Stability of time-sensitive networks: strategies for partial regulation.

Project report, by Nicolò Dal Fabbro*
Advisor: Ludovic Thomas
Professor: Jean-Yves Le Boudec

***Abstract*—In time-sensitive networks, regulators can be used to reshape traffic, and their usage may be necessary to guarantee stability by providing worst-case delay bounds. In this project, I study partial regulation of time-sensitive networks with cyclic dependencies, focusing on performance analysis. Using recently developed Network Calculus tools, such as TFA (Total Flow Analysis), TFA++, FP-TFA (Fixed Point Total Flow Analysis) I study how delay bounds can be improved by varying position and number of per-flow regulators (PFRs) in a FIFO-per-class network. The problem is non-trivial because in very complex networks the number of possible combinations of positions becomes huge.**

I find that the choice of position and number of regulators can bring to significant improvements in the computed delay bounds.

The main part of the analysis is done on a synthetic use-case network. The results of the analysis are discussed and some heuristics for the general case are inferred.

I apply the work done to a realistic network by proposing an algorithm that aims to jointly find good positions and number of regulators in a general topology through a parametric analysis. In the realistic case, I find significant correspondences with the synthetic use-case results and considerations.

I. Introduction

In time-sensitive networks with cyclic dependencies, the full deployment of regulators allows the computation of worst-case delay bounds without the need of solving fixed-point problems. In certain cases, with fixed-point methods is not even possible to provide a delay bound, and in that case the deployment of regulators may become necessary to guarantee the stability of a network with cyclic dependencies. The deployment of regulators can either be a full deployment, in which at each node a regulator is placed, or a partial deployment, in which regulators are placed only at some nodes. The problem of placing the lowest possible number of regulators to break all cyclic dependencies is faced and solved in [1], where the algorithm Low-Cost Acyclic Network (LCAN) is proposed. With LCAN we can find the optimum number of regulators both for per-flow regulators (PFRs) and interleaved regulators (IRs). While the focus of [1] is in breaking all the cyclic dependencies guaranteeing the feed-forward computability of the worst-case delay bounds of the network, in this project I focus on the possibility of deploying partial regulation, specifically of PFRs, with the purpose of improving the delay bounds. To do this, I start by analyzing a use-case academic network, studying how by changing position and number of regulators we can improve performances, i.e., get better delay bounds. After the analysis of the academic network is concluded, I do some heuristic considerations to generalize the analysis done to more complex and realistic networks. To conclude, applying the work done, I build a procedure that, by using an algorithm that returns positions and number of regulators in a network, allows to look for good combinations of these variables through a parametric analysis. The algorithm is based on the bottlenecks of the network, and on the idea of an optimal fraction of flows to be regulated w.r.t. the bottlenecks. The algorithm, furthermore, makes use of LCAN for its first step. In the realistic network analysis, I get results that have strong correspondences with the heuristics proposed. In terms of delay bounds, for high utilization and a class of flows that takes all the capacity, through the presented procedure is possible to get a partial deployment configuration that has the same performances of the full deployment by using much more less regulators. In section II, the synthetic network is presented in details, together with the parameters adopted and the methods used to compute delay bounds. In Section III is detailed the analysis of the academic network performances when the variable is the position of a single PFR. In Section IV I show what are the best bounds achievable in the network if we can use

*Exchange student at EPFL, Section of Communication Systems. Home University: Università degli studi di Padova: nicolo.dalfabbro@studenti.unipd.it

any number of regulators, from a single one to full deployment, placing them in the optimal positions. In Section V there are considerations for general heuristics that is possible to infer from the previous analysis. In Section VI, I illustrate the simple algorithm that selects the places where add regulators. A quick note on the software used for the measurement campaigns is shown in Appendix. Mathematical details on the delay computations are omitted, the methods used to compute delay bounds through Network Calculus tools are the same presented in [1]. The needed background to understand the work is also in Appendix.

II. System Description

The main work of this project consists of the analysis of an academic use-case network, shown in Figure 1. The network shown in the figure is a a convergent network forming an asymmetric ring with a cyclic dependency. The notion of cyclic dependency will be used in all the project report, so the same definition reported in [1] is quoted here:

Definition. For a given network, consider its underlying directed graph $G = (V, E)$ defined by: V, the set of vertexes, is the set of output ports in the network. For a, b two output ports in V, (a, b) is a directed edge in E if at least one flow crosses output port a and just after output port b. A cyclic dependency in the network is defined as a cycle in its underlying graph.

In all the report, a graph node will be identified as a tuple (network node, output port). In the network in the figure, each network node is associated with an output port, and in an equivalent way it could be associated to the scheduler representing the FIFO system preceding the output port. Since each network node is associated with only one output port, in the network of the figure is straightforward to derive the underlying graph. Each (network node, output port) tuple has the same structure as the one shown in Figure 2. In the nodes of Figure 1, the packetizer block is omitted, while the curved grey shapes inside the node represent the places where a per-flow regulator (PFR) could be put. In Figure 1 there are $N = 6$ nodes numbered from 0 to $N-1$. With the exception of node $N-1$, where two flows are generated, one flow is generated at each node. Flows are numbered from 0 to N. Each flow f_k is supposed to be generated inside the corresponding node n_k, with the exception of flow f_N, generated inside node n_{N-1}. Since node n_{N-1} is the

