Hindawi
Scientific Programming
Volume 2022, Article ID 5103199, 11 pages
https://doi.org/10.1155/2022/5103199

Research Article

High-Level Petri Nets-Based Modeling of Network Controlled Systems under Communication Constraints (Network-Induced Delay)

Khamsa Farah, Karim Chabir, and Mohamed Naceur Abdelkrim

Research Laboratory MACS LR16ES22, National Engineering School of Gabes (ENIG), University of Gabes, Gabes, Tunisia

Correspondence should be addressed to Khamsa Farah; farah.khamsa@hotmail.com

Received 25 April 2022; Revised 10 August 2022; Accepted 30 August 2022; Published 10 October 2022

Academic Editor: Pengwei Wang

Our work deals with the problems of communication delays, packets dropouts, and quantization errors in the signal transmitted in network-controlled systems (NCS). Although NCS provides a paradigm for adapting to frequent changes in the manufacturing industry, modeling and managing operations are difficult issues due to the complex interactions between system entities and the effect of the network-induced delay on the closed-loop system. Therefore, our paper is about detecting degradations resulting from the insertion of a network in the regulation loop, which can even lead in some cases to the destabilization of the NCS. This paper mainly studies the problem of the graphic modeling, based on the colored Petri nets (CPN), of the NCS under communication constraints. Therefore, models of the communication network (Ethernet) and system entities are presented. Then, the influence of network-induced delay is detected. Finally, as an idea for future work, we propose a solution based on an SDN controller in order to avoid precedent degradations.

1. Introduction

NCS belong to the class of real-time systems where the task values depend not only on the efficiency of the calculations but also on their uptime. Rapid technological development has increased, at a reasonable cost, the number of devices for implementing control and diagnostic algorithms sharing the same communication network [1, 2]. Real-time automation considers the limitations of computing resources computers and communication networks as design constraints. Specifically, the study of NCS deals primarily with the interaction between control systems and communication media [3]. A network-controlled system is a collection of computing devices that communicate with each other through a network. Sensors and actuators are devices for interacting with the physical world in a feedback loop [3–5]. This kind of system is closed via a real-time communication network that may be shared with other applications, as shown in Figure 1. Inserting a network into the control loop can provide unreliable and time-dependent service levels in terms of delays, vibrations, or losses, for example. Quality of service (QoS) can improve network behavior in real-time, but network behavior is still susceptible to interference, especially in wireless media, routing crossovers, and heavy traffic. On the other hand, network fluctuations can threaten the stability, integrity, and performance of modules in the physical environment.

A difficult problem in controlling orange network-based systems is the effect of network delay. The time required to read the measurement from the sensor and send a control signal to an operator on the network depends on network characteristics such as topology and routing diagrams. Therefore, the overall performance of a network control station can be greatly affected by network delays. The delay problem is exacerbated when data loss occurs during transmission. In addition, the delays not only degrade the performance of the network-based control system but can also destabilize the system [6–10].

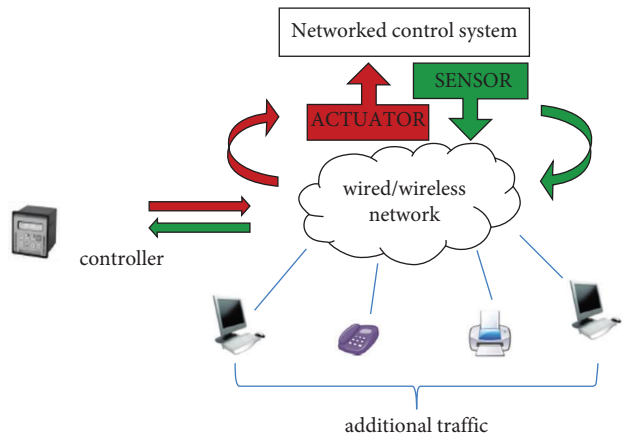

Figure 1: NCS architecture.

An original idea from [11] consists in proposing a parity relation-based FD system robust to the network-induced delay. Authors in [12] presented a new switched system model to describe the NCS with both delay and packet dropout. In [13, 14], authors have investigated the distributed control problem for a class of large-scale systems with communication constraints and topology switching. Strategies such as event-based communication and logarithmic quantization have been introduced to reduce the transmitted information. Besides, in [15], a novel networked predictive control algorithm based on k-order adaptive discrete-time sliding-mode control is proposed. Many other works have concentrated on network-induced constraints in NCS [16, 17]. Finally, authors in [18–20] have studied methods for robust stability and stabilization conditions for networked control systems with network-induced delay.

The general context of this paper is, therefore, network-controlled systems. NCS are systems where controllers, actuators, sensors, and other applications communicate through a communication network. To achieve the goal of designing a context-aware information system for NCS, models of the entities in the system are constructed based on colored Petri nets (CPN) and a multiagent system architecture in which each entity in the system is modeled as an agent to capture the interactions of entities in NCS. The use of colored Petri nets in our work is motivated by the fact that they are found to be an appropriate formal graphical language for modeling and analysis of concurrent and distributed systems. This is achieved by combining the strengths of Petri nets with the expressive power of high-level programming languages. Petri nets provide two constructions: the first is graphical for specifying synchronization of concurrent processes, and the second is a programming language for specifying and manipulating data values. CPN can be classed as a means of modeling and simulating network behavior. It provides a well-adapted and progressive framework for the representation and analysis of communications systems. For large systems, hierarchical colored Petri nets (HCPN) provide the possibility of modeling every part with a substitution transition, which is an abstraction of another model. That is to say, hierarchy is used to subdivide a model into different parts, which allows modular modeling. Such architecture must be able to represent the operations of storage, routing, classification, and scheduling in the network. There have been many results in theory and also in practical applications [21]. PN has been popular also for almost a half-century as a formalism and practical tool for modeling, simulating, and analyzing systems exhibiting behaviors of concurrency [22]. The first perspective of work related to PN is for Brauer (1980). From 1984 and for almost two decades, a significant part of the core of contributions to PN theory and applications was edited by Grzegorz Rozenberg. Most of those contributions came from informatics. Time was introduced in the field of PN in the middle of the seventies, when systems performance started to be considered. PN and time have been used in many works since 1973. This method is well suited for modeling the behaviour of distributed systems, but the absence of powerful structuring primitives has still been identified as a weakness. As Jensen says in [23], the absence of compositionality has been one of the main critiques raised against Petri net models. This has led to the development of hierarchical coloured Petri nets (HCPN), which essentially introduces a facility for building a PN out of subnets or modules. A simulation of the model has been conducted to demonstrate the practicality of the proposed method in terms of computation time and response time and in detecting the influence of the network on the system's stability. Then, to resolve this problem, we proposed, as an idea for future work, adopting a solution based on SDN controllers. This paper is structured as follows. Section 2 deals with the specifications of an Ethernet switch. Section 3 aims to present Petri net modeling. In section 4, we present the simulation results. Then, a discussion and a few promising areas that are open to future research are briefly explained in section 5. Finally, a general conclusion is given in section 6.

