Journal of Universal Computer Science, vol. 18, no. 5 (2012), 676-703
submitted: 3/4/11, accepted: 28/2/12, appeared: 1/3/12 © J.UCS

Modeling and Performance Evaluation of a Contract-based Electronic Signature Process

Ahmed Nait-Sidi-Moh [1]
(Université de Picardie Jules Verne, Saint Quentin, France
ahmed.nait-sidi-moh@u-picardie.fr)

Mohamed Bakhouya
(Aalto University, Shool of Engineering, Aalto, Helsinki, Finland
mohamed.bakhouya@aalto.fi)

Wafaa Ait-Cheik-Bihi
(Université de Strasbourg, 67000 Strasbourg, France
aitcheikbihi@unistra.fr)

Jaafar Gaber
(Université de Technologie de Belfort Montbéliard, Belfort, France
jaafar.gaber@utbm.fr)

Abstract: Distributed systems become ubiquitous by allowing users access to a wide range of services at any time, anywhere, and from a variety of devices. In these open environments where there are many opportunities for both fraudulent services and misbehaving clients, service discovery systems are subject to security challenges. Controlling services' access is one of the fundamental issues that must be faced in the context of service discovery in distributed and open environments. Therefore, secure accesses and utilization of available services must be ensured for users. In our previous work, a contract-based approach for controlling the service access in a distributed computing context was presented. In this paper, we address the purpose and the usage of digital signature on negotiated electronic queries between a server and clients in service discovery systems and web service composition. The paper discusses the combined use of Timed Event Graphs and $(max, +)$- algebra to model, evaluate and optimize the performance of the signature process and client requests validation by a service provider (server). Based on an optimization resource allocation algorithm, an improvement study of the quality of service offered to the clients, in terms of waiting times and validation of their requests, is proposed. The results are reported and show the efficiency of the use of the proposed formal tools for performance analysis, evaluation and tuning of the considered process.

Key Words: Web service access control; Electronic exchanges; Modeling, Evaluation and simulation; Dimensioning and improvement; Petri nets and $(max, +)$-algebra.

Category: G.1.3, G.2.1, F.4.3, E.3, H.4.0, H.5.1

[1] Corresponding author: ahmed.nait-sidi-moh@u-picardie.fr, Tel. +33 3 23 62 89 44

1 Introduction

Service discovery and secure access are important issues in distributed networks. Given a user request, a service discovery mechanism should locate and return a set of server addresses (i.e., providers) that match the description of the requested service and wherein adequate services are located [Gidron et al. 2001, Bakhouya and Gaber 2008]. In open and heterogeneous environments, to provide high confidence to users, these systems must have the capability of authenticating users and service providers, verifying the integrity of services, protecting the information confidentiality, controlling the access to services based on security policies, and detecting malicious services and users. However, very little studies have been done regarding the access control problem [Gidron et al. 2001]. More precisely, the main problem is how to manage access policies to disparate services that are not under the control of a single system administrator [Firozabadi and Sergot 2002]. A computational contract that defines the services exchanged between the client and server is required. Contracts also need to be negotiated and signed to ensure that both client and server accept their terms [Ruiz-Martínez et al. 2009], [Nait-Sidi-Moh et al. 2008].

In this paper, we focus on the modeling and the evaluation of the electronic exchanges and associated processes, including electronic authentications and signatures, which represent an ever-increasing interest in the field of security in information systems [Ninham 2004], [Wack et al. 2006], [Wu et al. 2009]. They become a vital component of emerging electronic business infrastructures. More precisely, we address the purpose and the usage of digital signatures and associated processes on negotiated electronic queries between a server and clients in service discovery systems and web service composition.

A unified methodology based on Petri nets (PN) combined with $(max, +)$-algebra is introduced to model and evaluate the performance of a distributed interactive system. Both formalisms have been chosen because of the specific advantages they present. Furthermore, the choice of these two complementary formalisms is motivated by their wide use to model and analyze distributed and concurrent systems. Among the research developed in the literature focusing on these tools, we mention [Murata 1989], [Wack et al. 2006], [Gaubert 1992], and [Collart-Dutilleul 2008]. PN and its several extensions allow both qualitative and quantitative analysis of modeled systems. $(Max, +)$-algebra enables the description and specification of the systems behavior with simple and linear equations that can be easily solved and handled.

The combined use of these formalisms is well-known in the literature, but the novelty of our approach is their use for modeling and analysis of web-services access control and composition. More precisely, our motivation using these formalisms is to describe, in a straightforward and effective way, the behavior of a distributed interactive system, by analyzing and verifying its quantitative and qualitative properties. Through a case study, we show that allying PN with $(max, +)$-algebra is not only a powerful methodology for specification and modeling, but also an adequate tool for performance analysis and behavior prediction.

Exploiting the obtained graphical and mathematical models, allow the prediction of several performance metrics that can be used, for example, to detect thresholds of system saturation and give solutions to elevate it. In addition, we propose a dimensioning and improvement study with the aim to optimize the process operation using minimum resources.

The remainder of the paper is structured as follows. Section 2 is devoted to a state-of-the-art on the use of Petri nets and $(max, +)$-algebra for the modeling and the evaluation of complex systems. Section 3 presents an overview of the web service access control model for discovery systems. In Section 4, the modeling and analysis using a Timed Event Graph and $(max, +)$-algebra techniques as well as the performance evaluation results are reported. Section 5 presents how system dimensioning can be carried out for performance tuning and improvement. Conclusions and future works are given in Section 6.

2 Related work

Modeling, performance analysis and evaluation of distributed interactive systems are issues arousing an ever-increasing interest in many fields, such as computer and communication systems, manufacturing systems, and transport systems. Different formalisms and tools have been proposed for modeling and analysis of these systems. We focus in our study on Petri Nets [Collart-Dutilleul 2008] and dioid algebra [Baccelli et al. 1992]. These tools, which constitute the most popular formalisms used in the past decades, are developed for the specification, performance evaluation and prediction issues. These tools and other used tools in the literature can be classified, based on the paradigm used (i.e., notation used to describe the system behavior), into two categories: state-transition and algebraic methods. In state-transition models, the system behavior is specified as a transition relation on the set of states. An example of models using this paradigm is Petri Net [Nait-Sidi-Moh and Wack 2005]. Different extensions of ordinary Petri Nets have been developed by adding some features as new characteristics of systems to be modeled and evaluated. For example, Timed Petri nets [Collart-Dutilleul 2008] allow timing aspects to be integrated into the graphical model. This model has some features that make it suitable for modeling and validation of a system's behavior. It is a graphical tool for describing states and transitions and enabling the modeling of concurrency, synchronization, scheduling, and parallelism. This tool also allows, through simulation softwares, e.g. [VisualObjectNet++ 2012], to simulate system behavior from its specification, to the verification and validation of its properties.

The tools mentioned previously provide powerful techniques for modeling, evaluating and analyzing deterministic systems. However, many research works and tools have been developed to model the stochastic behavior of these systems. Among them, we mention stochastic Petri nets [Ajmone Marsan et al. 1998], $(max, +)$ stochastic algebra [Wendell-H-Fleming 2004], PRISM model checker [Kwiatkows et al. 2002], Markov theory [Baala et al. 2003], queuing networks [Brand and Begin 2009] and Network Calculus [Bakhouya et al. 2009]. The common point of these tools is the modeling and analysis of probabilistic systems with different application domains such as decision making, fault-tolerant systems, queuing management systems, traffic networks, etc. In our case one of these stochastic tools may be used to model the stochastic arrivals of contracts for signature, or also to analyze the performance of each involved actor on the signature process. Nevertheless, our choice is to use the first class of models, namely deterministic models, to analyze and evaluate the system behavior. Performance modeling and evaluation of systems with stochastic behavior are out of scope of the work presented in this paper.

Analytically, system behavior can be described as a set of operations or functions representing system events. In this paradigm, events and their relationship are represented by mathematical equations. An example of methods using this paradigm is $(max, +)$-algebra, which is very powerful to perform quantitative evaluation. This algebra has been used in the last years to evaluate the performance of discrete event systems (DES) [Gaubert 1992]. However, to the best of our knowledge, the power of this modeling algebra has been little exploited to evaluate the performance of computer and distributed systems and very few practical applications have been reported. It is difficult to find a unified method to validate the qualitative properties as well as quantifying performance metrics of the studied system. We believe that combining Petri Net and $(max, +)$-algebra into a unified methodology for formal verification and analytical performance evaluation can eradicate the above limits.