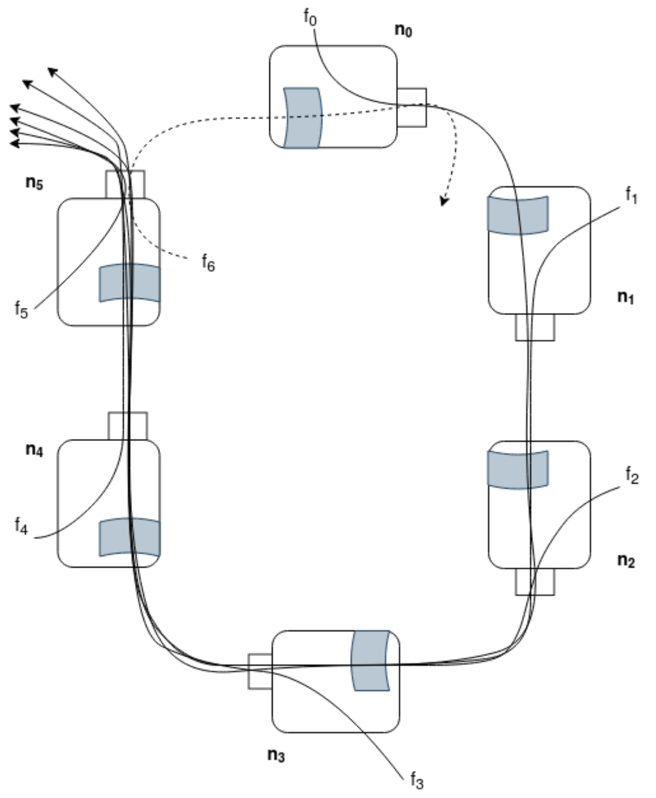

Fig. 1: Use-case network with $N = 6$ nodes. Flow f_6 is drawn with a dotted line to distinguish its path from the others. The curved grey shapes inside nodes are the positions where a regulator could be placed in.

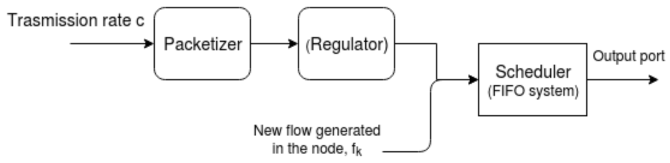

Fig. 2: Device model. Regulator is in parentheses because it may or may not be placed depending on the design choice.

node crossed by more flows, it will be referred to as the bottleneck of the network. Each flow is supposed to have the same initial arrival curve $\alpha_{r,b}$, that is a leaky-bucket arrival curve of rate r and initial burst b. To each (network node, output port) tuple is associated a FIFO system with rate-latency service curve $\beta_{R,T}$ of rate R and latency T. I will always consider one single class at a time. A PFR, by definition, will regenerate the initial arrival curve $\alpha_{r,b}$ for each single flow it is regulating, before these flows are given in input to the FIFO system that the PFR is preceding. The cyclic dependency in the network is caused by flow f_N that is closing the ring, hence generating a cycle in the network graph.

A. *Delay computation*

To compute the end-to-end (ETE) delay bounds, total-flow analysis (TFA) is used, with some improvements that will always be used, such as the line shaping improvement (presented in [2]), the results on packetization of [1], Mohammadpour et al. improvement shown in [3]. Since the network is cyclic, we can not directly do a feed-forward computation of the ETE delay bounds if no regulator is placed. On the other hand, one PFR placed in any of the possible positions breaks the cyclic dependency so that a feed-forward computation of the ETE delay bounds is possible. To compute the ETE delay bounds of the flows, three possibilities are considered: either we break the cyclic dependency by placing one or more regulators in some of the graph nodes (partial deployment), either we place a regulator in every graph node (full deployment), or we apply FP-TFA, presented in [1], a technique that uses a fixed point method to compute ETE delay bounds in networks with cyclic dependencies.

B. *Parameters and metrics*

All the measurement of the project are done with the same set of parameters, with some of them been varied for analysis purposes, so the parameters are all listed in this section.
The packets have all the same size that is $l = $ 12E3bits, the transmission line coefficient is $c = $ 1E9bits/s, the initial burst $b = $ 12E3bits, the arrival rate of the flows, denoted as r, depends on the utilization factor, denoted as ρ. The analysis of the considered networks is done considering two different values of the utilization factor, representing low and high utilization of the network, that is $\rho = 0.3$ and $\rho = 0.8$, respectively. Furthermore, I also consider two possible values for the ratio between the transmission rate c and the service rate R, that is $c/R = 1$ and $c/R = 2$, values that are also considered in [1]. The former configuration represents a case in which the service rate of the considered class takes all the available capacity, and is therefore likely a high priority class; the latter case is more general, and includes more possibilities common in industrial networks. In the case $c/R = 1$ the effect of line shaping will be evident, while when $c/R = 2$ the same effect will be strongly reduced in place of other factors, as will be clear in the next section.
To evaluate performances, the metric that has been adopted as the most suited to be considered is the delay bound per hop. This metric was suggested by the asymmetry in terms of hops that the flows cross in the use-case network. To obtain the measure of the delay bound per hop associated to a flow, I compute its ETE delay bound and divide it by the number of nodes that the flow crosses.
In the analysis I will focus on the worst delay bound per hop that is computed for a given configuration. Sometimes also the average of the computed delay bounds per hop will be shown.

