Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 58-1 (2024) 300–305

Modeling Time-Sensitive Networking Using Timed Continuous Petri Nets [⋆]

A. G. Torres-Macías [*] A. Ramírez-Treviño [*] J. L. Briz [**] J. Segarra [**] H. Blanco-Alcaine [***]

[*] *CINVESTAV-IPN Unidad Guadalajara, México (e-mail: {alitzel.torres, antonio.ramirezt}@cinvestav.mx).*
[**] *Dept. Informática e Ingeniería de Sistemas / I3A, Universidad de Zaragoza, España (e-mail: {briz, jsegarra}@unizar.es).*
[***] *Intel Deutschland GmbH (e-mail: hector.blanco.alcaine@intel.com).*

Abstract: Time-Sensitive Networking (TSN) is a set of standards designed to provide determinism to Ethernet or wireless networks for real-time applications. Network Calculus (NC) is the best-known formal tool for TSN analysis, but it is hard to apply in scenarios of changing conditions. Petri Nets can potentially overcome some limitations of NC, but they have rarely been used as an analytical tool for TSN. This work presents a novel modeling methodology exploiting Timed Continuous Petri Nets (TCPNs) to abstract and analyze TSN nodes.

Keywords: Petri nets, Communication networks, Discrete event systems, Discrete event modeling and simulation, Control on networks.

1. INTRODUCTION

Time-Sensitive Networking (TSN) is a set of standards designed to provide determinism to Ethernet or wireless networks, ensuring end-to-end latencies in the order of micro to few milliseconds (Ultra Low Latency Networks). They are the basis of future real-time technologies for control and automation, transportation, and infrastructure, enhancing proprietary standards by allowing the coexistence of multiple protocols, such as Profinet or EtherCAT, to name a few.

IEEE has been working on related standards over the last two decades, driving research into the several open questions that arise during the development and implementation of these standards. The field is extraordinarily dynamic, with standards extending or enhancing the capabilities of earlier standards that, due to demand, have already been or are in the process of implementation and commercialization (e.g. Intel® Ethernet Controller i210 or Intel® Ethernet Controller I226). The formal guarantee of deterministic QoS demands effective modeling and analysis tools.

Network Calculus (NC) (Cruz (1991a,b); Le Boudec (1996); Le Boudec and Thiran (2001)) is a well-known and powerful tool, with some limitations when it comes to complex modeling of spatio-temporal interactions between different stream classes and concurrent events (Bölöni (2011)). One of the most important current problems of TSN is the vulnerability to unexpected changes in network conditions because stream management policies, necessarily deterministic in this case, are static (Seol, Y. et al. (2021)). According to Bose, S.K. et al. (2014), NC is poorly applicable in scenarios of changing conditions.

These two factors require methods capable of monitoring changes in the network state, dynamically adapting the configurations of different streams, and preserving the fulfillment of the proper end-to-end latencies in critical streams.

Stochastic or Timed Petri Nets (PNs) have been used to model communications networks because their underlying graph captures the topology of the network, and the firing rate of transitions models the stochastic aspects and probabilities of sending messages. The (max, +) algebra (Heidergott, B. et al. (2006)), largely used in NC, also appears in Timed PNs when calculating temporal boundaries between begin-end pairs and sums of sequences. Similarly, the (min, +) algebra has been used to exploit the slacks between producer (supply) and consumer (demand) paths (Farhi, N. et al. (2009)). However, PNs have hardly been applied to TSN, only as a descriptive and not analytical tool (Tang, S. et al. (2020)).

The contribution of this preliminary work is a methodology to incrementally derive a Timed Continuous Petri Net (TCPN) model for TSN, based on the arrival and service curves obtained from NC. This model is dynamic, so it has the potential advantage of dealing with the aforementioned changing conditions in a network which hampers other models, and in addition, is amenable to analysis and control. In the future, the TCPN state equation may be used to implement different types of control over TSN gates (e.g. on/off control, regulation control, or sliding modes). These solutions have been exploited in TCPN models targeting control problems in other areas, such as Desirena-López, G. et al. (2019); Rubio-Anguiano L.E. et al. (2020); Rubio-Anguiano, L.E. et al. (2021), and open

[⋆] This work was supported by MCIN/AEI/10.13039/501100011033 (grants PID2019-105660RB-C21 and PID 2022-136454NB-C22), the Government of Aragon (T58_23R research group), Spain, and the CINVESTAV Unidad Guadalajara, Mexico.

up new opportunities to devise dynamic mechanisms not yet considered for TSN.

