Preface

Welcome to the Fifth International Workshop on Practical Applications of Stochastic Modelling, held in conjunction with the International Conference on Performance Engineering in Karlsruhe. PASM is organised as a friendly and inclusive event at which to share interesting and novel applications and developments in modelling theory. We strive to maintain an air of informality to the workshop which we hope fosters discussion and potential collaboration. This volume is the ‘on-the-day’ proceedings for delegates. In common with previous PASM workshops, the official proceedings will be published later in Elsevier’s Electronic Notes in Theoretical Computer Science. If you wish to cite a paper from the workshop, please use the ENTCS citation which we will place on the PASM website when it is available. The workshop organisers would like to take this opportunity to extend their thanks to Michael Mislove, executive editor of ENTCS, for his continued support for PASM.

This is the fifth time PASM has been held, previous events were held in London in 2004, Newcastle in 2005, Palma de Mallorca in 2008 and London in 2009. On each occasion PASM has been collocated with a different host conference. This year we are delighted to be associated with ICPE for the first time. We would like to thank Samuel Kounev for inviting us and for his help in organising PASM. We would also like to extend our thanks to Niko and Heiko for all their work in the local arrangements for PASM and for handling the registration process. Particular thanks must go to the programme committee, who thoroughly refereed all the submissions and made some potentially difficult choices easier by their detailed comments.

Finally we would like to thank you, the authors and delegates. Without you there would be no papers, no presentations, no questions, no discussion and consequently no PASM. This is your workshop, we hope you enjoy it!

Jeremy Bradley, William Knottenbelt and Nigel Thomas

PASM Co-chairs.
Programme Committee

Jeremy Bradley
Jeremy Bryans
David Daly
Nick Dingle
Paulo Fernandes
Katya Gilly
Stephen Gilmore
Marco Gribaudo
Uli Harder
Félix Hernández-Campos
András Horváth
Carlos Juiz
Peter Kemper
Leila Kloul
William Knottenbelt
Andrea Marin
Dave Parker
Anne Remke
Nigel Thomas
Mirco Tribastone
Aad van Moorsel
Maria Vigliotti
Sabine Wittevrongel
Katinka Wolter
Soraya Zertal
Avelino Zorzo
Contents

Invited talk: Markus Siegle .......................... 1
Invited talk: Giuliano Cassale ....................... 3
Paulo Fernandes, Afonso Sales, Alan R. Santos and Thais Webber
Performance Evaluation of Software Development Teams: A
Practical Case Study .................................. 5
Gabriel Alves Jr, Paulo Maciel, Ricardo Lima, Fábio Magnani and
Adilson Arcoverde Jr.
Towards the evaluation of environment and business trade-offs in
supply chains ....................................... 23
Alessio Angius and András Horváth
The Monte Carlo EM method for the parameter estimation of
biological models ................................... 43
Tzu-Ching Horng, Nikolas Anastasiou, Tony Field and William
Knottenbelt
LocTrackJINQS: An Extensible Location-aware Simulation Tool for
Multi-class Queueing Networks ...................... 55
Enrico Barbierato, Marco Gribaudo and Mauro Iacono
Defining formalisms for performance evaluation with SIMTHESys .... 73
Ricardo M. Czekster, Paulo Fernandes, Afonso Sales, Thais Webber and
Avelino F. Zorzo
Stochastic model for QoS assessment in multi-tier web services .... 93
Timothy L.Y. Leung and William J. Knottenbelt
Stochastic Modelling and Optimisation of Internet Auction Processes .... 111
Wen Zeng and Aad van Moorsel
Quantitative Evaluation of Enterprise DRM Technology ............. 131
Yasir Shoaib and Olivia Das
Web Application Performance Modeling Using Layered Queueing
Networks ............................................. 149
Osama Younes and Nigel Thomas
Analysis of the Expected Number of Hops in Mobile Ad Hoc
Networks with Random Waypoint Mobility ..................... 169
How Can Stochastic Modelling Techniques Meet Practical Needs?

Markus Siegle

Department of Computer Science
Universität der Bundeswehr München, Germany
markus.siegle@unibw.de

1 Introduction

Designing and managing today’s information and communication systems is a true challenge, due to their enormous, continuously increasing complexity. Such systems have to meet a wide range of formidable requirements, not only concerning their functionality, but also with respect to non-functional, quantitative aspects such as performance and dependability.

Stochastic modelling and analysis is a well-established discipline which focuses on the evaluation of quantitative measures of systems. Over more than four decades, a wide range of sophisticated modelling methods and analysis algorithms has been developed. Many of these have been implemented in available software tools, which took place mostly in the academic environment but also, to a lesser extent, in industry labs. Even though these methods and tools are highly developed and extremely powerful, for some reason or other they often reach their limits when facing the needs of real-world systems. Therefore, unfortunately, the practical success of stochastic modelling techniques has been limited. In our opinion, two main problems exist which constitute obstacles for a wider applicability of these techniques: Firstly, scalable parallel or distributed systems with a high degree of concurrency lead to models with extremely large state spaces, which renders most state-space-based analysis techniques intractable. Secondly, setting up an appropriate model of a complex system is a difficult task, which requires the combined knowledge of a system designer and a modelling specialist, which is rarely found in the same person.

2 Decision Diagrams: A Powerful Data Structure for Stochastic Modelling

Among the techniques addressing the first problem, the so-called symbolic approach, which relies on the use of decision diagrams (DD) as its basic data structure, has shown to be very effective. Starting from a formal model specification (expressed, for example, in the language of stochastic Petri nets or stochastic process algebra), a compact symbolic representation of the underlying transition system is generated automatically, and all subsequent steps of manipulation and analysis, including reachability analysis, elimination of vanishing states, as well
as different forms of numerical analysis and the computation of the measures of interest, can be performed in an efficient manner based on this data structure. The key to success with this symbolic approach lies in the proper exploitation of the system’s compositional structure, which is reflected in the structure of the decision diagram, leading to its compactness and efficiency of manipulation.

3 Automatic Generation of Models from Application-Oriented Formalisms

As for the second problem mentioned in the introduction, we are currently developing an approach which aims at automatically extracting dependability models from industrial-type specifications commonly used in systems engineering, such as AADL or SysML. Instead of a direct transformation from an application-level specification to a formal model, we propose a two-step methodology at whose core is an intermediate language dedicated to the description of a system’s dependability aspects. The design goals of this domain-specific language, called LARES, were to keep it very simple, clear and concise, but generally applicable, and to support a modular and compositional modelling style. Its main elements are so-called behaviours, expressed as finite state machines, and instantiable modules which are characterised by one or more such behaviours. Larger modules can be hierarchically composed from submodules, and Boolean conditions are used to guard transitions. LARES can express non-trivial failure behaviour (such as non-Boolean components, multiple failures with common cause, or fault propagation), and it can also express stochastic features, such as transition probabilities, Markovian transition rates, and in principle also general distributions. The overall methodology consists of three steps: In the first step, the relevant aspects are extracted from the application-oriented specification and expressed in the intermediate language. In the second step, the model in the intermediate formalism can be edited, thereby enhanced with dependability parameters not present at the application-oriented level. Finally, in the third step, the intermediate model is translated to the target formalism, which may be a stochastic Petri net, a stochastic process algebra, or a simulator.

4 Conclusion

The two approaches just sketched should be seen in the larger context of current work on quantitative modelling techniques, such as the mean field approach to the approximate analysis of massively scalable systems, probabilistic model checking techniques or the performance tree approach to the description and computation of complex performance or dependability queries, or several activities in the field of model-driven performance and schedulability analysis of reactive systems, only to name a few. We believe that approaches such as these will play an important role in making stochastic modelling more accessible to a larger group of users, thereby meeting the existing needs of practitioners.
Multi-Tier Headaches: Modelling Web Applications in a Non-Markovian World

Giuliano Casale
Imperial College London
Department of Computing
180 Queen’s Gate
London, SW7 2AZ; UK
g.casale@imperial.ac.uk

Abstract
Stochastic modelling techniques for web applications are often based on simplifying assumptions, such as exponential distributions of inter-arrival or service times, which enable analytical tractability of the underlying Markov process. Even though for many workloads and real systems these can be appropriate abstractions, it is easy to find cases where such assumptions lead to large modelling errors. For example, flash crowds and service time correlation defy the basic assumptions of Poisson or renewal processes, hence standard web applications usually yield very large errors. In a similar way, workloads arising from dynamic compositions in service-oriented environments often show variabilities and correlations that are much larger than in a Poisson model and that cannot be ignored in performance prediction.

In this talk, I will discuss a recently proposed family of queueing networks where service and arrival times are described by Markovian Arrival Processes (MAPs), a class of nonrenewal Markov-modulated processes. After introducing MAPs, I will show that the related queueing networks do not admit a product-form solution, but still performance metrics can be estimated efficiently by approximate methods such as linear programming bounds, entropy-based methods, and mean value analysis formulas which generalize established relations for product-form models.

Biographical Sketch
Giuliano Casale received the MSc and PhD degrees in computer engineering from Politecnico di Milano, Italy, in 2002 and 2006 respectively. He joined the Department of Computing at Imperial College London in 2010 where he is currently an Imperial College Junior Research Fellow. His research interests include performance modeling, workload characterization, stochastic scheduling, simulation, and resource consumption estimation. Prior to joining Imperial College London, he was a full-time researcher at SAP Research UK in 2009, and a postdoctoral research associate at the College of William and Mary, Virginia, between 2007 and 2008. In Fall 2004 he was a visiting scholar at UCLA studying bounds for queueing networks. He has published more than 60 papers in journals, conferences, and book chapters. He is a member of the ACM, the IEEE, and the IEEE Computer Society.
Performance Evaluation of Software Development Teams: a Practical Case Study

Paulo Fernandes¹,²,*, Afonso Sales³, Alan R. Santos, Thais Webber¹

Pontifícia Universidade Católica do Rio Grande do Sul
Avenida Ipiranga, 6681 – Prédio 32
90619-900 – Porto Alegre – RS – Brazil
Tel.: +55-51-3320-3611
Fax: +55-51-3320-3621

Abstract
Software development projects have become a challenge for both industry and academia regarding the performance evaluation of teams. Recently, a Stochastic Automata Networks (SAN) model was proposed as theoretical representation for performance prediction of software development teams. In this paper, we present an exercise of such SAN analytical modeling for a practical case study from an Information Technology company that has multiple sites and different participants’ roles and expertises. We present the matching of our model predictions with the actual project observations. Then, we focus our attention on the central entity varying its availability and the level of provided support in order to observe the impact on the participants’ performance. We summarize our study with further discussions of numerical results and possible model extensions.

Keywords: Performance Evaluation, Analytical Modeling, Stochastic Automata Networks, Global Software Development, Team Building Process

1. Introduction
In order to improve business results, there is a demand to re-structure the Information Technology (IT) sector extending companies operations to offshore software development centers. Globalization and the increasing need of companies to expand their markets spreading operations in several countries demands an organizational improvement including the Information Technology area and software development projects. Furthermore, there are research challenges in the software development field such as heterogeneous team configurations distributed in separate sites, or even different time zones, inter-cultural factors, different experience levels and technical background [1, 2, 3, 4].

*Corresponding author. The order of authors is merely alphabetical.
Email addresses: paulo.fernandes@pucrs.br (Paulo Fernandes), afonso.sales@pucrs.br (Afonso Sales), alan.ricardo@acad.pucrs.br (Alan R. Santos), thais.webber@pucrs.br (Thais Webber)
¹Authors receive financial support provided by PUCRS-Petrobras (Conv. 0050.0048664.09.9).
²Paulo Fernandes is also funded by CNPq-Brazil (PQ 307284/2010-7).
³Afonso Sales receives grants from CAPES-Brazil (PNPD 02388/09-0).

Preprint submitted to International Workshop on Practical Applications of Stochastic Modelling February 28, 2011
One of the main challenges in multi-site software development projects is the communication area, because it is hard to ensure the same understanding about projects goals among people from different cultures and knowledge level [5]. Large companies have been using a set of processes to facilitate the work of different teams. However, teams’ performance analysis becomes challenging for both industry and academia [6, 7, 8, 9, 10]. Theoretical models can be a useful tool to analyze, for instance, the evolution and intercommunication of software development processes in order to help project managers to better understand issues related to the development context [11, 12, 13]. Related works concerning stochastic models and simulations are developed towards to the specification of the dynamics of software projects [6, 14], and the usage of analytical models to analyze teams performance variability [15, 16].

Stochastic Automata Networks (SAN) [17, 18] is a powerful modeling formalism based on Markov chains [19] that provides a high-level description (abstraction) of a model. SAN is a suitable formalism for modeling software development projects due to the fact that development teams can be easily abstracted in a modular way. This formalism describes each module as a stochastic automaton depicted by a state-transition diagram, where the transitions are labeled with probabilistic and timing information. The behavior of a SAN model is given by the occurrence of events, which allows the changes of the states of one or more automata. In a SAN model, an estimated duration is associated to each event determining how often each event can occur. Giving this timing information, the numerical solution of the model provides steady-state probabilities from which measures of interest can be extracted, e.g., performance indices. Among a myriad of tools to provide these indices [20, 21, 19], we use a numerical solver called GTAexpress [22] to obtain the performance indices from the analytical models.

In this paper, we report our findings on the use of SAN for analytical modeling of software development teams in order to predict their performance in different scenarios. We present our results based on a case instance of a multi-site project analyzing the effect of availability and levels of support provided by a centralized management entity. In order to verify our prediction accuracy, we validate the numerical results obtained from the proposed model comparing with the actual hours spent in the project’s phases. Following, we turn our attention to a deeper analysis of possible scenario variations in the project considering different behaviors and skills of participants. Specifically, the availability and quality of the central entity support is analyzed in different scenarios, and the impact on the whole team productivity is predicted.

The remainder of this paper is organized as follows. Section 2 presents a study of analytical modeling of global software development teams and the representation of the central entity and team members. Section 3 introduces the practical case study to illustrate the scope and nature of the problem, and also presents the proposed analytical model. Section 4 presents the comparative numerical analysis of the proposed model with the actual project quantitative data. Moreover, in Section 4.3, we vary the availability and levels of support provided by the central entity to analyze the impact of these characteristics on the total project execution time. Finally, the conclusion draws an overview on future work and major contributions of the paper.

2. Analytical modeling of teams

Software development teams evaluation is a challenging process, since team building is a multi-variable problem in order to achieve better team’s performance. Despite the clear usefulness of analytical modeling to predict team’s behavior during project’s planning phase, other software development phases such as execution, monitoring and controlling, and closing [23], can also benefit from these theoretical predictions. Traditionally, literature presents different
approaches related to the usage of numerical analysis in the software engineering context, e.g.,
the automated software testing process [24, 25, 26] and quantitative evaluation of development
teams [15, 16, 27, 28].

Analytical modeling formalisms are commonly applied to describe many realities in a state-
based approach. Markov chains [19] and Markov-based formalisms are employed in several areas
such as economics, physics, engineering and bioinformatics, to cite a few. More information
about other formalisms, specialized tools, and applications is available in the literature [29, 30,
20, 31, 32, 18, 33, 34, 35, 36].

A modular and compact Markovian formalism is Stochastic Automata Networks (SAN) [17,
18]. SAN is used to model systems by small components (automata) with occasional and defined
interactions among them in a structured manner. The solution of a SAN model, i.e., the extraction
of numerical results, is usually performed by specific algorithms [22] designed to deal with rather
large state spaces. For this reason, SAN facilitates the modeling of distributed development teams
(in a software engineering context) by the individual description of each entity behavior, where
only some activities represent synchronizations among some entities. In this context, an entity
can be modeled as a participant or a whole team.

Distributed software development teams are usually dispersed among different locations
where there is a central entity or team to manage and coordinate them [9]. The central team
can be a single person or a group playing the role of delivery manager, research and development
manager, or even project manager, according to the particular denomination used. In the context
of geographically dispersed development projects, there are issues that are considered key factors
for the project success such as effective communication and interactions among teams [37]. In
fact, the interactions effect on team’s performance becomes even more important in projects with
a centralized control, where the availability and expertise of the central entity are usually more
relevant than the expertise of the developers [9].

Based on those concepts, we focus our attention on the central team’s availability and activ-
ities to obtain an analytical representation of this entity. Figure 1 presents two automata (Avail-
ability and Activities) that represent the main central team’s behaviors.

![Central Team Automata](image)

**Figure 1: Central team’s automata**
In Figure 1, automaton *Availability* is modeled with two states, where *state A* represents that the central team is available to cooperate with participants, and *state U* represents that the central team is unavailable for any reason, e.g., time-zone restrictions, other project assignments, or meetings. Automaton *Activities* indicates the main tasks performed by the central team, and it is composed of two states: *state M* represents that the central team is performing any other management activities (according to the specific scenario); and *state C* represents that the central team is effectively cooperating with one participant. Events are defined to enable the transitions among automata’s states, where each event has an associated occurrence rate, according to the project’s characteristics. Events *a* and *u* are local to the automaton *Availability* indicating the period of time in which the central team remains available to cooperate with participants. It is important to remark that a software development team is composed of *N* participants that communicate among themselves and with a central team to solve issues and collaborate. Then, event *co_i* (*i=1..N*) is a synchronizing event between central team and the *i*-th participant, where its occurrence represents that the central team changes its current activity from management (*state M*) to collaboration (*state C*). Event *s_i* reflects mostly the level of central team’s expertise, since its synchronizing occurrence indicates that the central team have provided a support for the *i*-th participant, *i.e.*, changing the central team activity from *state C to state M*.

In the context of software development teams, a participant may play a given role in the project such as software developer, business analyst, tester, data warehouse engineer, database administrator or an user. However, independent of team roles assignment, participants have a similar behavior that can be modeled in a generic form, taking into account their general duties to accomplish, including issues to be solved according to their roles. Figure 2 depicts the automaton that represents the entity *Participant* with his/her main activities.

![Participant Automaton](image)

**Figure 2:** Participant’s automaton

In Figure 2, a participant is modeled by one automaton composed of the following states: *state W* represents that the participant is working, *i.e.*, completing its tasks or collaborating with other members; *state S* represents that the participant is seeking for a specific solution, information, documentation, sources of data or even learning some technical issue by its own; and *state C* represents that the participant is collaborating with the central team to solve technical issues, or for example to discuss project decisions. As presented before, the events *s_i* and *co_i* represent the synchronization of the *i*-th participant with the central team *Activities* automaton, changing
the automaton’s state from collaborating (C) to working (W), and from seeking solution (S) to collaborating (C) respectively. Additionally, this automaton has two local events: \( e_i \) and \( r_i \). Event \( e_i \) indicates that the \( i \)-th participant has an impediment to accomplish his/her tasks and starts seeking a proper solution, i.e., the occurrence of this event changes the participant’s state from \( W \) to \( S \). Once an impediment is solved, the participant returns to work and this behavior is represented by the occurrence of event \( r_i \) that changes the participant’s state from \( S \) to \( W \).

Table 1 presents a summary of the events of the two entities (Central Team and Participant) previously illustrated in Figures 1 and 2.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>Available: this event is independent of other automata states and indicates that the central team becomes available to manage and collaborate.</td>
</tr>
<tr>
<td>( u )</td>
<td>Unavailable: this event is independent of other automata states and indicates that the central team becomes unavailable to collaborate.</td>
</tr>
<tr>
<td>( e_i )</td>
<td>Impediment: when this event occurs the ( i )-th participant goes to the state where he/she need to seek a solution.</td>
</tr>
<tr>
<td>( r_i )</td>
<td>Resume working: this event is independent of other automata states, indicating that the ( i )-th participant resumes work after seeking the solution by him/herself.</td>
</tr>
<tr>
<td>( co_i )</td>
<td>Collaborate: this event synchronizes ( i )-th participant automaton with central team Activities automaton, starting the collaboration between them.</td>
</tr>
<tr>
<td>( s_i )</td>
<td>Provided support: the occurrence of this event synchronizes both ( i )-th participant and central team Activities automata, indicating that the participant resumes work after the central team has provided support during the collaboration.</td>
</tr>
</tbody>
</table>

Using both entities depicted in Figures 1 and 2, one can build an analytical model representing a software development project with \( N \) participants that collaborate to solve issues. We present in the next section an instantiation of a case study of a real development process with fourteen participants \((N=14)\) and a central team (represented by 2 automata), i.e., a 16 automata model.

3. Practical case study

In order to demonstrate that analytical modeling is useful to complex state-based performance analysis of software development teams in multi-site context, we present a model of a practical case study. In the context of this paper, we named the case study project as ALPHA due to confidentiality.

ALPHA was a Data Warehouse project executed by an IT company using multiple locations: Brazil, USA, Malaysia and India. This project was executed during 11 months (i.e., \( 11 \times 22 \) workdays = 242 days) and its goal is to improve data availability and data analysis for a certain company management level. The project ran with an adequate infrastructure to integrate the participants in a global environment facilitating communication, interactions, knowledge sharing and web-based support.
Table 2 presents the teams’ configuration considering *senior/junior* participants distributed among sites. In this table the last column indicates the average percentage of working hours each participant was allocated to the project.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Role</th>
<th>Expertise</th>
<th>Location</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Delivery Manager</td>
<td>Senior</td>
<td>USA</td>
<td>25%</td>
</tr>
<tr>
<td>1</td>
<td>Project Manager</td>
<td>Senior</td>
<td>Brazil</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Developer</td>
<td>Senior</td>
<td>Brazil</td>
<td>100%</td>
</tr>
<tr>
<td>3</td>
<td>Developer</td>
<td>Junior</td>
<td>Brazil</td>
<td>75%</td>
</tr>
<tr>
<td>1</td>
<td>Tester</td>
<td>Senior</td>
<td>Brazil</td>
<td>20%</td>
</tr>
<tr>
<td>2</td>
<td>Business Analyst</td>
<td>Senior</td>
<td>USA</td>
<td>10%</td>
</tr>
<tr>
<td>1</td>
<td>Data Warehouse Engineer</td>
<td>Junior</td>
<td>USA</td>
<td>3%</td>
</tr>
<tr>
<td>3</td>
<td>User</td>
<td>Senior</td>
<td>USA</td>
<td>3%</td>
</tr>
<tr>
<td>1</td>
<td>System Engineer</td>
<td>Junior</td>
<td>Malaysia</td>
<td>5%</td>
</tr>
<tr>
<td>1</td>
<td>Database Administrator</td>
<td>Senior</td>
<td>India</td>
<td>5%</td>
</tr>
<tr>
<td>1</td>
<td>Data Warehouse Engineer</td>
<td>Senior</td>
<td>India</td>
<td>3%</td>
</tr>
</tbody>
</table>

In Project *Alpha*, interactions among participants were performed on different ways, *e.g.*:

- Data Warehouse Engineers (India/USA) working with Brazil development resources;
- Users from USA working with Brazil development resources;
- Users from USA working with Brazil test resource;
- Business Analysts from USA collaborating with development resources;
- Business Analysts from USA collaborating with test resource;
- System Engineer from Malaysia working with Brazil development resources;
- Database Administrator from India working with Brazil development resources.

Project *Alpha* was composed of the following phases:

- **Initiating**: project goals definition, creation of a preliminary scope statement to help teams work towards a common objective. In this phase, the team identifies the vision, scope and initial constraints;
- **Planning**: assessment of the existing environment on solution design level. A solution is planned and designed to map project constraints, *i.e.*, a master project plan is delivered;
- **Execution**: the application components are created based on the development plans, as well as software components are tested to ensure that the solution operates properly;
- **Monitoring and Controlling**: in this phase, the tested solution is promoted to production and transferred to operations.
This project had also different types of challenges and impediments detected during execution phase. The distance factor and central team allocation have created certain challenges as the support quality from central team and time-zones alignment. For this particular case, central team members are also allocated as resources of other projects setting the central team support quality from medium to low. The time-zone challenge can bring difficulties to members of a given site, making them to extend or change their workday window to be available to work with external site project team members. For example, sometimes a Malaysian participant needs to extend his/her workday window to be able to chat with a Brazilian participant earlier in the morning.

For our analysis, and also for the project execution, an impediment is an emerged issue that must be solved and, often, this issue is dependent of central team’s level of support. Examples of project impediments are enterprise environmental factors that refer to any factor that surround or influence the project success, or organizational shared services that refer to outside project services that also influence the project success. Enterprise environmental factors can come from any of the enterprises involved in the project and may include the organizational culture and structure, existing resources, marketing conditions, government or industry standards and project management information systems. Organizational shared services are linked to outside project services such as network engineers, system engineers and security consultants.

3.1. Instantiating the case study

Our proposed model uses both automata presented in Figures 1 and 2 (Section 2) that represents the abstraction of Project Alpha. Figure 3 presents the SAN model for Project Alpha based on the configuration presented in Table 2. In this model, the project and delivery managers are abstracted as a central team entity. Therefore, activities of these managers are encapsulated in automaton Activities, as well as their availability to interact with other participants are encapsulated in automaton Availability (Figures 1).

In the model presented in Figure 3, each participant is modeled as an automaton of three states (W, S and C) representing their possible activities in a workday. Note that the software development team (Table 2) is composed of $N = 14$ participants, where 5 participants are from Brazil, 6 from USA, 1 from Malaysia and 2 from India. The participants (developers, tester, business analysts, data warehouse engineers, users, system engineer and database administrator) located in different sites and with different expertises, need to report and collaborate with the project and delivery managers, as well as they collaborate with other participants when needed.

It is important to remark that the interactions among participants (previously described in Section 3) are encapsulated in participants’ state $W$. In this paper, we are only interested on evaluating the interactions between participants and central team. In this scenario, project and delivery managers act as a central team that coordinates the tasks and project completion. However, central team has an availability parameter that indicates the amount of time available for collaboration/management activities and, in many cases, this becomes a bottleneck for asynchronous and synchronous interactions among participants.

In this model, remark that the central team interacts only with one participant at each time, which represents a centralized management that is responsible to coordinate individual actions, reducing the risk of conflicts [9]. This approach aims to limit the dependencies among the participants, especially if they are distributed in multi-sites. Of course, in practice software development teams have adopted more flexible processes [38] to easily overcome communication difficulties in the context of global software development.
Summarizing, those interactions are dependent on central team availability (e.g., time-zones, other project tasks), levels of support, knowledge level, and capacity to solve issues or project impediments. We are not directly considering intercultural factors, levels of task complexity, effort and actual hours for each task. This model also assumes workload equally distributed among team members, interactions such as e-mails exchanges, instant messaging, conference calls and video conferences.

3.2. Setting model’s parameters

Beyond model mapping, event rates estimation is an important phase of analytical modeling. The model dynamics is given by the definition of events that enable the state transitions in the stochastic automata network (Figure 3). Table 3 shows the estimated values for the model’s event rates, considering an eight-hour workday regime. These estimated values were obtained from Project ALPHA’s historical/surveyed data and interviews with managers and participants, collecting quantitative data based on their experiences during the project.
Regarding participant’s local events $e$ and $r$, in this project instantiation, both have rates reflecting the level of expertise, where the participants are classified in two types: Junior and Senior. In addition, synchronizing events $co$ do not represent the frequency of cooperations between participants and central team, but these events are modeled only to represent the resource allocation, in this case central team allocation. Hence when the participant is in state $S$ (seeking solution) and the central team is available, the cooperation starts immediately.

Therefore, events $co$ have functional rates that are dependent of the state of central team’s automaton Availability, i.e., there is an availability function (named $disp$) that allows two possible evaluations: (i) events $co$ have a virtually infinite rate, if automaton Availability is in state $A$; and (ii) events $co$ have a zeroed rate, if automaton Availability is in state $U$.

Table 3: Estimated event rates for an eight-hour workday

<table>
<thead>
<tr>
<th>Type</th>
<th>Event</th>
<th>Description</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>loc</td>
<td>$a$</td>
<td>Central team is available to collaborate with participants on average 2 hours per workday, i.e., central team collaborates on a rate of 4 times per workday.</td>
<td>8/2</td>
</tr>
<tr>
<td>loc</td>
<td>$u$</td>
<td>Central team is unavailable to cooperate with participants on average 6 hours per workday.</td>
<td>8/6</td>
</tr>
<tr>
<td>loc</td>
<td>$e$</td>
<td>Junior participants work on average 1 hour per workday without any kind of central team support. Due to their expertise, senior participants work on average 7 hours per workday without any kind of central team support.</td>
<td>8/1 8/7</td>
</tr>
<tr>
<td>loc</td>
<td>$r$</td>
<td>Junior participants spend on average 7 hours per workday seeking solutions. Senior participants spend on average only 1 hour per workday seeking solutions because of their expertise.</td>
<td>8/7 8/1</td>
</tr>
<tr>
<td>syn</td>
<td>$co$</td>
<td>Once a participant needs to collaborate with the central team, the collaboration occurs immediately if the central team is available.</td>
<td>$disp$</td>
</tr>
<tr>
<td>syn</td>
<td>$s$</td>
<td>Due to the central team support quality (from medium to low), the collaboration takes on average 2 hours per workday.</td>
<td>8/2</td>
</tr>
</tbody>
</table>

The whole model with states, transitions, events and their associated rates can be numerically solved in order to obtain the steady-state probabilities of the model. Based on these probabilities, it is possible to calculate the team’s performance for different scenarios varying, for example, the central team’s availability and the level of expertise of participants.

4. Numerical analysis

The actual quantitative results about Project ALPHA were assembled from compiling project log files, informal interviews with the managers, and a survey for collecting quantitative information about participants interactions and issues found during project phases. The analytical model results were obtained using Project ALPHA data to instantiate the event rates (Section 3.2).
4.1. Project Alpha quantitative data

The project quantitative data was collected focusing mainly on the execution phase working hours and on the impediments occurrence during the project. Impediments were described by participants and central team managers as effective interactions among them.

Table 4 shows the estimated effort and actual hours related for each phase of Project Alpha. It is important to remark that approximately 3,317.22 hours were actually spent by participants completing their tasks in the execution phase.

<table>
<thead>
<tr>
<th>Project phase</th>
<th>Estimated</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initiating</td>
<td>611.70 h</td>
<td>771.65 h</td>
</tr>
<tr>
<td>Planning</td>
<td>1,529.25 h</td>
<td>895.60 h</td>
</tr>
<tr>
<td>Execution</td>
<td>3,364.35 h</td>
<td>3,317.22 h</td>
</tr>
<tr>
<td>Monitoring and Controlling</td>
<td>611.70 h</td>
<td>438.80 h</td>
</tr>
</tbody>
</table>

Project Alpha faced issues resulting in a significant amount of impediments in the project execution phase. Table 5 summarizes these impediments grouped in eight categories: (i) Resource indicates problems related to project resource constraints, allocation, changes and replacements; (ii) Technology indicates issues concerning tools used within the project; (iii) Process indicates problems related to the development process exceptions during project execution; (iv) Requirements indicates the requirement information gathering issues; (v) Schedule indicates delays in project deliverable dates; (vi) Deliverable indicates code or project artifacts, e.g., documents, issues; (vii) Scope indicates issues to project scope changes; and (viii) Infrastructure indicates problems with application infrastructure.

In Table 5, we present the quantity, category, average, and total duration of project’s impediments. We can notice in this table that Project Alpha has spent about 332 hours dealing with impediments.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Category</th>
<th>Average</th>
<th>Total duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Resource</td>
<td>8 h</td>
<td>72 h</td>
</tr>
<tr>
<td>3</td>
<td>Technology</td>
<td>4 h</td>
<td>12 h</td>
</tr>
<tr>
<td>2</td>
<td>Process</td>
<td>8 h</td>
<td>16 h</td>
</tr>
<tr>
<td>11</td>
<td>Requirements</td>
<td>4 h</td>
<td>44 h</td>
</tr>
<tr>
<td>4</td>
<td>Schedule</td>
<td>8 h</td>
<td>32 h</td>
</tr>
<tr>
<td>4</td>
<td>Deliverable</td>
<td>24 h</td>
<td>96 h</td>
</tr>
<tr>
<td>1</td>
<td>Scope</td>
<td>30 h</td>
<td>30 h</td>
</tr>
<tr>
<td>6</td>
<td>Infrastructure</td>
<td>5 h</td>
<td>30 h</td>
</tr>
<tr>
<td>40</td>
<td>Total</td>
<td>8.3 h</td>
<td>332 h</td>
</tr>
</tbody>
</table>
4.2. Model quantitative results

This section presents the quantitative results obtained from the proposed model numerical solution, according to the parameters defined in Table 3 (Section 3.2). The numerical solution is primarily expressed by the steady-state probability of the SAN model. From these probabilities and from the participants individual hours allocated to Project \textit{Alpha}, we can determine the average working hours \textit{per} eight-hour workday. Readers interested in more information about the software tool for numerical solution of SAN models refer to \textit{PEPS} (Performance Evaluation of Parallel Systems) [20].

Table 6 shows the results of the main entities of the proposed model (Central Team, Senior, and Junior participants) and their correspondent steady-state probabilities.

<table>
<thead>
<tr>
<th>Entity</th>
<th>State</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central Team</td>
<td>$A$</td>
<td>25.00%</td>
</tr>
<tr>
<td></td>
<td>$U$</td>
<td>75.00%</td>
</tr>
<tr>
<td></td>
<td>$M$</td>
<td>56.27%</td>
</tr>
<tr>
<td></td>
<td>$C$</td>
<td>43.73%</td>
</tr>
<tr>
<td>Seniors</td>
<td>$W$</td>
<td>87.09%</td>
</tr>
<tr>
<td></td>
<td>$S$</td>
<td>11.96%</td>
</tr>
<tr>
<td></td>
<td>$C$</td>
<td>0.95%</td>
</tr>
<tr>
<td>Juniors</td>
<td>$W$</td>
<td>14.70%</td>
</tr>
<tr>
<td></td>
<td>$S$</td>
<td>78.26%</td>
</tr>
<tr>
<td></td>
<td>$C$</td>
<td>7.04%</td>
</tr>
</tbody>
</table>

Observing the results presented in Table 6, it is possible to notice that senior participants have, as expected, high autonomy to deal with their tasks. However, the probability of working state ($W$) for senior participants is slightly smaller than the 87.5\% corresponding to the seven hours of working \textit{per} eight-hour workday. Observing the working percentage of junior participants, we find a quite low value of 14.7\% which corresponds to a little more than one hour of work \textit{per} eight-hour workday, while most of their time (more than six hours per workday) is spent in state $S$ (seeking solution). Junior participants do not collaborate very often with the central team (half an hour \textit{per} workday), which is probably a side effect of the low availability and low quality provided support.

Table 7 shows the participants individual working hours taking into account the average percentage of working hours each participant was allocated to the project (see Table 2). According to Table 4, Project \textit{Alpha} spent 3,317.22 hours in the execution phase, instead of the initially estimated 3,364.35 hours. This difference corresponds to less than 1.5\%, which already is a very small error of judgment.

However, Table 7 shows that the analytical model instantiated to Project \textit{Alpha} parameters pointed out an average of 13.69 working hours, considering all participants individual working hours. Consequently, using as basis 242 days (11 months with 22 workdays) of project execution, the total number of working hours calculated from the model probabilities is 3,312.98 hours. Note that the model provided an approximated value for the total working hours, which is even
closer to the actual number of hours spent in the execution phase. In fact, the predictions obtained from the SAN model deliver estimations with an impressive relative error of less than 0.2%.

Table 7: Project working hours obtained from the proposed model

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Expertise</th>
<th>Allocation (%)</th>
<th>State W (%)</th>
<th>Working hours per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Senior</td>
<td>100</td>
<td>87.09</td>
<td>6.97</td>
</tr>
<tr>
<td>3</td>
<td>Junior</td>
<td>75</td>
<td>14.70</td>
<td>2.65</td>
</tr>
<tr>
<td>1</td>
<td>Senior</td>
<td>20</td>
<td>87.09</td>
<td>1.39</td>
</tr>
<tr>
<td>2</td>
<td>Senior</td>
<td>10</td>
<td>87.09</td>
<td>1.39</td>
</tr>
<tr>
<td>1</td>
<td>Junior</td>
<td>3</td>
<td>14.70</td>
<td>0.04</td>
</tr>
<tr>
<td>3</td>
<td>Senior</td>
<td>3</td>
<td>87.09</td>
<td>0.63</td>
</tr>
<tr>
<td>1</td>
<td>Junior</td>
<td>5</td>
<td>14.70</td>
<td>0.06</td>
</tr>
<tr>
<td>1</td>
<td>Senior</td>
<td>5</td>
<td>87.09</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>Senior</td>
<td>3</td>
<td>87.09</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total</strong> 13.69</td>
</tr>
</tbody>
</table>

Analogously to the prediction of working hours, Table 8 presents the results from the analytical model regarding project impediments, using state C probabilities as statistical information about needed interactions to solve issues. The cooperation hours are summed in this table to indicate the time spent in solving project issues cooperating with the central team, considering an eight-hour workday. State C abstraction indicates that participants have found issues difficult to overcome by themselves and need effectively to cooperate with the central team managers to solve them.

Table 8: Project impediment hours obtained from the proposed model

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Expertise</th>
<th>Allocation (%)</th>
<th>State C (%)</th>
<th>Cooperating hours per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Senior</td>
<td>100</td>
<td>0.95</td>
<td>0.076</td>
</tr>
<tr>
<td>3</td>
<td>Junior</td>
<td>75</td>
<td>7.04</td>
<td>1.267</td>
</tr>
<tr>
<td>1</td>
<td>Senior</td>
<td>20</td>
<td>0.95</td>
<td>0.015</td>
</tr>
<tr>
<td>2</td>
<td>Senior</td>
<td>10</td>
<td>0.95</td>
<td>0.015</td>
</tr>
<tr>
<td>1</td>
<td>Junior</td>
<td>3</td>
<td>7.04</td>
<td>0.017</td>
</tr>
<tr>
<td>3</td>
<td>Senior</td>
<td>3</td>
<td>0.95</td>
<td>0.007</td>
</tr>
<tr>
<td>1</td>
<td>Junior</td>
<td>5</td>
<td>7.04</td>
<td>0.028</td>
</tr>
<tr>
<td>1</td>
<td>Senior</td>
<td>5</td>
<td>0.95</td>
<td>0.004</td>
</tr>
<tr>
<td>1</td>
<td>Senior</td>
<td>3</td>
<td>0.95</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total</strong> 1.43</td>
</tr>
</tbody>
</table>
The analytical model predictions for the collaboration hours to solve project impediments is on average 1.43 hours per day. Once again, considering 242 workdays as the project duration, the model results indicate a total of 346.06 hours of impediments. Comparing this prediction with the 332 impediments hours observed in Project Alpha (Table 5), a very small relative error of 4% is still found, which is even more impressive taking into account the amount of information that was abstracted while defining the model states and event rates.

4.3. Different scenarios analysis

Since the results obtained from the analytical model of Project Alpha were successful considering the low relative errors found in previous section, different scenarios were modeled in order to analyze the impact on varying availability hours and levels of support provided by the managers. The team configuration was not changed maintaining 14 participants and the central team, keeping the participants parameterized (events e and r) with original expertises (Table 3, Section 3.2). For the other events (a, u and s), we present in Table 9 different values for the model’s event rates, i.e., assuming different central team management behaviors. Specifically, we consider two levels of availability (High and Low) and four levels of quality of support (Lower, Low, High, and Higher). Taking into account these levels of availability and support, it is possible to define eight different scenarios according with the estimated event rates presented in the table.

