# An admission control method to provide higher sampling rates over space launchers networks

Dorine Petit, Jean-Philippe Georges, Thierry Divoux, Bruno Regnier, Philippe Miramont

# An admission control method to provide higher sampling rates over space launchers networks ⋆

**Dorine Petit** [*,**] **Jean-Philippe Georges** [*,**]
**Thierry Divoux** [*,**] **Bruno Regnier** [***] **Philippe Miramont** [***]

[*] *Université de Lorraine, CRAN, UMR 7039, Campus Sciences, BP 70239, Vandœuvre-lès-Nancy Cedex, 54506, France (e-mail: firstname.name@univ-lorraine.fr)*
[**] *CNRS, CRAN, UMR 7039, France*
[***] *CNES, Direction des Lanceurs, 52 rue Jacques Hillairet, 75612, Paris, France (e-mail: firstname.name@cnes.fr)*

**Abstract:** Nowadays, embedded network systems, like in space launcher application, trends to move from a bus communication to components off-the-shelf (COTS) such as Ethernet switched network in order to reduce the cost and the mass. In order to guaranty the real time and reliability constraints on an Ethernet switched network which is not deterministic, the network is oversized and redundant. In this way, lot of resources are not used. This paper focus on using these available resources to support higher sampling rates for better controllability, safety and freshness. A framework for rate admission control is defined and an algorithm is proposed to maximise the sampling rate (i.e. the throughput) of each flow while satisfying maximum end-to-end delays requirements. The purpose of using this algorithm has been brought to light through an experiment based on the next generation of space launcher network.

*Keywords:* Admission control; sampling rate; real-time and dependability; switched network; space vehicles.

## 1. INTRODUCTION

### *1.1 Motivations*

Quality of Service is important in the case of critical applications where packets are sensitive to delay, jitter or packet losses. This is the case in space launcher applications where (i) availability (be robust at least at one failure), (ii) observability (command messages require fresh measurements – see Petit et al. (2016)) and (iii) real-time (end-to-end delay for critical messages has to be inferior than 1 ms) requirements need to be satisfied. In order to respect these requirements and to have a deterministic behaviour, the network is redundant and oversized. For example, the traffic load of scenarios that have been experimented on a space launcher platform (Petit et al. (2017)) varies between 2 Mb/s and 7 Mb/s on 100 Mb/s switched Ethernet network. Therefore, a lot of resources are still available.

From this observation, the extra available resources could be used to offer higher sampling rate while satisfying delays, availability and throughput constraints. Thereafter, sampling corresponds both to the measure from sensor and also to the command from controller. Multiple motivations lead to implement, what we can call oversampling. Indeed, this can (i) improve launcher controllability by sending command and/or measurement messages more frequently, (ii) increase the stability in terms of safety with more tolerance to loss and (iii) limit the observability uncertainty as defined by Petit et al. (2017).

The purpose of this paper is to apply admission with oversampling. It means that all sources (irrespective sensors or controllers) send packets at the highest sampling rate, and then rates admission control is applied at network equipments. Usually, when admission control is applied, the controller gives to the flow exactly the bandwidth required by the demand (Greff et al. (2017b,a)) or even less (Hertiana et al. (2015)). In this paper, flows throughput might be higher than required as sources oversample. The controller objective is then to filter the rate of each flow such that the capacity of the network and delays of all flows are respected and throughput is maximized.

### *1.2 Related works*

Greff et al. (2017b) purchased two objectives. The first one is to find a suitable path under delay and bandwidth constraint while allowing load balancing. And the second one is to provide a stable network; meaning that the network has a maximum number of buffered periodic real time packets bounded. The minimum number of sub-channel that satisfy the bandwidth constraint is chosen. This value will be used, in order to select path that have enough sub-channel left. A load balancing approach is chosen to find the path that satisfies the delay constraint.

⋆ Co-founded study by CNES and CRAN in the frame of CNES Launchers' Research and Technology program.

