Abstract—Discrete Event Dynamic Systems (DEDS) are systems whose state transitions are triggered by events that occur at discrete instants of time. The communication networks are examples of this kind of systems. The mathematical constraints of some DEDS can be described more adequately using the Max-Plus algebra. Previous works show that the problem of determining performance bounds for communication networks is simplified if modeled using this algebra. The compilation of existing rules and results on this field is called Network Calculus. The goal of this article is to improve a systematic use of the Max-Plus algebra in the formulation and derivation of results on Network Calculus. To illustrate the introduced methodology, we analyze a window flow controller, a system that controls the traffic admitted by a network in order to limit its backlog in a specified manner.

Index Terms—DEDS Modeling, Diod Algebra, Max-Plus Systems, Network Calculus, Quality of Service, Service Curves.

I. INTRODUCTION

The evolution of technology brought about highly complex dynamic asynchronous systems whose state transitions are triggered by events that occur at discrete instants of time. These systems are called Discrete Event Dynamic Systems (DEDS). DEDS are not appropriately described by the mathematical arsenal centered on differential equations. For such systems, there are specific techniques that are more adequate, as the ones presented in [1]. Typical examples of DEDS are: computer systems, flexible manufacturing systems, telecommunication networks, traffic control systems, multiprocessor operating systems and logistic systems. The number of packets inside a communication network and the number of pieces in a manufacturing system, for instance, are examples of state variables that change at events like “arrival of item” or “departure of item”, where an “item”, in these cases, are packets or pieces.

DEDS lead to a nonlinear description in the usual algebra of real numbers, although there exists a subclass of DEDS that can be linearly modeled under the Max-Plus algebra.1 The characteristics of this algebra seem to be more adequate for treating such DEDS. Chang and colleagues [2], [3], for example, showed that the calculations of performance bounds on communication networks are simplified if the mathematical constraints of the systems are represented using the Max-Plus algebra. However, the potential of this algebra for solving such problems is still unexplored. The goal of this article is to improve a systematic use of the Max-Plus algebra in the formulation and derivation of NC results. In particular, a systematic use of the residuation operation is introduced.

Networks that support Integrated Services (IntServ) or Differentiated Services (DiffServ) can carry traffic from a wide variety of applications such as video-on-demand (VoD), videoconference and voice over IP (VoIP). Each application demands different Quality of Service (QoS) requirements, such as maximum end-to-end delay and jitter and minimum transmission rate. This scenario of traffic imposes the determination of QoS parameters bounds and traffic shaping mechanisms, in order to avoid the monopolization of the network resources by one traffic flow. As a consequence, the degradation of the QoS seen by all other traffic flows is avoided.

On this context, the Network Calculus (NC) appeared as a set of rules and results that can be used to compute bounds for QoS parameters of communication networks [4]. The theory is mainly presented in [3], [5], [6] and a short introduction is found in [7], [8]. NC is based on the idea that a detailed analysis of traffic flows is not required in order to specify a network performance in terms of service requirements. Originally, the NC theory and the concepts involved were described using the conventional algebra of real numbers but soon the adequacy of the Max-Plus algebra started to be evidenced.

The algebraic approach to NC based on the Max-Plus algebra herein proposed, simplifies and systemizes the task of modeling a typical network system. The analysis is based on two new concepts: basic constraints and basic elements. The article is divided as follows. Section II introduces the proposed methodology. Section III presents some important definitions and results on the Max-Plus algebra. Section IV presents the concepts of basic constraints and basic elements and characterize mathematically the basic elements. Finally, Section V illustrates the introduced methodology by the analysis of a greedy shaper and a window flow controller. In this section, decision procedures concerning these systems are proposed. The results obtained are in agreement with [8] and [9], for a time-invariant scenario of traffic and a constant

1The idea of “mathemathical constraints” in network modeling is not new (see Chang et al. [3]). However, in this work it is presented under a systematic Max-Plus framework, allowing the definition of basic constraints.

2A window flow controller is a system that controls the traffic admitted by a network in order to guarantee that the number of packets inside it never exceeds a predefined value, called the “window size”. 
II. METHODOLOGY

The objective of this article is to consider a new algebraic methodology to manipulate problems related to network performance. The elements manipulated are positive functions of the type \( x : (\mathbb{R}^+)^2 \rightarrow \mathbb{R}^+ \cup \{+\infty\} \). As Cruz [10], we consider a discrete time model, where a discrete function \( x(s, t) \) denotes the number of packets that arrive at (or leave) a given network element during the interval \((s, t]\). Function \( x \), therefore, must be a non-decreasing function in \( t \). We are interested on the determination of lower and upper bounds for functions \( x \). The adopted strategy is to present an algebraic structure over which functions \( x \) are manipulated. Based on this structure, the constraints of a network are described as a composition of basic elements (defined in Section IV). Therefore, the proposed methodology is modular.

