Modeling SpaceWire networks with Network Calculus [*]

Thomas Ferrandiz
Univ. de Toulouse, ISAE
France
thomas.ferrandiz@isae.fr

Fabrice Frances
Univ. de Toulouse, ISAE
France
fabrice.frances@isae.fr

Christian Fraboul
Univ. de Toulouse,
IRIT/ENSEEIHT-INPT
France
christian.fraboul@enseeiht.fr

ABSTRACT

The SpaceWire network standard is promoted by the ESA and is scheduled to be used as the sole on-board network for future satellites. This network uses a wormhole routing mechanism that can lead to packet blocking in routers and consequently to variable end-to-end delays. As the network will be shared by real-time and non real-time traffic, network designers require a tool to check that temporal constraints are verified for all the critical messages.

Network Calculus can be used for evaluating worst-case end-to-end delays. However, we first have to model SpaceWire components through the definition of service curves. In this paper, we propose a new Network Calculus element that we call the Wormhole Section. This element allows us to better model a wormhole network than the usual multiplexer and demultiplexer elements used in the context of usual Store-and-Forward networks. Then, we show how to combine Wormhole Section elements to compute the end-to-end service curve offered to a flow and illustrate its use on a industrial case study.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Performance attributes; C.3 [**SPECIAL-PURPOSE AND APPLICATION-BASED SYSTEMS**]: Real-time and embedded systems

WCTT '11 November 29 2011, Vienna, UNK, Austria

Keywords

performance evaluation, real-time networks, SpaceWire, Network Calculus

1. INTRODUCTION

SpaceWire [9], [3] is an on-board network for satellites designed by the European Space Agency and the University of Dundee. It uses high-speed serial, point-to-point links and simple routers to interconnect satellite equipment using arbitrary topologies. In the future, the ESA plans to use SpaceWire as the sole on-board network in their satellites. The idea is to use the same network for both the payload and command/control traffic. This will simplify the network architecture and reduce the costs of the satellites.

At the moment, SpaceWire provides enough bandwidth (up to 200 Mbps) to carry both types of traffic at the same time. However, in order to reduce memory size (radiation-hardened memory is very expensive), SpaceWire uses wormhole routing to transmit the data packets across the network. The downside of this technique is that it can lead to blocked packets and thus huge variations in the end-to-end delays of those packets. Furthermore, SpaceWire does not provide built-in real-time mechanisms that guarantee the timely delivery of urgent packets.

Thus, network designers need a tool to check that temporal constraints are verified for urgent packets before SpaceWire can be used to carry command/control traffic. Using simulations is not possible because covering every scenario would be very long and costly. A better solution is to design an analytical model to compute an upper-bound on the worst-case end-to-end delay of a flow.

We proposed a first model in [5] that allowed us to compute such an upper-bound for a SpaceWire network. This model works well in most cases but has some limitations. As it did not require a specific model of input traffic, it was pessimistic when the network was not fully loaded.

As a consequence, we chose to create a new model based on Network Calculus theory [8]. It is a theory designed to study deterministic queueing systems which we have already successfully used in [7] to study the AFDX network. One strong point of Network Calculus is that it allows us to model the input traffic precisely by using traffic envelopes.

However, Network Calculus has been mostly used to study Internet components and not wormhole routers. As a consequence, the usual multiplexer and demultiplexer elements are not adequate to model a wormhole network. Thus, in Section 2, we propose a new network element we call a "Wormhole Section". We can then divide the path of a flow

$\alpha_i^{S_{in}}$	arrival curve of f_i at the entrance of section S
$\alpha_i^{S_{out}}$	arrival curve of f_i at the exit of section S
α_i^j	arrival of f_i at port j
β_i^{dest}	service curve received by f_i until its destination
$(\beta)_\uparrow$	$\max(\sup_{0\leq s\leq t}\beta(s),0)$ (positive and non-decreasing upper closure)
δ_T	$\delta_T(t)=0$ if $t < T$ $\delta_T(t)=+\infty$ if $t \geq T$ for any $T\geq 0$
$h(\alpha,\beta)$	horizontal distance between the two positive, non-decreasing curves α and β

Table 1: Notations used in the paper

into a series of Wormhole Sections to deduce an end-to-end service curve.