2. Specifications of Studied NCS

Our regulation loop, as shown in Figure 2, is made up of a controller, a process, a sensor, and an actuator that exchange data via three Ethernet switches, and we add two PCs communicating via the same network. We choose to work with three switches to increase the induced delay so that its effect is powerful on the performance of the system and remarkable in the representative curve of its response. There are two PCs as an additive traffic source to charge the network.

A network switch is a piece of equipment that connects several segments (cables or fibers) in a computer and telecommunications network and that makes it possible to create virtual circuits. Unlike a concentrator, a switch does not reproduce on all the ports each frame that it receives; it knows how to determine on which port it must send a frame, according to the destination address of this frame. Switches are frequently used as a replacement for hubs because they take up less space on the network. In the case of an IP/Ethernet network, a switch is not interested in the same OSI layer as the router; they use MAC addresses and IP

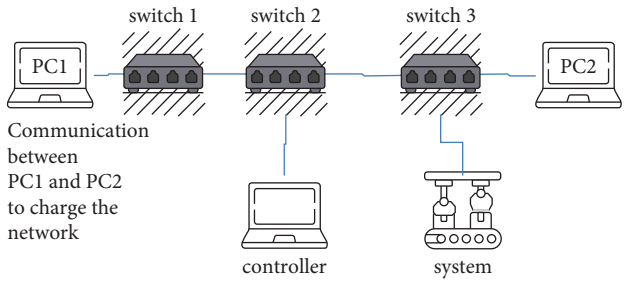

Figure 2: Studied system distribution.

addresses, respectively, to direct the data. Concretely, for an address which can be partially known, a frame is always sent on the same port, whatever the state of the traffic, once its routing and communication tables are filled.

The switch establishes and updates a table; in the case of the switch for an Ethernet network, it is the MAC address table, which indicates to it which ports to direct the frames intended for a given MAC address, according to the MAC source addresses of frames received on each port. The switch therefore dynamically constructs a table that associates port numbers and MAC addresses. The switch sends the frame back to the corresponding port. If the destination port is the same as that of the transmitter, the frame is not transmitted. If the recipient's address is unknown in the table. Then, the frame is treated as a broadcast (that is, it is forwarded to all ports on the switch except the send port). Each port has its own collision domain. The switch uses microsegmentation to divide the collision domains, one per connected segment. Thus, only the network interfaces directly connected by a point-to-point link request the medium. If the switch to which it is connected supports a full duplex, the collision domain is eliminated. There are four methods for forwarding packets:

(i) Direct mode (cut through): the switch simply reads the hardware address and transmits it as is. No error detection is performed with this method.

(ii) Store and forward mode: the switch buffers and usually performs a checksum operation on each frame before sending it (this mode is used in our work).

(iii) Fragment free: packets are passed at a fixed rate, allowing for simplified error detection. This is a compromise between the previous methods.

(iv) Automatic switching (adaptive switching): depending on the errors observed, the switch automatically chooses one of the three previous modes.

Ethernet is well known by network engineers in the industrial environment and provides an open and flexible communication system. Multiple research studies [24–26] show that Ethernet, with its advantages such as the nondeterministic access protocol and the propagation time of information in the network, is the solution to guaranteeing the properties of determinism and reactivity of the control. The switch modeled in our work is a Cisco Catalyst 2950 XL series switch. It is based on a shared memory of 8 MB. Once a frame is received as an input, it is directly copied into a global shared memory. The type of switches that we want to model offers two modes of classification of service, to which are added frame scheduling mechanisms. The first mode consists of recognizing the already labeled flows and directing them to the different queues according to their level of service. In the second mode, the network administrator can reclassify flows on the input port with a default value. Once these frames have been classified, they are routed to the appropriate output queue as observed in [27]. This switch has four queues per output port. One queue has an "absolute" priority (the highest priority) and is used to process real-time applications, and the other three are for low-priority packets. Note that there are as many output buffers as there are priorities. The modeled switch has the parameters summarized in Table 1.

3. Petri Nets Modeling

The colored Petri net (CPN) modeling method can describe a variety of resource types and execution logic, and it can be formally verified. The proposed methodology allows the construction of compact models for task scheduling problems. Moreover, a simulation process is possible within the constructed model, which allows the study of some performance aspects of the task allocation problem before any implementation stage.

3.1. Model of the Entire System. The network-controlled system as a whole is illustrated by the RDPCTH in Figure 3. Each abstraction transition (controller, Ethernet switch, actuator, process, sensor, and one additive traffic (PC1 and PC2) to charge the network) represents an RDPCT (or RDPCTH for the Ethernet switch abstraction transition). The places shown in this model represent the input and output ports between each module; however, the "retard" place is intended for delay detection. The inhibiting arc between transition and place means that the transition is validated only if the place does not contain any token. Its representation in colored Petri nets, specifically in the CPN Tools tool, is illustrated by a double-oriented arc (arrowed at both ends).

3.2. Ethernet Switch Modeling. Figure 4 shows the model of our Ethernet network based on CPN with all of its internal operations. The modeling and performance analysis of a switched Ethernet network architecture requires the ability to model the operations of storing, routing, classifying, and scheduling packets through which the flow passes, as detailed in [28, 29]. Namely, the storage in a FIFO input memory after receiving the frame. Then, the flow is confronted with the routing operation that sends the packets to the appropriate output queue. After that comes the operation of classification, modeled in Figure 5, which allows the selection of the packets according to their priorities in the appropriate output queues. Finally, these packets are transmitted to the output according to the scheduling algorithm implemented in the switch [24, 30].

Table 1: Switch parameters.