In this approach, the problem of obtaining bounds for network performance is treated as follows:

1) the particular problem is described as a composition of basic elements;
2) the algebraic constraints of each basic element of 1) are written;
3) the system of equations obtained in 2) is algebraically solved;
4) the results obtained are instantiated on the context of the particular network problem.

For the sake of clarity, we illustrate this methodology by analyzing a greedy shaper (a traffic regulator) and a window flow controller and show that these particular problems can be algebraically solved in a simple manner.

III. NETWORK CALCULUS AND THE MAX-PLUS ALGEBRA

In this section, we present definitions and results of the Max-Plus algebra that are necessary to understand the further analysis. Due to the lack of space, proofs of some theorems and propositions are omitted. Systems are considered empty at \( s = t = 0 \).

Definition 1 (Dioid and complete dioid): ([11]) A dioid \( \mathcal{D} \) is a set \( \mathcal{D} \) endowed with two closed operations called “addition” (\( \oplus \)) and “multiplication” (\( \odot \)). The usual notation is \((\mathcal{D}, \oplus, \odot)\). The operations of a dioid must satisfy the following axioms: associativity of addition and multiplication, commutativity of addition, distributivity of multiplication over finite sums and idempotency (i.e., \( \forall a \in \mathcal{D}, \ a \oplus a = a \)). Besides, there must exist neutral elements for both operations, where \( e \) is the unity element and \( \varepsilon \) is the null element, that must be absorbing (i.e., \( \forall a \in \mathcal{D}, \ a \odot e = a \odot \varepsilon = e \)). If the dioid is closed for infinite sums, and distributivity applies for infinite sums, the dioid is called complete.

Table I illustrates the axioms of a complete dioid.

Remark 1 (Order in a complete dioid): ([11]) Let \( a, b \in \mathcal{D} \) be two elements of a complete dioid \( \mathcal{D} \). A partial order in \( \mathcal{D} \) can be defined by the following way:

\[ a \succ b \iff a = a \oplus b. \]

\[ a \succ b \iff a \equiv b \]

\[ a \equiv b \iff a = a \oplus b. \]

Table I

| \( a, b \in \mathcal{D} \) and \( a_k \in \mathcal{D}, k \in \mathbb{N} \): |
|-----------------|-----------------|
| \( a \oplus b = b \oplus a \) | \( a \oplus a = a \) |
| \( (a \oplus b) \odot c = a \oplus (b \odot c) \) | \( (a \oplus b) \odot c = a \odot (b \odot c) \) |
| \( \bigodot_{k \in \mathbb{N}} a_k = \bigoplus_{k \in \mathbb{N}} (a_k \odot c) \) | \( c \odot \bigoplus_{k \in \mathbb{N}} a_k = \bigoplus_{k \in \mathbb{N}} (c \odot a_k) \) |
| \( \epsilon \odot a = a \odot \epsilon = a \) | \( \epsilon \odot a = a \odot \epsilon = a \) |

In particular, define \( \top \) as the sum of all elements in \( \mathcal{D} \). \( \top \) is, thus, the maximum element of dioid \( \mathcal{D} \).

Theorem 1: (Analogous to Theorem 4.75 in [11]) Given \( a \) and \( b \) two elements of a complete dioid \( \mathcal{D} \), the minimum solution of equation \( x = xa \oplus b \) is given by \( x = b \odot a^* \), where \( a^* = \bigoplus_{i \in \mathbb{N}} a^i, a^0 = e \), and \( a^i = a \odot \ldots \odot a \) (i times) (Kleene closure).

Definition 2 (Residuated function and residual): ([11]) Let \( f : \mathcal{A} \rightarrow \mathcal{B} \) be an isometric mapping from the complete dioid \( \mathcal{A} \) into the complete dioid \( \mathcal{B} \). If, for all \( b \in \mathcal{B} \), the set of solutions of \( f(x) \leq b \) is non-empty and has a maximum element, denoted \( f^\#(b) \) (unique), then \( f \) is called a residuated function and \( f^\#(b) \) is called the residual of \( f \) at \( b \).

Define functions \( l_a(x) = a \odot x \) and \( r_a(x) = x \odot a \). It can be proved that, for any dioid, \( l_a(x) \) and \( r_a(x) \) are residuated. The residual functions are denoted, respectively, \( l_a^\#(y) = a \odot y \) and \( r_a^\#(y) = y \odot a \). Table II presents some useful properties of the residuation operation.

