Performance analysis of composite web services using Stochastic Automata Networks over IP network

Jalel Ben Othman
Laboratoire Prism
Université de Versailles
jbo@prism.uvsq.fr

Lynda Mokdad
Laboratoire Lamsade
Université Paris-Dauphine
mokdad@dauphine.fr

Mohamed Ould Cheikh
Laboratoire Lamsade
Université Paris-Dauphine
cheikh@dauphine.fr

Mbaye Sene
Université cheikh Anta Diop
mbaye.sene@ucad.sn

Abstract—The emergence and success of IP networks make the development of new services like voice, multimedia or web services possible. One of the major problem of these type of application is that IP networks is not designed to provide Quality of Service (QoS) in an end to end architecture. Thus the performance evaluation of these applications is important to guarantee a high level of QoS. We focus in this study to evaluate the performance evaluation of Web services which is the most difficult to achieve because of the complexity of their architectures. The classical mathematical formalisms such as queueing networks can not be used in this kind of problems due the synchronization events and the state space explosion. In this paper we propose to use an adequate and efficient modelling tool: the Stochastic Automata Network (SAN) to evaluate the performance of Web services architecture over IP networks. Numerical results using PEPS are given for considered performance measures.1

I. INTRODUCTION

A Web service indicates primarily an application (or a program) available on Internet through a service provider, and accessible by the clients toward standard Internet protocols. However, for several types of applications, the combination of a set of Web services into more complex services (aggregated of many other unitary Web services or composites Web services) is necessary to meet requirements more complex.

The Quality of Service (QoS) represents a set of operational service properties (such as packet loss, blocking probability, delay) that must be considered in the composition of the service (speed, availability and response time, etc. ...).

These properties (or measures) represent a set of requirements concerning the implementation of the service. The degradation of these measures can generate serious consequences. To guarantee these properties, a system behavior analysis is necessary to identify the problems and to resolve them. It is therefore essential to have adapted methods and tools for analyzing and for understanding the behavior of these services driven by the performance and cost.

The evaluation of such systems is difficult due to the complexity of the associated architecture. Thus, the aim of this paper is to propose a new methodology for the evaluation of the composite Web services performances using the Stochastic Automata Network.

This paper is organized as follows. After a brief introduction of SAN formalism in next section, we present in section III the considered architecture based on composite Web services and its modelization by SAN. Numerical results are given in section IV. Main results are discussed in section V, and comments about further research items are given. Finally, we resume in an appendix an example of product and sum tensor used in this paper.

II. STOCHASTIC AUTOMATA NETWORK (SAN)

SAN is a formalism for the definition and the solution of complex systems with a very large state space. This formalism was proposed by Plateau [8] and its basic idea is to represent a whole system by a collection of subsystems with an independent behavior (local transitions) and occasional interdependencies (functional rates and synchronizing events). Each subsystem is described as a stochastic automaton, i.e., a certain number of states, along rules or probability functions that rule the movements from one state of the automata to the other.

The main advantage of using Stochastic Automata Network is that this formalism automatically generates its transition matrix in an implicit way toward tensorial algebra. The non-explicit generation of the transition matrix constitutes a significant gain in accordance with the use of the memory [10].

A. Formal description of SAN

To give a formal representation of a SAN, one will consider a SAN composed by :

- \( N \) stochastic automata \( A^{(i)}, i \in [1 \ldots N] \). Each Automaton \( A^{(i)} \) consists of a set of local states \( S^{(i)} \) and a set of the transitions \( T^{(i)} \). We denote by \( n^{(i)} = |S^{(i)}| \), the number of the local states of automaton \( A^{(i)} \), and by \( x^{(i)} \in S^{(i)} \), a local state of the automaton \( A^{(i)} \);
- a set of events \( \mathcal{E} \);
- reachability function \( F. F \) is a function of \( \hat{S} \rightarrow \{0, 1\} \). \( \hat{S} \) is the set of the global states. \( F \) associates to each

1 This work is supported by the project PErsasive Service Composition (PERSO) of the French National Agency for Research. ANR-07-JCJC-0155-01.
global state of $\hat{S}$ the value 1 if it is reachable, otherwise $F$ associates the value 0.