In this paper, we demonstrate how this methodology can be used for the validation of qualitative properties as well as performance analysis, evaluation, prediction, and tuning by considering the contract signature process as a case study. We underline that, in our previous work [Nait-Sidi-Moh et al. 2008], we have proposed a solution which meets the secured access for controlling services use in service discovery systems. This solution is based on contracts that define the rules and terms to be respected when exchanging messages between client and servers [Ninham 2004]. This process defines an exchange of a contract between clients and server to ensure that both of them accept their terms by signing it.

3 Web service access control: an overview

The life-cycle of the web service access control is composed of five phases: service location, negotiation of contract, contract signature, service activity, and contract termination and validation [Ruiz-Martínez et al. 2009], [Ben Mbarka 2011], [Gidron et al. 2001], and [Nait-Sidi-Moh et al. 2008]. In the first phase, a client wishing to purchase access to a service must locate potential service providers that meet his requirement.

In the second phase, a client enters into negotiation with one or more of these potential service providers, to see if they can agree on mutually acceptable terms for the required service. More precisely, the client seeking to use the service must negotiate with the providers offering the available service that match his needs. The outcome of the second phase is specifying the contract terms that both the client and server accept, such as price, delivery date, etc.. This agreement could be expressed using digital signatures to prove acceptance and validity of the contract.

After describing the contract terms, the client and the server should examine and sign it. This stage constitutes the third life-cycle phase. A copy of the contract together with both signatures proves that it was accepted. This represents the guaranteed permissions for accessing the offered service.

Any exchange of messages between the client and server must intrinsically include security information used to verify their integrity, authenticity and perform access rights checks. Historically, modern asymmetric cryptography is based on public and private key [Adams et al. 2005], [Wack et al. 2006]. The aim of electronic signatures is to verify the integrity of signed contents and their authenticity rather than searching for information confidentiality. Research work about this issue is proposed in [Gondrom and Fischer 2010].

In the fourth phase, a contract is eventually regulated through payment messages before using resources. The discovery server expects that the client will not attempt to access other services than those are required, and expects that the server will offer its services. Moreover, the transaction may be automatically monitored, and parties would be warned if any behavior outside the agreed terms of the contract takes place. It is worth noting that a contract can also be modified on-the-fly with a new contract in order to rectify a contract violation such as resource loss.

The fifth and final phase consists of optional termination requests or a final termination of the contract. This ensures that services are reserved from the time the contract is signed until its expiration. Once the contract is expired no service is valid, if needed, a new contract must be signed between the client and the provider.

All these phases are controlled by asynchronous messages for requesting actions and exchanging signatures between the client and the server. Many research efforts so far have focused on different aspects of contract management such as contract specification and languages and contract life-cycle management (i.e. service location, negotiation of contract, formation and examining of the contract, activity, and termination). However, in the best of our knowledge, little research focuses on the issue of the contract signature especially in service discovery systems.

4 Problem statement and modeling approach

In this section, we present the modeling and performance analysis and evaluation using $(max, +)$-algebra of the contract signature process made between the client and the service provider. We assume that the negotiation phase is already accomplished and will not be taken into account in the modeling and the evaluation studies. It is worth noting that we limit ourselves to a simple signature process, but the proposed study may be used for more complicated and specific cases. The contract signature process is first modeled using a Timed Event Graph (TEG), a subclass of PN, and then described by a $(max, +)$-linear system. These models enable us to easily derive some properties of the studied process such as correctness, monotony, reachability, etc. [Nait-Sidi-Moh and Wack 2005]. The main objective of using TEG with $(max, +)$-algebra is to be able to describe, in a straightforward and effectiveness way, the behavior of dynamic systems by mathematical and linear equations in order to evaluate certain of its performance metrics.

4.1 Modeling with TEG

In order to verify and validate the contract signing process, we represent its behavior by a graphic-based model using the TEG formalism. In this model, transitions represents the events (e.g. signature, authenticity, hash-coding) and their firing represent the occurrence of these events. A PN is a graph composed of two kind of nodes: places and transitions. The oriented arcs connect some places to some transitions, or conversely. To each arc, we associate a weight (non negative integer). More details about this formalism can be found in [Baccelli et al. 1992], [Murata 1989], and [Collart-Dutilleul 2008]. In a formal way, a PN is a 5 tuple $PN = (P, T, A, W, M_0)$ where:

- $P = \{p_1, ..., p_n\}$ is a finite set of places (represented by circles);
- $T = \{T_1, ..., T_m\}$ is a finite set of transitions (line segments);
- $A \subseteq (P \times T) \cup (T \times P)$ is a finite set of arcs;
- $W = A \rightarrow \{1, 2, ...\}$ is the weight function associated with arcs;

– $M_0 = P \to \{0, 1, 2, ...\}$ is the initial marking of the graph.

Event Graphs (EG) are a class of PN, where each place has only one input and one output transition. In addition, when times are associated with places, the EG model becomes a P-Timed Event Graph, called TEG in the rest of this paper. Also, when the sojourn times are given as time intervals, the graphical model is called a P-Temporal Event Graph, called PTEG. More details about this variety of PN are given in [Collart-Dutilleul 2008]. Without any ambiguity, when time (fixed time or time interval) is considered, the graphical model is called TEG. In our study, we do not consider timed transitions, which are associated with firing delays. All transitions are immediate transitions, which fire in zero time.

Figure 1 presents the EG model of the contract signature process between a server (service provider) and clients (service requesters). This graph is composed of three parts:

– Client part with its certificate authority *SSDS-Client*.

– Server side with its certificate authority *SSDS-Server*.

– Contract (client request), with all process operations (e.g. hash-coding, signature, time stamp, storage, validation).

The validation of the signature process between clients and a server is verified by the normal evolution of the event graph model of Figure 1. More precisely, when a contract is concluded between a client and a server (firing of the two transitions *Client* and *Contract*), it will be signed by each one. The client signature starts by asking the certification authority for a certificate (firing of the transition *SSDS-Clie*). A detailed study about certification authority and distribution of certificates and public keys is given in [Agarwal and Singh 2010]. Once the client signature process is achieved (firing of the transition *Clien-Sign*), the contract will be sent to the server in order to affix its co-signature (add a token to the place *Contr-Sign-Clie*), and the client will wait for an acknowledgment from the server side (add a token to the place *W-fb-Serv*). The server signature starts also by asking the certification authority for a certificate (firing of the transition *SSDS-Serv*). When a token is in each input place of the transition *Serv-Sign*, this last will be fired, i.e., the signature of the contract by the server. Other operations of the process like time-stamping and storage will then follow the signature operation (firing of the transitions *Ts-Serv*, *Ts-Clie* and *Storage*). After these operations, an acknowledgment will be sent respectively to the client and the server in order to validate the contract (firing of the transition *Valid-Clie* and *Valid-Serv*). If the waiting times of the acknowledgments, in both server side and client side, exceed a fixed threshold, the contract will not be valid and will be rejected. This tolerance threshold represents the conditional behavior of the

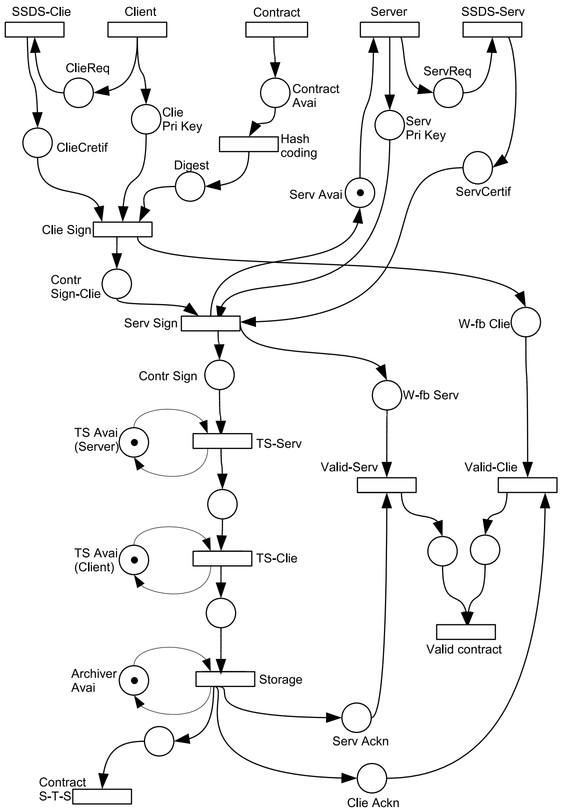

Figure 1: Event Graph model of the signature process

process. Indeed, the validation of a given contract after its signature depends on this threshold and not on other metrics such as disabling of a given transition while preventing the modeled event to occur.