Theorem 2: (Analogous to Theorem 4.73 in [11]) Given \( a, b \in \mathcal{D} \), define \( (a \land b) \in \mathcal{D} \) as the greatest element of \( \mathcal{D} \) that is smaller than or equal to \( a \) and \( b \). Then, the maximum solution of equation \( x = (x \land b) \land b \) is given by \( x = b \land a \).

Consider, now, the set \( \mathcal{C} \) of functions of the type \( x : (\mathbb{R}^+)^2 \rightarrow \mathbb{R}^+ \cup \{+\infty\} \),

\[ \mathcal{C} = \{x : x(s, t) \geq 0, \text{ if } s \leq t; x(s, t) = 0, \text{ if } s > t\}. \] (1)

Given \( a, b \in \mathcal{C} \), the following operations are defined in \( \mathcal{C} \):

\[ (a \oplus b)(s, t) = \min \{a(s, t), b(s, t)\}, \] (2)

\[ (a \odot b)(s, t) = \begin{cases} \min_{s \leq \tau \leq t} \{a(s, \tau) + b(\tau, t)\}, & \text{if } s \leq t, \\ 0, & \text{if } s > t. \end{cases} \] (3)

Theorem 3: \( \mathcal{C} = (\mathcal{C}, \oplus, \odot) \) is a complete dioid, where \( \mathcal{C} \), \( \oplus \) and \( \odot \) are defined by (1)-(3), and the zero and unit

| TABLE I
| AXIOMS OF A COMPLETE DIOID AND THE ORDER DEFINITION |
|-----------------|-----------------|
| \( \forall a, b \in \mathcal{D} \) and \( a_k \in \mathcal{D}, k \in \mathbb{N} \): |
| \( a \oplus b = b \oplus a \) | \( a \oplus a = a \) |
| \( (a \oplus b) \odot c = a \oplus (b \odot c) \) | \( (a \oplus b) \odot c = a \odot (b \odot c) \) |
| \( \bigodot_{k \in \mathbb{N}} a_k = \bigoplus_{k \in \mathbb{N}} (a_k \odot c) \) | \( c \odot \bigoplus_{k \in \mathbb{N}} a_k = \bigoplus_{k \in \mathbb{N}} (c \odot a_k) \) |
| \( \epsilon \odot a = a \odot \epsilon = a \) | \( \epsilon \odot a = a \odot \epsilon = a \) |

| TABLE II
| PROPERTIES OF THE RESIDUATION |
|-----------------|-----------------|
| \( ax \equiv b \iff x \equiv a \equiv b \) | \( xa \equiv b \iff x \equiv b \equiv a \) (P1) |
| \( b(a \equiv b) \equiv x \equiv (a \equiv b) \equiv x \) | \( (x \equiv a) \equiv b \equiv (x \equiv b \equiv a) \) (P2) |

In particular, define \( \top \) as the sum of all elements in \( \mathcal{D} \). \( \top \) is, thus, the maximum element of dioid \( \mathcal{D} \).
elements are defined as follows:
\[ \varepsilon(s, t) = \begin{cases} +\infty, & \text{if } s \leq t, \\ 0, & \text{if } s > t, \end{cases} \]
and
\[ e(s, t) = \begin{cases} +\infty, & \text{if } s < t, \\ 0, & \text{if } s \geq t. \end{cases} \]

For this dioid, \( \top(s, t) = 0 \), \( \forall(s, t) \) and \( (a \land b)(s, t) = \max \{a(s, t), b(s, t)\} \).

In Section II, we mentioned that a function \( x(s, t) \) that represents the number of packets that arrive at or leave a given network element during the interval \( [s, t] \) must be a non-decreasing function in \( t \). Thus, it is natural that if we seek bounds for \( x \), they must be non-decreasing functions as well. By the other side, the non-decreasing property is not required for general elements in \( C \) and, for this reason, the result of a multiplication in \( C \) can be a function with decreasing portions, even when the terms are non-decreasing functions. Let \( C \) be the set of non-decreasing functions in \( C \): \( C_t = \{ x \in C : x(s, t_1) \geq x(s, t_2), \text{if } t_1 > t_2 \} \). Propositions 4 and 5 below show that, under certain conditions, two functions \( a, b \in C_t \) lead to a non-decreasing function \( (a \otimes b) \).