Table 9: Estimated event rates for an eight-hour workday composing different scenarios

<table>
<thead>
<tr>
<th>Type</th>
<th>Event</th>
<th>Description</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>loc</td>
<td>a</td>
<td>High availability: Central team is available to collaborate with participants on average 7 hours per workday.</td>
<td>8/7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low availability: Central team is available to collaborate with participants on average 1 hour per workday.</td>
<td>8/1</td>
</tr>
<tr>
<td>loc</td>
<td>u</td>
<td>High availability: Central team is unavailable to cooperate with participants on average 1 hour per workday.</td>
<td>8/1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low availability: Central team is unavailable to cooperate with participants on average 7 hour per workday.</td>
<td>8/7</td>
</tr>
<tr>
<td>loc</td>
<td>e</td>
<td>Junior participants work on average 1 hour per workday without any kind of central team support.</td>
<td>8/1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Due to their expertise, senior participants work on average 7 hours per workday without any kind of central team support.</td>
<td>8/7</td>
</tr>
<tr>
<td>loc</td>
<td>r</td>
<td>Junior participants spend on average 7 hours per workday seeking solutions.</td>
<td>8/7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Senior participants spend on average only 1 hour per workday seeking solutions because of their expertise.</td>
<td>8/1</td>
</tr>
<tr>
<td>syn</td>
<td>co</td>
<td>Once a participant needs to collaborate with the central team, the collaboration occurs immediately if the central team is available.</td>
<td>disp</td>
</tr>
<tr>
<td>syn</td>
<td>s</td>
<td>Higher quality: Central team takes on average 0.5 hour per workday collaborating with participants.</td>
<td>8/0.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High quality: Central team takes on average 1 hour per workday collaborating with participants.</td>
<td>8/1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low quality: Central team takes on average 2 hours per workday collaborating with participants.</td>
<td>8/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lower quality: Central team takes on average 4 hours per workday collaborating with participants.</td>
<td>8/4</td>
</tr>
</tbody>
</table>
In Table 10, we present the configurations of scenarios used to predict Project ALPHA, varying levels of availability and support quality provided by the central team.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Level of availability (event a)</th>
<th>Level of quality of support (event s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High</td>
<td>Higher</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>3</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>Lower</td>
</tr>
<tr>
<td>5</td>
<td>Low</td>
<td>Higher</td>
</tr>
<tr>
<td>6</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>7</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>8</td>
<td>Low</td>
<td>Lower</td>
</tr>
</tbody>
</table>

Using those new configurations it is possible to calculate new time estimates for project execution phase on each scenario through the model solution. Table 11 presents the results for eight proposed scenarios from Table 10, showing the estimated working hours per day calculated for the whole team and, hence, the estimated project execution time.

<table>
<thead>
<tr>
<th>Actual project execution time</th>
<th>Scenario</th>
<th>Estimated working hours per day</th>
<th>Estimated project execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3,317.22 hours performed during 11.00 months</td>
<td>1</td>
<td>17.87 hours</td>
<td>8.44 months</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>15.40 hours</td>
<td>9.79 months</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>14.09 hours</td>
<td>10.70 months</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>13.41 hours</td>
<td>11.24 months</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>14.25 hours</td>
<td>10.58 months</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>13.85 hours</td>
<td>10.89 months</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>13.58 hours</td>
<td>11.10 months</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>13.37 hours</td>
<td>11.28 months</td>
</tr>
</tbody>
</table>

Observing the results presented in Table 11, scenario 8 (Low level of availability and Lower level of support provided by the central team) presented as expected the worst estimated time for the project execution phase, i.e., if the project was executed with this configuration it would take around 11.28 execution months. Actually, time estimates for project execution considering scenarios 5, 6, 7, and 8 (where the central team provides a Low level of availability) were close to actual execution time of Project ALPHA. There is a little gain of around 3.8% in scenario 5 (10.58 months to perform the project) in contrast to the actual execution time (11.00 months) of Project ALPHA.
Notice that the fact of increasing the central team availability level from Low to High does not necessarily improve the gain on project execution time. High levels of availability combined with Low or Lower level of support quality generates a small project execution time gain (about 2.7%) or even execution time loss (around 2.2%). This fact can be observed through results presented in Table 11 respectively for scenarios 3 and 4. On the other hand, High levels of availability combined with High levels of quality of support can provide significant gain on project execution time. For scenario 2, it was obtained a gain of 11% (i.e., the project could be performed during 9.79 months instead of the actual execution time of 11 months), and the best estimated time for executing the project was obtained from scenario 1, where the project could be executed during 8.44 months (i.e., a gain of approximately 23%).

The use of modeling formalisms to describe and analyze global software development projects is not a trivial effort. This section presented a modeling exercise, which has demonstrated to be useful on the context of software engineering for project planning and analysis. Our focus was on the analysis and prediction of project execution time. But, other project phases (e.g., monitoring and controlling) could be considered to measure the team’s performance in these phases.

5. Conclusion

The performance evaluation of software development teams using analytical models is an important tool in project management area. Moreover, these mathematical models allow the analysis and prediction of expected and non-expected behaviors of teams under complex situations, such as development projects with participants having different levels of expertise in multiple locations and multiple time zones.

An important contribution of this paper is to bring into a practical case scenario a theoretical modeling effort to describe a complex environment of global software development. Despite the numerous abstractions made in the modeling stage, the obtained numerical results demonstrated a very impressive accuracy when compared to actual project outcome (i.e., a relative error of less than 0.2%). Precisely, computing the steady-state probabilities of the instantiated model composed of 16 automata (i.e., more than 19 million states), we were able to predict a total number of 3,312.98 working hours to execute the project compared to 3,317.22 actual working hours spent in the execution phase. This fact by itself justifies our initial assumption that analytical modeling, in our particular case modeled using the SAN formalism, may be a worthy option to build teams in software development projects anticipating its overall performance.

We also propose in this paper different model instantiations (varying the availability and levels of support provided by the central team) to observe the impact of central team’s characteristics on the total time of the project execution phase. It was possible to notice that a central team that provides a high availability and high levels of support can reach considerable gains on the project execution time, i.e., a gain between 10% and 25%. However, a central team that provides a low availability, or even a high availability but low levels of support, does not impact significantly on the project execution time. For these cases, the increase of the predicted time to perform the project was inferior to two weeks.

This analysis can bring important contributions on team building process, where it can be economically interesting to compose a development team with a central team that has low availability, since the total project execution time has a little variation (less than a couple of weeks). On the contrary, if a participant interacts with a central team highly available, then the level of support provided by this central team must also be high in order to achieve project execution time gains.
Although this paper presents a successful modeling effort of a real case development project, it is important to keep in mind that a powerful stochastic modeling formalism was used, and an efficient solution tool is required to achieve the numerical predictions in a timely manner. Therefore, it is important to have either a performance evaluation expert, or a very suitable model to apply to different real case projects. This fact opens interesting future works about an automatic tool to facilitate the model instantiation in order to make this modeling approach much more easily applicable even for project managers that usually are not performance evaluation experts.

A possible future work is to extend the current theoretical model to encompass other different aspects that can be relevant to project outcome. For example, the results obtained in this paper suggest the extension of the model in order to represent explicit interactions between senior and junior participants, reducing the time spent for seeking issue solutions and increasing the working hours per day.

Another interesting future work is to thoroughly analyze model representations of a global software development project, varying the number of participants with different levels of experience and availability, where a project could be modeled by traditional and agile software development methodologies [38]. This numerical analysis may provide insightful conclusions not only to specific practical cases, but it also draws formal comparisons between software development methodologies.

Nevertheless, the proposed analytical model may be applied as it is to a large number of practical cases, and considering the accuracy achieved for the project ALPHA modeling, it is natural to expect good predictions about the outcome of software development teams.

References


Towards the evaluation of environment and business trade-offs in supply chains

Gabriel Alves Jr., Paulo Maciel, Ricardo Lima

Center of Informatics
Federal University of Pernambuco (UFPE)
Recife-PE, Brazil

Fábio Magnani

Department of Mechanical Engineering
Federal University of Pernambuco (UFPE)
Recife-PE, Brazil

Adilson Arcoverde Jr.

São Mateus Frigorífico
Recife-PE, Brazil

Abstract

Supply chains (SCs) are one of the most environment impacting systems. Analysis of such systems should thus take into account not only performance but also environment indicators. The amount of energy consumed for producing goods and the total emissions of greenhouse gases (GHG) of an activity are examples of such indicators. This paper presents a framework for assessing performance as well as Global Warming Potential (GWP) and exergy indicators in SCs. In order, exergy accounting helps on finding reliable GWP indicators for different energy sources adopted in the supply chain. This framework supports the evaluation of supply chains’ business and environment indicators trade-offs using a unified model. A real case study is conducted to demonstrate the application of the proposed modeling technique.

Keywords: Key Environmental Indicators, Life Cycle Assessment, Manufacturing Systems, Modeling, Performance Evaluation, Stochastic Models, Supply Chains

1 Email: gaaj@cin.ufpe.br
2 Email: prmm@cin.ufpe.br
3 Email: rmfl@cin.ufpe.br
4 Email: magnani@ufpe.br
5 Email: aoaj@saomateusfrigorifico.com

This paper is electronically published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs
1 Introduction

While economic and service level indicators were adequate to assess the performance of supply chains and manufacturing systems in the past, nowadays, environmental indicators are gradually becoming more relevant. Many prominent companies and academic research groups around the world are making efforts to provide environmentally responsible products and services. These topics are subjects of intensive study not only due to the respective impact of the production and transport systems in our planet but also particularly related to the image these companies aim to project to the society.

The Life Cycle Assessment (LCA) is a well known method for evaluating the environment impacts owing to the product existence [14]. Currently, there are some commercial tools used for LCA (e.g. SimaPRO). Within these tools, metrics like the Global Warming Potential (GWP) [14] are estimated based on a conversion database of resource consumption. Nevertheless, these tools are not well suited to conduct a performance evaluation of the activities involved in the product life cycle (e.g. machines utilization, reliability analysis), since it is not adressed by LCA.

The concept of exergy is linked to the Second Law of Thermodynamics (SLT) [5,16,27]. It assess the amount of energy that can be converted into useful work. Exergy analysis has been employed to measure and compare the use of different energy sources in systems and processes [13,27]. Some efforts have been made towards combining exergy and LCA in order to create a single sustainability metric [13,23]. The main difficulty to use an exergy based method is to capture the entire exergy flow for each resource used in the production of a good or service.

Modeling is quite often used to make quantitative and qualitative evaluation of systems [10,13,17,18,28]. Stochastic models have been widely used for evaluating supply chains and manufacturing systems [26]. These models are well suited for modeling systems where there is at least one variable that is assumed to follow a probability distribution. The strict mathematical modeling is often applied in such cases [8,22]. Although, queue networks, Markov chains, and Petri nets might also be adopted for stochastic modeling of these systems [10,24,26].

Stochastic Petri nets (SPN) [2,7,20] is a type of Petri net that deals with probabilistic distributed times. The use of SPNs to model systems might also require a deep knowledge of this technique. Model based performance evaluations might also require some tasks like the verification and validation of the models against the modeled system.

To tackle this problem, this work proposes the use of a library of SPN components to model supply chains and manufacturing systems. These components model specific entities or processes of the real system, focusing on the
product/information flows. This approach allows using SPNs as the modeling technique even without further knowledge on it. Moreover, the component-based approach tackles the requirement of verifying the model’s correctness. Although, a validation of the model might still be required. It happens, because the components guarantee that the structure of the systems will be correctly represented in the Petri net notation. But, it does not guarantee that the model’s parameters (e.g. mean time between failures and tasks delays) were assigned correctly in the model.

The graphical representation of SPNs permits to represent and estimate the impact of issues like buffers limits, failures, orders arrival rate and replenishment policies over operational, environmental, and cost metrics with relatively low costs. How this work addresses these topics will be discussed in following sections. Regarding sustainability, this work focuses on two environment indicators: exergy and Global Warming Potential (GWP). This work adopts a specific type of SPNs for modeling and evaluating models: the Stochastic Reward Nets (SRNs). We adopt the SRNs as modeling technique, in spite of other types of SPNs, since they allow the use of most of the SPNs features (e.g.: marking-dependent firing rates and arcs) and also embed rewards definitions within the SPNs [6, 7, 21]. This work contributes thus with a single model for assessing business and environmental indicators. Furthermore, the use of SRNs allows assessing supply chains’ sustainability indicators in probabilistic means. To the best of our knowledge, using stochastic Petri nets in such context is a novel approach.

2 Assessing Indicators with SRN

This section presents the proposed approach to assess environment impacting and business indicators using SRN models. In order to achieve this assessment, reward functions should be associated to transitions and places of a SRN. These functions are calculated for each state of the SRN model returning a result that represents the performance indicator.

Definition 2.1 presents a formal description for SRNs based on [7]. This definition groups the weight of immediate transitions and the rate of timed transitions into a single matrix, in spite of the original definition, where such elements are described in different matrices.

**Definition 2.1** [Stochastic reward nets] A SRN is a 10-tuple $N = (P, T, I, O, H, \Pi, G, M_0, W, \mathcal{R})$, where:

- $P$ is the ordered set of places;
- $T$ is the ordered set of transitions, $P \cap T = \emptyset$;
- $I \in (\mathbb{N}^{\lvert P \rvert} \rightarrow \mathbb{N})^{\lvert P \rvert \times \lvert T \rvert}$ is the matrix of marking-dependent multiplicities of input arcs. If place $p_j$ is an input place of transition $t_k$, then $i_{jk} \geq 1$ else
$i_{jk} = 0$;

- $O \in (\mathbb{N}^{|P|} \rightarrow \mathbb{N})^{|P| \times |T|}$ is the matrix of marking-dependent multiplicities of output arcs. If place $p_j$ is an output place of transition $t_k$, then $o_{jk} \geq 1$ else $o_{jk} = 0$;

- $H \in (\mathbb{N}^{|P|} \rightarrow \mathbb{N})^{|P| \times |T|}$ is the matrix of marking-dependent multiplicities of inhibition arcs. If place $p_j$ is an inhibition place of transition $t_k$, then $h_{jk} \geq 1$ else $h_{jk} = 0$;

- $\Pi \in \mathbb{N}^{|T|}$ is the vector of transitions’ priorities function. If transition $t_k$ is an immediate transition, then $\pi_k \geq 1$ else $\pi_k = 0$;

- $G \in (\mathbb{N}^{|P|} \rightarrow \{true, false\})^{|T|} \rightarrow \{true, false\}$ is the vector of marking-dependent transitions’ guards. If $t_k$ is enabled within $\mathbb{N}^{|P|}$, then $g_k = true$ else $g_k = false$;

- $M_0 \in \mathbb{N}^{|P|}$ is the vector of places’ initial markings, where $\mu_{0j} \geq 0$, $\forall p_j \in P$;

- $W \in (\mathbb{N}^{|P|} \rightarrow \mathbb{R}^+)^{|T|}$ is the vector of marking-dependent immediate transitions’ weights and timed transitions’ rates. For immediate transitions the $k-th$ element of $W$ is denoted by $\lambda_k$, representing its weight. Regarding timed transitions, $\lambda_k$ is the $k-th$ element of $W$ and depicts its rate, which in turn must be greater than zero;

- $R$ is a finite ordered set of rewards of $N$. Each element $\nabla_i \in R$ is a triplet $(\rho, r, \psi)$ representing the $i-th$ reward of the SRN, where: $\rho$ is a reward rate, $r$ is a reward impulse and $\psi$ is a reward based on the results of other rewards.

Since SRNs support marking-dependent timed transitions’ rates, these transitions can be defined as single-, k-, or infinite-server, in the same sense as queueing networks. Let $N$ be a SRN, where $p_j \in P$ is the only input place of a transition $t_k \in T$, with rate 0.5. The depicted server semantics are respectively represented by $\lambda_k = 0.5$, $\lambda_k = 0.5 \times min(m_j, L)$ and $\lambda_k = 0.5 \times m_j$, where $m_j$ is the marking of place $p_j$ in a given state and $L$ is the upper limit of the k-server semantics. Furthermore, the phase approximation technique [10] can be applied to represent poly-exponential distribution functions such as Erlang, hypo-exponential, and hyper-exponential distributions.

SRNs associate rewards with transition firing and place marking at the net level. The underlying SPN’s Markov chain is then transformed into a Markov reward model (MRM). An MRM associates rewards with each state of the Markov chain [29]. In MRM{s}, reward rates relate to the rate that the reward is accumulated while the system is in a state $s_i$. reward impulses determine the amount of a reward that is instantaneously accumulated when the system goes from a state $s_i$ to a state $s_j$. Such MRM rewards are respectively represented by $\rho$ and $r$ components of each SRN’s reward $\nabla_i \in R$.

Regarding $R$, a reward rate function $\rho_i$ of an SRN depends on its markings, and is defined as $\rho : \mathbb{N}^{|P|} \rightarrow \mathbb{R}$, where $P$ is the set of places of the SRN. Thus,
∀µ ∈ RS, ρ_i(µ) depicts the rate in which reward i is accumulated while the system is in marking µ, where RS is the reachability set [19]. The reward impulse function r_{i,t} refers to the amount of reward i accumulated when a transition t fires. Let P and T be the respective sets of places and transitions of a SRN, the reward impulse is a function r_{i,t}: \mathbb{N}^{|P|} → \mathbb{R}. Thus, ∀µ ∈ RS, r_{i,t}(µ) depicts the amount of reward i that is accumulated in marking µ when transition t fires. The reward functions can also be defined depending on the results other rewards. Let i represent the amount of CO_2 expelled in the system. It is possible to define a reward ψ_j that measures the probability for the amount of CO_2 being over the average amount, or the maximum amount of CO_2 expelled per unit of time. A detailed description of how these rewards are computed can be found in [7].

Before evaluation of a system, it is important to collect data to calculate the environmental indicators. After identifying the system’s components (e.g.: machines, entities, processes) that are going to be represented in the model, the modeler should gather information about:

- **Energy** - The amount of resources consumed for energetic means. It is important to define the energy source (e.g. electricity, biomass, gasoline, diesel);

- **Raw Materials** - The amount of resources used to produce a good or realize an activity. Raw materials should be categorized by type (e.g. water, wood, hazardous, non-hazardous) and its origin (e.g. first use, reuse, recycled);

- **Waste** - The amount of waste generated by system’s activities. This information should be structured by the type of the waste (e.g. wood, card, plastic) and by its destination (e.g. recycling, landfill, composting).

It is important to stress that a resource might be used as energy source, raw material or be a waste of an activity. For instance, wood might be a raw material in the production of a good, and some amount of this wood might be wasted. It can also be burned, providing energy for an activity.

The proposed classification aims at providing means to separately measure GWP and exergy outputs of each activity/process, without being over-detailed avoiding a complex and inefficient evaluation process. Furthermore, a different value of GWP or exergy efficiency can be assigned to the same substance depending on its classification. For instance, a block of wood has a different GWP value when used as raw material of a good, disposed for recycling, or disposed in landfill. We chose this categorization based on the conversion factors usually adopted in LCA [4,9,12], in order to provide detailed description of the GWP of consumed/disposed resources.

Let N be a SRN that models the evaluated system, I is its set with the classified energy, raw material, and waste items. For each element in the set of classified items (I) it should be defined a reward ∇_i ∈ \mathcal{R} related to its
consumption or disposal. For convenience, the set with these basic rewards is denoted $\mathcal{R}_I$, where $\mathcal{R}_I \subseteq \mathcal{R}$.

An important remark considering the rewards definition is that they do not distinguish between places of the SRN. Instead, reward rates are based on the state of the SRN. But, sometimes it is wanted to have an insight of a specific process or a set of processes of the modeled system. In such cases, the rewards should be defined for each place and transition of the SRN.

If such strategy is used, the total reward of a classified item should be derived from the sum of the rewards for each (or some) place and transition of the SRN. Let $N$, $P' \subseteq P$ and $T' \subseteq T$ be a SRN and its respective sets of places and transitions of $N$, for which it is intended to obtain the expected time-averaged reward of $\nabla_i \in (\mathcal{R} - \mathcal{R}_I)$. $\nabla_i$ is measured as depicted in Equation 1.

$$\nabla_i = \sum_{j=0}^{j=|\mathcal{R}'|} \nabla_j$$

where $\mathcal{R}_I' \subseteq \mathcal{R}_I$ is the set of rewards related to $\nabla_i$ that were defined for $p \in P'$ and $t \in T'$.

Assuming that the evaluated system produces physical goods (not virtual ones, as occurs with most informatics services) a mass balance analysis might be directly derived from the sum of all raw materials inputs and output goods (Equation 2).

$$\nabla_i = \frac{Q t y_{\text{good}}}{\sum_{j=0}^{j=|\mathcal{R}'|} \nabla_j}$$

where $\mathcal{R}' \subseteq \mathcal{R}$ is the set rewards that represents the input of raw materials (in kg/time) used in the production of the good and $Q t y_{\text{good}}$ is the amount of goods produced per unit of time (in kg/time). $Q t y_{\text{good}}$ could be obtained from the throughput of a SRN transition that represents the production of goods.

There are another three important rewards that should be defined in terms of each classified item. These rewards are: cost, global warming potential, and exergetic input/output. For each reward $\nabla_i \in \mathcal{R}_I$, a cost reward $\nabla_j \in (\mathcal{R} - \mathcal{R}_I)$ must be defined. The financial reward should assign a financial profit (positive signal) or cost (negative signal) related to the classified item.

This reward is defined as

$$\nabla_j = K + \beta \times \nabla_i$$

where $K$ is a constant and $\beta$ is the unitary profit/cost for the classified item. The total value is simply depicted by the sum of the financial rewards.

For each reward $\nabla_i \in \mathcal{R}_I$, a global warming potential reward $\nabla_j \in (\mathcal{R} - \mathcal{R}_I)$ can also be defined as

$$\nabla_j = g \times \nabla_i$$

where $g$ is the GWP for each unit of the classified item. The total GWP is thus simply depicted by the sum of the GWP rewards.
For each reward $\nabla_i \in \mathcal{R}_I$, that refers to energy consumption, an exergy input, output, and lost reward $\nabla_j, \nabla_k, \nabla_l \in (\mathcal{R} - \mathcal{R}_I)$ can be respectively defined as

\[
\nabla_j = x_{ch} \times \nabla_i \tag{5}
\]

\[
\nabla_k = \eta_{II} \times \nabla_j \tag{6}
\]

\[
\nabla_l = \nabla_k - \nabla_j \tag{7}
\]

where $\eta_{II}$ and $x_{ch}$ are the weighted-average exergetic efficiency and chemical exergy of the used energy. The total exergy is thus simply depicted by the sum of the exergy rewards.

For each type of energy source consumed, the estimated exergetic efficiency of fuel $f$ regarding activity/location $act$ represented by the SRN’s transition/place should be informed ($\eta_{II,act,f}$). This efficiency factor in conjunction with the already known fuel’s chemical exergy ($x_{ch,f}$) allows calculating the exergy output in the activity $X_{out,act}$. Based on the exergy output (Equation 8), it is possible to compare the adoption of different types of energy sources. This comparison is carried out by considering that the exergy output of each activity must be the same regardless of the energy source. The amount (in kg) of the energy source of the new energy source could be calculated using Equation 9. It is important to stress that changing the energy source would probably vary the exergetic efficiency $\eta_{II}$ in the activity.

\[
X_{out,act,i,f_1} = \eta_{II,act,i,f_1} \times x_{ch,f_1} \times Qty_{act,i,f_1} \tag{8}
\]

\[
X_{in,act,i,f_2} = \frac{X_{out,act,i}}{\eta_{II,act,i,f_2}} \times Qty_{act,i,f_2} = \frac{X_{out,act,i}}{x_{ch,f_2} \times \eta_{II,act,i,f_2}} \tag{9}
\]

3 Basic Models

This section presents some SRN models that were conceived to represent facilities and processes of a supply chain and manufacturing systems. The manufacturing systems models were based on [10]. These models were conceived with the aim of developing a library of reusable components that could be used to model systems in a bottom-up approach. Furthermore, the composition of these modules result in a final model that has some properties like boundedness, allowing either a steady state or transient evaluation [1].

Figure 1 presents the proposed components. Some of these components are different when being used to model a pull, push or reverse supply chain [3,11,25]. In a push or reverse flow, the consumer component is not explicitly modeled. Instead, it is represented by transition $ta$ of the flow model, which models the arrival of goods in the destination. The set of models used to represent entities of a push SCs are similar to that ones used in the context of reverse ones.
Fig. 1. SRN models for entities and flows of a GSC.
In the components presented in Figure 1, places named \( pxDual \) are the dual places of places named \( px \). These places were included in order to guarantee that the final model is structurally bounded [20], allowing a stationary analysis of it. Each producer model (Figure 1(a)) is a SRN defined as \( PRD_i = (P^{PRD_i}, T^{PRD_i}, O^{PRD_i}, H^{PRD_i}, \Pi^{PRD_i}, G^{PRD_i}, M_0^{PRD_i}, W^{PRD_i}, R^{PRD_i}), i = 1, 2, \ldots, j \). Place \( pst^{PRD_i} \) represents producer’s finished goods inventory. The initial marking of place \( pstDual^{PRD_i} \) depicts the producer’s maximal storage capacity of finished goods. The place \( pp^{PRD_i} \) depicts the producing orders. In the context of reverse supply chains, this model represents the consumer of the supply chain. This consumer becomes the “producer” of the reverse flow product.

Each consumer model (Figure 1(c)) is a SRN defined as \( ZN_i = (P^{ZN_i}, T^{ZN_i}, I^{ZN_i}, O^{ZN_i}, H^{ZN_i}, \Pi^{ZN_i}, G^{ZN_i}, M_0^{ZN_i}, W^{ZN_i}, R^{ZN_i}), i = 1, 2, \ldots, j \). The place \( po^{CSM_i} \) represents a recent order of the consumer. Place \( pa^{CSM_i} \) represents the orders that have not yet been delivered to the consumer. If the marking of \( paDual^{CSM_i} \) reaches zero in any reachable state, the consumer’s demand should be inhibited, what is not desired. Therefore, its initial marking \( (M^{CSM_i}) \) must be high enough to avoid this situation with a high probability.

The occurrence of transition \( td^{CSM_i} \) depicts the request of \( n \) items to a producer. When the amount requested from the producer equals the predetermined amount of \( c \) tons or items, the products are shipped. This amount \( c \) is often a quantity close to the complete load of the vehicle class allocated to the consumer. It is possible to set the rate of transition \( td^{CSM_i} \) with the time necessary to request the amount \( c \). This approach reduces the state space size without loss of expressiveness. The reader should bear in mind that arc weights \( k \) must equal \( c \) in the flow model.

Intermediaries have characteristics of consumers and factories. They act like consumers to the facilities that supply their demands, and like a factories to entities that requests their products. Explanations given for consumer and producer models are thus valid for intermediary models as well. Each intermediary model (Figure 1(b) and Figure 1(g)) is a SRN defined as \( INT_i = (P^{INT_i}, T^{INT_i}, I^{INT_i}, O^{INT_i}, H^{INT_i}, \Pi^{INT_i}, G^{INT_i}, M_0^{INT_i}, W^{INT_i}, R^{INT_i}), i = 1, 2, \ldots, j \). This model represents any intermediary of the logistics network, such as warehouses and wholesalers. Therefore, it is possible to have an intermediary model connected to another one, representing the supplying relationship between a distributor and a wholesaler, for example.

Within the intermediaries models, the occurrence of transition \( ta^{INT_i} \) represents arrival of \( k \) items for replenishing the inventory. Furthermore, the value of \( k \) must be equal to the shipped load per travel to the intermediary \( (c) \), represented in the flow component.

The flow model represents information flow from a customer to a supplier and goods flow from a supplier to a customer. Each flow model (Figure 1(d)
and Figure 1(e) is a SRN defined as $FLW_i = (P^{FLW_i}, T^{FLW_i}, I^{FLW_i}, O^{FLW_i}, H^{FLW_i}, \Pi^{FLW_i}, G^{FLW_i}, M_0^{FLW_i}, W^{FLW_i}, R^{FLW_i})$, $i = 1, 2, \ldots, j$. Places $pst^{FLW_i}$ and $pstDual^{FLW_i}$ have the same meaning as the equally named ones in the producer models. When composing models, these places will be merged with these corresponding ones. Place $po^{FLW_i}$ has the same meaning as in the customer model and will also be merged with its corresponding place. Place $ps^{FLW_i}$ depicts orders that have not been shipped to the consumer yet, due to a lack of vehicles or inventory (backorders).

Place $pt^{FLW_i}$ depicts the transportation vehicle used to serve the consumer. This place could be merged with the homonymous places of other flow models, in order to represent shared resources. Firing transition $ts^{FLW_i}$ models shipping of products to a consumer. When it fires, $c$ tokens are consumed from place $pst^{FLW_i}$, meaning the removal of $c$ items from the producer’s store. The arc weight $c$ cannot be higher than the maximal load capacity of the kind of vehicle used to send products to the consumer. Immediate transition $ts^{FLW_i}$ allows representing a priority and weight between consumers orders fulfillment.

Occurrence of transitions $to^{FLW_i}$, $tt0^{FLW_i}$, $ta^{FLW_i}$, and $tt1^{FLW_i}$, models order reception from a customer, traveling from producer to consumer, and delivering of goods to consumer and traveling back to producer, respectively. In a real situation, it is possible to place more than one order at the producer, or to have more than one vehicle traveling from/to a consumer at the same time. Therefore, the depicted transitions have infinite-server semantics (ISS).

Each manufacturer’s process model (Figure 1(h)) is a SRN defined as $PRC_i = (P^{PRC_i}, T^{PRC_i}, I^{PRC_i}, O^{PRC_i}, H^{PRC_i}, \Pi^{PRC_i}, G^{PRC_i}, M_0^{PRC_i}, W^{PRC_i}, R^{PRC_i})$, $i = 1, 2, \ldots, j$. Place $pM^{PRDi}$ represents a resource that is required to accomplish a task represented by transition $tp^{PRDi}$. This place can be merged other places $pM^{PRDi}$ in order to represent a shared resource. Transition $tp^{PRDi}$ must have a infinite-server semantics.

Process components might be connected to buffers or directly connected with other processes models. Depending on the level of abstraction adopted, this component might represent a single process, a machine operation, or even a whole production line of the manufacturer.

Each buffer model (Figure 1(i)) is a SRN defined as $BFR_i = (P^{BFR_i}, T^{BFR_i}, I^{BFR_i}, O^{BFR_i}, H^{BFR_i}, \Pi^{BFR_i}, G^{BFR_i}, M_0^{BFR_i}, W^{BFR_i}, R^{BFR_i})$, $i = 1, 2, \ldots, j$. The initial marking of place $pPB^{DualPRDi}$ represents the buffer’s limit, while markings in $pPB^{PRDi}$ denotes the used space of the buffer.

In the context of supply chains, faults occur quite often. Delivering failures, products, vehicles or machines breaks are examples of such faults that might temporarily halt an activity or impact its usual rate. Furthermore, depending on the fault/repair rate, the overall system’s performance might also be affected. The failure model (Figure 1(j)) is a SRN defined
as $FLTR_i = (P^{FLTR}_i, T^{FLTR}_i, I^{FLTR}_i, O^{FLTR}_i, H^{FLTR}_i, \Pi^{FLTR}_i, G^{FLTR}_i, M_0^{FLTR}_i, W^{FLTR}_i, R^{FLTR}_i), i = 1, 2, \ldots, j$. Transitions $tMTBF^{FLTR}_i$ and $tMTTR^{FLTR}_i$ respectively depict the mean time between failures and the mean time to repair. The initial marking of $pOk^{FLTR}_i$ denotes the maximum amount of resources that might be used in an activity that is susceptible to faults.

If $R > 1$, thus the rates of the timed transitions might depend on the marking of its input places (infinite server semantics). For instance the fault rate $2.5 \times \#pOk^{FLTR}_i$ denotes each of the resources available fails with a rate of 2.5. The repair rate might also depend on the marking of $pRepair^{FLTR}_i$. Furthermore, this rate might represent the usage of a limited maintenance team. For instance, the rate $0.5 \times \min(\#pRepair^{FLTR}_i, 3)$ associated with $tMTTR^{FLTR}_i$ denotes that once a resource fails, it is repaired with a rate of 0.5, but there is a limited amount of 3 resources in the maintenance team.

If the rate of $tMTTR^{FLTR}_i$ denotes the repair rate limit, the guard of $tRepair^{FLTR}_i$ allows representing the limited allocation of the maintenance team. It is useful when the model contains two or more $FLTR$ components. For instance, if there are two components $FLTR_1$ and $FLTR_2$, and the maintenance team is limited to 3 resources, the guard of $tRepair^{FLTR}_1$ and $tRepair^{FLTR}_2$ should be $\#pRepair^{FLTR}_1 + \#pRepair^{FLTR}_2 < 3$. Furthermore, it might also be adopted different repairing priorities for each failure, by changing the priority of these transitions.

This model might also represent the failures in one or more activities. The rate associated with $tMTBF^{FLTR}_i$ represents the failure rate when a set of activities are being executed, or the absolute time between failures. In the first case, it is necessary to assign to transition $tMTBF^{FLTR}_i$ a guard $[t_k, \forall t_k \in T']$, where $T'$ is the set of transitions that represents activities susceptible to the modeled fault.

The guards and rates of such transitions must also depend on the failure model. If a transition $t_k \in T'$ must have at least $n$ resources working to be fired, it must have a guard like $\#pOk^{FLTR}_i \geq n$, where $n$ is an integer. If $n = R$ it means that if a single resource is in the fail state, the activity represented by $t_k$ halts. Alternatively, the $FLTR$ can be reduced by removing transition $tRepair^{FLTR}_i$ and place $pRepair^{FLTR}_i$. It can be adopted when it is not necessary to represent the limited allocation of the maintenance team.

4 Case Study

This section presents a case study conducted in a Brazilian meat processing industry. This study considers a production line composed of different machines and sub-processes. These elements were grouped in stages of the production line. It was thus mapped three main stages which will be called $Stage 1$, $Stage$
2 and Stage 3.

This case study focuses on the following environment impacting aspects: energy consumption and waste generation. Beyond environment issues, we also model the failures at each stage. We address this issue to assess the impact of fails in the system performance. This impact might provide information for decisions on the maintenance of the production line’s machines.

Table 1 details the values for the resources used in the production line. We categorized wastes as depicted in Section 2. The alias column refers to an abbreviation used in metrics and graphics presented along this section. Column I/O shows that if the resource is used as input (consumption) or output (disposal) in the production stage. The electricity is used for powering machines, whilst the natural gas is used for cooking goods.

The system works as a pipeline, having each component sequentially connected to the next one. We collected the data history for each evaluated stage and removed the outliers. Such outliers were detected through the Interquartile range (IQR) analysis. Since data history presented a small number of outliers, it is possible to assure that such data are reliable.

Figure 2 shows the SRN model for the production line. As observed in such a model, the failures were also represented. It is thus possible to compare the effects of failures over performance and environmental metrics. Since for this kind of problem the failure rates tend to affect not only the availability but also the system performance, they could not be modeled in separate, for instance using reliability block diagrams.
Table 2 provides a summary of the exergetic values adopted for following calculations [16]. Such efficiencies are used in the exergy/GWP comparison. The natural gas and fuel oil efficiencies considered for powering machines represent the efficiency for converting the energy source into electricity, that in turn could be directly used by machines.

Table 2
Exergy efficiency per source and use.

<table>
<thead>
<tr>
<th>Source</th>
<th>Use</th>
<th>Efficiency ($\eta_{II}$)</th>
<th>$x_{ch,f}$ (kJ/kg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electricity</td>
<td>Power</td>
<td>0.92</td>
<td>3600</td>
</tr>
<tr>
<td>Electricity</td>
<td>Cooking</td>
<td>0.115</td>
<td>3600</td>
</tr>
<tr>
<td>Natural Gas</td>
<td>Power</td>
<td>0.2931</td>
<td>51702</td>
</tr>
<tr>
<td>Natural Gas</td>
<td>Cooking</td>
<td>0.233</td>
<td>51702</td>
</tr>
<tr>
<td>Fuel Oil</td>
<td>Power</td>
<td>0.3207</td>
<td>47101</td>
</tr>
<tr>
<td>Fuel Oil</td>
<td>Cooking</td>
<td>0.233</td>
<td>47101</td>
</tr>
</tbody>
</table>

Table 3 presents the reward functions adopting the SPNP tool syntax [15]. We used the SPNP tool to compute these rewards in the steady-state. Table 4 depicts the results of three experiments that were carried out. The first experiment, removes the failures from the model. The second one, includes failures but considers that there are no limitations for the maintenance team.
The third experiment, consider that there is only one resource available in the maintenance team.

### Table 3
Reward functions expressions.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Stage</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate1 (un./hour)</td>
<td>1</td>
<td>return rate(&quot;tp_PRC_0&quot;)/44.0;</td>
</tr>
<tr>
<td>rate2 (un./hour)</td>
<td>2</td>
<td>return rate(&quot;tp_PRC_1&quot;)/22.0;</td>
</tr>
<tr>
<td>rate3 (un./hour)</td>
<td>3</td>
<td>return rate(&quot;tp_PRC_2&quot;)/19.0;</td>
</tr>
<tr>
<td>Utilization1 (un./hour)</td>
<td>1</td>
<td>return enabled(&quot;tp_PRC_0&quot;)?mark(&quot;p_PRC_0&quot;)/44.0:0.0;</td>
</tr>
<tr>
<td>Utilization2 (un./hour)</td>
<td>2</td>
<td>return enabled(&quot;tp_PRC_1&quot;)?mark(&quot;p_PRC_1&quot;)/22.0:0.0;</td>
</tr>
<tr>
<td>Utilization3 (un./hour)</td>
<td>3</td>
<td>return enabled(&quot;tp_PRC_2&quot;)?mark(&quot;p_PRC_2&quot;)/19.0:0.0;</td>
</tr>
<tr>
<td>el1 (kWh/hour)</td>
<td>1</td>
<td>return (63.6812*rate1());</td>
</tr>
<tr>
<td>el2 (kWh/hour)</td>
<td>2</td>
<td>return (102.9402*rate2());</td>
</tr>
<tr>
<td>el3 (kWh/hour)</td>
<td>3</td>
<td>return (22.9600*rate3());</td>
</tr>
<tr>
<td>gas2 (m³/hour)</td>
<td>2</td>
<td>return (26.7559*rate2());</td>
</tr>
<tr>
<td>hr3 (un./hour)</td>
<td>3</td>
<td>return (6.52*rate3());</td>
</tr>
<tr>
<td>card1 (kg/hour)</td>
<td>1</td>
<td>return (3.7423*rate1());</td>
</tr>
<tr>
<td>org3 (kg/hour)</td>
<td>3</td>
<td>return (6.2870*rate3());</td>
</tr>
<tr>
<td>wood1 (kg/hour)</td>
<td>1</td>
<td>return (0.1516*rate1());</td>
</tr>
<tr>
<td>denspi_1 (kg/hour)</td>
<td>1</td>
<td>return (0.9167*rate1());</td>
</tr>
<tr>
<td>film_plst3 (kg/hour)</td>
<td>3</td>
<td>return (6.6881*rate3());</td>
</tr>
<tr>
<td>ferrous1 (kg/hour)</td>
<td>1</td>
<td>return (0.3441*rate1());</td>
</tr>
<tr>
<td>nferrous1 (kg/hour)</td>
<td>1</td>
<td>return (0.0355*rate1());</td>
</tr>
<tr>
<td>X_in_el1 (MJ/hour)</td>
<td>1</td>
<td>return (3.6*el1());</td>
</tr>
<tr>
<td>X_in_el2 (MJ/hour)</td>
<td>2</td>
<td>return (3.6*el2());</td>
</tr>
<tr>
<td>X_in_el3 (MJ/hour)</td>
<td>3</td>
<td>return (3.6*el3());</td>
</tr>
<tr>
<td>X_in_gas2 (MJ/hour)</td>
<td>2</td>
<td>return (51.7020.714*gas2());</td>
</tr>
<tr>
<td>X_out_power (MJ/hour)</td>
<td>system</td>
<td>return 0.92*(X_in_el1()+X_in_el2()+X_in_el3());</td>
</tr>
<tr>
<td>X_out_cooking (MJ/hour)</td>
<td>system</td>
<td>return 0.233*X_in_gas2();</td>
</tr>
<tr>
<td>repairing1 (un./hour)</td>
<td>1</td>
<td>return mark(&quot;p_Repair_FLTR_0&quot;);</td>
</tr>
<tr>
<td>repairing2 (un./hour)</td>
<td>2</td>
<td>return mark(&quot;p_Repair_FLTR_1&quot;);</td>
</tr>
<tr>
<td>repairing3 (un./hour)</td>
<td>3</td>
<td>return mark(&quot;p_Repair_FLTR_2&quot;);</td>
</tr>
<tr>
<td>waiting_repair1 (un./hour)</td>
<td>1</td>
<td>return mark(&quot;p_Fault_FLTR_0&quot;);</td>
</tr>
<tr>
<td>waiting_repair2 (un./hour)</td>
<td>2</td>
<td>return mark(&quot;p_Fault_FLTR_1&quot;);</td>
</tr>
<tr>
<td>waiting_repair3 (un./hour)</td>
<td>3</td>
<td>return mark(&quot;p_Fault_FLTR_2&quot;);</td>
</tr>
</tbody>
</table>

The results presented in Table 4 shows that the inclusion of failures reduces in almost 8% the production rate (from 4.13629 to 3.81551). This rate means that in 3.81551 units of time, a tonne of goods is produced. The lower utilization of the second stage sugests that it represents a bottleneck in the system. So, investments in this stage should be prioritized. The experiment that considers the limitation in the maintenance team presents results that are quite
Similar to those provided by the scenario without this limitation. Thus, considering the current failures and maintenance rates, a single maintenance team could meet the needs of this production line. But if such failures increase, new experiments could be conducted in order to check if this assumption remains true.