The process evolution consists in verifying seven fundamental properties: correctness, monotony, reachability, boundedness, liveness, deadlock-freeness, and persistence. All these properties have a concrete meaning in the studied process.

- Correctness: means the existence of a marking M allowing to fire any sequence of EG transitions. This means that, during the signature process, there is at least a signed contract between a client and the server.

- Monotony: means that the minimal conditions to fire a given transition of the PN model are satisfied. In other words, this means that when a service is asked by a client, the other actors of the process react to provide the

Legend

Server	Places	- InvokServ: Invoke the server - Serv Req: server request - Serv Avai: server availability - Serv PriKey: server private key - Serv Certif: server certificate - TS-Avai (Serv): availability of timestamp - W-fb-Serv: server waiting feedback - Serv Ackn: server acknowledgement
	Transitions	- Server: server entity - SSDES-Serv: certification authority - Serv Sign: server signature - TS-Serv: time stamping - Valid-Serv: valid contract (server side)
Client	Places	- Clie Req: client request - Clie Pri Key: client private key - Clie Certif: client certificate - Contr-Sign-Clie: contract signed by a client - TS-Avai (Client): availability of timestamp - W-fb-Clie: client waiting feedback - Clie Ackn: client acknowledgement
	Transitions	- Client: client entity - SSDES-Clie: certification authority - Clie Sign: client signature - TS-Clie: time stamping - Valid-Clie: valid contract (client side)
Contract	Places	- Contract Avai: contract availability - Digest: contract digest (hashcoding) - Archiver Avai: archiver availability
	Transitions	- Contract: contract entity - Hash Coding: contract hashcoding - Storage: storage of signed contract - Contr Sign: singed and non valid contract - Contract S-T-S: signed, timestamed and stored contract - Valid Contract: final validation of the contract

Figure 2: Legend of figure 1

requested service.

- Boundedness: a place p is called k-bounded if for all reachable marking M from M_0 (initial marking), $M(p) \leq k$, $\forall p \in P$. To ensure that this property is verified in a PN model, one supposes that the sojourn time of a token in model places is bounded by a fixed value; this means that the execution of a given action is bounded. When this property is verified, each considered signature process will be accomplished and the contract will be valid. Otherwise, the client query can not be satisfied. In our TEG model, the sojourn time of tokens in some places is included in a time intervals. The upper bound of some of these intervals is not defined (equals to $+\infty$, nevertheless in reality this upper bound will never be reached because each asked query can be validated or not according to the fixed time threshold. Hence, the proposed TEG is bounded in all cases.
- Reachability: this is a fundamental basis for studying the dynamic properties of any process. This property consists in checking whether an event

(signature, time stamping, storage) can be accomplished. The purpose is to check whether there is a transition sequence allowing the transformation of the marking M_0 into a given marking M.

- Liveness: ensures that the firing of each transition, independently of the net evolution from its initial marking, is possible. We say that the transition is alive and each process event can be reached.

- Deadlock-freeness: a reachable marking M is a deadlock if no transition can be fired from the marking M. This property is verified when the process of signature and validation of contract is blocked at a given marking M.

- Persistence: an EG model is said to be persistent if, for any couple of enabled transitions, the firing of one of them will not disable the other. In our case, this property means that the execution of an event will not prevent the execution of any other event which runs simultaneously. Persistence is useful in the context of parallel execution events. An example of persistence appears when at least two signatories should sign a contract simultaneously without any interference from each other.

The standard qualitative analysis of the proposed signature process is done; in particular, all considered properties are analyzed and verified on the EG model (Figure 1) using the *VisualObjectNet++* software [VisualObjectNet++ 2012]. The designed model is used for verifying the process working and the obtained results such as sojourn times of tokens in each place, transition firings, reachable states, etc, show that the considered properties are verified.

After the validation of these properties, and in order to evaluate and analyze other process performances such as evaluation of time occurrence of each event (e.g. signature, time-stamping, final validation), process improvement, etc., the process behavior is described, by translating the EG model, using a representation state in the $(max, +)$ algebra.

4.2 Basic elements of $(max, +)$ algebra

Before describing the process behavior by a $(max, +)$ linear system, let us recall some basic elements of this algebra. For more details, we refer the reader to [Baccelli et al. 1992]. All operations will done in the dioid $(\mathbb{R}_{max}, \oplus, \otimes)$ where $\mathbb{R}_{max} = \mathbb{R} \cup \{-\infty\}$ and operators " $\oplus$ " and " $\otimes$ " are respectively "max" and "the usual addition" ($\forall a, b \in \mathbb{R}_{max}$, $a \oplus b = max(a, b)$ and $a \otimes b = a + b$). The following algebraic laws hold for $\oplus$ and $\otimes$.

- $\oplus$ and $\otimes$ are associative;

- $\otimes$ is distributive over $\oplus$;

- $\otimes$ is commutative for scalars, but not for matrices;
- $\varepsilon = -\infty$ is the neutral element of the law $\oplus$, $(\forall a \in \mathbb{R}_{max}, a \oplus \varepsilon = a)$;
- $e = 0$ is the neutral element of the law $\otimes$, $(\forall a \in \mathbb{R}_{max}, a \otimes e = a)$;

We recall also the following results needed to solve implicit equations in dioid algebra. Let us consider a matrix A and a vector b two elements in a dioid D, the quantity $A^* \otimes b$ (if it exists) is the smallest solution of the implicit equation $x = A \otimes x \oplus b$, where the expression of A^* (called $Kleene$ star) is given by: $A^* = Id \oplus A \oplus A^{\otimes 2} \oplus ... \oplus A^{\otimes n} \oplus A^{\otimes n+1} \oplus ...$ (with $A^{\otimes 2} = A \otimes A$ and Id is the identity matrix, which contains e on the diagonal and ε elsewhere). Since the entries of A^n denote the weights of paths with length n in the corresponding graph and A does not have paths of length greater than n, we get $A^k = \varepsilon$, $\forall k \geq n$. We can then write $A^* = Id \oplus A \oplus A^{\otimes 2} \oplus ... \oplus A^{\otimes n}$. In this case A^k, for $k > n$, does not contribute to the sum of A^* [Baccelli et al. 1992]. In what follows, and since there no risk of ambiguity, we omit "$\otimes$" and write $A \otimes b$ as Ab or $A.b$, and $A^{\otimes n} = A \otimes A \otimes ... \otimes A$ as A^n. We underline that all these operations are also true for scalars, we consider the matrix A and the vectors x and b as simple scalars.

4.3 $(Max, +)$-State representation

In this section, we show how $(max, +)$ algebra will be used for modeling and analytical performance evaluation. The behavior of the system can be described by the following $(max, +)$-state model: $\forall k \geq 2$,

$$\begin{cases} X(k) = A_0 \otimes X(k) \oplus A_1 \otimes X(k-1) \oplus B \otimes U(k) \\ Y(k) = C \otimes X(k) \end{cases} \tag{1}$$

The first equation of the system (1) computes the system state, and the second one computes the system output. The three terms on the right of the first equation are given such that the two first terms $(A_0 \otimes X(k))$ and $(A_1 \otimes X(k-1))$ represent the impact of the internal state of the process on its evolution, and the second one $(B \otimes U(k))$ models the influence of the process input on its evolution. with:

- k is the k^{th} contract (query) between a client and the server;
- $U(k)$ is the arrival time of the k^{th} contract to be signed;
- $X(k)$ contains the execution times of all operations of the process (certification, hash-coding, signature, time stamping, storage, validation) for the k^{th} contract;

- A_0, A_1, B and C are the characteristic matrices of the process. These matrices contain the required times to perform all tasks from certification until contract validation;
- $Y(k)$ is the validation time of the k^{th} contract;

In order to translate (describe) the obtained graphical model (Figure 1) into mathematical equations in $(max, +)$-algebra as the system (1), we first associate a variable to each transition as shown in Figure 3. Thus, we associate input variables (denoted by u_1 and u_2) with input transitions and state variables $(x_1, x_2, ..., x_{11})$ with internal transitions, and finally we associate output variables (denoted by y_1 and y_2) with the output transitions. Also, we assign time intervals to certain places. A time interval associated with a given place means that the sojourn time of a token in this place varies between a lower and an upper bound. All places with time intervals represent the process components, e.g. server, for which the responses are not often immediate and require a time for answering a query. Time intervals can also be assigned to places wherein the tokens wait for responses, e.g. a contract waiting to be validated. These time intervals will be the key elements in the performances improvement of the process. Fixed times, which represent the necessary time to accomplish a task of the process, are assigned with other places of the model.