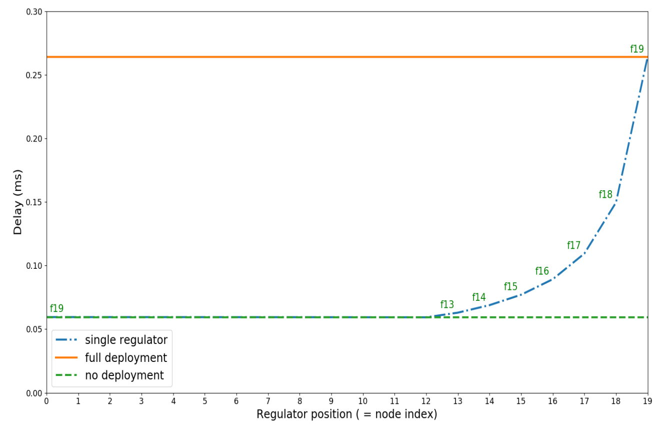

Fig. 3: Worst delay bound per hop vs index of the node where the regulator is placed, for the network of Figure 1 with $N = 20$ nodes, load factor $\rho = 0.3$, $c/R = 1$. Other parameters detailed in Section II-B. The flow suffering the delay is indicated close to the point as f_k. When omitted, the flow does not change w.r.t. the previous point (previous to the left along the x-axis.)

III. Single regulator analysis with $N = 20$

In this section, I analyze in detail how the performances of the network in terms of computed delay bounds per hop change with respect to the positioning of one single regulator in the use-case network of Figure 1, introduced in section II, when the number of nodes is $N = 20$. In each of the plots that will follow, three cases are shown and compared: the no-deployment case (delay bounds computed with FP-TFA), the single PFR case, and the full deployment of PFRs case. When one or more PFRs are placed in the network, as stated before, the cyclic dependency is broken and then it is possible to compute the delay bounds through a simple feed-forward computation. In all the plots of this section: on the x-axis is put the index of the node n_k where the regulator is placed, on the y-axis the worst delay bound per hop computed; close to the plotted point is also indicated which is the flow suffering that delay, f_k. When non specified, the flow suffering the delay of a plotted point is to be found as the previous specified flow (previous means: going to the left along the x-axis).

A. *Case* $c/R = 1$

When considering $c/R = 1$, the line shaping effect has an important role. This is clear from Figure 3, where delay bounds are computed at low utilization ($\rho = 0.3$). I first focus on the single regulator plot: as the single regulator placed in the network approaches the bottleneck (i.e. node n_{19}, in the case $N = 20$), the delay bound gets worst, especially as we get closer than a certain position, that in the specific case is node n_{12}, to the bottleneck. This is related to the line shaping effect. Since we consider a set of convergent flows, we originally expect, if non giving much importance to the line shaping effect, to see an improvement in the delay bounds as we approach the bottleneck with the regulator. This because the PFR will regenerate the original bursts for all the flows, consequently canceling the burst propagation effect due to the several FIFO systems crossed by the flows previously. The key point to be considered here is that the regulator also eliminates the line shaping effect for all the flows it regulates before the FIFO system it is preceding. The fact that this regenerative property of the PFR is causing the increasing of the delay bound is also suggested by the information shown in Figure 3 about which flow suffers the worst delay bound: in the considered case, after position $k = 12$, it is always the flow generated inside the same node where the regulator is placed. For example, for the PFR placed in node n_{13} the flow suffering the worst delay bound is f_{13}, and this holds for all the positions n_k with $k \geq 13$. In this configuration there is not a strong unique optimal position for the single regulator.

The full deployment is the configuration giving the worst possible delay bound per hop. This delay is equal to the one corresponding with the worst position of the single regulator, and the flow affected by this delay is again f_{19} as for the case of the single regulator put in position n_{19}. It is meaningful to be noticed that the full deployment bad delay computed does not depend on nothing but the final node n_{19} and the corresponding regulator: the flow suffering is f_{19} generated in the node itself, and what this flow faces is the set of all the flows of the network whose bursts are regenerated exactly before f_{19} enters in the system. The bad delay of the full deployment is due then to the very specific topology considered, and cannot therefore be considered a result with general validity. What is worth noting is, instead, the importance of the line shaping effect in this configuration and how the burst regeneration property

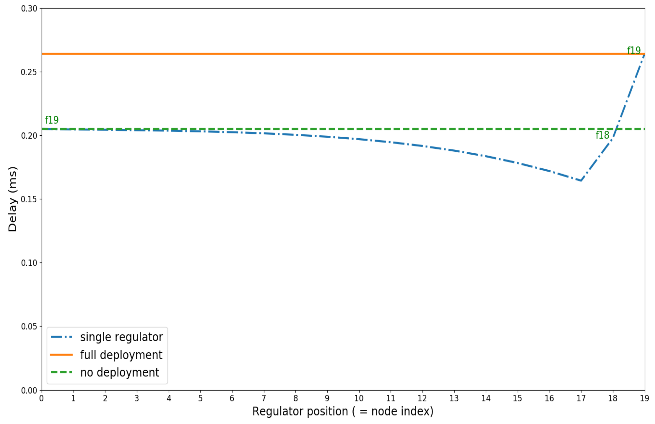

Fig. 4: Same as Figure 5, but for $\rho = 0.8$

of the regulator can have a negative impact on the delay bounds. The no-deployment case, as also was shown in the analysis done in [1], results to be equal or better, at low utilization and $c/R = 1$, both than partial and full deployment. In Figure 4 are shown the same measures that were shown in Figure 3 but now the utilization factor is $\rho = 0.8$. In the high utilization case, there is an optimal position corresponding to the regulator placed in node n_{17}. In this configuration, due to the higher rate, we see how the regulator brings an advantage with respect to both full deployment and no deployment. The regulator brings an advantage if placed closer to the bottleneck, shaping, in particular, 17 flows out of 21: at that point, in the convergent network, the overall propagated burst is pretty high, and the line shaping effect, even if $c = R$, is giving a weaker advantage with respect to the regeneration of the original bursts for all the flows. The improvement we get both with respect to no-deployment and to a position like node n_0, in terms of delay bound per hop, is of around $40\mu s$. In relative terms, the delay bound per hop is improved of roughly 20% with respect to no deployment. With respect to the worst positioning of node n_{19}, and so also of full deployment, the improvement is of around $100\mu s$, with a relative improvement of almost 40%.