The formal descriptions of the used concepts in the SAN, are presented in the following:

- **Events**: An event $e \in \varepsilon$ is defined by:

  - the occurrence rate $\lambda_e$, which is a function of $\hat{S} \to \mathbb{R}$
  - set of automata implied by this event $O_e$
  - the master automaton $A^{(n_e)} \in O_e$

  An event $e$ is classified as local event, if $|O_e| = 1$, otherwise $e$ is classified as synchronized event.

- **transitions**: for each automaton $A^{(i)}$, a transition $t^{(i)} \in T^{(i)}$ is defined by:

  - a starting state $x_{t^{(i)},\text{dept}} \in S^{(i)}$ and a state of arrival $x_{t^{(i)},\text{arr}} \in S^{(i)}$
  - an event associated with the transition, $\text{ev}t^{(i)} = e \in \varepsilon$;
  - a routing probability $\pi_{t^{(i)}}$

- **Functional element**: $f(A^w)$ is a function of $\Pi_{i \in w} S^{(i)} \to \mathbb{R}^+$, where $w \in [1,...,N]$.

- **Automaton**: each automaton $A^{(i)}$ is defined by:

  - $S^{(i)}$ is a set of the automaton states $A^{(i)}$
  - $Q^{(i)}$ is the transition function of the automaton $A^{(i)}$

  defined of $S^{(i)} \times S^{(i)} \to T^{(i)}$

- **Markovian descriptor**: The advantages of this formalism (SAN) is the facility to describe the generator of the Markov chain using a mathematical formula called descriptor [8, 9]. The descriptor allows to store in a compact form the infinitesimal generator of the Markov chain associated to the SAN.

  - let $E \subset \varepsilon$ the set of the synchronized events. The behavior of each automaton $A^{(i)}$, $i = 1...N$, is described by a set of square matrices of dimension $n_i$. These matrices can be classified in two families:

    * $Q^{(i)}_{e^+}$ correspond to local events of each automaton $A^{(i)}$, $i = 1...N$;
    * $Q^{(i)}_{e^-} + Q^{(i)}_{e^+}$, $2E$ matrices containing all the triplets of synchronization for each events $e$ of $E$.

  The Markovian descriptor is given by:

  $$Q = \bigoplus_{i=1}^{N} Q^{(i)} + \sum_{e \in E} \left( \bigoplus_{i=1}^{N} Q^{(i)}_{e^+} + \bigoplus_{i=1}^{N} Q^{(i)}_{e^-} \right)$$

  The definitions of tensor product and tensor sum are given in the appendix.

III. MODELLING THE COMPOSITE WEB SERVICES ARCHITECTURE

A. Architecture model

We consider an architecture of a company which has two sites (A and B) connected using an IP network. It happens that this company needs to develop a new application in site A to meet the new requirements. The designer noticed that a part of the application that needs to develop, is already running in the form of a Web service on the site B. The designer knows that thanks to the use of technologies of the Web services[1], [2], [3] and [4], it became possible to make a composition of the latter by using an orchestration language [5]. He therefore decides to conceive and to develop a new service but by considering only the needs that are not taken into account during the development of the Web service B. Then, he creates a business process which he deploys in an orchestration engine in the form of a composite Web service. The considered architecture is shown in figure 1.

![Fig. 1. Architecture model](image_url)

B. Assumptions

In our model we assume that all configurations are static. We also assume that the requests messages from customers have a size of 1 Kb, and the response sent by the orchestration engine is 1.4 Kb. The requests sent by the engine and the responses sent by the unit Web Services have the same size (800 bytes).

To calculate the input parameters we assume that the engine (CPU + Disk) can treat 1000 requests/s for the CPU and 500 requests/s for Disk. The servers can treat 1000 requests/s for the CPU and 500 requests/s for Disk.

C. SAN Model

To model the proposed system, we use a model of an open network with blocking. The blocking means that if one buffer of different interfaces is full, then the traffic is blocked. Thus, the routers send to the other servers only the number of requests which they can treat per unit of time (flow control).

This system can be modelled by a set of automaton synchronized by using functional elements, as described in the following.

