Contributions to the Performance Modeling of Computer Networks

Thomas Begin

Académie de Lyon
Université Claude Bernard Lyon 1

Habilitation à Diriger des Recherches

préparée au Laboratoire de l'Informatique du Parallélisme
ENS de Lyon, UCBL, CNRS et Inria

Spécialité : Informatique

Contributions to the Performance Modeling of Computer Networks

par

Thomas Begin

Manuscript rapporté par :

André-Luc Beylot, Professeur, Université Toulouse ..... Rapporteur
Andrzej Duda, Professeur, Université Grenoble Alpes ..... Rapporteur
Nihal Pekergin, Professeur, Université Paris-Est ..... Rapporteure

Soutenue le 10 décembre 2018 à Lyon, devant le jury composé de :

André-Luc Beylot, Professeur, Université Toulouse ..... Rapporteur
Andrzej Duda, Professeur, Université Grenoble Alpes ..... Rapporteur
Nihal Pekergin, Professeur, Université Paris-Est ..... Rapporteure
Mohand-Saïd Hacid, Professeur, Université de Lyon ..... Examinateur
Catherine Rosenberg, Professeur, University of Waterloo ... Examinatrice
Fabrice Valois, Professeur, Université de Lyon ..... Examinateur

Contents

1 Introduction **3**

1.1 Purpose of this Habilitation Thesis 4
1.2 Performance Modeling of Computer Networks 4
1.3 A Few Words about Performance Evaluation 6
1.4 Studied Topics 8
1.5 Selected Contributions 9
1.6 Outline of the Thesis 10

2 DPDK-based Virtual Switches **13**

2.1 Research Context 14
2.2 Outline 15
2.3 Motivation 15
2.4 Description of a vSwitch 16
2.4.1 Context and definition 16
2.4.2 DPDK library 17
2.4.3 Scenarios 19
2.5 Modeling a vSwitch as decoupled queues with server vacation 20
2.5.1 System notation 20
2.5.2 Performance parameters of interest 21
2.5.3 Decomposition principle 22
2.6 Solution to the queues and their performance parameters 23
2.6.1 Markovian assumptions 23
2.6.2 Markov chain model associated with each RX queue 23
2.6.3 Estimating the service rate μ_i 24
2.6.4 Estimating the switch-over rate β 25
2.6.5 Estimating the vacation rate α_i 25
2.6.6 Fixed-point solution 26
2.6.7 Computing the performance parameters 26
2.7 Accuracy of the Proposed Approach 27
2.8 Examples of Application 29
2.9 Related Works 31
2.10 Conclusions 33

3 IEEE 802.11 networks **37**

3.1 Research Context 38

3.2 Outline . . . 39
3.3 Motivations . . . 39
3.4 State of the Art . . . 40
3.5 System Description . . . 41
3.6 Model and its Solution . . . 44
3.6.1 Decomposing into subnetworks . . . 44
3.6.2 Solving each subnetwork as one or more Markov chain(s) . . . 45
3.6.3 Combining subnetwork solutions . . . 49
3.7 Numerical Results . . . 50
3.7.1 Model validation . . . 50
3.7.2 Modeling complexity . . . 54
3.7.3 Possible application: Channel assignment . . . 54
3.8 Conclusions . . . 57

4 Reduced State Description **63**

4.1 Research Context . . . 64
4.2 Outline . . . 65
4.3 Motivation and Related Work . . . 65
4.4 Model with Infinite Buffer . . . 67
4.5 Model with Finite Buffer and State Dependencies . . . 69
4.6 Numerical results . . . 71
4.6.1 Accuracy for the Mean Number in the queue . . . 72
4.6.2 Loss Probability with finite buffers . . . 73
4.6.3 Wait Probability . . . 74
4.6.4 Speed of Asymptotic Convergence for queues with unrestricted buffer 75
4.6.5 Speed of Convergence of the Fixed-Point Iterative Solution . . . 75
4.6.6 Model with State Dependencies . . . 75
4.6.7 Computational Complexity . . . 76
4.6.8 Example of comparison with simple approximations . . . 77
4.7 Conclusions . . . 78
4.8 Appendix . . . 82

5 Conclusions **83**

5.1 "Missing" Contributions . . . 84
5.2 The Importance of Accuracy in Modeling . . . 85
5.3 Work methods and Good practices . . . 87
5.4 Scientific Challenges and Prospects for the Future . . . 89

6 Co-Authored Publications **95**

Résumé: Les réseaux informatiques sont devenus une partie intégrante de nos sociétés modernes. Au cours des deux dernières décennies, le nombre d'internautes est passé de 147 millions à plus de 4 milliards et de nouvelles applications sont apparues (p. ex. messagerie instantanée, voix sur IP, réseau social, vidéo à la demande). Pour faire face à ces nouvelles demandes, les réseaux ont fortement évolué, en augmentant leurs performances et leurs services et en offrant un accès sans fil à leurs utilisateurs. Le développement rapide du standard IEEE 802.11 (commercialisé sous le nom de WiFi) depuis ses débuts en 1997 en est un bon exemple. Plus récemment, le déploiement du NFV (Network Function Virtualization) devrait permettre une gestion plus flexible et efficace des réseaux en remplaçant le matériel spécialisé et propriétaire par du logiciel exécuté sur du matériel banalisé. Le renouvellement constant des technologies réseaux et le besoin croissant de qualité de service rendent crucial la modélisation des performances des réseaux.