It is important to underline that in order to simplify the description and transformation of the system behavior by $(max, +)$- implicit equations or/and recurring equations (of 1^{st} or 2^{nd} order), we will not consider the global marking of each place of the TEG describing the structural behavior of the system (see Figure 1). The overall marking of the model is considered during its execution and validation through the *VisualObjectNet++* software [VisualObjectNet++ 2012]. Using this tool, the whole qualitative properties are verified while considering the marking of the model.

While taking into account these remarks and introducing the new variables, the fixed times and the time intervals, we obtain the TEG model of Figure 3.

The second stage of the analytical modeling is to label each model variable x_i $(1 \leq i \leq 11)$ by $x_i(k)$, called "*dater*", which denotes the time of the k^{th} firing of x_i. Similarly we define the daters $u_i(k)$ and $y_i(k)$ $((1 \leq i \leq 2))$.

Using all daters and times (time intervals and fixed times), we obtain various equations that model the process. For doing so, we distinguish two cases according to delays associated with the places. We give the rules to translate the graphical model, with time intervals, into the mathematical equations. Also, we recall how to translate the graphical model behavior, with fixed delays, into $(max, +)$-linear equations. In the following examples, we express the time of the

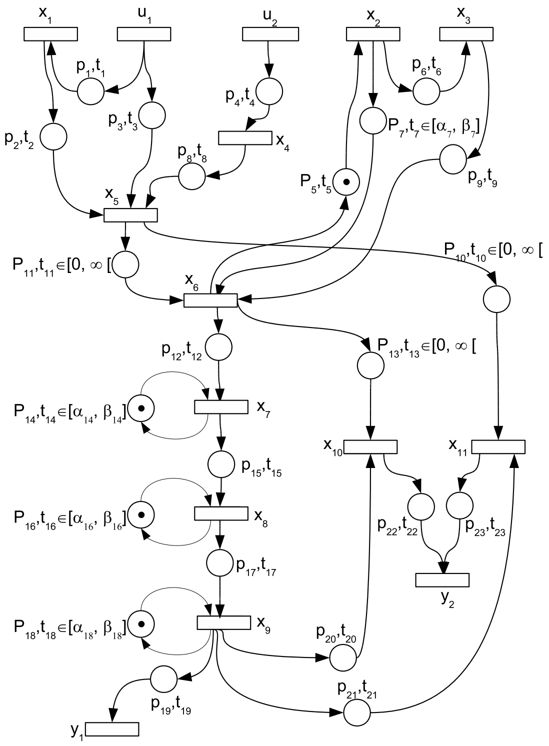

Figure 3: TEG model of the signature process labeled with variables and times

k^{th} firing of a given transition.

– Places with fixed delays: e.g. the transition x_5

$$\begin{aligned} x_5(k) &= max(t_2 + x_1(k), t_3 + u_1(k), t_8 + x_4(k)) \\ &= t_2 \otimes x_1(k) \oplus t_3 \otimes u_1(k) \oplus t_8 \otimes x_4(k) \end{aligned}$$

– Places with time intervals: e.g. the transition x_6

$$\begin{aligned} x_6(k) &= max(t_7 + x_2(k), t_9 + x_3(k), t_{11} + x_5(k)) \\ &= t_7 \otimes x_2(k) \oplus t_9 \otimes x_3(k) \oplus t_{11} \otimes x_5(k); \\ &\text{with } t_7 \in [\alpha_7, \beta_7] \text{ and } t_{11} \in [0, +\infty[\end{aligned}$$

All $(max, +)$-equations representing the system are given by the model (2).

$$\begin{cases} \forall k \geq 2, & \\ x_1(k) & = t_1 \otimes u_1(k) \\ x_2(k) & = t_5 \otimes x_6(k-1) \oplus t_0 \otimes u_2(k) \\ x_3(k) & = t_6 \otimes x_2(k) \\ x_4(k) & = t_4 \otimes u_2(k) \\ x_5(k) & = t_2 \otimes x_1(k) \oplus t_3 \otimes u_1(k) \oplus t_8 \otimes x_4(k) \\ x_6(k) & = t_7 \otimes x_2(k) \oplus t_9 \otimes x_3(k) \oplus t_{11} \otimes x_5(k) \\ x_7(k) & = t_{12} \otimes x_6(k) \oplus t_{14} \otimes x_7(k-1) \\ x_8(k) & = t_{15} \otimes x_7(k) \oplus t_{16} \otimes x_8(k-1) \\ x_9(k) & = t_{17} \otimes x_8(k) \oplus t_{18} \otimes x_8(k-1) \\ x_{10}(k) & = t_{13} \otimes x_6(k) \oplus t_{20} \otimes x_9(k) \\ x_{11}(k) & = t_{10} \otimes x_5(k) \oplus t_{21} \otimes x_9(k) \\ y_1(k) & = t_{19} \otimes x_9(k) \\ y_2(k) & = t_{22} \otimes x_{10}(k) \oplus t_{23} \otimes x_{11}(k) \end{cases} \tag{2}$$

with: $t_7 \in [\alpha_7, \beta_7], t_{10}, t_{11}, t_{13} \in [0, +\infty[, t_{14} \in [\alpha_{14}, \beta_{14}], t_{16} \in [\alpha_{16}, \beta_{16}], t_{18} \in [\alpha_{18}, \beta_{18}]$. The upper bounds β_i (with $i \in 7, 14, 16, 18$) are fixed while taking into account unforeseen circumstances such as network saturation, breakdown, connexion loss.

With the aim to avoid waiting for a long time for a requested service (validate a given contract), we assume that the sojourn times of tokens, that represent contracts, in p_{10}, p_{11} and p_{13} should be lower than a fixed limit μ (threshold). Afterwards, we are interested only in the final validation of each contract. So we suppose that waiting time of client and server are respectively Δt_c and Δt_s. These times should be lower than the threshold μ. If $min(\Delta t_c, \Delta t_s) > \mu$ then the current contract will not be valid and will be destroyed. This means that the service is not offered due to the surrendering of the client or the absence of the response from the server.

The equations of the system (2) will be written as a 1^{st}-order recurrent matrix equation in order to facilitate its resolution. In doing so, we define the following vectors:

- Input vector $U = [u_1, u_2]^t$;
- State vector $X = [x_1, x_2, x_3, ..., x_{11}]^t$;
- Output vector $Y = [y_1, y_2]^t$.

By using these vectors, the equations of the system (2) can be written, as (1), in the following way: $\forall k \geq 2$,

$$\begin{cases} X(k) = A_0 \otimes X(k) \oplus A_1 \otimes X(k-1) \oplus B \otimes U(k) \\ Y(k) = C \otimes X(k) \end{cases} \tag{3}$$

where $A_0 \in \mathbb{R}_{max}^{11\times 11}$, $A_1 \in \mathbb{R}_{max}^{11\times 11}$, $B \in \mathbb{R}_{max}^{11\times 2}$ and $C \in \mathbb{R}_{max}^{2\times 11}$ are the characteristic matrices of the model.