B. *Case* $c/R = 2$

When $c/R = 2$, the line shaping effect is strongly reduced, and the burst propagation has a stronger impact in the delay bounds w.r.t. the case $c/R = 1$. Is expected, therefore, that the properties of the PFR will have a more crucial impact. The delay bounds computed in this configuration, with load factor $\rho = 0.3$ and $\rho = 0.8$ are plotted, in the same framework of the previous case, in Figure 5 and Figure 6. With this parameters configuration, the full deployment is not anymore providing

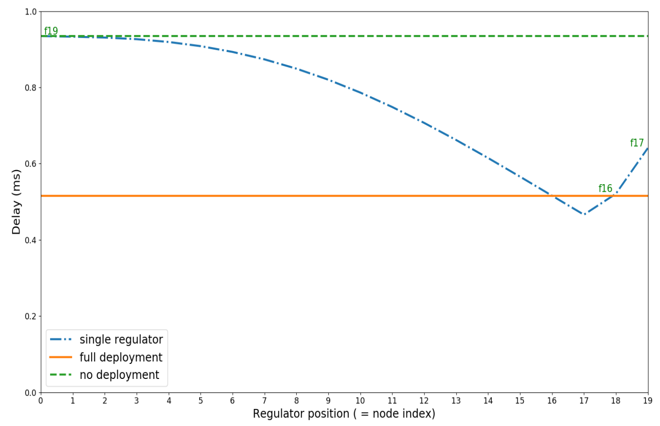

Fig. 5: Worst delay bound per hop vs index of the node where the regulator is placed, for the network of Figure 1 with $N = 20$ nodes, load factor $\rho = 0.3$, $c/R = 2$. Other parameters detailed in Section II-B. The flow suffering the delay is indicated close to the point as f_k. When omitted, the flow does not change w.r.t. the previous point (previous to the left along the x-axis.)

the worst bound, that is instead provided by the no-deployment. With both high and low load, there is a strong optimal position for the single regulator, that in this case is given by node n_{17} for the low utilization case and node n_{16} for the high utilization case. The reason behind this result was not immediate to understand, since as we know and see that the line shaping is not anymore playing a key role we may expect that the best position is as close to the bottleneck as possible. The reason for which the optimal position is instead some node before is easier to understand noticing which flows are suffering the worst bound. As is possible to see from both the figures, these flows are, as the node at which the PFR is placed approaches too much the bottleneck, those generated some hops before, e.g. flow f_{16} for the regulator positioned in node n_{18}. From this consideration I can infer the reason for the increasing of the worst delay bound per hop as the regulator approaches the bottleneck to be the unbalance due to the non regulation of all the convergent flows that come before. These flows generate a burst propagation so big that, if non regulated, produces the flows joining after a certain position (but before the PFR) to accumulate very big delay bounds, getting then the worst delay bound per hop. The improvement in terms of delay bound per hop that we get from choosing the optimal position instead of the no deployment or the worst position is of around $500\mu s$ with a relative improvement of roughly 50%. Interestingly, as shown in Figure 5, with low utilization, the optimal position is such that the single regulator provides a better improvement w.r.t. the full deployment. The improvement, in particular, is of around $50\mu s$, with

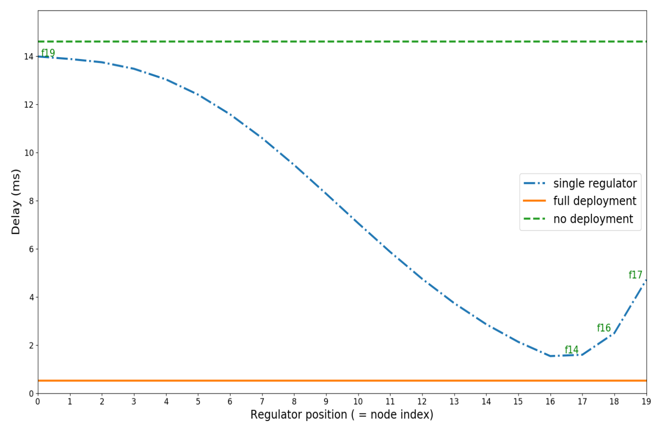

Fig. 6: Same as Figure 5, but for $\rho = 0.8$

a 10% relative improvement.

At high utilization, the improvement obtained by wisely positioning the regulator or by using full deployment is huge. The optimal position can improve the delay bound per hop of up to $12.5ms$ w.r.t. the worst position, and even more w.r.t. the no deployment. Of course, this huge improvement is also due to the very specific topology considered. The full deployment in this case provides the best bound, even if the difference with the single regulator in the optimal position is relatively small, if compared with the other possibilities.

In the next section, I show how increasing the number of regulators it is possible to improve the performances for these last two cases.

