detected actuation transmission faults [DF].

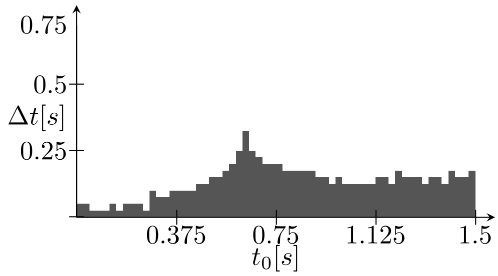

Figure 6 Stability region for simulations using undetected actuation transmission faults [UF].

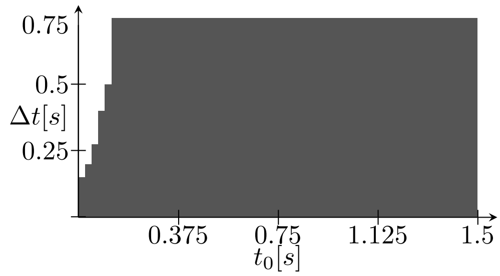

Figure 7 Stability region for simulations using measurement transmission faults [MF].

We ran the simulations using Matlab[2]. In particular we simulated the timing interactions between the network components and the control system using the research tool called *TrueTime* [3][3]. TrueTime is a Matlab multi-purpose toolbox primarily used for analyzing the complex timing properties of real-time, networked control systems.

We injected the burst disturbances (as described in Section 2.4) into the control system during a transient. A transient is when the process moves from an arbitrary state to an equilibrium. An equilibrium is a state from which the system will not diverge unless in presence of some external input or disturbance. This is a standard, however not exhaustive, way to evaluate the performance of a control system.

We ran simulations for each type of fault with one burst each. We considered bursts of different length Δt, starting at different points t_0 of the transient, as discussed in Section 2.4. The stability results from the combinations of t_0 and Δt values are then plotted together. We define a simulation stable when the angle of the MinSeg does not exceed a given threshold α for the entire duration of the simulation. Consequently, when the threshold is exceeded, the simulation is marked as unstable. In fact, exceeding this threshold implies that the angle of the process is too large for the control system to be able to bring the MinSeg back to an upright position. Figures 4, 5, 6, and 7 show the stability regions for each of the faults defined in Section 2.3 given different configurations of Δt and t_0. Each point (x, y) in the plot represents a simulation using a specific $(\Delta t, t_0)$ combination. Unstable simulations are marked with white and stable simulations are marked with gray.

---

[2] `https://se.mathworks.com/products/matlab.html`
[3] `http://www.control.lth.se/research/tools-and-software/truetime/`

As we can see in Figure 2, for values of $t_0 < 0.75s$, the burst happens while the system is still in a transient state. Conversely, for values of $t_0 > 0.75s$ the system has reached its equilibrium, and the simulations are not changing anymore. For all the faults, this provides the intuition that in an equilibrium state the robustness to burst for a control system could be quantified. Still, there are differences between the specific faults.

Figure 5 shows the results of the simulations in presence of detectable actuation transmission faults. The behaviour of the system appears similar to the case of computation faults (Figure 4). Intuitively, in both cases, the system is not actuated. However, the difference between the two faults is the fact that the state observer is not updated during the computation faults burst. According to the simulations, the difference does not affect the robustness of the system. In the future we plan to investigate the generality of this finding.

Figure 6 shows the results of the simulations in presence of undetectable actuation transmission faults. The system shows less robustness with respect to the previous two cases. This can be attributed to the state observer being updated with the wrong control signal and therefore diverging from the actual state of the system. Despite this, the general behaviour of the stability region shows a similar trend to the two faults considered previously. In general, from the comparison of Figure 5 and 6, undetectable actuation transmission faults can be more harmful than detectable ones. This should be taken into account when the Fog infrastructure is implemented, e.g. by implementing detection algorithms for transmission faults.