Parameters	Ethernet switch
Debit (Mbit/s)	10
Duration of a bit (microsecond)	0.1
Max data field size (byte)	1500
Min message size (byte)	64

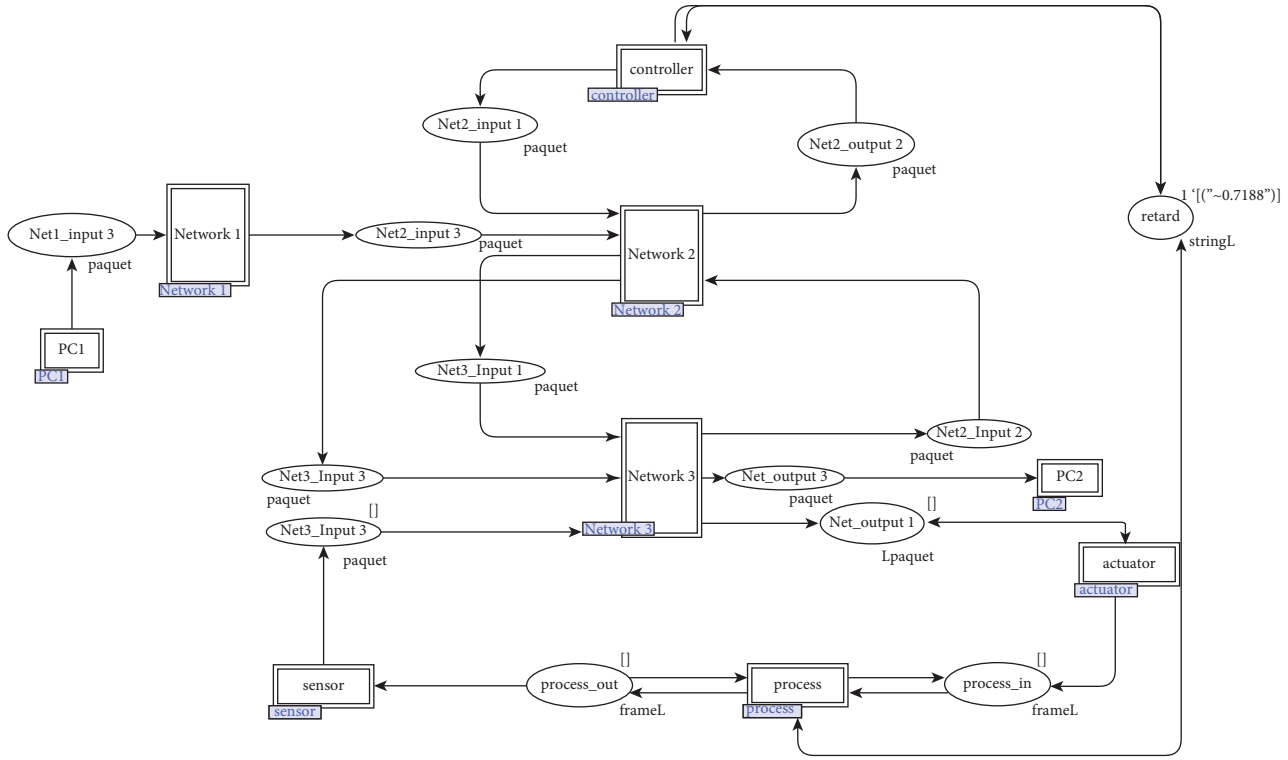

Figure 3: CPN modeling of NCS.

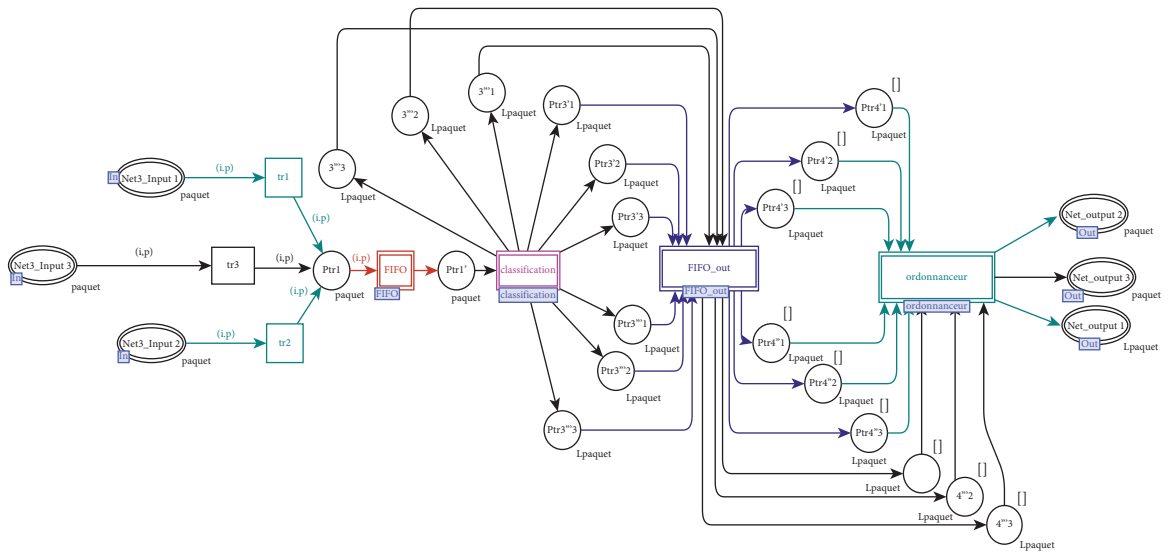

Figure 4: Model of Ethernet switch.