Then, in Section 3, we show how to use this element to compute an end-to-end service curve for a flow. Finally, we illustrate the use of this model on an industrial application provided by Thales Alenia Space in 4 and conclude in Section 5.

2. THE WORMHOLE SECTION ELEMENT

A complete overview of Network Calculus would be beyond the scope of this paper. Readers not familiar with this theory can refer to [1] for a short introduction.

Here, we just recall this fundamental theorem.

THEOREM 1. *Concatenation of two systems*
Assume that a flow traverses two systems S_1 and S_2 in sequence. Assume that S_1 and S_2 offer the service curves β_1 and β_2 respectively. Then the concatenation of the two systems offers the service curve $\beta_1\otimes\beta_2$ to the flow. $\beta_1\otimes\beta_2(t)=\inf_{0\leq s\leq t}\{\beta_1(t-s)+\beta_2(s)\}$ is the Min-Plus convolution of β_1 and β_2.

This allows us to combine the network elements successively traversed by a flow to obtain an end-to-end service curve offered to the flow by the network as a whole.

2.1 Assumptions

We consider a network composed of SpaceWire routers and terminals. Each terminal has only one SpaceWire interface but can be the source and/or destination of any number of flows. Each flow f is modeled by a source, a destination, a path through the network and an arrival curve α_f. Usually, the arrival curve is a staircase function which gives a more precise model of the input traffic than an affine function.

Since SpaceWire routers use static routing, we consider only a static network. We also assume that the network is stable, i.e. that no link is required to transfer more data than its capacity. In Network Calculus terms, this can be written as follows (see [2]). For each link j, we note I_j the number of flows traversing that link and α_i^j the arrival curve of flow f_i at link j. The stability condition is now:

$$\forall j,\forall i\in\{1,\ldots,I_j\},\lim_{t\to+\infty}[\beta^j-\sum_{i=1}^{I_j}\alpha_i^j](t)=+\infty \qquad (1)$$

Throughout the paper, we will use the notations in Table 1.

2.2 A new network element

To use Network Calculus theory, we first need to determine a service curve for each element traversed by a flow. We can then compute an End-To-End (ETE) service curve using Theorem 1. However, in a wormhole network, the service offered by a router is not independent from the service offered by downstream routers. Because of the flow control, a router can output data at an average rate far inferior to its nominal capacity.

For this reason, we do not propose a classic multiplexer/ demultiplexer model of each router. Instead, we adopt a macroscopic view of the network which tries to optimize the ETE service curve of each flow while accounting for the interdependency between routers.

When a flow encounters interferences with other flows, the impact of each conflict is twofold. First, there is a delay when the flow is multiplexed with the interfering flow. Then, when the interfering flow is demultiplexed, the flow control mechanism will propagate its own delays backward to the studied flow.

To properly model this, we propose a new network element that we call a "Wormhole Section" [4]. The basic idea is to divide the path followed by a flow into a serie of sections. Each section is composed of a set of successive output ports shared by the same flows. Thus, a Section corresponds to the arrival or the departure of an interfering flow from the path of the studied flow.

Each wormhole section offers a service curve that depends on the arrival curves of the interfering flows. Once we have computed the service curve of every section in the path of a flow, we can deduce the end-to-end service curve by using Theorem 1.

Of course, once an interfering flow has left the path of the studied flow, it will go through other wormhole sections before reaching its destination. The delays caused in those sections are those that will be carried over to the studied flow.

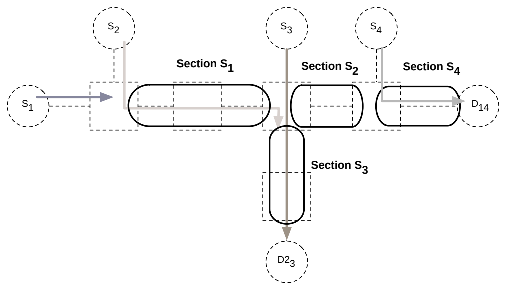

Figure 1: A network divided into wormhole sections

As an example, consider the network in Figure 1. Each flow f_i goes from its source S_i to its destination D_i. When a destination is shared by two flows i and j it is denoted $D_{i,j}$.