The following analysis are based on the second experiment that represents the actual situation of the production line. Assuming the current operation of the industry, it is possible to infer that this production line assigns a GWP of 147 kg CO$_2$e/ton of goods. We performed this estimation considering the

<table>
<thead>
<tr>
<th>Metric</th>
<th>Stage</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>rate1 (un./hour)</td>
<td>1</td>
<td>4.13629</td>
<td>3.81797</td>
<td>3.82243</td>
</tr>
<tr>
<td>rate2 (un./hour)</td>
<td>2</td>
<td>4.13629</td>
<td>3.81739</td>
<td>3.82111</td>
</tr>
<tr>
<td>rate3 (un./hour)</td>
<td>3</td>
<td>4.13629</td>
<td>3.81551</td>
<td>3.81879</td>
</tr>
<tr>
<td>Utilization1 (un./hour)</td>
<td>1</td>
<td>0.48134</td>
<td>0.44428</td>
<td>0.44479</td>
</tr>
<tr>
<td>Utilization2 (un./hour)</td>
<td>2</td>
<td>0.41707</td>
<td>0.38491</td>
<td>0.38526</td>
</tr>
<tr>
<td>Utilization3 (un./hour)</td>
<td>3</td>
<td>0.43849</td>
<td>0.40446</td>
<td>0.40494</td>
</tr>
<tr>
<td>el1 (kWh/hour)</td>
<td>1</td>
<td>263.40376</td>
<td>243.13300</td>
<td>243.41676</td>
</tr>
<tr>
<td>el2 (kWh/hour)</td>
<td>2</td>
<td>425.79028</td>
<td>392.96251</td>
<td>393.34586</td>
</tr>
<tr>
<td>el3 (kWh/hour)</td>
<td>3</td>
<td>94.96916</td>
<td>87.60421</td>
<td>87.67934</td>
</tr>
<tr>
<td>gas2 ($m^3$/hour)</td>
<td>2</td>
<td>110.67010</td>
<td>102.13761</td>
<td>102.23724</td>
</tr>
<tr>
<td>hr3 (un./hour)</td>
<td>3</td>
<td>26.96860</td>
<td>24.87715</td>
<td>24.89849</td>
</tr>
<tr>
<td>card1 (kg/hour)</td>
<td>1</td>
<td>15.47923</td>
<td>14.28799</td>
<td>14.30467</td>
</tr>
<tr>
<td>org3 (kg/hour)</td>
<td>3</td>
<td>26.00484</td>
<td>23.98814</td>
<td>24.00871</td>
</tr>
<tr>
<td>wood1 (kg/hour)</td>
<td>1</td>
<td>0.62706</td>
<td>0.57880</td>
<td>0.59486</td>
</tr>
<tr>
<td>dense-plst1 (kg/hour)</td>
<td>1</td>
<td>3.79173</td>
<td>3.49993</td>
<td>3.50402</td>
</tr>
<tr>
<td>film-plst3 (kg/hour)</td>
<td>3</td>
<td>27.66391</td>
<td>25.51854</td>
<td>25.54043</td>
</tr>
<tr>
<td>ferrous1 (kg/hour)</td>
<td>1</td>
<td>1.42330</td>
<td>1.31376</td>
<td>1.31530</td>
</tr>
<tr>
<td>ferrous2 (kg/hour)</td>
<td>1</td>
<td>0.14684</td>
<td>0.13554</td>
<td>0.13570</td>
</tr>
<tr>
<td>X$_{\text{in}}$el1 (MJ/hour)</td>
<td>1</td>
<td>948.25354</td>
<td>875.27880</td>
<td>876.30034</td>
</tr>
<tr>
<td>X$_{\text{in}}$el2 (MJ/hour)</td>
<td>2</td>
<td>1532.84500</td>
<td>1414.66503</td>
<td>1416.04509</td>
</tr>
<tr>
<td>X$_{\text{in}}$el3 (MJ/hour)</td>
<td>3</td>
<td>341.88899</td>
<td>315.37515</td>
<td>315.64561</td>
</tr>
<tr>
<td>X$_{\text{in}}$gas2 (MJ/hour)</td>
<td>2</td>
<td>4985.41194</td>
<td>3770.43302</td>
<td>3774.11120</td>
</tr>
<tr>
<td>X$_{\text{in}}$power (MJ/hour)</td>
<td>1</td>
<td>2597.14854</td>
<td>2396.89347</td>
<td>2399.35176</td>
</tr>
<tr>
<td>X$_{\text{in}}$cooking (MJ/hour)</td>
<td>1</td>
<td>951.90098</td>
<td>878.51089</td>
<td>879.36791</td>
</tr>
<tr>
<td>repairing1 (un./hour)</td>
<td>1</td>
<td>-</td>
<td>0.00580</td>
<td>0.00573</td>
</tr>
<tr>
<td>repairing2 (un./hour)</td>
<td>2</td>
<td>-</td>
<td>0.03292</td>
<td>0.03437</td>
</tr>
<tr>
<td>repairing3 (un./hour)</td>
<td>3</td>
<td>-</td>
<td>0.04039</td>
<td>0.03794</td>
</tr>
<tr>
<td>waiting-repair1 (un./hour)</td>
<td>1</td>
<td>-</td>
<td>0.00000</td>
<td>0.00006</td>
</tr>
<tr>
<td>waiting-repair2 (un./hour)</td>
<td>2</td>
<td>-</td>
<td>0.00000</td>
<td>0.00007</td>
</tr>
<tr>
<td>waiting-repair3 (un./hour)</td>
<td>3</td>
<td>-</td>
<td>0.00000</td>
<td>0.00013</td>
</tr>
</tbody>
</table>
conversion factors provided by DEFRA [9]. Figure 3 presents the GWP participation separated for the energy sources and disposed resources. The energy sources are responsible for more than 95% of the overall GWP. It is important to spot that the electricity conversion factor might vary from country to country. This case study adopted the UK factors provided by DEFRA. Taking into consideration the electricity participation in the total GWP, if the Brazilian’s conversion factor (which is lower than in UK), the GWP resultant from the production line should considerably decrease.

Fig. 3. Participation of resources in the total GWP.

We calculated the amount of exergy input necessary to generate the same exergy output (see Table 4) with a single energy source. Based on that exergy input, we calculate the GWP and compared it to the actual operation of

Fig. 4. Impact of energy source over GWP.
the production line. Figure 4 presents that comparison result. The graphs labeled as “ideal efficiency” assume a hypothetical situation where the current efficiency $\eta_{II,a,f}$ is preserved. The “real efficiency” graphs, depict the variation in a real scenario where the exergetic efficiency changes according to the energy source.

It is possible to observe that considering an hypothetical situation where the exergy efficiency is preserved, the use of natural gas as the single energy source decreases the GWP in European countries, whilst in Brazil, this value increases. It occurs due to the fact that in Brazil, the GWP factor of the energy is very low when compared to other countries, due to the extensive use of hydropower energy.

Regarding the real efficiencies, despite of the fact that the exergetic efficiency of the electricity for cooking processes is lower than that one of the natural gas, the GWP variation remains almost constant when the electricity is used as the only energy source in Brazil. Furthermore, although the fuel oils have a high chemical exergy, their high GWP concentration make them be the worst alternative from the environment issue. Analysis of costs might justify their usage in some points of the production line in detriment to environment impacts.

5 Concluding Remarks

This paper presented the evaluation of GWP and exergetic indicators in manufacturing systems and supply chains using stochastic models. It presents a comparison of exergetic values for different energy sources and the corresponding GWP resultant from the use of such sources. It was observed the importance of considering not only the energy source, but also the localities, that means, the effects of the system location (e.g. country, city, etc) over evaluated metrics. Especially for the electricity, the GWP factor might vary substantially according to the country that is using such issue. Since resources are detailed, its costs could be directly assessed. In conjunction with the analysis of costs, this kind of comparison might support the cost/environment trade-off analysis.

The proposed approach uses a single model to measure environmental and performance indicators. Using stochastic Petri nets to measure such indicators allows the calculation of measurements like the probability of having an indicator over a limit amount. Furthermore, this modeling technique allowed the definition of high-level components that could not be defined using other techniques like Markov chains. The library of components could thus be used to model a whole system using a bottom-up approach.
References


The Monte Carlo EM method for the parameter estimation of biological models

Alessio Angius

Department of Computer Science, University of Torino, Torino, Italy

András Horváth

Department of Computer Science, University of Torino, Torino, Italy

Abstract

It is often the case in modeling biological phenomena that the structure and the effect of the involved interactions are known but the rates of the interactions are neither known nor can easily be determined by experiments. This paper deals with the estimation of the rate parameters of reaction networks in a general and abstract context. In particular, we consider the case in which the phenomenon under study is stochastic and a continuous-time Markov chain (CTMC) is appropriate for its modeling. Further, we assume that the evolution of the system under study cannot be observed continuously but only at discrete sampling points between which a large amount of reactions can occur.

The parameter estimation of stochastic reaction networks is often performed by applying the principle of maximum likelihood. In this paper we describe how the Expectation-Maximisation (EM) method, which is a technique for maximum likelihood estimation in case of incomplete data, can be adopted to estimate kinetic rates of reaction networks. In particular, because of the huge state space of the underlying CTMC, it is convenient to use such a variant of the EM approach, namely the Monte Carlo EM (MCEM) method, which makes use of simulation for the analysis of the model. We show that in case of mass action kinetics the application of the MCEM method results in an efficient and surprisingly simple estimation procedure. We provide examples to illustrate the characteristics of the approach and show that it is applicable in case of systems of reactions involving several species.

Keywords: parameter estimation, mass action kinetics, maximum likelihood, expectation-maximisation method

1 Email: angius@di.unito.it
2 Email: horvath@di.unito.it

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

43
1 Background

As described by Gillespie in [5] the temporal behaviour of a biochemical system can be described by a stochastic process, in particular, by a continuous time Markov chain (CTMC). In order to have a complete description of the CTMC model describing the phenomenon under study and to be able to perform its analysis, the estimation of the kinetic rates is a "conditio sine qua non". In this context, the parameter estimation is essentially an optimisation problem which aims to find the set of parameter such that the model is able to reproduce the experimental observations with high probability. The problem is not trivial for several reasons. The studied phenomenon can be very complex with several reagents interacting through many reactions. Moreover, it is often unrealistic to consider the process as perfectly and continuously observable. In particular, the measurement techniques are often unable to observe the system behaviour as a continuous process and provide observations of the system state only at a limited set of time instants. Moreover, consecutive time instants can be so far from each other that a considerable amount of reactions occur between them. This means that we have to face an optimisation problem with incomplete data in hand. A method to maximum likelihood estimation in case of incomplete data, namely the Expectation-Maximisation (EM) method, has been presented by A. Dempster in [3]. The basic idea of EM method is to rebuild the missing data in expectation and apply optimisation to find parameters that maximises the probability of the reconstructed complete data. It is often the case that the exact reconstruction of the missing data is a hard task. In these cases, as proposed by Wei and Tanner in [15], simulation can be used to complete the data and this approach is called the Monte Carlo EM (MCEM) method.

In this work we adopt the MCEM method to the estimation of kinetic rates in stochastic reaction networks. In particular, we consider stochastic reaction networks evolving according to mass action kinetics and show that the MCEM method leads to a simple and efficient estimation procedure.

Several works exist on estimating kinetic rates of reaction networks by applying optimisation methods [10,4,12]. Most of these works however do not consider stochasticity but apply a deterministic view of the evolution of the phenomenon under study. In theory, it is possible to transform the rates obtained for the deterministic model into rates that can be used in a stochastic setting but, as it is pointed out in [11], this is not always possible. This observation led to attempts to give an estimate of the kinetic rates in accordance with the stochastic view introduced by [5]. Bayesian inference methods were used in [7,1], maximum likelihood methods were applied in [11,2,13]. The strength of our approach, with respect to the ones cited above, is that it works with limited information (i.e., it is possible to apply it with
very infrequent observations between which thousands of reactions occur) and
does not involve heavy optimisation tasks.

The paper is organised as follows. In the next section we provide the
reference stochastic model. Then a brief introduction of the EM method in
general is given. Subsequently, we describe the application of the MCEM
method to the estimation of parameters of stochastic reaction networks. The
last but one section is dedicated to the numerical illustration of the proposed
approach. In the last section we draw the conclusions.

2 Considered model

We consider a continuous time Markov chain (CTMC) describing the interac-
tion of $M$ reagents through $R$ reactions. The state of the system is given by
a vector of $M$ integers providing the quantity of the reagents. The effect of
reaction $i$, $1 \leq i \leq R$ is described by a vector of $M$ integers denoted by $\mathbf{e}_i$ in
such a way that if reaction $i$ occurs in state $\mathbf{x}$ then the next state is $\mathbf{x}' = \mathbf{x} + \mathbf{e}_i$.
The transitions of the CTMC represent the reactions and their intensity de-
pends on the state of the system. By $f_i(\mathbf{x})$ we denote the function providing
the intensity of reaction $i$ in state $\mathbf{x}$. Throughout the paper we assume that
the intensities are of the form

$$f_i(\mathbf{x}) = k_i \prod_{j=1}^{M} \left( \frac{x_j}{a_{i,j}} \right)$$

where the constants $a_{i,j}$ provide the stoichiometry of the $i$th reaction with
$a_{i,j} \in \mathbb{N}$ and $k_i$ is the kinetic rate constant (i.e., it describes the speed of the
interaction). Consequently, the model corresponds to mass action kinetics and
the CTMC is exactly the one simulated by the classical algorithm of Gillespie
[5].

As an example of the model consider the following system of reactions [6]
which will be used among the numerical examples as well:

$$DNA_{off} \xrightarrow{k_1} DNA_{on},$$
$$DNA_{on} \xrightarrow{k_2} DNA_{off},$$
$$DNA_{on} \xrightarrow{k_3} mRNA + DNA_{on}.$$

The above set of reactions describes that $DNA$ is switched on/off by poly-
merase binding/unbinding and polymerase bound (i.e., switched-on) $DNA$ is
transcribed into $mRNA$. Switch on is described by the first reaction trans-
forming $DNA_{off}$ to $DNA_{on}$, switch off is described by the second reaction
transforming $DNA_{off}$ to $DNA_{on}$ and transcription is due to the third reac-
tion which produces $mRNA$ leaving the actual quantity of $DNA_{on}$ unchanged.
As three reagents are involved, the state of the system is a triple $\mathbf{x} = [x_1, x_2, x_3]$ describing the quantities of $DNA_{off}$, $DNA_{on}$ and $mRNA$, respectively. The vectors describing the effect of the three reactions are: $\mathbf{v}_1 = [-1, 1, 0]$, $\mathbf{v}_2 = [1, -1, 0]$ and $\mathbf{v}_3 = [0, 0, 1]$. The intensities associated with the reactions assuming mass action kinetics are: $f_1(\mathbf{x}) = k_1x_1$, $f_2(\mathbf{x}) = k_2x_2$ and $f_3(\mathbf{x}) = k_3x_2$. Based on the above description, the stochastic simulation of the system is a straightforward task.

3 EM method

The Expectation-Maximisation (EM) method is an algorithm for maximum likelihood parameter estimation in case of incomplete data. The input of the EM algorithm is composed by a set of samples and a stochastic model characterised by a set of parameters denoted by $\lambda$.

The EM method is iterative, i.e., it starts from an initial guess of the set of parameters, $\lambda_0$, and then improves it step by step in such a way that the behaviour provided by the model is more and more similar to the behaviour described by the samples. The set of parameters after $i$ steps is denoted by $\lambda_i$. Each iteration is composed of two steps called, respectively, Expectation step (E-step) and Maximisation step (M-step).

The role of the E-step is to compute the missing information in expectation. Formally, denoting by $Y$ the set of incomplete data, and by $Z$ the complete data, the E-step computes the conditional expectation $E[Z|Y, \lambda_i]$. In the context of our problem, $Y$ contains the samples at discrete time points and the E-step aims to calculate, given the current set of parameters in $\lambda_i$, the most typical full trajectory, $Z$, that goes through the observed samples given in $Y$.

The M-step is applied then to find a new set of parameters $\lambda_{i+1}$ such that the likelihood of the trajectory generated during the E-step is maximal. Once the new set of parameters $\lambda_{i+1}$ is found, it is used as the starting point for the next iteration.

In many situations, including the one considered in this paper, the strength of the EM method lies in the fact that finding such parameters that maximise the likelihood of the incomplete data is much harder than finding parameters that maximise the likelihood of the complete data. In other words, the optimisation required in the M-step is less burdensome than the original optimisation problem. In particular, in relation to the problem considered in this paper, the M-step is very simple in the case of mass action kinetics. In turn, in many cases, including ours, the E-step can be hard both from a theoretical and computational point of view.

For those cases in which the computations required by the E-step are particularly complex, a variant of the EM method can be applied. In this
variant, the exact computation of the conditional expectation in the E-step is substituted by simulation. This approach is known as the Monte Carlo EM (MCEM) method and it is particularly useful in situations when performing the E-step in an exact manner is either too time consuming or even unfeasible. For the problem considered in this paper, because of the huge state space of the involved CTMC, the only viable approach is provided by the MCEM method.

The convergence characteristics of the MCEM method are poorer than those of the EM method and the simulation can introduce fluctuations of the parameters. It is still possible however to prove that the convergence of the method is preserved if the number of iterations is high [15].

4 MCEM for biochemical systems

Problem formulation

We assume that we are given a network of reactions and experimental observations of quantities of the involved species at discrete time instants. The time instants of the observations will be denoted by \( t_0 = 0, t_1, t_2, \ldots, t_N \) and the associated observations by \( \mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_N \) where \( \mathbf{y}_i \) is a vector of integers providing the state of the system at \( t_i \). (We consider here only a single sequence of observations but the extension to multiple observation sequences is straightforward.) We assume that all or some of the kinetic rates are not known, i.e., there are unknown constants in the functions \( f_i(\mathbf{\pi}) \) which provide the intensity of the reactions. The set of these unknown constants will be denoted by \( \lambda \) and we will write \( f_i(\mathbf{\pi}, \lambda) \) to make explicit the dependence of the intensities on the unknowns. Our aim is to give a maximum likelihood estimate for the unknown kinetic rates by the MCEM method. In the following two subsections we describe the E-step and the M-step.

E-step

Given the set of samples and the current estimate of the parameters, the E-step aims to build the most probable full trajectory that goes through the observed states. This requires to find the most probable trajectory between each two consecutive sample points. The E-step hence requires to find most likely random walks over CTMCs. This is possible in theory but, unfortunately, as in our context the considered CTMC almost always has a huge state space, it cannot be performed in an exact manner. As anticipated, in this situation the E-step can be solved by simulation which provides a good approximation of the most probable trajectory. Generating trajectories of a CTMC by simulation is straightforward.

As we are given \( N + 1 \) samples, we need \( N \) subtraces to connect the observation time instants. As the CTMC is huge and the current set of parameters
can be far from the real set of parameters, it is very unlikely that a single simulation run arrives exactly (or even close) to the observed states. For this reason the E-step is composed by the following two phases.

(i) **Generation of traces.** For each interval \([t_i, t_{i+1}], 0 \leq i \leq N - 1\), we generate \(K\) random walks of length \(t_{i+1} - t_i\) starting from \(\bar{y}_i\) and choose the one that arrives closest in “distance” to \(\bar{y}_{i+1}\). The concept of “distance” between the sample point and the last state of the random walk is expressed as the sum of the relative errors over the species.

(ii) **Improvement of traces.** In this phase we improve the subtraces by modifying them. We pick up randomly a reaction from the subtrace and check if it can be substituted by another reaction in such way that the subtrace arrives closer to the observed state. The substitution is accepted only if all the remaining reactions are still possible. The times between consecutive reactions remain unchanged. The extent of the modification is determined by a parameter \(\rho \in [0, 1]\) which defines the proportion of the reactions that we attempt to substitute.

The proportion defined by \(\rho\) has a delicate role in the estimation process. When the current estimate is far from the real parameters, a higher \(\rho\) is necessary in order to come up with reasonable subtraces and to have faster convergence of the estimation process. Instead, when the estimates are already good, a lower \(\rho\) (even \(\rho = 0\)) has to be used in order to not to alter too much the stochastic behaviour induced by the actual estimate.

In terms of complexity, the cost of the generation of random walks is linear in the number of reactions occurrences, and the storage of the best random walk is very cheap, since each subtrace can be uniquely identified by means of the seed of the pseudo-random number generator. For these reasons, the improve of the best trace represents the most expensive phase of the method, since each trial forces the unroll and the check of the enabling of all subsequent reactions\(^3\). As a consequence, the implementation of this phase could be not trivial. A first hint could be the use of a preprocessing which after the random selection of the “candidates” for the substitution sorts them in order of occurrence. In this manner the whole trace can be unrolled just ones and for each attempt the number of checks becomes smaller and smaller. The second hint is to verify more than one reaction at a time and discard (without additional costs) all substitutions if the control is not satisfied. This solution could discard some valid substitutions but is more convenient in terms of computation time.

\(^3\) Note that the unroll of the trace is expensive almost as its generation.
**M-Step**

In the M-step we have to find the next set of estimates, \( \lambda_{i+1} \), that maximises the likelihood of the full trajectory generated in the E-step. The \( i \)-th subtrace generated by the E-step, reconstructing the most probable trajectory between state \( \bar{y}_{i-1} \) and state \( \bar{y}_i \) in the time interval \([t_{i-1}, t_i]\), will be denoted by \( S_i \) and it has the form

\[
\bar{y}_{i-1} = \mathcal{S}_{i,1} \xrightarrow{r_{i,1},u_{i,1}} \mathcal{S}_{i,2} \xrightarrow{r_{i,2},u_{i,2}} \cdots \xrightarrow{r_{i,H_i},u_{i,H_i}} \mathcal{S}_{i,H_i} \mathcal{S}_{i,H_i}
\]

where \( H_i \) denotes the length of the \( i \)-th subtrace and \( s_{i,j}, r_{i,j} \) and \( u_{i,j} \) are the states, the reactions and the sojourn times of the \( i \)-th subtrace, respectively.

We have that

\[
\sum_{j=1}^{H_i} u_{i,j} = t_i - t_{i-1}.
\]

The last arrow is without a reaction and this represents the fact that the process remains in state \( \mathcal{S}_{i,H_i} \) for at least \( u_{i,H_i} \) time units. The closer state \( \mathcal{S}_{i,H_i} \) is to state \( \bar{y}_i \), the better the \( i \)-th subtrace reflects the observed behaviour.

It follows from the theory of CTMCs that the likelihood of the \( i \)-th subtrace, denoted by \( L_i \), can be calculated as

\[
L_i = \left( \prod_{j=1}^{H_i-1} f_{r_{i,j}}(s_{i,j}, \lambda) e^{-f(s_{i,j}, \lambda) u_{i,j}} \right) \cdot e^{-f(s_i, \lambda) u_{i,H_i}} \quad (2)
\]

where

\[
f(\mathcal{S}, \lambda) = \sum_{k=1}^{R} f_k(\mathcal{S}, \lambda)
\]

is the sum of the intensities of the reactions in state \( \mathcal{S} \). The product in (2) gives the likelihood of the transitions of the \( i \)-th subtrace while the last exponential term is the probability that the process does not leave state \( \mathcal{S}_{i,H_i} \) for at least \( u_{i,H_i} \) time units. The likelihood of all the subtraces is simply given by

\[
L = \prod_{i=1}^{N} L_i
\]

and we have to find such \( \lambda \) that maximises this product.

In order to find the maximum of \( L \), it is useful to take its logarithm in which products are transformed into sums as

\[
\ln(L) = \sum_{i=1}^{N} \ln(L_i) = \sum_{i=1}^{N} \left( \sum_{j=1}^{H_i-1} \ln(f_{r_{i,j}}(s_{i,j}, \lambda)) - \sum_{j=1}^{H_i} f(s_{i,j}, \lambda) u_{i,j} \right). \quad (3)
\]
Naturally, the difficulty of finding the maximum of (3) depends on the functions \( f_i(\bar{x}, \lambda), 1 \leq i \leq R \). As mentioned earlier, we consider the case in which the intensity of the reactions corresponds to mass action kinetics (1). Moreover, we assume that the stoichiometry of the reactions (described by \( a_{i,j}, 1 \leq i \leq R, 1 \leq j \leq M \)) is known which is the typical case in parameter estimation problems. Accordingly, the parameters to estimate are the kinetic rate constants, i.e., \( \lambda = \{k_1, \ldots, k_R\} \). Without loss of generality, we focus our attention on finding such \( k_1 \) that maximises (3). Applying (1), the derivative of (3) with respect to \( k_1 \) is

\[
\frac{\partial \ln(L)}{\partial k_1} = \frac{N}{\sum_{i=1}^N \left( \sum_{j=1}^{H_i} I\{r_{i,j} \text{ is reaction 1}\} \right)} - \sum_{j=1}^{H_i} \sum_{k=1}^M \prod_{k=1}^M \left( \frac{x_k}{a_{1,k}} \right) u_{i,j}
\]

where \( I \) is 1 if its argument is true and 0 otherwise. Denoting by \( f_{i,j} \) the number of times reaction \( j \) occurs in the \( i \)th subtrace we have

\[
\frac{\partial \ln(L)}{\partial k_1} = \sum_{i=1}^N \left( \frac{f_{i,1}}{k_1} - \sum_{j=1}^{H_i} \prod_{k=1}^M \left( \frac{x_k}{a_{1,k}} \right) u_{i,j} \right).
\]

(4)

It is easy to check that the value of \( k_1 \) with which (4) is 0 maximises \( L \). Consequently, the estimate is

\[
k_1 = \frac{\sum_{i=1}^N f_{i,1}}{\sum_{i=1}^N \sum_{j=1}^{H_i} \prod_{k=1}^M \left( \frac{x_k}{a_{1,k}} \right) u_{i,j}}.
\]

(5)

Accordingly, in case of mass action kinetics, the optimisation required in the M-step boils down to the explicit formula given in (5).

Handling other forms of \( f_i(\bar{x}, \lambda) \) is out of the scope of this paper. We only mention here that with general \( f_i(\bar{x}, \lambda) \) functions the optimisation required by the M-step can become more complex but even in this case it is possible to divide the optimisation problem into smaller subproblems. In order to show this, let us denote by \( \nu_i \) the variables on which \( f_i(\bar{x}, \lambda) \) depends. It is reasonable to assume that the sets \( \nu_i, 1 \leq i \leq R \), are mutually disjoint. In this case the derivative of the log-likelihood function (given in (3)) according to a variable belonging to \( \nu_i \) does not depend on the variables belonging to the other sets \( \nu_j, j \neq i \). This means that the original optimisation problem that involves all the variables can be tackled by \( R \) smaller optimisation problems of much smaller dimensions.
5 Illustrative numerical examples

In this section we show numerical results obtained using the MCEM method. We apply the method to two models. For both cases, the samples are generated “in silico” by means of simulations. In these tests we put aside the biological meaning of the models and our aim is to illustrate the method and to show that it is able to reconstruct the set of parameters. For all the cases we provide tables to compare the original values and their estimates. Moreover, in order to provide a visual comparison of the behaviours with the original and the estimated values, we provide in figures the evolution of the systems according both to the corresponding ordinary differential equations (ODEs) and to the corresponding stochastic setting. The ODEs are useful to get a quick glimpse of the goodness of the estimates that were obtained in the stochastic setting.

The MCEM method has been implemented in a prototype JAVA tool. All the experiments have been performed on an Intel Centrino Dual Core with 4Gb of RAM.

5.1 Gene transcription model

Our first example is the model already introduced in Section 2 describing binding and unbinding of the DNA and its transcription into mRNA [6]. We assume to have a single unit of DNA and the initial condition is $[DNA_{off}] = 1$ and $[DNA_{on}] = [mRNA] = 0$.

In order to evaluate the method in different situations, we use the model with different levels of “granularity”, i.e., we use different levels of discretizations to obtain discrete models from the originally continuous concentrations. The discretization step will be denoted by $h$. The initial state of the CTMC modeling the three reactions is $[1/h, 0, 0]$. For the sake of having models that evolve on the same time scale independently of $h$, the kinetic rate constants have to depend on $h$. Specifically, the intensity of the reactions are $k_1h$, $k_2h$ and $k_3h$ with $k_1 = 0.027$, $k_2 = 0.166$ and $k_3 = 0.4$. The effect of $h$ is twofold: the smaller $h$ the larger the state space and the less variable the behaviour of the model. Indeed, as $h$ tends to 0, the behaviour of the model tends to the solution of the corresponding ODEs [8,14].

We have generated “in silico” samples in such a way that between consecutive sampling points there are about 25000 reactions. The number of the samples is 25. This means that only a small fraction of what happens in the model is available to the estimation procedure.

The E-step was performed with $K = 20$, i.e., 20 traces were generated in the first phase of the E-step. In the second phase the parameter $\rho$ plays a crucial role. It is convenient to start with a high value of $\rho$ and then to lower it gradually as the estimates become more reliable. We chose to start with $\rho = 0.4$ and to decrease it as the generated traces get closer to the
observations.

Table 1 reports the original parameters and those obtained by the MCEM approach after 100 iterations computed in about a second of CPU time. The initial guess of the parameters was random in the range [0:10]. A possible observation about the results could be that some estimations of \(k_1\) and \(k_2\) are far from the original values. This is caused by the fact that the low number of infrequent discrete samples reflect the ratio \(k_1/(k_1+k_2)\times k_3\) (determining the increase of mRNA) and, to some extent, the ratio \(k_1/(k_1+k_2)\) (determining the quantity of DNA) but not the value of parameters. More samples with higher sampling frequency could however alleviate this problem. It can also be observed that with finer discretization the results are more accurate. A way of illustrating all the cases on the same figure is to apply the results in the ODEs representing the model. This is depicted in Figure 1. It can be seen that all cases catch well the asymptotic increase rate of mRNA and with \(h = 0.001\) the estimates reproduce well the original model. The ODEs could not provide an accurate representation of the CTMC trajectories, for this reason we took in consideration also the stochastic setting. Figures 2 depicts the mean and the variance of the 100,000 simulation traces. As last observation, it is important to point out that by using smaller values of \(h\) the method gets better.

<table>
<thead>
<tr>
<th></th>
<th>(k_1)</th>
<th>(k_2)</th>
<th>(k_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.027</td>
<td>0.166</td>
<td>0.4</td>
</tr>
<tr>
<td>Estimate, (h=1)</td>
<td>0.6595</td>
<td>1.5386</td>
<td>0.1879</td>
</tr>
<tr>
<td>Estimate, (h=0.1)</td>
<td>0.1279</td>
<td>0.6385</td>
<td>0.3431</td>
</tr>
<tr>
<td>Estimate, (h=0.01)</td>
<td>0.1098</td>
<td>0.6683</td>
<td>0.4115</td>
</tr>
<tr>
<td>Estimate, (h=0.001)</td>
<td>0.0397</td>
<td>0.2591</td>
<td>0.4045</td>
</tr>
</tbody>
</table>

Table 1
Results of the parameter estimation for the gene transcription model

5.2 DFG degradation pathway

In order to test the method with a higher number of variables we use a model which describes the control of the N-(deoxy-D-fructos-1-y1)-glycine (DFG) degradation pathway [9]. The model can be found in the database available on the site www.sbml.org. It involves 14 reagents interacting through 16 reactions with mass action kinetics (the reactions are reported in Table 2). The original ODE model was transformed into a CTMC with discretization step \(h = 0.0001\) and started with initial concentration \([DFG] = 9\) and quantity 0 for all other reagents. Note that this choice of \(h\) leads to a CTMC with huge state space. The samples contained 20 observations with about 10000 reactions between consecutive samples. The E-step was performed with \(K = 20\) and \(\rho\) started from 0.2 and decreased throughout the calculations. The results are given in
Fig. 1. Gene transcription model: ODE computed with the original parameters and the estimates for different values of \( h \) (DNA\textsubscript{Off} on the left, DNA\textsubscript{On} on the right and mRNA below).

Table 3. It can be seen that the method gives good estimate for almost all the involved parameters. Figure 3 depicts the evolution of some of the reagents of the model according to the corresponding system of ODEs with the original and the estimated parameters. The stochastic setting is depicted in Figure 4 where in honor of synthesis we report the variance only. For most species there is a good correspondence between the original behaviour and the one provided by the estimate. In Figure 5 we depict instead the likelihood with which the estimates reproduce the original “in silico” generated trace. After 350 interactions the likelihood with the estimates is very close to the likelihood with the original parameters.

<table>
<thead>
<tr>
<th>Reactions</th>
<th>DFG ( k_1 \rightarrow E_1 )</th>
<th>DFG ( k_2 \rightarrow E_2 )</th>
<th>DFG ( k_3 \rightarrow Gly + Cn )</th>
<th>E1 ( k_4 \rightarrow Gly + DG3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DG3 ( k_5 \rightarrow Cn )</td>
<td>DG3 ( k_6 \rightarrow wFA )</td>
<td>E2 ( k_7 \rightarrow Gly + DG1 )</td>
<td>DG1 ( k_8 \rightarrow Cn )</td>
<td></td>
</tr>
<tr>
<td>DG1 ( k_9 \rightarrow AA )</td>
<td>E1 ( k_{10} \rightarrow Gly + Man )</td>
<td>E1 ( k_{11} \rightarrow Gly + Glu )</td>
<td>Man ( k_{12} \rightarrow Glu )</td>
<td></td>
</tr>
<tr>
<td>Glu ( k_{13} \rightarrow DG3 )</td>
<td>Gly + Cn ( k_{14} \rightarrow Mel )</td>
<td>Cn ( k_{15} \rightarrow AA + FA + MG )</td>
<td>E2 ( k_{16} \rightarrow Gly + Fru )</td>
<td></td>
</tr>
</tbody>
</table>

Table 2
Reactions of the DFG degradation pathway
Fig. 2. Gene transcription model: The average (left) and the variance (right) of the quantity of $D_{\text{na.off}}$, $D_{\text{na.on}}$, $mRNA$ computed with original parameters and the estimates for different values of $h$.

<table>
<thead>
<tr>
<th>Case 3</th>
<th>$k_1$</th>
<th>$k_2$</th>
<th>$k_3$</th>
<th>$k_4$</th>
<th>$k_5$</th>
<th>$k_6$</th>
<th>$k_7$</th>
<th>$k_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>0.005</td>
<td>0.015</td>
<td>0.015</td>
<td>0.079</td>
<td>0.090</td>
<td>0.027</td>
<td>0.212</td>
<td>0.181</td>
</tr>
<tr>
<td>Result</td>
<td>0.0048</td>
<td>0.0175</td>
<td>0.0119</td>
<td>0.058</td>
<td>0.068</td>
<td>0.010</td>
<td>0.252</td>
<td>0.706</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case</th>
<th>$k_9$</th>
<th>$k_{10}$</th>
<th>$k_{11}$</th>
<th>$k_{12}$</th>
<th>$k_{13}$</th>
<th>$k_{14}$</th>
<th>$k_{15}$</th>
<th>$k_{16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>1.908</td>
<td>0.070</td>
<td>0.113</td>
<td>8.0E-4</td>
<td>0.002</td>
<td>0.003</td>
<td>0.015</td>
<td>0.013</td>
</tr>
<tr>
<td>Result</td>
<td>1.847</td>
<td>0.0651</td>
<td>0.122</td>
<td>7.8E-4</td>
<td>0.014</td>
<td>0.003</td>
<td>0.0147</td>
<td>0.0122</td>
</tr>
</tbody>
</table>

Table 3
Result of parameter estimation for the DFG degradation pathway

**Conclusions**

In this work we adopted the MCEM method to the estimation of kinetic rates in stochastic reaction networks. We have shown that the resulting technique
is efficient and leads to surprisingly simple calculations in the case of mass action kinetics. The strength of the proposed approach is that it can be applied even with a limited set of observations of the modeled phenomenon. Several numerical examples have been provided to illustrate the computational characteristics of the method.
Fig. 4. The variance of some of the reagents involved in the DFG degradation pathway

References


Fig. 5. DFG degradation pathway: likelihood with estimates as function of the number of iterations


LocTrackJINQS:
An Extensible Location-aware Simulation Tool
for Multiclass Queueing Networks

Tzu-Ching Horng, Nikolas Anastasiou, Tony Field and William Knottenbelt
Department of Computing,
Imperial College London, South Kensington Campus
London SW7 2AZ
Email: {th107,na405,ajf,wjk}@doc.ic.ac.uk

KEYWORDS
Simulation, Queueing Network, Location Tracking

ABSTRACT
This paper presents LocTrackJINQS, a flexible and extensible spatio-temporal simulation tool for systems that involve the flow and processing of customers at multiple service centres. Developed based on the multi-class queueing network simulation package JINQS, LocTrackJINQS retains the abstract model specification power of JINQS while providing additional low-level models of entity movement. Besides traditional performance metrics, LocTrackJINQS produces as output a trace of each entity’s location in the system over time. It can thus be used to generate synthetic location tracking data for location-based research or applications.

1 INTRODUCTION
Simulation has long seen its use in both academic and business applications for system performance modelling or evaluation, workflow analysis, process improvement, asset/personnel management, etc. Simulation software tools such as [10], [3], [9], and [4] are offered to help organisations identify their system bottlenecks and gain better insights into the implications of different resource or personnel investments on overall system performance. By using simulation software decision makers can rapidly experiment with different scenarios and compare them at a fraction of the cost of real implementation in the system. However, some simulation tools have limited application domains (e.g. [3], [4]) and many of them also fail to represent the stochastic nature of time delays in the system.

The recent development of real-time location systems (RTLSs) enables the automatic collection of large amounts of high-precision location tracking data
in real-time. RTLSs have been widely deployed by enterprises, especially in the fields of supply chain management and healthcare [11, 8], because they provide enhanced insights into asset and personnel flows [2]. These in turn lead to improvements in performance efficiency, asset management and system safety. Several research endeavours have also been made recently to extract workflow patterns and models of system performance from location tracking data. For example, research work conducted in [6] constructs compressed probabilistic flow models for commodities in a supply chain by mining large RFID (Radio Frequency Identification) data sets. [7] and [1] show how stochastic performance models (queueing networks and stochastic Petri nets, respectively) can be automatically derived from high-precision location tracking data.