Explicitly, these matrices are given by :

$$A_0 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & t_6 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ t_2 & \varepsilon & \varepsilon & t_8 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & t_7 & t_9 & \varepsilon & t_{11} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{12} & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{15} & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{17} & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{13} & \varepsilon & \varepsilon & t_{20} & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{10} & \varepsilon & \varepsilon & \varepsilon & t_{21} & \varepsilon & \varepsilon \end{pmatrix}, A_1 = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_5 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{14} & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{16} & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{18} & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix},$$

$$B^t = \begin{pmatrix} t_1 & \varepsilon & \varepsilon & \varepsilon & t_3 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & t_0 & \varepsilon & t_4 & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{pmatrix}, C = \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{19} & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & t_{22} & t_{23} \end{pmatrix}.$$

4.4 $(\boldsymbol{Max}, +)$-state resolution

In order to solve the implicit equation given by the system (3), we proceed by the following way: we replace in the first equation of (3), successively, $X(k)$ with its expression to obtain the smallest solution of the first equation of (3):

$$\begin{cases} X(k) = A_0 \otimes X(k) \oplus A_1 \otimes X(k-1) \oplus B \otimes U(k) \\ \qquad = A_0[A_0X(k) \oplus A_1X(k-1) \oplus BU(k)] \oplus \\ \qquad\quad A_1X(k-1) \oplus BU(k) \\ \qquad = A_0^2X(k) \oplus A_0A_1X(k-1) \oplus A^0BU(k) \oplus \\ \qquad\quad A_1X(k-1) \oplus BU(k) \\ \qquad = A_0^2X(k) \oplus [A_0 \oplus Id]A_1X(k-1) \oplus \\ \qquad\quad [A_0 \oplus Id]U(k) \\ \qquad = ... \\ \qquad = A_0^nX(k) \oplus \oplus_{i=0}^{n-1} A_0^i[A_1X(k-1) \oplus BU(k)] \\ \qquad = \oplus_{i=0}^{n-1} A_0^iA_1X(k-1) \oplus \oplus_{i=0}^{n-1} A_0^iBU(k)] \\ \qquad = (\oplus_{i=0}^{n-1} A_0^i)A_1X(k-1) \oplus (\oplus_{i=0}^{n-1} A_0^i)BU(k)] \\ \qquad = A_0^*A_1X(k-1) \oplus A_0^*BU(k) \end{cases} \tag{4}$$

where A_0^* is defined by: $A_0^* = \oplus_{i=0}^{+\infty} A_0^i$. As mentioned above about the calculation of the Kleene star A_0^*, A_0^n for $n \geq 11$ (A_0 is an (11x11) matrix), does not contribute to the sum of A_0^*. In other terms, $\forall n \geq 11$, $A_0^n = \varepsilon$. So, $A_0^nX(k) = \varepsilon$, $\forall n \geq 11$ and $\forall k \geq 1$. Let us recall that the matrix Id introduced in the system 4 is the identity matrix in $(max, +)$ algebra.

The evaluation of the system will be done knowing that the numerical values of the system input $U(k)$, for all k, and the system initial state $X(1)$ are given. The solution of (3) is then given by: $\forall k \geq 2$,

$$\begin{cases} X(k) = A_0^* A_1 X(k-1) \oplus A_0^* BU(k) \\ \qquad\;\; = (A_0^* A_1)^{k-1} \otimes X(1) \oplus \oplus_{i=0}^{k-2} (A_0^* A_1)^i \otimes \\ \qquad\quad\;\; (A_0^* B)U(k-i) \\ \qquad\;\; = R^{k-1} X(1) \oplus \oplus_{i=0}^{k-2} R^i \otimes QU(k-i) \\ Y(k) = C \otimes X(k) \\ \qquad\;\; = C \otimes (R^{k-1} X(1) \oplus \oplus_{i=0}^{k-2} R^i \otimes QU(k-i)) \end{cases} \tag{5}$$

where $R = A_0^* A_1$ and $Q = A_0^* B$.

4.5 Evaluation study

For the evaluation study, we will assign numerical values to various parameters (see Table 1). These various numerical values are defined as follows. Each system operation can be done within a given time interval $[a, b]$, where the lower bound "a" is the required minimum time to perform the operation, and the upper bound "b" is the maximum time to execute the task. The values of t_7, t_{14}, t_{16}, t_{17} and t_{18} are fixed within the temporal intervals (as given in Table 1) according to some criteria, such as: the network load, availability of an actor to perform a task. Other parameters t_{10}, t_{11}, and t_{13} represent the waiting for an acknowledgment (synchronization phenomenon). These waiting times vary from "0" , which means that the waiting time of an acknowledgment is null, to $+\infty$ which means that the acknowledgment will never be received. In most cases, the acknowledgment time is defined and bounded. In addition, we propose a feasibility study and performances improvement of the process. All timing parameters are chosen according to an estimation of required times to achieve each task of the process. For a concrete application, these timing parameters may be changed slightly but the principle remains the same.

For the system input, the firing of the transitions u_1 and u_2 follow a uniform law. For example they are fired every 5 time units (u.t.), thus : $\forall k \geq 2$

$$\begin{cases} U(1) = [u_1(1), u_2(1)]^t = [0, 0]^t = [e, e]^t \\ U(k) = 5 \otimes U(k-1). \end{cases} \tag{6}$$

It is worth noting that the inputs (or arrivals) of the system are supposed to be deterministic and follow a uniform law. This law is chosen just as illustrating example. When choosing another law (e.g. stochastic law), the developed model in equation (3) and its solution given by equation (5) remain the same. These equations express the system behavior and its output regardless of the nature of the system input (deterministic or stochastic). Nevertheless, in the stochastic

Times	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Numerical values	0	5	3	0	0	0	3	$\in [0, 10]$
	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
	2	3	$\in [0, +\infty[$	$\in [0, +\infty[$	0	$\in [0, +\infty[$	$\in [0, 5]$	0
	t_{16}	t_{17}	t_{18}	t_{19}	t_{20}	t_{21}	t_{22}	t_{23}
	$\in [0, 5]$	0	$\in [0, 5]$	0	0	0	0	0

Table 1: Numerical values of system parameters

behavior case, the characteristic matrices of the model and its behavior will changed. Also the firing rules of model transitions will not be the same.

By replacing parameters of the characteristic matrices A_0, A_1, B and C by their numerical values, and by using the solution (4), giving $X(1)$ and $U(k)$ for all k, we can evaluate the various system state, in terms of occurrence of events (creation of certificates, signatures, storage and validation of contract terms, etc.). In our numerical application, we consider a set of $M_{max} = 10000$ elements (contracts to be signed between server and clients).

One of the important elements that we will study is the evolution of signatures and validation of contracts between a server and clients taking into account several factors: server capacity, network load, availability of various confidence actors. Recall that Δt_s and Δt_c, introduced previously, are the waiting times for an acknowledgment receipt. After solving the state vector (solution (5)), it is possible to calculate these parameters for each contract introduced into the signature process. On the server side, the expression of Δt_s (waiting time to validate a contract with a client) is given by:

$\forall k \geq 1, \Delta t_s(k) = (x_9(k) + t_{20}) - x_6(k)$

On the client side, the expression of Δt_c (waiting time to validate a contract with the server) is given by:

$\forall k \geq 1, \Delta t_c(k) = (x_9(k) + t_{21}) - x_5(k)$

Let's define $\Delta t(k)$, $\forall k \geq 1$, by:

$$\Delta t(k) = \Delta t_s(k) \oplus \Delta t_c(k) \tag{7}$$

For each k, if $\Delta t(k)$ exceeds the tolerance threshold μ, then the k^{th} contract will not be valid between the server and the client, and then the access, by the client, to the required service will not take place. Considering the numerical values of Table 1, and the $(max, +)$-solution (equation (5)), the evaluation results are depicted in Figure 4. The validation percentage and non-validation

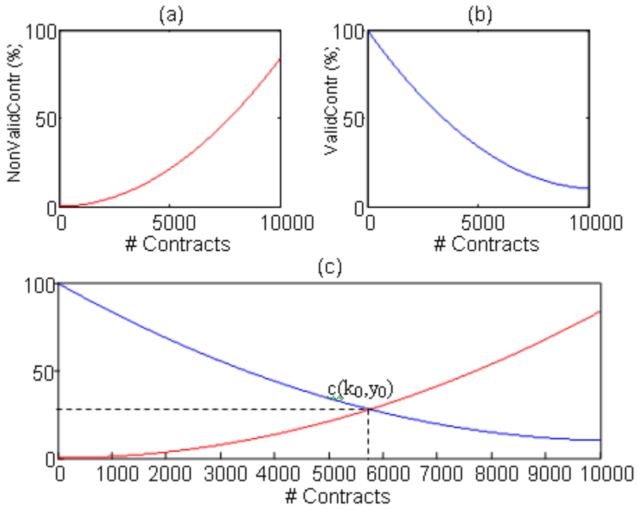

Figure 4: Evaluation results with numerical values of Table 1

percentage of contracts are computed according to the waiting times Δt.

