# A Loosely Coupled Architecture for Networked Control Systems

Milton Armando Cunguara,Tomás António Mendes Oliveira e Silva
IEETA / University of Aveiro
Aveiro, Portugal
{milton.cunguara,tos}@ua.pt

Paulo Bacelar Reis Pedreiras
IT / University of Aveiro
Aveiro, Portugal
pbrp@ua.pt

***Abstract*—Nowadays the use of so-called Network Control Systems is a common practice. Although convenient, networks introduce several perturbations to the control system, due to the additional latency, jitter and even errors that arise when messages fail to be delivered. This paper presents a control architecture that attenuates the impact of the network on the controller performance. The proposed architecture is based on a series of data buffering and estimation, decoupling the controller execution from the instants in which messages are transmitted. In the sequel, important bandwidth are also attained, due to the batch transmission of sensor and control data.**

## I. Introduction

In recent years networked systems and related electronic systems have evolved, thus becoming somewhat more suitable for control purposes [1] [2]. However, some issues remain open, such as the scheduling of networks traffic — periods (which are application dependent), phases (which may be interdependent) and other schedule related issues — and inherent network errors. Consequently, messages may be delayed by the network for large amounts of time, eventually failing deadlines, or may even never arrive, in case of errors.

From the control standpoint it would be desirable if all such issues could be masked and that an image of the system could be presented without the peculiarities of the network. A vast body of work already exists that addresses this problem, as can be seen in, for example, [3] and references therein. The most promising approach appears to be a control-aware network design and a network-aware controller.

This paper proposes an architecture that, to some extent, decouples the network data exchanges from the control action. To achieve this goal, the controller integrates an estimator that computes future states and actuation values. The unavoidable imperfections of the system model and the perturbations that always exist in real systems (e.g. noise, quantification errors) naturally cause such estimations to differ from the real system state. To keep the estimations quality within certain limits, messages are sent occasionally to feed the estimator with real process data. One important fact to retain here is that the controller becomes loosely coupled to the reception of messages, since its computations are based in the estimator output. Additionally, several sensor values, as well as actuation values, can be grouped within the respective nodes and sent inside a single message, which permits obtaining important benefits in terms of bandwidth, since the data efficiency (i.e. ratio between the useful data and the total message length) is improved.

This work is partially funded by the EU program FEDER through "Programa Operacional Factores de Competitividade COMPETE" and by the Portuguese Government through"FCT Fundação para a Ciênciae a Tecnologia" in the scope of project FCOMP-01-0124-FEDER-007220.

## II. Related Work

This work encompasses many aspects that can be found in the control/network literature, namely:

- The ability to change the control rate at run time. One such approach was presented in [4], which suggested the design of several controllers, each one for a different sampling rate, and the use of a selection mechanism responsible to switch the controller (and corresponding sampling rate) when certain error criteria were met (or not met). In [5] controller rate adaptation was also discussed, but with focus on CPU slack and other centralized resources optimization. A different approach is presented in [6]. There it is proposed that the rate should be proportional to the quadratic error and to a constant given to each process, so that more important processes still somehow have a higher priority. A similar approach had been proposed before (see [7] and in references therein). These approaches cause slight oscillations. This is so because the controller process is given a short period when it has large errors, which gives rise to a small error. This in turn triggers the switch to a slower rate controller, which gives rise to a larger error. This problem does not occur in systems with properly designed static schedulers, that uses a schedule that minimize the expected error over a large run-time. In fact, in order to avoid such oscillations, dynamic schedulers must change based on a low-pass filtered version of the error, not in the instantaneous value.
- The use of buffers for improving state estimation in lossy networks. In [8], it is shown that a buffer in the sampler can be used to improve state estimation. In the same paper, the probability of the estimation error covariance matrix being smaller than a given matrix is explored in detail.

  The simpler systems deal with missing actuation values by holding their outputs in the absence of new control messages. Systems in which the controller can somehow

compute a new value, whenever a controller message is not present, represent a logical evolution.