This paper is structured as follows. Sec. 2 provides an overview of the principles and basic concepts related to TCPNs, TSN, and NC. Sec. 3 presents our contribution: a modeling methodology for TSN nodes using TCPNs. Finally, Sec. 4 summarizes conclusions and future work.

2. PRELIMINARIES

This section briefly recalls the basic concepts of PNs and TCPNs and some of the principles of TSN and NC. An interested reader may consult Murata (1989); David and Alla (2010); IEEE 802.1 (n.d., 2022); Cruz (1991a,b); Le Boudec and Thiran (2001) for a deeper insight.

2.1 Timed Continuous Petri Nets

PNs are mathematical and graphical tools for modeling, analyzing, and controlling Discrete Event Systems (DES). They are especially suited to describe concurrent, parallel, and asynchronous systems, to name a few (Murata (1989)).

Definition 1. A *Petri Net structure* is a bipartite digraph $N = (P, T, Pre, Post)$, where P and T are finite non-empty disjoint sets of places and transitions, respectively, and $Pre, Post : P \times T \to \mathcal{N} \cup \{0\}$ are functions representing the weighted arcs going from places to transitions and from transitions to places, respectively.

The $|P| \times |T|$ incidence matrix $\boldsymbol{C}$ of N is defined by $\boldsymbol{C}[i,j] = \boldsymbol{C}^{+}[i,j] - \boldsymbol{C}^{-}[i,j]$, where $\boldsymbol{C}^{+}[i,j] = Post(i,j)$, and $\boldsymbol{C}^{-}[i,j] = Pre(i,j)$. A transition (place) that has more than one input place (transition) is named *join transition* (*attribution place*); a transition (place) with more than one output place (transition) is called *fork transition* (*choice place*). A place $p \in P$ is called a *self-loop place* if it is both an input and output place of a transition $t \in T$. The continuous marking function $m : P \to \mathcal{R}^{+}$ assigns a non-negative real number to each place.

Definition 2. A *Continuous Petri Net system* is defined as $CPN = (N, \boldsymbol{m}_0)$, where N is a Petri Net structure, and $\boldsymbol{m}_0 \in \{\mathcal{R}^{+}\}^{|P|}$ is the initial marking.

A transition t_i is *enabled* at a marking $\boldsymbol{m}$ if and only if $\forall\, p_j \in {}^{\bullet}t_i$, $\boldsymbol{m}[p_j] > 0$ and its *enabling degree* is defined as

$$enab(t_i, \boldsymbol{m}) = \min_{p_j \in {}^{\bullet}t_i} \left(\frac{\boldsymbol{m}[p_j]}{\boldsymbol{C}^{-}[p_j, t_i]} \right). \quad (1)$$

Firing t_i in any real amount between the interval $0 \le \alpha \le enab(t_i, \boldsymbol{m})$ leads to a new marking $\boldsymbol{m'} = \boldsymbol{m} + \alpha \boldsymbol{C}[P, t_i]$. Place p_j is constraining t_i when $enab(t_i, \boldsymbol{m}) = \frac{\boldsymbol{m}[p_j]}{\boldsymbol{C}^{-}[p_j, t_i]}$.

Marking $\boldsymbol{m}$ is said to be reachable if it can be reached from $\boldsymbol{m}_0$ by firing the sequence σ of enabled transitions. The Reachability set contains all reachable markings from $\boldsymbol{m}_0$, and they satisfy the fundamental equation $\boldsymbol{m} = \boldsymbol{m}_0 + C\boldsymbol{\sigma}$, where $\boldsymbol{\sigma} \in \{\mathcal{R}^{+}\}^{|T|}$ is the firing count vector.

Definition 3. A *Timed Continuous Petri Net system* is defined as $TCPN = (N, \boldsymbol{\lambda}, \boldsymbol{m}_0)$, where $(N, \boldsymbol{m}_0)$ is a CPN and the vector $\lambda \in \{\mathcal{R}^{+}\}^{|T|}$ represents the transitions firing rates determining the temporal evolution of the system.

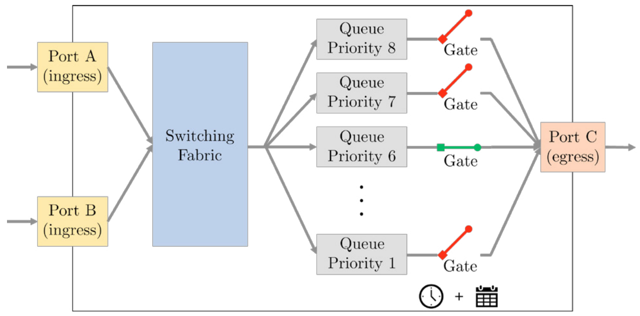


Fig. 1. Simplified TSN Switch as defined in the IEEE 802.1AS and 802.1Qbv standards; adapted from Serna Oliver, R. et al. (2018).