Figure 4(a) shows that the percentage of non valid contracts increases when the number of requests increases and the server capacity remains the same. We remark that at the beginning of the process all requests are answered and all contracts are validated. For $k \geq 7600$ requests, the non-validation percentage of contracts is above 50 %. When the number of requests exceeds $k = 7600$, the validation percentage remains below 50 %. Figure 4(b) shows the validation evolution of contracts: when the number of received contracts increases, the validation percentage decreases, because of limited resources, or resources availability. Figure 4(c) merge the two figures given previously. The benchmark $c(k_0, y_0)$ is the intersection point wherein the two percentages of valid contracts and Non-valid contracts are identical. For each point $c(k, y)$ where $(k \geq k_0)$, the number of Non-valid contracts exceeds the number of valid contracts, i.e. $y \geq y_0$.

It is important to note that the percentages of valid and non-valid contracts should sum up to 100 % at the end of the process. Nevertheless, the results shown in Figure 4 and Figure 10, are taken before the end of the process. In this case, a part of all submitted requests (100 %) are underway of signature and validation. The point $c(k, y)$ corresponds to 35 % of valid contracts, 35 % of non-valid contracts and 30 % of contracts underway for signature and validation.

In order to shun this situation and satisfy the majority of clients, the server should be dimensioned in such way that the intersection of the two curves (see Figure 4(c)) will not take place, or at least the new value of k_0 of the benchmark

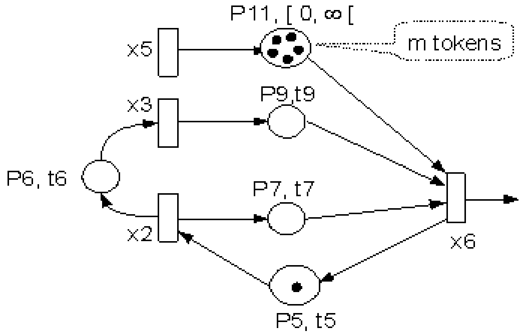

Figure 5: Accumulation of tokens (contracts) in p_{11}

$c(k_0, y_0)$ becomes the higher. This would allow a maximum number of requests to be answered.

5 Dimensioning and improvement

5.1 An optimization-based resource allocation algorithm

To improve the quality of service offered to the clients, in terms of waiting times and validation of their requests, we propose to act on the server side. Because of the limited capacity of this server, the answer of a large number of client requests is a hard situation to reach with only one server. Indeed, if several tokens (m tokens), which model the contracts arrive at the place p_{11} situated between *Clie-sign* (x_5) and *Serv-sign* (x_6) (see Figure 5, extracted from PN of Figure 3), and the capacity of the used server is limited and fixed, these tokens will be accumulated in p_{11}. This situation increases the waiting time of tokens in this place, which causes enormous delays of the contracts to be signed by the server. Considering these delays, the number of valid contracts, at a given time, becomes lower compared to the number of non valid contracts.

In our model, the main critical places, wherein the sojourn times might exceed the fixed threshold, are p_{10}, p_{11}, and p_{13}. So, the improvement of the signature process will be done by decreasing the delays associated with these places. To do so, we propose to resize the system in order to accelerate the process and answer a maximum number of client requests.

The improvement policy we propose in this paper allows decreasing the delay d_i of each token i (i^{th} contract) waiting in the downstream place p_{11} of the transition *Clie-sign* (x_5). We propose to optimally duplicate the server into several servers with the same capacities as the first one. It is an optimization problem

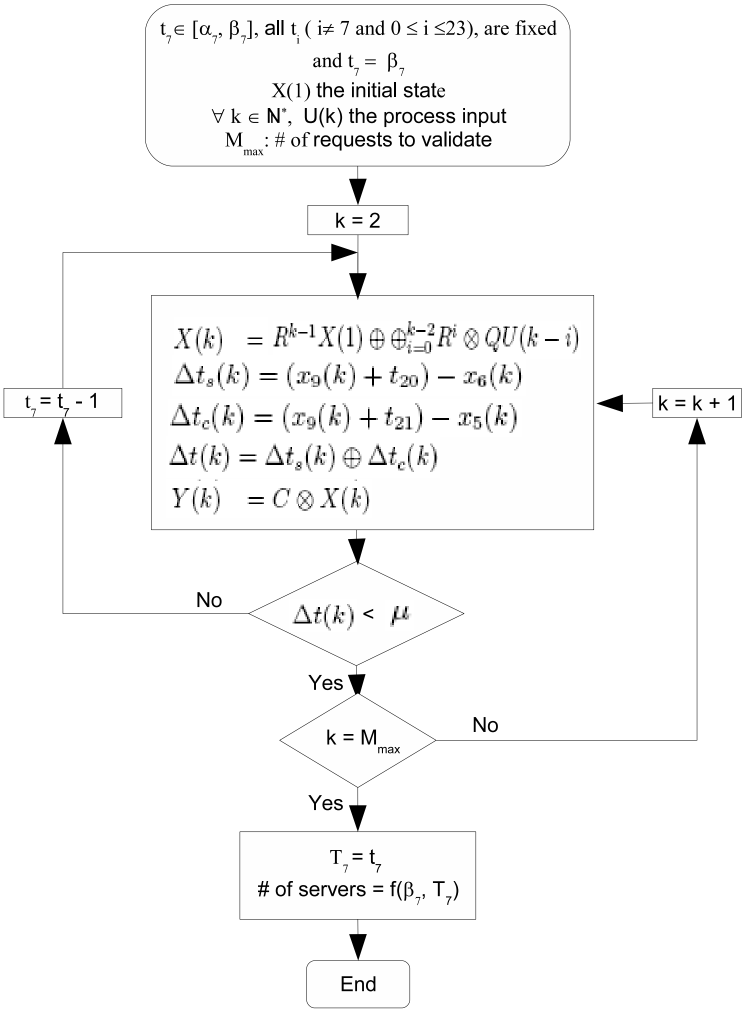

Figure 6: The flowchart of dimensioning and improvement

wherein we seek to maximize the number of valid requests while minimizing the number of servers used. This optimization study is based on the algorithm presented in Figure 6, which operates as follows: all times t_i, $(0 \leq i \leq 23$ and $i \neq 7)$ are supposed to be fixed except $t_7 \in [\alpha_7, \beta_7]$ which represents the required time for the server to sign a request. For this optimization study, we act on this time in order to determine the minimum time required to validate a client request by a server. After the initializing step of the algorithm wherein all input parameters are defined such as times, the initial state $X(1)$, the arrival times of requests $U(k)$ for each k, the number M_{max} of requests to be answered, we analyze the solution 5 of the sate equation describing the process. As explained previously, we calculate, for each k, $\Delta t(k)$ while starting with $t_7 = \beta_7$ as the associated delay with P_7, if this delay is less than the fixed threshold μ, which means that the considered time t_7 is enough to answer the client request, so this k^{th} contract is valid. Otherwise, we decrease the time t_7 in order to maintain $\Delta t(k)$ less than μ. The operation will be repeated until reaching M_{max} requests. At the end of the algorithm, when M_{max} valid requests is reached, we obtain the minimal time interval, $[\alpha_7, T_7]$ (which is included in $[\alpha_7, \beta_7]$), required to answer all requests. We get then, from this time interval, the minimum number of servers to be used to answer the requests of clients.

To summarize, in order to satisfy all requests, a single used server should answer each request, at the latest, at T_7 time units. Nevertheless, knowing that a server can not always react in time due to, for example, connexion loss, network saturation, breakdown and so on, we propose to optimally use more than one server while keeping the response time of each server under the upper bound β_7 time units, taking into account any unexpected situation. Considering the algorithm results, the minimum number of servers N, to be used to answer and validate all requests, while maintaining the waiting time of each request less than μ, and response time of each server can reach β_7 time units (considering unexpected situations) is given by:

$$N = \lceil \frac{\beta_7}{T_7} \rceil \tag{8}$$

By using N servers, the validation percentage reaches 100 %. But, if less than N servers are used, the validation process will be improved but the 100 % will not be reached. In what follows, and with the aim to show how the process evolves by increasing the number of used servers, we use more than one server and less than N servers.

After determining the optimal number of servers to be used to validate M_{max} requests, the system will be dimensioned by changing the marking of the PN while replacing the only token (representing the single server) in the place p_5 (see Figure 3) by the obtained optimal number of servers to be used. By changing the marking of the PN model describing the process, the associated $(max, +)$-state model becomes a N-order recurrent equation. This requires the reformulation of the $(max, +)$- state model in order to rewrite it under the form of the system (1), and then use the solution given by the equation (5).