Table 1. State of the art (C:constraint,O:objective)

| Reference | throughput | nb of accepted flows | delay | loss | effective bandwidth | buffer |
|---|---|---|---|---|---|---|
| Greff et al. (2017b) | C | O | C | C | | |
| Guck et al. (2016) | C | O | C | | | |
| Kumar et al. (2017) | C | O | C | | | |
| Hertiana et al. (2015) | C | | C | C | | |
| Huang et al. (2014) | C | | C | C | O(min) | O(min) |
| This paper | O | | C | C | | |

Greff et al. (2017a) add a new feature, resilience, to the SDRN protocol presented in Greff et al. (2017b). It gives an algorithm that commands the paths and weight of the queues while respecting delay and bandwidth constraints for real time packets in the nominal case, in the failure case and during the transient reconfiguring period.

Guck et al. (2016) focus on the combined problem of routing and ressource allocation in order to achieve hard real-time guarantees in industrial network. An algorithm which routes flows on different queues is presented. The objective is to minimize the path cost function which is dependent on the burstiness, the buffer usage and the allocated buffer space. The algorithm guarantees three constraints: (i) rate of each queue, (ii) burst of each queue and (iii) end to end delay. It searches to allocate the maximum number of flows on the network.

Kumar et al. (2017) use a heuristic algorithm to solve a multi-constraint path problem. The purpose is to find a path that fulfilled delay and bandwidth requirement giving each flow is own queue. There is no dynamic in this paper, only path are implemented on OpenFlow switches in order to reproduce an AFDX network.

Huang et al. (2014) face the challenge of per flow admission control to scalability in an Open-Flow based SDN. It investigates flow aggregation for QoS and provides a solution to configure bandwidth and buffer space while meeting performance requirements in terms of delay and packet loss. Results show that their solution can reduce the total buffer space and the total effective bandwidth.

Hertiana et al. (2015) face the performance, in terms of delay, throughput and losses, of network congestion control problem in order to maintain excellent performance in the network. They design an OF SDN architecture which proposes both multipath routing and rate adaptation, in order to reduce delay, losses and maximize the throughput for each flow. The result shows that their method multipath and rate adaptation compare to single path and multipath method is better in terms of delay, throughput and losses. They do not precise when the algorithm stops searching for all paths and the convergence time.

The Table 1 shows that the objective function of Greff et al. (2017b); Guck et al. (2016); Kumar et al. (2017) is to maximise the number of accepted flows on the network. Indeed, throughput, delays, and potentially losses are considered as constraints and a flow is accepted only if all the constraints are respected. Otherwise the flow is not accepted on the network. In Hertiana et al. (2015), all flows are accepted, but constraints are potentially not respected. The objective of Hertiana et al. (2015) is to minimize the degradation of these constraints by optimizing network resources. Huang et al. (2014) focus on the scalability. They will try to minimize the effective bandwidth and buffer by using aggregation in order to accept more traffic on the networks. Finally, papers presented in this state of the art do not allocate more "throughput" higher frequency sampling than required by the flow. Indeed, they either guaranty the required throughput while others give less bandwidth than required, but never higher. In our case, we want to maximise flows frequency sampling by giving at least the required sampling and if possible a higher sampling.

The rest of the paper is organized as follows. The section 2 gives definitions. Section 3 proposes a framework to control the frequency sampling. Section 4 proposes an adaptive and progressive allocation. Section 5 shows on an example the benefits of the solution proposed and the last section 6 concludes.

## 2. DEFINITIONS

### *2.1 Topology*

The topology is defined by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. $\mathcal{V}$ corresponds to the set of nodes $i \in 1, 2, ..., |V|$ including switches and end devices. $\mathcal{E}$ is the set of links $(i, j)$. A link corresponds to the association of two nodes. This type of graph is defined as a simple and non oriented graph $((i, j) = (j, i))$. For a simple graph with a set of nodes $|V|$, the adjacency matrix is noted $\mathcal{A}$ such as $\mathcal{A}_{i,j}$ is equal to 1 when a link exist between these nodes $i$ and $j$ and 0 when it does not exist. The diagonal elements of this matrix are equal to zero, because one link connected to itself (loop) is not allowed in a simple graph.