**Proposition 4**: Let \( a, b \in C_t \) and \( b(s, s) = 0 \), \( \forall s \). Then \( (a \otimes b) \in C_t \).

**Definition 3** (Vertical displacement function): ([3]) For a given function \( B : \mathbb{R}^+ \to \mathbb{R}^+ \), define the vertical displacement function, \( \lambda_B \in C \), as follows:
\[
\lambda_B(s, t) = \begin{cases} 0, & s > t, \\ B(t), & s = t, \\ +\infty, & s < t. \end{cases}
\]

Function \( \lambda_B \) has some interesting properties as shown by Table III.

**Proposition 5**: For all \( a \in C_t \) and \( B(t) \in \mathbb{R}^+ \) non-decreasing in \( t \), then \( (a \otimes \lambda_B) \in C_t \) and \( (\lambda_B \otimes a) \in C_t \).

**Definition 4** (Delay Function): For a given number \( D \in \mathbb{R}^+ \), define \( \delta_D \in C \) by the following way:
\[
\delta_D(s, t) = \begin{cases} 0, & t - s \leq D, \\ +\infty, & \text{otherwise}. \end{cases}
\]

This function can be used as a delay function, as shown by Table IV.

**TABLE III**
**Special function**: \( \lambda_B \in C \)

| \( (x \otimes \lambda_B)(s, t) = x(s, t) + B(t) \) |
| \( (\lambda_B \otimes x)(s, t) = x(s, t) + B(s) \) |

**TABLE IV**
**Special function**: \( \delta_D \in C \)

For all \( x \in C \) such that \( x(s, s) = 0 \), \( \forall s \):
\[
(x \otimes \delta_D)(s, t) = x(s, t - D), \text{if } x \text{ is non-decreasing in } t \\
(\delta_D \otimes x)(s, t) = x(s + D, t), \text{if } x \text{ is non-increasing in } s
\]

**TABLE V**
**Defined operations in \( C \)**

| \( (a \land b)(s, t) = \min \{a(s, t), b(s, t)\} \) |
| \( (a \lor b)(s, t) = \begin{cases} \min \{a(s, t) + b(s, t)\}, & \text{if } s \leq t, \\ 0, & \text{otherwise}. \end{cases} \) |
| \( (a \land y)(s, t) = \max \{y(s, t) - a(s, t)\}, \text{if } s \leq t, \) otherwise. |
| \( (y \land a)(s, t) = \max \{y(s, t) - a(t, t)\}, \text{if } s \leq t, \) otherwise. |
| \( (a \land b)(s, t) = \max \{a(s, t), b(s, t)\} \) |

**Proposition 6**: For dioid \( C \), if \( s \leq t \):
\[
(a \land y)(s, t) = \max \{y(s, t) - a(s, t)\}, \\
(y \land a)(s, t) = \max \{y(s, t) - a(t, t)\};
\]
and \( (a \land y)(s, t) = (y \land a)(s, t) = 0 \), otherwise. Table V shows the definitions of the operations used in this article.

As previously mentioned, \( x(s, t) \) represents the number of packets that arrive at or leave a network element during the interval \( [s, t] \). In many cases, we are interested in the evaluation of \( x \) just in the interval \( [0, t] \). In these cases, it is convenient to associate \( x(0, t) \) to the function \( \tilde{x} \in C \) defined below, by Definition 5.

**Definition 5** (Function \( \tilde{x} \in C \)): Define function \( \tilde{x} \in C \) by the following manner:
\[
\tilde{x}(s, t) = \begin{cases} x(0, t), & \text{if } s = 0, \\ \varepsilon(s, t), & \text{if } s \neq 0. \end{cases}
\]

**Remark 2**: Symbols \( \succ \) and \( \preceq \) represent the partial order in \( C \) and correspond, respectively, to \( \preceq \)(\( s \), \( t \)) and \( \succeq \)(\( s \), \( t \)), in the usual algebra (notice the inversion). Table VI shows this result.

**TABLE VI**
**Equivalent constraints**

| \( \forall s, t \geq 0 : y(s, t) \leq x(s, t) \) | \( y \succeq x \) | \( y = y \oplus x \) |
| \( \forall t \geq 0 : y(0, t) \leq x(0, t) \) | \( \bar{y} \succeq \bar{x} \) | \( \bar{y} = \bar{y} \oplus \bar{x} \) |

**Proposition 7**: \( \forall x \in C \), \( (x \sigma) = \tilde{x} \sigma \).

**IV. Basic network constraints**

This paper concerns the determination of QoS parameters such as end-to-end delay, maximum backlog and maximum throughput. Mathematical constraints can be associated to these parameters, expressing performance specifications as well as physical limitations of the system. Further, we describe four basic types of constraints: flux constraints, regulation constraints, service constraints and backlog constraints. We do not claim the completeness of this set of constraints to solve all network problems. However, it covers a large set of fundamental problems related to the QoS...
parameters described above and for this reason we call it the basic set of constraints.