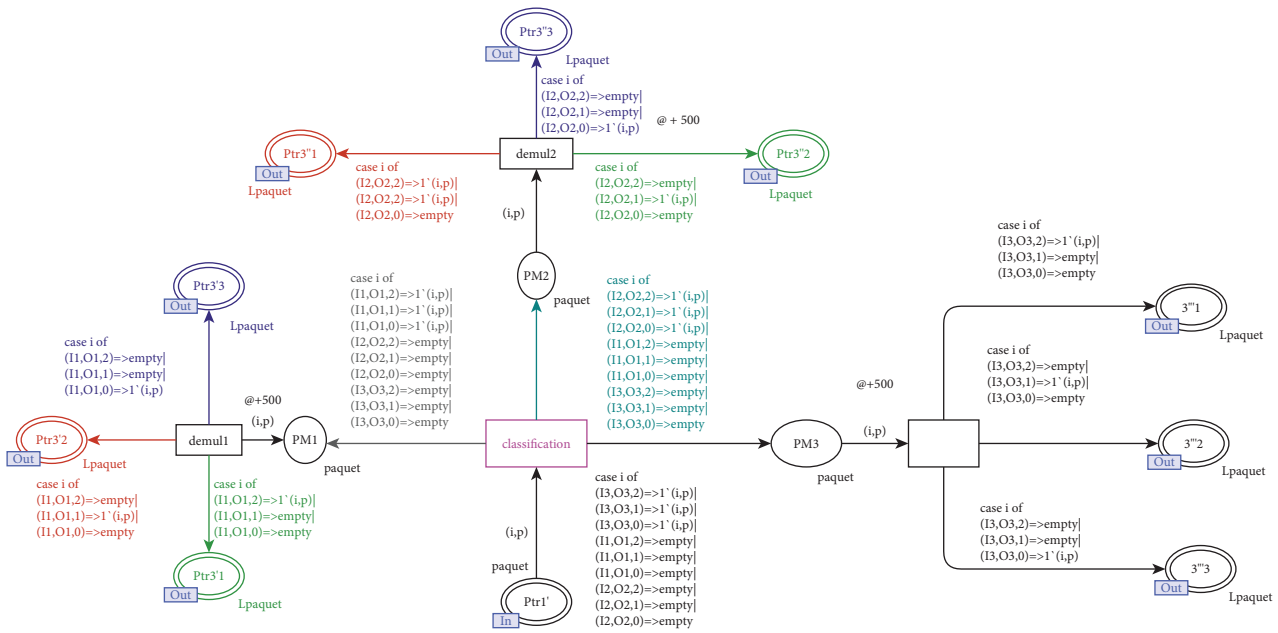

Figure 5: Classification modeling.

3.2.1. Scheduling Policy Modeling. The scheduling operation occurs at the output of the FIFO output queues. The frames are transmitted according to a weighted round robin scheduling algorithm.

Weighted round robin scheduling (WRR): assigns to each flow a normalized weight according to the average packet size of the flow and serves the tails (not empty) in turn and according to their weight (see Figure 6). This algorithm is based on a fair method (serve in turn and according to their weight). High-priority packets will be served until the desired number is reached and "w1" becomes 0, which allows moving to lower-priority packets as observed by [29].

3.3. Model of Detection Delay Mecanism (Process). To consider the delay induced by the Ethernet switch, a calculation procedure has been developed and attached to the model of the process.The command value calculated in the command module is sent to the switch module by the place port "net-input1" and in the delay calculation place, which is a process input port place (Figure 7). As soon as the actuator receives the frames, it sends them to the process through the process-in port. The time labels associated with the tokens that arrive at the process-in place and at the delay computation place are extracted using the intTime () function: fun intTime (i) = IntInf.toInt (time ()). This function translates the conversion of the time label, which is of the form @ + T (exp:@ + 5), into an integer. A subtraction operation is applied to these values thanks to the function subtract (i, y) (C-D transition), which is associated with a transition.

The difference is obtained as soon as it is executed. Then, this value can be reused from in the two following manners:

(i) First case: the times are greater than one or more sampling periods. As the transitions in "cpnTools" (the Petri nets simulator) are P-timed, when the plant transition is active, the token waits Te units of time to send the calculated state xk. However, imagine that the token carrying the information uk has a time tag (t1) whose value is greater than the time tag (t2) associated with the token in the timer. Then, the token in the timer must wait (t1-t2) units of time before the plant transition can be crossed. Then, the new calculated value xk is sent to the sensor after Te units of time. In short, the token in the timer will now have a label of (t1-t2) + Te, which may cause an erroneous shift in the data processing (Transition selection uk-xk). The xk-uk selection function associated with the transition selection uk-xk chooses the states to be taken into account for the calculation of the next state of the system.

(ii) Second case: knowing this value is useful for controlling the network to minimize delays.

4. Simulation Results

Consider a simple controlled system described by the following difference equation:

$$x(k+1) = Ax(k) + Bu(k),$$
$$y(k) = Cx(k). \quad (1)$$

This system is unstable without any command feedback, but the system is controllable.

The optimal command is given as follows:

$$u(k) = -Fx(k), \quad (2)$$

this state feedback command stabilizes the system and minimizes the quadratic criterion J, as detailed in [27].