From the perspective of controller estimation, a scheme to compensate for networks delays was presented in [9]. There it is suggested that the controller should predict the state of the system at the control instant and then apply a control law based in the predicted state. A similar approach was studied in [10], but with the prediction of a future (network delayed) state and the application of the control law. Less sophisticated approaches include the use of basic interpolators to achieve the same goal.

Another approach to control that relates to the approach presented in this paper is the Send-On-Delta (SOD) [11], [12], in which data is sent if either a significant change has occurred or a given amount of time has passed since the last message was sent (*I am alive* message). A rather evolved version of SOD is presented in [13], that is state-space based, in which the sampler runs a version of the estimator running in the controller and an estimator based in all samples. A significant change, in that context, refers to the difference between the estimated spaces. Another publication pertaining to this article is [14]. There it is argued that computation can be exchanged for bandwidth, in the sense that the number of messages exchanged in the network can be reduced if more complex control techniques are used. An example of a system with an estimator/controller capable of such trade-off is provided.

## III. Architecture Rationale

It has been shown that networks can be made somewhat more resilient to external perturbations provided that a suitable error correction mechanism is in place. In fact, Shannon [15] proved that the maximum error-free transmission bandwidth is a function of the medium bandwidth and the signal-to-noise ratio.

The scheme presented here tries to achieve a bandwidth-noise tradeoff. For example, in a system with no perturbations, once model, system state at a given time instant and future input signal(s) are known, the controller could possibly compute all future states and control actions in advance. Therefore, only one message (potentially long but of finite duration) would need to be sent, rendering the transmission rate close to zero.

As control perturbations increase, the transmission rate also has to increase to guarantee a small output error. It should be stressed that what is being increased is the message transmission rate, not the control execution rate. The former is determined based on the conditions of the output (error) and the availability of bandwidth in the bus, while the later is determined by the dynamic of the system, using some empiric values [16] [17] [18]. This low-level of coupling between the network transactions and the controller execution is extremely positive from the system design point of view, since turns both subsystems quasi-independent. In fact, in this architecture, the only requirement of the controller is that the networking subsystem should be able to feed data to bound the estimator error, not being relevant the particular instant in time in which the messages are sent and received.

From a network perspective, it is a well known fact that the number of nodes and the bandwidth requirements of each node, have been increasing steadily in the past decades. The network bandwidth may eventually become insufficient. One immediate and obvious solution is increasing the bandwidth of the network. However, such approach is, in many cases, undesirable since has enormous implications. For instance, Controller Area Network (CAN) [19] is still very used in many domains (e.g. automotive, automation). CAN bandwidth is fundamentally limited to 1Mbps (or less, depending on the network length). Thus, moving to higher speeds requires abandoning this protocol, which implies huge costs (training, development tools, migrating existing systems, ...).

A less obvious, but more intelligent, approach is to improve the efficiency of the network bandwidth usage. One method to achieve this goal is to take advantage of the bandwidth vs. computation trade-off. The scheme presented here has the potential to do so efficiently, since the estimator, which requires additional processing, permit the reduction of network load, by allowing several sensor samples and/or actuation values to be sent into a single message, therefore reducing the communication overhead. This scheme can be used either to increase the number of nodes and control loops that can be supported in a given network or, alternatively, may be used to increase the effective bandwidth available to a given control loop or set of control loops.

## IV. Description of the Proposed Architecture

The control systems presented here have the usual distributed control structure, i.e., sampler, controller and actuator nodes, see Fig. 1. As the name states, sampler nodes sample a number of physical variables and send the values to the controller. The controller receives a number of sampled variables and then computes a series of control outputs, that are subsequently sent to the actuators. Actuators receive messages containing a desired value for physical variables and ensure (sometimes not completely effectively - e.g. due to output noise) that they actually have such values. Figure 1 depicts a control system with the proposed architecture.