Under infinite server semantics, the firing flow $\boldsymbol{f(m)}$ of a transition t_i is proportional to its enabling degree and is defined as $f_i(\boldsymbol{m}) = \lambda_i enab(t_i, \boldsymbol{m})$. A configuration of a *TCPN* at $\boldsymbol{m}$ is a set of (p,t) arcs describing the effective flow of each transition. The vectorial form of the firing flow is given by $\boldsymbol{f(m) = \Lambda\Pi(m)m}$, where $\boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_{|T|})$ is the firing rate matrix and

$$\boldsymbol{\Pi_{j,i}(m)} = \begin{cases} \dfrac{1}{\boldsymbol{C}^{-}[p_j, t_i]} & \text{if } p_i \text{ is constraining } t_j, \\ 0 & \text{otherwise} \end{cases}$$

is the configuration matrix defined for each configuration.

The state equation that describes the dynamic behavior of a *TCPN* is

$$\dot{\boldsymbol{m}} = \boldsymbol{C f(m) = C \Lambda \Pi(m) m}. \quad (2)$$

A control action can be applied to (2) by adding a term $\boldsymbol{u}$ to every transition t_i such that $0 \le u_i \le f_i$, indicating that its flow can be reduced or stopped. Thus, the controlled flow of transition t_i becomes $w_i = f_i - u_i$, and the forced state equation is $\dot{\boldsymbol{m}} = \boldsymbol{C[f - u] = C\Lambda\Pi(m)m - Cu}$, where $\boldsymbol{0 \le u \le \Lambda\Pi(m)m}$.

2.2 Time-Sensitive Networking

A TSN is composed of *end stations* and *bridges* (also referred to as *switches*). End stations are the sources and destinations of the data streams. Switches are capable of receiving and transmitting the frames of a stream according to a schedule defined as a set of *Gate Control List* (GCL). The gate state (open or closed) in a GCL determines whether or not frames from the respective queue will be transmitted over the physical link (Serna Oliver, R. et al. (2018); Lo Bello and Steiner (2019)). Fig. 1 represents a simplified TSN switch.

2.3 Network Calculus and the modeling of TSN

Based on tropical algebras as (min, +) and (max, +) algebras, NC provides a mathematical framework for analyzing performance guarantees in TSN (Bemten and Kellerer (2016)). These algebras model network elements such as streams, nodes, or even complete networks as functions.

One of the most fundamental operations in NC is the *convolution*, defined in the (min, +) algebra as

$$(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(t-s) + g(s)\}. \quad (3)$$

This operation is used to compute the output of a traffic shaper or to merge nodes in series into one single node (Bemten and Kellerer (2016)).

The incoming and the outgoing traffic of each node are abstracted with curves, which are cumulative functions $R(t)$ and $R^*(t)$, respectively. They represent the number of bits observed on the stream throughout the time interval $[0, t]$ (Maile, L. et al. (2020)).

We can derive two quantities of interest from $R(t)$ and $R^*(t)$: the *backlog* $x(t)$ and the *virtual delay* (Le Boudec and Thiran (2001)). The *backlog* is the number of bits being held inside a node at time t, while the *virtual delay* is the delay experienced by a bit arriving at time t if all bits received before t are served before t.

In order to provide the performance bounds for a stream, we need to know two parameters: the maximum data a stream will send and the minimum service a network guarantees to offer. These two parameters are respectively given by *arrival curves* (stream modeling) and *service curves* (modeling of the network and its nodes) (Bemten and Kellerer (2016)).

Arrival curves An arrival curve $A(t)$ represents the maximum number of bits that arrive at the node during any period of length t (Maile, L. et al. (2020)). The most commonly used arrival curve is the *affine function* $\gamma_{r,b}(t)$ (a.k.a. *leaky bucket* or *token bucket*) and is defined as:

$$A(t) = \gamma_{r,b}(t) = \begin{cases} b + rt & \text{if } t > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

Here, r is the *rate* and b is the *burst.* This type of arrival curve allows a source to send b bits at once but no more than r b/s over the long run (Bemten and Kellerer (2016)). For any two affine functions, $\gamma_1 \otimes \gamma_2 = \min(\gamma_1, \gamma_2)$ (Le Boudec (1996)).