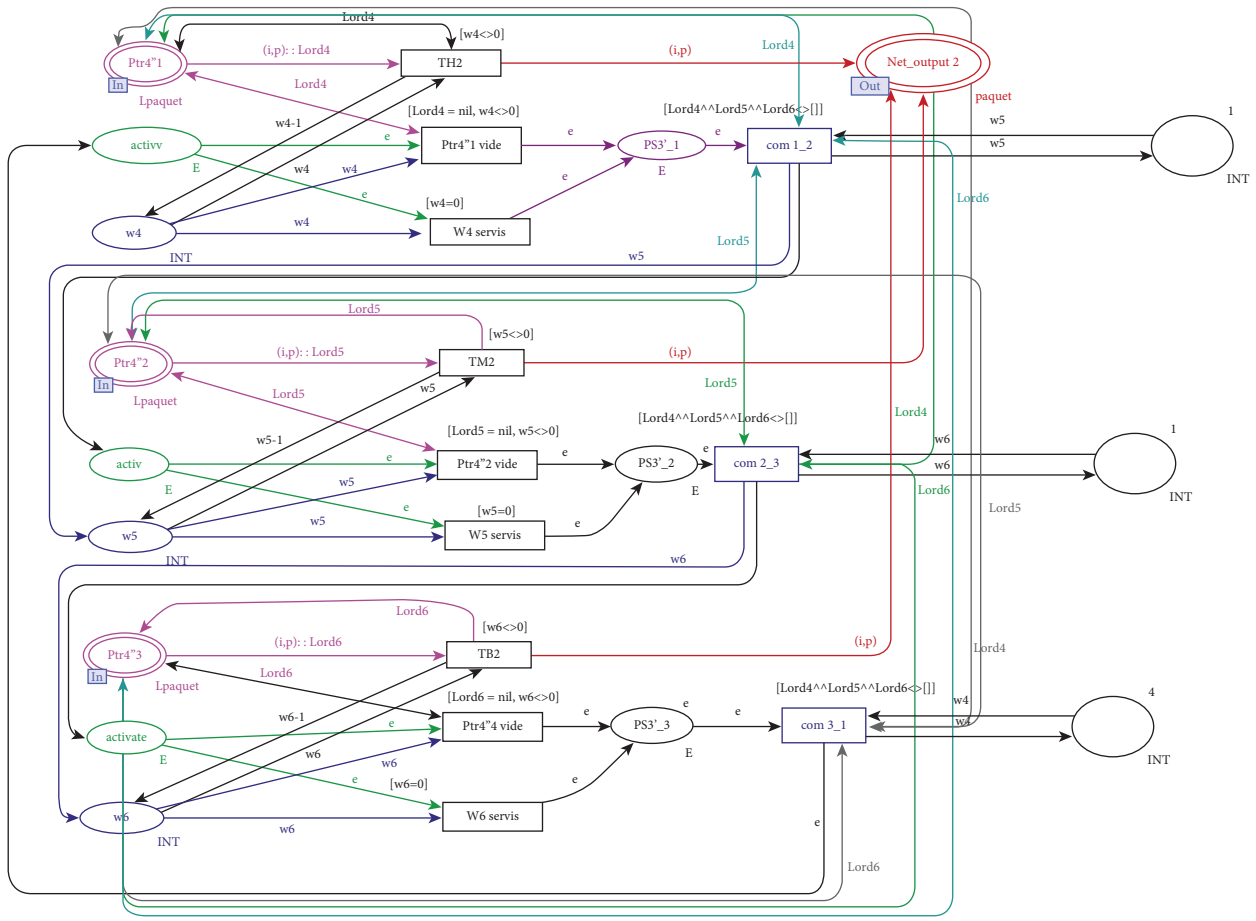

Figure 6: CPN model of WRR scheduler.

with: $A = 1.05$, $B = 1.8$, $F = 0.3594$. All variables, colors, and functions are declared as shown in Figure 8, Figure 9, Figure 10, and Figure 11. The simulation of the proposed models leads to the results presented in Figure 12 and Figure 13. The simulation parameters are given in Table 2

5. Discussion and Promising Areas

Figure 12 shows the response of the system without a network. the system follows the set point, and it behaves almost ideally. Figure 13 shows the response of the system, in the presence of the network, that has fluctuated, and the curve is characterized by overshoots before the state reaches the setpoint. We notice that the system losses its stability because of the delay induced by the network.

To overcome this annoying problem, we propose the use of SDN as a method of controlling the network to minimize the delay induced and to protect the system from its destabilizing effect. This solution consists of installing a WRR weight controller connected wire-to-wire to all the switches as shown in Figure 14 and which has the following functions:

(i) Detect the instability of the process (delay in the observation, normally faster because this observation is almost immediate).

(ii) Transmit the WRR weights to the various switches via the dedicated links (wire-to-wire) (configuration transmission delay), which is faster because configuration messages are not congested by other traffic.

(iii) Update the weights in the switches (delay in the configuration and in the treatment of the saturated buffers before returning to a stable system).

SDN is a networking technology focused on opening new possibilities in network management and coordination. This is important in future networks, where the virtualization of resources and network functions is the basic paradigm.

Many works propose SDN to control networks in a programmatic way, facilitate the deployment of new services and applications, as well as the tuning of network scheduling policies and performance. It represents a significant change in the way networks are architected, built, and managed.

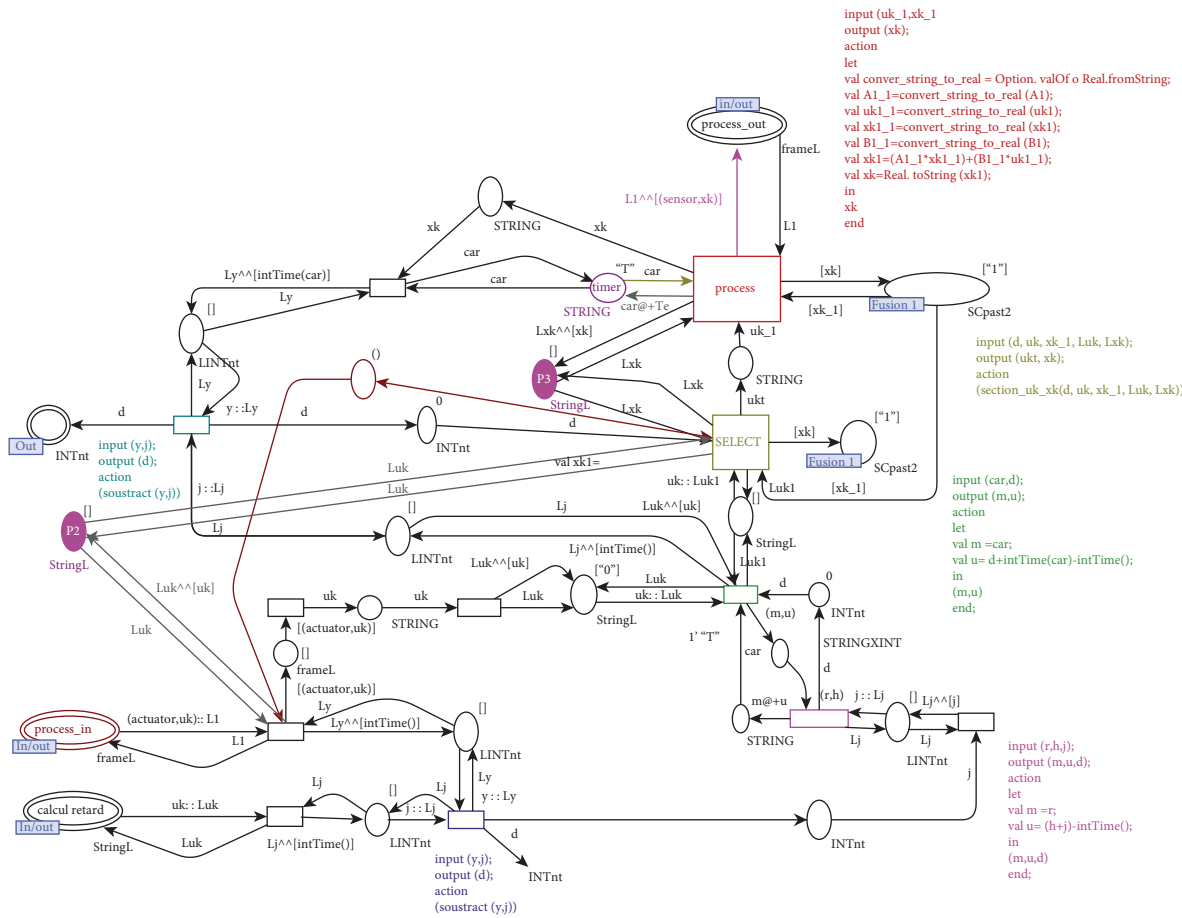

Figure 7: A process with detection delay mechanisms.

```
colset reference=list STRING;
var cons1:reference;
colset SCpast2=list data;
colset E=unit with e timed;
colsetE1=unit
colset frame= product recpt*data;
colset frameL=list frame;
var L1:frameL;
var w1,w2,w3,w4,w5,w6,w7,w8,w9:INT;
colset LINT= list INT;
var Lw1,Lw2,Lw3,Lw4,Lw5,Lw6:LINT;
fun picko(i::Lord)=
  let
  val c = discrete (0, (List.length(i::Lord))-1)
  in
  SOME(List.nth((i::Lord),c), List.take((i::Lord),c)^^List.drop((i::Lord),c+1))
  end
  |picko[] = NONE;
fun intTime(i)=IntInf.toInt(time());
```

colset frame= product recpt*data;