Let us take a closer look at f_1. f_1 has to go through three wormhole sections (S_1,S_2,S_4) to reach $D_{1,4}$ with sections S_1 and S_4 shared with other flows. S_1 and S_4 are thus the two sources of direct delays for f_1. In addition, since after leaving S_1, f_2 traverses section S_3 which it shares with f_3, S_3 will be another source of delay for f_1 but only indirectly.

As can be seen in this example, the wormhole section network element makes it easier to analyse the conflicts in a wormhole network. In the next section, we present a detailed model of this new network element.

2.3 Section sharing

We will present the model when there is only one interfering flow. We note f_1 the studied flow. f_2 is the interfering flow. f_i has $\alpha_i^{S_{in}}$ as an arrival curve at the entrance of section S. Let us assume that section S comprises M routers.

For $j = 1, \ldots, M$, β^j is the service curve offered by the output port of router R^j to the two flows. Because of the flow control mechanism, the amount of data which is transmitted by the port may actually be inferior to β^j. Thus, β^j should be seen as an intermediate parameter used to determine the service guaranteed to a given flow by this section of the network. Once we have analysed the conflicts in each output port traversed by a flow, we can use Theorem 1 to compute a service curve for the section, then for the complete path. This end-to-end curve is now valid because it takes into account the influence of all the ports used by a flow, including the indirect delays. As a consequence, it represents the real end-to-end output of the network.

Since all the routers in the section are shared between the two flows and no other interference occurs, we can view these routers as a single router with service curve $\beta^S = \bigotimes_{j=1}^{M} \beta^j$.

SpaceWire routers use a simplified Round-Robin access scheme that guarantees that each input port gets a fair acces to each output port. However, each packet can use the output port for an unlimited duration. The usual round-robin model attributes some weight to each flow and shares the bandwidth in proportion to those weights. Since the SpaceWire standard does not define such weights, we have no way of using this model and have to rely on the more pessimistic "blind multiplexing" model [8], Theorem 6.2.1.

Thus, the service curve offered by the section to f_1 is

$$\beta_1^S = (\bigotimes_{j=1}^{M} \beta^j - \alpha_2^{S_{in}})_{\uparrow}. \quad (2)$$

$(\beta)_{\uparrow}$ is the *positive and non-decreasing upper closure* of β defined as $(\beta)_{\uparrow} = \max(\sup_{0 \le s \le t} \beta(s), 0)$.

β_1^S is a worst-case service curve that implies that all the interfering flows have a higher priority than f_1 and can instantly preempt it. Of course, in reality the packets are not interrupted but this model ensures that we have a worst-case service curve for any possible scheduling of the packets.

2.4 Section demultiplexing

2.4.1 Limits of existing models

In a classic Store-And-Forward network model, the packets are instantaneously demultiplexed. Furthermore, once a packet has left the router, the delays it can endure are not propagated backward on the network. Thus, the demultiplexer has no impact on the delay (see [2] for example).

However, this is not true for a wormhole network. In fact, after two flows f_1 and f_2 have been demultiplexed, f_2 can still have an impact on f_1. This is because the flow control mechanism will carry over the delays caused to f_2 on the end of its path backward to f_1.

A possible way to model this phenomenon is given in [10]. In this paper, the authors consider the situation described on Figure 2. In this network, after they have been demultiplexed both flows f_1 and f_2 have to go through a flow

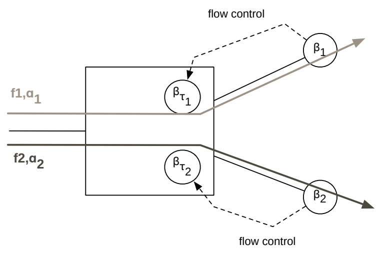

Figure 2: Demultiplexing of two flows in a wormhole network

controller. Flow controller τ_i models the impact of the flow control on the downstream output link on flow f_i with a service curve β^{τ_i}. In turn, β^{τ_i} depends on the service curve of the downstream router.