IV. Arbitrary number of regulators with $N = 20$

In the previous section, the analysis on the possibility of delay bounds improvement was done w.r.t. a single placed regulator. As was shown, the choice of the single regulator position can bring significant improvements in the delay bounds of the considered use-case network. In this Section, instead, I explore the possibility of improving the delay bounds in the same network even more by placing more than one regulator. I am not anymore interested in the specific position that the PFRs will be placed in, since the reasons behind the result would be solely related to the specific topology, and hardly generalizable. What I am really interested in is what is *achievable*, to get some more general suggestions easier to extend to a more general case. For this reason, in the following, I show what is the better bound we can achieve with a specific number of PFRs in the network, without specifying their position. Obtaining the result for this task was computationally demanding, more details in Appendix. When $c/R = 1$, as suggested by the

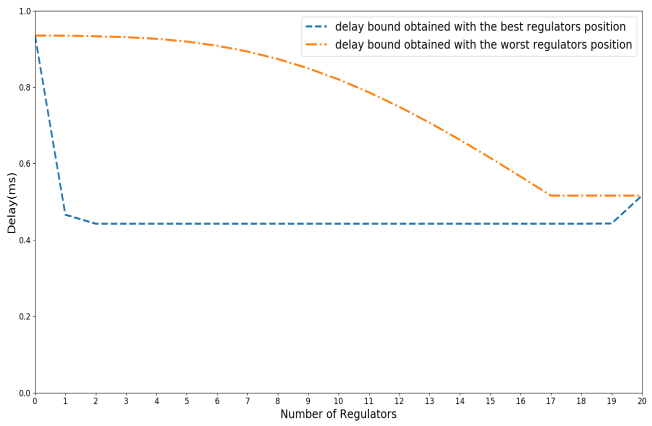

Fig. 7: Worst and best delay bound per hop vs number of regulators. To obtain the plotted delays, the regulators are positioned in the worst and best positions, respectively. Results obtained with $\rho = 0.3$, $c/R = 2$, details on the parameters in Section II-B

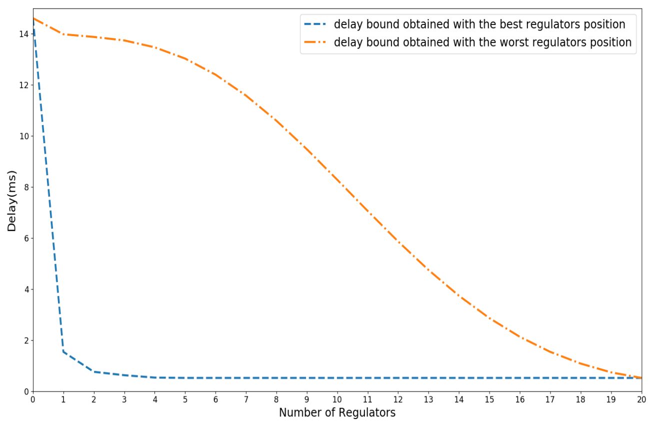

Fig. 8: Worst and best delay bound per hop vs number of regulators. To obtain the plotted delays, the regulators are positioned in the worst and best positions, respectively. Results obtained with $\rho = 0.8$, $c/R = 2$, details on the parameters in Section II-B

analysis done in the previous section, the bad effect of the regeneration property of the regulators w.r.t. the advantage given by the line shaping is such that using more than one regulator does not bring any improvement. In particular, with the number of regulators defined as K, in the case of $c/R = 1$ and low utilization (the single PFR analysis was shown in Figure 3), as expected from the fact that no deployment provides a better result than both full and single PFR deployment, having $1 < K < N$ does not provide any advantage. For the other case, i.e. still with $c/R = 1$ and $\rho = 0.8$, that was illustrated in Figure 4, the best bound we can get with one PFR is the same for $2 \leq K \leq N - 3$ PFRs. This is related to what was discussed in Section III-A: the full deployment very bad delay bound is much related to the specific topology and has not a meaningful general validity. The case $c/R = 2$ brings more information, and therefore the results obtained are shown in figures. The low utilization case, illustrated in the previous section in Figure 5, had the interesting result in which a single regulator provided a better delay bound than the full deployment if put in the optimal position. Surprisingly, by placing a number of regulators $2 \leq K \leq N - 1$ in the best possible positions, we get a slight improvement with respect both to the single optimal position and to the full deployment, as is possible to see in Figure 7. At high utilization, even when placing the PFR in the optimal position the delay bound obtained with full deployment was not achieved. The expectation for this case is that increasing the number of PFRs the delay bound obtained with the best positioning will approach the full deployment case. As shown in Figure 8, this is exactly what happens. What is interesting to note is that with $K = 5$ PFRs we already reach the same bound provided by the full deployment, that uses $K = 20$ PFRs. The other curves plotted in Figures 7 and 8 are useful to show the importance of the positioning of the PFRs: they show the worst delay bound obtained with the same number of PFRs if these are bad positioned. In the next section, the result of Sections III, IV are summarized and discussed.

V. Considerations and Heuristics

In Sections III, IV has been shown how for the academic use-case network illustrated in Figure 1 the best position and number of regulators can bring to a significant improvement of the delay bounds per hop. The question that this section and the remaining of the report tries to answer is the following: is it possible, based on the analysis done, to infer some heuristics valid for a general topology? To answer this question, several considerations have been done, and further analysis.

The considerations can be distinguished in two levels. The first is more general and related to the parameters, i.e., utilization factor, ρ, and the ratio c/R. What emerged from the previous analysis can be summarized in the following:

- $\rho = 0.3$, $c/R = 1$. No deployment and wise partial deployment provide equivalently good delay bounds. Depending on the topology, full deployment may provide less good bounds because of the trade-off between line shaping and regenerative property of regulators.
- $\rho = 0.8$, $c/R = 1$. Partial deployment provides a better delay bound than no deployment and there is an optimal positioning of regulators bringing to significant improvements. Full deployment as in the previous point.

- $\rho = 0.3$, $c/R = 2$. Full deployment provides good delay bounds, but partial deployment can provide equivalent or better bounds, if regulators are placed in good positions. No deployment is not a good option anymore.
- $\rho = 0.8$, $c/R = 2$. Full deployment provides the best delay bounds, achievable also with partial regulation choosing a minimum number of regulators and placing them in a good position. No deployment is not a good option anymore.