Figure 8: Colors, variables, and functions declarations.

```
fun soustract (y:INTnt, j:INTnt)=
  let
  val d=(y-j);
  in
  d
  end;
colset STRINGXINT=product STRING*INT;
```

Figure 9: Declaration function soustract.

```
fun selection_uk_xk(d:INTnt, uk:data, xk_1:data, Luk:dataL, Lxk:dataL)=
  let
  val n=length (Luk);
  val a=length (Lxk);
  in
  if d=0 then (if n=1 then (uk, xk_1)
  else if n=2
  then (List.nth(Luk, (n-2)), List.nth(Lxk,(a-1)))
  else if n>=3 then (List.nth(Luk, (n-1)), List.nth(Lxk, (a-1)))
  else (uk,xk_1))
  else if d<0 andalso d>(~Te) then
  (if n=1 then (uk,xk_1) else if n>=2 then
  (List.nth(Luk, (n-2)), List.nth(Lxk, (a-1)))
  else (List.nth(Luk, (n-2)), List.nth(Lxk, (a-1))))
  else if d=(~Te) then (if n= 1 then(uk,xk_1)
  else if n =2 then
  (List.nth(Luk, (n-2)), List.nth(Lxk, (a-1)))
  else if n=3 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (a-1)))
  else if n>=4 then
  (List.nth(Luk, (n-2)), List.nth(Lxk, (a-1)))
  else (uk,xk_1))
  else if d>(~2*Te) andalso d<(~Te) then
  (if n=1 then (uk, xk_1) else if n=2 then
  (List.nth(Luk,(n-2)), List.nth(Lxk, (a-1)))
  else if n>=3 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (a-1)))
  else (uk,xk_1))
  else if d = (~2*Te) then (if n=1 then (uk, xk_1)
  else if n=2 then
  (List.nth(Luk,(n-2)), List.nth(Lxk, (a-1)))
  else if n=3 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (a-1)))
  else if n=4 then
  (List.nth(Luk, (n-4)), List.nth(Lxk, (a-1)))
  else if n>=5 then
  (List.nth(Luk, (n-3)), List.nth(Lxk, (a-1)))
  else (List.nth(Luk, (n-4)), List.nth(Lxk, (a-1))))
  else (uk, xk_1)
  end;
```

Figure 10: Declaration function select.

```
Standard declarations
  colset UNIT
  colset INT = int;
  var s,a,l11,l22,l33,l44,l55,l66,b,u,h,n:INT;
  colset INTnt=INT;
  var j,d,y,q:INTnt;
  colset LINTnt=list INTnt;
  var Ly,Lj,Ld:LINTnt;
  val Te=1;
  val A1="1.05";
  val B1="1.8";
  val K1="0.3594";
  colset BOOL = bool;
  colset STRING = string timed;
  var m,r,cons,car:STRING;
  colset stringL= list STRING;
  colset inp = with I1| I2| I3;
  colset recpt= with sensor|actuator|process;
  colset outp= with O1|O2|O3;
  val H=2;
    val M=1;
    val B=0;
  colset data =STRING;
  var uk,xk, xk_1,uk_1,p,ukt:data;
  colset prio = int with B..H;
  var pr:prio;
  colset dataL=list data;
  var Luk,Lxk,L,Luk1:dataL;
  colset paquetd= product inp*outp*prio;
  colset paquet=product paquetd*data timed ;
  var i:paquetd;
  colset Lpaquet=list paquet;
  colset ordo= INT;
  var g:ordo;
  colset paquet_ordo=product paquet*ordo timed;
  var ord:paquet_ordo;
  colset Lpaquet_ordo=list paquet_ordo;
  var resto,Lord,Lord1,Lord2,Lord3,Lord4,Lord5,Lord6,Lord7,Lord8,Lord9,
    Lord10,Lord11,Lord12,Lord13,Lord14,Lord15,Lord16,Lord17,Lord18,
    Lord19,Lord20,Lord21,Lord22,Lord23,Lord24,Lord25,Lord26,Lord27,
    Lord28,Lord29,Lord30,Lord31,Lord32,Lord33,Lord34,Lord35,Lord36,
    Lord37,Lord38,Lord39,Lord40:Lpaquet;
  colset reference=list STRING;
  var cons1:reference;
```

Figure 11: Colors and variables declaration.

SDN is therefore more widely recognized today as an architecture that opens up the network to applications [26, 31]. It enables the network to better identify the applications transported, so it can better manage them (quality of service, security, traffic engineering, etc.).

Despite it still being restricted in the enterprise environment, the widespread replacement of traditional networks with SDN offers the possibility of controlling the network based on application requirements. One of the main problems that arise when an emergency happens is minimizing the delay time in resource forwarding so as to reduce both human and material damages [32]. Solutions focus on moving a switch between two controller instances. Careful planning is required when migrating multiple switches due to controller resource constraints and to ensure minimal downtime on the network. The experiments in [33] show that the delay of the emergency traffic improves by 33 percent when that solution is running. In [34], the authors present a new security system for networks based on SDN, which can be easily integrated with the existing network infrastructure and provide security for all network components. This system enables the creation of additional boundaries within the network to provide a multilevel defense system, solves a single point of failure problem, and

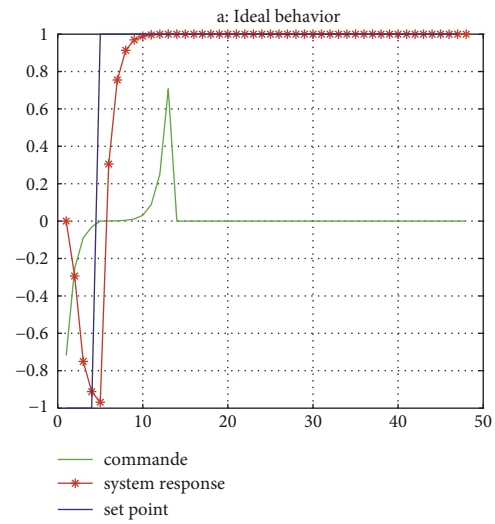

Figure 12: Ideal behavior.