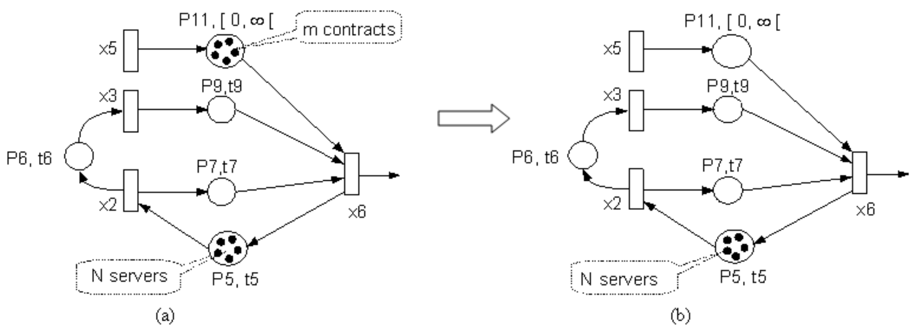

Figure 7: Duplication of a server S into: N servers $s_1, s_2, ..., s_N$ (N tokens in p_5, (a) TEG with accumulation of tokens into p_{11} (before any firing of x_6), (b)TEG without accumulation of tokens in p_{11} (after m firings of x_6)

5.2 $(Max, +)$-state reformulation

This section deals with the reformulation of the $(max, +)$-state model of the process using several servers. Recall N is the number of servers to be used (see Figure 7). By introducing these servers, the average waiting delay, (T_{av}), of each request is computed as follows:

$$T_{av} = \frac{1}{N} \sum_{i \geq 1} d_i \tag{9}$$

The TEG model with several servers will be different from the model illustrated in Figure 3 in terms of the initial marking M_0. But the signature process remains the same. We suppose that the N servers, $s_1, s_2, ..., s_N$ (N tokens in the place p_5, Figure 7(a)) provide the same service. This means that they are identical, and have the same characteristics (capacity, speed, service, ...). This new configuration is modeled by TEG model of Figure 7(b). It will replace the server part of the model of Figure 1. For this new model, we suppose that only one certification authority (*SSDS-Server* represented by x_3) is enough to provide certificates to all considered servers.

Using this model and its new marking, the problem of accumulation of tokens in p_{11} can be solved (Figure 7(b)). Indeed, after the signature of each client request there exist, among all used servers, one available to sign the next request. Using this TEG model, the process operates appropriately and the maximum of requests will be answered without enormous delays.

In order to facilitate the description and the translation of the system behavior by introducing the sub model of Figure 8(b), we prove the equivalence

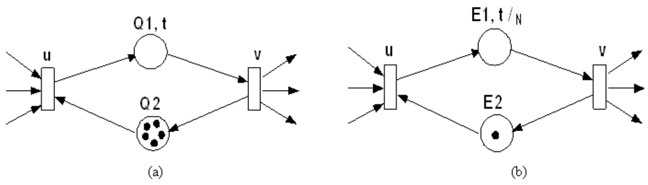

Figure 8: Two TEG models: (a) first model, (b) second model

between the two following TEG models of Figure 8 in terms of their respective $(max, +)$-models. We underline that, for the model of Figure 8(a), the delay associated with the place Q_1 is "t", and Q_2 contains N tokens; for Figure 8(b), the delay associated with the place E_1 is "$\frac{t}{N}$" and E_2 contains only one token. We consider the following assertion:

Assertion: The $(max, +)$-equations of the two TEG of Figures 8(a) and 8(b) are equivalents.

Proof: The TEG of Figure 8(a) is expressed with the followinf $(max, +)$ equations: $\forall k > N$,

$$\begin{cases} u(k) = v(k-N) & (\alpha) \\ v(k) = t \otimes u(k) & (\beta) \end{cases} \tag{10}$$

And the TEG of Figure 8(b) is described by: for all $k > 1$,

$$\begin{cases} u(k) = v(k-1) & (\theta) \\ v(k) = \frac{t}{N} \otimes u(k) & (\omega) \end{cases} \tag{11}$$

In order to prove the equivalence of the two models of Figure 8(a) and Figure 8(b), we prove the equivalence of their associated $(max, +)$ linear models given respectively by (10) and (11). We analyze the system 10 in which the equation (ω) becomes, using (θ):

$$v(k) = \frac{t}{N} \otimes v(k-1)$$

Otherwise, by developing the recurrence equation (θ) of (11), we obtain:

$$\begin{cases} v(k) = \frac{t}{N} \otimes v(k-1) \\ \quad\quad = \frac{t}{N} \otimes \frac{t}{N} \otimes v(k-2) = (\frac{t}{N})^{\otimes 2} \otimes v(k-2) \\ \quad\quad = \ldots \\ \quad\quad = (\frac{t}{N})^{\otimes N} \otimes v(k-N) \\ \quad\quad = t \otimes v(k-N) \end{cases} \tag{12}$$

On the other hand, if we consider the equations of system (10) and by replacing $u(k)$ (the equation (α)) by its expression in (β), we obtain:

$$v(k) = t \otimes v(k - N) \tag{13}$$

We remark then that using system (10) or (11) the expression are same, (12) and (13). This means that the date $v(k)$ of the k^{th} firing of the transition v remains the same for the two TEG of Figure 8. This implies that the two systems (10) and (11) are equivalent. Consequently, the two models 8(a) and 8(b) are equivalent, and then can be solved using $(max, +)$ algebra techniques for 1^{st}-order recurrent equations.

This assertion shows that it is possible to move from a high-order of a recurrent equation to a low-order of the same equation while describing the behavior of the same system. In our case, we reduce an N-order recurrent equation to a 1^{st}-order recurrent equation for which the handling and the resolution is easier.

Using the given assertion, and replacing the server part by the sub model of Figure 7(b), the $(max, +)$ model describing the behavior of the TEG model of Figure 3 is similar to the system (2), and its expression under a matrix form will be the same as the system (3). To solve the associated $(max, +)$ model of the new TEG model (with N servers), we proceed in the same way as in the Section 4.4, and the resolution of the obtained system will be the same as for the system (5) with different characteristic matrices.

5.3 Dimensioning results

Now, we evaluate and analyze the process for this new dimensioning. By analyzing the obtained results from the $(max, +)$-linear representation associated with the new TEG (with N servers) and using the same numerical values given in Table 1, we obtain simulation evaluation results given hereafter. For comparison purpose, some comparative results are given in Figures 9 and 10. It is worth noting that this number of servers ($N = 3$) is less than the optimal number to be used to reach the percentage 100 % of requests validation. This choice is made to show how the validation of requests evolves according to the number of received contracts and also the number of servers used. The curves representing the evolution of the number of validated contracts using the optimal number of servers are not presented since the validation percentage is 100 %.

The simulation results show that, when only one server is used, the total waiting delay increases when the number of requests from clients increases (Figure 9, $N = 1$). Nevertheless, when three servers are used, this waiting time increases more slowly (Figure 9, $N = 3$). This means that, when only one server is used the percentage of valid contracts decreases in a remarkable fashion when the number of contracts (client requests) increases and vice versa (see Figures

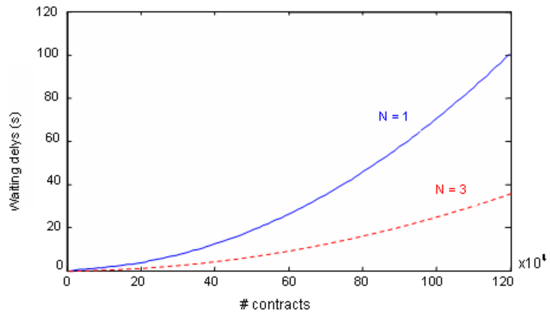

Figure 9: Waiting delays of requests with: only one server ($N = 1$) and three servers ($N = 3$)

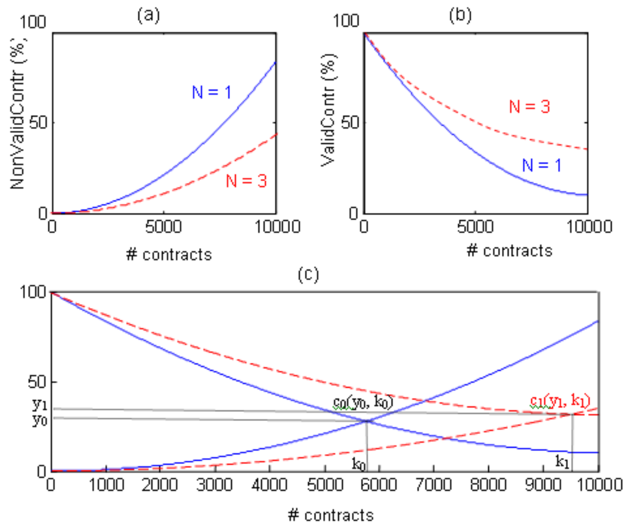

Figure 10: Evaluation results considering values of Table 1 and using one ($N = 1$) and three servers ($N = 3$).