Cette thèse d'habilitation décrit une sélection des mes contributions scientifiques dans le domaine des réseaux informatiques et de l'évaluation de performances. Le Chapitre 1 comprend une description des mes activités de recherche et une discussion sur quelques défis majeurs à venir.

Le Chapitre 2 est consacré à mes contributions dans le contexte du NFV. Il présente un modèle analytique de type file d'attente pour évaluer les performances d'une fonction réseau (implémentée comme un logiciel) qui commute des paquets, c'est-à-dire un commutateur virtuel (vSwitch). Le modèle calcule des mesures de performances comme l'occupation des tampons, le taux de perte et le temps de séjour des paquets dans le vSwitch. La solution proposée est conceptuellement simple, peu coûteuse en calcul et généralement précise. Un exemple basé sur un cas d'étude réel illustre comment le modèle peut aider à bien régler les paramètres d'un vSwitch.

Le Chapitre 3 traite du problème de modélisation des réseaux sans fil basé sur IEEE-802.11 (WLANs). Il décrit une méthode de modélisation qui estime le débit atteint par chaque Point d'Accès (AP) en fonction du graphe de conflit du WLAN, de la charge des AP, de la taille des trames et des taux de transmissions. L'approche proposée repose sur une stratégie type Diviser pour régner qui sépare le problème initial en plusieurs sous-problèmes dont les solutions sont combinées pour obtenir celle du problème initial. Le modèle est en général précis et son utilisation peut aider à allouer efficacement les canaux de fréquence aux APs lors de la configuration d'un WLAN.

Le Chapitre 4 étudie les files $Ph/Ph/c$ et $Ph/Ph/c/N$ qui sont des modèles communs pour les systèmes multi-serveurs. Il présente une approximation simple pour calculer les probabilités stationnaires de ces files grâce à une description d'état réduite qui permet d'éviter la croissance combinatoire des états inhérente à la description d'état classique. Le nombre d'équations à résoudre dans cette approche croît linéairement avec le nombre de serveurs et le nombre de phases dans la distribution du temps de service. La précision de l'approximation est en général très bonne et tend à s'améliorer pour un nombre important de serveurs.

Cette thèse se termine avec le Chapitre 5 qui présente des méthodes de travail et des bonnes pratiques fondées sur ma propre expérience ainsi que quelques perspectives de recherche pour le futur.

Abstract: Computer networks have become part of our daily life and, to some extent, a key element of our modern society. Over the last two decades the number of Internet users have surged from 147 million to over 4 billion and many applications have emerged (e.g., instant messaging, voice over IP, social networking, video on demand). To meet these new demands, computer networks have undergone tremendous changes, augmenting their performance, their services and providing access through wireless communications. A case in point is the fast development of the IEEE 802.11 standard (commercially known as WiFi) that experienced many changes since its release in 1997. More recently, NFV (Network Function Virtualization) emerged as a promising paradigm to bring flexibility and efficiency to the networks by replacing specialized and proprietary hardware with software ran over commodity hardware. Because of the constant renewal of networking technologies and increasing needs for Quality of Service (QoS), the performance modeling of computer networks remains a challenging and crucial issue.

This habilitation thesis describes a selection of my scientific contributions in the scope of computer networks and performance evaluation. Chapter 1 comprises a description of my research activities and a discussion on some potential forthcoming challenges.

Chapter 2 is devoted to my contributions in the context of NFV. More precisely, it presents an analytical queueing model to evaluate the performance of a network function (implemented as a software) commuting packets, aka a virtual switch (vSwitch). The model delivers performance metrics such as the buffer occupancy, the loss rate and the sojourn time of packets in the vSwitch. The proposed solution is conceptually simple, computationally efficient and generally accurate. An example based on a real-life case study illustrates how the model can help in determining an adequate setting of the vSwitch parameters.

Chapter 3 deals with the issue of modeling IEEE-802.11 based Wireless Local Area Networks (WLANs). It describes a performance modeling method that estimates the attained throughput of each Access Point (AP), as a function of the WLAN's conflict graph, the AP loads, the frame sizes, and the link transmission rates. The modeling approach employs a Divide-and-Conquer strategy that breaks down the original problem into multiple sub-problems, whose solutions are then combined to provide the solution to the original problem. The model accuracy is generally good and its application may help to assign AP channels when configuring a WLAN.

Chapter 4 considers $Ph/Ph/c$ and $Ph/Ph/c/N$ queues that can be viewed as a common model of multi-server facilities. It introduces a simple approximate solution for the equilibrium probabilities in such queues based on a reduced state description in order to circumvent the well-known combinatorial growth of the number of states inherent in the classical state description. The number of equations to solve in our approach increases linearly with the number of servers and phases in the service time distribution. The overall accuracy of the proposed approximation appears very good, and tends to become excellent as the number of servers increases.

