NETWORKED CONTROL SYSTEM OVER AN EPICS BASED ENVIRONMENT

M. Eguiraun[1*], J. Jugo[1,2†], I. Badillo[2‡], I. Arredondo[1,2§]
[1]ESS Bilbao, Spain, [2] Univ. of Basque Country, Spain

Abstract

The use of distributed control systems can improve the overall control system's performance in aspects as the increment of computational power, robustness and load balance. Thus, the importance of developing control systems across a networked environment is rising, leading to the appearance of new schemes.

Moreover, the interest on TCP based networks in industrial environments has been increasing due to its advantages in cost and easy integration. However, this protocol has non-deterministic characteristics, which make difficult its use for networked control systems. The use of EPICS can be an approach for minimizing this behaviour, due to its soft real time capabilities. In fact, a lot of research effort is focused on developing middleware based solutions.

This work presents a networked control scheme where control loop is closed under periodic sampling over the net, managed by an EPICS control system. As opposed to usual way of working with EPICS, where an IOC implements desired control algorithm, two different strategies are compared. In the first one, two IOCs are used; one performs data acquisition, while the second one calculates the control signal. In the second strategy, a CA client connected to a IOC for data acquisition closes the loop, being the control signal calculation performed in the client.

Similar timing results are achieved with both experiments, but, the CA client based one, enables more accuracy in the scan period and more versatility in the design of the controllers. These characteristics make the second option a good alternative in control field.

INTRODUCTION

In large scientific facilities, the distributed nature of the diverse elements involved leads to monitoring and control solutions which must be distributed too. From the control point of view, allocating control tasks among several machines can carry out jobs which cannot be achieved with a single control device. But, on the contrary, this fact increases complexity and programming time, due to the necessity of communicating different elements and introducing new paradigms especially related to stability issues.

However, the use of distributed control systems can improve the overall system's performance in aspects such as the increase of computational power, robustness and load balance. Thus, the importance of developing control systems across a networked environment (NCS) is rising and an important research effort must be focused to solve problems related to NCS, as for example the effect of communication delay and losses over closed-loop stability using different control schemes, [1].

On the other hand, in complex and distributed environments, where important amount of data must be processed and different control devices must be integrated, the use of a middleware based approach is very advisable. For instance, large industrial and scientific facilities use middleware based control mechanisms, [2, 3]. In particular, the Experimental Physics and Industrial Control System (EPICS) set of software tools and applications is very extended around the world, being its communication approach based on standard ethernet TCP/IP connections.

In this paper, a networked closed loop control scheme based on EPICS is analyzed. The main goal is to study the advantages and drawbacks of using standard EPICS solutions for such purpose, comparing two different schemes. In the first scheme, two Input Output Controllers (IOCs) are used, one performing the data acquisition, while the second one calculates the control signal. In the second strategy, a Channel Access (CA) client connected to a IOC for data acquisition closes the loop, being the control signal calculation performed on the client.

The present work is structured as follows: firstly the laboratory testbed, where it is implemented the networked control, is presented. After that, results for each EPICS based NCS control scheme are summarized in Section . Finally, some conclusions and future work are explained.

EXPERIMENTAL SETUP

Two different approaches have been implemented in present work. Both are plotted in Figure 1. The first scheme is a pure EPICS based solution, where the NCS is implemented by mean of two IOC servers, located in *host1* and *host 3*. One IOC server acts as sensor (*epics:daqai*) and actuator (*epics:daqao*) device, performing data acquisition and applying control signal. It emulates analog input and analog output records, while loop period is set by a dummy record (*epics:calc*). The second IOC (*epics:algor*) calculates the control signal, and writes into first IOC's field, using usual CA methodology. The communication link is a standard Ethernet based network, deployed specifically for these kind of experiments. The control algorithm consists on a simple counter; since the main goal of this work is

* meguiraun@essbilbao.com
† josu.jugo@ehu.es
‡ inari.badillo@ehu.es
§ iarredondo@essbilbao.com