10(a) and 10(b)). However when three servers are used, we remark clearly the improvement of service, and the validation percentage of client requests becomes more important.

Similarly to Figure 4(c), the Figure 10(c) gathers the two Figures 10(a) and 10(b) and also contains the curves of Figure 4 (for comparison). This figure shows the improvement of the proposed service to the clients. Indeed, with the new dimensioning of the system, the intersection point of the two curves (Valid-contracts and Non-valid contracts) is shifted to the right and the new benchmark is $c_1(k_1, y_1)$. This means that, for all k with $(k_0 \leq k \leq k_1)$ the percentage of Valid-contracts becomes higher than those of Non-valid contracts. So, considering the new value k_1, a significant number of client requests are satisfied.

6 Conclusions and future work

In this paper, a contract-signature process was modeled, evaluated and improved using $(max, +)$-algebra. The process was first modeled using a dynamic timed event graph, and some appropriate properties of this process were evaluated through this graphical model. $(max, +)$-equations describing the analytical behavior of the process are then derived from the Petri net model. The required performance metrics are evaluated using these equations. A performance tuning algorithm was proposed to improve the quality of service offered by the service provider. This tuning method allows to study the trade off between the number of requests and the number of servers required to satisfy them, e.g., satisfy more requests by using a minimum number of servers. Throughout all this study, we demonstrated, using a real-world application, how the proposed methodology can be used for validation of qualitative properties as well as performance analysis, evaluation, and improvement. In future work, we will extend this methodology to model and evaluate the performance of complex and large distributed systems. More precisely, the behavior of the proposed system will be modeled as a probabilistic/stochastic process. Under this aspect, we will deal with some issues related to the process such as "what is the expected number of successful contracts?", or also "what is the probability that the number of successful contracts at a given time is less than x contracts ?". Other issues will be addressed such as the conditional behavior modeling of the process using alternative proposals. For example, instead of the validation conditional of contracts using a fixed time as tolerance threshold, we will represent this conditional behavior while disabling certain transitions of the model when some signature or validation conditions are not satisfied.

References

[Adams et al. 2005] Adams, C., Farrell, S., Kause, T., Mononen, T.: "Internet X.509 public key infrastructure certificate management protocol (CMP)"; Technical Report RFC 4210, IETF. 107. 2005.

[Agarwal and Singh 2010] Agarwal, G., Singh, S.: "A comparison between public key authority and certification authority for distribution of public key"; International Journal of Computer Science and Information Technologies, 1(5):332-336, 2010.

[Ajmone Marsan et al. 1998] Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: "Modelling with Generalized Stochastic Petri Nets"; ACM SIGMETRICS Performance Evaluation Review - Special issue on Stochastic Petri Nets, Volume 26 Issue 2, August 1998, doi 10.1145/288197.581193.

[Baala et al. 2003] Baala, H., Flauzac, O., Gaber, J., Bui, M., El-Ghazawi, T.: "A self-stabilizing distributed algorithm for spanning tree construction in wireless ad hoc networks"; J. Parallel Distrib. Comput. 63 (2003) 97104. doi:10.1016/S0743-7315(02)00028-X.

[Baccelli et al. 1992] Baccelli, F., Cohen, G., Olsder, G-J., Quadrat, J-P.: "Synchronisation and linearity. Algebra for discrete Event Systems"; John wiley et sons; 1992.

[Bakhouya et al. 2009] Bakhouya, M., Suboh, S., Gaber, J., El-Ghazawi, T.: "Analytical Modeling and Evaluation of On-Chip Interconnects Using Network Calculus"; In: The 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS 09) 74-79, 2009.

[Bakhouya and Gaber 2008] Bakhouya, M., Gaber, J.: "Adaptive Approaches for Ubiquitous computing"; In Mobile networks and wireless sensor networks. Eds. Hermes Science, ISBN 2-7462-1292-7, pp. 129-163, 2008.

[Ben Mbarka 2011] Ben Mbarka, M.: "Advanced electronic signature: Modeling long-term validation and the security of certification authorities"; Phd thesis of the university of Bordeaux 1, 191 pages, 2011.

[Brand and Begin 2009] Brandwajn, A., Begin, T.: "Higher-order distributional properties in closed queueing networks"; Performance Evaluation (2009), doi:10.1016/j.peva.2009.05.002.

[Collart-Dutilleul 2008] Collart-Dutilleul, S.: "Les réseaux de Petri P-temporels: Modélisation et validation d'exigences temporelles"; HDR (Habilitation à diriger des recherches); Université des Sciences et Technologies de Lille, 2008.

[Firozabadi and Sergot 2002] Firozabadi, B., Sergot, M.: "Contractual access control"; In Proce. of 10th International Workshop of Security Protocols, Cambridge (UK), 2002.

[Gaubert 1992] Gaubert, S.: "Théorie des systèmes linéaires dans les dioides"; Thése de doctorat, Ecole Nationale Supérieure des Mines de Paris. Juillet 1992.

[Gidron et al. 2001] Gidron, Y., Ben-Shaul, I., Holder, O., Arior, Y.: "Dynamic Configuration of Access Control for Mobile Components in FARGO"; Concurrency and Computation, V 13, N 1, pp. 5-21, January 2001.

[Gondrom and Fischer 2010] Gondrom, T., Fischer-Dieskau, S.: "Validation and long term verification data for evidence records and signed documents"; Technical report, IETF, 2010. 30.

[Kwiatkows et al. 2002] Kwiatkows ka, M., Norman, G., Parker, D.: "PRISM: Probabilistic Symb olic Mo del Checker"; In Proce. of Computer Performance Evaluation / TOOLS, 2002.

[Murata 1989] Murata, T.: "Petri nets: properties, analyse and applicatios"; In Proce. of IEEE 77 (4), 541-580, 1989.

[Nait-Sidi-Moh et al. 2008] Nait-Sidi-Moh, A., Bakhouya, M., Wack, M.: "A Secure Approach Based on Contract Negotiation for Service Discovery in Ubiquitous Computing Environments"; International Transactions on Systems Science and Applications, Vol. 4, No. 2, May 2008, pp. 173-180. tai: itssa.2008.05.039.

[Nait-Sidi-Moh and Wack 2005] Nait-Sidi-Moh, A., Wack, M.: "Modelling of Process of Electronic Signature with Petri Nets and (max, plus) Algebra"; Lecture Notes in Computer Science (LNCS), Editeur Springer, Vol. 3483, Issue IV, pp. 792-801, ISBN: 3-540-25863-9. 2005.

[Ninham 2004] Ninham Shand, B.: "Trust for Resource Control: Self-enforcing Automatic Rational Contracts between Computers"; UCAM-CL-TR-600, ISSN 1476-2986, Aug 2004.

[Ruiz-Martínez et al. 2009] Ruiz-Martínez, A., Marín-López, C.I., Baño-López, L. Gómez-Skarmeta, A.F. : "A New Fair Non-repudiation Protocol for Secure Negotiation and Contract Signing"; Journal of Universal Computer Science, Vol. 15, Issue 3 (2009), 555-584. doi: 10.3217/jucs-015-03-0555.

[VisualObjectNet++ 2012] VisualObjectNet, http://www.r-drath.de/Home/Visual_Object_Net++.html, last visit March 2012.

[Wack et al. 2006] Wack, M., Nait-Sidi-Moh, A., Lamrous, S., Cottin, N.: "Meaningful Electronic Signatures and Associated Processes Formalization Proposals"; Journal of Artificial Intelligence and Law, Editor Springer, 2006. ISSN 0924-8463.

[Wendell-H-Fleming 2004] Fleming, W-H.: "Max-Plus Stochastic Processes"; Applied Mathematics and Optimization 49: 159-181, 2004.

[Wu et al. 2009] Wu, W., Mu, Y., Susilo, W., Huang, X.: "Certificate-based Signatures Revisited"; Journal of Universal Computer Science, Vol. 15, Issue 8 (2009), 1659-1684. doi: 10.3217/jucs-015-08-1659.