This thesis ends with Chapter 5 that provides work methods and good practices derived from my experience, as well as a number of prospects for the future.

Remerciements

Je souhaite d'abord adresser mes plus sincères remerciements à mes rapporteurs que sont André-Luc Beylot, Andrzej Duda et Nihal Pekergin pour le temps qu'ils ont consacré à la lecture et à l'évaluation de ce manuscrit. Je tiens également à remercier Catherine Rosenberg, Mohand-Said Hacid et Fabrice Valois pour avoir accepté d'être membres de mon jury.

Durant ces 10 années passées au laboratoire LIP, j'ai eu la chance de travailler avec des collègues d'une grande qualité scientifique mais aussi humaine. Je remercie Alexandre, Bruno, Isabelle, Paulo, Anthony, Serge ainsi que les (actuels ou anciens) postdocs et doctorants que sont Roy, Thiago, Nghi, Guillaume et Marija. Plus généralement, je souhaite remercier l'ensemble des personnes ayant appartenu à l'équipe RESO ou appartenant à l'équipe Dante. Je souhaite également remercier mes collègues du département Informatique de l'UCBL avec qui j'assure mes enseignements ainsi que l'ensemble du laboratoire LIP.

Je remercie également ma famille et en particulier Margit qui m'a toujours encouragé et soutenu ainsi que mes parents qui ont toujours cru en mon projet professionnel.

Enfin je remercie Charlotte dont les ronronnements et les câlins ont mâtiné mes journées de rédaction de ce manuscrit.

Preamble

A few words about me

My interest in Sciences really started in 1998 thanks to an excellent Math teacher who managed to make Mathematics much more compelling and interesting to me. I obtained an Engineer Diploma in Telecommunication and Network from ISEP (Institut supérieur d'électronique de Paris, France) in 2003, as well as a Master of Computer Science from Université Pierre et Marie Curie (UPMC, France) in 2005. I spent the next 3 years preparing a Ph.D. in Computer Science, Telecommunication, and Electronics at the LIP6 lab under the supervision of Prof. Serge Fdida (UPMC, France) and Assoc. Prof. Bruno Baynat (UPMC, France). At the end of 2008, I defended my Ph.D. and then I started a PostDoc position under the guidance of Prof. Alexandre Brandwajn at the University of California, Santa Cruz (UCSC, USA). In 2009, I was hired on an Assistant Professor position at the Université Claude Bernard Lyon 1 (UCBL, France). During the 2015-2016 academic year, I was on research leave in the lab of Prof. Azzedine Boukerche at the University of Ottawa (uOttawa, Canada). This leave was funded through a CNRS grant.

Since the completion of my Ph.D., I have been involved in several Research Projects and I have co-supervised the thesis of 4 Ph.D. students (Dr Doreid Ammar, Dr Shubhabrata Roy, Dr Thiago Abreu, Dr Nghi Nguyen) and of 12 M.Sc. students. I became member of the Program Committees for well-established conferences such as IEEE LCN and ACM MSWiM. Lately, I was appointed as the Technical Program Co-Chair for the Algotel conference 2019 edition.

Chapter 1

Introduction

Contents

1.1 Purpose of this Habilitation Thesis **4**

1.2 Performance Modeling of Computer Networks **4**

1.3 A Few Words about Performance Evaluation **6**

1.4 Studied Topics **8**

1.5 Selected Contributions **9**

1.6 Outline of the Thesis **10**

1.1 Purpose of this Habilitation Thesis

This *Habilitation Thesis* is built on my research activities since the end of my Ph.D. It comprises a selection of my Scientific Contributions in the scope of Computer Networks and Performance Evaluation, as well as work methods and good practices derived from my experience over these years. It ends with a number of prospects for the future.

The goal of this thesis is to provide a summary of my research activities that is both mature and critical but also accessible to any Computer Scientists (having or not a background in my domain of interest). Scientific contributions are presented at high-level of description and some technical details may be omitted for the sake of conciseness. The reader can refer to the corresponding published paper(s) for more details. On the other hand, a special attention is given to other aspects of these contributions. Thus, I'll discuss the following points for each contribution: explaining the motivations and research context that led to this work, assessing fairly the strengths and limitations of the proposed contribution, describing possible extensions and follow-up works, discussing its potential impact on the related field and community, and finally, summing up the learned experiences.

Virtually all my scientific contributions result from joint-efforts with other researchers and students. Therefore, I consider each of these works as a collective achievement in which my role (as a contributor) has varied. In Computer Science, as in many scientific disciplines, noticeable works generally comprise several aspects (e.g., proposing new algorithms and modeling techniques; analyzing and understanding the studied system; coding and measuring; defining the case studies and scenarios; reviewing the existing associated literature; writing the corresponding reports and papers). Although I contributed to each of these aspects, my degree of commitment may vary from one contribution to another. And I sometimes find it hard to identify the main contributor(s) on a given aspect as good ideas often arise from brainstorming sessions and intense discussions. Without further ado, I summarize my research activities.