A *constant bit rate* (CBR) *connection* is a stream constrained by an affine arrival curve. A *variable bit rate* (VBR) *connection* is a stream constrained by two affine curves in a series, i.e., by an arrival curve of the form:

$$A(t) = \gamma_{r_1,b_1}(t) \otimes \gamma_{r_2,b_2}(t) = \min(b_1 + r_1 t, b_2 + r_2 t). \tag{5}$$

This kind of arrival curve is used in the Internet Integrated Services (IntServ) framework (Braden, R. et al. (1994)).

Service curves The details of packet handling by a node or a network are abstracted in TSN using service curves $S(t)$. The selection of these curves requires a comprehensive analysis of the queuing, scheduling, and forwarding mechanisms (Maile, L. et al. (2020)). The *rate-latency function* $\beta_{R,T}(t)$ is one of the most common functions used as a service curve and is defined by:

$$S(t) = \beta_{R,T}(t) = R\,[t - T]^+ = \begin{cases} R(t - T) & \text{if } t > T \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where R is the *rate* and T is the *delay.* Bits might have to wait up to T before being served with a rate greater or equal to R (Bemten and Kellerer (2016)).

A *maximum service curve* is of the form of a *peak rate function* $\lambda_R(t)$, defined as

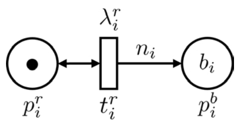


Fig. 2. TCPN module for an arrival or a service curve.

$$S(t) = \lambda_R(t) = \begin{cases} Rt & \text{if } t > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where R is the *rate.* A rate-latency service curve is usually used to model an approximation of a generalized processor sharing (GPS) node, while a maximum service curve models a perfect GPS node (Bemten and Kellerer (2016)).

Shaping devices A *policer* is a device deciding if the traffic conforms to an arrival curve; it is located at the edge of the network (Shenker, S. et al. (1997)). A *shaper* is a device forcing the traffic to conform to an arrival curve; it is located at the source branch and merge points (Shenker, S. et al. (1997)). A *greedy shaper* is a shaper delaying the traffic in a buffer whenever sending it would transgress the constraints, but serving it as soon as possible (Le Boudec and Thiran (2001); Bemten and Kellerer (2016)).

A *buffered leaky bucket* is a shaper associated with a shaping curve $\sigma(t)$ of the form:

$$\sigma(t) = \min(Rt, rt + b). \tag{8}$$

It enforces both a peak rate R and a sustainable rate r with burstiness parameter b (Le Boudec (1996)).

3. TSN MODELING WITH TCPN

NC elements (i.e., arrival and service curves) represent the streams and nodes in TSN. The modeling methodology herein proposed models the basic curves presented in Sec. 2.3 as the TCPN module in Fig. 2, which is described in detail below:

(1) The rate at which a data stream i arrives or is served on a node is represented as the firing rate λ_i^r of the transition t_i^r and the self-loop place p_i^r whose marking must be 1.
(2) The marking at place p_i^b represents the data that has arrived or has been served on a node. For the specific case of arrival curves, the initial marking of p_i^b, given by b_i, represents the burst; for the case of service curves $b_i = 0$.
(3) And n_i, the weight of the input arc of the place p_i^b, represents the possible multiple streams of the same curve, as required.

The differential equations that dictate the marking evolution of the TCPN module are as follows:

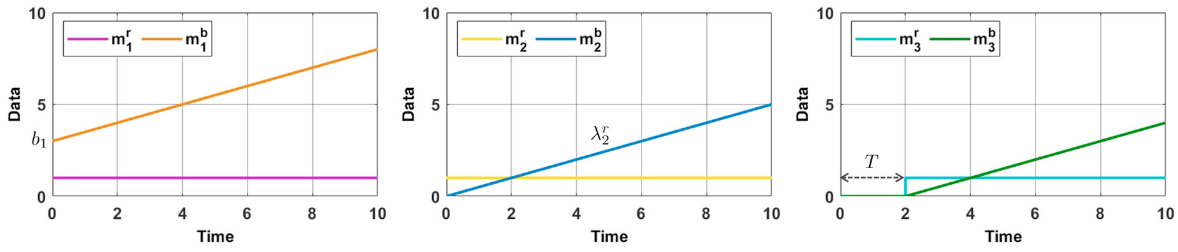
$$\begin{aligned} \dot{m}_i^r &= 0, \\ \dot{m}_i^b &= n_i \lambda_i^r m_i^r. \end{aligned} \tag{9}$$

To model a rate-latency service curve, it is necessary to delay the marking of p_i^r, i.e., the marking at this place will be:

$$m_i^r = \begin{cases} 1 & \text{if } t > T \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

where T represents the delay.