### *2.2 Demands*

The set of demands is noted $K = \{1, 2, \ldots, N\}$. A demand $K_k$ is defined by a vector $\{src_k, dst_k, p_k, \underline{\rho_k}, \overline{\rho_k}, \delta_k\}$ where:

- $src$ is the MAC address source.
- $dst = \{dest_1, dest_2, \ldots, dest_d\}$ is one unicast or multicast MAC address destination.
- $p$ is the period or the minimal frame inter-arrival in seconds.
- $\underline{\rho}$ is the minimal rate in b/s.
- $\overline{\rho}$ is the maximal rate in b/s.
- $\delta$ is the end to end delay requirement in seconds.

As a reminder, the purpose is to maximize the frequency $1/p$ (the length of the frames is assumed constant) for every demand while respecting the constraints. Here, a minimal and maximal value of rate is given. Minimal value $\underline{\rho}$ corresponds to the required value of throughput, and consequently to the minimal end-to-end bandwidth to be offered by the network. Maximal value $\overline{\rho}$ are defined in

the demand to avoid absurd values (as for example a throughput of 100 Mb/s if a flow does not have any crossed traffic (i.e. that electronics will not support for instance). The maximal value depend on applications and hardware limits. Applications limits correspond to a logical value linked to the context, for example send to human a room temperature every millisecond is extreme.

### *2.3 Data plane*

The data plane defines, for every node, the next hop to reach a given destination and those for each demand. Multiple paths can be available for a pair origin-destination in a redundant network. Only paths that satisfy both required throughput and network capacity are taken into account. The matrix $\Pi$ gathers frames forwarding decisions for all flows/paths on every node. $\Pi$ is $|K| \times |V|$ such as $\Pi_{k,v} = v+1$ corresponds to the adjacent node $v+1$ on the path $\pi_k$, or $\emptyset$ if the flow is not forwarded by the node $v$. For a demand $k$, a path can then be easy to recover from the $\Pi$ matrix since $\pi_k = \{src_k, \Pi_{k,src_k}, \Pi_{k,\Pi_{k,src_k}}, \ldots, dst_k\}$.

### *2.4 Control plane*

Throughput is the amount of data sent during a time period for a given flow. Assuming a constant frame size, it means here that the throughput depends only on the rate $\rho$. The purpose of this paper is hence to maximize the rate while respecting constraints (delays, required throughput, availability) for every flow.

The control plane gathers throughput decisions (i.e. the rates) for all demands. For a given demand $k$, it is assumed that the admission control of each switch that belongs to the path will be configured with the same rate. Finally, the control plane is noted $\Gamma$ such that $\forall k \in K, \Gamma = \{\rho_0, ..., \rho_k, ..., \rho_{|K|-1}\}$.

## 3. PROPOSED FRAMEWORK

### *3.1 End to end delay computation*

One of the main constraint of an embedded network in space launcher, is the respect of the maximum end to end delay requirement. As known, Ethernet network is not deterministic and does not guaranty bounded delays. In this way the methodology used to calculate the delays needs to be deterministic. In this paper, we are considering the network calculus strategy since it has been used in real time Ethernet systems as shown in many papers (Boyer et al. (2012); Georges et al. (2011)) and multiple domains (industrial, controlled system, avionics, ...). This theory uses nodes concatenation, residual service curve and performances theorems as presented in Schmitt et al. (2008). Network calculus is based on min-plus calculations and allows to calculate maximum end to end delays. In order to calculate the end to end delays, we use the same methodology described in Petit et al. (2018) for feed forward networks.