This paper presents **LocTrackJINQS**, originally developed to support location-based research ([7] and [1]), is an open-source simulation library for constructing simulations with location awareness. **LocTrackJINQS** is an extension of JINQS, a Java simulation library for multiclass queueing networks [5]. JINQS provides a suite of primitives that allow developers to rapidly build simulations for a wide range of stochastic queueing network models. It offers not only simplicity for simulation construction but also flexibility for application-specific functionalities through the use of inheritance [5]. However, JINQS only allows the creation of high-level simulations of abstract queueing networks and does not support realistic low-level features found in the physical world. This limitation makes it difficult to support simulations that can approximate entities’ physical movements in a real-life system, where entities’ travelling time might significantly influence the overall system response time. **LocTrackJINQS** retains the abstract high-level model specification power of JINQS while providing additional low-level models of entity movement. **LocTrackJINQS** also provides primitives for generating synthetic location tracking data similar to that collected from actual real-time location tracking systems. This eliminates the need for the heavy upfront investment and long-running observation periods that an RTLS installation requires, which benefits research in data mining location tracking data or for developing location-based applications.

The remainder of the paper first gives an overview of the software architecture of **LocTrackJINQS** and explains how the new features are implemented in **LocTrackJINQS**. We then present a case study to demonstrate how one can build up a simulator for a real-life system. The paper concludes with a summary of **LocTrackJINQS**’s new features and possible future extensions.

2 SOFTWARE ARCHITECTURE AND MAJOR EXTENSIONS

2.1 Overview

**LocTrackJINQS** is a simulation library that offers functionalities for simulating a real-life customer-processing system as a queueing network with low-level location information (see Figure 1). The user can not only specify the high-level features of the network such as the customer flow structure and time delay distributions (e.g., service time and inter-arrival time); but also low-level ones including entities’ geographic locations and their moving speeds and paths.

**LocTrackJINQS** inherits many features from JINQS. In both JINQS and
Figure 1: An example of simulating a real-life system using LOCTRACKJINQS:
Figure 1(a) demonstrates how a customer processing system is represented as
a high-level queueing network with low-level location information; Figure 1(b)
shows a screen shot of the simulation in progress and the generated location
traces.
LocTrackJINQS, there are two main Java packages: **network** and **tools** [5]. Classes in **network** are used to define the structure of queueing networks. Package **tools** provides utility classes for setting up simulations, defining common families of probability distributions and calculating performance metrics. The two packages have been designed to be easily extensible: developers can add on application-specific features only by subclassing those existing classes and overriding the inherited methods [5].

JINQS supports simulations of queueing networks with features including multiple servers, multiple customer classes and various queueing disciplines (e.g. FIFO, LIFO and priority-based); these are also supported by LocTrackJINQS. Inherited from JINQS, LocTrackJINQS also provides primitives for maintaining performance measurements at each service point as well as for the whole network. Two types of measurement variables are maintained – customer-oriented (e.g. mean response time) and system-oriented (e.g. mean population) [5]; the summary of all the measures can be displayed at the end of a simulation.

Extended from the features mentioned above, there are three main distinguishing features implemented in LocTrackJINQS:

- Support for location-aware simulations. As mentioned earlier, JINQS can only be used for constructing high-level abstract simulations, where each entity has no physical geographical location in the system and entities travel from one server to another instantaneously. LocTrackJINQS introduces location-related features to support more realistic simulations of real-life customer processing systems. In particular, each entity in the queueing network is assigned a geographical location represented in a 2D Cartesian coordinate system; entity movements occur along user-defined paths at speeds sampled from a user-specified distribution.

- Generate location updates. One of the applications of LocTrackJINQS is to support research on mining agent flow patterns. It thus offers the ability to generate synthetic, yet reasonably realistic, location tracking data traces from its simulations.

- Graphic user interface. LocTrackJINQS offers a graphical interface (defined in package **gui**) for setting up the simulation environment, specifying related parameters and monitoring the simulation process visually.

The following sections give more details on the important additions and modifications made to JINQS for implementing the new features.

### 2.2 Queueing Network Structure

In JINQS, a queueing network comprises three main types of entities, defined by **Node**, **Link** and **Customer** classes (see Figure 2). A network is structured as a collection of **Nodes** connected together by **Links**. The network is populated by **Customers**, which move among different **Nodes** along the connecting **Links** and request service or resources from the system. A subclass of **Node**, called **Source**, represents entry points where **Customers** are injected into the network according to a user-specified interarrival time distribution; **Sink** nodes are where **Customers** exit the network.

LocTrackJINQS defines queueing networks in a similar way, but introduces the entities’ geographical locations (specified as 2D Cartesian coordinates)
The system provides a limited pool of service points. Thus incoming Customer entity follows a user-specified distribution. A waiting time (e.g., interarrival time distributions, service time distributions, routing probabilities) to incoming Customer entities to their next destinations (selected probabilistically from out-going links). An InfiniteServer class provides no “service” to the customers; the service time for each served Customer entity is used as the criterion to decide whether the service point’s service area, either being served or queuing for service, can request and receive service) or a cluster of servers (known as a MultiQueueingServer which share a common queue. Figure 3 gives an overview of INode and the important types of INode. LocTrack-JINQS uses three classes (all implementing the INode interface) – that is Server and sub-classes InfiniteServer and QueueingServer – to define three basic types of service points in the system (see Figure 3). Service points are currently assumed to have fixed locations. Their service areas are assumed to be circular (the radii are user-specified) due to the fact that a Customer entity’s proximity to a service point is used as the criterion to decide whether the Customer entity has entered the service point’s service area, either being served or queuing for service.

An instance of the Server class provides no “service” to the customers; after it accepts a Customer entity to its service area, it immediately forwards Customer entities to their next destinations (selected probabilistically from out-going links). An InfiniteServer entity provides immediate service (i.e., no waiting time) to incoming Customer entities; the service time for each served Customer entity follows a user-specified distribution. A QueueingServer provides a limited pool of service points. Thus incoming Customer entities must queue for service if all the service points within the service area are busy.

Like JINQS, LocTrack-JINQS supports multiple customer classes, allowing for simulations of scenarios where Customer entities have class-dependent interarrival time distributions, service time distributions, routing probabilities and allows meaning to be assigned to their spatial relationships (e.g., defining how close a customer must be to a server in order to be served). LocTrack-JINQS further extends the notion of Node to include any space that a Customer entity might enter and stay for a period of time before departing for its next destination. This is implemented as an interface called INode, which defines the basic functions for accepting and forwarding Customer entities. Classes implementing this interface include the entry and exit points of the system (defined by Source and Sink classes in JINQS, respectively), a Server (comprising one or more service points and a corresponding service area within which Customer entities can request and receive service) or a cluster of servers (known as a MultiQueueingServer which share a common queue).
Figure 3: UML diagram of the important classes implementing INode interface in LocTrackJINQS

and service priority. Queueing discipline is priority-based and may be FIFO, LIFO or random within customer classes of equivalent priority.

The class MultiQueueingServers supports simulations of some commonly-seen scenarios in daily life, where several geographically-separated service points share one or more common queueing areas (an instance of the QueueingArea class); examples of these types of systems include post-office counters or hospital treatment rooms with patients waiting in a common waiting area.

In order to simulate Customer entities travelling from one location to another, LocTrackJINQS introduces classes PhysicalLink and TransportLink as subclasses of Link (as shown in Figure 4). Instead of the abstract connection that a Link provides, a PhysicalLink represents the physical path that Customer entities follow when moving between two INode entities. A path is composed of several line segments connected to each other by break points. Unlike JINQS, where the forwarding of Customers between two Node entities takes place in no time, a call to the moveCustomers() function of a TransportLink updates the locations of the Customers following the link based
on each Customer entity’s speed and direction of movement.

2.3 Generating Synthetic Location Tracking Data

Usually the entities we are interested in monitoring in a physical customer processing system are either those providing service/resources or those who are requesting service/resources, which correspond to Server (and its sub-classes) and Customer, respectively. LocTrackJINQS introduces a new class NetworkElement as the superclass of both Server and Customer classes (see Figure 5); it corresponds to an entity in a system that is “tagged” and thus “monitored” by the RTLS. Another new class, NetworkMonitor, is implemented to simulate the behaviour of a RTLS tracking and to output tagged entities' latest geographical locations. Similar to some of the existing RTLSs, each tag is assigned a QoS (Quality of Service) value, which is the tag’s location update rate requirement and defined as an attribute in NetworkElement. At each location update time slot, NetworkMonitor selects a tag and outputs its latest location with the best effort to satisfy each tag’s QoS, while avoiding starvation (i.e. a tag’s location is not updated for a long period of time). One thing to note is that different wireless technologies adopted by RTLS (e.g. Ultra Wide Band, Wi-Fi, and RFID), along with other factors (e.g. the density of sensors in range and objects that might interrupt sensors’ signal) contribute in practice to different error/noise patterns/distributions on location readings. It is thus impractical to assume any noise distributions in LocTrackJINQS. Accordingly, LocTrackJINQS allows the user to specify appropriate error distributions. A typical generated location update is of the form (tagName, type, time, x, y, stderr). tagName is a unique identifier for each entity in the system. type indicates the entity’s class, which can be used to differentiate a customer entity from a service point or to specify which customer class a customer entity belongs to. time is the timestamp when the location update is recorded and x and y specify the location of the tag. stderr is the expected deviation between the generated location update and the actual location of the entity.
2.4 New Event classes

Like JINQS, LocTrackJINQS follows the discrete-event simulation model. Under this model, a time-ordered diary of simulation events is maintained and time “hops” to the next event of interest. Processing (or invoking) an event may result in other events being added to the event diary. In both JINQS and LocTrackJINQS the Event class and its subclasses define possible events, such as a customer arrival event or a service completion event. We introduce two new subclasses of the Event class to support the location-based features implemented in LocTrackJINQS:

- **TransportCustomersEvent** class. The triggering of such an event invokes the moveCustomers() method of each TransportLink entity. By scheduling such an event to occur on a regular basis (e.g. every few milliseconds), we are able to simulate customer movement at a high resolution.

- **TagReadEvent** classes. When such an event is triggered, it invokes the updateTagReads() method defined in NetworkMonitor. According to their update rates, the “read” location of the tags (i.e. their true location adjusted according to user-defined error/noise distributions) are output to the trace file.

2.5 Graphical User Interface

To facilitate the creation and visualisation of location-enhanced simulation models, LocTrackJINQS includes a graphical user interface. This allows users to easily lay out the topology, to specify the parameters of a customer processing system and to animate it without having to write code (as is always required in JINQS). The facilities provided by the GUI are illustrated in the next section.

3 SIMULATION CASE STUDY

For the purpose of demonstrating the new capabilities implemented in LocTrackJINQS we present a case study of a small airport with two passport control and two security check points. Each of the check points has its own queue to hold waiting customers, who may be of three different classes:
- Pilots and flight attendants. This customer class has high priority.
- EU passengers. This customer class has low priority.
- Non-EU passengers. This customer class also has low priority.

The speed of all customers in this simulation environment is assumed to follow a Normal distribution with a mean and standard deviation of 0.3 and 0.1 m/s respectively. The remaining parameters of the simulation – i.e. interarrival time and service time distribution for each customer class – are depicted in Table 1. In all cases service priority-based with a FIFO queueing discipline within each priority class. The location update error has been set to be normally distributed with a mean of 0.15m and a standard deviation of 0.2m.

<table>
<thead>
<tr>
<th>Source (Interarrival time)</th>
<th>Pilots &amp; Flight Attendants</th>
<th>EU Passengers</th>
<th>Non-EU Passengers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal(2, 0.5)</td>
<td>Exp(0.02)</td>
<td>Exp(0.25)</td>
<td>Erlang(2, 0.33)</td>
</tr>
<tr>
<td>Passport Control (Service time)</td>
<td>Normal(2, 0.5)</td>
<td>Exp(0.25)</td>
<td>Erlang(2, 0.33)</td>
</tr>
<tr>
<td>Security Check (Service time)</td>
<td>Exp(0.2)</td>
<td>Erlang(4, 0.1)</td>
<td>Erlang(4, 0.1)</td>
</tr>
</tbody>
</table>

Table 1: Interarrival and service parameters for each customer class.

The high-level topology of the airport environment is depicted in Figure 6. It can be easily constructed using the GUI features of LocTrackJINQS as shown in Figure 7. Here the Source node (N1) is connected to the two passport control nodes (N2 and N3) through via two branches having equal probability. These two nodes are then connected with to server (N4) which is an infinite server with zero service time and which routes customers to one of two security check points (N5 and N6) with equal probability. Customers exit the system by the Sink node (N7).

![Figure 6: High-level description of the airport layout. Dashed arrow lines indicate the customer flow in the system.](image)
Figures 8 and 9 show the on-going simulation process at different time instants. Light blue, grey and dark blue coloured graphical figures are used to represent moving pilots and flight attendants, EU, and non-EU passengers respectively. In the same order as above, red, orange and pink colours are used to indicate that the entities are queueing. Three different shades of green are used to represent the customers from each class receiving service.

Table 2 and Table 3 shows the calculated mean response time, its standard deviation, and the mean queue length experienced by different customer classes at each service point and for the whole system. As shown in Table 2 and Table 3, the customer class with the highest priority (pilots and flight attendants) receive much better service quality than the other customer classes with lower priorities. Their mean response times at passport control and security check points are close to the assigned service time distributions and have low standard deviations, which means that they do not need to queue for long period of time and the service they enjoy is consistently of good quality. The other two customer classes experience much lower service qualities, with long response times with high standard deviations. From Table 2, we can also see that customers of European/UK and Non-EU passengers encounter long queues at passport control and security check points.

4 CONCLUSION AND FUTURE WORK

This paper presented LocTrackJINQS, a location-aware simulation tool developed based on the discrete-event simulation library for queueing networks, JINQS. It inherits many characteristics from JINQS, such as extensibility and simplicity for simulation construction, but incorporates location information of entities in the system and allows meaning to be assigned to their spatial relationships. To facilitate studies on location tracking data mining and analy-
sis, LocTrackJINQS provides functionalities for generating synthetic location tracking data from the simulation environment. LocTrackJINQS also provides a graphical user interface to support visual construction of queueing network models and parameter setting for the simulation. This was demonstrated in the context of an airport case study. Possible future work includes support for more sophisticated customer routing policies (e.g. join shortest queue), multidimensional customer classifications (e.g. a customer can belong to more than one classes based on multiple class definitions), as well as customer speed settings (e.g. customers would slow down when approaching a service area). We also plan to relax some of the current restrictions. For example, the service areas can have different shapes from being circular; instead of having fixed locations, server points can move around providing service to customers (e.g. nurses visiting hospitalised patients).

References

Figure 9: The airport simulation as time progresses.


[7] T.-C. Horng, N. Dingle, A. Jackson, and W. Knottenbelt. Towards the automated inference of queueing network models from high-precision loca-
<table>
<thead>
<tr>
<th>Class</th>
<th>Passport Control 1 (N2)</th>
<th>Passport Control 2 (N3)</th>
<th>Security Check 1 (N5)</th>
<th>Security Check 2 (N6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 0</td>
<td>µ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>8.754</td>
<td>4.830</td>
<td>24.979</td>
</tr>
<tr>
<td></td>
<td>σ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>4.793</td>
<td>2.918</td>
<td>17.221</td>
</tr>
<tr>
<td>Class 1</td>
<td>µ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>164.708</td>
<td>153.794</td>
<td>297.914</td>
</tr>
<tr>
<td></td>
<td>σ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>97.861</td>
<td>84.034</td>
<td>177.663</td>
</tr>
<tr>
<td>Class 2</td>
<td>µ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>155.025</td>
<td>152.709</td>
<td>272.432</td>
</tr>
<tr>
<td></td>
<td>σ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>88.892</td>
<td>96.924</td>
<td>200.560</td>
</tr>
<tr>
<td>µ&lt;sub&gt;q&lt;/sub&gt;</td>
<td>49.747</td>
<td>37.124</td>
<td>54.837</td>
<td>50.263</td>
</tr>
</tbody>
</table>

Table 2: Mean (µ<sub>rt</sub>) and standard deviation (σ<sub>rt</sub>) – in seconds – of the response time for each customer class and service point of the airport simulation. Customer classes 0, 1 and 2 correspond to Pilots & Flight Attendants, EU passengers and non-EU passengers respectively. µ<sub>q</sub> represents the mean queue length for each service point.

<table>
<thead>
<tr>
<th></th>
<th>Class 0</th>
<th>Class 1</th>
<th>Class 2</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>µ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>246.160</td>
<td>520.750</td>
<td>479.937</td>
<td>420.563</td>
</tr>
<tr>
<td>σ&lt;sub&gt;rt&lt;/sub&gt;</td>
<td>82.376</td>
<td>229.931</td>
<td>223.166</td>
<td>225.557</td>
</tr>
</tbody>
</table>

Table 3: Mean (µ<sub>rt</sub>) and standard deviation (σ<sub>rt</sub>) – in seconds – of the response time for each customer class, as well as the aggregate for the system. Customer classes 0, 1 and 2 correspond to Pilots & Flight Attendants, EU passengers and non-EU passengers respectively.


Defining formalisms for performance evaluation with SIMTHESys

Enrico Barbierato
DI, Università degli Studi di Torino
corso Svizzera, 185, 10129 Torino (Italy)

Marco Gribaudo
DEI, Politecnico di Milano,
via Ponzio 34/5, 20133 Milano (Italy)

Mauro Iacono
DEM, Seconda Università degli Studi di Napoli
Belvedere Reale di San Leucio, 81100 Caserta (Italy)

Abstract

Tools for the analysis and modeling of complex systems must be able to support the extensibility of formalisms, reusability of models and customization of formalism compositions. From this perspective, SIMTHESys (Structured Infrastructure for Multiformalism modeling and Testing of Heterogeneous formalisms and Extensions for SYStems) is a new approach to the specification of performability oriented formalisms and the evaluation of models. Its originality emerges from the explicit definition of both syntax and evolution semantics of the considered formalism elements. The solution of models is made possible by using a set of non-specialized solving engines used to generate automatically formalism-specific reusable solvers. This paper explains how formalisms can be created in SIMTHESys by showing how three widely known modeling languages are successfully implemented.

Email addresses: enrico.barbierato@mfn.unipmn.it (Enrico Barbierato), gribaudo@elet.polimi.it (Marco Gribaudo), mauro.iacono@unina2.it (Mauro Iacono)

Preprint submitted to PASM 2011

February 28, 2011
1. Introduction

Some of the challenges that current systems can present to the scientific and professional community in terms of performance, reliability and service level agreement can be addressed by analyzing models based on many different performability formalisms.

Modeling is a complex task because of the heterogeneity of the system needed to be studied. The customization of some of the formalisms for a specific task can simplify the modeling process at a cost of creating a tool to solve a model based on an extended formalism. The idea of creating a tool for each formalism extension seems not to be feasible. Notwithstanding the huge number of known modeling formalisms, they share some basic fundamental ideas that suggests a classification in families. These can be re-grouped using a common solver that can be specialized for the different cases. The SIMTHESys approach, presented in [14] moves onward from such premise to propose a new solution to the problem. SIMTHESys offers a compositional, metamodeling based framework to describe and extend formalisms.

This paper aims to demonstrate how to design formalisms belonging to the Exponential Event Formalisms (EEF) family. Three cases are presented, showing how three common performance evaluation frameworks, namely Stochastic Petri Nets (SPN) [15], Tandem Finite Capacity Queueing Networks (TFCQN) [13] and Gordon and Newell Queueing Networks (GNQN) [10] can be defined using the SIMTHESys approach. Two solution components architectures have been designed for the EEF family. Respectively, they perform stochastic simulation and numerical solution. Additionally, a simple model is analyzed to show the possibilities offered by the solution components.

The paper is organized as follows. After a brief review of the SIMTHESys approach to multiformalism modeling in Section 2, in Section 3 the exponential transition based family of formalisms is introduced. Section 4 describes

---

1 Also visit the SIMTHESys web site at www.dem.unina2.it/simthesys
2 The approach also supports multiformalism models, obtained by connecting together submodels written in different formalisms by exploiting their dynamics, but this feature is out of the scope of this paper
the implementations of SPN, TFCQN and GNQN. Section 5 presents the solution engines for the mentioned family. Conclusions and future work are described in section 6.

2. SIMTHESys overview and related works

The early experiences of Sharpe [17] [19], SMART [2] and the DEDS toolbox [1] to Mobius [3] [6] [18] [4] [5] and OsMoSys [8] [21] [16] [20] [9] (that are the closest references for this research) proved the value of a formal (multiformalism/multisolution\(^3\)) approach to modeling and evaluation of systems.

SIMTHESys is a new framework for the definition and analysis of performance oriented formalisms. It is based on the explicit definition of both syntax and semantics of all atomic components of a formalism and on a set of non-specialized solving engines, that are used to generate automatically and transparently (multi)formalism-specific reusable solvers.

The main advantage of the SIMTHESys approach is that it allows rapid prototyping of new formalisms and solution techniques, with native multiformalism support.

With respect to [14], which aims to define how the SIMTHESys methodology can be used to support interaction between multiformalisms models, this paper focuses on the process of creating a new formalism by presenting the steps required to implement three known formalisms.

A SIMTHESys formalism is a formal description language. It is defined in terms of its elements, that are its atomic components. Each element is characterized by a set of attributes called properties and a set of descriptions of its dynamics called behaviors. Properties can be constant characteristics of the element, state information (useful for the solution of the model) or results (obtained by the solution process). Behaviors describe the semantics of the element (the effects of their presence in the model on the other elements, e.g. its execution policy, if applicable). A behavior is an algorithm implemented in a high-level programming language (that currently follows

\(^3\)Multiformalism refers to the possibility of using different formal languages to specify different portions of a model. Multisolution identifies the possibility of integrating different existing performance evaluation tools to compute the performance indices of a complex model.
the syntax of Java). Every formalism has an additional *container element* that represents a (sub)model written in that formalism and whose properties include global characteristics of a model. Formalisms are described by the Formalism Description Language (FDL).

A *model* is the description of a system being evaluated, written according to a formalism. Models can be hierarchically composed, to separate reusable (sub)models of its subsystems, even if written in different formalisms. Models are described by the Model Description Language (MDL) documents.

The framework is composed of three components: the SIMTHESys Metamodeling Structure (SimMS), the *Interfaces* (IFs) and the *Solving Architecture* (SA). The IFs fill the gap between the high abstraction level of formalisms and the generic applicability of the solvers. IFs constitute the foundation on which elements behaviors are built, supplying general reusable functions and access to solving engines. IFs can be seen as a middle layer supporting the interactions between SimMS and SA. They decouple the problem of solving a model in the best way available from the problem of describing it with the best suitable formalism. As a result, IFs are different from the Abstract Functional Interface (AFI) considered by Mobius.

This software architecture is designed to represent efficiently the development of extensions of formalisms, the evaluation of formalism variants, and to support multiformalism.

The SA offers an extensible set of *Solving Engines*, suitable for the evaluation of performance indices or other significant properties of a model. Solving engines are generic solvers, that are meant to be used or composed to obtain a solution component for a certain formalism defined in a FDL document. Solution engines offer proper interfaces to the IFs and will be discussed in more detail later in the paper.

The SIMTHESys framework has been developed to be integrated with the DrawNET tool and the OsMoSys framework. DrawNET is a

---

4A part of the project's future work in this area will be committed to the definition of a high-level programming language to allow the developer to implement new behaviors. Such a language will then be translated in Java or C++ depending on the numerical precision requested from the solving engine.

5A model with heterogeneous submodels is written in a *composition formalism*, that is a formalism capable of semantically connecting concepts belonging to different formalisms.

6Although the focus of this paper is not on multiformalism and multisolution, many of the design choices aim at simplifying the interoperability between different formalisms.
formalism definition environment that provides data structures and the automatic creation of graphical user interfaces for the rapid development of performance evaluation tools. OsMoSys is a framework that supports the definition of formalisms and models to allow the use of multiformalism and multisolution techniques.

3. Exponential Event Formalisms

The family of formalisms presented in this paper are labeled as Exponential Event Formalisms (EEFs). In these formalisms, primitives represent entities capable of generating events. Such events (that can be used to represent the firing of a PN transition, or the end of the service in a queue) are characterized by the fact that they occur after an exponentially distributed time.

Each formalism belonging to this class should define a behavior called InitEvents. Its purpose is to determine the events that are enabled in a given state and the constant rate $\lambda$ that characterizes the exponential distribution. If more than one event is enabled at the same time, a race policy is used to solve the conflict: the event with the smallest firing time is selected for firing.

The solution engines expose a behavior Schedule that is used by the implementations of the InitEvents behavior to define what must be executed whenever an event occurs. The code associated with the event updates the state of the elements to reflect the evolution of the model. The scheduled behavior updates the state of the affected elements of the model. For example a transition can move tokens among the places to which is connected by using specific arcs behaviors.

3.1. Performance Indices

EEFs may be characterized by performance indices calculated by using State Rewards and Impulse Rewards, similarly to those defined in [6].

State Rewards return the mean value of a function of the state of the model. They are used to compute the mean length of a queue, or the mean number of tokens in a PN place. Each EEF define a set of state rewards by implementing three behaviors. Firstly, CountStateRewards returns the number of state rewards that a model is able to expose. Secondly, ComputeStateRewards computes the value of all the rewards associated with
the model in a given state. Finally SetStateRewards allows the solution engine to return the computed indices to the model.

Impulse rewards are used to compute measures related to the firings of events, such as the throughput of queues or PN transitions. Each impulse reward is characterized by a unique reward name, and is defined by implementing two behaviors and by passing appropriate parameters to the previously defined solution engine Schedule. A formalism should list the reward names of all the impulse rewards by implementing the behavior ListImpulseRewards. Every time an event is scheduled, the formalisms pass to the Schedule behavior the name of a reward that should be updated, and the magnitude of the impulse. As for the state rewards, the behavior SetImpulseRewards allows the solution engine returning the computed reward to the model.

4. Case study: formalisms implementations

This section shows how the behaviors defined in Section 3 should be implemented by the TFCQN, GN-QN and SPN formalisms to compute the solution of their corresponding models using a solution engine defined for a EEF.

4.1. Tandem Finite Capacity Queueing Networks

Queueing Networks (QN) is a formalism suitable for the analysis of systems in which a number of servers are connected to serve customers, which wait in a queue. A QN is composed by two kinds of elements: the queue and the arc. A queueing network can be closed (the same N customers keep being continuously served in the network) or open (some customers join the network according to a given interarrival time distribution and some of them leave).

4.1.1. Formalism analysis

TFCQN is a variant of QN in which every queue has a finite number M of places for waiting customers (Finite Capacity). Only a single arc can leave a queue (Tandem network)\(^7\). If there is no room for a customer in a queue, the

\(^7\)Note that though probabilistic branching from service nodes allows to represent a more general model, only TFCQN have been considered for the sake of simplicity.
input stations stop serving until a place is available. This occurs as a result of a blocking mechanism.

The three most common are: Blocking After Service (BAS), Blocking Before Service (BBS) and Repetitive Service (RS). In the first case, the customer who does not find room in the destination queue is processed anyway and it is blocked right after service completion. In the second case, the source queue is blocked before processing the customer. In this case, either the customer enters the server (BbsSoQueue) or not (BbsSnoQueue, as a result the number of places in the queue is set to M-1 until it is unblocked). In the third case, if the destination queue does not have room left, a customer who finishes the service is reinserted in the same queue to be served again later (RSqueue).

Note that BAS queues is not supported. Even if their service time satisfies the EEF property, in case an unblocking condition is set they are supposed to transfer the waiting customer immediately to the destination queue.

On an abstract level, TFCQN stations are characterized by two structural properties: the maximum capacity of the queue and the rate of the exponential distribution corresponding to the service time. The state of each queue is uniquely specified by its length at a given instant of time. The performance indices that are commonly computed on TFCQNs are the mean queues length and the throughput of each station.

4.1.2. Formalism implementation

TFCQN is implemented either by defining a generic queue object with an associated property that specifies the kind of block or by defining a different element type for each blocking policy. We chose the second alternative because it simplifies the coding of the behaviors. The next step to consider is to identify the kind of (discrete) event corresponding to the termination of a service provided to a customer.

Table 1 presents the elements, properties and behaviors that can be used to define a TFCQN in SIMTHESys. All the queues have the same attributes and behaviors, but the latter are implemented differently. Properties reflect the description provided in Section 4.1.1.

---

8They belong to the ExpAndImmediateEvent family (EAIEF), also available in SIMTHESys.

9Structural properties are static information associated with the elements of the formalism that do not change during the model evolution.
Note that the ‘Modifier’ attribute associated with each property represents its role in the element definition. Structural properties have a ‘const’ modifier, and dynamic information are stored in properties with the ‘status’ modifier. Finally, performance indices have a ‘computed’ modifier.

<table>
<thead>
<tr>
<th>Element</th>
<th>Property</th>
<th>Type</th>
<th>Modifier</th>
<th>Behaviors</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFCQN</td>
<td></td>
<td></td>
<td></td>
<td>InitEvents, ComputeStateRewards, CountStateRewards, SetStateRewards, ListImpulseRewards, SetImpulseRewards</td>
</tr>
<tr>
<td>Arc</td>
<td>from to</td>
<td>element</td>
<td>const</td>
<td>HasSpace, Push</td>
</tr>
<tr>
<td>BbsSnoQueue</td>
<td>length</td>
<td>integer</td>
<td>status</td>
<td>IsActive, AddOccupancy, Fire, CanSend, CanAccept</td>
</tr>
<tr>
<td></td>
<td>meanlength</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capacity</td>
<td>integer</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rate</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>throughput</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BbsSoQueue</td>
<td>length</td>
<td>integer</td>
<td>status</td>
<td>IsActive, AddOccupancy, Fire, CanSend, CanAccept</td>
</tr>
<tr>
<td></td>
<td>meanlength</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capacity</td>
<td>integer</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rate</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>throughput</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSQueue</td>
<td>length</td>
<td>integer</td>
<td>status</td>
<td>IsActive, AddOccupancy, Fire, CanSend, CanAccept</td>
</tr>
<tr>
<td></td>
<td>meanlength</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>capacity</td>
<td>integer</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rate</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>throughput</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Elements of the TFCQN SIMTHESys definition

The InitEvents behavior is defined in Algorithm 1. The object oriented like dot notation is used to name the behaviors and the properties associated to the elements of a model. The external object solver refers to the solution engine, that exposes the method Schedule to enable the events. This method has four parameters that identify: i) the rate of the exponential distribution that characterizes the firing time, ii) the piece of code that must be executed.
when the event occurs, iii) the name of the impulse reward that is associated to the event, and iv) the increment of the reward. The notation $q.id$ is used to identify the name of the queue. Since every queue has an associated throughput, the name of the queue is used as the name of the corresponding reward. Due to the fact that throughput counts the number of services in a queue, its corresponding reward value is always 1.

**Algorithm 1 InitEvents**

1: for all $q \in \text{RSQueue} \cup \text{BbbSoQueue} \cup \text{BbsSnoQueue}$ do
2:   if $q.\text{isActive}() \land q.\text{CanSend}()$ then
3:     solver.\text{Schedule}(q.\text{rate}, “q.\text{Fire}()”, q.id, 1);
4:   end if
5: end for

The $\text{IsActive}$ behavior is identical for all the types of queues, and simply checks if the length of the corresponding queue is greater than 0. The $\text{CanSend}$ behavior is used to check whether a station can start its service or it is blocked because it has reached the full capacity of the destination node. In an RSQueue, the behavior always returns $true$ (because the service is always enabled, and it is re-issued if the destination node is full). Regarding the other types of queue, the $\text{CanSend}$ behavior is specified in Algorithm 2.

**Algorithm 2 CanSend**

1: for all $a \in \text{Arc}$ where $a.\text{from} = \text{this}$ do
2:   if NOT $a.\text{HasSpace}()$ then
3:     return $false$;
4:   end if
5: end for
6: return $true$;

The special keyword $\text{this}$ is used to identify the queue to which the behavior is associated, and the purpose of the $\text{forall}$ statement is to look for all the possible destinations of the queue (in the tandem assumption, it is at most one station). The $\text{HasSpace}$ behavior associated to an arc, shown in Algorithm 3, calls the $\text{CanAccept}$ behavior of the destination (specified by the property $to$).

Each queue defines a behavior $\text{CanAccept}$ that returns $true$ if the queue has enough space to accept an incoming customer. Concerning the RSQueue
Algorithm 3 HasSpace
1: return to.CanAccept();

and the BbsSoQueue, the algorithm checks if the total length of the queue
(property length) is less than the available space (property capacity) as re-
ported in Algorithm 4.

Algorithm 4 CanAccept - BbsSoQueue and RSQueue
1: return length <= capacity;

In the case of the BbsSnoQueue, the algorithm considers the fact that the
capacity can be reduced because the destination is full, as shown in Algorithm
5.

Algorithm 5 CanAccept - BbsSnoQueue
1: if length < capacity then
2: return true;
3: else
4: return CanSend() && length < capacity;
5: end if

The Fire behavior implements the end of the service of a customer in
a queue. In the case of the BbsSoQueue and BbsSnoQueue, it sends the
customer to the next station using the definition provided in Algorithm 6.

The RSQueue algorithm checks if the destination queue is empty before
sending the customer there. If the destination is full, it reschedules the same
service. This is implemented in Algorithm 7. Note that instructions for the
rescheduling are not necessary (this is automatically done in the InitEvents
behavior).

Both algorithms use the Push behavior of the connecting arc to send
the customer to the next station. The Push behavior in turn calls the
AddOccupancy(1) behavior of the queue at the other end of the arc, which is
implemented in the same way for all the queue types. The AddOccupancy(c)
behavior adds c customer to the length of each queue.

The behaviors used to compute the performance metrics have been im-
plemented as follows: i) CountStateRewards returns the number of queues
in the model; ii) ListImpulseRewards returns a list of their identifiers; iii)
Algorithm 6 Fire - BbsSoQueue and BbsSnoQueue
1: for all $a \in$ Arc where $a\.from = this$ do
2: \hspace{1em} a\.Push();
3: \hspace{1em} end for
4: length = length - 1;

Algorithm 7 Fire - RSQueue
1: for all $a \in$ Arc where $a\.from = this$ do
2: \hspace{1em} if $a\.HasSpace()$ then
3: \hspace{2em} a\.Push();
4: \hspace{2em} length = length - 1;
5: \hspace{1em} end if
6: end for

ComputeStateRewards returns a vector with the length of the queues in the current state; iv) SetStateRewards fills the meanLength property, and v) SetImpulseReward sets the Throughput property.

4.2. Gordon and Newell Queueing Networks

GNQN is another variant of QN. A GNQN is a closed network in which every queue has a server with service time according to the EEF property and first come first served policy.

Table 2 defines the elements required to implement GNQNs in SIMTHESys. The main differences with respect of the previous example are that in this case a queue can have more than one possible destination and has no space constraint. In this case a probability associated with the outgoing arc (property $prob$) determines the frequency at which a given destination is chosen.

All the behaviors have exactly the same implementation as for the TFCQN case, except for InitEvents and Fire. Gordon and Newell queueing networks are characterized by the property of choosing the next station either at the end of a service or at its beginning. Given the probability $p_i$ of choosing the $i$-destination for a service operating at rate $\lambda$, due to the EEF property the action of scheduling a service at rate $\lambda$ first and then choosing the destination with probability $p_i$ is equivalent to the action of scheduling one service for each possible destination $i$ at rate $p_i \cdot \lambda$. The InitEvents exploits this property to implement the choice of the next station, following Algorithm 8.

Note that in this case the event is scheduled at rate $a\.prob \cdot q\.rate$. In
<table>
<thead>
<tr>
<th>Element</th>
<th>Property</th>
<th>Type</th>
<th>Modifier</th>
<th>Behaviors</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNQN</td>
<td></td>
<td></td>
<td></td>
<td>InitEvents, ComputeStateRewards, CountStateRewards, SetStateRewards, ListImpulseRewards, SetImpulseRewards</td>
</tr>
<tr>
<td>Arc</td>
<td>from</td>
<td>element</td>
<td>const</td>
<td>Push</td>
</tr>
<tr>
<td></td>
<td>to</td>
<td></td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>Queue</td>
<td>length</td>
<td>integer</td>
<td>status</td>
<td>IsActive, AddOccupancy, Fire</td>
</tr>
<tr>
<td></td>
<td>meanlength</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rate</td>
<td>float</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td></td>
<td>throughput</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Elements of the GNQN SIMTHESys definition

Algorithm 8 InitEvents

1: for all $q \in$ Queue do
2: if $q$.IsActive() then
3: for all $a \in$ Arc where $a$.from = $q$ do
4: solver.Schedule($a$.prob * $q$.rate, "$q$.Fire($a$)", $q$.id, 1);
5: end for
6: end if
7: end for

In this case the Fire behavior of a Queue has a parameter that defines the destination of the service. The Fire behavior is thus implemented following Algorithm 9.

Algorithm 9 Fire($a$)

1: $a$.Push();
2: length = length - 1;

4.3. Stochastic Petri Nets

Petri Nets (PN) is a formalism suitable for modeling concurrent systems. A PN is composed of four kinds of elements: the arc, the inhibitor arc, the
place and the transition.

SPN (\cite{15}) are a variant of PN that takes into account a time variable and in which enabled transitions fire according to the EEF property: thus a transition in SPN is characterized by a rate.

SPN can be defined in SIMTHESys in table 3. The complete FDL description of SPN is given and commented in \cite{14}.

<table>
<thead>
<tr>
<th>Element</th>
<th>Property</th>
<th>Type</th>
<th>Modifier</th>
<th>Behaviors</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPN</td>
<td>bounded</td>
<td>boolean</td>
<td>computed</td>
<td>InitEvents, ComputeStateRewards, CountStateRewards, SetStateRewards, ListImpulseRewards, SetImpulseRewards</td>
</tr>
<tr>
<td>Arc</td>
<td>weight</td>
<td>integer</td>
<td>const</td>
<td>IsActive, Push, Pull</td>
</tr>
<tr>
<td></td>
<td>from to</td>
<td>element</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>Inhibitor Arc</td>
<td>weight</td>
<td>integer</td>
<td>const</td>
<td>IsActive</td>
</tr>
<tr>
<td></td>
<td>from to</td>
<td>element</td>
<td>const</td>
<td></td>
</tr>
<tr>
<td>Place</td>
<td>marking</td>
<td>integer</td>
<td>state</td>
<td>GetOccupancy, AddOccupancy</td>
</tr>
<tr>
<td></td>
<td>meantokens</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
<tr>
<td>Transition</td>
<td>rate</td>
<td>float</td>
<td>const</td>
<td>IsActive, Fire</td>
</tr>
<tr>
<td></td>
<td>throughput</td>
<td>float</td>
<td>computed</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Elements of the SPN SIMTHESys definition

In the \texttt{InitEvents} behavior the formalism checks if all transitions are enabled and schedules the firing of those enabled at the corresponding rate following Algorithm 10.

\textbf{Algorithm 10 InitEvents}

1: \textbf{for all} \( T \in \text{Transition} \) \textbf{do}
2: \hspace{1em} if \( T.\text{IsActive}() \) \textbf{then}
3: \hspace{2em} \texttt{solver.Schedule}(T.rate, “T.Fire()”, T.id, 1);
4: \hspace{1em} \textbf{end if}
5: \textbf{end for}

The \texttt{IsActive} behavior of a transition looks both for incoming arcs and
inhibitor arcs, and is reported in Algorithm 11. Both inhibitor and standard arcs implement the \texttt{IsActive} behavior by checking that the marking of the incoming place (read using the \texttt{GetOccupancy} behavior of the place) is respectively less, or greater or equal to their weight. Finally, when a transition fires it updates the marking following the \texttt{Fire} behavior described in Algorithm 12.