The authors consider that, as far as the aggregate flow $f_{\{1,2\}}$ is concerned, τ_1 and τ_2 are parallel traffic regulators. As a consequence, the service offered to the aggregate flow $f_{\{1,2\}}$ is $\beta_{\{1,2\}} = \min(\beta^{\tau_1}, \beta^{\tau_2})$. This aggregate service is then shared between the two flows to derive the service offered to each individual flow.

This service is valid but is is very pessimistic. Let us consider the example in Figure 2. The arrival curves are affine: $\alpha_i(t) = r_i.t + b_i$ and the service curves are linear: $\beta_i(t) = C_i.t$ We use the following values:

	f_1	f_2
r_i (Mbps)	50	10
b_i (bits)	1000	200
C_i (Mbps)	100	20

On the one hand, the service offered to $f_{\{1,2\}}$ is $\beta_{\{1,2\}}(t) = \min(C_1, C_2).t = C_2.t$. On the other hand, the arrival curve of $f_{\{1,2\}}$ is $\alpha_{\{1,2\}}(t) = \alpha_1(t) + \alpha_2(t) = r_{1,2}.t + b_{1,2}$ with $r_{1,2} = r_1 + r_2 = 60$ Mbps and $b_{1,2} = b_1 + b_2 = 1200$ bits.

The horizontal distance $h(\alpha_{\{1,2\}}, \beta_{\{1,2\}})$ between the two curves is clearly infinite and we have to conclude that the network cannot carry both flows. This is very pessimistic because it is easy to see that this network can handle both flows.

Thus we need a new, more precise network model of wormhole demultiplexer.

2.4.2 A new model of demultiplexing

Since both flows go in different directions in the network, we cannot determine an exact service for the aggregated flow. Each flow receives its own service but can still cause delays to the other flow thanks to the flow control mechanism.

The delay actually caused to an individual packet of f_1 is hard to determine because it depends on which exact conflicts occur farther on the path of f_2. However, we can determine an upper-bound on this delay.

In fact, the maximum delay caused by f_2, which we will denote d_2, is at most the maximum delivery delay from the demultiplexer to the destination of f_2. Thus,

$$d_2 = h(\alpha_2^{S_{out}}, \beta_2^{dest}) \quad (3)$$

where $\alpha_2^{S_{out}}$ is the arrival curve of f_2 at the end of S and β_2^{dest} the service curve offered to f_2 between the end of S and its destination.

Here, since we have assumed a blind multiplexing with f_2 as the higher priority flow,

$$\alpha_2^{S_{out}} = \alpha_2^{S_{in}} \oslash \bigotimes_{j=1}^{M} \beta^j \tag{4}$$

where $(\alpha \oslash \beta)(t) = \sup_{s \geq 0}\{\alpha(t+s) - \beta(s)\}$ is the Min-Plus deconvolution of α and β.

With this model, in the example in Figure 2 the delay caused by f_2 to f_1 is only $h(\alpha_2, \beta_2) = \frac{200}{10^7}s = 10\mu s$. As can be seen, this model is far less pessimistic than the model from [10].

2.4.3 Complete service curve offered by a Wormhole Section

We can combine the results from Subsection 2.3 and 2.4.2 to obtain the complete service curve offered by section S. When two flows share Section S, the service curve offered to f_1 is

$$\beta_1^S = (\bigotimes_{j=1}^{M} \beta^j - \alpha_2^{S_{in}})_\uparrow \otimes \delta_{h(\alpha_2^{S_{out}}, \beta_2^{dest})}. \tag{5}$$