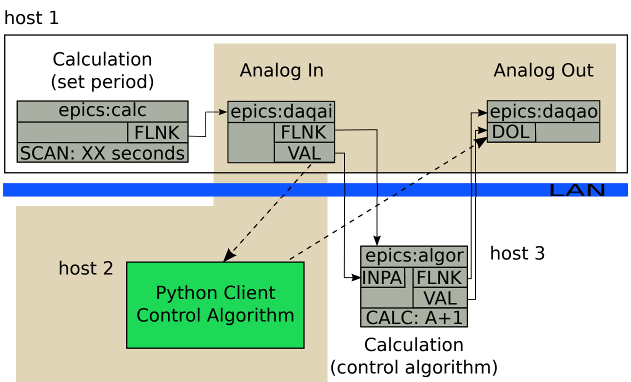

Figure 1: Experimental setup overview.

to analyze timing constraints, a basic calculation has been set. The second approach replaces the control algorithm IOC by a python program which implements a CA client (see *host 2* in Figure 1). In this case, Epics CA for Python module provides an interface to Epics CA [4].

In this scenario, two different kind of tests has been performed. The first one consists on a set of three records in *host 1*, while in the second one the number of record is multiplied by a factor of 400.

In both cases, a single record set is used to perform the calculation of the simulated control signal. The rest of the records are placed with the aim of disturbing the network communication and increasing the system load.

RESULTS

The following section presents the main results obtained with the aforementioned experiments. These are preliminar results to indicate the tendency of the measurements, but longer tests with computer's effort monitoring have to be performed, in order to obtain more precise conclusions.

Pure Epics Scheme

Figures 2 and 3 show the case where pure EPICS is used. Considering that period's values are not very low, network environment should deal with them without important problems (except the typical ones in TCP based non deterministic networks). However, analyzing the results for each period, differences can be observed, specially in lower periods. A wider histogram means a higher jitter, defined as the difference between obtained period and its desired value. In conclusion, when lower periods are used, higher jitters appear. This result can be previewed since the system load is increased in such situation.

The main conclusion is that EPICS behaves well enough in such circumstances, since extra network load does not add dramatic changes in its timing performance. This clearly shows that its soft real time capabilities can be used to close the control loop over the net, although it has a non-deterministic nature, under several conditions.

Tables 1 and 2 summarize the main timing parameters for studied sampling times.

Table 1: EPICS single PV, all values in milliseconds.

Period	*Mean*	*Std. deviation*	*Maximum*
100	100,16	0,73	152,90
200	200,14	0,44	207,65
500	500,13	0,84	507,74
1000	1000,1	1,27	1003,00

Table 2: EPICS multiple PVs (in milliseconds).

Period	*Mean*	*Std. deviation*	*Maximum*
100	100,15	0,30	116,97
200	200,21	5,51	281,12
500	500,16	0,23	503,23
1000	1000,17	0,25	1000,44

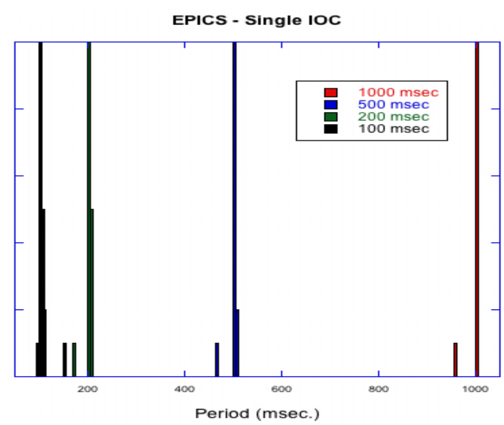

Figure 2: EPICS against one PV.

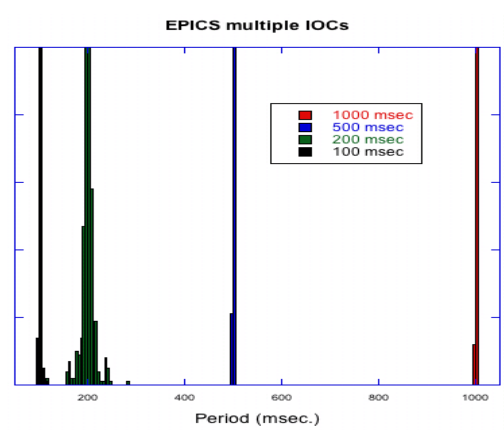

Figure 3: EPICS against multiple PVs.

Python CA Client Based Scheme

In this case, the NCS system can be implemented with more flexibility, admitting a wide range of periods. Unlike in standard EPICS, which have the minimum scan period at 100ms, with Python it is possible to work with smaller time constraints. So, in order to study the limits introduced by the sampling period, lower periods are set. As expected, Figures 4 and 5 show higher jitters for lower periods. In addition, similar behaviour is obtained when disturbing PVs add an overload on the network.