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{for all} $a \in \text{Arc} \cup \text{InhibitorArc}$ where $a.to = \textbf{this}$ \textbf{do}
\State \quad \textbf{if NOT} $a.\texttt{IsActive}()$ \textbf{then}
\State \qquad \textbf{return} \texttt{false};
\State \quad \textbf{end if}
\State \textbf{end for}
\State \textbf{return} \texttt{true};
\end{algorithmic}
\end{algorithm}

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{for all} $a \in \text{Arc}$ where $a.from = \textbf{this}$ \textbf{do}
\State \quad $a.\texttt{Push}()$
\State \textbf{end for}
\State \textbf{for all} $a \in \text{Arc}$ where $a.to = \textbf{this}$ \textbf{do}
\State \quad $a.\texttt{Pull}()$
\State \textbf{end for}
\end{algorithmic}
\end{algorithm}

Note that in this case the arcs implement two behaviors, i) \texttt{Push} and ii) \texttt{Pull} that respectively add and subtract as many tokens as their weight from the place connected to the other end of the arc using the \texttt{AddOccupancy} behavior. Performance indices are implemented by defining a state reward for each place (its mean number of tokens), and an impulse for each transition (its throughput).

5. Solution Engines

\textsc{Simthesys} provides two solution engines for the EEF presented in Section 3, that solve models using discrete event simulation and steady state analysis of the underlaying Markov chain (the engine requires this to be finite). Both engines take a \textit{snapshot} of the state of the model by storing all the properties with the \texttt{status} modifier, and then back track to it.
they implement the Schedule behavior by storing all the scheduled events into a list. Note that as SIMTHESys aims at supporting multiformalism development, currently the solving engines have not been optimized. They are presented here to show that it is possible to solve models written in a large variety of formalisms without having to re-design new solvers by hand. The comparison with other known approaches to modeling and evaluation of systems is outside the scope of this paper.

5.1. Stochastic Simulation

Due to the EEF property, it is sufficient for the simulator to reschedule all the events after each firing. The simulator repeats the analysis for a fixed number $N_{\text{runs}}$ of runs. Each run is executed until a global time $T_{\text{max}}$ is reached; statistics are collected only after a transient time of fixed length $T_{\text{trans}}$. The parameters $N_{\text{runs}}$, $T_{\text{max}}$ and $T_{\text{trans}}$ are constants defined by the modeler. A snapshot of the initial state is taken, and after each run has finished, the snapshot is used to start a new simulation from the same initial state. The execution of each run calls the InitEvents behavior to find all the enabled events, and then draws an exponentially distributed sample for each of them. The event with the shortest sample is executed, and time is advanced accordingly until $T_{\text{max}}$ is reached. At the end of all the simulation runs, statistics are collected and returned to the model using the SetStateRewards and SetImpulsReward behaviors.

5.2. Numerical Analysis

The numerical solution solver generates the CTMC (Continuous Time Markov Chain) that describes the stochastic process equivalent to the model. Starting from the initial state, the solver calls the InitEvents behavior to compute all the enabled events. The solver then builds a transition graph, executing each enabled event. Then, it checks if the snapshot of the obtained state has already been encountered (if not, it adds a new state) and backtracks to the initial event. The process is repeated until all the states have been visited. The use of snapshots is twofold: to reset the properties to a previously encountered state, and to backtrack to the current state whenever an enabled event is considered. Event firing rates are used to label the arcs of the transition graph. Each time a new state is found, the solver computes the associated reward vector: for state rewards it accounts for the values returned by the ComputeStateRewards behavior, while for impulse rewards, it considers the reward value multiplied by the rate of the corresponding
enabled event. The generator matrix $C$ of the underlying Markov chain is next computed from the transition graph. Finally, the solver computes the steady state solution vector $\pi$ of the Markov chain by computing $\pi C = 0$, and normalizing the solution such that all the components of $\pi$ sums up to 1. Performance indices are computed by multiplying $\pi$ times the reward vectors, and stored back into the model using the `SetStateRewards` and `SetImpulseReward` behaviors.

5.3. Example

In Figure 1 a closed TFCQN with three servers is taken as example \(^{10}\).

![Figure 1: Example TFCQN, with N initial customers per queue](image)

The capacity $L_{\text{max}}$ and the rate $\lambda$ for each queueing station are specified together with the initial number of customers $N$. The model has been solved with both analytical and simulative engines for different values of $N$. The results for the simulative and analytical engines are sketched respectively in Figure 2 and 3. Confidence intervals for the simulation are omitted for the sake of brevity. For $N = 20$ the model has 1801 states. For $N = 100$, as the number of states grows proportionally to $N^3$, the java implementation of the numerical solution is not able to compute the results (though the optimization of the engines is not in the scope of this paper, a C++ implementation has provided promising results). As expected, all queueing stations have the same throughput except for RS, that re-issues the requests in case of block. The slowest queueing station is saturated while RS serves all the customers that did not manage to have access to it. The third server has a stable number of customers, tending to the mean length of the same queueing station if isolated in an open network with infinite capacity ($\rho/(1-\rho)$), where $\rho = \lambda/\mu$, $\lambda = 1.5$ and $\mu = 2$.

\(^{10}\)More complex examples can be found on SIMTHESys web site at www.dem.unina2.it/simthesys
6. Conclusions and future works

In this paper three example formalisms have been implemented by a new approach to multiformalism modeling. The SIMTHESys approach is based on the use of the concept of behavior to define formalisms and their semantics independently of the solver used. The three examples exploit the same solution engines and demonstrate the advantages of this approach. This research is the first detailed description of how a formalism and its solver can be implemented in SIMTHESys. Research in this area aims to better understand the possibilities opened up by the behavior mechanism. This could be achieved by implementing more formalisms linked to different families, solution engines and enhancing the usability of the behaviors by designing a dedicated scripting language. Other interesting perspectives are given by further investigation into multiformalism models and their application to different real world problems.
References


Abstract
Service Level Agreements (SLAs) are used to guarantee quality of service (QoS) between customers and service providers. In an SLA, parties establish a common set of rules and responsibilities. In this paper we propose a practical stochastic modeling of a multi-tier architecture considering SLAs for specific transactions. The model is parameterized with available performance testing data for a real web service, and with a testing environment having unpredictable and unknown external workloads of simultaneous execution. In addition, we present multiple scenarios of external applications impacting on the SLAs in our target architecture. Having a previous knowledge about the average time demanded by some external applications, our model results can provide evidences when the system under test will not respect the agreed-upon SLAs. Finally, we discuss possible model extensions towards further unknown workload characterizations and considerations about application execution profiling.

Keywords: Analytical Modeling, Stochastic Automata Networks, Service Level Agreements, Quality of Service, Performance Evaluation

1. Introduction

Service Level Agreements (SLAs) are contracts between service providers and customers and are used to ensure that an application will deliver a high quality of service (QoS) in a timely manner. Ensuring SLAs to execute within healthy environments is important to performance testing since it enables the capacity planning process as well as scalability analysis due to predictable load increase, user growth or based on patterns of future use.

According to Software Performance Engineering (SPE) practices [1], performance analysts doing Performance Testing are expected to enforce that the System Under Test (SUT) is executed...
in isolation and following a specific test plan according to a well-defined test objective [2, 3]. Respecting SPE practices adds a layer of responsibilities, new roles (human resources such as performance analysts and similar capacities) invariably leading to additional costs, and also new risks to software projects. However, it is not always possible to follow SPE guidelines rigorously due to the low priority (or impossibility) that is associated with pure performance testing in software projects. In contrast, Functional Testing is considered more important for stakeholders and thus, has a higher priority level. In software projects, one must balance the advantages of using SPE to devise responsive and performant applications in respect to the effort to be spent.

Both types of tests are equally important and usually executed in testing environments to comply with distinct objectives. The former is interested in testing the system for failures (according to the taxonomy proposed by Avizienis et al. [4]) with respect to the functional aspects of software, whereas the latter works with the non-functional requirements of software, e.g., availability, usability, quality of service or compliance with design specifications, to name a few. Generally, functional testing is applied to validate or to verify software products and performance testing is used to attest quality attributes of systems under particular workloads. Despite the fact that performance testing is often considered after functional testing, this process is equally important because it helps improving overall quality of service.

One should remark the fact that perfect conditions for having a dedicated performance testing environment seldom exists. Problems such as servers belonging to remote locations, security constraints (e.g., static firewall machines), unavailability of the maintenance team for emergency repairs and general updates, general project miscommunications on the test purpose (both for failure discovery and performance), among other factors contribute to degrade web services responsiveness. Web Servers, for instance, for small or large organizations are configured to run multiple types of services, e.g., mail server, multiple instances of java virtual machines governed by application server options or external applications under software testing.

To enhance QoS, web services stipulate contracts to be executed within certain predefined amounts of time, i.e., they should respect an SLA to avoid contractual penalties. In this context, we are interested to devise a stochastic model that describes the operation of web services under performance testing subjected to a testing environment running several external applications with unpredictable workload intensities. Note that external applications also share important resources that may deteriorate the response time of the application under performance testing, and sometimes contribute to break SLAs. Our proposed model is parameterized with data obtained from a performance testing study of a critical application of a large software corporation (omitted here due to signed non-disclosure agreements). Our results demonstrate that characterizing a set of known external workloads intensities it is possible to devise maximum levels of response times given the service demands and verify if the SLA is not respected.

The remainder of this paper is organized as follows. Section 2 addresses software testing and performance testing generally. Our target application is explained on Section 3 as well as its architecture and internal operational details. In Section 4 we discuss stochastic modeling and the formalism known as Stochastic Automata Networks. Section 5 presents our model and an analysis of our results. Finally, in Section 6, we present our final considerations and future works.

2. Software Testing for Performance and Failure Analysis

Software Testing is a crucial task in current Information Technology (IT) organizations because it helps ensuring the delivery of high quality products to end customers [5]. There are two aspects of testing that must be considered for every type of software project (small, medium
and huge sizes), i.e., failure analysis (or functional software testing) and performance testing. **Functional Software Testing** (FST) is a process that follows a rigid set of rules allowing testers and developers to repeat error conditions and fix issues, hopefully, in a timely manner. The main interest of FST is to ensure if a functionality is producing the expected output for given input.

**Performance Testing** is an important component of **Software Performance Engineering** (SPE) practices [1]. In contrast to FST, it is directed towards the non-functional aspects of systems, e.g., availability, security, reliability, responsiveness, among other attributes [6]. There are three major objectives to test the performance of an application: (i) determine the load intensity at which the system fails; (ii) discover bottlenecks that impairs operation and; (iii) perform capacity planning [1, 2]. The interest is to evaluate the **quality** of the product that is subjected for consideration. To test a given application in terms of performance, it is recommended to follow guidelines and principles according to a methodology. The methodology must relate to a precise objective, for instance, discover the major application bottleneck or assess the network impact on performance indices. It also describes the workload that must be characterized to assess the overall performance.

It is usual to set up the same environment for both FST and Performance Testing. Multi-tier applications, e.g., web servers, application servers and database servers, are deployed in the same environment, causing overloads that diminish the server’s original capacity. This excess of execution is attributed to the installation of multiple services with multiple workloads and executing profiles (e.g., CPU or IO bounded processes). The organizations choose a single server to act as the external accessible server, installing the firewall for security reasons. Therefore, the main reason for building such rigid and underachieving infrastructures is due to security concerns rather than performance. To test the performance of applications, one must be aware that external and unpredictable workloads will be present and eventually disturb the monitoring as all applications are sharing resources (processor time, memory, and so on).

Unfortunately, performance testing is expected to be carried out only by the conclusion of projects rather than simultaneously with the product development. Nevertheless, if development and performance teams somehow managed to work together, as recommended by experts and researchers, maybe the product would complete faster and with more quality [1]. An even more critical aspect to this is that the stakeholders already dispensed large sums of financial incentives to have a high quality software, ready to be used, responsive, and having reasonable resource allocation. Nevertheless, even if it runs in the very last phase of any given project, performance testing must be executed at least until resource usage and application’s response time have acceptable levels according to the design specification. More importantly, the application must conform with the predefined SLAs stipulated with the clients.

SLAs are high-level contracts established by stakeholders, e.g., service providers and customers. The main objective to set up an agreement stems from the need to guarantee that quality of service is present and ensured throughout a business relation. Defining SLAs between interested parties helps the understanding of responsibilities and conditions to deliver performant services. SLAs relates to non-functional application testing since it helps devising a compromise in terms of expected quality of service.

When applications are fully tested for both functional and non-functional specifications, they are ready to be deployed in a production environment. There is a huge research effort to characterize the behavior of applications and map to distributions in order to enhance the comprehension of how the system will behave under certain conditions, anticipating and efficiently reacting to problems. Next section discusses related works regarding stochastic models of multi-tier architectures and also some approaches where SLAs are under consideration.
2.1. Related works on modeling SLAs

SLAs were discussed in seminal works regarding Service Oriented Computing [7, 8, 9, 10]. Dealing with SLAs and cost models associated with contracts was researched by several authors in a recent past. Ashok et al. [11] investigated location-aware SLA contracts and quality of service measurements whereas Liu et al. [12] build a cost model to analyze the impact of SLA to maximize profit. Cost models for SLAs are a hot topic for research as it was investigated by Ardagna et al. [13], where the authors designed a resource allocation scheduler to study SLAs presenting heuristics as to how maximize the associated profits. SLAs and Queueing Networks modeling has also been used before to represent and analyze the effects of service deadlines in several domain applications. For instance, Abrahao et al. [14] devised a self-adaptive SLA capacity planner for Internet applications and Menascé et al. [15] discussed policies for managing web related resources for e-commerce servers using a Customer Behavior Model Graph. The technique used to extract performance indices was the development of a simulation model of an electronic bookstore as the main example. Ferrari et al. [16] used Queueing Networks (and also simulation for validation purposes) to model a tiered system comprising an Application Server and a Database Server. This work is closely related to the approach adopted in the present paper, where a simple stochastic model is presented and offer a fast approach to extract performance indices readily available to decision makers.

A more structured approach was presented by Clark et al. [17], using a formalism close to a Process Algebra known as Markovian Calculus as the main mechanism to study SLAs and quality of service. Clark and Gilmore [18] used Performance Evaluation Process Algebra (PEPA) to describe a stochastic model and then they converted it to a Petri Nets representation for analysis using a special set of compiling tools. The example considered an Automotive Crash Scenario where the deployment of certain car attributes (e.g., the air bag) in combination with actions to be taken was studied as well as the modeling of event durations with uncertainty data (an aspect of special interest to the present work). A Layered Queueing Network was proposed in Diao et al. [19] to model differentiated services in multi-tier web applications. Once each tier is evaluated, the authors proceed analyzing per-tier concurrency limits and cross-tier interactions. Finally, Sauvée et al. [20] proposed a method to reduce costs in IT realities by defining Service Level Objectives with examples in the context of multi-tier architectures.

2.2. Discussion

The research presented here distinguishes itself from related works from previous authors on the description and analysis of a stochastic model specially tailored for applications under performance testing subjected to external workloads and observance of SLAs. It is important to map the amount of external influence or unknown workloads to predict if the contract established by the service agreement will not be met. Case the time to complete jobs in the main load balance servers is taking a time that is superior to the threshold computed by our stochastic model, given the external load, it indicates that the SLAs will probably never be met. It is reasonable to consider that external influences are the main cause for the test fail, not because of some bottleneck problem. After some time or if the main server experiences less amount of loads, one could restart the Performance Testing process and resume SUT operation.

In fact, the bottleneck for such types of environments is directed towards the main server acting as a dispatcher that distributes the workload among the remaining servers. As stated before, after the transactions pass this server, they are executed in a clean and dedicated environment, where more reliable usage statistics are enabled. Thus, it is possible to populate a stochastic
model with parameters measured after the transaction passed the main server. One clear advantage of such stochastic models is the possibility of computing the average time necessary to process a given transaction, using the SLA deadline to calculate the available time that can be spent in the main server. An online monitoring tool could keep observing the process that are under execution in the main server and decide whether or not the SLA will be met. If a break of contract is imminent, one could stop the performance test and resume afterwards, to continue searching for application bottlenecks or some other performance testing objective.

3. Overall System Architecture and Operation

This section presents the architecture of the target application. This is a common setup for an IT infrastructure having Application Servers and Database Management Servers (DBMS). The SUT runs in a shared execution environment with external and unpredictable service demands. All computational resources (e.g., processor, disks, main memory, and network) are common in the environment. The DBMS was configured to behave as a Storage Area Network with high memory capacity (superior to one TByte).

The architecture is presented by Figure 1. It consists of a multi-tier architecture having web servers for presentation, application servers for running business logic and database servers for data storage, and a DBMS for query management. Transactions arrive with rate $\lambda$ in the first web server called PRF01, which sole purpose relies on routing them to one of two other web servers, named PRF02 and PRF03. Both PRF02 and PRF03 consult one of the available DBMS in the system, named DBMS01 and DBMS02. The SUT was implemented in Java and uses a pool of threads of fixed capacity and two pools of connections to the DBMS for performance reasons. The main distinction between server PRF01 and its counterparts PRF02 and PRF03 is the fact that this is the only machine accessible externally, e.g., there is also a firewall installed for security concerns. Because PRF01 is the only machine with a valid external IP address, every application is installed, e.g., applications that must submit to software failure analysis before its deployment in the production environment. For this reason, PRF01’s Processor Time is usually high due to unscheduled executions that systematically runs in the server in a daily basis.

Once the transaction is successfully processed by PRF01 and routed to PRF02 or PRF03, the execution becomes dedicated for the SUT. All three web servers run on a Pentium IV 2.66GHz machines with 16 GBytes of RAM, dualcore, running WebLogic 8.1 as application server and Windows 2003 Server Edition. The DBMS runs on a Pentium IV 3.2GHz, quadcore, with the same amount of RAM and running Linux RedHat Enterprise Edition and Oracle 10i with dedicated execution. Transactions that disrespects the SLA are stored for counting reasons (for subsequent quality assurance purposes) and exits the system. Ideally, no transaction ought to pass the higher limit of the agreement ensuring high QoS to the system and certainty that every transaction runs below specified thresholds.

Since PRF02 and PRF03 are dedicated, the best possible execution scenario is to enable transactions to bypass PRF01 and directly reach them without loosing time and disputing processing slices with every application present in the first server. However, due to security precautions this configuration is forbidden and PRF02 and PRF03 should remain hidden to the outside world, e.g., configured with only local IP addresses. This decision also makes it very hard to install new software, update the hardware or perform other configurations on the main machine. Moreover, the machine was located outside the main development site, making maintenance very difficult.

The measurements ruled out the network as an application bottleneck since every server is present in the same environment and the monitoring tool was able to attest that the system was
robust and healthy for the totality of the test plan execution. The critical aspect of the architecture is towards the PRF01 server since it encompasses every demand for every application that is running. Since the SLA must comply with a set of rules, our main strategy is to divide the system into two distinct blocks and analyze them separately, in order to isolate our problem into more manageable pieces.

The execution profile of the PRF01 machine showed that most of the time, its processor is running the Application Server and creating multiple instances of the Java Virtual Machine. Operating Systems naturally consume continuous (sometimes fixed) amounts of processing power. On our case, the servers are installed with the Windows 2003 and its operation does not compromise the access to resources (we are assuming that the machine is well configured and ready to execute processes). Occasionally, a process is loaded to memory and remains executing indefinitely. This can be mitigated by a soft reset (only a few processes are destroyed) or, more extremely, a hard reset (the whole machine is reinitialized). In production, the same problems related to server instabilities may or may not appear, depending on the execution profile. Since we are testing the application in the most problematic situation, our effort in this work will present a worst case scenario analysis.

3.1. Transaction lifecycle

Transactions in the SUT follow a specific pattern from creation until they are processed and stored. Customers positioned around the globe start the whole process under the form of issues that must be solved, e.g., repairments, questions, operation of devices, and so forth (in Figure 1, it corresponds to the Preprocessing Delay). All transactions are sent to a general purpose queue

Figure 1: Main architecture and details of the testing environment.
that follows a First Come First Served policy that translates them to a format understandable by the Main Application Server. The format uses an XML based structure with particular attributes such as general issue descriptions, time stamps, or customer identifier, to name a few. The next step for the transaction is to enter the queue of processes of the main application server, sharing execution and time with external processes from the point of view of the SUT. Once in the queue, operators accessing the system in other global locations are allowed to take ownership of the issue, taking an amount of time to either solve it or relaying it forward (also known as thinking time, associated with mouse clicks, click presses or general system operation). If this happens, the ownership is removed for that operator, and the transaction returns to the queue of transactions. Every interaction between operators and the system are accumulated to the overall time to solve the transaction, i.e., under the time limit imposed by the existing SLAs. The time operators take to process transactions are based on averages readily accessible in the application’s log files and available for us to use in our stochastic model.

3.2. Research opportunity

The architecture presented earlier (Figure 1) is common to many IT organizations. Evidently, SLAs must be met in production to avoid customer dissatisfaction or even legal problems. Performance Testing techniques are used to verify if SLAs are being respected, however, it can cloud the QoS assessment regarding the amount of time dispensed for each transaction. Due to this fact, applications having SLAs must be tested for performance in an environment that emulates production as closely as possible.

Our objective here is two-folded: firstly, we are profiting that testing environments have external demands just like happens in production, determining the conditions on which SLAs will not be met; and secondly, to enumerate the set of executing conditions that imbalance the time to process every transaction. As a positive side effect, the data available during testing will be used without loss of generality to parameterize our stochastic models.

The approach taken here innovates the way of mixing performance testing and stochastic modeling altogether, offering a way to match distinct problems in Software Performance Engineering: stochastic model parameterization, meeting SLAs in production and profiting of usual external workloads present in testing environments to study its influence on overall time to process transactions.

4. Stochastic Modeling and Stochastic Automata Networks

The objective of our model is to describe an IT infrastructure and compute, in average, the amount of time needed to process a transaction after it is routed by a server that executes important services (e.g. load balancing, firewall service, among other). The main characteristic of our environment is that we have a server that is overwhelmed processing various requests, acting as the only externally accessible machine. When transactions reach dedicated servers (just as presented in the architecture - Figure 1), they operate in full capacity, accessing resources located in the vicinity (without additional cost due to network exchanges). There is a deadline that a transaction must respect, defined between customers and service providers, measured since creation until it exits the system. The deadline is defined according to an agreed-upon SLA with the customer, and the main objective is to offer QoS and ensure that once a transaction is present in the system, it will have an associated time to completion, otherwise it will incur in losses (e.g. monetary).
4.1. Stochastic Models

For this particular problem, one could model the reality choosing standard Queueing Networks (QN) [21, 22]. QN is a very powerful formalism to model systems and extract performance indices such as average state permanence probabilities, transient behavior or scalability analysis [23]. It has been used in the past in many applications with significant results, from economic models to distributed computing. However, our reality implements a unique behavior to balance transaction routing that is hard, if not impossible, to model with QNs. The problem under analysis needs a more sophisticated manner to convey the fact that transactions are routed following patterns that must know the apparent load within each server.

For that matter, we took the decision to model the reality with a structured formalism based on Markov Chains [24, 25] named Stochastic Automata Networks (SAN) [26]. The reason to adopt such formalism stems from the fact that SAN allows easy definition of modular structures, also known as automaton, having states and transitions among states according to a list of events (one or more). Events can be defined as one of two types: local events, happening in the context of a single automaton; and synchronizing events, which needs to act accordingly to other automaton (or more) to be fired. Each event is mapped to a frequency of occurrence, termed a rate. Every rate in a SAN model is governed by a constant or a functional value [27, 28]. Constants are based on pure observations of the reality, whereas functional rates are dynamically computed based on the states of other automata’s states.

SAN is used to model realities where parallel and synchronizing behavior (resource sharing, for instance) is expected to occur. It is specially suited for distributed systems but can be applied to several application domains. It has been successfully used to extract useful performance indices of Global Software Development realities [29], Non-Uniform Memory Access architectures [30], Master/Slave parallel computing platforms [31] and Mobility patterns [32] to name a few. In a mathematical point of view, SAN uses Tensor Algebra properties to compute the probability vector that withholds the performance indices. It basically multiplies a vector by a non-trivial structure called a Markovian Descriptor, i.e., a list of small sized matrices that captures the occurrence of every event present in a given model. These matrices are operated with tensor sums for local events and tensor products for synchronizing behavior. One of the greatest advantages of using SAN to represent and solve stochastic models is due to its power of description and efficient storage mechanism.

![SAN example](image)

Figure 2: SAN example having local and synchronizing events with constant and functional rates.
Figure 2 shows an example of a SAN model. It has two automata with states, transitions, events and associated rates. The automata are named $AUT_1$ and $AUT_2$, respectively with states $I$, $J$, $K$ and $X$, $Y$. The cardinality of the state space set for this model is six. It is calculated by the product of local state spaces of each automaton, where the global states are $IX$, $IY$, $JX$, $JY$, $KX$, and $KY$. The transitions among states vary from each automaton, where a corresponding event is set and follow a given rate value. In this example, there are four events, one synchronizing ($s_1$) and three local ($l_1$, $l_2$, $l_3$), having constant rates and a functional rate for the event $l_1$. Due to the synchronization defined by the event $s_1$, the local states of both automata are changed simultaneously (from $K$ to $I$ in $AUT_1$, and from $Y$ to $X$ in $AUT_2$). Considering this global change in terms of the underlying Markov chain that is created for the model, the state combination of $KY$ changes to $IX$ with rate $r_1$. The functional rate verifies if the state of automaton $AUT_1$ is different of $I$. If this condition is true, the event can occur with a rate $r_2$ according to the function definition ($f$).

5. Numerical Analysis

This section presents our stochastic model and a numerical analysis to inspect the maximum amount of time that could be used on each phase (according to Figure 1). We begin our analysis presenting the results for the dedicated environment defined for Phase II because its behavior is more stable, in Section 5.1. In Section 5.2 we analyze Phase I in detail, where we calculate the response time of the main application server in observance to the SLA. The section also explains the model, its parameters and the numerical results. We finalize the numerical analysis discussing future model extensions in Section 5.3.

To model our reality we will assume a SLA for each transaction of 10 seconds. To our SUT, the SLA total time must be computed taking into account the time since the beginning of the processing until the transaction leaves the system. This time is divided by the time spent to preprocess the transaction ($T_{proc}$) plus the response time of Phase I ($T_{phaseI}$) and Phase II ($T_{phaseII}$). The value must be less than 10 seconds, otherwise the SLA will not be met. Our measured data on the preprocessing server accounted a time of 0.02 seconds for every transaction to be properly formatted to an XML definition and other modifications necessary to serve as valid input to the SUT. So, we have $T_{proc} = 0.02$ seconds, and the remaining available time to respect the SLA is equal to 9.98 seconds, distributed between Phases I and II.

5.1. Average time to completion analysis for Phase II

This section presents how we calculate the time needed to process the Phase II, i.e., the total response time for this phase ($T_{phaseII}$). The main idea is to profit from the isolation of the application and database servers in terms of execution. Once the transaction arrives in this phase, it is processed in dedication with full use of available processing power and memory. We are assuming the servers with a high level of stability, i.e., needing system restorations and management operations only occasionally.

Let $N$ be the average queue length, $X$ the throughput, $S$ the service time, $R$ the response time, and $U$ the utilization. To compute the values we used well established QN formulas available in the theory [21]. We used Little’s Law ($N = X \times R$), the Utilization Law ($U = X \times S$), average queue length derivation ($N = U / (1 - U)$) and Response Time ($R = S / (1 - U)$) which was derived from the previous formulas.

According to our performance testing data, the workload intensity is 50 Transactions Per Second (TPS), in average, corresponding to the global incoming transactions (Figure 1). The
chart in Figure 3 shows the average response time for different values of $U$ and $S$. For instance, if the utilization is equal to 90%, the best response time will be 1.1 seconds when the service time is equal to 0.12 seconds. However, as the time increases, it dramatically changes the response time from 1.1 seconds to the maximum value of 8.8 seconds per transaction, assuming the same utilization of 90%. This example shows the worst response time, i.e., for higher utilization values. The figure shows that if the utilization decreases, the response time follows the same pattern, e.g., approximately one second for utilizations of 10% or 20%.

In this paper, once the main application server routes transactions forward, we assume that on the worst case scenario a maximum theoretic value for the response time of precisely 8.8 seconds, i.e., we assume the worst case scenario, $T_{phaseII} = 8.8$ seconds. This value will be used to derive the final SLA time needed to complete the transaction.

The real existing problem to meet the SLA is set by the amount of time needed to route the transactions in our reality due to external workloads with somewhat unpredictable behavior. If no process is executing, i.e., just after a system reboot, every transaction is routed with full capacity. However, if some other process is running at the same time, the routing is impaired in direct proportion to the intensity of external applications that we must share resources with.

5.2. Maximum value for the response time of Phase I

We proceed our analysis on studying the response time for Phase I ($T_{phaseI}$). The model presented here was described by a SAN instead of a QN because our reality has several different behaviors that must be captured in terms of external influence modeling. Next, in Section 5.2.1 we explain our stochastic model in detail and its results are presented in Section 5.2.2.
5.2.1. Proposed model

We choose to model the transaction queue and several external processes that are either stopped or running within the application server. A depiction of our SAN based stochastic model is presented in Figure 4.

The number of automata in the model varies, depending on the desired number of external processes that will be created. We defined the main automaton, named PRF01, that represents the transaction queue to be processed by the main application server. This queue has \( K + 1 \) positions where \( K \) is the queue capacity, representing that at least \( K \) transactions could be saved to be served by PRF01. Event \( \text{arrival} \) indicates the number of incoming transactions that arrives in the system, with constant rate \( \lambda \). Event \( \text{service} \) sets the frequency on which transactions are processed, and it has a functional rate defined by \( f_s \). The service time of this queue is influenced by the number of existing external applications, \( i.e., \) its performance depends on the amount of work that needs to be processed by the main server.

Complementing the first automaton, we have created \( P \) other automata called \( \text{APP}_i \), where \( i \in 1..P \) of two states. \( P \) represents the amount of different types of applications present in the system for execution, having distinct times and patterns of execution (more CPU bound or more IO bound processes, for instance). Each application type has a \textit{weight} or a \textit{proportion} of influence applied to the service time of the main server (defined by \( \tau_i \)), slowing it down in this case. An automaton \( \text{APP} \) has two states: \( \text{Stop} \), indicating that the application has been stopped; and \( \text{Run} \) otherwise. There are two events present, named \( \text{start} \) and \( \text{stop} \) with rates respectively equal to \( \alpha \) and \( \beta \). When either event is triggered, the application starts or stops its execution under different rates depending on the case.

<table>
<thead>
<tr>
<th>Function</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f_s )</td>
<td>( \text{service time} \times (1 + f_1 + \ldots + f_P) )</td>
</tr>
<tr>
<td>( f_i )</td>
<td>( (\text{state } \text{APP}_i = \text{Run}) \times \tau_i )</td>
</tr>
<tr>
<td>( f_P )</td>
<td>( (\text{state } \text{APP}_P = \text{Run}) \times \tau_P )</td>
</tr>
</tbody>
</table>
As a final remark, we are abstracting the fact that the SUT has two cores of execution, modeling our main queue with a single server to process every transaction. Our proposed model focuses on the external execution aspect of applications, and its influence on the QoS requirements that must be enforced by the SLA.

5.2.2. Model results

Stochastic Automata Networks are solved using a software package called GTAexpress [33]. On the core of its implementation, the tool is equipped with the latest version of the Vector-Descriptor Product algorithm called Split [34], which depending on the model to be solved, accelerates convergence and allows modelers to assess performance faster. GTAexpress is also under new developments such as the addition of Multi-valued Decision Diagrams [35] and Perfect Simulation [36] techniques for state spaces that exceeds 65 million of states, i.e., the current limit of the tool for machines with 4 GBytes of RAM.

To sum up the time spent until now, we have $T_{\text{proc}} = 0.02$ seconds spent in preprocessing, $T_{\text{phaseII}} = 8.8$ seconds to run the business logic of our application in a dedicated environment at most and 10 seconds to comply with the SLA. So, it remains 1.18 seconds to use in Phase I and still guarantee that the system will meet the time requirements, i.e., $T_{\text{phaseI}}$ must be inferior to 1.18 seconds.

We conducted several performance tests and monitored important resources such as Processor Time and Available Memory (only to name a few) for all application servers and the database servers. We used LoadRunner as the main tool to perform the load testing procedure on our SUT, where the verified throughput ($X$) was 50 TPS. Following a methodology, our performance testing objective was to verify that at least 90% of the total number of transactions were being met by the SLA.

We have measured that the main server (PRF01) had, in average, an utilization of 75% for the workload proposed in the performance test plan. Having the throughput ($X = 50$ TPS) and the utilization ($U = 0.75$), we used the Utilization Law to compute the service time $S$ for this application. The value that we have obtained was 0.015 seconds to execute a transaction. It is worth mentioning that we ascertain that no external execution was present in PRF01 in order to determine the utilization, i.e., the service time was not contaminated by external processes (when none external applications are present in PRF01, then all $\tau$ are equal to zero).

Using this service time of 0.015 seconds in function $f_s$ of our model, we have a service rate of 66.67 TPS for an arrival rate of 50 TPS (assuming balance in the system, i.e., the arrival rate equals the throughput). The average number of processes in the queue is $N = 3$ and the response time is $R = 0.06$ seconds, which is quite low and would ever respect the remaining possible time to ensure the SLA, calculated in 1.18 seconds (an order of almost 20 times as low as the possible value). However, this is also quite unrealistic to our testing environment, where it is almost impossible to test the system without interference. It is only natural to have multiple sources of interference, impairing the service time and thus increasing the response time.

To estimate what is this influence in the overall performance of our SUT we need to assign external workload profiles from different application types that are executed in the machine (also termed application profiles). We will assume $P = 5$, e.g., five types of different applications such as scientific software, service daemons, firewall server, or other web services. We also choose to define fixed applications that corresponds to the machines’ native operational system that is running continuously and constantly using computational resources (or any other application that endlessly runs within the testing environment).
Table 1 shows the average time that each application is under execution on PRF01 in a single day. Note that APP3 represents the operating system of the machine, and we measured that, in the worst case in a one day time frame, at least one 10 minute full reboot happens. These data were obtained through average usage monitorings that was done prior and it was run according to a measurement methodology adapted from [2]. We used the usual performance counters (% Processor Time, Available Memory) for the Windows machines and a combination of monitoring scripts (e.g. vmstat, iostat, etc.) for the ones executing Linux. We have created several scripts to convert log files from the measurements information. We combine this file with the one created by the Windows machines, which allowed us to analyze the average values for each time frame of each experiment.

<table>
<thead>
<tr>
<th>APP1</th>
<th>APP2</th>
<th>APP3</th>
<th>APP4</th>
<th>APP5</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 hours</td>
<td>18 hours</td>
<td>20 hours</td>
<td>5 hours</td>
<td>23:50 hours</td>
</tr>
</tbody>
</table>

Table 2 shows the \( \tau \) configured for each application. It basically states the weight associated to every external application that influences the service time of the main server. This factor is important to our model since it relates to the amount of external work that must be done by the server, usually having variable processing demands.

### Table 2: Different execution scenarios based on application profiles.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>( \tau_1 )</th>
<th>( \tau_2 )</th>
<th>( \tau_3 )</th>
<th>( \tau_4 )</th>
<th>( \tau_5 )</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17%</td>
<td>13%</td>
<td>6%</td>
<td>4%</td>
<td>5%</td>
<td>45%</td>
</tr>
<tr>
<td>2</td>
<td>4%</td>
<td>2%</td>
<td>33%</td>
<td>1%</td>
<td>5%</td>
<td>45%</td>
</tr>
<tr>
<td>3</td>
<td>9%</td>
<td>16%</td>
<td>6%</td>
<td>14%</td>
<td>5%</td>
<td>50%</td>
</tr>
<tr>
<td>4</td>
<td>11%</td>
<td>3%</td>
<td>27%</td>
<td>4%</td>
<td>5%</td>
<td>50%</td>
</tr>
<tr>
<td>5</td>
<td>3%</td>
<td>19%</td>
<td>1%</td>
<td>32%</td>
<td>5%</td>
<td>60%</td>
</tr>
<tr>
<td>6</td>
<td>19%</td>
<td>4%</td>
<td>21%</td>
<td>11%</td>
<td>5%</td>
<td>60%</td>
</tr>
<tr>
<td>7</td>
<td>2%</td>
<td>11%</td>
<td>3%</td>
<td>49%</td>
<td>5%</td>
<td>70%</td>
</tr>
<tr>
<td>8</td>
<td>22%</td>
<td>17%</td>
<td>14%</td>
<td>12%</td>
<td>5%</td>
<td>70%</td>
</tr>
<tr>
<td>9</td>
<td>3%</td>
<td>12%</td>
<td>8%</td>
<td>52%</td>
<td>5%</td>
<td>80%</td>
</tr>
<tr>
<td>10</td>
<td>33%</td>
<td>2%</td>
<td>37%</td>
<td>3%</td>
<td>5%</td>
<td>80%</td>
</tr>
<tr>
<td>11</td>
<td>6%</td>
<td>8%</td>
<td>4%</td>
<td>67%</td>
<td>5%</td>
<td>90%</td>
</tr>
<tr>
<td>12</td>
<td>36%</td>
<td>28%</td>
<td>17%</td>
<td>4%</td>
<td>5%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Next, in Table 3 we show our main performance indices in terms of the utilization, the average number of transactions in the queue, throughput and response time for the scenarios explained in Table 2.

The results show that the response time extracted from the model is varied and directly proportional to the workload intensity of the applications. However, it is interesting to verify that not necessarily decreased load levels impacts on equally decreased response times. For instance, in Scenarios 1 and 2, the total time added in the application’s service time is up to 45%. The time spent by the external applications to execute presents different response times for every scenario. As an example of this behavior, Scenario 1 had a response time of 0.43 seconds, which is less...
than the available time to meet the SLA (estimated as 0.7 seconds), contrary to what is observed by the results showed by Scenario 2 that spent 2.75 seconds. The increase in time was due to the variation in terms of average queue length as well as a throughput decrease that happened accordingly. For these cases the arrival rate is greater than the service rate, bringing congestion, causing delays and queueing in the system. The same situation is verified in Scenarios 4, 6, 8 and 12, where the workload has impacted the overall response time under different circumstances.

For the rest of our analyzed case scenarios where we varied the workload intensity, we were able to compute response times within the available SLA time, i.e., cases where the external application influence over service time enables meeting the time constraint. Our stochastic model was able to help us understand the fact that depending on the application execution profile we can anticipate if the response time for our SUT will fall under the SLA threshold. This is very interesting because the model could forecast, for instance, the unfavorable or advantageous conditions present on the environment to allow the execution of performance testing. Case such conditions are causing external delays, it is safe to conclude that it is not our SUT that have a problem, maybe our executing environment is not yet suitable to perform accordingly.

5.3. Model extensions discussion

Our model could be extended to compose other behavior arising in multi-tier architectures. For instance, one could consider studying the effect on end-to-end response time when the external applications are observed to have workload burstiness, i.e., several transactions arriving almost instantaneously and long periods of idle time. We are aware of an efficient overload management in multi-tier environments having bursty workloads recently studied by Lu et al. [37]. In our model, this issue can be adapted by setting the parameters Stop-Run of the external loads accordingly. The analysis would show how the burstiness affects performance and SLA requirements.

Another interesting aspect to be further inspected concerns extracting performance indices when the external applications are executing stress testing within several contexts. During stress
test, the limits for a given application are tested and usually it increases the amount of existing resource sharing and also the rate of failures (or even faults). Our model could define different application testing profiles to inspect its relation to the performance testing of our SUT.