Fig. 3 shows examples of the marking evolution of the TCPN module for affine (Fig. 3a), peak rate (Fig. 3b)



(a) Affine curve. (b) Peak rate curve. (c) Rate-latency curve.

Fig. 3. Marking evolution graphs of some curves' TCPN models.

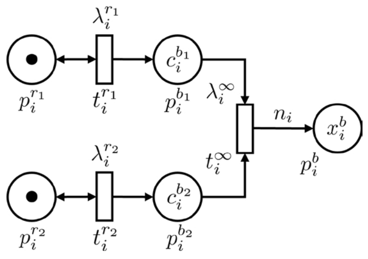


Fig. 4. TCPN model for a VBR connection.

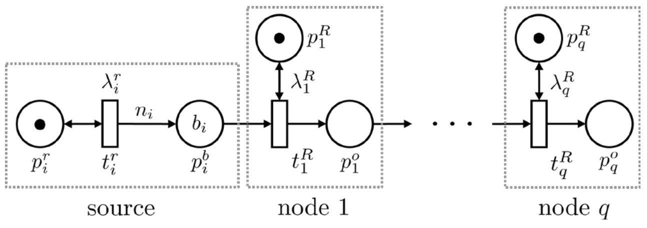


Fig. 5. TCPN model for a leaky bucket through q nodes.

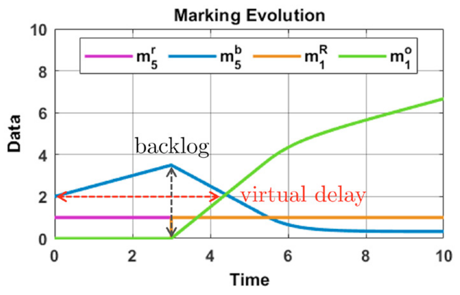


Fig. 6. Graphical depiction of backlog and virtual delay.

and rate-latency (Fig. 3c) type curves. We are mainly interested in the m_k^b markings, with $k = \{1, 2, 3\}$, as they represent the final shape of these curves.

Using this module as a basis, we model more types of curves (such as VBR connections), shapers, and streams through nodes (complete networks). A global TCPN model is an aggregation of TCPN modules. Moreover, we know what is happening in the network with different places and transitions in the TCPN. For example, we respectively merge and split traffic with attribution and choice places p_i^b, and we perform convolution with join transitions t_i^r.

Fig. 4 shows the TCPN model of a VBR connection. It consists of two modules in parallel synchronized by a join transition t_i^∞, whose firing rate λ_i^∞ must be exaggeratedly large to avoid slowing down the stream. The rates and bursts of the affine curves describing the VBR connection are given by $\lambda_i^{r_k}$ and b_k, respectively, with $k = \{1, 2\}$. The initial marking of the place p_i^b represents the total burst, defined as $x_i^b = \min(b_1, b_2)$, while the initial marking of places $p_i^{b_k}$ is given by $c_i^{b_k} = b_k - x_i^b$, with $k = \{1, 2\}$.

The differential equations for this TCPN model are:

$$\begin{aligned}
\dot{m}_i^{r_1} &= 0,\\
\dot{m}_i^{b_1} &= \lambda_i^{r_1} m_i^{r_1} - \lambda_i^\infty \min(m_i^{b_1}, m_i^{b_2}),\\
\dot{m}_i^{r_2} &= 0,\\
\dot{m}_i^{b_2} &= \lambda_i^{r_2} m_i^{r_2} - \lambda_i^\infty \min(m_i^{b_1}, m_i^{b_2}),\\
\dot{m}_i^{b} &= n_i \lambda_i^\infty \min(m_i^{b_1}, m_i^{b_2}).
\end{aligned} \tag{11}$$

Fig. 5 depicts the model of a leaky bucket stream passing through multiple nodes. The outgoing stream of one node is the incoming stream of the next, and so on. There are two relevant aspects of this model:

(1) The rate of the leaky bucket is the firing rate λ_i^r of the transition t_i^r, while the rate of the nodes is the firing rate λ_k^R of the transition t_k^R, with $k = \{1, ..., q\}$.

(2) The initial marking of p_i^b, given by b_i, represents the burst of the leaky bucket. The marking at places p_k^R, with $k = \{1, ..., q\}$, will depend on the type of service curve of the node. The marking at intermediate places p_i^b, p_1^o, ..., p_{q-1}^o corresponds to the backlog.