The flux constraint guarantees that the quantity of data that leaves a system, \( y(0,t) \), is never greater than the one that enters in it, \( x(0,t) \). Mathematically, \( y(0,t) \leq x(0,t), \forall t \). From Definition 5 and Table VI, this equation can be written in \( \mathcal{C} \) by the following way: \( \tilde{y} = \tilde{y} + \tilde{x} \).

The regulation constraint shapes the output flow by limiting it. Mathematically:

\[
y(0,t) \leq \min_{0 \leq s \leq t} \{ y(0,s) + \sigma(s,t) \},
\]

where \( \sigma \in \mathcal{C} \) is called the regulation curve of the system. In \( \mathcal{C} \), from Table VI and Prop.7, this equation can be written as

\[
\tilde{y} = \tilde{y} + \tilde{y} \sigma.
\]

**Proposition 8:** For all \( 0 \leq s \leq t \), if \( y(s,t) \) is such that \( y(s,t) = y(0,t) - y(0,s) \) then \( y(s,t) \leq \sigma(s,t) \). In other words, \( \sigma(s,t) \) is the maximum quantity of data that can leave the system during the interval \( (s,t) \).

**Proof:** From (4), \( y(0,t) \leq \min_{0 \leq s \leq t} \{ y(0,s) + \sigma(s,t) \} \leq y(0,s) + \sigma(s,t), \forall s \leq t \). Therefore, \( \forall s \leq t, y(0,t) - y(0,s) \leq \sigma(s,t) \) and equivalently, \( y(s,t) \leq \sigma(s,t) \).

The service constraint is related to the minimum quantity of data served by a network element. Mathematically, it is expressed by the following way:

\[
y(0,t) \geq \min_{0 \leq s \leq t} \{ x(0,s) + \beta(s,t) \}
\]

or, in \( \mathcal{C} \),

\[
\tilde{x} + \tilde{x} \beta = \tilde{x} + \tilde{y},
\]

where \( \beta \in \mathcal{C} \) is called the service curve or the minimum service function of the system.

**Proposition 9:** If \( y(s,t) \) is such that \( y(s,t) = y(0,t) - y(0,s) \), then there always exists an \( s \leq t \) such that \( y(s,t) \geq \beta(s,t) \). It is guaranteed, then, that a quantity of data of at least \( \beta(s,t) \) leaves the system during the interval \( (s,t) \).

**Proof:** From (6),

\[
y(0,t) \geq \min_{0 \leq s \leq t} \{ x(0,s) + \beta(s,t) \}
\]

\[
\geq \min_{0 \leq s \leq t} \{ y(0,s) + \beta(s,t) \},
\]

because of the flux constraint: \( y(0,s) \leq x(0,s), \forall s \). Therefore, there exists at least one \( s \leq t \), such that \( y(0,t) \geq y(0,s) + \beta(s,t) \) or, equivalently, there exists an \( s \leq t \), such that \( y(s,t) \geq \beta(s,t) \).

Finally, the backlog constraint represents limitations on buffer capacity. Define the backlog of a system at instant \( t \) as the vertical deviation between the input and output curves at instant \( t \), i.e., \( x(0,t) - y(0,t) \). We say that a given queue has maximum backlog capacity \( B(t) \geq 0 \), if the following inequality is satisfied, \( \forall t \) : \( x(0,t) - y(0,t) \leq B(t) \). From Table III, this inequality can be rewritten by the following way: \( x(0,t) \leq (y + \lambda B)(0,t) \) or, in \( \mathcal{C} \),

\[
\tilde{x} = \tilde{x} + \tilde{y} \lambda B.
\]

For practical purposes, we define pictorial elements based on the mathematical constraints described before (the flux constraint is always present for causal systems): the \( \sigma \)-regulator, the \( \beta \)-network and the \( B \)-queue. These definitions allow the modular description of more complex network systems, as will be exemplified in Section V. Table VII presents the systems of equations in \( \mathcal{C} \) that characterize each basic element. We consider that the basic elements have infinite buffer capacity, so that the results obtained by this methodology represent a limit when losses are avoided.