facilitates the protection of the network from attacks and malicious users. Poster [35] also presents a model and a solution for migration scheduling, taking a set of switch migrations as input and generating a migration schedule

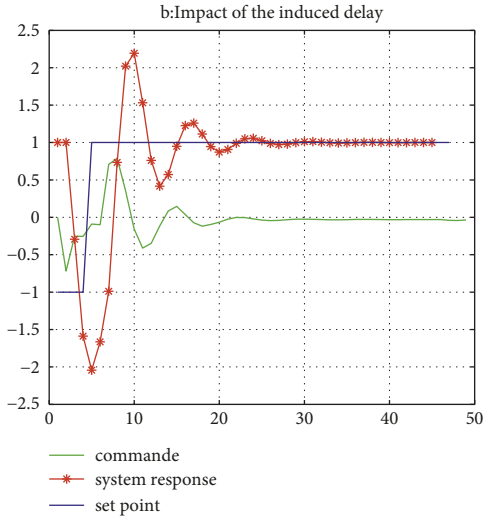

Figure 13: Behavior under network-induced delay.

Table 2: Simulation parameters.

Parameters	Real values	Model values (ns)
Latency	500 ns	500
Producer period	1 ms	1000000
Consumer period	1 ms	1000000
Delay due to store and forward mode	51.2 μs	51200

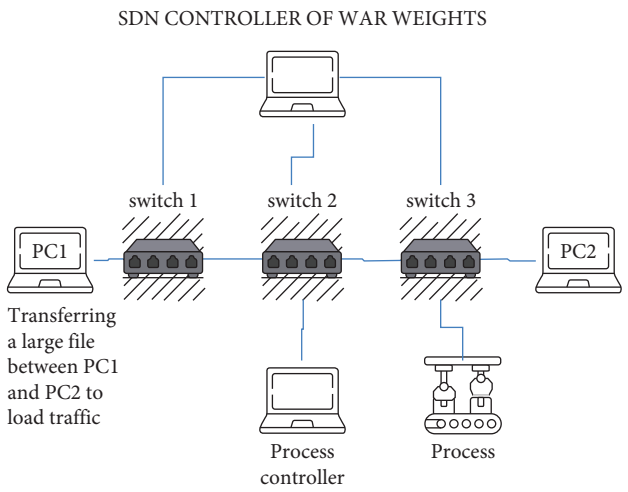

Figure 14: Proposed architecture based on an SDN controller.

with respect to controller resource and service interruption constraints. Additionally, in [36], a hybrid network control system is developed and discussed.

6. Conclusion

This article illustrates high-level Petri net-based graphical modeling of a network-controlled system and the detection of the network-induced delay's impact on system stability. The distributed feedback control scheme is designed based on the physical connectivity of the subsystems. The local controller allows the exchange of control input data and sensor information between them to reduce disturbances resulting from physical interconnections. The stability of the system is examined under network-induced delay. The simulation results showed that an induced delay can infect the normal behaviour of the system. The proposed model has succeeded in detecting the degradation caused by the induced delay. Therefore, we consider that enabling a protection mechanism to learn from experience and use existing knowledge about a delay to infer and prevent delay influence is an important and potentially fruitful future research area. We also believe that the development and deployment of network security policies are essential in networks with a dynamic environment.

Abbreviations

NCS: Network controlled systems
CPN: Colored Petri nets
PN: Petri nets
WRR: Weighted round Rrobin
FIFO: First in first out
SDN: Software defined network.

Appendix

Formal presentation

A Petri net is a bipartite graph of which we particularize the two families of vertices: the places and the transitions. As in any bipartite graph, an arc never connects two vertices of the same family. States are represented by circles, while transitions are represented by lines or rectangles. Their dynamics are derived from the marking or the distributed state.

Definition of colors

The defined colors represent the Ethernet frame with some abstractions, depending on the requirements and the assumptions made. The elements that we have modeled are chosen according to our needs and their relevance. These elements are the destination address, the source address, the tag representing the level of priority of the frame, and the data to be transmitted. The transmission mode is selected as store and forward, which suggests that the frames are not erroneous. The frame size is assumed to be constant and equal to 64 bytes (without the preamble and SFD). The colors and variables needed for modeling this network and its internal modules in our work are declared as follows:

Colset inp = with I1| I2;
Colset outp = with O1| O2;
Colset prio = int with B···H;
Colset data = INT;
Val $H = 2$;
Value $M = 1$;
Val $B = 0$;
Colset paquet = product inp * outp* prio timed;
Var i: paquet;
Colset Lpaquet = list paquet

Colset reference = list STRING
Colset SCpast2 = list data
Colset E = UNIT with e timed
Var xk,uk, xk 1, uk 1,uk1,xk1,p: data
Var Lord, Lord1, Lord2, Lord3, Lord4,Lord5, Lord6: Lpaquet
Var cons1: reference
Var w1,w2,w3,w4,w5, w6: INT
Simulation settings
The destination address is defined by the color "outp," the source address by the color "inp," and the tag expressing the priority level of the frame by the color "prio." Here three levels of priority are defined: $H = 2$ (high priority associated with the command frame), $M = 1$ (medium priority), and $B = 0$ (low priority) "paquet" is composed of a complex color made using the product function of several colors.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] B. C. de Oliveira, I. D. Melo, and M. A. Souza, "Bad data detection, identification and correction in distribution system state estimation based on pmus," *Electrical Engineering*, vol. 104, no. 3, pp. 1573–1589, 2021.

[2] A. P. Mukherjee, P. K. Kundu, and A. Das, "Classification and localization of transmission line faults using curve fitting technique with principal component analysis features," *Electrical Engineering*, vol. 103, no. 6, pp. 2929–2944.

[3] F. Y. Wang and D. Liu, *Networked Control Systems: Theory and Applications*, Springer, Berlin, Germany, 2002.

[4] R. Alur, *Principles of Cyberphysical Systems*, MIT press, London, England, 2015.

[5] F. Sicard, E. Zamai, and J. M. Flaus, "Cyberdefense of industrial control systems: a filter approach based on the distance to critical states for securing against cyberattacks," in *Cesar Data protection against the Cyber Threat*, 2017.

[6] Y. Z. Lun, A. DoInnocenzo, and F. Smarra, "State of the art of cyber physical systems security an automatic control perspective, m.d.di," *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[7] J. P. Georges, "*Quality of service and network controlled systems qds and scr*," in *Real Time Summer School*, Tech. report, 2011.