If both Pure EPICS and Python CA Client experiments are compared, there is little difference when focusing on similar periods. This fact proves that a python client is a good alternative for implementing a networked control strategy over an EPICS based net.

Tables 3 and 4 give detailed information.

Table 3: Python and single PV (in milliseconds).

Period	*Mean*	*Std. deviation*	*Maximum*
10	9.99	1,38	59,07
25	24,99	2,41	79,66
50	49,99	1,84	105,91
100	99,99	2,16	136,31
150	149,98	4,54	197,10
250	249,98	1,25	274,64
500	499,99	2,10	527,39
1000	999,98	1,09	1015,18

Table 4: Python with disturbing PVs (in milliseconds).

Period	*Mean*	*Std. deviation*	*Maximum*
10	9.99	1.49	69.46
25	25.00	2.74	102.55
50	49,99	1.47	111.81
100	99,99	1.92	151.62
150	149,99	1.53	206.36
200	200.00	2.35	239.17
250	250.00	2.29	307.55
500	500.00	1.61	515.83
1000	1000.00	2.96	1020.00

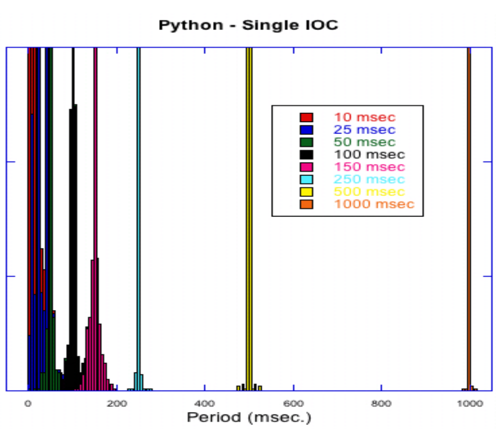

Figure 4: Python and single PV.

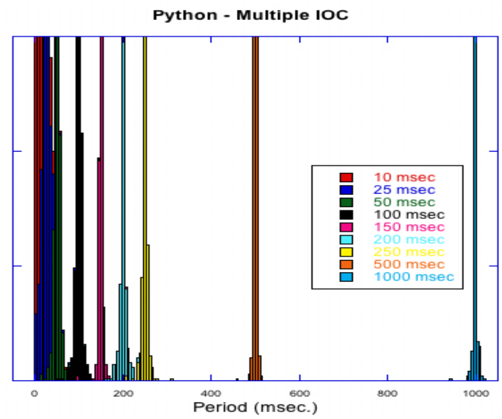

Figure 5: Python with disturbing PVs.

CONCLUSION AND FUTURE WORK

Presented results show that EPICS acts as soft real-time system, with the limitations derived from a non-deterministic network connection. Moreover, reasonable timing characteristics have been observed in both schemes. Those results corroborate the possibility of developing NCS systems using EPICS under several conditions, for a large variety of feedback control systems, e.g. systems where hard real-time is not really needed.

When Pure EPICS and Python CA Client schemes are compared, it is observed that the timing results are similar, but being more flexible the scheme based in Python CA client, in period time definition and in suitability for control algorithms. This approach allows the use of richness of the python programming language, which enables, for instance, developing interfaces with hardware devices. Therefore, this procedure can be used to deploy hardware drivers within EPICS from python, calling C code directly. This leads to a faster deployment and previous knowledge can be reused.

Finally, although the results are relatively satisfactory, the proposed scheme can be improved in several ways. First, the IOC servers have been implemented in Linux systems, being the use of a real time operating system (vx-works, real-time Linux, ...), a viable enhancement way.

Another possible improvement comes from the use of fine-tuned timing for the Ethernet network. Here, the Network Time Protocol and the Precision Time Protocol IEEE 1588 are interesting tools.

REFERENCES

[1] Y. Tipsuwan and M. Chow, *"Control methodologies in networked control systems"*, Control Engineering Practice 11, 2003, 1099-1111.

[2] EPICS home page, http://www.aps.anl.gov/epics/

[3] TANGO home page, http://www.tango-controls.org/

[4] Epics Channel Access for Python, http://cars9.uchicago.edu/~newville/Epics/Python/