1) Modelling the interactions related to the requests of the customers and their responses:

   a) **Automaton RA1**: The arrival of requests from customers and their commutations in router noted by RouterA (see figure 1) are modelled by an automaton RA1 where the number of states corresponds to the buffer size of a fast Ethernet interface.

   The considered different events are described in the following:
• **arr_c_ra** is an event which models the arrival of requests from customers at RouterA level. This event is considered as local event. Its rate is defined by \( \lambda_{c,ra} \).

• **comm_c_mtr** is an event which models the commutation of the requests from clients towards the orchestration engine. Synchronization between the engine and the router is ensured by a list of synchronization triplets. It is important to notice here that the occurrence rate associated with these events is not the throughput of the fast Ethernet interface \( (\mu_{ra,fast}) \), but this service rate multiplied by the corresponding routing probability \( p_{ra,mtr} \).

The function \( f \) which appears in the triplets of synchronization allows a restricting arrival according to the queue capacity of the Fast Ethernet interface of RouterA. This function can be defined as follows:

\[
f = \delta(\text{number of responses sent by the orchestration engine} + \text{number of responses sent by the distant server} < C)
\]

Where, \( \delta(b) \) is a Boolean function that returns 1 if the condition is true and 0 otherwise and \( C \) is the buffer capacity.

\[
p_{ra,mtr} = \frac{\text{number of requests sent by the customers}}{\text{numbers of packets in the buffer}}
\]

\[
\mu_{RA1} = (\mu_{ra,fast} * p_{ra,mtr}) * (\mu_{ra,fast} * p_{ra,mtr} \leq \mu_{MTR}) + (\mu_{ra,fast} * p_{ra,mtr} > \mu_{MTR}) * \mu_{MTR}
\]

The automaton RA1 is given in figure 2.

![Fig. 2. Automaton RA1](image)

**b) RA2:** This automaton models the arrival of the responses sent by the orchestration engine and their commutation towards the customers. The different events that may take place are:

- **comm_mtr_ra_c:** this event models the arrival of the responses sent by the engine. The master automaton of this event is the automaton modelling the orchestration engine. Therefore, the occurrence rate of this event is 1.

- **comm_ra_c:** this event models the commutation of the responses sent by the engine towards the customers. This event models the departure of those responses towards outside (customers), therefore this event is a local event. The routing probability of this event is:

\[
p_{ra,c} = \frac{\text{number of responses sent by the orchestration engine}}{\text{numbers of packets in the buffer}}
\]

\[
\mu_{RA2} = (\mu_{ra,fast} * p_{ra,c})
\]

The corresponding automaton is described in figure 3.

![Fig. 3. Automaton RA2](image)

2) Modelling the interactions related to the requests sent by the engine to the remote Web service and their responses:

When the requests of the customers arrive at the level of the orchestration engine, it must invoke one of two Web services to meet those requests. For the WEB service B, the requests must go through the two routers to reach the remote server and the responses will follow the same path but in the way back.

However, before describing this process for the Web service B, we give a description of two automata modelling the engine and the Web service A.

- **Orchestration engine (MTR):** The orchestration engine can be modelled by an automaton with five states. The engine can be in the state:

  - **Available:** in this state, the engine waits the requests of the customers which can arrive at any time;
  - **Examine:** in this state, the engine analyzes the received requests; it extracts the parameters of the methods exposed by the Web Services that it must invoked;
  - **AwaitingA:** in this state the engine waits the WEB service A responses time which it invoked;
  - **AwaitingB:** in this state the engine waits the responses of the WEB service B which is invoked;
  - **Prepare:** in this state the engine build the responses of the customer requests.

  The different events that may take place are:

  - **comm_c_mtr:** this event models the commutation of requests from customers towards the engine. The occurrence rate for this event is 1. This event makes it possible to pass from the **Available** state towards the **Exa min e** state.
  - **invoke_SWA:** this event models the invocation of the Web service A. This event makes it possible to move from **Exa min e** state towards the **AwaitingA** state. Its occurrence rate is the \( \mu_{MTR} \) multiplied by the routing probability \( \pi_A \).
  - **invoke_SWB:** this event models the invocation of the Web service B. This event makes it possible to pass from **Exa min e** state towards the **AwaitingB** state. Its occurrence rate is the \( \mu_{MTR} \) multiplied by the routing probability \( \pi_B \).