When there are N flows sharing a wormhole section, the service curve for flow k is

$$\beta_k^S = (\bigotimes_{j=1}^{M} \beta^j - \sum_{i=1, i \neq k}^{N} \alpha_i^{S_{in}})_\uparrow \otimes \delta_{d_{S_k}} \tag{6}$$

with

$$d_{S_k} = \sum_{i=1, i\neq k}^{N} d_i = \sum_{i=1, i \neq k}^{N} h(\alpha_i^{S_{out}}, \beta_i^{dest}).$$

3. HOW TO COMPUTE AN END-TO-END SERVICE CURVE

3.1 Model of the terminals

The first elements in the networks are the terminals themselves. Since SpaceWire links are full-duplex, we consider that each terminal is composed of a source terminal and a destination terminal that we can model independently.

A source terminal has a FIFO input buffer that is shared by any number of applications running on the terminal. All the applications try to emit on the SpaceWire interface. For each application, we define a separate data flow. Since the flows share the output port of the terminal just like they would share the output port of a router, we consider that this output port is part of a Wormhole Section just like any routers' output port.

A destination terminal has a reception buffer large enough to receive at least one full packet of the maximum size. It can impose a constant delay to each packet before reading it. It then reads this packet at a constant service rate, which is usually less than the speed of the links.

Thus, the service curve of a destination terminal D is $\beta^D(t) = r_D.(t - T_D)^+$ where r_D is the service speed of D and T_D the delay it forces on the data.

3.2 Solving more complex interferences

The model presented in Section 2 can only be directly applied when all the interfering flows arrive and depart from the same router. When it is not the case, we could simply divide the path of the studied flow into many small wormhole sections. We could even have a section for each router crossed by the flow.

However, this would give us a suboptimal result because we would have to count several times (once for each router) the influence of a flow that shares several consecutive routers with the studied flow. It is better to try and optimize the end-to-end service curve by combining a set of flows sharing some routers as an aggregate flow. We can then determine wormhole sections for this aggregate flow and deduce the service curves offered to it. Those service curves will then be shared between the individual flows composing the aggregate flow.

To automate this process, we use the interference patterns defined in [10] to determine the order in which the residual services are computed. The authors define three interference patterns describing how one studied flow and two interfering flows interact with one another. They also show that all conflicts, involving any number of interfering flows, can be solved once those three patterns are solved.

Below we define the interference patterns and the residual service curves for the studied flow in each case.

Let us call β^j the service curve offered by router R_j. See Table 1 for the other notations.

We have $d_i = h(\alpha_i^{out}, \beta_i^{dest}), i = 2, 3$. We will explicit α_i^{out} in each case since it depends on the interference between f_2 and f_3.

3.2.1 Nested interference pattern

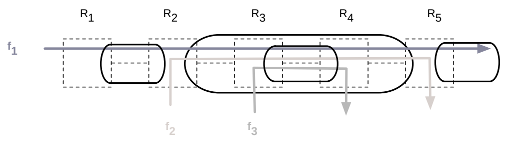

Figure 3: Nested interference pattern

In this configuration (Figure 3), we first treat f_1 and f_2 as an aggregate flow sharing the section comprising the output port of R_3 with f_3. Then, we consider that f_1 and f_2 share the section comprising the output ports of R_2, R_3 and R_4.

The service curve offered to f_1 by the section $R_1 \to R_5$ is thus

$$\begin{aligned}\beta_1^{1\to5} =& \beta^1 \otimes ([\beta^2 \otimes (\beta^3 - \alpha_3^3)_\uparrow \otimes \delta_{d_3} \otimes \beta^4] - \alpha_2^2)_\uparrow \\ & \otimes \delta_{d_2} \otimes \beta^5\end{aligned} \tag{7}$$

Here, $\alpha_2^{out} = \alpha_2^2 \oslash [\beta^2 \otimes (\beta^3 - \alpha_3^3)_\uparrow \otimes \delta_{d_3} \otimes \beta^4]$ and $\alpha_3^{out} = \alpha_3^3 \oslash [(\beta^3 - \alpha_2)_\uparrow \otimes \delta_2]$

3.2.2 Parallel interference pattern

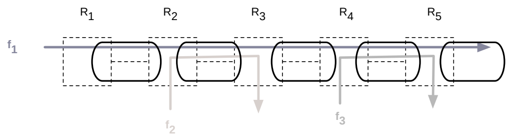

Figure 4: Parallel interference pattern

In this configuration (Figure 4), f_1 shares the section comprising the output port of R_2 with f_2 and the section comprising the output port of R_4 with f_3. The other sections are not shared.