Since we have half of our system under dedicated operation, we could adapt our environment to receive incoming transactions from external workloads as well. Then, we could assess and analyze the sharing resources (in this case, every server have shared execution) and the impact to meet the contract stipulated by the SLA.

6. Final Considerations

The present work proposed stochastic models and SLA assurance applied in the context of multi-tier web services with external workloads. There is an increasing interest for practical applications of stochastic modeling for performance evaluation. We modeled the reality and performed a worst case scenario analysis to verify if the contracts between users and service providers were being respected. Our performance indices computations presented means to decide if the SLA would be executed below its deadline and the impact of external workloads in this time.

The IT multi-tier infrastructure modeled in this work shares resemblance with many real web services deployed throughout the world. Many organizations use SLAs in their business operations to ensure high quality of service to their customers. The present paper proposed a stochastic model to represent and evaluate such architectures and studied the influence of external workloads to meet SLAs. The model was parameterized with data obtained from a testing environment, where performance testing processes were being executed with unpredictable workloads caused by the presence of external services. As mentioned before, a good side effect of this is that the same model could be adapted and parameterized with data obtained from a production environment. This will enable the verification of SLAs in customer side applications, assessing overall quality of service that is being provided for.

As future works, we consider applying the same model ideas for busy environments with more intense workload variation. Such work may demand a deeper analysis of stochastic distributions leading to the extensions of the model to consider more complex distributions. One option is the inclusion of phase-type transitions to approach some non-exponential phenomena like timeouts. In this case, some prior studies on phase-type representation for SAN formalism [38] could be used.

In another interesting future work we could also develop and install a daemon with both a stochastic model and a numerical solver (e.g., GTAexpress [33] or similar) in the application servers of interest to monitor the execution and self-parameterize a model with the obtained data to decide whether or not the SLA will be met in a timely fashion. This research will allow decision makers to stop the execution of some external workloads or to control the incidence of external applications that are allowed to run. This will undoubtedly help to assure a higher level of availability to the web service, avoiding economical losses and ensuring user satisfaction.

Acknowledgments

The authors thank Dr. Alberto Avritzer from Siemens Corporate Research (SCR), Princeton/NJ/USA, for insightful discussions that led to the elaboration of this paper and the performance testing research team of the Computer Science Department at Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre/RS/Brazil.
References


Stochastic Modelling and Optimisation of Internet Auction Processes

Timothy L. Y. Leung¹

Department of Computing
Imperial College London
London, U.K.

William J. Knottenbelt²

Reader in Applied Performance Modelling
Department of Computing
Imperial College London
London, U.K.

Abstract

Internet auctions are an attractive mechanism for the exchange of goods at a non-fixed price point. The operation of these auctions can be run under a variety of parameters. In this paper, we provide a theoretical analysis of fixed time forward auctions in cases where a single bid or multiple bids are accepted in a single auction. A comparison of the economic benefits and the corresponding buyer and seller surpluses between the auctions where a single bid is accepted and the auctions where multiple bids are accepted is made. These models are verified through systematic simulation experiments, based on a series of operational assumptions, which characterise the arrival rate of bids, as well as the distribution from which the private values of buyers are sampled. Decision rules for optimising surplus under different auction fee structures are also given.

Keywords: Online Auctions, Internet Auctions, Auction Income, Auction Duration, Multiple Bids.

1 Introduction and Related Work

The early days of E-Commerce saw the B2C (Business to Consumer) model as dominant, however, the current trend is increasingly embracing the C2C (Consumer to Consumer) model. Thus, instead of companies selling items to

¹ Email: timothy.leung08@imperial.ac.uk
² Email: wjk@doc.ic.ac.uk

©2011 Published by Elsevier Science B. V.
the consumers, consumers are selling items to fellow consumers on a global-scale, and a common mechanism of achieving this is to use online auctions. Unlike conventional auctions, Internet auction mechanisms have the scope of incorporating procedures of greater complexity and can take on a wide variety of forms.

A few features of Internet auctions can be attributed to the sheer size and volume of the buyer and seller markets. This allows for a variety of different auctioning mechanisms, e.g. timeshift auctions [1] and penny auctions, and yet still have enough liquidity and market activity to attract the interest of both buyers and sellers. In this sense, conventional auctions depend on getting the right people to the auctions in order to obtain an accurate valuation of the good and service sold — people that are interested and willing to buy. Internet auctions take the other side and depend on the large amount of traffic generated on the site in such a way that with the number of items sold and the amount of potential buyers browsing the site, they are bound to see something they are interested in.

The online auction website, eBay, is a popular and recent implementation of the auction mechanism. It is classed as a consumer-to-consumer (C2C) auction and it runs open-bid second-price auctions that are of a fixed length. Due to its fixed length, eBay auctions are susceptible to sniping, which see bidders submit their bids moments before the close of an auction preventing other bidders from submitting counter-bids [2]. While this is seen to be problematic, eBay has always maintained the policy that a bidder should bid his private value. Since the winner pays the second price, there is little reason for a bidder to shade his bid. In order to counteract sniping, other online auction websites, such as Amazon, have employed auctions with a soft close, automatically extending the length of the auction. The investigation of different types of auction terminations has been undertaken in [3], where it is found that late bidders of eBay type of hard close tend to be associated with highly experienced bidders, whereas those of Amazon type tend to be relatively inexperienced bidders. In [4], it is found that sniping often leads to winning, and it observes that many sellers tend to set the starting bid price unrealistically low to stimulate bidder participation.

Studies of Internet auction bidding behaviour have been undertaken in [3,2,5] In [5], different properties of online auctions such as consumer surplus, sniping, bidding strategy and their interactions are studied, where a significant correlation between sniping and surplus ratios is found. It also examines the efficiency of online auctions, where Pareto efficiency is used as the optimality criterion. In [2], it is suggested that the strategic advantages of sniping are eliminated or severely eroded in auction mechanisms that apply an auction extension rule, and that there is noticeable difference between sniping on eBay and Amazon in proportion to user experience.
Experimental studies of Internet auction behaviour have been undertaken in [6,7,8]. In [8], it concentrates on the Dutch auction and first-price sealed bid auction formats, using laboratory experiments and human subjects, where values are drawn from the uniform distribution between 0 and 100, focusing primarily on the effect of clock speed on sellers revenue. In [7], laboratory experiments with human subjects are also conducted, and the operational efficiency of Internet auctions is studied. Collusion behaviour such as shilling, in which the seller plays a part in the bidding process, is studied in [9], where two types of shilling strategies are examined, which deploys competitive bidding and reserve price mechanism and each of these exhibits a characteristic pattern of behaviour. While an auction can be defined as a market institution whereby offers are made only by the buyers, i.e. bids, or only by the sellers, i.e. asks, a double auction is one where both buyers and sellers are able to make offers [10]. Viewing the interlinking relationship between bidders and sellers as networks is proposed in [11], and the competitions in auctions is investigated in [12]. Price variation characteristics and consumer surplus are studied in [13,14]. The use of various types of curves for fitting price data for Internet auctions have been proposed in [15], in which monotone splines and beta functions are used. Empirical investigations of eBay auctions have also been undertaken in [16] where the auction of coins is conducted. It makes use of regression models to estimate the price of items and examines the influence of seller ratings (which measures the reliability and services provided by the seller) on the final price. It has also found that the effect of positive and negative ratings is not symmetrical, with the latter having a much greater (adverse) influence on the price. It also suggests that longer auctions tend to have a beneficial effect in achieving a higher price. Moreover, in [17], it has found that there is a reluctance on the part of users to give negative feedbacks compared with giving positive feedbacks. The K-means clustering algorithm has been employed in [18] which classifies bidders into five categories based on factors such as entry time, number of bids placed, and exit time. It also examines the use of automated agents in carrying out bidding as well as the different experience levels of bidders. The use of analogies from physics to study price movements have been applied in [15,19,20], which make use of the concepts of price-velocity to characterise the dynamics of price changes and may subsequently be exploited to produce forecasts.

In this paper, we describe the economic benefits to buyers and sellers when there are multiple identical lots available for sale and when multiple bids are accepted in an auction. In Section 2, a stochastic framework for analysing Internet auctions is described, which is then extended to include multiple accepted bids from an auction. Section 3 introduces the concept of allocative efficiency, using the buyer and seller surpluses to compare the performance of a multiple accepted bid auction with that of a singularly accepted bid auction.
Finally, these theoretical findings are verified by way of simulation in Section 4, and the conclusions presented in Section 5.

2 Basic Stochastic Model of Internet Auctions

The Independent Private Values model is often associated with auctions [21]. The characteristics of this model include the assumptions of privacy and independence where the value of the commodity in question is private to the individual buyers, and that different buyers do not know the values other buyers attached to the commodity. In addition, these values are drawn from a common distribution which is known to the buyers. In probabilistic terms, this essentially amounts to a series of values which are independent and identically distributed. A common distribution used is the uniform distribution [8]. In our subsequently analysis, we shall follow the independent private values model using the uniform distribution. Here we assume the bids \( \{Q_k\} \) to be ascending ordered values taken from the uniform distribution over the interval \((0, L)\), and that, as in [22], we assume that the bids arrive over time in a Poisson manner with rate \( \lambda \).

Internet auction mechanisms take on a variety of forms. Many of them would be forward auctions (e.g. eBay.com), such as English or Vickrey auctions, while others may be reverse auctions (e.g. oltiby.com). As indicated above, a key difference between Internet and conventional auctions is that the former tend to be of fixed duration, while ending an auction at a pre-specified time point tends to be unacceptable for the latter. Usually only one bid is accepted per auction, but sometimes there are auctions that allow more than one bid being accepted (e.g. Google’s ad auctions).

If there are \( N \) bids, we denote these ordered values by \( Q_{(1)} < Q_{(2)} < \ldots < Q_{(N)} \). A forward auction is an electronic auction where buyers compete for items or services, with the price going up over time, and the items or services for sale are displayed and specified in a particular website (e.g. uBid.com). In the present model, we assume that the auction time is fixed with duration \( T \). Let \( N \) be the number of bids received, and the largest bid \( Q_{(N)} \) received over the time interval \((0, T)\) is accepted. A high value for \( T \) will produce a larger average accepted bid but the auction duration will be longer. For practical meaningful auction operation, \( T \) should be significantly greater than the mean bid inter-arrival time \( 1/\lambda \), i.e. \( T \gg 1/\lambda \). At the close of the auction, the auctioning mechanism will select the maximum bid \( Q_{(N)} \) to be accepted. The exact algorithm is shown in Figure 1.

From the results of order statistics, it can be shown [23] that the conditional income per auction is

\[
E[Q_{(N)}|\text{Number of bids} = N] = \frac{NL}{N+1}, \tag{1}
\]
begin
L = 0;
accept_id = null;
while clock < T do
begin
for an arriving bid of magnitude R,
if L < R, then do
begin
L = R;
accept_id = bidder_id;
end;
end;
return bid L offered by accept_id;
end;

Fig. 1. Fixed time forward auction.

Since
\[ \frac{dE[Q(N)]}{dN} \frac{\text{Number of bids} = N}{(N+1)^2} = \frac{L}{N!} > 0, \]  
we see that, as the number of bids \( N \) increases, the corresponding average income per auction will also increase. To determine the average income \( E[Q(N)] \), we remove the condition on \( N \) in Equation 1 using the Poisson probabilities; i.e.

\[
E[Q(N)] = \sum_{N=1}^{\infty} NL \frac{e^{-\lambda T}(\lambda T)^N}{N(N+1)}
= \sum_{N=1}^{\infty} \left[ L - \frac{L}{N+1} N \right] \frac{e^{-\lambda T}(\lambda T)^N}{N!}
= \sum_{N=1}^{\infty} L \frac{e^{-\lambda T}(\lambda T)^N}{N!} - \sum_{N=1}^{\infty} \frac{L}{N+1} \frac{e^{-\lambda T}(\lambda T)^N}{N!}
= L \left(1 - e^{-\lambda T} \right) - \frac{Le^{-\lambda T}}{\lambda T} \sum_{N=1}^{\infty} \frac{(\lambda T)^{N+1}}{(N+1)!}
= L \left(1 - e^{-\lambda T} \right) - \frac{Le^{-\lambda T}}{\lambda T} \left(e^{\lambda T} - 1 - \lambda T \right).
\]  

We omit the term \( N = 0 \) in the above, since when \( N = 0 \), the income will
be zero. This gives an average income per auction of

\[ E[Q(N)] = \frac{L}{\lambda T} \left( \lambda T + e^{-\lambda T} - 1 \right), \]  

(4)

and an income rate, i.e. income per unit time, of

\[ \frac{L}{\lambda T^2} \left( \lambda T + e^{-\lambda T} - 1 \right). \]  

(5)

Figure 2 shows \( E[Q(N)] \) for different values of \( \lambda \) for \( L = 100 \), and \( T = 10 \). We see that the increase in bid rate up to \( \lambda = 4 \) produces rather steep average auction income improvement. There seems to be a critical bid rate at around \( \lambda = 6 \), above which the improvement in income becomes less pronounced.

![Fig. 2. Auction income as a function of the bid rate.](image)

Figure 3 shows the rate of income for \( \lambda = 1 \) and \( L = 100 \) for different values of the auction duration \( T \). We see that the income rate favours short duration auctions, which tends to stabilise for \( T > 10 \), beyond which varying the auction duration will not bring about any significant change in the income rate. Figure 4 shows the expected transaction price for \( \lambda = 1, L = 100 \) for different values of the auction duration \( T \). We see that it follows much the same shape as Figure 2.

Indeed, since both \( \lambda \) and \( T \) occur together in Equation 4, this is what we would expect. Letting \( z = \lambda T \), we have

\[ E[Q_N] = \frac{L}{z} \left( z + e^{-z} - 1 \right), \]  

(6)
Differentiating with respect to $z$, we have

$$\frac{dE}{dz}[Q_{(N)}] = \frac{L}{z^2} \left( 1 - e^{-z} - ze^{-z} \right) \quad (7)$$

$$= \frac{L}{z^2} \left[ 1 - \frac{1 + z}{e^z} \right]. \quad (8)$$

Since $z > 0$, and

$$e^z = 1 + z + \frac{z^2}{2!} + \ldots, \quad (9)$$
we have \((1 + z) < e^z\), so that \((1 + z)/e^z < 1\), and thus

\[
\frac{dE[Q(N)]}{dz} = \frac{L}{z^2} \left[ 1 - \frac{1 + z}{e^z} \right] > 0. \tag{10}
\]

Consequently, we have

\[
\frac{dE[Q(N)]}{d\lambda} = \frac{dE[Q(N)]}{dz} \times \frac{dz}{d\lambda} = T \times \frac{dE[Q(N)]}{dz} > 0. \tag{11}
\]

Likewise, we have

\[
\frac{dE[Q(N)]}{dT} = \lambda \times \frac{dE[Q(N)]}{dz} > 0. \tag{12}
\]

Thus, the transaction price can be increased by either increasing the bid rate or the auction duration. As we shall see later, since the auction duration is controlled by the seller, he/she can use the auction duration as a mechanism for raising his/her surplus.

It is interesting to compare the auction income using an approximation based on Equation 1. If we remove the condition on \(N\) by simply replacing \(N\) by its average (from the Poisson distribution) of \(z = \lambda T\), we have approximately

\[
E[Q(N)] = \frac{Lz}{z + 1}. \tag{14}
\]

Figure 5 compares the average auction income from Equations 6 and 14 for \(L = 100\) for different values of \(z = \lambda T\), and we note that for \(\lambda = 1\), we have \(z = T\). We see that the approximation is quite good for moderate to large values of \(z\). For very large values of \(z \gg 1\), the exact formula and the approximation are virtually indistinguishable, and we shall be making use of this approximation to obtain closed-form solutions in the surplus analysis below.

3 Surplus Analysis

The concept of allocative efficiency (or sometimes called operational efficiency) is often employed to evaluate how Internet auctions perform [7]. The seller surplus is the difference between the transaction price and the seller’s costs (sometimes generically called production cost), while the difference between the buyer’s value and the transaction price gives the buyer surplus or consumer surplus (see Figure 6). The total surplus is the seller surplus plus the
buyer surplus, and the allocative efficiency is given by the total actual realised surplus expressed as a fraction of the total possible surplus [7]. For simplicity, these quantities are represented in Figure 6 as linear functions, but the ideas remain the same if one or more of these are non-linear.

From the buyers’ point of view, their valuation of the auction item is indicated by the maximum price $L$ that they are willing to pay. Thus the buyer surplus $\beta$ is given by the difference in buyer valuation and the transaction
price, which is

\begin{align}
\beta &= L - \frac{L}{\lambda T} (\lambda T + e^{-\lambda T} - 1) \\
&= \frac{L}{\lambda T} (1 - e^{-\lambda T}) \\
&= \frac{L}{z} (1 - e^{-z}) > 0.
\end{align}

(15) \quad (16) \quad (17) \quad (18)

Since \( z = \lambda T \) is the average number of bids, the above can be interpreted as follows: the consumer surplus is the private value evenly divided by the number of bids times the probability of having a non-empty auction (i.e. an auction where there is at least one bid). We also see that the higher the value of \( L \), the greater is the buyer surplus. On the other hand, bidder collusion behaviour — where bidders collude in order to lower the transaction price — may be incorporated by having a lower value of \( L \). Shilling behaviour — where the seller artificially inflates the transaction price through disguising as bidders — may be reflected by a higher value for \( \lambda \), and hence a higher value for \( z = \lambda T \), which from Equation 10, will result in a higher transaction price and consequently reducing the buyer surplus while raising the seller surplus.

From the seller’s point of view, the aim of auction is to attain improvements in seller surplus through expending more time to achieve a higher price or income. If one simply accepts the first bid that comes along, then its average magnitude is \( E[Q_i] = L/2 \). By holding an auction, the average gain in surplus per bid acceptance due to one auction, assuming there are \( N \) bids, is

\[
E[Q_{(N)}|N] - E[Q_i|N] = \frac{LN}{N+1} - \frac{L}{2}
\]

(19) \quad (20)

If there are two or more identical items for sale, to speed things up one might accept the two highest bids, instead of the just the highest one. Accepting more than one bid per auction is quite common in Internet auctions; e.g. Googles ad auctions often accept several bids. From (Barry C. Arnold et. al. 2008), it is shown that the kth order statistic of N samples from a uniform distribution distributed over a given interval is \( k/(N+1) \) of the length of the interval. Thus, in accepting the two highest bids in one auction, the average
gain in seller surplus per bid acceptance is

\[
\frac{1}{2} \left\{ \left( E \left[ Q_{(N)} \mid N \right] - E \left[ Q_i \mid N \right] \right) + \left( E \left[ Q_{(N-1)} \mid N \right] - E \left[ Q_i \mid N \right] \right) \right\} \\
= \frac{1}{2} \left( E \left[ Q_{(N)} \mid N \right] - E \left[ Q_{(N-1)} \mid N \right] \right) - E \left[ Q_i \mid N \right] \\
= \frac{1}{2} \left[ \frac{LN}{N+1} + \frac{N-1}{N+1} \right] - \frac{L}{2} \\
= \frac{(N-2)L}{2(N+1)},
\]

which is less than the gain in seller surplus in the case where only one bid is accepted, but it takes only one instead of two auction times and associated costs to achieve two acceptances. Correspondingly, the average buyer surplus will increase, since the transaction price of the second item is lower.

In general, the average of the highest \(K\) bids, given there are \(N\) bids, is

\[
\frac{1}{K} \sum_{j=0}^{K-1} E \left[ Q_{(N-j)} \mid N \right] = \frac{L}{K} \sum_{j=0}^{K-1} \frac{N-j}{N+1} \\
= \frac{(2N-K+1)L}{2(N+1)}.
\]

Thus, the conditional average gain in surplus per acceptance is

\[
\frac{1}{K} \sum_{j=0}^{K-1} \left\{ E \left[ Q_{(N-j)} \mid N \right] - E \left[ Q_i \mid N \right] \right\} \\
= \frac{1}{K} \left\{ \sum_{j=0}^{K-1} E \left[ Q_{(N-j)} \mid N \right] \right\} - E \left[ Q_i \mid N \right] \\
= \frac{1}{K} \sum_{j=0}^{K-1} E \left[ Q_{(j)} \mid N \right] - \frac{L}{2} \\
= \frac{(2N-K+1)L}{2(N+1)} - \frac{L}{2} \\
= \frac{(N-K)L}{2(N+1)},
\]

which from Equation 20 is always below the gain in surplus resulting from accepting a single bid per auction. The average total income in accepting the top \(K\) bids, from Equation 26, is

\[
\sum_{j=0}^{K-1} E \left[ Q_{(N-j)} \mid N \right] = \frac{(2N-K+1)LK}{2(N+1)}.
\]
Consider a variation of the basic model, in which the $K$ highest bids are accepted in one auction. Note that accepting $K$ highest bids requires that there are at least $K$ arrivals (and of course at least $K$ items for sale), and for meaningful operation, this requires that $T \gg K \times \text{Mean Inter-arrival Time}$ or $\lambda T \gg K$. Removing the condition on $N$, and noting that $N \geq K$, we have, for the average total income in accepting the top $K$ bids,

$$
\sum_{j=0}^{K-1} E[Q_{(N-j)}|N] = \sum_{N=K}^{\infty} \frac{L(K+1)N!}{2(N+1)} \frac{e^{-\lambda T}(\lambda T)^N}{N!}
$$

(33)

$$
= \frac{L}{2} \sum_{N=K}^{\infty} \left[ 2 - \frac{K+1}{N+1} \right] \frac{e^{-\lambda T}(\lambda T)^N}{N!}
$$

(34)

$$
= LK \sum_{N=K}^{\infty} \frac{e^{-\lambda T}(\lambda T)^N}{N!} - \frac{LK}{2\lambda T} \sum_{N=K}^{\infty} \left[ \frac{K+1}{N+1} \right] \frac{e^{-\lambda T}(\lambda T)^{N+1}}{N!}
$$

(35)

$$
= LK \sum_{N=K}^{\infty} \frac{e^{-\lambda T}(\lambda T)^N}{N!} - \frac{LK(K+1)}{2\lambda T} \sum_{N=K}^{\infty} \frac{e^{-\lambda T}(\lambda T)^{N+1}}{N!}
$$

(36)

$$
= LK \left\{ 1 - \sum_{j=0}^{K-1} \frac{e^{-\lambda T}(\lambda T)^j}{j!} \right\} - \frac{LK(K+1)}{2\lambda T} \left\{ 1 - \sum_{j=0}^{K} \frac{e^{-\lambda T}(\lambda T)^j}{j!} \right\}
$$

(37)

That is, we have for the expected income $I_K$ when we choose to accept $K$ top bids in a single auction

$$
I_K = LK \left\{ 1 - \sum_{j=0}^{K-1} \frac{e^{-\lambda T}(\lambda T)^j}{j!} \right\} - \frac{LK(K+1)}{2\lambda T} \left\{ 1 - \sum_{j=0}^{K} \frac{e^{-\lambda T}(\lambda T)^j}{j!} \right\}
$$

(38)

We see that for $K = 1$, the above reduces to Equation 5, and for the important special case $K = 2$, we have

$$
I_2 = \frac{L}{\lambda T} \left( 2\lambda T + \lambda T e^{-\lambda T} - \frac{\lambda T)^2 e^{-\lambda T}}{2} + 3e^{-\lambda T} - 3 \right)
$$

(39)

$$
I_2 = \frac{L}{\lambda T} \left( 2\lambda T + \lambda T e^{-\lambda T} - \frac{\lambda T)^2 e^{-\lambda T}}{2} + 3e^{-\lambda T} - 3 \right).
$$

(40)

Consider the cost $\Omega$ of holding an auction, which may be related to the auction time and associated costs such as fees paid to the auction site, and payments to financial intermediaries. We assume that these costs are otherwise not incurred if the item is sold through other channels. Let the price of a unit of the good be $C$. If no auctions are held, then the expected seller surplus would simply be $(L/2 - C)$, where, as indicated from the arguments above, $L/2$ represents the average value of the first offer, and we assume that it will
be accepted. By holding an auction, the seller surplus $S$ becomes

$$S = \frac{L}{\lambda T} \left( \lambda T + e^{-\lambda T} - 1 \right) - (C + \Omega). \quad (41)$$

Thus, the break-even point of holding an auction is given by the improvement in surplus offset by the auction cost

$$\frac{L}{\lambda T} \left( \lambda T + e^{-\lambda T} - 1 \right) - \frac{L}{2} = \Omega. \quad (42)$$

As shown in Equation 13 above, the longer the auction, the higher the expected income. Supposing one wishes to attain a certain level of seller surplus $S_0$, then the minimal auction duration $T^*$ is given by the solution to the following equation

$$S_0 = \frac{L}{\lambda T^*} \left( \lambda T^* + e^{-\lambda T^*} - 1 \right) - (C + \Omega). \quad (43)$$

While we may solve for the above using numerical methods, we may obtain closed-form solutions by using the approximation from the previous section. Using this approximation, the above becomes

$$S_0 = \frac{Lz'}{z' + 1} - (C + \Omega), \quad (44)$$

giving

$$z' = \frac{S_0 + C + \Omega}{L - (S_0 + C + \Omega)}. \quad (45)$$

and this will provide a reasonable approximation for $z' \gg 1$. Thus, the approximate optimal auction duration $T'$ is

$$T' = \frac{S_0 + C + \Omega}{\lambda \left( L - (S_0 + C + \Omega) \right)}. \quad (46)$$

Letting $S_0 + C + \Omega = 90$, $L = 100$, and $\lambda = 1$, and numerically solving Equation 43 provides the exact minimum $T^*$ in order to achieve a minimum surplus of $S_0$ which in this case is found to be $T^* = 10$ (see Figure 7). As can be seen from Figure 7, any value of $T > 10$ will yield at least a surplus of $S_0$. The corresponding approximate solution gives $T' = 90/(100 - 90) = 9$, which yields an error of just under 10%. From the sellers point of view, in order to quickly determine the optimal $T^*$, while avoiding the elaborate procedure of finding a numerical solution to Equation 43, one can simply first solve for $T'$, and then add a safety factor to ensure that the resultant surplus $\geq S_0$, which
in the present case may be 10%. A higher safety factor may be used to ensure
greater certainty of achieving the required level of minimum surplus.

Next, the overall surplus in accepting $K$ bids per auction is

$$ I_K - K \Delta C - \Omega $$ (47)

While the overall surplus in selling $K$ items through $K$ separate auctions
would be

$$ \frac{L}{\lambda T} \left( \lambda T + e^{-\lambda T} - 1 \right) - K(C + \Omega). $$ (48)

Thus, it would be more profitable to sell $K$ items in $K$ separate auctions
instead of selling them in a single auction if the expected surplus of the latter
is higher, i.e.

$$ \frac{L}{\lambda T} \left( \lambda T + e^{-\lambda T} - 1 \right) - K(C + \Omega) > I_K - K \Delta C - \Omega $$ (49)

or

$$ \frac{L}{\lambda T} \left( \lambda T + e^{-\lambda T} - 1 \right) - (K - 1)\Omega > I_K $$ (50)

For the important special case $K = 2$, the above becomes

$$ \frac{2L}{\lambda T} \left( \lambda T + e^{-\lambda T} - 1 \right) - \Omega > I_2 $$ (51)
and making use of Equation 40, this condition simplifies to

\[ e^z \left( 1 - \frac{z \Omega}{L} \right) > 1 + z - \frac{z^2}{2}. \]  

(52)

We see that in terms of magnitude, the left hand side increases exponentially in \( z \), while the right hand side increases quadratically. Thus, for sufficiently large \( z \), the left hand side will go negative with a large magnitude, while the right hand side will also go negative with a comparatively smaller magnitude; consequently the above inequality will not hold for large \( z \). Thus, when the number of bids is large, it is always preferable to sell the items in single auctions.

Sometimes, the auction fee structure is such that the auction website would charge for a certain percentage of the income payment, which for instance is the common practice of eBay. Denoting by \( \xi \) such a percentage, then the overall surplus in accepting \( K \) bids per auction is

\[ (1 - \xi)I_K - KC. \]  

(53)

In adopting the same approximation as before by suitably replacing \( N \) by \( \lambda T \), then from Equation 32, we have

\[ I_K \simeq \frac{(2\lambda T - K + 1)LK}{2(\lambda T + 1)}. \]  

(54)

Using this approximation for the special case \( K = 2 \), we have

\[ I_2 \simeq \frac{(2\lambda T - 1)L}{\lambda T + 1}, \]  

(55)

so that the overall surplus is approximately

\[ \frac{L(1 - \xi)(2\lambda T - 1)}{\lambda T + 1} - 2C. \]  

(56)

Thus it would be preferable to accept two bids per auction rather than to accept a single bid in two separate auctions if

\[ 2(1 - \xi)I_1 - 2C < (1 - \xi)I_2 - 2C, \]  

(57)

or,

\[ \frac{2\lambda TL}{\lambda T + 1} < \frac{2\lambda TL - L}{\lambda T + 1}, \]  

(58)

which is never the case for \( L > 0 \). Thus, for this particular auction fee structure, unlike the previous case, it would always be preferable to run two
separate auctions rather than a single auction given that the number of bids is large. In fact, even when the number of bids is not large, the general validity of this choice can be seen from Equation 32, where the total income from running a single auction is

\[(1 - \xi)I_{K|N} = \frac{LK(1 - \xi)(2N - K + 1)}{2(N + 1)},\]  

(59)

where \(I_{K|N}\) signifies the total income conditioning on \(N\). The corresponding quantity in running \(K\) separate auctions is

\[KI_{1|N} = \frac{(1 - \xi)NLK}{N + 1}\]  

(60)

Thus, it is preferable to run separate auctions if

\[\frac{N}{N + 1} > \frac{2N - K + 1}{2(N + 1)}\]  

(61)

which will be valid whenever \(K > 1\). Thus, for this particular auction fee structure, unlike the previous one, it is always more advantageous for the seller to sell the items in separate auctions.

4 Simulation

To enable the comparison between observed and theoretical values and to validate the mathematical models, an auction process simulator that implements the pseudo-code for the auction algorithm, has been constructed in C++. In order to sample values from the uniform and exponential distributions for the private value of bidders and the rate of bids respectively, the Boost C++ Library is used. In particular, we use the \texttt{variate_generator} with the \texttt{uniform	extunderscore 01} and \texttt{exponential	extunderscore distribution} headers, which is implemented on top of the \texttt{mersenne	extunderscore twister} pseudo-random number generator. The result is outputted as a space-delimited text string that states lambda, which is the incoming rate of bids and usually the variable we change, the duration of that auction, and the revenue generated from that auction. Ten thousand trials are run for each arrival rate, which is sampled in 0.01 intervals in the units concerned over the desired interval.

Figures 8, 9 and 10 superimpose simulation data on Figures 2, 3 and 4, respectively. The simulation is a validation of the theoretical results and from the close alignment of the theoretical and simulated curves, the simulation seems to corroborate the above theory.
5 Summary and Conclusions

A stochastic analysis of Internet auction process is presented, and closed-form expressions are obtained for the key performance metrics of transaction price, and auction income per unit time. It is found that the auction income critically depends on the number of competing bids, so that as the number of bids increases, the auction income increases, and it climbs relatively sharply at the beginning but gradually does so slowly, while the income rate tends to favour shorter auctions. Both exact expressions and approximate formulae are given, with the latter providing a good estimate when the number of bids is large.
Analysis of surplus for buyers and sellers in Internet auction processes are given, and by suitably adjusting the different parameters, bidder collusion and seller shilling behaviours may be represented. Compared with other means and channels for the exchange of goods, the aim of the sellers selling items through auctions is to increase the transaction price. Appropriately controlling the auction process would be effective in raising the surplus for the seller, who can optimise this by adjusting the auction duration and the number of bids accepted per auction. The exact optimal auction duration may be obtained through numerical methods, while closed-form results are obtained for an approximate solution. Depending on the auction fee structure and seller utility, it may sometimes be advantageous to accept two or more bids per auction. Throughout the analysis, it is found that the average number of bids is key determinant of performance. Simulations have been performed and close agreement with theoretical analysis is observed. In future works, more complex auction fee structures may be incorporated and analysed. In addition, the ratings of sellers — which relate to such factors as seller reputation, reliability, readiness to resolve disputes and provide refunds, delivery efficiency as well as seller surplus — are key considerations in Internet auctions and may also be incorporated into future optimisation models.
References


Quantitative Evaluation of Enterprise DRM Technology

Wen Zeng      Aad van Moorsel

Centre for Cybercrime and Computer Security, School of Computing Science
Newcastle University, NE1 7RU, UK.

wen.zeng@ncl.ac.uk      aad.vanmoorsel@ncl.ac.uk

Abstract:

It is of critical business importance for organizations to keep confidential digital documents secure, as the potential cost and damage incurred from the loss of confidential digital documents have increased significantly in recent years. Digital Rights Management (DRM) was developed to help organizations keep digital documents secure, as one of many digital information security solutions.

In this study, the functions of eight popular DRM products currently available on the market are reviewed, and the impact of using of these DRM products is evaluated quantitatively. A group of security metrics is defined reflecting the potential costs and impact to the organization incurred by implementing DRM products. Stochastic models are used to quantitatively evaluate the costs and impact of implementing a particular DRM product. In this study, it is found that although DRM products protect digital assets by encryption and by providing central control on information within the organization, this comes at a cost, since these security mechanisms typically reduce the productivity of the staff. The reduction in productivity is in turn measured in the form of non-productive time (NPT) which is an inherent part of the stochastic modeling process.

Keywords: Petri-nets, stochastic modeling, DRM products, digital security, non-productive time
1. Introduction

Many organizations maintain sensitive information or documents that should be accessed only by authorized personnel, for example, personal health records in health institutions, bank statements and account balances for financial organizations. Confidential information leakage and sensitive information dissertation have been identified as major information security threats that cause reputation damage, identity theft and even threaten the viability of the organization (Kaspersky Lab, 2006).

Although hacker attacks, virus epidemics and system vulnerabilities have been identified previously as the main causes of loss of sensitive information, the number of security incidents caused by internal attacks has increased significantly in recent years (IDC, 2009). In a comprehensive survey conducted by IDC (2009), 400 surveyed organizations admitted to 6244 incidents of employee negligence, 5794 incidents of excessive privilege and access control rights. The number of internal security incidents totaled 57,485 (IDC, 2009). Therefore, the behavior of internal staff has a significant impact on information security and attacks associated with the inappropriate behavior of internal staff are on the rise and are posing a great threat. For example, some staff members save sensitive information on laptops, USB devices or smart phones for convenience purposes. When these devices are lost, organization will not be able to exert any control on the information saved in these devices. In addition, employees in the company might occasionally send e-mail messages that contain confidential files as attachments without noticing that the files should not be distributed, or otherwise that the recipient should not see the files or is an unintended recipient who has been incorrectly (accidentally or otherwise) chosen to receive the e-mail. All of these behaviors could cause digital information to leak outside the company. It is essential that companies and organizations keep these information and document safe. Digital Rights Management (DRM) systems are developed to address these concerns and needs.

The term, Digital Rights Management (DRM), is used to describe any technology that delivers the capability of controlling the access rights of digital content both within and outside a digital environment (Umeh, 2007). Commercial DRMs are DRM systems that protect digital content in music, film and print industry. Enterprise DRM is one of many information security products and DRM claims to have the ability to protect the confidentiality, integrity and availability of information in the organization. Since organizations have to invest a significant amount of capital and continue to have operational expenditure on enterprise DRM products and enterprise DRM products might have negative effects on the efficiency of the organization, it is necessary to demonstrate that the benefits from DRM products exceed the costs of information security investment. However, it is challenging to quantitatively evaluate the potential impact of DRM products on the organization. For example, DRM technologies use access control (username and password) to limit unauthorized use of sensitive documents. However, authorized users might be unable to open a document and lose significant amount of time in finding the correct password to open the protected document. Thus, the business process is halted and productivity of employees is reduced.
Therefore, the contributions of this study are, 1) to review popular DRM products currently available on the market in terms of their common features; 2) to develop a methodology so that the impact of DRM technology can be evaluated quantitatively by stochastic modeling techniques. These techniques include defining security metrics and constructing and running Petri-nets stochastic models.

2. DRM technology

Eight of the currently popular DRM products are selected and reviewed in this document from the following four aspects: administrative models, content management, users monitoring, rights management (Zeng et al. 2010). Most of these products are developed by leading computer technology firms. The aim of this part is to understand the common features of DRM products and the working of DRM within the business processes of an organization. This serves as the fundamental work to analysis their effectiveness and costs.

- Oracle Information Rights Management (formerly SealedMedia E-DRM) (Oracle Corporation, 2009)
- Microsoft Windows Rights Management Services for Windows Server 2003 (Microsoft Corporation, 2009)
- Documentum IRM Services (Authentica, 2009)
- Liquid Machines Document Control (Liquid Machines, 2009)
- Secure2Trust (Avoco Secure, 2009)
- PDF Document Security (LockLizard, 2009)
- Workshare Protect (Workshare Inc., 2009)

2.1 Administrative models:
All eight DRM products reviewed have a similar centralized administrative model (Figure 1). Documents are stored in a centralized space (Zeng et al. 2010). Each document has a unique identifier, which is linked with users and rights (Beek, 2007). Only administrators who do not have access limitations can access the space to manage protected documents. Administrators can define the rights for each document and each user.

The organization also can control all the encrypted information centrally regardless of its location (Zeng et al. 2010). Dynamic policy centralized control allows the administrators to change the policy of the document no matter it has been distributed or not. The policy becomes effective immediately, when users open the document. This helps preventing unauthorized users from accessing documents.
Through centralized administration, organizations can manage digital information more effectively (Figure 1). Administrators have the highest access rights, and can define policies for all the documents and change policies for users. They can control the usage policies, revocation and exclusion. They can centrally define who can access the documents, what kind of things they can do with the content of the document after authorized user access, and the expiry date of the document.

When authorized user can not open a document, the user will have to contact the administrator, who will then check the status and rights of the user and make appropriate changes (Figure 2). This process might take some considerable time thus delay due business process and create non-productive time (NPT) for the employee.
Figure 2: Workflow of authorized users to open protected document when access is denied

When a user tries to open a protected document, access right request is sent to the DRM server, which records this request. If the user does not pass the authentication on the server, the user will not be able to use the document. The user will have to contact the administrator and ask for access rights. The administrator then views the activities associated with this document and checks the user’s role. If the user is allowed to access the document and the user has not been given access rights or the access rights are not enough for this user to use the document, the administrator will create the access rights for the user or change the usage polices for this user via the DRM server. Finally, the administrator sends the access rights to the user.

2.2 Content Management:
All eight DRM products reviewed use encryption to keep the information secure no matter where it has been transferred (Zeng et al. 2010). Authorized users use decryption keys to open the document and access the information. This helps organizations comply with government regulations. All of the products provide creation functions for each authorized user so that they have the ability to create various types of digital documents with unique identity for each document and secure these documents with rights for other users (Beek, 2007).

2.3 Users Monitoring:
All eight DRM products reviewed provide the function to manage user lists for every document, so administrators can create new users or delete existing users from the user list of the document (Zeng et al. 2010). All the products provide unique identity for each user, e.g. user
name and password, email address and password, fingerprint, and etc. In addition, administrators have the ability to restrict the usage of documents by defining usage policies for the documents; for example, number of times accessing document, expiration date of access rights to the document, and etc. (Beek, 2007).

2.4 Rights Management:
All eight DRM products reviewed provide functions for organizations to provide rights for each document and each user (Zeng et al. 2010). Full control, modify, read, print are basic rights for DRM products. All DRM products can provide limited document usage by using expiration dates. After expiration date, only administrators can access “expired” documents.