### *3.2 Admission control principle*

A centralized approach is chosen both to obtain an omniscient view of the topology and demands and to facilitate the verification of the delay constraint (the network calculus theory needs to know all crossed traffic of each flow). In this paper, the limitation of the throughput is not done directly by the source but through network equipments (switches). The hypothesis is that end-devices maximise the sending frequency of the frames, such as the throughput at the source is equal to the maximum rate ($\overline{\rho}$) of the demand. Then on each queue belonging to the flow path, an admission control mechanism discards frames of a given flow $k$ such that the forwarding rate is limited to $\rho_k$.

Based on this framework, the problem consists then in determining for each demand $k$ the maximal value $\rho_k$.

## 4. ADAPTIVE AND PROGRESSIVE ALLOCATION

### *4.1 Objective and constraints*

The framework described above look for the value $\rho$ to admit on each switch. We decided for this paper to maximise the total utilization rate (in terms of throughput versus capacity) of the links such that we are proposing here to assess candidates according to the following novel objective function:

$$\left(\hat{\Pi}, \hat{\rho}\right) = \arg\max_{\Pi,\rho} \sum_{(v,w)\in E} \frac{\sum_{k\in K} \lambda_{\pi_k}^{v,w} \rho_k}{C_{v,w}} \quad (1)$$

with $\pi_k$ the path of a demand $k$, $C_{v,w}$ the capacity between nodes $v$ and $w$ and $\lambda_{\pi}^{v,w} = \begin{cases} 1 \text{ if } (v,w) \in \pi \\ 0 \text{ otherwise} \end{cases}$ a boolean to determine if a given edge belongs to a given path.

This equation is looking simultaneously for $\Pi$ data and $\Gamma$ control planes so that the cost is maximized. In fact, it is searching to use the full capacity of each link of the topology. The maximum value that can get this equation is when all the links are saturated.

We introduced then the following constraints, such that $\forall k \in K$:

$$\underline{\rho_k} \leq \rho_k \leq \overline{\rho_k} \quad (2)$$

$$D_k \leq \delta_k \quad (3)$$

where (2) is the verification of the admission rate regarding the minimal and maximal throughput bounds and (3) is the verification that the delays $D_k$ will remain inferior to the delay constraint $\delta_k$.

Maximising network performances to allocate maximal admission rate is a complex problem. We decide to not try to find the optimal solution in terms of throughput and we develop a heuristic which gives the best solution in a set of calculated solutions. In order to reduce the complexity, paths calculation and throughput allocation are done in two serials steps :

(1) firstly, the data plane is created: paths of all flows are calculated based on the number of crossed traffic instead of throughput (that will be allocated in the second step)
(2) secondly, the maximum admission rate, that can be allocated to every flow, based on the data plane created at the first step, is calculated.

*4.2 Data plane calculation*

As the purpose is both the maximisation of the forwarding rate offered to each flow and the exploitation of the network at its maximal capacity, paths are selected according to a double metric, taking into account:

- the maximum number of crossed traffic on one link of the path (weakest link), which is equals for a link $(v, w)$ to $\sum_{k \in K} \lambda_{\pi_k}^{v,w}$
- and the number of links where there is no crossed traffic, i.e. it corresponds to links such that $\left(\sum_{k \in K} \lambda_{\pi_k}^{v,w}\right) = 0$.

A path is better than another one if the first metric is lower, aiming at increasing the utilisation of all links. The second metric only applies in case of equality regarding the first one. In such case, the path relying on more links that have not yet been used (by the data plane) will be preferred.

One may note that the search can lead to longer paths (compared to a Shortest Path First strategy) and that the order into which flows and paths are considered may impact the data plane definition.

*4.3 Progressive control plane calculation*

The maximization of the throughput is then achieved according to the algorithm 1.

The inputs of the algorithm are the list of demands, the network graph and the data plane. The algorithm returns the control plane $\Gamma$.