### A. Sampler

The sampler is a simple system, akin to the one proposed in [8] (though there is a big semantic difference associated with the values stored in the buffer). The sampler stores in a circular buffer the last $N$ sampled values. The buffer is obviously updated whenever there is new data. When this

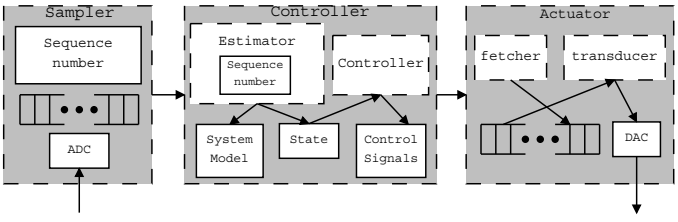


Fig. 1. Architecture Diagram

happens the entire buffer content plus a sequence number are sent to the bus. The sequence number tells the receivers (controllers) if any messages were lost and, if so, how many. If less than $N$ consecutive messages were lost, then no information is lost — though part of it is received late.

The presence of an estimator in the controller node allows messages to be sent in batches. Since all communications imply overhead, such reduction in the number of communications leads to bandwidth savings. Depending on the particular network protocol, this strategy will produce slightly different results. For instance, in CAN, the maximum payload is 8 bytes, thus the length of the packing is limited. However, the data efficiency of CAN is extremely low and thus packing even a few bytes has a significant impact on the aggregated efficiency. Ethernet exhibits a different behavior. The minimum payload is 46 data bytes and thus, when the sensors produce small amounts of data (e.g. 12 bit ADC), several samples may be packed without requiring any excess bandwidth with respect to sending a single value. Additionally, the packaging capacity is very long, since an Ethernet frame can carry up to 1500 data bytes.

### B. Controller

Many application domains are very cost-sensitive, thus the processing power tends to be as low as possible. Hence, the controller is most likely the only node with a considerable processing power. For this reason, whenever possible, the *heavy lifting* is done by controllers.

The controllers proposed in this paper are composed of two subunits or tasks, that are executed with a certain degree of independence.

1) Whenever a message is not received, the estimation is made based on the previous estimate and the previous control signal. When a new message is received the state from the instant that the last previous message was received up to the present is recomputed, based on the actually sampled values, through a process explained in the next two paragraphs. Thus, estimates built in the absence of samples are discarded and in the long run all estimates are made based on actual samples. This has two advantages: i) at any time there is the best possible estimation (even when the best estimation is not based on a sample) and ii) the covariance matrix is certainly bounded and from time to times equal to the lowest possible, i.e. the covariance matrix of the case in which all samples were received.
   The first part of the controller interacts with the samplers. The controller has a sequence number of its own, that represents the last sensor message that was successfully received. Whenever a new message is received, its sequence number is compared to the sequence number of the controller. If it is the next number in sequence then no message has been lost. If the difference is $n$, $n > 1$, then the total number of lost messages is $n - 1$. After receiving a message, the controller sets its sequence number to the last received message.
   Knowing the number of lost messages, the controller extracts this same number of past input values from the current message (buffer), plus the current sample. This approach allows the controller, with the information of its previously computed control signals, to reconstruct the state trajectory of the system (or any other equivalent estimation mechanism) up to the present. It should be remarked, (again), that in doing so the estimates computed without the sampled values are disregarded.
   To summarize, this task builds an estimate of the system state and, if necessary, a system identification is performed. This task is executed whenever there is a new message from a sensor.
2) The other task running in the controller is responsible for computing control values and sending them to the respective actuators. To this end, the controller generates a series of $M$ control outputs, one for each of the next $M$ actuating time instants. The first one is computed directly from the previously estimated state. Subsequent control signals are computed assuming the estimated state given the computed control signal and the control law.
   After having this set of control signals, the controller sends them in an a single message to the respective actuators. Once again, sending larger messages may have the implications that were discussed in the sampler description.
   This second task is both self-activated or activated by the first task. It is activated by the estimator when the estimator reaches the conclusion that the states estimated by the controller are considerable different from those reported by the sampler. It activates itself if it remains inactive for a constant number of periods after its last activation.