1.2 Research Activities: Performance Modeling of Computer Networks

My research activities are primarily concerned with the *Performance Modeling of Computer Networks*. To provide a better understanding of my field of interest, I start by discussing each of these terms.

Modeling

Modeling refers to the art of using mathematical and theoretical tools to describe the behavior of a system. Theoretical models can take many different forms, including statistical models, differential equations, Markov Chains, game theoretic models, or queueing models. Any model generally relies on a set of assumptions that eases its analysis so that the model can be viewed as an abstraction of the system. A model may help to explain a system, to help its parameterization, to study the effects of its different components, and to make predictions about its behavior.

With this short definition of a model in mind, I would like to discuss one recurrent concern about modeling. Almost any system existing in Computer Science has been the subject of numerous models. Although this may appear as disproportionate and unneeded at first

glance, there are also legitimate reasons for this abundance. First, as the common saying goes, *There are no good models for a system, but there are good models for the purpose they serve*. This saying conveys that, depending on the specific aspect being studied on a given system, different models must be considered since a detailed description, or conversely an abstraction, of some of the system features and components may then be desired. Therefore, when considering a model, one needs to also consider its scope, which defines its domain of applicability. Second, aside from their scope, models may differ by their complexity and accuracy. These two seemingly disjoint aspects may be considered jointly since modeling generally involves a trade-off between complexity and accuracy. Broadly speaking, adding complexity to a model usually improves its realism. On the other hand, model complexity tends to make the model difficult to understand and to solve. In practice, ouf of two models having about the same level of accuracy, the simplest one is quite straightforwardly the most preferable, following the Occam's razor principle (aka law of parsimony) However, things get more complicated when dealing with two models with a different level of accuracy. Depending on the intended application for the model, one may favor the simplicity over the accuracy, or vice versa. Third, models of a same system may also differ by the mathematical formulation and tools used to their analysis. Overall, models of a same system may differ in a number of aspects (including their scope, complexity, and accuracy) explaining why many have been proposed in the literature.

Performance

In Computer Science, the word *Performance* denotes the efficiency of a computer system, and was informally defined by Arnold Allen as follows: *"How well is the computer doing the work it is supposed to do?"*. When evaluating the performance of a computer system, a number of parameters (or metrics) are generally used to determine the result. A non-exhaustive list of the performance parameters includes throughput, completion time, transmission time, response time, sojourn time, loss rate, blocking probability, resource utilization, CPU usage, queue length, buffer occupancy, and availability.

Computer Networks

A *Computer Network* comprises all components that enable communication between computers or other digital devices (e.g., phones, sensors). Computer networks make use of different types of links (e.g., wired, wireless), comprise specialized devices (e.g., routers, switches, access points), follow different topology (e.g., bus, star, ring, mesh, fully connected, tree), involve various protocols (e.g., IEEE 802.3 commercially known as Ethernet, IEEE 802.11 commercially known as WiFi), are of different sizes and scales (ranging from personal area networks (PAN) and local area networks (LAN) up to wide area networks (WAN) and backbone networks).

Computer networks have become part of our daily life and, to some extent, a key element of our modern society. They have undergone tremendous changes over the last two decades with the number of Internet users surging from 147 million to over 4 billion and the emergence of many applications (e.g., instant messaging, voice over IP, social networking, video on demand). To meet these new demands, computer networks have significantly evolved, augmenting their performance, their services and providing access through wireless communications. A case in point is the fast development of the IEEE 802.11 standard (commercially known as WiFi) that undergone many changes since its release in 1997.

More recently, NFV (Network Function Virtualization) emerged as a promising paradigm to bring flexibility and efficiency to the networks by replacing specialized and proprietary hardware with software ran over commodity hardware.

With these definitions in mind, and to illustrate the range of topics falling in the scope of the performance modeling of computer networks, I briefly mention a couple of them that I actually studied these last years: analyzing the efficiency of a virtual switch within a wired computer network; understanding the behavior of an IEEE 802.11-based network; predicting the congestion occurrence on a given network link; reproducing the workload volatility of a Video on Demand system.

1.3 A Few Words about Performance Evaluation

Theoretical or Applied Research?

In my opinion, the art of Performance Evaluation lies at the crossroads between Theoretical and Applied Research. On the one hand, it typically involves the use of probability, Markov Chains, elements of queueing theory as well as methods of statistical data analysis such as hypothesis testing, confidence intervals, regression and correlation analysis. On the other hand, evaluating the performance of a system requires both an expertise of the system under study and skills in coding simulations and/or in performing measurements from real-life experiences. Therefore, it is quite common for scientific contributions in the domain of performance evaluation to include both theoretical results and practical findings. For instance, the accuracy of a proposed theoretical model may be validated with real-life measurements collected from the system under study. This duality between theory and practice hardens the process of performance evaluation, but it also greatly participates to make it interesting.

New Challenges for Performance Evaluation

Since its beginning with the earliest theoretical work on its logical foundations in 1930's, Computer Science has quickly grown and expanded into a broad range of fields including Computer Architecture, Computer Algebra, Programming and Algorithms, Information Theory, Complexity Theory, Computer Networks, Distributed Systems, Operations Research, Artificial Intelligence, etc.