It starts by the initialisation of all the throughput $\rho_k$, for each demand, with required throughput $\underline{\rho_k}$ (line 4). Then, a loop (from line 5 to 26) allows a progressive and proportional throughput allocation until each demand $K'$ has an offered throughput going to the maximum capacity of the network or the maximum throughput $\overline{\rho}$ if the maximum end to end delay is respected. First, we will verify if the maximum delay is respected (line 10) for each demand (line 9) if the throughput is increased (multiplied by 2) (line 8) for flows' throughput that can still be increased (line 7). If at least one flow does not respect the delay, then we consider that none of the flows can be increased and $\mathcal{K}' \leftarrow \emptyset$ (line 12). Then if there is still some demand that can still be increased, for each of them (line 14), it will be verified if the next proportional incrementation will not generate congestion (flag) or will not exceed the maximum throughput $\overline{\rho_k}$ (line 20). Congestion detection (from line 16 to 19) verifies for each link of the path flow if the sum of the flows' throughputs that cross this link will not exceed the physical capacity of the link (line 17) after the proportional incrementation of the flows' throughput that can still be increased. If the throughput increase do not generate congestion or an excess of the maximum throughput then the throughput is doubled (line 21). Otherwise, the demand is added to the demand list $Z$ corresponding to demands that cannot be increased any more (line 23). Once each demand have been processed, demands that cannot be increased any more are removed from the demands' list $K'$ that can still be increased (line 25). And so on, until all the demands' throughput cannot be increased any more (line 26).

**Algorithm 1:** Maximisation of the throughput

**Data:** $\mathcal{K}$ set of initial ordered demands, $G$ network graph, $\Pi$ data plane
**Result:** $\Gamma$ control plane

1 *control plane generation (progressive and proportional throughput allocation)*;
2 $\mathcal{K}' \leftarrow \mathcal{K}$;
3 delay_respected←true;
4 **foreach** $k \in \mathcal{K}$ **do** $\rho_k \leftarrow \underline{\rho_k}$;
5 **repeat**
6 $\quad \Gamma' \leftarrow \Gamma$; $Z \leftarrow \emptyset$;
7 $\quad$ **forall** $k' \in \mathcal{K}'$ **do**
8 $\quad\quad \rho'_{k'} \leftarrow 2 \times \rho'_{k'}$
9 $\quad$ **forall** $k \in \mathcal{K}$ **do**
10 $\quad\quad$ **if** networkcalculus($\Pi$,$\Gamma'$,$\mathcal{K}$,$k$)$> \delta_k$ **then**
11 $\quad\quad\quad$ delay_respected←false;
12 $\quad\quad\quad$ $\mathcal{K}' \leftarrow \emptyset$;
13 $\quad\quad\quad$ break;
14 $\quad$ **foreach** $k \in \mathcal{K}'$ **do**
15 $\quad\quad$ flag←true;
16 $\quad\quad$ **foreach** $(u, v) \in E$ *such as* $\Pi_{k_u} = v$ **do**
17 $\quad\quad\quad$ **if** $\sum_{k' \in \mathcal{K} - \mathcal{K}' | \Pi_{k'_u} = v} \rho'_{k'} + \sum_{k' \in \mathcal{K}' | \Pi_{k'_u} = v} 2 \times \rho'_{k'} > C_{u,v}$ **then**
18 $\quad\quad\quad\quad$ flag←false;
19 $\quad\quad\quad\quad$ break;
20 $\quad\quad$ **if** *flag and* $2 \times \rho_k \leq \overline{\rho_k}$ **then**
21 $\quad\quad\quad$ $\rho_k \leftarrow 2 \times \rho_k$;
22 $\quad\quad$ **else**
23 $\quad\quad\quad$ $Z \leftarrow Z \cup \{k\}$;
24 $\quad$ **if** *delay_respected* **then**
25 $\quad\quad$ $\mathcal{K}' \leftarrow \mathcal{K}' - Z$;
26 **until** $\mathcal{K}' = \emptyset$;

## 5. EXPERIMENTATION

This section shows the benefit of the algorithm presented in the previous section. Architecture and traffic definition correspond to the embedded network in the next generation of space launcher. The topology (Fig. 1) is a 100 Mb/s switched Ethernet network architecture. The architecture is divided in two networks and is composed of 46 machines and 4 switches (2 per network).