An SOD-like system was implemented, in which the sensor and the controller have threshold values. Whenever the sampled value is outside a given range of the last sent value or a given amount of time has passed, a new message is sent. Similarly, when the control signals estimated in the last message differ significantly from the control signals computed given the actually state trajectory, or a given amount of time has passed.

### C. Actuator

The actuator *translates* the latest available value into the physical world. The actuator is synchronous, changing its value at well defined time instants (mostly likely periodically). When a new message is received, the next output value is set to the first value of the array. If the time to output the next value is reached before a new message is received, the second value of the array is outputted. Obviously, this assumes that messages that have not arrived until a given moment were lost, as opposed, for example, to messages arriving out of order. Whenever all values of the messages are outputted, the actuator keeps outputting the last value. However, this situation

will likely occur with a very low probability, since the array size can be chosen in order to avoid it.

## V. Performance Assessment

To perform an experimental validation of the proposed architecture — in this case a verification of used bandwidth reduction — one system was simulated, in the Matlab® numerical computation environment. TrueTime [20] was used to implement the networks (Ethernet) and kernels.

The simulated system had the following continuous-time state-space representation:

$$S1 : \dot{x} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u, \quad y = [0 \quad 1]\, x + v \qquad (1)$$

which has ($s$-plane) poles at $-1$ and $0$. The system was sampled every $10ms$. Pole-placement was used to put a closed-loop double pole at $0.98$ ($z$-plane). The input of the system was perturbed with white noise, for six equidistant RMS points in the range $0 - 10$.

Two groups of experiments were performed. The first group was done without output noise and the second was simulated with an output noise similar to those generated by a $10$ bit ADC with no other non-idealities ($10$ bit due to its high availability in contemporary micro-controllers).

Each experiment was performed for six threshold values. The threshold values of the sampler and the controller were set proportionally, i.e. $Th_c = k * Th_s$, where $Th_c$ is the threshold in the controller, $Th_s$ is the threshold in sensor and $k$ is a constant factor. A value for the *gain* $k$ that gave a good relation between used bandwidth and the Integral Squared-Error (ISE) was found by trial-and-error. The value used in the simulations was $0.1$. Due to the random nature of the experiments (noise), each experiment was performed $160$ times.

Figure 2 depicts the variation of the ISE as a function of the SOD threshold value for several input noise levels (the ISE graphs with output noise are similar to the ones presented here). From this figure it follows that, using this control strategy, the greatest contribution to the ISE comes from the SOD threshold and that the input and output noise levels have a very small effect on the ISE, which was expected due to the SOD nature of the system.