The above considerations are general, and they are discussed again in the next section, where a realistic network is analyzed.

The second level of heuristic considerations is now explained, always starting from the analysis done previously on the use-case network illustrated in Figure 1. The use-case network single regulator positioning has some interesting properties. For example, when putting a regulator on a certain node, with the exception of node n_0 and node n_{N-1}, we are also specifying the number of flows that we are regulating: for a PFR placed in node n_k, $k+1$ flows are being regulated. When we identify an optimal position of the single PFR, then, we are also identify a specific number of flows regulated. In particular, we are selecting a fraction of the flows that cross the bottleneck. From this observation comes the idea that is used in the next section to develop a simple algorithm with the purpose of selecting good positions and number of regulators. The idea is mainly that in a network there may be a good value, like a threshold, for the fraction of flows that go through a bottleneck that we would like to regulate to get good performances. The parameter related to the number of flows that we want to regulate w.r.t. a bottleneck is the parameter that is studied and tuned for a general topology.

VI. Application to a realistic network

In this section, the analysis done in the previous sections and the heuristic considerations are applied to a more general topology. In the first part of this section, I propose a simple algorithm to place regulators in a general topology according to a parameter. In the second part, I apply the algorithm to a realistic network and analyze the results.

A. Algorithm for a general topology

The algorithm that I propose here is based on the use of a parameter, that could be defined as a threshold, specifying a fraction of flows w.r.t. a bottleneck in a network, to place regulators. The procedure, illustrated in Algorithm 1, starts by using Low-Cost Acyclic Network (LCAN) to have the minimum number and position of PFRs to break all the cyclic dependencies of the network. After that, the algorithm scans the bottlenecks of the network, i.e. the graph nodes crossed by more flows.

The reason for which LCAN is used for the first step is because the algorithm is designed for complex networks with cyclic dependencies, where the number of cyclic dependencies may be huge, and therefore the base point of the positions that LCAN provides is needed. For each bottleneck, the procedure looks for neighbors satisfying a specific condition related to the input parameter, the threshold. The condition is satisfied if the neighbor shares with the bottleneck a fraction of flows bigger than the threshold. If the condition is satisfied, a recursion is done on the neighbors satisfying it, to look for other nodes satisfying the condition w.r.t. the original bottleneck.

Algorithm 1 Returns a set of edges for PFRs placing, according to an input parameter th. The "recursion" operation is explained in VI-A

```
Input: (Network graph $g$, Threshold $th$)
Output: Set of edges where to put PFRs, $P$
$P = \emptyset$
 // Sort (in decreasing order) the
 nodes according to the number of
 flows they are crossed by
$bNodes$ = sort(nodes($g$))
 // Use LCAN to get the regulators to
 break all the cyclic dependencies
$P = P \cup LCAN(g)$
foreach $node \in bNodes$ do
    foreach $n \in neighbors(node)$ do
        if $n$ satisfies threshold $th$ then
            $chosen$ = recursion($n$)
             // $chosen$ will be the edge
             between $n$ and $node$ or between
             $n$ and a neighbor of $n$ (or of
             a neighbor of the neighbor,
             etc...) satisfying the
             threshold condition
            $P = P \cup chosen$
return $P$
```

A good position for a regulator is established as the edge between the last node on which the condition was satisfied in the recursion and the successor of that node in the direction of the bottleneck. The procedure explained adds some regulators to the minimum set of regulators. Depending on the threshold parameter, the number of regulators added will change. If the threshold is small, I will add more regulators, and vice versa if it is closer to 1. The algorithm, then, will return both position and number of regulators to be placed. The proposed algorithm allows a parametric analysis of a given network, with the objective of finding a value that would produce a configuration with improved delay bounds. The algorithm and the proposed approach allow to find interesting results when applied to a realistic network, as will be shown in the next sub-section. An extension of this algorithm may involve a parametric analysis with more than one parameter, with an increasing in complexity.

B. Application to Orion network

I apply the algorithm and the approach described in the previous sub-section to Orion crew exploration vehicule (CEV) network. Its architecture is detailed in [4, p.328]. The physical topology illustrated in Figure 9. I adopt the same routing and analysis structure, such as number of flows and their mapping, that was adopted in [1]. So the number of flows is 119, and the routing configuration brings to have 293912 cyclic dependencies. The number of edges in the network graph is 249, so the total number of possible combinations of regulators placed in the network with partial deployment, if one wanted to check all the possible combinations, is of the order of $\approx 2^{249}$. Of course this would be an impractical way of finding good positions. In Figure 10 are shown the results in terms of delay bounds per hop, at high utilization, i.e. $\rho = 0.8$, and $c/R = 1$. As it is possible to see from the figure, partial deployment, for the value of the threshold equal to 0.275, allows to reach the performances of the full deployment, both for the worst delay bound per hop and for the average of the delay bounds per hop. When the value of the threshold is 0.275, the number of corresponding placed regulators in the network is 35. So, with a partial deployment using around 14% of the PFRs used by the full deployment, we manage to get the same performances of the full deployment. Compared with the result obtained by only using LCAN, the improvement is of around $55\mu s$ in delay bound per hop, relative improvement of around 30% (the same that full deployment provides). Of course this is not the true optimum positioning, but still the result can be considered as good. To see if the positioning of the regulators in the nodes specified by the algorithm is really a good choice, I compared the result obtained with