All the traffic going through the network is scheduled and known. In this way it is possible to use network calculus in order to calculate *a priori* the maximum end to end delay for each demand. Only messages going to controllers (OBC1-2) and sniffer is taken into account because it represents the maximum crossed traffic. 44 flows (per network) are multicast commands messages that are sent to OBC1, OBC2 and sniffer. The required frequency (which will define further the minimal rate $\underline{\rho_k}$) of these flows varies from 1,7 mHz to 100 Hz. 5 flows (per network) are unicast telemetry messages that are sent to sniffer only with a required frequency of 60 Hz. And in order to charge the network, 5 additional flows (per network) corresponding to camera flows are sent in unicast to the sniffer with a throughput of 2 Mb/s and packet size of

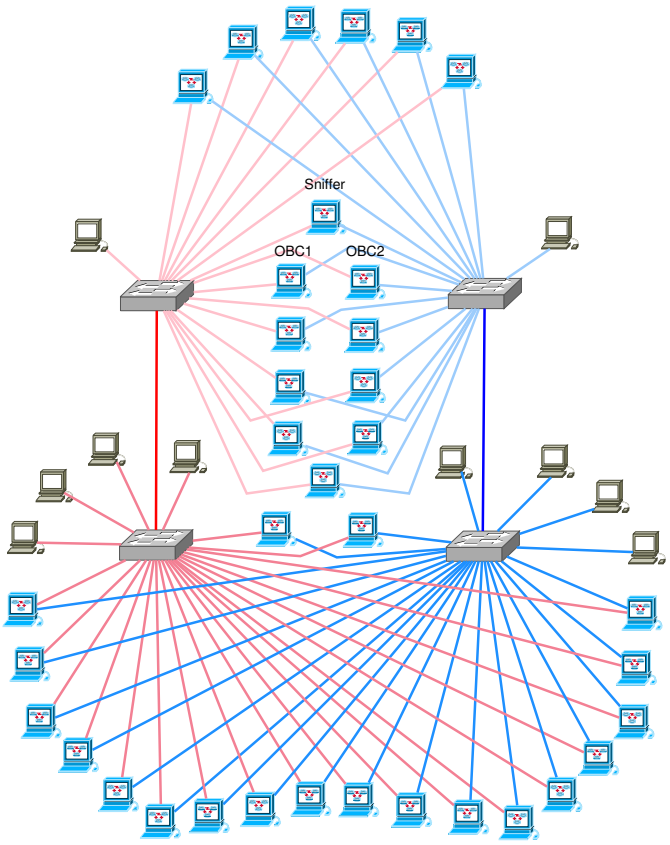


Fig. 1. Network architecture

1000 bytes. Camera's flows do not have end-to-end delay requirement while command and telemetry's flows must have an end-to-end delay $\delta_k$ lower than 1 ms.

The algorithm presented in the previous section will be used to increase commands and telemetry flows throughput/sampling. The maximum sampling that will be given corresponds to 1 kHz for each flow (i.e. $p_k = 1\ ms$). The additional traffic used to charge the network will be considered as crossed traffic to calculate the maximum end-to-end delays but its throughput will not be increased.

In our experimentations, the DiscoDNC library is used in order to calculate the end-to-end delay. However only unicast flows can be defined while we use multicast flows. Following the multicastFFA procedure presented by Bondorf and Geyer (2016), a pretreatment is done for multicast flows. Each multicast flow is transformed in $n$ unicast flows with $n$ the number of destinations. For instance, for the 44 multicast flows, each will be defined as 3 unicast flows (direction OBC1, OBC2 and sniffer). These flows are not considered as crossed traffic of each others and the delay of each one will be calculated one by one based on the related crossed traffic. Finally, instead of calculating the delay of 98 flows, we will calculate the delay of 274 unicast flows. In our case, we define the maximum end-to-end delay of a multicast flow as the maximum end-to-end delay of the unicast related flows.