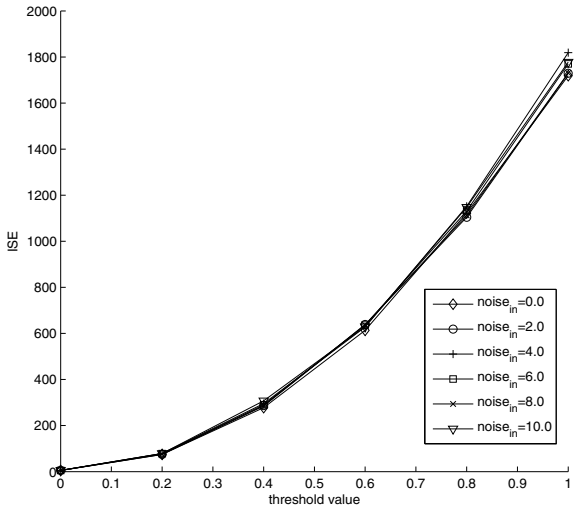


Fig. 2. ISE versus threshold for several noise values

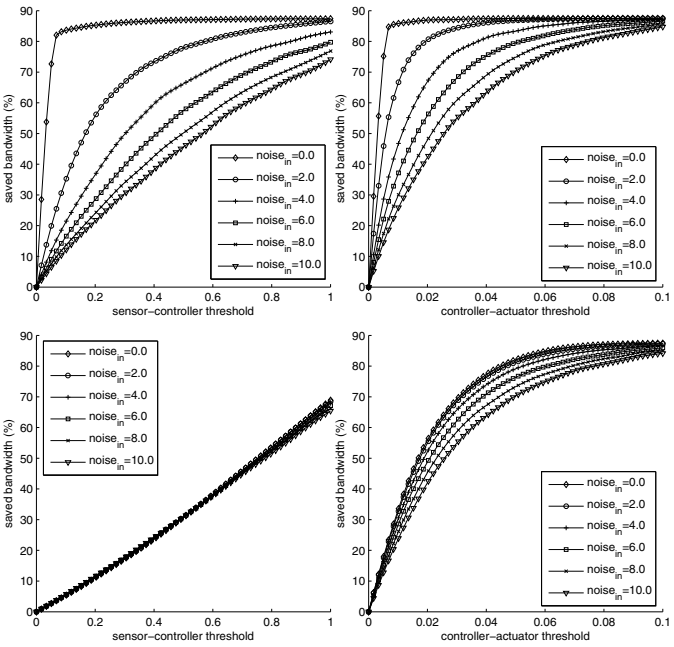


Fig. 3. Threshold vs bandwidth for several noise values (top—no output noise, bottom—with output noise, left—sensor to controller, right—controller to actuator)

Figures 3 shows a series of bandwidth measurements as a function of threshold for several noise levels. In particular they show the saved bandwidth in percentage, which is computed as:

$$sb = 100 \times \frac{b_{\text{standard}} - b_{\text{used}}}{b_{\text{standard}}}, \qquad (2)$$

where $b_{\text{used}}$ is the bandwidth that was measured in the simulation and $b_{\text{standard}}$ is the bandwidth used by a standard controller, under similar systems and networks.

The first set of graphs (upper-left) show that the saved bandwidth tends to $87.5\%$ when the threshold values grow unboundedly. This happened because at least one out of $N$ messages is transmitted even if the thresholds were not reached (as described before). Thus, when the threshold is significantly larger than the error in the signals, the saved bandwidth tends to $100 \times (N-1)/N$ percent. In this case, $N = 8$ (thus converging to $87.5\%$). Since Ethernet was used, a message with $8$ blocks of $2$ bytes each is well within the minimum payload of $46$ bytes, hence the message fits in what normally would be padding.

On a similar experiment (down-left) done with ADC reading error, the transmission rate was limited by the ADC noise, i.e, its effects were enough to trigger a transmission (sooner than the effects of the input noise alone). Hence, the saved

bandwidth was less sensible to the input noise.

The controller actuator transmission without ADC reading errors — top right — is similar to its sensor-actuator counterpart. However, in the presence of ADC reading errors — bottom right — there is no significant qualitative difference. Mostly because those errors got filtered out due to the higher rate at which the sensor send its messages

Note that not all messages sent from the sensor to controller triggered a message from the controller to the actuator. If the controller notes that the new messages do not change significantly its state estimate and consequently its control outputs, no new messages will be sent. That is the primarily reason why the two rightmost graphs of figure 3 are rather similar.

Other aspect worthy of remark is the convergence rate of the curves. In particular, the curves are converging to the value mentioned above at different rates. This is due to the fact that under the same threshold value, systems with lower noise transmit less messages. In the extreme case of no noise, as depicted in the top curve of the two top graphs in figure 3, the curve converged to a step function. When the threshold is zero it sends all the messages. When it is not zero, due to the estimators, it only sends 1 message out of $N$.

## VI. Summary and Conclusion

A new control architecture was presented. Similarities and differences between it and controllers or architectures described in the literature were discussed.

The new architecture was tested on simple system. Results showed a clear reduction of used bandwidth, under Ethernet, without a significant ISE increase. Future contributions may include numerical computations of the ISE under a given threshold, that will allow the computation of the best threshold for a given ISE. And it may also include the optimal value of $N$ (described above) for a given threshold.