The \( \sigma \)-regulators represent elements that delay the input traffic when necessary in order to satisfy (5). The \( \beta \)-networks represent systems that have some guarantee of minimum service. This guarantee is represented by (7). Finally, the \( B \)-queues represent systems whose maximum backlog is limited. In this case, (8) must be satisfied.

**Remark 2:** The arrows on the graphical representations of the basic elements indicate the direction of the flux constraints and not necessarily the physical path of packets.

## V. Examples

In this section, we illustrate the methodology presented in Section II by the analysis of two systems: a greedy shaper and a window flow controller.

The greedy shaper is a \( \sigma \)-regulator that outputs data whenever the backlog is non-null, i.e., it does not insert delay unless it is necessary in order to satisfy the regulation constraint, (5). In Section V-C, we analyze the effect of using a greedy shaper as a window flow controller.

### A. The greedy shaper

Let \( a, x \in \mathcal{C} \) be the input and output flows of a \( \sigma \)-regulator, as illustrated by Fig. 1.

This system is constrained by the following equations in \( \mathcal{C} \) (see Table VII): \( \tilde{x} = \tilde{x} + \tilde{a} \) and \( \tilde{y} = \tilde{y} + \tilde{y} \sigma \), or equivalently, by the addition of these equations (see Table I),

\[
\tilde{x} = \tilde{x} (e + \sigma) + \tilde{a}.
\]

![Fig. 1. Lossless regulator with characteristic curve \( \sigma \in \mathcal{C} \).](image-url)
The output of the greedy shaper is the optimal solution of (9). That is, according to Theorem 1:
\[ \hat{x} = \hat{a}(e \oplus \sigma)^* = \hat{a}\sigma^*. \]  

**Proposition 10:** The greedy shaper with characteristic function \( \sigma \in \mathcal{C} \) admits a service curve \( \beta \in \mathcal{C} \), given by \( \beta = (e \oplus \sigma)^* = \sigma^* \).

*Proof:* According to (7), it is sufficient to prove that \( \hat{a}(e \oplus \sigma)^* = \hat{a}(e \oplus \sigma)^* \hat{x} \). This is verified directly from (10) and the idempotency of the addition. \[ \square \]

Therefore, this system acts as a \( \sigma \)-regulator and simultaneously as a \( \sigma^* \)-network and that suggests a special symbol for this system, as depicted by Fig. 2.

**B. The window flow controller**

In this section, we analyze a window flow controller. This problem was previously studied in [8] and [9], for a time-invariant scenario of traffic.

Consider that a given flow \( a \in \mathcal{C} \) is transmitted over a network that guarantees a service curve \( \beta \in \mathcal{C} \), as depicted by Fig. 3. The window flow controller limits the amount of packets inside the network as described next. It is assumed that there are \( W(t) \) “tokens” available for the network at instant \( t \) in a token buffer, where \( W(t) \in \mathbb{N} \) is the “window size” of the adaptive session at \( t \). Whenever a packet accesses the network, a token is subtracted from the token buffer; whenever a packet leaves the network, a token is added to the token buffer. Therefore, the window flow controller acts as a regulator, delaying the packets of \( a \) when necessary, in order to guarantee that the backlog of the \( \beta \)-network is always less than or equal to \( W(t) \). Fig. 3 presents each part of the physical system. In this figure, it is evidenced the ideal behavior of this system, where the number of packets inside the \( \beta \)-network, \( B(t) \), is fed to the \( W \)-controller, so that it can calculate the number of tokens still available at \( t \). In Fig. 4, the mathematical constraints of this system are illustrated as an assembly of basic elements. Remark that, in this figure, the \( W \)-controller acts as an open loop pre-compensator, in the sense that the number of packets inside the \( \beta \)-network is not fed back to the controller.

We want to determine: (1) a possible region for the characteristic curve of the controller, \( \sigma \in \mathcal{C} \), so that it can assure that the number of packets inside the network will never exceed \( W(t) \); (2) a lower and an upper bounds for the output function \( y(0, t) \), that represents the number of packets that leave the system during the interval \( (0, t) \), i.e., the number of packets that leave the system up to instant \( t \). Consider that Proposition 4 or Proposition 5 can be applied for all multiplications that appear next, e.g., \( \beta(s, t) \) is an increasing function in \( t \) and \( \beta(s, s) = 0 \), \( \forall s \).