The following differential equations rule the marking evolution of this TCPN model:

$$\begin{aligned}
\dot{m}_i^{r} &= 0,\\
\dot{m}_i^{b} &= \lambda_i^{r} m_i^{r} - n_i \lambda_1^R \min(m_i^{b}, m_1^{R}),\\
\dot{m}_1^{R} &= 0,\\
\dot{m}_1^{o} &= n_i \lambda_1^R \min(m_i^{b}, m_1^{R}) - \lambda_2^R \min(m_1^{o}, m_2^{R}),\\
&\vdots\\
\dot{m}_q^{R} &= 0,\\
\dot{m}_q^{o} &= \lambda_q^R \min(m_{q-1}^{o}, m_q^{R}).
\end{aligned} \tag{12}$$

Fig. 6 shows an example of the marking evolution of a leaky bucket stream through a rate-latency node. We can obtain the same outcomes from this graph as from NC, such as the backlog (the value of m_5^b when $t = T$) and the virtual delay (the value of t when m_5^b and m_1^o intersect). We also obtain the evolution of the system state (Eq. 12).

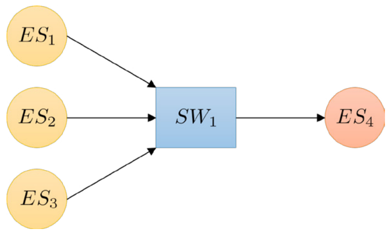


Fig. 7. Example network topology.

3.1 Modeling methodology

In this section, using what was already discussed and by means of a simple example, the TSN modeling methodology with TCPNs is presented. The example is based on the one from (Georgiadis, L. et al., 1996, Sec. 4.3) and targets bit rate guarantees but can be adapted to meet latency constraints.

We assume an OC-3 channel (155 Mb/s) output link at a switch. This link only transports the three types of streams listed in Table 1, with n as the number of streams, L as the max packet size, p as the max peak rate, b as the token bucket size, and r as the token accumulation rate; each traffic type sourced from a different end station. Fig. 7 shows the network topology.

Table 1. Streams characteristics

Traffic Type	n	L kB	b kB	p Mb/s	r Mb/s
64 Kb/s Voice	200	0.1	0.1	0.064	0.064
Video Conference	26	1.5	10	10	0.5
Stored Video	10	1.5	100	10	3

The first thing to do is to identify the network elements to build the TCPN model. The streams generated at the end stations (ES) are modeled by arrival curves, and from Table 1, we know that these curves are given by a *VBR connection*, with $A(t) = \min(L + pt, b + rt)$. The switch (SW) is modeled by a service curve given by a *peak rate function*, i.e., $S(t) = Rt$, where R is the speed of the OC-3 link. Thus, we have three VBR connections passing through a single peak rate node.

The next and last step consists of grouping the TCPN modules based on the identification of the network elements. Since we are dealing with a traffic mix, the p_i^b places of each VBR connection (Fig. 4) will be merged, obtaining an attribution place. This new place will be a new input to the transition t_i^r of the module of the peak rate node (Fig. 2), performing the convolution. Fig. 8 shows the complete TCPN model with the following values:

$$n_1 = 200,\ n_2 = 26,\ n_3 = 10,\ \lambda_1^\infty = \lambda_2^\infty = \lambda_3^\infty = 2000000,$$
$$c_1^L = c_1^b = c_2^L = c_3^L = 0, c_2^b = 0.068, c_3^b = 0.788, x^b \approx 0.025.$$

According to Georgiadis, L. et al. (1996), this mix is feasible as long as we can use the entire 155 Mb/s of transmission speed, and the node uses a Rate Controlled Service (RCS) discipline with the Earliest Deadline First (EDF) policy as the scheduler. Fig. 9a graphically depicts the schedulability check for EDF at the node for the traffic mix considered. The dotted blue line passing through the origin corresponds to the OC-3 link, while the red curve is the sum of all the traffic curves.