We are now comparing the performance for three cases. In the two first ones, the rate is limited according to the demands requirements whereas it is computed according to our proposal in the third case. The rate $\rho$ is hence equals to:

(1) $\underline{\rho}$ the minimum rate,
(2) $\overline{\rho}$ the maximum rate,
(3) the optimal solution respecting end-to-end delays for every critical flow given by the algorithm 1.

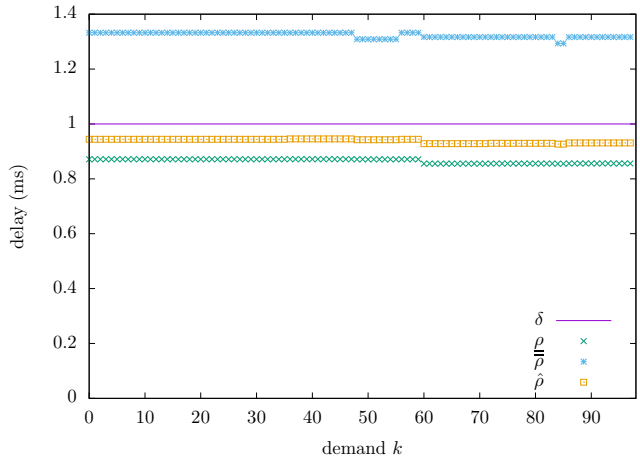


Fig. 2. Delay per flow

First, the delay is calculated and verified for each flow. The Fig. 2 represents the delay constraints of 1 ms for each command and telemetry message in the both networks and the maximum delay calculated using DiscoDNC for each flow and those with the throughput of $\underline{\rho}$, $\overline{\rho}$ and $\rho$. The worst end-to-end delay with $\underline{\rho}$ is 0,87 ms, 1.33 ms with $\overline{\rho}$ and 0.94 ms with $\rho$. From this observation, giving the maximum possible sampling (1 kHz) to flows does not allow to respect the delay requirement. Indeed, giving more sampling/throughput to one flow increases the worse end to end delay. The algorithm presented in the previous section is the solution to increase the sampling while respecting the end-to-end delay constraint. In fact, as presented on the Fig. 2, every flow has a worst end-to-end delay lower than 1 ms delay constraint.

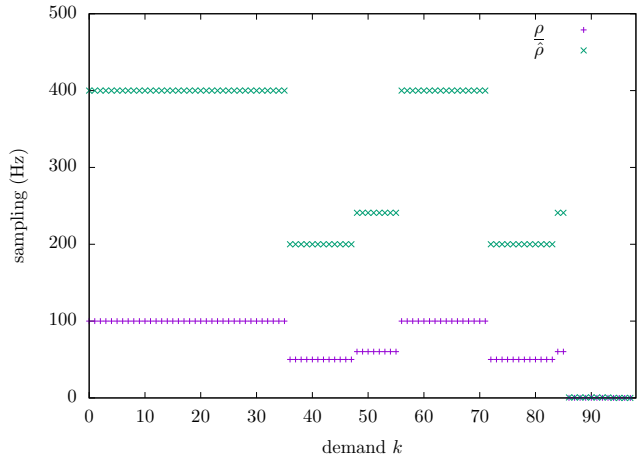


Fig. 3. Sampling per flow

Second, the value of sampling given by our solution is compared with the nominal sampling for each flow (Fig. 3). Initially, the frequency values vary between 1,7 mHz and

100 Hz. The solution proposed increases the frequency, varying now between 6,9 mHz and 400 Hz. It represents a gain of 300% for each flow. The network load with a gain of 300% is 39 Mb/s while in the nominal case, the network load is 25 Mb/s. Finally, the solution proposed allows to increase sampling and throughput of 300% while respecting the end-to-end delay for each flow and having a network load of 39 Mb/s.