## References


[1] S.-L. Jämsä-Jounela, "Future trends in process automation," *Annual Reviews in Control*, vol. 31, no. 2, pp. 211 – 220, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0H-4R1140T-1/2/080915af1460491c8f0767c348ee1ed0

[2] E. T. Nuno Pereira, Luis Miguel Pinho, "Ethernet-based systems: Contributions to the holistic analysis," *Proceedings of the 16th Euromicro Conference on Real-time Systems (ECRTS'4)*, vol. 4, pp. 25–28, july 2004.

[3] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," in *Proceedings of the IEEE*, 2007, pp. 138–162.

[4] A. Antunes, P. Pedreiras, and A. Mota, "Adapting the sampling period of a real-time adaptive distributed controller to the bus load," in *Emerging Technologies and Factory Automation, 2005. ETFA 2005. 10th IEEE Conference on*, vol. 1, september 2005, pp. 4 pp. –1084.

[5] M. Velasco, P. Martí, J. M. Fuertes, C. Lozoya, and S. A. Brandt, "Experimental evaluation of slack management in real-time control systems: Coordinated vs. self-triggered approach," *System Architecture*, vol. 56, pp. 63–74, January 2010. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2009.11.005

[6] A. Anta and P. Tabuada, "On the benefits of relaxing the periodicity assumption for networked control systems over can," in *Proceedings of the 2009 30th IEEE Real-Time Systems Symposium*, ser. RTSS 09. Washington, DC, USA: IEEE Computer Society, Dec 2009, pp. 3–12. [Online]. Available: http://dx.doi.org/10.1109/RTSS.2009.39

[7] R. Marau, P. Leite, M. Velasco, P. Marti, L. Almeida, P. Pedreiras, and J. Fuertes, "Performing flexible control on low-cost microcontrollers using a minimal real-time kernel," *Industrial Informatics, IEEE Transactions on*, vol. 4, no. 2, pp. 125 –133, May 2008.

[8] M. Epstein, L. Shi, A. Tiwari, and R. M. Murray, "Probabilistic performance of state estimation across a lossy network," *Automatica*, vol. 44, no. 12, pp. 3046 – 3053, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/B6V21-4V1K52C-3/2/7c213f2088570d38e68e911e5d612275

[9] W. Zhang, M. Branicky, and S. Phillips, "Stability of networked control systems," *Control Systems Magazine, IEEE*, vol. 21, no. 1, pp. 84 – 99, Feb 2001.

[10] C. Lozoya, P. Marti, M. Velasco, and J. Fuertes, "Analysis and design of networked control loops with synchronization at the actuation instants," in *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, vol. 34, Nov. 2008, pp. 2899–2904.

[11] R. Tomovic and G. Bekey, "Adaptive sampling based on amplitude sensitivity," *Automatic Control, IEEE Transactions on*, vol. 11, no. 2, pp. 282–284, Apr 1966.

[12] P. Otanez, J. Moyne, and D. Tilbury, "Using deadbands to reduce communication in networked control systems," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 4, 2002, pp. 3015–3020 vol.4.

[13] Y. S. Suh, V. H. Nguyen, and Y. S. Ro, "Networked estimation using a send-on-delta method," in *IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on*, nov. 2006, pp. 4622 –4626.

[14] J. Yook, D. Tilbury, and N. Soparkar, "Trading computation for bandwidth: reducing communication in distributed control systems using state estimators," *Control Systems Technology, IEEE Transactions on*, vol. 10, no. 4, pp. 503–518, Jul. 2002.

[15] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, p. 379423 & 623656, July 1948.

[16] K. Ogata, *Digital Control Systems* ., 2nd ed. Saunders College Publishing, 1992.

[17] B. W. Karl J. Astrom, *Computer Controlled Systems: Theory and Design*, 3rd ed. Prentice Hall, 1997.

[18] G. B. L. Constantine H. Houpis, *Digital Control Systems: Theory, Hardware, Software*, ser. McGraw-Hill series in electrical engineering, K. J. Hunt, Ed. McGraw-Hill, 1985.

[19] *CAN Specification : Version 2.0*, Robert Bosch GmbH, Postfach 30 02 40, D-70442 Stuttgart, september 1991.

[20] 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*, vol. 3, no. 23, pp. 16–30, June 2003.