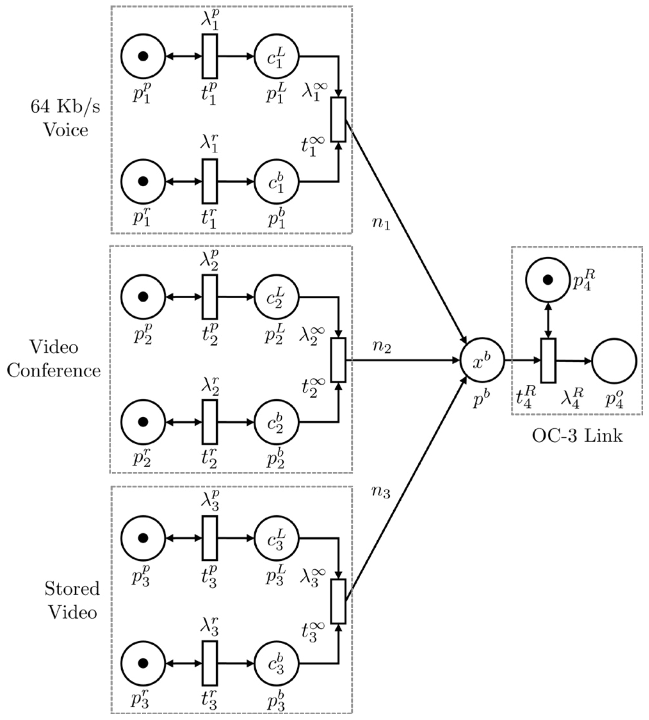


Fig. 8. TCPN model for three traffic through an OC-3 link.

Fig. 9b shows the marking evolution of place p_4^o, from which we observe that the TCPN model closely mirrors the NC schedulability check (Fig.9a), representing the traffic mix over the OC-3 link as the NC model does.

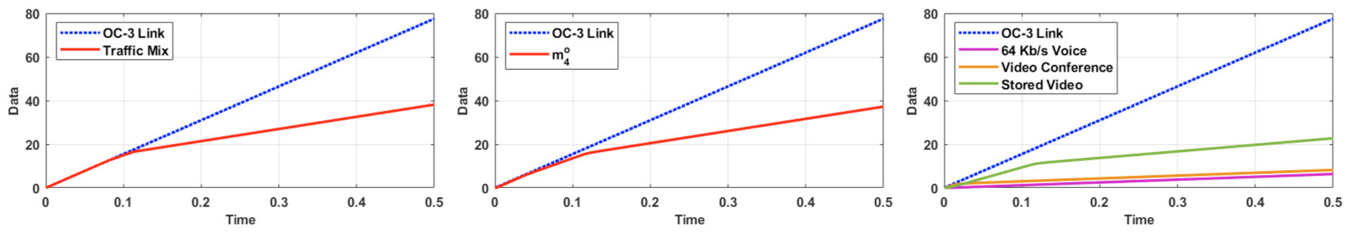
As an advantage, the TCPN model can take into account scheduling decisions because it keeps the three types of traffic streams separately and encompasses the gates of the TSN switch; we omit this TCPN model for the sake of space, but Fig. 10 below provides an insight into this idea. Fig. 9c shows the three separate traffic streams after passing through the OC-3 link. If we sum the data quantities at $t = 0.5$, we get the same results as in Fig. 9b, so mass conservation exists.

4. CONCLUSIONS AND FUTURE WORK

We present a novel modeling methodology with TCPNs, which can be applied to model streams, network nodes, and complete networks. We show that our model achieves identical results to NC without resorting to (min, +) and (max, +) algebras.

There is another potential advantage of TCPN over NC models: we may exploit the dynamic properties and state model of TCPNs to design an online control over TSN gates (see Fig. 1), establishing how they will be opened and closed to satisfy end-to-end delay, jitter, and other parameters, and to increase the robustness of the system against manageable perturbances or parametric variations.

We outline the basis for carrying out this idea with Fig. 10. We decide which stream will be served first (according to priorities and the scheduling policy) by controlling the firing of transitions $t_{k,i}^{g_o}, t_{k,i}^{g_c}$ or $t_{k,j}^{g_o}, t_{k,j}^{g_c}$. Immediate future work includes determining the best type of control to achieve these deterministic bounds, with the objective of regulating the scheduler.



(a) Schedulability check. (b) Marking evolution of place p_4^o. (c) Traffic passing through an OC-3 link.

Fig. 9. Schedulability check for EDF for traffic in Table 1.