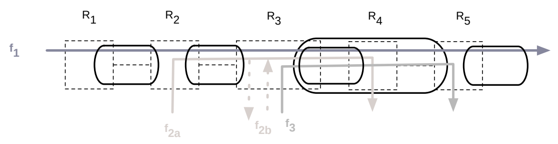

Figure 5: Crossed interfering pattern

The service curve offered to f_1 by the section $R_1 \to R_5$ is

$$\begin{aligned}\beta_1^{1\to5} =& \beta^1 \otimes (\beta^2 - \alpha_2^2)_\uparrow \otimes \delta_{d_2} \\ & \otimes \beta^3 \otimes (\beta^4 - \alpha_3^4)_\uparrow \otimes \delta_{d_3} \otimes \beta^5\end{aligned} \tag{8}$$

Here, $\alpha_2^{out} = \alpha_2^2 \otimes \beta^2$ and $\alpha_3^{out} = \alpha_3^4 \otimes \beta^4$.

3.2.3 *Crossed interfering pattern*

In this configuration (Figure 5), we have to split f_2 into two sub-flows f_{2a} and f_{2b} at R_3. First, f_1 shares the section comprising the output port of R_2 with f_{2a}. Then, the aggregate flow $f_1 + f_3$ shares the section composed of the output port of R_3 with f_{2b}. Finally, f_1 shares the section comprising the output port of R_3 andR_4 with f_3.

In the end, the service curve offered to f_1 by the section $R_1 \to R_5$ is

$$\begin{aligned}\beta_1^{1\to5} =& \beta^1 \otimes (\beta^2 - \alpha_2^2)_\uparrow \\ & \otimes ((\beta^3 - \alpha_2^3)_\uparrow \otimes \delta_{d_2} \otimes \beta^4 - \alpha_3^3)_\uparrow \otimes \delta_{d_3}\end{aligned} \tag{9}$$

Here, $\alpha_2^{out} = \alpha_2^3 \oslash [(\beta^3 - \alpha_3^3)_\uparrow \otimes \delta_{d_3}]$, $\alpha_2^3 = \alpha_2^2 \oslash \beta^2$, $\alpha_3^{out} = \alpha_3^3 \oslash [(\beta^3 - \alpha_2^3)_\uparrow \otimes \delta_{d_2} \otimes \beta^4]$.

3.3 Computing the arrival curves of the interfering flows

As seen above, we need to compute the arrival curves of all the interfering flows both at the beginning of the wormhole section where they meet the studied flow and at the end of this section.

For each interfering flow, the arrival curve at the beginning of the section depends on the arrival curve of the flow at the source and on the service received between the source and the beginning of the shared Wormhole Section. The arrival curve at the end of the section can then be computed from the arrival curve at the beginning of the section.

Therefore, the service received by the interfering flow is itself dependent on other interfering flows. Furthermore, since the service curve β^{dest} depends on the arrival curve of other flows which may in turn depend on the studied flow, we risk being stuck with a circular-dependency problem.

3.4 Fixed-point method

To solve this problem, we use a fixed-point method. All the arrival curves at the sources and all the service curves of the output ports are known. In addition, the residual service curve of a port can be deduced immediately from the knowledge of the arrival curves of the conflicting flows at this port. Thus, we only have to determine the arrival curves at each output port to solve the problem.