Digital content and the rights assigned to each user can be dynamically changed (Zeng et al. 2010). Users who own full control accounts have the ability to change the rights for digital content (Beek, 2007).

3. Background on Stochastic Models

Stochastic modeling is used to create an abstraction of a business process so that organizations can understand how DRM technologies change employees’ working processes, and evaluate the benefits, costs and impact of implementing a DRM solution. Therefore, stochastic Petri-net theory methodology is introduced in this section.

3.1 Classic Petri Net
Petri net is a graphical and mathematical modeling tool for the formal description of systems whose dynamics are characterized by concurrency, synchronization, mutual exclusion and conflict, which are typical features of distributed environments (Marsan, 1995). Petri nets have been widely used for structural modeling of workflows and have been applied to a wide range of qualitative and quantitative analysis (Marsan, 1995; Van der Aalst, 1998; Adam, 1998; Salimifard, 2001).

A definition of a classic Petri Net:

A Petri net is a 5-tuple \( N = (P, T, F, W, M_0) \), where:

- \( P = \{p_1, p_2, \ldots, p_n\} \), \( P \) is the set of places, \( p_n \) is the name of each place;
- \( T = \{t_1, t_2, \ldots, t_n\} \), \( T \) is the set transitions, \( t_n \) is the name of each transition;
- \( F \subseteq (P \times T) \cup (T \times P) \) is a set of arcs;
- \( W: F \rightarrow \{1, 2, 3, \ldots\} \) is a weight function;
- \( M_0: P \rightarrow \{0, 1, 2, 3, \ldots\} \) is the initial marking;
- \( P \cap T = \emptyset \) and \( P \cup T \neq \emptyset \).

A petri net consist of place, transition and arcs that connect them. Input arcs start at places and end at transitions, while output arcs start at a transition and end at a place. Places can contain tokens, which are used in the network to simulate the dynamic and concurrent activities of the system. The current state of the modeled system (the marking) is given by the number of tokens in each place. Transitions are active components, when a transition fires, the model activates, then tokens move in the model, which changes the states of the system. Transitions
are only allowed to fire when they are enabled, at least a token in every input place. More
rules of transition enabling and firing are defined at (Murata, 1989; Marsan, 1995).

3.2 Stochastic Activity Networks
For Petri net, if a firing delay associated with each transition, which specifies the time that the
transition has to be enabled. Before it can actually fire, if the delay is a random distribution
function, the Petri net is called Stochastic Petri net. Stochastic Activity Networks (SANs) are
stochastic extensions to Petri nets (Sanders, 2008). SANs consist of four primitive objects:
places, transitions, input gates and output gates. A place represents each state of the modeled
system; transitions represent actions of the modeled system that take some specified amount
of time to complete; input gates are used to control the enabling of activities and define the
marking changes that will occur when an activity completes; and output gates are used to
define the marking changes that will occur when activities complete.

3.3 Reward Formalism
“Reward models” are used to specify measures of system behavior (Sanders, 1991). Reward
model has two different reward structures: one is “rate rewards”, which is the rate at which
reward accumulates while the process is in the state during an interval of time; another is
“impulse rewards”, which is used to count the number of times an transition fires during an
interval of time.

Reward Structure:
The functions to express transition and marking oriented reward structure of a SAN with places
P and transitions A :

\[ C: A \rightarrow \mathbb{R}, \text{ where for } a \in A, \ C_a \text{ is the reward obtained due to completion of transition } a; \]

\[ R: \mathcal{P}(P, \mathbb{N}) \rightarrow \mathbb{R}, \text{ where for } v \in \mathcal{P}(P, \mathbb{N}), \ R(v) \text{ is the rate of reward obtained when for each } (p, n) \in v; \text{ there are } n \text{ tokens in place } p. \]

\( \mathbb{N} \) is the set of nature numbers, \( \mathcal{P}(P, \mathbb{N}) \) is the set of all partial functions between \( P \) and \( \mathbb{N} \).

Impulse rewards are associated with transition completion (via \( C \)) and rates rewards are
associated with number of taken in sets of places (via \( R \)).

\[ Y_{[t,t+1]} = \sum_{v \in \mathcal{P}(P,\mathbb{N})} R(v) M^v_{[t,t+1]} + \sum_{a \in A} C_a N^a_{[t,t+1]} \]

\[ RW_{[t,t+1]}_{t \rightarrow \infty} = \frac{Y_{[t,t+1]}}{l} \]

In this function, reward accumulated is related to the number of times each transition
completes and time spent in particular markings during an interval of time \([t, t+1]\). \( M^v_{[t,t+1]} \)
represents the total time that the SAN is in a marking such that for each \( \mathcal{P}(P, \mathbb{N}) \in v \), there are
\( n \) tokens in \( p \) during \([t, t+1]\).

\( N^a_{[t,t+1]} \) represents the number of completion of transition \( a \) during \([t, t+1]\).

3.4 Möbius software
Möbius is a software tool for modeling the reliability, availability and performance of complex
systems (Sanders, 2008). It supports majority of modeling techniques, specifically supports SANs (Sanders, 2008). It works by making different modeling processes (SANs modeling formalisms, compositional formalisms, reward formalisms, solvers) modular. Möbius provides simulation and numerical solvers for obtaining solutions on measures of interest. The simulation solver can be used to solve models using discrete event simulation (Sanders, 2008). Numerical solvers can be used on only models that have only exponentially and deterministically distribution transitions (Sanders, 2008).

4. Quantitative Evaluation of DRM

As a firm moves towards planning security strategies, some questions security managers would ask are: Is the document control necessary? Will the document control bring high costs to the organization? To answer these questions, security metrics should be defined as a standard to evaluate the effectiveness and the effect of using DRM (Sohn, 2006). The data of security metrics can help security managers make sound security investment decisions (Jaquith, 2007).

All eight DRM products provide centrally defines policies for each document, and centralized administration that allows organizations to manage authorization information from a single facility or location. Centralized administration also allows organizations to keep a minimum number of IT staff (administrators). However, if a significant number of users are trying to seek help from the administrators at the same time, some users will have to wait. This result in non-productivity time (NPT) and reduces the operational efficiency of the organization. In some cases, users might give up waiting and choose to precede their work without appropriate information.

In addition, since IT resource is not unlimited, it is necessary for organizations to classify documents into different levels of confidentiality so that higher-value documents will have higher levels of security (Humphreys, 2010). In this study, documents are divided into two levels: normal and high-value (classified documents). Users only need a username and password to open normal documents; on the other hand, users have to have the particular password of that document to open a classified document except user’s own username and password.

In this document, three aspects are considered to evaluate DRM: the number of users that can read protected documents under the encryption mechanism; the number of users that can change protected documents under the usage policy, and how the document classification strategy impacts the effect of using the DRM. Therefore, eight security metrics are defined to measure the effect of using DRM from these three aspects (table 4), e.g. the percentage of authorized users can read documents under documents classification.
Security Metrics

<table>
<thead>
<tr>
<th>User’s Type</th>
<th>Executive</th>
<th>Document Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>authorized user</td>
<td>can read documents</td>
</tr>
<tr>
<td>2</td>
<td>authorized user</td>
<td>cannot read documents</td>
</tr>
<tr>
<td>3</td>
<td>authorized user</td>
<td>can read documents</td>
</tr>
<tr>
<td>4</td>
<td>authorized user</td>
<td>cannot read documents</td>
</tr>
<tr>
<td>5</td>
<td>authorized user</td>
<td>can change documents</td>
</tr>
<tr>
<td>6</td>
<td>authorized user</td>
<td>cannot change documents</td>
</tr>
<tr>
<td>7</td>
<td>authorized user</td>
<td>can change documents</td>
</tr>
<tr>
<td>8</td>
<td>authorized user</td>
<td>cannot change documents</td>
</tr>
</tbody>
</table>

Table 4: Security Metrics are defined as standards to help evaluate DRM.

Microsoft information rights management (MS IRM) is used as the case study in this document. From the experiment and information provided by Microsoft technical centre, a model is built in this part to evaluate the effect of using MS IRM. This model focuses on function utility and user’s tasks when they use data. The core feature of MS IRM is: protected documents are centrally managed. In order to ensure access permissions of employees successfully, a number of IT staff (administrators) are needed in the organization to help employees solve problem, when users cannot use protected documents properly. In this document, two help desks are defined. One help desk helps users deal with the access issue, the other help desk helps users deal with the usage issue. Documents in the organization are divided into two levels: normal documents and high-value documents.

A stochastic Petri-net model is built to simulate this business process and quantitatively evaluate the effect of using MS IRM on organization’s operational efficiency.

![Figure 3: A Petri net model representing authorized users' access protected documents](image)

The model (Figure 3) consists of six input gates, six output gates, twelve places, and fourteen transitions, eight timed transitions and six immediate transitions. Timed transitions are associated with random exponentially distributed firing delays, and immediate transitions are...
fired in zero time.

Authorized users (Users) use four documents per day during working hours (the time taken to use the documents is given by Open_Doc). When authorized users try to open documents, two things might happen, if the document is a secret document (Secret_Doc), firstly, the user needs a username and password in order to login to the system (given by the time taken by Login_In_Account). If the user passes user authentication, the user needs to input in a document password (Enter_PassWord), which takes time (given by the time taken by Ask_Password). If the user passes the document authentication, then the user can open the document and become an active user (Active_User_1), but if the user cannot pass the document authentication or the user cannot pass the user authentication, the user would contact the administrators (Need_help_1) for help. If the user can get help from the administrators (the time taken is given by Admin_help_1), the user inputs the document password to open the document. If the user cannot get help from the administrator (No_Help_1), the user would give up (Give_up_1). Active users might change protected documents (change_1), but they might have not enough usage authority to change documents, so they will contact administrators to ask for help (Need_Help_2). If users can get help from the administrators (the time taken given by Admin_Help_2), they can change documents, if they cannot receive help (No_Help_2), they might be give up.

If the document is a normal document (Normal_Doc), the user just needs to input a username and a password to open the document (the time taken for this is given by Login_Account). If the user can open the document, the user can open the document (Active_User_2). Otherwise the user needs to contact the administrators (Need_Help_1) for help. Active users might change protected documents (change_2), but they might not be given enough usage authority to change documents, so they will contact administrators to ask for help (Need_Help_2).

All of the input parameters in this model are in table 5. The time scale of the model is in minutes. It assumes that administrators need spend 10 minutes average to help each user, if less than seven users are in the queue waiting for help, the users would be patient enough until they get help from administrators. But if more than seven users are in the queue, users only at most can wait for one hour, if after one hour users still cannot get help, they would give up.
<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numbers of documents, an user might use every day on average</td>
<td>4</td>
</tr>
<tr>
<td>Numbers of normal working hours per day</td>
<td>8</td>
</tr>
<tr>
<td>Numbers of working weeks per year</td>
<td>40</td>
</tr>
<tr>
<td>Numbers of help desk help to solve documents access issue</td>
<td>1</td>
</tr>
<tr>
<td>Numbers of help desk help to solve documents authority issue</td>
<td>1</td>
</tr>
<tr>
<td>Numbers of documents an user uses per working hour on average</td>
<td>0.5</td>
</tr>
<tr>
<td>The average time users need to spend to pass user authentication</td>
<td>0.5 minutes</td>
</tr>
<tr>
<td>The average time users need to spend to pass document authentication</td>
<td>0.5 minutes</td>
</tr>
<tr>
<td>The average time administrator spends to help one user</td>
<td>10 minutes</td>
</tr>
<tr>
<td>The percentage of secret documents, if the organization uses a document</td>
<td>50%</td>
</tr>
<tr>
<td>classification strategy</td>
<td></td>
</tr>
<tr>
<td>The percentage of time when users experience a login system failure or users cannot remember the password for secret documents</td>
<td>5%</td>
</tr>
<tr>
<td>The percentage of documents that can be changed by authorized users</td>
<td>40%</td>
</tr>
</tbody>
</table>

Table 5: Input parameters of the Petri-net model

The behavior of the model can be measured by the Impulse Rewards model and Rates Rewards model, which is supported by the Möbius software. The throughput of transitions are computed according to the formula which is described in Section 3.3: \[ \sum_{a \in \mathcal{A}} c_a N_{[t,t+1]}^a \] ; The number of tokens in sets of places are computed according to the formula: \[ \mathcal{R}(v) \in \mathcal{P}(\mathcal{P}, \mathcal{N}) \]

To measure the number of authorized users that can read documents, the sum of the throughput of transitions: Usage_Doc_1 (case1&2), Usage_Doc_2 (case1&2) are computed. To measure the number of documents that authorized users can change, the sum of the throughput of transitions: Changble_1 (case1), Changeable_2 (case2), Admin_Help_2 (case2) are computed. To measure the numbers of documents that authorized users have tried to use, the sum of the throughput of transition: Open_Doc (case1 & 2) is computed. To measure the non-productive time (NPT), the fraction of time users spend in any place other than (Users) are computed, it is considered time loss because of MS IRM provide identity and access control technology.

4.1 Result analysis:

The deployment of the DRM product has a significant impact on the operational efficiency of the organization. Within six months of the deployment of MS IRM in the network system in this case study (48000 time units in the model), a middle-sized organization that has 500 authorized users will incur about 9000-11000 hours, or about 375-458 days, of non-productive time (Figure 4), under the assumption made in Table 5. This total loss of productive time has two components: the time spent on authentication procedures and the time spent on waiting response from the administrators. In this case study, under the assumption made in Table 5, the total authentication time loss amounts to 2500-3000 hours (Figure 5) and the total waiting time loss amounts to 6500-8000 hours (Figure 6).
Figure 4: Total non-productive time (NPT) associated with the deployment of MS IRM under the assumptions made in Table 5. NPT increases significantly when the number of authorized users served by each administrator increases. Proper classification of the company documents will reduce the NPT impact from the DRM deployment.

Figure 5: Total authentication time loss associated with the deployment of MS IRM under the assumptions made in Table 5. Proper classification of the company documents will reduce the authentication time loss impact from the DRM deployment.
The total non-productive time (NPT) is closely related to three factors: user behaviors, the number of employees supported by the administrators and the company document classification process (Table 5 and Figure 4). User behavior statistics are critical inputs into this Petri-nets model (Table 5), for example, the average time that users have to spend to pass user authentication and the frequency in which users forget the proper password and have to seek assistance. User behavior statistics can be improved by proper security awareness training (Stephanou, 2008). The number of employees served by each administrator is another important factor. The larger the ratio between the number of employees to administrators, the more difficult it is for employees to get timely assistance.

In addition, if the organization adopts a proper classification system, the non-productive time and availability of documents associated with the deployment of DRM products will be reduced (Figure 4,5,6,7,8). That is, a proper document classification will help offset part of the negative effect that DRM product has on the efficiency of the organization. However, when the size of the organization is smaller than 100 per administrator, the difference in NPT and availability of documents between the modeled two scenarios, with and without a document classification, is not significant (Figure 4,5,6,7,8). Therefore, when an organization makes decision on whether to adopt a more complex digital file classification system when planning the deployment of a DRM product, both the size of the organization and the number of administrators have to be taken into consideration.
Figure 7: The total numbers of documents users can read associated with the deployment of MS IRM under the assumptions made in Table 5. The numbers of documents users can read decreases significantly when the number of authorized users served by each administrator increases. Classification of the company documents will reduce availability of documents impact from the DRM deployment.

Figure 8: The total numbers of documents users can change associated with the deployment of MS IRM under the assumptions made in Table 5. The numbers of documents users can change decreases significantly when the number of authorized users served by each administrator increases.

5. Conclusions

The stochastic Petri-net method is applied in this case study to build a model for the purpose of simulating the business process of deploying a DRM product. This model is used to quantify the potential impact of DRM products on the efficiency of the organization.
DRM products use encryption to make information in the organization secure so that only authorized users can access to the documents. However, it is found that DRM products reduced the availability of documents within the organization even to authorized users. The authentication processes will result in non-productive time (NPT) for the employees, therefore, reducing the overall operational efficiency of the organization. This is particular the case when authorized users cannot open secured documents for various reasons and have to wait for the assistance from the administrators.

The stochastic Petri-net model quantified the NPT incurred by the deployment of the DRM product, and assisted in identifying the three major factors that affect the NPT: user behavior, the number of authorized served by each administrator and document classification.

User behavior statistics are part of the input data into the Stochastic Petri-net model. They have direct impact on the amount of NPT incurred by the DRM products, as is output from the simulation model.

The NPT associated with the waiting for responses from administrators increases significantly when the number of authorized users served by each administrator increases, although centralized control functionality of DRM helps to limit the number of administrators within an organization.

In addition, a proper document classification process can help reduce the negative impact of the deployment of DRM products on operational efficiency, although the benefit from classifying documents is limited, when the organization is small, or there is a large number of DRM administrators within the organization.

In summary, it is demonstrated that this stochastic Petri-net based business model has the potential to be used to help organizations to quantify the benefit and cost of implementing DRM products in order to make sound information security investment decisions.

Acknowledgements:

The authors want to express their sincere appreciations to Simon Parkin at Newcastle University for his constructive suggestions and feedbacks during both the research and writing stages of this work. Detailed, critical reviews by four anonymous reviewers from the workshop committee significantly improved the quality of this manuscript. The authors feel deeply obliged to their contributions.

The second author has been funded in part by UK Technology Strategy Board (TSB), grant nr. P0007E (“Trust Economics”).
References


Abstract
In this paper, a Layered Queueing Network (LQN) performance model is used for studying an Apache-PHP web application with PostgreSQL backend-database. Performance evaluation is done by obtaining load test measurements and by solving the LQN model. Model validation is performed by comparing the model results with the load test results. With average error of 3.77% for throughput and 12.15% for response times the model is shown to capture the web application’s performance. Furthermore, performance analysis is done to determine the system configuration which would ease the identified bottleneck resource.

Keywords: Performance modeling and validation, Layered Queueing Networks, Software Performance Engineering, Performance measurement, Load Testing, PHP

1. Introduction
The complexity and nature of web applications require their development to follow a Software Development Life Cycle (SDLC). As users expect web applications to be quick and responsive, attention to system’s performance is paid from an early software development stage and followed throughout the SDLC by means of a well-defined systematic process: Software Performance Engineering (SPE) [23]. As part of the SPE process, measurements and performance modeling can be used to study a system. To directly assess if an application will meet the required performance objectives based on available resources, for the purposes of capacity
planning [24], a measurement-based approach is adopted. The behaviour of the system under given customer workload can provide results which can help identify performance bottlenecks [25]. Performance modeling, which uses performance models, also finds use in capacity planning by means of predicting system's performance and by pinpointing system bottlenecks. Other uses of modeling include capacity provisioning – i.e. allocating and preparing resources to handle the demands – and finding application configuration parameters that meet the desired objectives [8].

For performance modeling, use of well known Queueing Network (QN) models is very common. However, basic QN models fall short in being able to account for software contention as seen in software servers [28]. The QNs depict software as customers only, whereas software servers behave at times as servers when they are serving their client requests and behave other times as clients themselves when requesting service from other servers (such as database) [28]. If effects of queueing due to software contention are ignored then response times are understated leading to inaccurate results. Furthermore, aspects such as parallel software execution that is seen through creation of a child process from an existing process (fork) cannot be directly represented [29].

Layered Queueing Networks (LQN) [3] analytical models are based on extended QNs and are designed to eliminate the aforementioned shortcomings of QN. In case of Remote Procedure Calls (RPC), the queueing that occurs at lower layers due to software contention is included in the upper layer response time of an LQN model [3]. Furthermore, they can also model nested RPCs [9]. The parallel execution of software and servers which send “early reply” [9] (i.e. some processing at the server is to happen in second phase after reply is sent to the client) can also be modeled [28]. They are well-suited to depict both complex software applications and the hardware resources that these software entities run on [3]. With the ability to incorporate varying degrees of details in the model, LQN performance modeling can easily be integrated with the SDLC. LQNs are ideal for representing the interactions and intricacies of multi-tier application and this work therefore uses LQN for performance modeling.

In this paper, an LQN performance model of a Linux Apache-PHP web application with PostgreSQL backend-database is presented. The solution of the model provides performance metrics such as steady-state throughput and response times. The model results are compared with measurement results derived from load testing the system for model validation. The analysis of the model results is provided and includes identification of the bottleneck resource with mention of an approach that would scale the system by avoiding the bottleneck resource saturation early.

The paper is organized as follows: Section 2 lists related works in the area of web performance modeling. Section 3 gives a brief overview of LQN. Section 4 provides the design details of Web Application. Section 5 describes
the Load Testing setup. Section 6 presents the LQN model of the application. Section 7 presents the measurements and model evaluation results with further analysis to ease bottleneck resource. Section 8 presents the conclusions and future work.

2. Related Works

One of the earlier works pertaining to web system performance analysis is done in [26] where a four-station open QN model – consisting of Client, Web server and Network – of a simple file Web server has been presented and analyzed.

Kounev & Buchmann in [10] describe a closed queueing model of SPECjAppServer2002 (J2EE) benchmark comprising of Client, Application Server Cluster, Database Server and Production Line Stations. The paper explains how service demands – which represent the demands that requests place on the computing resources and serve as inputs to the model – were obtained through use of Operational Laws of QNs. The model validation shows high accuracy of performance prediction with average error of 2% for throughput, 6% for CPU utilization and 18% for response time.

Urgaonkar et al. in [8] present a closed QN model of multi-tier internet applications which considers caching, concurrency limits and multiple session-based class requests. The model is validated through two J2EE applications: RUBiS and RUBBoS.

[27] describes a closed QN model of a 3-tiered web application comprising of Apache Web server, Tomcat Application server and MySQL Database server. To model the concurrency limits – such as maximum number of concurrently running threads/processes – of Apache and MySQL servers, multi-station queues are used.

The above works are very useful representations of web system modeling and their measurements; however, with respect to modeling they encounter the same drawbacks as those of QNs, which LQN overcomes through easily representing large software and hardware systems while also incorporating aspects such as software contention that affects the performance.

LQN performance models for web systems have been studied in [4, 5, 6, 7]. In [7], the SPECjAppServer2001 EJB benchmark is modeled as a simple LQN model, validated by measurements obtained from [17]. [6] models a JavaEE ECPerf benchmark based on LQN EJB templates introduced in [5]. However, as [8] indicates, most of these works have focused on Java Enterprise applications. In this work we utilize the versatility of LQN to study a PHP web application. Some other useful works based on LQN can be found in [3, 18, 19, 20, 21].

[21] models a J2EE bank application using LQN templates and performs model validation. The test system for measurements comprises of client load generator, EJB Application server and Database server machines. Tools like
JProbe and sar were used for profiling and obtaining usage information of resources such as CPU, network and disk. For tests, the beans were either accessed sequentially or in random order. The validation results for higher client numbers reported errors ranging from 6.2% to 23.9% (sequential) and from 2.1% to 24.5% (random).

It is interesting to realize that many works have been done with regards to modeling but very few compare the model results with actual measurements. Alongside LQN modeling, only [7] and [21] from above use measurements to perform model validation, and these are considered as similar pieces of works to ours.

Also, similar to our work, [22] and [2] have used LQN to model a CGI Web server and an Apache-PHP-MySQL system, respectively. The model in [22] incorporates serving of both dynamic and static contents by the server, however no database tier is considered in the study. The response time from the model evaluation is found to match closely with response time measurements over a period of two months. Measurement data collected through custom instrumentation serve as input parameters to the LQN model and are also used for model-validation. In [2] the LQN model of a web system is evaluated for 40 users and compared with the load test results, which show the successful prediction of performance by the model. The system consists of two Web servers, a Load balancer and a Database server. Other performance modeling formalisms such as Stochastic Process Algebra (SPA) and Stochastic Petri Nets (SPN) have also been evaluated in this paper.

However, in contrast to these two aforementioned papers (i.e. [22] and [2]) our work uses LQN activities to define in detail the web software entities and the precedence of interactions between them.

3. Layered Queueing Networks

An example LQN model is shown in Figure 1. The outermost parallelograms represent the Tasks and the parallelogram within the Tasks corresponds to the Entries. Entries are akin to customer classes of QNs [9].
Furthermore, each Entry can be sub-divided into smaller units of work known as Activities [9], represented by rectangles. Processors are shown as oval shapes. Multiplicity of Tasks signifies multiple threads of a software process shown in the figure within braces.

Communication between the Entries of Tasks can be of three types: Synchronous, Asynchronous and Forwarding. In synchronous communication the requesting task (client) blocks until a response is received from a server task. In asynchronous interaction the client task does not block after sending a request. In forwarding communication, the server (task A) that received the request forwards the request to another task (task B). Task A at this point starts execution in phase two and when task B is done processing its request, it sends a response back to the client after which task B will begin its execution in phase two. In Figure 1, synchronous calls are shown with ‘normal’ arrows and asynchronous calls are shown as ‘vee’ arrows.

Figure 1 represents a system where there are 100 Clients interacting with a single-processor, two-threaded Server process. The Client initiates the requests performing the Interact operation. The data is stored in the DB database where data access and manipulation from the Client happens through three synchronous Read(s) and an asynchronous Write operation on the Server. Once the Client receives a response back from the Server, it thinks for 7 seconds (Z=7) and then initiates another set of requests for Read and Write, repeating this process infinitely. Each entry has an associated service time per phase. Due to lack of space, only the Client task’s service times are presented – shown within square brackets. The arrows show the direction in which the requests are made. Unless specified otherwise, the frequency of interactions is one.

Inputs to the LQN model are scheduling discipline of the hardware resources, customer workload intensity and the service demands of the customers for the model components at each phase. The main performance metrics available from LQN model evaluation are steady-state throughput, response times, and utilizations of the modeled components.

4. Web Application

4.1 Overview

The web application under study, MyBikeRoutes-OSM, is a repository of bicycle routes that allows the users to create and share bicycle routes from anywhere in the world. Users can search for the best bike route between given source and destination locations, i.e. best path search. The best path search functionality is currently a prototype where the search is essentially performed using the roadways instead of the bicycle routes. This however does not affect the study as our main focus is towards the system’s performance. The application uses OpenLayers JavaScript API to display
OpenStreetMap\(^3\) (OSM) maps [11]. In the particular case of this application, the maps are generated on a Browser through an ordering of pre-rendered map tiles/images residing on the server. PostgreSQL database functions as the storage facility of the bike routes data and provides best path routing features, where the routing functionality is made available by the pgRouting\(^4\) project [12]. For displaying bike routes data or for route-search, OpenLayers at client-side basically communicates with the PostgreSQL backend-database through an Apache-PHP server.

MyBikeRoutes-OSM derives from the MyBikeRoutes\(^5\) web application [13], which is functionality-wise similar but an online web replica of its derivative project. The apparent difference is the use of mapping services from Google Maps API\(^6\) [14] to display Google Maps and route information by MyBikeRoutes website, where the bicycle routes data is housed in a MySQL backend-database. Furthermore, the best path search on the MyBikeRoutes site actually uses bike routes for the search. Here the client-side JavaScript interacts with Google Maps API, whereas the MyBikeRoutes-OSM project uses pgRouting at the Database server layer to provide the search functionality.

MyBikeRoutes and MyBikeRoutes-OSM projects are both recent development endeavours and there are subsequent enhancements of these applications that have been envisioned for future. However, before proceeding with involved development efforts, it is best to know about system bottlenecks and find the performance indicators following the SPE process. Thus, we have analyzed the performance of MyBikeRoutes-OSM application in this work.

4.2 Process Flow Diagram

Figure 2 displays the process flow of the web application. As a user visits the website from their browser, the bicycle routes are displayed on the OSM map. Next, the user may draw bicycle routes or perform a best path search between their chosen source and destination locations. If a search is performed then the best path algorithm is run and the best path is displayed on the map with the option to then save the results. On the other hand, the user may draw and store bicycle routes to share them with other users.

---

3 http://www.openstreetmap.org/
4 http://www.pgrouting.org/
5 http://www.mybikeroutes.com
6 http://code.google.com/apis/maps/index.html
5. Load Testing

Load Testing is a measurement-based approach where behaviour of the System Under Test (SUT) is studied under different workloads. The outputs include actual throughputs and response times. In our case, load testing is used to study the system's performance, obtain model service demands and validate performance model results. For this work, a free open-source load tester, JMeter\(^7\) [15], was used for load testing.

5.1 Configuration and Topology

![Load Test Topology](image)

Figure 3: Load Test Topology

Figure 3, outlines the Remote Testing setup (Master-Slave configuration) for the measurements, where the Jmeter-Master machine initiates the test while the Jmeter-Slave is the actual client machine that simulates the virtual users for sending requests to the Server under test. Each of these machines have identical physical configuration. The Server runs the Apache-PHP Application Server and PostgreSQL Database.

The following are the machine configurations and also the details of software used: Pentium 4 3.4 GHz (32-bit), 993 MB RAM, Ubuntu 10.04, 1000 Mb/s Network, Apache Prefork 2.2.14, PHP 5.32, PostgreSQL 8.4, JMeter 2.3.4.

\(^7\) [http://jakarta.apache.org/jmeter/](http://jakarta.apache.org/jmeter/)
5.2 Test Plan

As a part of the SPE process, it is essential to determine the performance-intensive scenarios of the application early [23]. To achieve this, first, all the HTTP requests generated by Mozilla Firefox browser while navigating the MyBikeRoutes-OSM application were recorded in JMeter. A load test with a single user was run to determine the performance intensive requests, i.e. requests which had high response time. From the previous step the base scenario (or base test plan in JMeter) was created by removing non-critical requests. Figure 4 shows the scenario’s sequence diagram for one user session. The figure represents the actions of a User who initiates the web communication. The user actions are shown for clarity, however, the actual load test plan consists of the HTTP requests of the Client/Browser only. The AppServer or Application Server represents the Apache-PHP server and the DB corresponds to the backend-database.

Based on Figure 4, in all, there are nine request classes sent to the Application Server in one complete user session and the initial five requests consists of an HTML file, three JavaScript files, and bike routes data, representing the requests that are sent on visiting the site. In this first group, except for the last request of bike routes data which interacts also with the Database, all the other requests just interact with the Application Server.
After this, one request for best path search and another to save the best path results is run. Very similar requests are again made for the second bike routes search, however with different start and end destinations. This final set of requests, communicate with both the Application Server and the Database.

For the load tests, a think time of 7 seconds was added before calling *reqHTML*. This scenario which includes think time of 7 seconds will be referred as *Base-Scenario* from this point onwards. The load tests were run for a duration of 1800 sec for each *N* Virtual Users, where *N* = {1, 2, 4, 6, 10, 20, 30, 40, 50, 80, 120}. To make sure that the Client machine could simulate the users and was not the bottleneck, JMeter tests were run in command-line mode with summary reporting. Furthermore, by applying *Little’s Law* to the JMeter results it was easily verified that the number of users active in the SUT were close to the number of virtual users initiated by JMeter [30]. The results obtained from the load tests were the session throughput and average session response time. These results are presented in section 7.

6. LQN Performance Modeling

The sequence diagram described earlier in Figure 4 is used for LQN model creation. The following section explains the LQN model.

6.1 Base-Scenario Model

![Figure 5: Base-Scenario LQN Model](image)

Figure 5 shows the LQN model of *Base-Scenario* in a load test. *Reply* activities are not shown to ease in understanding of the figure. There are *N Browsers* running on Infinite Servers (*pClient*). The model is evaluated...
separately for each value of $N$. This is a closed model where once a request session is completely processed, the customer is sent back to begin a new set of requests after waiting for a given think time. Since the processing at the client machine did not include browser rendering or page generation in the load test, the service demands for the Browser entries in the model are set to 0. Each Browser sends requests to the AppServer task through the entries of the NetworkClient, which represents the delay incurred when sending messages through the LAN from Browser to the AppServer. Similarly, NetworkServer represents delay due to sending reply from the AppServer. The two Network tasks are modeled as infinite threads running on Infinite Servers. Based on the MaxClients [31] directive of Apache server, which sets a limit to the maximum number of processes that are available to handle client requests concurrently, the AppServer multiplicity is set to 150. Similarly, based on max_connections [16] parameter of PostgreSQL database, which specifies the maximum concurrent connections to PostgreSQL database, the DB task multiplicity is set to 100. Both AppServer and DB tasks execute on the same uni-processor (pApache), which has Processor Sharing (PS) scheduling discipline. The disk is modelled by the Disk task, which has nine entries. Each entry relates to the nine requests issued by the Browser, i.e., the first request (reqHTML) has its disk service demand provided by disk1 entry, then reqJS1 has the disk service demand provided by disk2, and following the same pattern for other requests.

An assumption regarding Disk entries has been made in the model. For a request, only one interaction with the Disk entries is assumed, i.e. if both AppServer and DB task perform certain number of disk I/Os for a particular request, then only one call to the Disk at the end of the nested interaction is depicted in the model. In this case, the service time of the respective Disk entry is for one visit only found by multiplying the number of disk I/Os and the average service time at the disk. For finding the number of disk I/Os, the Forced Flow Law (refer section 6.2) has been used, where both the system throughput and disk I/O throughputs have been found through measurements. Similar ideas have been used previously by [27] and [10] for the database-tier in their works. Like-wise this work also makes the same assumption as the Disk entries for the DB entries.

There are five request classes that require database access, i.e. reqViewRoutes, reqROUTING1, reqROUTING2, reqADD1 and reqADD2. There is one SQL query executed for reqViewRoutes, which just involves retrieving the routes data. Both the routing requests (reqROUTING1 and reqROUTING2) require running three queries – relating to start and end points and finally the routing search. There is one query executed to insert a route for reqADD1 and reqADD2 requests. However, as mentioned in the earlier paragraph, each entry of DB task has only one visit made to it in the model, i.e. dbViewRoutes, dbRouting1, dbAdd1, dbRouting2 and dbAdd2 have only one visit from the upper AppServer task layer. Based on this, considering dbViewRoutes, the
service times of each SQL query executed for \textit{dbViewRoutes} was summed and presented finally as the service time of the \textit{dbViewRoutes} entry. Similarly, service times for other entries of the \textit{DB} task have been found. The reader may refer to section 6.2 for further details regarding the calculations of service demands for the entries. In the following paragraph, the sequence of execution that the model represents is explained in detail.

In the model, the \textit{reqHTML} activity initiates the \textit{Browser} requests to which the \textit{AppServer} responds back after execution of \textit{sendHTML}. Any disk operations by \textit{sendHTML} happens using the \textit{disk1} entry at the \textit{pDisk}. The network delay of \textit{NetworkClient} is also incorporated as the request goes from \textit{reqHTML} through the \textit{n1c} entry to the \textit{sendHTML} entry. Following \textit{reqHTML}, there are three JavaScript (\textit{reqJS1, reqJS2, reqJS3}) and \textit{reqRoutes} requests that are sent sequentially, completing the actions of a web site visit. In a similar pattern, the other four remaining requests are also sent in order and processed. Some \textit{AppServer} activities need to retrieve or amend data on the DB, forming a nested interaction, where the \textit{AppServer} only sends reply back to the \textit{Browser} when the \textit{AppServer's} request has been responded back by the DB. Each \textit{AppServer} reply also guarantees incorporation of the \textit{Network} delay in the response time by interacting either directly or indirectly – through a Disk – with the \textit{NetworkServer} before sending any reply. This models a complete session of the \textit{Base-Scenario} of the application. The service demands for the \textit{Base-Scenario} model are discovered by applying the \textit{Utilization Law}, details of which are presented in the following sections.

### 6.2 Discovering Service Demands

If the utilizations and throughputs of system resources are available – from load testing or monitoring – the \textit{Utilization Law} can be applied to derive service demands. Previously, Kounev & Buchmann in [10] have used the \textit{Utilization Law} to find service demands and this work uses the same approach. For Linux, utilities such as \textit{iostat} and \textit{sar} are useful monitoring tools to accomplish this job. The following paragraph gives a brief overview of the calculations related to queueing theory that involve finding service demands using the aforementioned ideas.

Consider a station \(i\) in a queueing system and a request class \(c\), with average utilization of the station due to requests of \(c\) as \(U_{c,i}\), average system throughput as \(X_c\), and throughput at station \(i\) as \(X_{c,i}\), then the service demand \(D_{c,i}\) at the station due to the request class can be calculated by applying \textit{Utilization Law} as follows, \(D_{c,i} = U_{c,i}/X_c\) [29]. Service demand is also the product of number of visits \(V_{c,i}\) and the average service time per visit \(S_{c,i}\), i.e. \(D_{c,i} = V_{c,i} * S_{c,i}\). Here, \(V_{c,i}\) represents the number of visits made to a station for each request of a class-request [29, 32]. If \(X_{c,i}\) and \(X_c\) are known then \(V_{c,i} = X_{c,i}/X_c\) (\textit{Forced Flow Law}) [29]. Thus, given \(X_c\) and \(U_{c,i}\) then \(D_{c,i}\) can be derived. And then if \(V_{c,i}\) is found from \textit{Forced Flow Law}, then \(S_{c,i}\) can be calculated where both \(V_{c,i}\) and \(S_{c,i}\) serve as inputs to the LQN model. For this work, \(X_c\) was
found using JMeter, and $U_{c,i}$ was found from sar utility. $X_{c,i}$ is required for disk I/O service demands and also found from running sar.

To determine service demands, the first step was to create separate JMeter tests with one user load having no think time for each request class of Base-Scenario, thereby creating nine tests. Since there is only one user, the contention due to queueing is at the lowest. Each test was run for duration of 900 seconds while the SUT was monitored using sar. Utilization output of sar includes iowait% and idle%, which specify the % of time the CPU is waiting for IO processing and the % of time CPU is not processing, respectively. The CPU utilization can then be obtained by subtracting the idle% from 100. The throughput obtained from JMeter for each test and the CPU utilization obtained from sar was used to find the total service time of each request at the SUT processor including service time for disk I/O. Furthermore, sar was used to find the average disk service time, and the disk tps (throughput per second). The number of disk I/Os was found by dividing the disk tps by the JMeter throughput, thereby using the Forced Flow Law. As mentioned earlier in section 6.1, only one visit from the upper layers to the disk entries is assumed in the model, therefore, the average disk service time was multiplied by the number of disk I/Os to obtain the normalized service time for one disk I/O, which is the input for the Disk entries at the pDisk. The service time found for a request’s disk I/O was subtracted from the total service time at the SUT processor, thereby giving the service time for the request class at the AppServer task at the pServer processor. Note that if the CPU utilization was obtained earlier by subtracting both idle% and iowait% from 100 then there would be no need to subtract the request’s disk I/O from the total service time at the SUT. The total Network delay was found by subtracting the total service time at the SUT by the JMeter response time for the request. Note for obtaining the NetworkClient and NetworkServer service times, the Network delay is divided by 2. An example of above follows.

**Example1:** For the first request class (reqHTML) load test, the average CPU Utilization was 59.18%, i.e. the % of time the CPU was not idle. The JMeter throughput was 61.44 requests/s and the average response time was 11 ms. Applying Utilization Law, the total service demand at the SUT processor and disk would be $0.5918/61.44 = 9.63$ ms. The disk I/O service time was found to be 0.36 ms and the disk tps was 1.69. Using Forced Flow Law, the average number of disk visits was $1.69/61.44 = 0.02751$. Therefore, the normalized service time for disk1 entry is $0.02751 \times 0.36 = 0.01$ ms. The service time at the sendHTML entry is $9.63 - 0.01 = 9.62$ ms. Based on this the total Network delay is $11 - 9.63 = 1.37$ ms, and therefore the service time for n1c and n1s entries is $1.37/2 = 0.685$ ms.

For requests that involve database calls, the following approach was adopted. For request classes that have very small service times of about 1 ms at the DB task, such as start and stop point queries of the routing search (4 of such queries in total), the EXPLAIN ANALYZE [16] query command of
PostgreSQL was used. The latter command provides the runtime details of a query that is to be evaluated. The EXPLAIN ANALYZE command was executed five times for each start and stop queries and the average runtime was used as the service time.