As for the field of Performance Evaluation, the "golden age" of this research community probably occurred in the 1960's and 1970's with the discovery of many major groundbreaking results such as the product-form solutions for Jackson Networks [A12, A13], the Gordon-Newell theorem [A11], the BCMP networks [A2], etc. Besides, in the early 1960's, theoretical works conducted by Kleinrock [E15] to model the performance of packet-switched networks have underpinned the development of the ARPANET (which later gave birth to the current Internet). Another major step for performance evaluation was done in 1971 with the introduction of the central server model [E6] by Buzen. While earlier models had studied separately the performance of individual system components such as processors and I/O devices (e.g., disks), the central server model brought major progress as it provided a convenient means to represent the interactions taking place among components, and ultimately, to determine the overall system performance [A8]. Although this community has kept producing remarkable contributions up to this day, I believe that the excitement and the publicity around this community have somehow

faded, and that today's scientific largest efforts and biggest progress have been shifted to other fields of Computer Science. Nonetheless, the rapid changes of computer and communication systems with the continuous release of new devices, protocols, technologies, and usages, fuels the need for new theoretical models, ensuring thereby substantial work and recurrent challenges for the performance modeling community.

For Performance Evaluation to regain a central role in the development of Computer Science, I believe that new approaches specially designed to cope with the complexity of today's computer and communication systems are needed. Indeed, most of today's computer and communication systems involve many components (e.g., be it the number of nodes in a network, the number of CPU cores of a machine, or the number of Virtual Machines hosted on a server). Additionally, today's systems often exhibit highly variable distributions (e.g., in their inter-arrivals, service times, file lengths, call holding times) that may invalidate simplifying assumptions such as the memoryless assumption (e.g., [A9, A19, A17, A10]). Overall, I think that the growing complexity of today's systems hinders existing modeling approaches and tools. For instance, let us consider a given Markov chain modeling a computer or communication system that does not possess any closed-form solution. In order to obtain the associated steady-state probabilities, one typically relies on the use of numerical solutions (after having transformed this problem into a linear system formed by the balance equations). Depending on the structure of the underlying stochastic matrix (e.g., sparsity, eigenvalues), direct solutions can handle linear systems with up to several hundreds of states before numerical instabilities (largely due to rounding errors) occur. On the other hand, iterative solutions significantly postpone the onset of instabilities and can handle several hundreds of thousands of states [A20, A5]. Nonetheless, because the state space of underlying Markov chains typically grows combinatorially with the system parameters, those solutions may reach their limitations when modeling a large real-life system. Another example includes the solution to the steady-state probabilities of a multi-server queue with general service times. Several approximate solutions have been proposed in the literature, but most, if not all, have been validated on a small number of servers, typically less than a dozen. It may raise concerns as, in practice, the number of CPU cores on a machine or the number of Virtual Machines on a server can easily exceed 16.

Efforts have been made to devise new approaches to alleviate, or better yet, to circumvent these dimension limitations. For example, the mean field theory is well suited to deal with large and complex stochastic systems in which a large number of small individual components interact with each other. This approach approximates the effect of all the other individuals on any given individual by a single averaged effect, thus reducing a many-body problem to a one-body problem. The mean field theory has been successfully applied to large computer and communication systems (e.g., [A3]) as well as to queueing theory problems to evaluate the asymptotic behavior when the number of queues goes to infinity (e.g., [A1, A4]). Another interesting lead to deal with high-dimensional systems is the reduced state description that has been proposed lately by Pr. Brandwajn and me [A6], which is the subject of Chapter 4. This approach drastically reduces the complexity of analyzing multi-server queues by describing explicitly the state of only one server. In fact, not only is the reduced state description well-behaved for queues with a large number of servers, but its accuracy actually increases with the number of servers.

Relations of Performance Evaluation with the other fields

Any researcher could arguably benefit from developing additional skills in another field of Computer Science. In the particular case of a performance modeling researcher, it is my belief that he or she would benefit the most from having additional skills in Optimization and, to a lesser extent, in Complexity Theory. Indeed, having a solid background in Complexity Theory is often mandatory to clearly and accurately assess the complexity of a model. On the other hand, Optimization routines (e.g., Linear programming, Combinatorial optimization, Stochastic programming, Dynamic programming) will find many applications in the use of performance models and can greatly help to increase their value.

Conversely, in my opinion, many researchers and engineers would benefit from developing skills in Performance Evaluation. And by that, I don't necessarily mean getting a sharp knowledge of Markov chains, queueing theory, stochastic processing or renewal theory. Despite a commonly held opinion, I think that Performance Evaluation does not restrict to these theoretical aspects. In fact, it also includes more practical aspects such as double checking results with Little's law [A16], differentiating transient and stationary regimes, estimating proper averages for a given performance parameter, running discrete-event simulations, that can prove helpful in many other research works. For instance, while discrete-event simulations are widely used to validate new solutions for computer networks, some researchers have shown that they are not always properly used [A15, A18], likely due to insufficient knowledge in performance evaluation.