  The function \( g \) which appears in the triplet of this event allows a restricting of arrived according to the queue capacity of the serial interface of RouterA. This function can be defined as follows:

\[
g = \delta(\text{current number of requests sent by the orchestration engine} < C)
\]

- **comm_dist_ra_mtr:** this event models the arrival of
the responses of the remote WEB service. Its occurrence rate is equal to 1 (automaton of this event is RA3). This event makes it possible to pass from the AwaitingB state towards the Prepare state.

- **comm_swa_mtr**: this event models the arrival of the responses of the WEB service A. Its occurrence rate is equal to 1. This event makes it possible to move from the AwaitingA state towards the Prepare state.

- **comm_mtr_ra_c**: this event models the sends of the responses towards the customers. Its occurrence rate is $\mu_{MTR}$. This event makes it possible to pass from state prepare towards the available state. The function $f$ which appears in the triplet of this event allows a restricting of arrived according to the queue capacity of the RouterA Fast Ethernet interface.

The corresponding automaton is described in figure 4.

3) **Invocation Process of the remote Web service**: The arrival of the requests and the responses to the level of the RouterA is modelled by two automata RA3 and RA4. RA3 models the insertion of the requests into the buffer of the serial interface and their commutations is given in figure 7, whereas RA3 models the same behavior but with the Fast Ethernet interface is given on figure 6.

- **RA3**: The number of states from the automaton RA3 corresponds to the buffer size of the router A Fast Ethernet interface. The different events that can take place in this automaton are:
  - **comm_rb_ra**: this event models the arrival of the remote server response times at the routerA level. The occurrence rate of this event is 1.
  - **comm_dist_ra_mtr**: this event synchronizes the two Automata (MTR and RA3). This event allows the commutation of remote server responses times towards the engine (the master automaton of this event is the automaton RA3).

The occurrence rate of this event is

$$\mu_{RA3} = (\mu_{ra\_fast} \cdot p_{rb\_ra\_mtr}) + (\mu_{ra\_fast} \cdot p_{rb\_ra\_mtr} \leq \mu_{MTR}) + (\mu_{ra\_fast} \cdot p_{rb\_ra\_mtr} > \mu_{MTR}) \cdot \mu_{MTR}$$

- **RA4**: The number of states from the automaton RA4 corresponds to the serial interface of router A buffer size. The various events that can take place in this automaton are:
  - **Invok_SWB**: this event models the invocation of the Web service B by the engine, which wants to say that this automaton is slave of this event. Thus this event has an occurrence rate equal to 1.
  - **comm_ra_rb**: this event models the commutation of the requests sent by the engine towards the remote Web service.

The occurrence rate of this event is

$$\mu_{RA4} = \mu_{ra\_serial}$$
c) **RB1**: The request arrival and the responses to the level of RouterB are modelled by two automata RB1 and RB2. RB1 models the insertion of the requests in the buffer of Fast Ethernet interface and their commutations, whereas RB2 models the same behavior but with the serial interface.

The automaton RB1 state numbers corresponds to the buffer size of the Fast Ethernet interface. The various events being able to take place in this automaton are:

- **comm_ra_rb**: this event models the commutation of requests sent by router A towards the router B. The master automaton of this event is the automaton RA4. Therefore, the occurrence rate of this event is 1.
- **comm_rb_dist**: this event models the commutation of these requests towards the remote server. Its occurrence rate can be calculated by the following procedure: $\mu_{RB1} = \mu_{rb\_fast} \times (\mu_{ra\_fast} \leq \mu_{srv\_dist}) + (\mu_{ra\_fast} \prec \mu_{srv\_dist}) \times \mu_{srv\_dist}$.

![Diagram of RB1 automaton](image)

**Fig. 8. RB1**

**d) RB2**: RB1 automaton states number corresponds to the serial interface buffer size. The various events being able to take place in this automaton are:

- **comm_dist_rb**: this event models the arrival of the responses of the remote server to the RouterB level.
- **comm_rb_ra**: this event models commutation of the remote server received responses. This automaton is the master automaton of this event. The occurrence rate of this automaton is $\mu_{RB2} = \mu_{rb\_serial}$. The function $T$ which appears in the triplet of this event allows a restricting of arrived according to the queue capacity of the Fast Ethernet interface.

$$T = \delta(\text{Numbers of messages in the buffer of FastEthernet interface of the Router A < C})$$

![Diagram of RB2 automaton](image)

**Fig. 9. RB2**

e) **SWB**: The remote server is modelled by an automaton SWB with two states: in the available state or in the state nonavailable (it treats the requests).