[8] K. Chabir, T. Rhouma, J. Y. Keller, and D. Sauter, "State filtering for networked control systems subject to switching disturbances," *International Journal of Applied Mathematics and Computer Science*, vol. 28, no. 3, pp. 473–482, 2018.

[9] K. Chabir, M. A. Sid, and D. Sauter, "Fault diagnosis in a networked control system under communication constraints a quadrotor application," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 4, pp. 809–820, 2014.

[10] X. Ge, F. Yang, and Q. L. Han, "Distributed networked control systems a brief overview," *Information Sciences*, vol. 380, pp. 117–131, 2017.

[11] H. Ye and S. X. Ding, "Fault detection of networked control systems with network induced delay,"vol. 1, pp. 294–297, in *Proceedings of theICARCV 8th Control Automation Robotics and Vision Conference*, vol. 1, pp. 294–297, IEEE Press, Kunming, China, 2004.

[12] W. A. Zhang and L. Yu, "Modelling and control of networked control systems with both network induced delay and packet dropout," *Automatica*, vol. 44, no. 12, pp. 3206–3210, 2008.

[13] D. Zhang, S. K. Nguang, and L. Yu, "Distributed control of large scale networked control systems with communication constraints and topology switching," *IEEE Transactions on Systems Man and Cybernetics Systems*, vol. 47, no. 7, pp. 1746–1757, 2017.

[14] E. Liu, K. Fridman, and Y. Xia, *Networked Control under Communication Constraints, a Time Delay Approach*, Springer Nature, Berlin, Germany, 2020.

[15] M. Li and Y. Chen, "Robust tracking control of networked control systems with communication constraints and external disturbance," *IEEE Transactions on Industrial Electronics*, vol. 64, pp. 4037–4047, 2017.

[16] C. Tan, L. Li, and H. Zhang, "Stabilization of networked control systems with both network induced delay and packet dropout," *Automatica*, vol. 59, pp. 194–199, 2015.

[17] L. Zhang, H. Gao, and O. Kaynak, "Network induced constraints in networked control systems, a survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 403–416, 2013.

[18] T. Zhao, M. Huang, and S. Dian, "Robust stability and stabilization conditions for nonlinear networked control systems with network induced delay via t s fuzzy model," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 3, pp. 486–499, 2021.

[19] R. Wang, H. Jing, J. Wang, M. Chadli, and N. Chen, "Robust output feedback based vehicle lateral motion control considering network induced delay and tire force saturation," *Neurocomputing*, vol. 214, pp. 409–419, 2016.

[20] X. Zhang, Y. Zheng, and G. Lu, "Stochastic stability of networked control systems with network induced delay and data dropout," in *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 5006–5011, San Diego, CA, USA, December 2006.

[21] T. Miyamoto and S. Kumagai, "A survey of object oriented petri nets and analysis methods," *IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88-A, no. 11, pp. 2964–2971, 2005.

[22] M. Zhou and N. Wu, *System Modeling and Control with Resource Oriented Petri Nets*, CRC Press, Boca Raton, Florida, United States, 2018.

[23] K. Jensen and G. Rozenberg, *High Level Petri Nets, Theory and Application*, Springer Science and Business Media, Berlin, Germany, 2012.

[24] B. Brahimi, C. Aubrun, and E. Rondeau, "Modelling and simulation of scheduling policies implemented in ethernet switch by using coloured petri nets," *Emerging Technologies and Factory*, Automation, ETFA 2006, IEEE Conference, 2006, pp. 667–674.

[25] D. Thiele and R. ErnstMaryak, "Formal analysis based evaluation of software defined networking for time sensitive ethernet," in *Proceedings of the Design, Automation Test in Europe Conference, DATE 2016*, pp. 31–36, Arlington VA, USA (Arlington VA, USA), 2016.

[26] M. Herlich, J. L. Du, F. Schrghofer, and P. Dorfinger, "Proof of concept for a software defined real time ethernet," in *Proceedings of the 21st International Conference on Emerging Technologies and Factory Automation, ETFA 2016*, pp. 759–762, Berlin, Germany, 2016.

[27] B. Brahimi, *Proposal of an Integrated Approach Based on High-Level Petri Nets to Simulate and Evaluate Networked Controlled Systems*, Tech. report, Henri Poincare University, Nancy I,France, 2007.

[28] K. Farah, K. Chabir, and M. N. Abdelkrim, "Colored petri nets for modeling of networked control systems," in *Proceedings of the 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering, STA 2019*, pp. 226–230, Sousse, TUNISIA, 2019.

[29] K. Farah, K. Chabir, and M. N. Abdlkrim, "A petri nets modeling of scheduling policies in ethernet switch," in *Proceedings of the International Conference on Signal, Control and Communication , SCC 2019*, pp. 30–34, Hammamet, TUINISIA, 2019.

[30] B. Brahimi, C. Aubrun, and E. Rondeau, *Network Calculus Based Fdi Approach for Switched Ethernet Architecture*, Supervision and Safety of Technical Processes 2006, Fault Detection, pp. 312–317, 2007.

[31] D. Henneke, L. Wisniewski, and J. Jasperneite, "Analysis of realizing a future industrial network by means of software defined networking (sdn)," in *Proceedings of the 2016 IEEE World Conference on Factory Communication Systems, WFCS 2016*, pp. 1–4, Aveiro, Portugal, 2016.

[32] T. Feng, J. Bi, and K. Wang, "Allocation and scheduling of network resource for multiple control applications in sdn," *China Communications*, vol. 12, no. 6, pp. 85–95, 2015.

[33] A. Rego, L. Garcia, S. Sendra, and J. Lloret, "Software defined network based control system for an efficient traffic management for emergency situations in smart cities," *Future Generation Computer Systems*, vol. 88, pp. 243–253, 2018.

[34] F. Nife, Z. Kotulski, and O. Reyad, "New sdn oriented distributed network security system," *Appl. Math. Inf. Sci*, vol. 12, no. 4, pp. 673–683, 2018.

[35] M. A. Beiruti and Y. Ganjali, "Migration scheduling in distributed sdn controllers," in *Proceedings of the IEEE 27th ICNP*, pp. 1-2, 2019.

[36] F. Kuliesius and M. Giedraitis, "Sdn legacy hybrid network control system," in *Proceedings of the 2019 Eleventh International Conference on Ubiquitous and Future Networks ICUFN*, pp. 504–509, 2019.