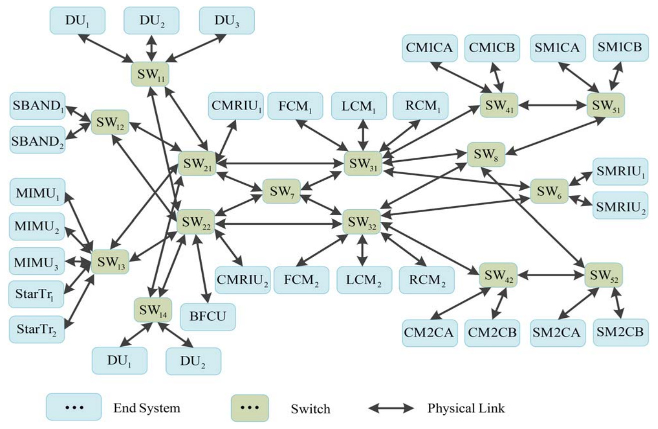

Fig. 9: Physical topology of the Orion CEV network

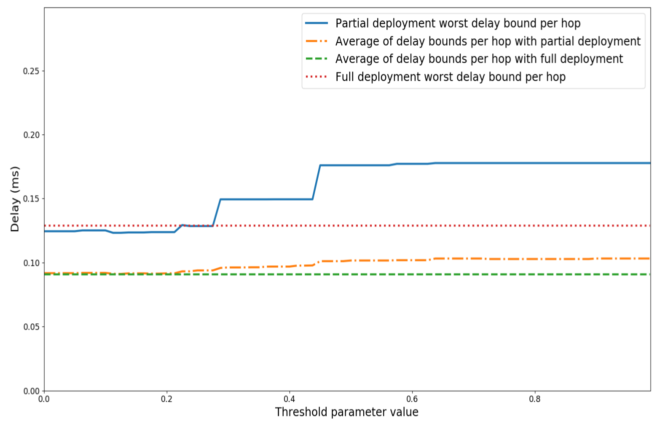

Fig. 10: Delay bounds per hop for Orion network vs value of the threshold parameter for which the algorithm returns certain regulators positions. The number of PFRs used at threshold=0.275 is K=35. Results obtained with $\rho = 0.8$, $c/R = 1$, details on the parameters in Section II-B

that configuration with the result obtained by randomly placing a number of regulators equal to the one of the regulators added by the algorithm in the network. The result obtained is shown in Figure 11. To obtain the plot, 2000 random choices for the 26 PFRs added by the algorithm have been done. As it is possible to notice, no one of the random choice delay bound obtained is better than the one obtained through the algorithm, and the margin between the majority of the bounds obtained with the random choice and the bound obtained through the procedure is significant, of around $30\mu s$ in delay bound per hop. For the other combinations of parameters that were the objective of the considerations of Section V, the results obtained with Orion bring confirmations and similarities. At low utilization, for $c/R = 1$, as was already seen in [1], the minimal number of PFRs to break the cyclic dependencies, that is 9, allows to obtain already better results w.r.t. full deployment. For the case

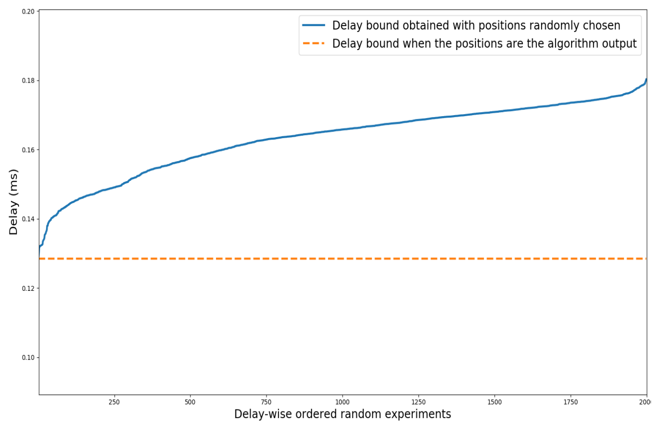

Fig. 11: Delay bounds per hop for Orion network vs number of random chosen 26 PFRs experiment, explained in this Section VI. The experiments have been ordered delay-wise. Results obtained with $\rho = 0.8$, $c/R = 1$, details on the parameters in Section II-B

$c/R = 2$ the plots obtained, omitted, are similar to the one of Figure 10, with the difference that at high utilization the partial deployment, even if getting close, does not reach the full deployment performances, so non reaching exactly the result promised by the general considerations of Section V.

VII. Conclusion

In the academic network example I have shown how crucial the choice of the position of a regulator can be in the delay bounds computation. The fact that increasing the number and jointly choosing the best position of regulators could bring further advantages gave other hints on the potential of partial deployment to improve performances. Analyzing the reasons that brought to the improvements it has been possible to infer some heuristics and to test a simple approach to improve the performances of a complex network as the Orion one, finding some confirmations on the previous considerations.

VIII. About project development

In this section I provide notes on project development, including the skills acquired, the report of the major events of the project, the difficulties, and a self-assessment.

A. Skills acquired

The work of the project started by getting familiar with Network Calculus tools. I first studied from the Network Calculus Tutorial [5], and then I consolidated my learning by studying from the paper ”On cyclic dependencies and regulators in time-sensitive networks”, [1], from the paper on fundamentals of the line shaping improvement, [2], and by checking weekly my learning and understanding with my advisor. After this first phase, I started to get familiar with the academic use-case network proposed and with the software to analyze it. In this phase, I compared results and computations done on the network through the software and done by hand, to ensure the correctness of the software added by me for the specific case to be considered, and being sure to have fully understood the mathematical procedures through which delay bounds are computed in Network Calculus, at least in the most basic cases. In the skills acquired for the project then for sure there are the fundamentals of deterministic networks, some insight of the existing literature, and also practical methods to compute measures used in the context of time-sensitive networks, such as delay bounds, in big networks, through the usage of software tools. Furthermore, this project was an opportunity to work on an individual project for three months, then collecting all the material developed and putting it in a final report, that was of course demanding but also instructive and satisfactory. Finally, during this project I improved my capability of using python object oriented programming.