## 6. CONCLUSION

This paper addresses the problem of the maximisation of network performances. In order to use available resources, maximum sampling is applied at sources (sensors and controllers) for better controllability, stability and freshness. However, giving a maximum sampling at sources does not guaranty real time constraint and network capacity respect. So this paper provides a framework for rate admission control at switches when sources oversample. The objective is to admit as much as possible throughput while respecting network capacity and maximum end to end delay requirements for each flow in a switched Ethernet network. The experiment based on the next generation of space launcher network shows that the proposed solution allows an increase of 300% of frequency sampling compared to the nominal configuration.
Future works aim at encapsulating the proposed algorithm above a Software-Defined Networking architecture in order to calculate, on line, new data and control planes in case of topology or traffic changes.

## REFERENCES


Bondorf, S. and Geyer, F. (2016). Generalizing Network Calculus Analysis to Derive Performance Guarantees for Multicast Flows.

Boyer, M., Navet, N., and Fumey, M. (2012). Experimental assessment of timing verification techniques for AFDX. In *6th European Congress on Embedded Real Time Software and Systems*. Toulouse, France.

Georges, J.P., Divoux, T., and Rondeau, E. (2011). Network Calculus: Application to switched real-time networking. In *5th International ICST Conference on Performance Evaluation Methodologies and Tools, ValueTools 2011*, CDROM. Paris, France.

Greff, F., Song, Y.Q., Ciarletta, L., and Samama, A. (2017a). Combining source and destination-tag routing to handle fault tolerance in software-defined real-time mesh networks. In *Proceedings of the 25th International Conference on Real-Time Networks and Systems, RTNS 17*. ACM Press. doi:10.1145/3139258.3139264.

Greff, F., Song, Y.Q., Ciarletta, L., and Samama, A. (2017b). A dynamic flow allocation method for the design of a software-defined real-time mesh network. In *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. IEEE. doi: 10.1109/wfcs.2017.7991949.

Guck, J.W., Reisslein, M., and Kellerer, W. (2016). Function split between delay-constrained routing and resource allocation for centrally managed QoS in industrial networks. *IEEE Transactions on Industrial Informatics*, 12(6), 2050–2061. doi:10.1109/tii.2016.2592481.

Hertiana, S.N., Hendrawan, and Kurniawan, A. (2015). A joint approach to multipath routing and rate adaptation for congestion control in OpenFlow software defined network. In *2015 1st International Conference on Wireless and Telematics (ICWT)*. IEEE. doi: 10.1109/icwt.2015.7449209.

Huang, J., He, Y., Duan, Q., Yang, Q., and Wang, W. (2014). Admission control with flow aggregation for QoS provisioning in software-defined network. In *2014 IEEE Global Communications Conference*. IEEE. doi: 10.1109/glocom.2014.7036969.

Kumar, R., Hasan, M., Padhy, S., Evchenko, K., Piramanayagam, L., Mohan, S., and Bobba, R.B. (2017). End-to-end network delay guarantees for real-time systems using SDN. In *2017 IEEE Real-Time Systems Symposium (RTSS)*. IEEE. doi: 10.1109/rtss.2017.00029.

Petit, D., Georges, J.P., Divoux, T., Regnier, B., and Miramont, P. (2016). Freshness analysis of functional sequences in launchers. *IFAC-PapersOnLine*, 49(30), 80–85. doi:10.1016/j.ifacol.2016.11.131.

Petit, D., Georges, J.P., Divoux, T., Regnier, B., and Miramont, P. (2017). A demonstrator of an ethernet based embedded network in space launchers. *IFAC-PapersOnLine*, 50(1), 16021–16026. doi: 10.1016/j.ifacol.2017.08.1914.

Petit, D., Georges, J.P., Divoux, T., Regnier, B., and Miramont, P. (2018). A strategy to implement a software defined networking controller in a space launcher. *IFAC-PapersOnLine*, 51(10), 235–240. doi: 10.1016/j.ifacol.2018.06.268.

Schmitt, J.B., Zdarsky, F.A., and Fidler, M. (2008). Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch... In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. IEEE. doi:10.1109/infocom.2008.228.