The different events that may take place are:

- **comm_rb_dist**: this event models the commutation of these requests by the router B towards the remote server. But it also models the arrival of these requests at the level of this server. Its occurrence rate is thus 1.
- **comm_dist_rb**: this event models the sends of responses by the remote server towards the orchestration engine. Its occurrence rate is $\mu_{SWB}$. The function $H$ which appears in the triplet of this event allows a restricting of arrived according to the queue capacity of the serial interface of Router B. $H = \delta(\text{Numbers of messages in the buffer of serial interface of the RouterB < C_s})$.

![Diagram of SWB automaton](image)

**Fig. 10. SWB**

### IV. Numerical results

In this section, we give some numerical results in order to show that the architecture ensures the good behavior of the system and to ensure that the performance requirements are always respected. For these results, we used **PEPS** (Performance Evaluation of Parallel Systems) which is a tool that allows at the same time the definition and the resolution of models using this formalism (SAN) and tensor algebra [9]. We have used this tool to provide performance measures. First, we give the parameters of our model.

The performance indices which we seek to evaluate in this model are according to the number of requests of the customers and who are the average number of requests in each buffer of two routers, the utilization rate of two unit Web servers, the orchestration engine and the load of system.

4) The average number of messages in the buffers of the various interfaces: In figure 11, we can notice how the length of the queue varies according to a number of the customer requests. As our model is an open network with blocking; if the buffer of the interface is full, one blocks the traffics coming from outside. And still means that the two routers send to the engine and the other server only, the number of the requests which they can treat per unit of time (flow control). Therefore it is normal that the queue length on the Fast Ethernet interface level of the router believes quickly (the arrival of the requests from outside, the arrival of the responses sent by the remote server and the arrival of the responses sent by the orchestration engine). When this queue becomes full (for 6000 requests), not only the requests coming from outside will be lost but still that will cause the blocking of the whole system.

5) The utilization rate of servers: The figure 12 shows the relationship between the system utilization rates according to the number of customer requests.

6) The load of the system: The figure 13 shows the relationship between the load of the system according to the number of requests of the customers.
V. CONCLUSION

With the emergence of new technologies as web services over IP networks, the performance evaluation is a major issue to ensure the Quality of Service. Using traditional tools as Markov Chains to evaluate the performance evaluation of such systems are not suitable due to the space state explosion. Thus, we used a more adapted method for such systems with Stochastic Automata Networks, to evaluate the performance of the composite Web services. We have used the PEPS tool which allows the representation of the model described with SAN and the calculation of the performance measures. The perspectives of this work is to extend this performance evaluation for WEB services over broadband wireless/Mobile networks and in discrete-time.

REFERENCES


APPENDIX

- The tensor product: The tensor product of two matrices A and B, of dimensions respectively $(n_1, m_1)$ and $(n_2, m_2)$, is a matrix of dimensions $(n_1 \times n_2, m_1 \times m_2)$. We note by $\otimes$ the tensor product. $\times/\div$ represents the usual operation of matrix product/addition and $I_n$ is the identity matrix of order $n$. $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \end{pmatrix}$.

The tensor product $C = A \otimes B$ is given by:

$C = \begin{pmatrix} a_{11}B & a_{12}B \\ a_{21}B & a_{22}B \end{pmatrix}$

Thus:

$C_{i,k,l} = a_{i,j}b_{k,l}$ (1)

with $i \in [1 \ldots n_1]$, $j \in [1 \ldots m_1]$, $k \in [1 \ldots n_2]$ and $l \in [1 \ldots m_2]$.

- The tensor sum: The tensor sum of two square matrices A and B is defined in terms of tensor products as: $A \oplus D = (A \otimes I_{n_2}) + (I_{n_1}B)$.

You can notice that the tensor product is defined for non square matrices, whereas the tensor sum is defined only for the square matrices.