B. Main events and difficulties

The first main event of the project was the first software computation of delay bounds for the academic network in the simplest configuration and with $N = 4$ nodes, together with the paper and pencil computation of the same measure. From that, it followed the analysis of the high priority low and high load computation and plot vs the position of the single regulator, from which it emerged the first optimum position found, that required some time to be fully understood. After that, the main events were the computation of the best delay bounds with an arbitrary number of regulators and finally the idea and work for the application to the realistic network. The concluding moment of the project was the moment in which was plotted the comparison between the delay bounds obtained by randomly choosing the regulators positions and the ones obtained by the configuration returned by the algorithm. Getting familiar with the Network Calculus tools required effort, especially at the beginning, but did not present unexpected difficulties, thanks to the material and the help from my advisor. The work with python software did not present particular difficulties for the majority of the time, except in some cases where the usage of dictionaries made it difficult some ordering operations and data processing. Some technical aspects that took unexpected time were also related to the will of getting some metrics from the analysis that were not immediate to be provided by the software, requiring some debugging. Extending the work of the synthetic network to a general topology required effort and presented some difficulties, but with the proper amount of time also those were solved.

C. Self-assessment

The work on the project through the semester was done regularly, and I managed to use a significant amount of time each week for the project, often achieving my week targets. From this point of view, I am satisfied with the project work and with my learning outcome. From the project results point of view, I think that I managed to accomplish the objectives' main features that were specified in the project contract. I consider to have succeeded in particular in the first aim, that was understanding how network performances could change by varying number and position of regulators in a network with a cyclic dependency. For the second aim, that concerned inferring heuristics for identifying best position and/or number of regulators without the need of computing delay bounds, I think I partially accomplished it by analyzing the results obtained in the work done and reasoning on the network configuration, and also testing the heuristics on the realistic case, but still I was not able to provide a completely strong set of rules. What instead I provide is the parametric approach that followed by the considerations done, and an example of its application.

REFERENCES

[1] L. Thomas, J.-Y. Le Boudec, and A. Mifdaoui, "On cyclic dependencies and regulators in time-sensitive networks," *2019 IEEE Real-Time Systems Symposium (RTSS). Proceedings*, 2019.

[2] A. Mifdaoui and T. Leydier, "Beyond the accuracy-complexity tradeoffs of compositional analyses using network calculus for complex networks.," *10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (co-located with RTSS 2017)*, Dec 2017.

[3] E. Mohammadpour, E. Stai, and J.-Y. Le Boudec, "Improved delay bound for a service curve element with known transmission rate," *IEEE Networking Letters*, 2019.

[4] R. Obermaisser, "Time-triggered communication," *CRC Press, Inc., 1st ed*, 2011.

[5] J.-Y. Le Boudec, "An introduction to network calculus," 2019.

IX. APPENDIX

In this appendix I add further information on the project, as the background needed to understand the work, information on the software used and measurement campaigns, and some details on project events.

A. Background

To fully understand the project, it is sufficient to have the fundamental notions on Network Calculus about the topics listed in the following:

- Deterministic worst-case delay bounds.
- Arrival and service curves, in particular Leaky Bucket Arrival Curves and Rate Latency Service Curves. Computation of the delay bounds for these curves. Computation of the output arrival curve for a FIFO system (min-plus deconvolution).
- The TFA method.
- Line shaping effect, and how it is applied to improve delays with TFA++ method [2].
- Packetization effect.
- Traffic shaping by means of regulators, in particular per-flow regulators.
- The contents on partial regulation presented in the paper "On cyclic dependencies and regulators in time-sensitive networks", [1].

These topics can be retrieved in the Tutorial "An Introduction to Network Calculus" by Prof. Le Boudec [5], and in the papers in Bibliography, [1], [2].

B. On software used and measurement campaigns

To do the analysis presented in the project I used the python software that I had available from the lab, in particular it was the code through which were done also computations for the results of [1]. This code uses object oriented programming, it is organized in classes the source code of which I had access to. I used and modify these classes, adding methods and structures when needed for my purposes. I coded some new functionalities and scripts for my measurement campaigns. The plots are realized using the python library matplotlib, the code to plot is integrated with the scripts with the code to obtain the results. The values obtained are usually put in lists that then are plotted. All the code is available on the github space that was made available for me by the lab.
The LCAN algorithm does not always return the same set of edges where the regulator should be placed. For this reason, repeated executions of the code and of the algorithm proposed in Section VI may produce slight different values of computed delay bounds. Furthermore,

also the number of added regulators may change. For example, for the threshold parameter equal to 0.275, sometimes we could get 30 regulators, sometimes 35. Anyways, the difference in the results is never substantial, and it does not contradict the conclusion and the considerations done. For example, for the threshold equal to 0.275, partial regulation as shown in Figure 10 always reaches the performances of full deployment, both in average and in worst bound.

C. Some insight on the project

To obtain the plots shown in Figure 7 and 8, for each curve were necessary roughly 2^{20} executions of TFA++ on the synthetic network. For each curve then several hours of computation were needed, and this was the most computationally demanding procedure. The code that implements the algorithm for the general topology the code was written starting from the cost function computation of LCAN. What the code does is assigning a cost to the edges between a node graph and a neighbor, and when the cost reaches a value of zero, that will be the place for a regulator. In this way the set of edges where to put regulators is identified.