1.4 Studied Topics

Over the last ten years, my research activities have covered various topics. I attempt to exhaustively list them here from the least recent to the most recent: Higher-order distributional properties in queueing systems [G5],[H18, H12],[I4]; Efficient solution to queueing systems [G7, G8, G9, G1, G10, G12, G11], [H19, H20, H21, H11], [I5]; WiMAX performance modeling [H23]; Admission control [H5, H4, H7, H6, H8],[I1, I2]; VoD buzz workload [G13], [H22, H29], [I8]; Multi-constrained routing [H27]; IEEE 802.11 networks [G4, G15], [H28, H3, H1, H2, H30]; High-Level modeling [H13]; Estimator for the end-to-end delay [H24, H25, H26]; DPDK-based virtual switches [G2],[H9, H32, H31]; vehicular networks [G14],[H10, H16]. Note that associated publications are listed in Chapter 6 (pages 95 - 100).

To give another perspective on the studied topics, Figure 1.1 depicts a word cloud of the most recurring words in the 450 pages of publications I co-authored since the end of my Ph.D. in 2008.

This relatively large number of studied topics is mostly the result of the numerous opportunities that I was given since the end of my Ph.D. Indeed, following my PostDoc, I joined the RESO team (LIP lab, France) whose main focus was on wired networks. I was fortunate to get immediately involved in several existing national (MISSION, RESCUE) and European (SAIL) projects that covered different topics. A couple of years later, I integrated the Dante team (resulting from the merger of RESO with another team) whose main concerns are dynamic networks. I contributed to the setting and completion of two scientific projects (DISCO and Reflexion) that explored new aspects of computer networks. Furthermore, since the end of my Ph.D., I have maintained long-lasting collaborations with Pr. Alexandre Brandwajn (UCSC, USA) and Ass. Pr. Bruno Baynat (UPMC, France). I also developed new collaborations with colleagues of my team: Pr.

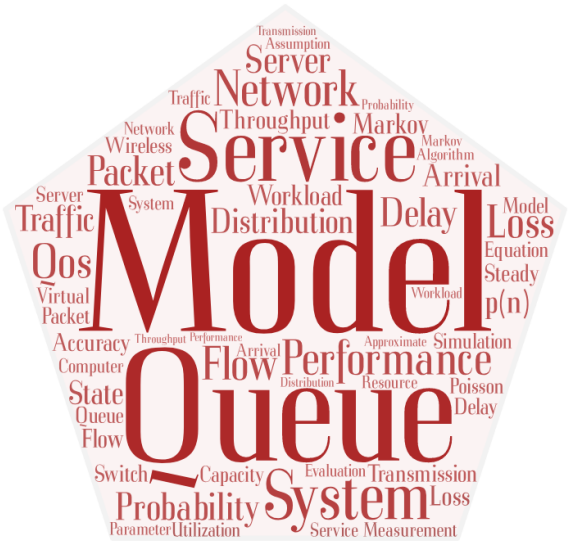


Figure 1.1: Word cloud created from the content of all my publications since 2008.

Isabelle Guérin Lassous, Pr. Anthony Busson and DR Paulo Gonçalves, as well as with colleagues abroad such as Pr. Azzedine Boukerche (uOttawa, Canada) and researchers from UPC, Spain. Last but not least, this large number of topics was also fuelled by my desire (or at least, by the lack of refusal) to renew my research topics and to develop new technical skills (the latest being the study of Vehicular Networks and the field of Graph Signal Processing).

The common thread running through all the topics I studied during these years holds in the scientific approach used to address any of them. This approach proceeds in a series of stages as follows: (i) understanding the system under study; (ii) modeling the system performance of interest; (iii) validating the accuracy of the model against simulation or better yet real-life experiences, otherwise return to stage (ii); (iv) applying the model on several use cases to demonstrate the abilities of the model. Another common aspect to all my contributions is that I always avoided the use of complex mathematics, and instead looked for simpler yet sensible ways to circumvent the complexity of a problem. I discuss this point in more detail in the Conclusions of this thesis (Chapter 5).

1.5 Selected Contributions

This thesis highlights a selection of my scientific contributions to the field of Performance Modeling of Computer Networks. This selection includes the following three topics: DPDK-based virtual switches; IEEE 802.11 networks; Reduced state description.

Therefore, some contributions - such as Higher-order distributional properties in queueing systems; Admission control; VoD buzz workload; Estimator for the end-to-end delay; and Vehicular networks - are not discussed in this Thesis. I picked out the presented contributions mostly based on three factors: their degree of matureness, their investment in time and energy, and their potential impact on the community.

1.6 Outline of the Thesis

In the next chapter, I discuss my contribution to the performance modeling of DPDK-based virtual switches. Chapter 3 deals with the issue of modeling IEEE 802.11-based wireless networks. In Chapter 4, I present a new way of addressing the inherent complexity of some queueing systems using a reduced state description. Chapter 5 concludes this thesis. All my co-authored publications are listed in Chapter 6.

References for Chapter 1

[A1] F. Baccelli, F. Karpelevich, M. Y. Kelbert, A. Puhalskii, A. Rybko, and Y. M. Suhov. A mean-field limit for a class of queueing networks. *Journal of Statistical Physics*, 66(3-4):803–825, 1992.