We proceed iteratively. Let α_i be the arrival curve of flow f_i at its source and α_i^j be the arrival curve at port j. We start with $\alpha_i^{j,(0)} = \alpha_i$ for all j. This is equivalent to assuming that the burstiness of a flow does not increase as it traverses the network. Of course, this is an approximation. Then, we express each $\alpha_i^{j,(n+1)}$ as a function of α_i, of the service curves β^j and of any number of arrival curves $\alpha_k^{l,(n)}$ from the previous step. With each iteration, the computed arrival curves $\alpha_i^{j,(n)}$ increase, until they reach their real value at every point in the network.

At some point, there exists p such that $\forall i, \forall j, \alpha_i^{j,(p+1)} = \alpha_i^{j,(p)}$ and the computation has converged toward a solution for the system.

We have implemented and tested this method on several configurations. In each case, the computation converges in a few steps toward a solution, provided that the stability condition is respected (see 2.1).

4. INDUSTRIAL APPLICATIONS

We will now present the results given by our model for an industrial configuration. This study was based on a network architecture provided by Thales Alenia Space (see Figure 6) and designed for use in an observation satellite. Our goal was to determine a worst-case end-to-end delay for each flow in the network.

4.1 Description of the network

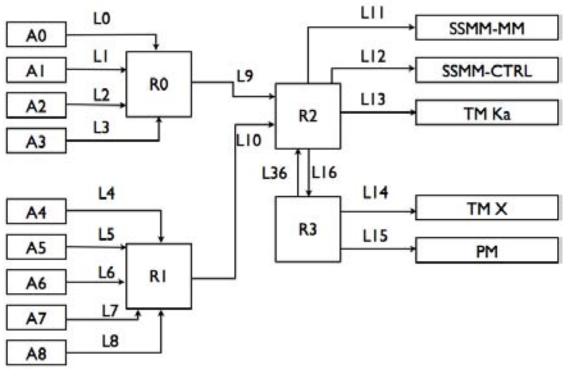

Figure 6: Network of the industrial application

As can be seen in Figure 6, the network is composed of two parts. The platform equipment on the right which includes a mass memory unit (SSMM-MM and SSMM-CTRL), a processor module (PM) and two Transmission Modules (TM Ka and TM X). The processor module monitors the other nodes and sends back commands. The mass memory is split in two parts. Data is stored in the MM module but must go through the controller unit first. Thus, other nodes send packets to the CTRL unit. This unit then processes the data during a constant delay and sends the packet to the MM unit. The CTRL unit only stores one packet at any given time. On the left, the application terminals (A_0 to A_8) represent the payload instruments. These include cameras and any kind of sensors. They send data packets to the CTRL unit and monitoring traffic to the PM unit.

All the links have the same capacity $C = 50\ Mbps$ except L_{13} and L_{14} which have a capacity $C_{slow} = 20Mbps$.

We can split the network traffic into four categories (see Table 2). The table gives the network path and the packet size for each type of traffic. Among all the nodes, only the CTRL and PM units introduce a delay for every packet they receive. The delay is the same for both: $D_{PM} = D_{CTRL} = 10\mu s$.

Furthermore, the PM reads data packets at 1250 bytes/s and the two TM units at 87.5 kbytes/s.

Traffic type	Path	Packet size (bytes)
SC (scientific)	$A_i \rightarrow CTRL \rightarrow MM$	4000
MON	$A_i \rightarrow PM \rightarrow CTRL \rightarrow MM$	20
(MONitoring)	$PM \rightarrow CTRL \rightarrow MM$	100
CMD	$PM \rightarrow CTRL$	1000
(Command)	$PM \rightarrow A_i$	1000
TM	$MM \rightarrow TM - KA$	4000
(TeleMetry)	$MM \rightarrow TM - X$	4000

Table 2: Network traffic

Some traffic is further divided into several flows according to the definition we used in Section 2.1. In that case, we simply take the summation of the delays for each included flow.

We implemented the model using MATLAB and the RTC Toolbox [11]. This toolbox implements all the common operations of Network Calculus like the min-plus convolution and deconvolution.

Our software takes a description of the network and of the network traffic as input and gives an upper-bound on the end-to-end delay of each flow as output. It implements the fixed-point method and all the computations based on the interference patterns.