From Table VII, the following equations must be satisfied for this system:
\[ \hat{x} = \hat{x} \oplus \hat{a}, \]  
\[ \hat{x} = \hat{x} \oplus \hat{x} \sigma, \]  
\[ \hat{y} = \hat{y} \oplus \hat{x}, \]  
\[ \hat{x} \beta = \hat{x} \beta \oplus \hat{y}, \]  
\[ \hat{x} = \hat{x} \oplus \hat{y} \lambda_W. \]  

If we consider that the system is causal, then (11) and (13) are directly satisfied. Besides, from (12) and Table VI, \( \hat{x} \gg \hat{x} \sigma \) and then, from Table II-(P1), \( \hat{x} \ll \hat{x} \not\in \sigma \). Therefore, from (14) and Table II-(P2),
\[ \hat{y} \ll (\beta \not\in \sigma) \beta \ll (\beta \not\in \sigma) \beta. \]

Thus, \( \hat{x} \gg \hat{y} (\beta \not\in \sigma) \) and then, from (15),
\[ \hat{x} \gg \hat{y} (\beta \not\in \sigma) \oplus \lambda_W. \]

Finally, if we choose \( \sigma \) such that \( \beta \not\in \sigma \gg \lambda_W \), then (16) is reduced to \( \hat{x} \gg \hat{y} (\beta \not\in \sigma) \) and (15) will be satisfied for all \( \hat{x}, \hat{y} \) satisfying (11)-(14). Therefore, \( \sigma \) can be defined on the following region:
\[ \sigma \gg \beta \lambda_W. \]

Now, to determine a lower bound in \( \mathcal{C} \) for \( \hat{y} \), define \( u = \hat{x} \beta \), such that \( u \gg \hat{x} \beta \) and \( u \ll \hat{x} \beta \). Thus, (11)-(15) can be rewritten by the following way:
\[ [X \ u] = [X \ u] \oplus K \oplus L, \]  
\[ u \ll \hat{x} \beta, \]  

with
\[ X_T = \begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix}, \quad K = \begin{bmatrix} e \oplus \sigma \ e \ \\ \lambda_W \ e \ e \end{bmatrix} \quad \text{and} \quad L_T = \begin{bmatrix} \hat{a} \\ \varepsilon \\ \varepsilon \\ \varepsilon \end{bmatrix}. \]

From Theorem 1, the optimal solution of (18) is
\[ \begin{bmatrix} x_\beta \\ y_\beta \\ u_\beta \end{bmatrix} = L K^* \text{ given by the following way:} \]
\[ \begin{bmatrix} x_\beta \\ y_\beta \\ u_\beta \end{bmatrix} = \begin{bmatrix} \hat{a} \ e \ e \ e \\ \hat{a} \ e \ e \ e \\ \hat{a}(e \oplus \sigma)^* \hat{a}(e \oplus \sigma)^* \hat{a}(e \oplus \sigma)^* \end{bmatrix}. \]
Notice that (19) is verified for \( u_b \) and \( \tilde{x} \). In fact, \( u_b = a((e \circ \sigma)^* \beta = \tilde{x}_p \beta \), therefore, \( \tilde{y}_b = a((e \circ \sigma)^* \beta = \tilde{x}_p \beta \), is a possible lower bound for \( \tilde{y} \) in \( C \) and then \( \tilde{y} \geq \tilde{y}_b \), with

\[
y_b(t) = \tilde{y}_b(0, t) = [\tilde{x}(e \circ \sigma)^* \beta = \tilde{x}_p \beta \], where the last equality comes from Definition 5.

Finally, we determine an upper bound for \( \tilde{y} \) in \( C \). From (14), \( \tilde{y} \leq \tilde{x} \beta \) and, from (15) and Table II-(P1), \( \tilde{y} \leq \tilde{x} \beta \). Then, (13)-(15) can be rewritten by the following way:

\[
\tilde{y} = \tilde{y} \wedge \tilde{x} \beta \wedge (\tilde{x} \beta \not\in W),
\]

\[
\tilde{y} \geq \tilde{x} \beta.
\]

From (20) and Theorem 2, \( \tilde{y} = \tilde{x} \beta \wedge (\tilde{x} \beta \not\in W) \) is a possible upper bound for \( \tilde{y} \) in \( C \), since it satisfies (21), for all possible \( \tilde{x} \). Therefore, \( \tilde{y} \leq \tilde{y}_b \), and then

\[
y_b(t) = \tilde{y}_b(0, t) = [\tilde{x} \beta \wedge (\tilde{x} \beta \not\in W)](0, t)
= [\tilde{x} \beta \wedge (\tilde{x} \beta \not\in W)](0, t)
\]

(22)

(23)

is a possible bound for \( y(0, t) \).