[A2] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, 1975.

[A3] M. Benaim and J.-Y. Le Boudec. A class of mean field interaction models for computer and communication systems. *Performance Evaluation*, 65(11-12):823–838, 2008.

[A4] A. Bobbio, M. Gribaudo, and M. Telek. Analysis of large scale interacting systems by mean field method. In *IEEE QEST*, 2008.

[A5] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[A6] A. Brandwajn and T. Begin. Reduced complexity in *M/Ph/c/N* queues. *Performance Evaluation*, 78:42–54, 2014.

[A7] J. P. Buzen. *Queuing Network Models of Multiprogramming Ph. D.* PhD thesis, Thesis, Harvard University, Cambridge, MA, 1971.

[A8] J. P. Buzen. From the Central Server Model to BEST/1©. In *Performance Evaluation: Origins and Directions*, pages 485–489. Springer, 2000.

[A9] A. Feldmann and W. Whitt. Fitting mixtures of exponentials to long-tail distributions to analyze network performance models. *Performance Evaluation*, 31(3-4):245–279, 1998.

[A10] D. R. Figueiredo, B. Liu, V. Misra, and D. Towsley. On the autocorrelation structure of TCP traffic. *Computer Networks*, 40(3):339–361, 2002.

[A11] W. J. Gordon and G. F. Newell. Closed queuing systems with exponential servers. *Operations Research*, 15(2):254–265, 1967.

[A12] J. R. Jackson. Networks of waiting lines. *Operations Research*, 5(4):518–521, 1957.

[A13] J. R. Jackson. Jobshop-like queueing systems. *Management Science*, 10(1):131–142, 1963.

[A14] L. Kleinrock. *Message delay in communication nets with storage*. PhD thesis, Massachusetts Institute of Technology, 1963.

[A15] S. Kurkowski, T. Camp, and M. Colagrosso. MANET simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, 2005.

[A16] J. D. Little. A proof for the queuing formula: L= λ w. *Operations Research*, 9(3):383–387, 1961.

[A17] K. Park, G. Kim, and M. E. Crovella. Effect of traffic self-similarity on network performance. In *Performance and Control of Network Systems*, volume 3231, pages 296–311. International Society for Optics and Photonics, 1997.

[A18] K. Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee. On credibility of simulation studies of telecommunication networks. *IEEE Communications magazine*, 40(1):132–139, 2002.

[A19] V. Paxson and S. Floyd. Wide area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking (TON)*, 3(3):226–244, 1995.

[A20] W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.

Chapter 2

Performance Modeling of DPDK-based Virtual Switches

Contents

2.1 Research Context **14**
2.2 Outline **15**
2.3 Motivation **15**
2.4 Description of a vSwitch **16**
2.4.1 Context and definition 16
2.4.2 DPDK library 17
2.4.3 Scenarios 19
2.5 Modeling a vSwitch as decoupled queues with server vacation **20**
2.5.1 System notation 20
2.5.2 Performance parameters of interest 21
2.5.3 Decomposition principle 22
2.6 Solution to the queues and their performance parameters **23**
2.6.1 Markovian assumptions 23
2.6.2 Markov chain model associated with each RX queue 23
2.6.3 Estimating the service rate μ_i 24
2.6.4 Estimating the switch-over rate β 25
2.6.5 Estimating the vacation rate α_i 25
2.6.6 Fixed-point solution 26
2.6.7 Computing the performance parameters 26
2.7 Accuracy of the Proposed Approach **27**
2.8 Examples of Application **29**
2.9 Related Works **31**
2.10 Conclusions **33**

Abstract

With the development of the Network Function Virtualization (NFV) paradigm, networking functions will gradually move from specialized and proprietary hardware to open-source software run over a Virtual Machine (VM) deployed on commodity hardware. Arguably, the foremost network service is packet switching. In this regard, Open vSwitch (OVS) is the most prominent open-source solution implementing a virtual switch (vSwitch), i.e., a software relaying packets. Besides, DPDK (Data Plane Development Kit) is a set of specialized libraries to speed up the packets processing. In particular, DPDK allows packets to be processed by batches.

In this chapter, we present an analytical queueing model to evaluate the performance of a DPDK-based vSwitch. Such a virtual equipment is represented by a complex polling system in which packets are processed by batches. To reduce the complexity of the associated model, we develop a general framework that decouples the polling system into several queueing subsystems, each one corresponding to a given CPU core. We resort to servers with vacation to capture the interactions between subsystems. Our model delivers performance metrics such as the buffer occupancy, the loss rate and the sojourn time of packets in RX queues. Our proposed solution is conceptually simple, computationally efficient and generally accurate. We illustrate how our model can help in determining an adequate setting of the vSwitch parameters using several real-life case studies.

2.1 Research Context

In the wake of the ANR (French National Research Agency) funded *DISCO*[1] project that focused on Software-Defined Networking (SDN) issues, I was involved in another ANR funded project, called *Reflexion*[2] whose consortium included Thalès, Orange, Inria, UPMC, 6WIND, ENS Lyon, and Telecom ParisTech. The project ran from 2015 until 2017 and I served as the site leader for ENS Lyon. Reflexion dealt with the hot topic of Network Virtualization underpinned by the new paradigms of SDN and Network Function Virtualization (NFV). Note that these two terms are defined in Section 2.3.