4.2 Comparison with simulation results

To estimate the tightness of the bounds we computed, we compared them to the result of the simulations done by Thales Alenia Space on this network. Those simulations were implemented on Thales Alenia Space's MOST simulator [6] that completely models the low-level behavior of SpaceWire.

The results are given in Table 3.

The critical traffic includes the CMD, TM and MON flows. The non-critical traffic includes all the SC flows.

As can be seen, the maximum delays observed during the simulations and the upper-bound computed using Network Calculus are in the same order of magnitude. This shows that our methods is not too pessimistic and gives exploitable results for a network designer.

It is also worth noting that we do not know if the maximum observed delay is the worst-case delay or not. The real worst-case delay may be higher and, thus, closer to the bound we computed because worst-case delays are extremely rare events that are hard to observe with simulations.

For the non-critical traffic, the bound is less tight but it is not a problem fot this type of traffic. The only thing we are interested in for non-critical traffic is whether the bound is finite. If it is, it shows that the network is able to carry all the traffic in a finite time. This knowledge is sufficient for non-critical traffic.

Traffic type	MOST	NetCal
non-critical	16.6	102
critical	145	439

Table 3: Comparison between MOST results and our model (all results are in ms)

5. CONCLUSION

In this paper, we have defined a new Network Calculus element that can be used to model a SpaceWire network more accurately than the usual multiplexer and demultiplexer elements. We call this new element a Wormhole Section since it represents a part of the path followed by a flow. The Wormhole Section should allow us to obtain tighter bounds by considering successive routers as only one element.

Furthermore, we also described a new way to compute the delay caused by a flow leaving a wormhole section. We showed on a simple example that our method is a lot less pessimistic than existing demultiplexer models for a wormhole network.

Our model is based on the Network Calculus theory and uses the interference patterns defined in [10] by Qian *et al* to model complex dependencies with the interfering flows. Furthermore, it uses a fixed-point computation to solve circular dependencies between the arrival curves of the various flows.

We evaluated our model on an industrial configuration in Section 4. On this example, we showed that the bounds obtained by our method are close to the maximum delay observed during the simulations.

In the future, we plan to pursue two objectives. The first goal is to formally prove the convergence of the fixed-point method use in Section 3. The second will be to find a more realistic model of the non-preemptive section sharing. This should allow us to tighten the upper-bounds even more.

Acknowledgment

This work was supported by a PhD grant from the CNES and Thales Alenia Space.

6. REFERENCES

[1] J. L. Boudec and P. Thiran. A short tutorial on network calculus. i. fundamental bounds in communication networks. *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, 4, 2000.

[2] R. L. Cruz. A calculus for network delay, part i: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1), Jan 1991.

[3] ECSS. Spacewire – links, nodes, routers and networks. Aug 2008.

[4] T. Ferrandiz, F. Frances, and C. Fraboul. Using network calculus to compute worst-case end-to-end delays in spacewire networks. *ECRTS 11 Work-in-Progress session.*

[5] T. Ferrandiz, F. Frances, and C. Fraboul. A method of computation for worst-case delay analysis on spacewire networks. *Industrial Embedded Systems, 2009. SIES '09. IEEE International Symposium on*, 2009.

[6] P. Fourtier, A. Girard, A. Provost-Grellier, and F. Sauvage. Simulation of a spacewire network. *Proceedings of the International SpaceWire Conference 2010*, Oct 2010.

[7] F. Frances and C. Fraboul. Using network calculus to optimize the afdx network. *ERTS 2006 : 3rd European Congress ERTS Embedded real-time software*, Jan 2006.

[8] J. LeBoudec and P. Thiran. Network calculus a theory of deterministic queuing systems for the internet. *Springer Verlag*, (LNCS 2050), Apr 2004.

[9] S. M. Parkes and P. Armbruster. Spacewire: A spacecraft onboard network for real-time communications. *IEEE-NPSS Real Time Conference*, (14), Feb 2005.

[10] Y. Qian, Z. Lu, and W. Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip-Volume 00*, 2009.

[11] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`.