For the other longer queries, i.e. the two routing searches, the viewing routes query, and the two insertion of new routes queries, JMeter tests with one user load and no think time were run for a duration of 900 sec. Applying Utilization Law, and following similar calculations as for Example 1 the service demands were found.

Note that only one visit is made from the AppServer layer entries/activities to a corresponding database query based on the web application, therefore for a “database query” the visit count is one and the service demand is equal to the service time. However, it is key to realize that based on the web application, the Application Server may call multiple database queries for a particular request class, i.e. for reqROUTING1 request, one visit is made for start point, one for end point and finally one for routing, thereby a total of three queries are called. In this case, the assumptions regarding the model have been made as clearly outlined in section 6.1. For the reqROUTING1 request example just considered, the service times of the three queries is summed to form one entry, dbRouting1.

Table 1: Service Demand Parameters for Base-Scenario Model

<table>
<thead>
<tr>
<th>Request class</th>
<th>AppServer (ms)</th>
<th>DB (ms)</th>
<th>Disk (ms)</th>
<th>Network (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>reqHTML</td>
<td>9.62</td>
<td>-</td>
<td>0.01</td>
<td>1.37</td>
</tr>
<tr>
<td>reqJS1</td>
<td>0.85</td>
<td>-</td>
<td>0.01</td>
<td>0.14</td>
</tr>
<tr>
<td>reqJS2</td>
<td>4.8</td>
<td>-</td>
<td>0.01</td>
<td>1.19</td>
</tr>
<tr>
<td>reqJS3</td>
<td>0.64</td>
<td>-</td>
<td>0.02</td>
<td>0.34</td>
</tr>
<tr>
<td>reqViewRoutes</td>
<td>95.55</td>
<td>29.38</td>
<td>0.06</td>
<td>2.01</td>
</tr>
<tr>
<td>reqROUTING1</td>
<td>211.67</td>
<td>252.72</td>
<td>0.28</td>
<td>0.0002*</td>
</tr>
<tr>
<td>reqADD1</td>
<td>53.42</td>
<td>0.43</td>
<td>0.32</td>
<td>0.0002*</td>
</tr>
<tr>
<td>reqROUTING2</td>
<td>186.16</td>
<td>52.09</td>
<td>0.23</td>
<td>0.0002*</td>
</tr>
<tr>
<td>reqADD2</td>
<td>53.43</td>
<td>0.41</td>
<td>0.33</td>
<td>0.0002*</td>
</tr>
</tbody>
</table>

Based on above methodology and calculations presented, the service times for each entry of the Base-Scenario model are shown in Table 1. Here, the reqHTML service time on the AppServer is 9.62 ms, which corresponds to the sendHTML entry service time. Similarly, reqHTML service time on Disk is 0.01 ms – corresponding to disk1 entry. The service time on Network entries (n1c and n2c) due to reqHTML is 1.37/2 =0.685 ms each. Similar pattern follows for other request classes. Note that the service times for Network entries with (*) were found to have negative service times. The reqROUTING1, reqADD1, reqROUTING2 and reqADD2 classes had network service times of -0.00067, -0.00117, -0.00048 and -0.00117 (ms) respectively. This is considered as an anomaly and therefore, the network service times for each of these have been changed to 0.0002 ms, i.e. n1c and n1s each have service times of 0.0001 ms. Future work would include identifying the reason for such an anomaly.
7. Results

7.1 Base-Scenario Performance Results

Table 2: LQN Model Results - Base-Scenario

<table>
<thead>
<tr>
<th>Users</th>
<th>Throughput (sessions/s)</th>
<th>Response Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Load Test</td>
<td>LQN</td>
</tr>
<tr>
<td>1</td>
<td>0.12558</td>
<td>0.12567</td>
</tr>
<tr>
<td>2</td>
<td>0.24844</td>
<td>0.24864</td>
</tr>
<tr>
<td>4</td>
<td>0.49897</td>
<td>0.48027</td>
</tr>
<tr>
<td>6</td>
<td>0.79300</td>
<td>0.68434</td>
</tr>
<tr>
<td>10</td>
<td>0.92838</td>
<td>0.96871</td>
</tr>
<tr>
<td>20</td>
<td>1.03278</td>
<td>1.09454</td>
</tr>
<tr>
<td>30</td>
<td>1.05149</td>
<td>1.08406</td>
</tr>
<tr>
<td>40</td>
<td>1.02687</td>
<td>1.07721</td>
</tr>
<tr>
<td>50</td>
<td>1.04190</td>
<td>1.07176</td>
</tr>
<tr>
<td>80</td>
<td>1.01935</td>
<td>1.06507</td>
</tr>
<tr>
<td>120</td>
<td>1.01470</td>
<td>1.06017</td>
</tr>
<tr>
<td>150</td>
<td>1.05922</td>
<td></td>
</tr>
<tr>
<td>AVG ERROR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The model evaluation has been carried out for up to 150 users. This number was chosen because the AppServer task multiplicity is set to 150, which signifies the maximum number of running Apache processes. The model was evaluated on a dual Intel Xeon 3.0Ghz CPU with hyper-threading enabled, running on 2 GB of RAM. Taking an average of 10 runs for 120 users case, the Base-Scenario model was solved in 376ms while utilizing 47% of CPU.

Table 2 shows the throughputs and response times from Base-Scenario model evaluation and compares them with the measurements. The relative error% for the throughput from the model matches very closely with the actual results with an average error of 3.77%. The response time results show that for 6 users the error% is higher than other user counts, however as also seen the error% is very close to accurate for low numbers of 1 and 2 users and also for higher numbers of 10-120 users. For response times the average error is 12.15%.

Figure 6: LQN Base-Scenario (Throughput vs. Users)
Figure 6 and Figure 7 show the graphs for Throughput vs. Users, and Response Time vs. Users, respectively. As seen from both these graphs, the model closely follows the behaviour of the system’s performance. Figure 8 shows the $p_{Apache}$ CPU utilization from LQN evaluation that approaches about 100% utilization for more than 20 users. This is the hardware bottleneck and saturates before other resources.

![Response Time vs. Users](image1)

**Figure 7: LQN Base-Scenario (Response Time vs. Users)**

![pApache Util. vs. Users](image2)

**Figure 8: LQN Base-Scenario ($p_{Apache}$ Utilization)**

[29] provides rough error percentages for model validation. For multiple classes throughput error between 5% and 10% and for response time error between 10% and 30% are considered acceptable. For our model results, all the throughputs are very accurate with average error lower than 5%. Considering response times, the exceptions are 4 and 6 users which have high error however in real situations the system will probably witness higher number of users. One possible reason for this shortcoming is that CPU cache hit ratio is higher with lower user counts and CPU caching has not been considered in the model. Based on average response time error of about 12%, the model represents the actual results. After the model validation, the next steps will be to determine the improvements to the system.
The above results are helpful but they show that web application will not be able to support large number of users at satisfactory response times, showing poor performance. Based on the functionality provided by the web application, a reasonable performance objective is to sustain 40 to 50 users with a session response time of 12 seconds without think time. Considering such as objective, the web application performance has to be improved. Through previous analysis the bottleneck resource has already been identified to be pApache CPU. Next, we use intuitive modifications to the model such that performance objectives are met.

7.2 Attaining Performance Objectives

7.2.1 Overview

To achieve performance objectives, designs that ease the bottleneck have to be determined. From previous analysis it is found that the pApache server is the bottleneck for the MyBikeRoutes-OSM web application. To scale the system, options include adding threads, using a multi-processor system or using copies (replicas) of the server [20]. Since the bottleneck is hardware, addition of threads of AppServer task or the DB task will not be helpful. We evaluate the performance model for two possible solutions: 1. Multi-processor machine and, 2. Separating AppServer and DB task into separate identical machines. Following are the details of the modifications:

1. **Multiprocessor pApache**: The models have been evaluated for processors with multiplicity of two and four, which are referred as Base-Scenario-m2 and Base-Scenario-m4 models respectively. (Note that LQNS did not support PS scheduling for pApache multiprocessor. For two and four multiprocessors, First Come First Serve discipline was used).

2. **Separate machine for Database**: Instead of having both Application and Database software servers run on the pApache machine, the database-tier can be deployed on a separate and identical machine. The changes to the model include creating a new pDB processor which will be hosting the DB task and its entries while the Disk2 task on pDisk2 will now handle the disk I/O for the DB task. The previous pDisk is renamed as pDisk1, and the Disk task is renamed as Disk1 which will handle disk I/O for AppServer. The model will be referred as SeparateDB-Scenario. There are two assumptions made for this model. One is that service times for the disk I/O is divided by two to derive the service times of each entries of Disk1 and Disk2 tasks. Second, no network delay between the pApache and the pDB machines are considered. In a LAN environment the delay would not be very large, however, for very detailed study significant delays should be considered.
7.2.2 Performance Analysis

Figure 9, Figure 10 and Figure 11 show the throughput, response time and the utilization graphs comparing the Base-Scenario, Base-Scenario-m2, Base-Scenario-m4 and SeparateDB models. As depicted, Base-Scenario can sustain 22 users within a response time of 12 seconds, whereas the separate database can sustain 30 users, the dual processor 40 users, and quad-processor 80 users within this response time with respective throughputs of about 1 sessions/s, 1.5 sessions/s, 2 sessions/s and 4 sessions/s. From the model evaluations both dual and quad processors meet the performance objective set earlier. Although having a dedicated separate machine for the database was an attractive choice, the results suggest towards choosing from multiprocessor system to scale the system. Choice between dual or quad processor would be based on cost-benefit analysis, where it is to be determined if it sustaining 80 user with a higher throughput justifies spending extra on a quad-processor. For this work, we choose quad-processor considering that sustaining the increasing future demands on the web application will require more processing power.
8. Conclusions

A LQN performance model of MyBikeRoutes-OSM web application has been introduced in this work. For the base scenario described, performance results have been obtained from both measurement-based and model-based evaluations, the methodologies of which have clearly been explained and the results analyzed. JMeter was used for load testing and Utilization Law was applied to obtain service demand parameters. The model is validated by comparison with the load test results. With average error of 3.77% for throughput and 12.15% for response times the model is shown to capture the web application’s performance.

The analysis of the base model shows that the processor running the Application Server is the hardware bottleneck. To ease the bottleneck such that desired performance objectives are satisfied, modeling is used to represent the configuration options. The best configuration is found to use a multiprocessor machine for Application Server instead of having separate machines for Application and Database servers.

Future work includes incorporating caching in the performance model. Also, estimation of network delay using network utilization data from OS is a strong candidate for future research.

One of the strengths of adopting LQN model based approach as seen from this work is the short time required for model creation, manipulation and evaluation with good accuracy. Before modification of any system component, the effect of the change can be predicted and a decision can be reached just through quick model evaluation.

9. References


M. Tribastone, P. Mayer and M. Wirsing, "Performance prediction of service-oriented systems with layered queueing networks," in Leveraging Applications of Formal Methods,


Analysis of the Expected Number of Hops in Mobile Ad Hoc Networks with Random Waypoint Mobility

Osama Younes and Nigel Thomas
School of Computing Science, Newcastle University, UK
Email: {Osama.Younes | Nigel.Thomas} @ncl.ac.uk

ABSTRACT

The number of hops between the source and destination nodes is a key parameter in studying multi-hop ad hoc networks analytically. To the best of our known, there is no analytical work that considers the hop count of paths in MANETs in a random mobility environment. This paper presents a theoretical study for the expected number of hops between any random source-destination pair in multi-hop ad hoc networks where nodes move according to the random waypoint mobility model. The effects of network parameters such as node density, size of the network area, and node transmission range are studied. Simulation experiments for different network parameters have been conducted to validate the proposed analytical approach.

I. INTRODUCTION

A mobile ad hoc network (MANET) is a collection of wireless mobile nodes, moving with unpredictable mobility pattern, which dynamically form a network without any infrastructure elements. MANETs are self-organizing and self-configurable networks where the network is formed as soon as one of the nodes wants to send data to one or more of the other nodes. They are multi-hop wireless networks because the destination node is usually out of the transmission range of the source node. Therefore, the packets reach the destination after some hops on the intermediate nodes between the source and destination. As a result, the mobile nodes work as both sources and routers for other mobile nodes in the network. MANETs were initially designed to be used in the military and emergency relief applications. Lately, they have attracted researchers because of the need for flexible and efficient networks, so they have been utilised in many other applications [1].

In MANETs, the route or path is the sequence of mobile nodes which data packets pass through in order to reach the intended destination node from a given source node. Due to the mobility of nodes, mobile ad hoc networks have inherently dynamic topologies. Therefore the routes are prone to frequent breaks which reduce the throughput of the network compared to wired or cellular networks. Consequently, the route followed by packets to reach the destination varies frequently. This is a crucial factor that affects the performance of the network.

The hop count specifies the number of hops on the path between source and destination nodes. The analysis of the hop count in multi-hop networks is very important because it can provide design guidelines for ad hoc networks. It can be used in many applications such as, 1) estimation of the delivery ratio of packets, 2) with per hop delay, the end to end delay can be estimated, 3) with the number of simultaneous communications in the network, the network traffic can be estimated, 4) performance comparison between different multi-hop routing protocols, 5) evaluating the flooding cost and search latency for on-demand routing protocols and determining the optimal flooding strategy [2], 6) studying of connectivity and estimation of the capacity of the network. In addition, the hop count is a key parameter for performance analysis of multi-hop ad hoc networks using analytical methods.

Many studies have been issued to analyze how the performance of MANETs is affected by the hop count of paths [3-5]. The impact of hop count on searching cost and delay in ad hoc routing protocols has been investigated in [3]. Li et al. [4] have simulated the impact of different traffic patterns on the scalability of per node throughput. They showed that the network throughput deteriorates when the number of hops of the path
increases due to interference between nodes. In [5], Gamal et al introduced a scheme to analyze the impact of the transmission range, degree of node mobility and number of hops on the trade-off between the delay and throughput in fixed and mobile ad hoc networks.

Although the impact of the hop count of multi-hop paths on the performance of MANETs have been well recognized, there have been a very limited number of studies that focussed on the theoretical analysis of the number of hops in multi-hop paths in MANETs [6-9]. In [6], Jia-Chun and Wanjiun modelled the behaviour of packet forwarding on a multi-hop path for mobile ad hoc networks with high node density as circles centred at the initial location of the destination node. However, the results are not accurate because it is assumed that the progress per hop is equal to the transmission range. The relation between source-to-destination Euclidean distance and the hop count has been examined in [7]. The authors considered a greedy routing approach called Least Remaining Distance (LRD) which attempts to minimize the remaining distance to the destination in each hop. An analytical model for LRD and bounds on the number of hops for a given Euclidean distance between source and destination has been developed. Unfortunately, the accuracy of LRD approach is good only when the node density is very high.

In [8] an analytical model describing the hop count distribution for each source destination pair in multi-hop wireless networks has been developed. Also, the trade-off between flooding cost and search latency for target location discovery, used in most ad hoc routing protocols, has been evaluated. The drawback of this work is that it supposed that the distance between the source and destination nodes is uniformly distributed, and the impact of the size of the simulated network area is neglected. A mathematical model for the expected number of hops based on a Poisson randomly distributed network has been presented in [9]. The probability of n-hop count is derived and used to compute the expected number of hops. Unfortunately, all of these previous studies suppose that the nodes are stationary (no mobility) and are either uniformly or exponentially distributed over the network area.

Random mobility models, such as Random Way Point, Random Walk (random direction), Free Way, and Manhattan, play an important role in simulation of mobile ad hoc networks. To the best of our knowledge, there is no analytical work that computes the expected hop count of paths in MANETs in a random mobility environment. This is the motivation for our work, in which we develop a simple closed form analytical approach to estimate the expected number of hops between any source-destination pair in MANETs where the nodes are scattered in a square area and move according to the random waypoint mobility model (RWPMM). The RWPMM is selected because it is one of the most commonly used mobility models in MANETs studies. The hop count of paths for other mobility models can be investigated using the proposed approach.

For a given distance between the source and destination, to analytically compute the expected hop count, we need a packet forwarding algorithm which uses an optimization criteria to choose a relay node from neighbour nodes that minimizes the number of hops a packet has to traverse in order to reach the destination. We proposed a new packet forwarding strategy called Maximum Hop Distance (MHD) that attempts to minimize the number of hops needed for a packet to reach its destination by forwarding the packet to a neighbour node with the maximum forward distance in the direction of the destination.

To calculate the average number of hops analytically using MHD without the need to run time-consuming simulations, the probability density function of the distance between the source (or a relay node) and its neighbour nodes is derived using geometric probability. Then, it is used to compute the expected value for the maximum forward distance toward the destination which is essential to compute the expected value for the remaining distance to the destination. By recursive computing for remaining distance to the destination, the expected hop counts can be computed.

The number of hops between the source and destination in multi-hop ad hoc networks is jointly affected by many network factors, such as the node density, transmission range of nodes, mobility pattern, and the size of the simulated network area. The proposed approach is used to analyze the effect of these factors on the expected number of hops of paths in MANET. The main contribution of our work is twofold: (1) For the first time, an expression for the expected Euclidean distance between any source and destination nodes moving
according to the random waypoint mobility model is driven, (2) A novel analytical approach called Maximum Hop Distance (MHD) is proposed to compute the expected hop count for a given Euclidian distance between a source and destination.

MHD approach is a greedy routing approach which is inspired by LRD approach introduced in [7], but it is simpler and more accurate, as clear from the comparison between the two approaches in Section IV. In addition, MHD can be used for networks with low nodes density. The proposed process that uses the MHD approach to analytically compute the expected hop count between source and destination nodes moving according to the RWPM can be summarized as follows:

1) With a given network size, the expected distance between any source-destination pair is computed.
2) Compute the maximum expected distance (maximum forward distance) between any two nodes in the route for a given transmission range.
3) With a given node density, the per-hop progress is calculated.
4) By recursive computation, the expected number of hops for each packet to traverse from a source to a destination is derived.

The rest of this paper is organized as follows. In Section II we drive an expression for the expected Euclidean distance between any random source and destination nodes moving according to RWPM. Theoretical analysis of per hop progress and hop count is presented in Section III. In Section IV, the proposed approach is validated via simulation. Finally, the paper is concluded in Section V.

II. EUCLIDEAN DISTANCE BETWEEN THE SOURCE AND DESTINATION NODES

This section drives an expression for the expected Euclidean distance between any random source and destination nodes moving according to RWPM. First, we drive it for one dimension and then consider the square area.

I. Expected Distance on One Dimension

We first consider the distance between two nodes in a line segment. Suppose that two random points $X_1$ and $X_2$ are located in a line segment with length $L$. The distance between $X_1$ and $X_2$ is $S$. $X_1$ and $X_2$ are independent identically distributed random variables. According to [10], for the Random Waypoint Mobility Model the distribution of $X_1$ or $X_2$ is non-uniform at the long run. The probability distribution function of the location of a point $X_n$ moving on a line with length $L$ according to the RWPM is [10]

$$f_{X_n}(x_n) = \frac{6}{L^2} x_n + \frac{6}{L^3} x_n^2$$

Because $X_1$ and $X_2$ are i.i.d, the probability distribution function (pdf) of the location of the two points is

$$f_{X_1,X_2}(x_1,x_2) = f_{X_1}(x_1) \cdot f_{X_2}(x_2)$$

Where $0 \leq x_1 \leq L$ and $0 \leq x_2 \leq L$

The cumulative distribution function (CDF) of the distance $S = |x_2 - x_1|$ between the two points can be obtained by integration of $f_{X_1,X_2}(x_1,x_2)$ over the bounds of $S$ as follows:

$$P(s \leq d) = \iint f_{X_1,X_2}(x_1,x_2) \, dx_2 \, dx_1 = \iint f_{X_1}(x_1) \cdot f_{X_2}(x_2) \, dx_2 \, dx_1$$

$$= \int_0^d \int_0^{d+x_2} f_{X_1}(x_1) \cdot f_{X_2}(x_2) \, dx_2 \, dx_1 + \int_{d}^{L-d} \int_{x_2-d}^{L-d} f_{X_1}(x_1) \cdot f_{X_2}(x_2) \, dx_2 \, dx_1$$

171
\[ + \int_{L-d}^{d+x_2} \int_{x_2-d}^{d} f_{X_1}(x_1) \cdot f_{X_2}(x_2) \, dx_2 \, dx_1 \]

The integrations in the last equation can be evaluated yielding the following result:

\[ P(s \leq d) = \frac{12d}{5L} - \frac{4d^3}{L^3} + \frac{3d^4}{L^4} - \frac{2d^6}{5L^5} \]  

By definition the probability density function \( f(d) \) of \( d \) is given by the derivative of the Equation 1.

\[ f(d) = \frac{12}{5L} - \frac{12d^2}{L^3} + \frac{12d^3}{L^4} - \frac{12d^5}{5L^6} \]

II. Expected Distance on Two Dimension

Now, consider two random points \( X_1 \) and \( X_2 \) located in a square area of size \( L \times L \) with coordinates \((x_1, y_1)\) and \((x_2, y_2)\), respectively. If \( d \) is the distance between \( X_1 \) and \( X_2 \), \( d \) is given by

\[ d = (x_1 - x_2)^2 + (y_1 - y_2)^2 \]

If \( f_{dx} \) and \( f_{dy} \) are the pdf of the events \((x_1 - x_2)^2\) and \((y_1 - y_2)^2\), respectively. Then, the pdf of the distance \( d \) is given by the convolution of \( f_{dx} \) and \( f_{dy} \) as follows:

\[ f_{xy}(d) = \int f_{dx}(z) \cdot f_{dy}(d - z) \, dz \]  

Let \((x_1 - x_2)^2 = dx\), then the CDF of \( dx \) can be obtained by using Equation 1 by substituting \( d \) by \( \sqrt{dx} \). We get the following:

\[ F(dx) = \frac{12\sqrt{dx}}{5} - \frac{4\sqrt{dx^3}}{L^3} + \frac{3dx^2}{L^4} - \frac{2dx^3}{5L^5} \]

The pdf of \( dx \) is obtained as follows:

\[ f(dx) = \frac{\partial}{\partial dx} F(dx) = \frac{6}{L^4} - \frac{6dx^2}{5L^6} + \frac{6}{5L\sqrt{dx}} - \frac{6\sqrt{dx}}{L^3} \]

In the same way, \( f(dy) \) can be obtained.

Because the domain of \( d \) is divided in two parts, \( 0 < d \leq L^2 \) and \( L^2 < d \leq 2L^2 \), there are two cases for Equation 2 which are

\[ f_{xy}(d) = \begin{cases} 
I_1(d) = \int_{0}^{d} f_{dx}(z) \cdot f_{dy}(d - z) \, dz & 0 < d \leq L^2 \\
I_2(d) = \int_{d-L^2}^{L^2} f_{dx}(z) \cdot f_{dy}(d - z) \, dz & L^2 < d \leq 2L^2 
\end{cases} \]  

The first integral \((I_1)\) in Equation 3 can be solved and with some simplification and reduction of its terms, we obtain the following
The second integral ($I_2$) can also be solved, but the result is very long which can’t be provided here because of the limited space. Either $I_1$ or $I_2$ can be easily evaluated numerically. Because $d$ is the square distance between $X_1$ and $X_2$, the expected distance between the two nodes ($\delta$) is given by

$$E(\delta) = \int_0^{2L^2} \sqrt{d} f_{xy}(d) \, dd = \int_0^{\sqrt{2L}} \delta f_{xy}(\delta) \, d\delta$$  \hspace{1cm} (4)$$

Where $\delta = \sqrt{d}$. The integral $I_1(d)$ in Equation 3 constitutes $\left(1 - 2 \frac{\sqrt{L} - \sqrt{d}}{L^2}\right) \times 100$ (about 99%) of the expected value of the distance between the two nodes. Therefore $E(\delta)$ can be approximated to

$$E(\delta) \approx \int_0^{\sqrt{2L}} \delta I_1(\delta) \, d\delta = \frac{(179684 - 47619n)L}{75075}$$ \hspace{1cm} (5)$$

For uniformly distributed nodes in a square area of size $L \times L$, the expected distance between two random nodes is [11]

$$E(\delta) = 0.5214054 \, L$$

Figure 1 shows the expected Euclidian distance between any random source and destination nodes that are uniformly scattered or moving according to the RWPMM in a square area, plotted against different values of the side length of the square area ($L$). It is clear that the expected distance between the two nodes in the case of the RWPMM is much less than uniform distributed nodes, especially for large value of $L$. This is because the spatial distribution of nodes moving according to the RWPMM at long run is non-uniform, since the probability that a node is located at the centre of the square area is high, and it reaches zero at the border of the area [12].
III. EXPECTED HOP COUNT

To analyze the expected hop count in MANETs where the nodes move according to the RWPMM, we consider any source node $S$ that tries to send its packet to a destination node $D$, as shown in Figure 2, where the circle with radius $R$ around any node indicates the transmission area. The expected distance between any source and destination nodes is $d$. If $d$ is greater than the transmission range $R$, which is equal for all nodes in the network, the source uses the intermediate nodes to forward the packets to the destination through two or more hops. The routing protocol searches all routes to the destination and chooses the shortest one. If the source has $N_h$ neighbour nodes (the nodes within the transmission range), the routing protocol in $S$ will choose the closest neighbour node to the destination (e.g., the node $A$ in Figure 2) to work as the next relaying node to forward the packet in the path. The number of hops in the path depends on the distance between the source and destination nodes ($d$) and the remaining distance to the destination per hop (per hop progress).

![Figure 2: Packet forwarding in a multi-hop path](image)

To compute the expected hop count analytically without the need to run time-consuming simulations, a greedy routing approach called Maximum Hop Distance (MHD) is proposed. MHD is a packet forwarding algorithm that uses the maximum forward distance toward the destination as the optimization criterion to choose the relay node from neighbour nodes that minimizes the number of hops a packet has to traverse in order to reach the destination. The geometric probability is used to drive the pdf of the distance between the source (or a relay node) and its neighbour nodes which is used to compute the expected value for the
maximum forward distance toward the destination. Also, the expected remaining distance to the destination, which is used to calculate the expected hop count, is computed using the geometric probability.

MHD approach succeeds if at least one router is located towards the destination (shaded regions shown in Figure 2) in each hop to prevent back forwarding of packets. Otherwise it fails. For example, as shown in Figure 2, for node $S$ and $C$, node $A$ and $E$ are located in the grey half circle towards the destination $D$ to forward the packets from $S$ and $C$, respectively, to the destination. Intuitively, to keep the connectivity of the route, each node needs at least two neighbour nodes; one is for the previous hop and the other is for the next hop. Therefore, the node density must exceed a certain threshold to ensure the route and network connectivity. In [13] and [14], the authors showed that the average number of neighbour nodes required to ensure one-connectivity is eight. Hence, in all validation scenarios, introduced in Section IV, the total number of nodes in the network ($N$) and size of the network area are chosen to make the number of neighbour nodes is eight.

Let $M$ be the potential router that used to forward the packets from $S$ to $D$ for the first hop, as shown in Figure 3. Also, let $r$ and $X$ be the distance between the source and the router $M$ (the maximum forward distance) and the remaining distance from $M$ to $D$, respectively. The pdf and expected value for $r$ and $X$ must be derived to compute the expected hop count.

First, we drive the pdf of the maximum forward distance $r$ that is used by MHD approach as the optimization criterion to minimize the hop count. Suppose that there are $n$ forwarding neighbour nodes ($M_1, \ldots, M_n$) distributed over the half circle towards of the destination $D$. The distances and angles from the source $S$ to the neighbour nodes are $\rho_i$ and $\alpha_i$, where $i = 1\ldots n$, as shown in Figure 4. For simplicity of the analysis the neighbour nodes are assumed to be uniformly distributed around $S$. So, the expected value of $n$ equals to the half of the expected number of neighbour nodes ($N_h$). The value of $N_h$ for the RWPMM can be computed using the methods introduced in [15]. The pdf of the distance ($\rho$) between $S$ and the neighbour nodes is

$$f_{\rho}(\rho) = \frac{2\rho}{R^2}$$

Where $0 \leq \rho \leq R$ and $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$. Integrating the last Equation over $\alpha$ gives the pdf of $\rho$ as

$$f_{\rho}(\rho) = \frac{2\rho}{R^2}$$

![Figure 4: The distances between $S$ and neighbour nodes](image)

To minimize the hop count to the destination, the neighbour node with the maximum distance ($\rho_{\text{max}}$) from the source $S$ is chosen to forward the packets. According to [16], because $\rho_1, \ldots, \rho_n$ are i.i.d random variables each with pdf $f_{\rho}(\rho)$, the pdf of $\rho_{\text{max}}$ is

$$f_{\rho_{\text{max}}}(\rho) = n F_{\rho}(\rho)^n f_{\rho}(\rho) = 2n \frac{\rho^{2n-1}}{R^{2n}}$$

175
Where $F_{\rho}(\rho)$ is the CDF of $\rho$. By definition, the expected value of $\rho_{\text{max}}$ is

$$E(\rho_{\text{max}}) = \int_0^R \rho \, f_{\rho_{\text{max}}}(\rho) \, d\rho = \frac{2n}{2n + 1} R$$

Therefore, the expected distance $r$ between the source $S$ and router $M$, shown in Figure 3, is given by

$$r = E(\rho_{\text{max}}) = \frac{2n}{2n + 1} R \quad (6)$$

The resulting function for $r$ for a given $R = 250$ or $220$ and increasing values of $n$ is shown in Figure 5. Clearly, for a given transmission range $R$, for small values of $n$, $r$ increases rapidly. For large values of $n$, $r$ may reach $R$. Therefore, increasing the node density decreases the expected hop count, but it increases the interference between neighbour nodes. Equation 6 can be used for analysis of the distance between the source and other nodes in the path which is important to study the survivability of the path.

To drive an expression for the remaining distance, we consider that the router $M$ may be located at any point on the circumference of a half circle with a radius $r$ computed using Equation 6, as shown in Figure 3. Let $M$ is located at random angle $\alpha$. So, the domain of $\alpha$ is $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$. The remaining distance $X$ can be described using a pdf as follows

$$f_{\alpha}(\alpha) = \frac{1}{\pi} \quad -\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$$

The probability that $\alpha$ is smaller than a given value $a$ can be computed by the integral of the last equation as

$$P_{\alpha}(\alpha \leq a) = \int_{-a}^{a} f_{\alpha}(\alpha) \, d\alpha = \frac{2}{\pi} \, a \quad (7)$$

From geometry, $d^2 + r^2 - 2 \, r \cos \alpha = X^2$. Therefore, by substitution in Equation 7, we get CDF of $X$ as

$$F_X(X) = P_X(X \leq x) = \frac{2}{\pi} \arccos\left(\frac{d^2 + r^2 - X^2}{2 \, d \, r}\right)$$

The last equation is differentiated to get its pdf of $X$ as
By definition, the expected value of \( X \) can be deduced from Equation 8 as follows

\[
f_X(X) = \frac{2X}{\pi d r \sqrt{1 - \left(\frac{d^2 + r^2 - X^2}{2 d r}\right)^2}}
\]

By definition, the expected value of \( X \) can be deduced from Equation 8 as follows

\[
X_r = E(X) = \int_{d-r}^{\sqrt{d^2+r^2}} X \cdot f_X(X) \, dX
\]

The last equation can be easily evaluated numerically.

After getting the remaining distance \( X_r \) from the router \( M \) to the destination for the first hop, to get the expected number of hops, the current distance to the destination \( d \) in the next hop is replaced by the remaining distance \( X_r \) obtained using Equation 9. Then, the process is repeated and the hops are counted until \( X_r \) falls below the transmission range \( R \). The following procedure summarizes this process:

Step 1: Set the inputs \( N \), \( R \), and \( L \)

Step 2: Set the number of hops count to \( hop\_count = 0 \)

Step 3: Compute the expected distance between the source and destination \( (d) \) using Equation 4

Step 4: Compute the expected maximum distance between the source and router \( (r) \) using Equation 6

Step 5: If \( d \leq R \), then  \( hop\_count = \frac{d}{R} \), go to the End

Step 6: Set \( hop\_count = hop\_count + 1 \)

Step 7: The remaining distance between the router and destination \( (X_r) \) is computed using Equation 9

Step 8: If \( X_r \leq R \), then \( hop\_count = hop\_count + \frac{X_r}{R} \)

Step 9: If \( X_r > R \), then set \( d = X_r \), and go to step 6

Step 10: End

IV. VALIDATION

In this section, the proposed approach is validated by comparing the theoretical and simulation results. We first validate the theoretical analysis of the expected Euclidean distance between any random source and destination nodes, introduced in Section II, by simulation. For this validation we used the MobiSim tool [17] that uses topological characteristics to analyze and manage the mobility scenarios for ad hoc networks. We consider a simulation scenario consists of a square system area of a side length \( L \) that varies from 400 to 1000 m. A set of 200 nodes are uniformly scattered in the square area and move according to the RWPMM. Every node moves towards the destination point with a velocity chosen uniformly from 0 to maximum speed \( V_{max} \). When it reaches the destination it chooses and moves towards a new destination in a similar manner. The maximum moving speed is set to 20 m/s. A zero pause time was chosen to make the nodes move all the time. All nodes have a radio range of 250m.

The expected distance between any source-destination pair is computed by taking the average of the distances between every pair of nodes. We generated many different mobility scenarios until we get 95% confidence interval and maximum relative error 1%. Figure 6 shows simulation and analysis results for the expected distance between any two nodes for varying values of the side length of the square area. The comparison between analytical and simulation results shows the accuracy of the proposed analysis.

To validate the proposed theoretical analysis and procedure to compute the expected number of hops for a packet transmission in ad hoc networks, we performed a series of simulation tests using NS-2 [18]. The
simulation settings consist of a network with a square area. The side length of the square area varies from 700 to 1600m. The maximum speed of a node is set to 20 m/s. The simulation time is set to 1500 seconds. To be sure that the average number of neighbour nodes is greater than 8 nodes, the node density is varied depending on the size of the system area. To illustrate its effect on the expected number of hops, the transmission range is considered to be 200 or 250m. The RWP mobility patterns used in all simulation tests are generated using setdest tool which is a node movement generator tool implemented by the current ns-2 version.

![Figure 6: Expected distance for different size of network area](image)

Table 1: Analytical and simulation results for expected hop count for increasing values for the side length of the network area where $R = 200$ or $250$ m

<table>
<thead>
<tr>
<th>$L$</th>
<th>Expected Number of Hops</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R = 200$</td>
<td>$R = 250$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sim</td>
<td>Ana</td>
<td>Sim</td>
</tr>
<tr>
<td>700</td>
<td>2.61</td>
<td>2.73</td>
<td>2.01</td>
</tr>
<tr>
<td>800</td>
<td>2.93</td>
<td>2.84</td>
<td>2.14</td>
</tr>
<tr>
<td>900</td>
<td>3.19</td>
<td>2.97</td>
<td>2.69</td>
</tr>
<tr>
<td>1000</td>
<td>3.65</td>
<td>3.8</td>
<td>2.94</td>
</tr>
<tr>
<td>1100</td>
<td>4.10</td>
<td>3.92</td>
<td>3.18</td>
</tr>
<tr>
<td>1200</td>
<td>4.58</td>
<td>4.76</td>
<td>3.57</td>
</tr>
<tr>
<td>1300</td>
<td>4.93</td>
<td>4.86</td>
<td>3.86</td>
</tr>
<tr>
<td>1400</td>
<td>5.28</td>
<td>4.99</td>
<td>4.16</td>
</tr>
<tr>
<td>1500</td>
<td>5.79</td>
<td>5.82</td>
<td>4.52</td>
</tr>
<tr>
<td>1600</td>
<td>6.19</td>
<td>5.93</td>
<td>4.63</td>
</tr>
</tbody>
</table>

The number of hops between nodes can be computed on the fly during simulation runs. But this method consumes a long time (may be days) especially with a large number of nodes and network area size. Alternatively, we used an object called General Operations Director (GOD) which is implemented with setdest tool and used to manage the shortest path information between nodes. For the whole simulation period, GOD is aware of any changes in mobile wireless network topology. GOD is an omniscient observer, where it is used to store global information about the topology of the network. This global information is not totally available to any node, but partial information is provided to each node when needed. GOD is used to store an array of the optimal path length in hops between every pair of nodes. This information is used to analyze and develop ad hoc network routing protocols.
For the same network settings, the expected number of hops is computed by averaging the number of hops between every pair of nodes. We generated many mobility patterns for the same network settings and computed the expected number of hops until the relative error is less than 2% for 95% confidence interval. Table 1 shows the simulation and theoretical results for the expected number of hops for two different values of transmission range ($R = 200$ or 250) and increasing values for side length of the square area of the simulated network. As shown in Table 1, for a given transmission range, the expected number of hops increases significantly as the network size increases because of increasing the expected distance between the source and destination. In addition, as expected, the expected number of hops decreases with increasing of the transmission range because of increasing of the per hop progress. As can be seen in Table 1, the theoretical results are accurate compared to simulation results.

To compare the LRD and MHD approaches, Table 2 shows the expected number of hops computed using the two approaches and simulation for the same network settings used to validate the MHD approach where $R=150$. Compared with simulation results, it is clear that the accuracy of the MHD approach is much better than LRD approach, as shown in Table 2. The expected number of hops computed using the LRD approach is much less than simulation especially for long routes. This is because the LRD approach supposes that the routers with the minimum remaining distance to the destination constitute the shortest path to the destination which is only true when the node density is very high.

Table 2: Comparison between simulation and LRD and MHD results for expected hop count for increasing values for $L$ where $R = 150$

<table>
<thead>
<tr>
<th>$L$</th>
<th>Sim</th>
<th>LRD</th>
<th>MHD</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>3.37</td>
<td>2.39</td>
<td>3.74</td>
</tr>
<tr>
<td>800</td>
<td>3.83</td>
<td>2.62</td>
<td>3.86</td>
</tr>
<tr>
<td>900</td>
<td>4.40</td>
<td>2.86</td>
<td>4.75</td>
</tr>
<tr>
<td>1000</td>
<td>4.84</td>
<td>3.40</td>
<td>4.86</td>
</tr>
<tr>
<td>1100</td>
<td>5.42</td>
<td>3.63</td>
<td>5.75</td>
</tr>
<tr>
<td>1200</td>
<td>5.83</td>
<td>3.88</td>
<td>5.86</td>
</tr>
<tr>
<td>1300</td>
<td>6.35</td>
<td>4.4</td>
<td>6.74</td>
</tr>
<tr>
<td>1400</td>
<td>6.72</td>
<td>4.64</td>
<td>6.85</td>
</tr>
<tr>
<td>1500</td>
<td>7.06</td>
<td>4.89</td>
<td>6.99</td>
</tr>
<tr>
<td>1600</td>
<td>7.35</td>
<td>5.42</td>
<td>7.83</td>
</tr>
</tbody>
</table>

Compared to simulation results, the computation time required for theoretical analysis is trivial. For example, in the case of $N = 250$, $R = 150$, $L = 1600$, simulation time = 1500s, for 95% confidence interval and 2% relative error, the time required for generating the mobility patterns and computing the expected hop count is about 28.2 hours, whereas the time required for theoretical analysis is less than 2 seconds, where the simulation and theoretical analysis was conducted on desktop workstation equipped with 2.6GHz (Intel Q9400 Core 2 Quad) processor, 4 GB of RAM and Ubuntu Linux version 8.10.

V. CONCLUSION

In this paper, we presented a theoretical analysis for the expected number of hops in mobile ad hoc networks where nodes move according to random waypoint mobility model. The proposed approach can be used to analyze the hop count for other mobility models. It depends on computing the expected distance between source and destination nodes, per hop distance, and per hop progress which are used to compute the expected hop count. The proposed approach has been validated using network simulation for different network parameters. The impacts of the transmission range, node density, and size of network area on the hop count have been investigated.
REFERENCES