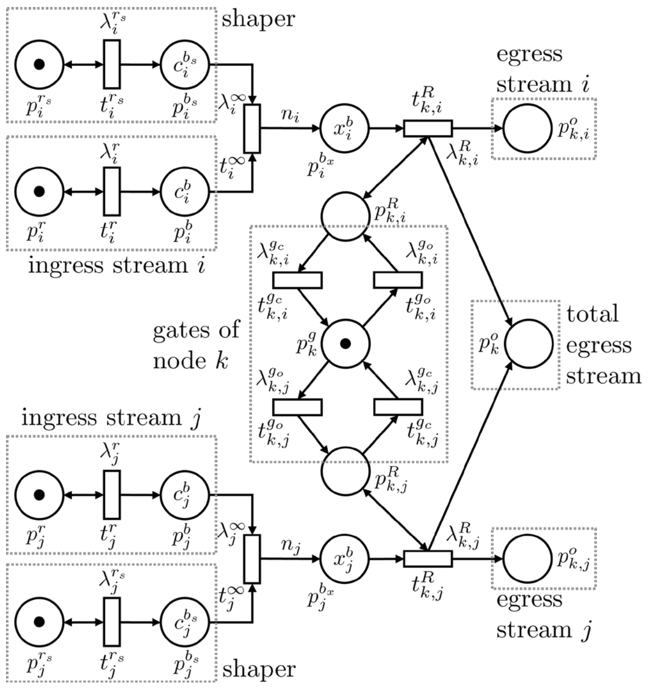


Fig. 10. TCPN model of a single egress port (Fig 1), with two input streams and two output queues and gates.

REFERENCES

Bemten, A.V. and Kellerer, W. (2016). Network Calculus: A Comprehensive Guide. Technical report, Technische Universität München.

Bose, S.K. et al. (2014). Network Calculus for Time Sensitive Networks: A Survey. *IEEE Communications Surveys Tutorials*, 16(1).

Braden, R. et al. (1994). RFC1633: Integrated Services in the Internet Architecture: An Overview.

Bölöni, L. (2011). Petri Nets and Network Calculus: A Comparative Study. *IEEE Transactions on Industrial Informatics*, 7(4).

Cruz, R. (1991a). A calculus for network delay. I. Network elements in isolation. *IEEE Trans. on Information Theory*, 37(1), 114–131.

Cruz, R. (1991b). A calculus for network delay. II. Network analysis. *IEEE Transactions on Information Theory*, 37(1), 132–141.

David, R. and Alla, H. (2010). *Discrete, Continuous, and Hybrid Petri Nets*. Springer, Berlin, Heidelberg.

Desirena-López, G. et al. (2019). Thermal-aware Real-time Scheduling Using Timed Continuous Petri Nets. *ACM TECS*, 18(4), 36:1–36:24.

Farhi, N. et al. (2009). Road Traffic Models Using Petri Nets and Minplus Algebra. In *Traffic and Granular Flow '07*, 281–286. Springer.

Georgiadis, L. et al. (1996). Efficient Support of Delay and Rate Guarantees in an Internet. *SIGCOMM Comput. Commun. Rev.*, 26(4), 106–116.

Heidergott, B. et al. (2006). *Max Plus at Work: Modeling and Analysis of Synchronized Systems: A Course on Max-Plus Algebra and Its Applications*. Princeton University Press.

IEEE 802.1 (2022). IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks. *IEEE Std 802.1Q-2022 (Rev. of IEEE Std 802.1Q-2018)*, 1–2163.

IEEE 802.1 (n.d.). Time-Sensitive Networking Task Group. URL `https://1.ieee802.org/tsn/`.

Le Boudec, J.L. (1996). Network Calculus Made Easy. Technical report, École Polytechnique Fédérale de Lausanne.

Le Boudec, J.L. and Thiran, P. (2001). *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, Berlin, Heidelberg.

Lo Bello, L. and Steiner, W. (2019). A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems. *Procs. of the IEEE*, 107(6), 1094–1120.

Maile, L. et al. (2020). Network Calculus Results for TSN: An Introduction. In *2020 Information Communication Technologies Conf. (ICTC)*, 131–140.

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4), 541–580.

Rubio-Anguiano L.E. et al. (2020). Real time scheduler for multiprocessor systems based on continuous control using Timed Continuous Petri Nets. *IFAC-PapersOnLine*, 53(4), 371–377.

Rubio-Anguiano, L.E. et al. (2021). Maximizing Utilization and Minimizing Migration in Thermal-Aware Energy-Efficient Real-Time Multiprocessor Scheduling. *IEEE Access*, 9, 83309–83328.

Seol, Y. et al. (2021). Timely Survey of Time-Sensitive Networking: Past and Future Directions. *IEEE Access*, 9, 142506–142527.

Serna Oliver, R. et al. (2018). IEEE 802.1Qbv Gate Control List Synthesis Using Array Theory Encoding. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium*, 13–24.

Shenker, S. et al. (1997). RFC2212: Specification of Guaranteed Quality of Service.

Tang, S. et al. (2020). Modeling and Security Analysis of IEEE 802.1AS Using Hierarchical Colored Petri Nets. In *2020 IEEE Global Comm. Conf.*, 1–6.