Figure 7 shows the results of the simulations in presence of measurement transmission fault. Most of the simulations expose a stable behaviour. This is due to the perfect coherence between the model used in the observer and the model used to simulate the process. Since the two models are the same, despite the lack of feedback, the estimated states do not diverge from the actual states of the process. This will not be true for a real implementation of the system due to modeling errors and disturbances. The unstable simulations, that appear for small values of t_0, are due to the observer not having enough time to start tracking the states of the process.

4 Conclusions

This paper presents preliminary work investigating the robustness to computation and communication faults of control systems in the Fog. The simulations show that a control system has an intrinsic robustness to the faults characteristic of this environment. Further investigations should be based on a formal analysis of the system properties. The relevance of the state observer, in the presence of the discussed faults, has been emphasized through simulations. In future work we plan to evaluate solutions that handle burst faults.

References

1 M. Aazam, S. Zeadally, and K. A. Harras. Deploying fog computing in industrial internet of things and industry 4.0. *IEEE Transactions on Industrial Informatics*, 14(10):4674–4682, October 2018. doi:10.1109/TII.2018.2855198.

2 Karl Johan Åström and Richard M. Murray. *Feedback Systems : An Introduction for Scientists and Engineers*. Princeton University Press, 2008.

3 A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. E. Årzén. How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime. *IEEE Control Systems Magazine*, 23(3):16–30, 2003. doi:10.1109/MCS.2003.1200240.

4 Tommaso Cucinotta, Mauro Marinoni, Alessandra Melani, Andrea Parri, and Carlo Vitucci. Temporal isolation among lte/5g network functions by real-time scheduling. In *Proceedings of the 7th International Conference on Cloud Computing and Services Science*, CLOSER 2017, page 368–375, Setubal, PRT, 2017. SCITEPRESS - Science and Technology Publications, Lda. doi:10.5220/0006246703680375.

5 F. Foukalas, P. Pop, F. Theoleyre, C. A. Boano, and C. Buratti. Dependable wireless industrial iot networks: Recent advances and open challenges. In *2019 IEEE European Test Symposium (ETS)*, pages 1–10, May 2019. doi:10.1109/ETS.2019.8791551.

6 B. Howard and L. Bushnell. Enhancing linear system theory curriculum with an inverted pendulum robot. In *2015 American Control Conference (ACC)*, pages 2185–2192, July 2015. doi:10.1109/ACC.2015.7171057.

7 Dimitrios Hristu-Varsakelis, William S. Levine, R. Alur, K.-E. Arzen, John Baillieul, and T. A. Henzinger. *Handbook of Networked and Embedded Control Systems (Control Engineering)*. Birkhauser, 2005.

8 S. Linsenmayer and F. Allgower. Stabilization of networked control systems with weakly hard real-time dropout description. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 4765–4770, December 2017. doi:10.1109/CDC.2017.8264364.

9 Daniel Jun Xian Ng, Arvind Easwaran, and Sidharta Andalam. Contract-based hierarchical resilience framework for cyber-physical systems: Demo abstract. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '19, page 324–325, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3302509.3313323.

10 M. Pohjola, S. Nethi, and R. Jantti. Wireless control of mobile robot squad with link failure. In *2008 6th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops*, pages 648–656, April 2008. doi:10.1109/WIOPT.2008.4586154.

11 Per Skarin, Karl-Erik Årzén, Johan Eker, and Maria Kihl. Cloud-assisted model predictive control. In *2019 IEEE International Conference on Edge Computing*, pages 110–112. IEEE - Institute of Electrical and Electronics Engineers Inc., August 2019. doi:10.1109/EDGE.2019.00033.

12 W. Wang, D. Mosse, and A. V. Papadopoulos. Packet priority assignment for wireless control systems of multiple physical systems. In *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, pages 143–150, May 2019. doi:10.1109/ISORC.2019.00036.

13 L. Zhang, L. Zhao, and L. Li. Sliding mode control for network control systems with packets dropout. In *2015 34th Chinese Control Conference (CCC)*, pages 6592–6596, July 2015. doi:10.1109/ChiCC.2015.7260677.