In [9], Agrawal and Rajan showed that function \( y_{AR}(t) = [\beta x(\beta \lambda_W)^* x(\beta \lambda_W)^* \beta = \tilde{x}_p \beta \), is a possible bound of \( y(t) \), for the particular case where \( W(t) = W \), \( \forall t \), and \( \beta \), \( x \), \( \lambda \) and \( y \) are time-invariant functions. Next, we show that \( y_{AR}(t) \) is a better bound for \( y(t) \), i.e., \( y(t) \geq y_{AR}(t) \geq y_{AR}(t) \), \( \forall t \), if \( \sigma \) does not satisfy (17); and \( y_{AR}(t) \leq y_{AR}(t) \), otherwise, for the same scenario of traffic.

On one hand, we have, \( \forall x, \beta, \lambda_W \in C \),

\[
x \beta \wedge (\tilde{x} \not\in W) \leq x \beta \leq x \beta(\beta \lambda_W)^* \cdot
\]

Therefore, \( \forall (s, t) \): \( [x \beta \wedge (\tilde{x} \not\in W)](s, t) \geq [x \beta(\beta \lambda_W)^* x(\beta \lambda_W)^* \beta = \tilde{x} \beta \), and then, from (23),

\[
y_b(t) = [x \beta \wedge (\tilde{x} \not\in W)](0, t)
\leq [x \beta(\beta \lambda_W)^* x(\beta \lambda_W)^* \beta = \tilde{x} \beta \not\in W)](0, t)
\]

(24)

(25)

where (24) comes from the commutativity of the product of time-invariant functions, as shown in [3].

On the other hand, if \( \sigma \geq \beta W \), then, from (12),

\[
\tilde{x} \geq \tilde{x} \sigma \geq \tilde{x}(\beta W) = (\beta W)
\]

Thus, from Table II-(P1), \( \tilde{x} \beta \leq \tilde{x} \beta \not\in W \) and then \( \tilde{x} \beta \wedge \tilde{x} \beta = \tilde{x} \beta \). Now, from (12), recursively, \( \tilde{x} = \tilde{x} \sigma \). Therefore,

\[
\tilde{x} \beta \wedge (\tilde{x} \not\in W) = \tilde{x} \beta = \tilde{x} \sigma \beta
\]

(26)

because of the assumption that \( \sigma \geq \beta W \). Then, from (22), (26), Definition 5 and Table VI, \( \forall t \):

\[
y_b(t) = [\tilde{x} \beta \wedge (\beta \not\in W)](0, t)
\leq [\tilde{x}(\beta \lambda_W)^* x(\beta \lambda_W)^* \beta = \tilde{x} \beta \not\in W)](0, t)
\]

(27)

Finally, from (25), \( y_{AR}(t) \geq y_{AR}(t) \), \( \forall t \). Moreover, when \( \sigma \) satisfies (17), then (27) is also satisfied and, therefore, \( y_b(t) = y_{AR}(t) \).

C. The greedy shaper as a window flow controller

In this section, we find possible bounds for \( y(t) \), when the \( W \)-controller presented in the previous section is the greedy shaper analyzed in Section V-A with \( \sigma = \beta W \).

When \( \sigma = \beta W \), \( \tilde{y}_b = \tilde{x} \beta \not\in W \). Moreover, (17) is satisfied and then, as previously commented, \( \tilde{y}_b = \tilde{x} \beta \not\in W \), (26). On the other hand, as a greedy shaper, the \( W \)-controller satisfies (10). Thus, \( \tilde{x} = \tilde{x} \beta \not\in W \), and then \( \tilde{y}_b = \tilde{x} \beta \not\in W \). Therefore,

\[
y(t) = [\tilde{x}(\beta \lambda_W)^* x(\beta \lambda_W)^* \beta = \tilde{x} \beta \not\in W)](0, t)
\]

(28)

(29)

and \( y_b \geq y_b \). Now, for the particular case where \( W(t) = W \), \( \forall t \), and \( \beta \) and \( \sigma \) are time-invariant functions, then, from (28), \( y_b(t) = [\beta(\lambda_W)^* x(\beta \lambda_W)^* \beta = \tilde{x} \beta \not\in W)](0, t) \). This last result was previously achieved in [8].

VI. CONCLUSION

In this article, we further develop the ideas of the NC (see, for example, [5] and [3]) through a deeper utilization of some Max-Plus algebraic properties. To exemplify the introduced methodology, we analyzed a greedy shaper and a window flow controller. The defined methodology is modular and the obtained results show that it can be applied for the solution of complex systems. Our results are in agreement with [8] and [9] for a time-invariant scenario of traffic with constant window size and represent an extension for the time-variant case with variable window size.

REFERENCES