One major research axis in the Reflexion project was to study the performance of network services run in a virtualized environment. Indeed, while companies like 6WIND sell NFV solutions to customers (typically network operators), the actual performance of these solutions, as well as their resource setup, are still raising some concerns. My contributions to Reflexion essentially pertained to this axis of research and involves a tight collaboration with UPMC and 6WIND. More precisely, they resulted from a joint work with Bruno Baynat (UPMC) Guillaume Artero Gallardo (ENS Lyon), Zidong Su (ENS Lyon), Vincent Jardin (6WIND), and to a lesser extent, with Thomas Delbono (6WIND), Amine Kherbouche (6WIND), Thomas Monjalon (6WIND), and Tanguy Pomas (6WIND). As part of the Reflexion project, I co-authored three conference papers [B1, B24, B25] and one journal paper (recently accepted) [B2].

[1] `http://anr-disco.ens-lyon.fr/`
[2] `http://anr-reflexion.telecom-paristech.fr/`

2.2 Outline

The remainder of this chapter is organized as follows. In Section 2.3, we briefly discuss the scientific context around this work as well as its motivations. Section 2.4 describes the internal behavior of a DPDK-based vSwitch as well as the real-world inspired scenarios that motivate our study. In Section 2.5, we present the main principles underlying our modeling framework for a vSwitch. Section 2.6 details its solution in order to obtain the vSwitch performance parameters. We study the accuracy of our models in Section 2.7 and we explore their potential applications in Section 2.8. We discuss the related works in Section 2.9. Section 2.10 concludes this chapter.

2.3 Motivation

Server virtualization has become ubiquitous in the modern Information Technology (IT) environment. Decoupling virtual servers from physical servers helps to leverage the computing resources, and brings important gains in scalability and agility. More recently, the virtualization of networks has attracted much attention. Scalability, agility, and multi-tenancy (with the concept of network slicing) are the envisioned improvements that virtualization will bring to traditional computer networks.

This trend towards more flexible networks, often known as "softwarization", is driven by two main paradigms: Software-Defined Networking (SDN) and Network Function Virtualization (NFV). The former aims at removing all the decision-making networking functions from network nodes and regrouping them into a (set of) controller(s). Thus, network nodes, such as routers, switches, load-balancers, firewalls, etc. are replaced by appliances receiving their instructions directly from the controller(s) (using a standard interface like OpenFlow [B18]). On the other hand, NFV refers to the gradual move of network functions from dedicated hardware to commodity hardware running specialized software. For example, functions such as routing, switching, load-balancing, firewalling, etc. will be run as software on standard x86 servers, and are thus referred to as Virtualized Network Functions (VNFs). Note that VNFs may be executed directly by the hypervisor or within a Virtual Machine (VM) (or a container). In any case, to allow communications between the VMs (of a same physical server) and the rest of the physical network, the hypervisor can create a virtual switch (aka vSwitch) that logically connects the VMs to the outside world.

While software-based solutions are generally viewed as more flexible than their hardware counterparts, the network softwarization raises concerns about its expected performance. To address this issue, a consortium including companies like Intel and 6WIND has devised Data Plane Development Kit (DPDK) [B3]. DPDK is an open-source project and works as a specialized library for x86, ARM and PowerPC processors. In particular, it enables vSwitches to accelerate the processing of incoming packets by (i) balancing the incoming flow of packets over all the vSwitch CPU cores, (ii) avoiding unnecessary re-copy of the packets, (iii) keeping all operations out of the OS kernel and, instead, within the user space and (iv) processing packets by batches, thereby having a better use of the CPU cache. While other libraries exist, DPDK has become a de facto standard for vSwitches. Nonetheless, due to the relative novelty and complexity of virtual switches, their performance modeling (e.g., to analytically derive estimates of throughput, loss rate, latency) remains a challenging problem. We believe that an analytical model can provide some helpful guidelines by suggesting adequate values for the large number of parameters that

can be adjusted in a virtual switch.

In this chapter, we investigate the performance of a virtual switch, i.e., a software relaying packets (possibly after modifying their content) between the ports of VMs or containers hosted on a same physical machine and the rest of the physical network. We assume that the vSwitch is equipped with DPDK. We propose a conceptually simple and easy-to-implement modeling approach for evaluating customary performance parameters of a vSwitch such as its mean throughput, its mean latency, its loss rate as well as the level of utilization of its CPU cores. We consider several examples inspired by real scenarios and we assess our model accuracy by comparing its predictions with the actual values delivered by a discrete-event simulator. To illustrate the application of our model, we explore the effects of changing a vSwitch configuration (e.g., batch sizes) as well as adjusting the vSwitch resources (number of allocated CPU cores) to satisfy a given Service Level Agreement (SLA) policy (e.g., zero-loss).

2.4 Description of a vSwitch

2.4.1 Context and definition

Like traditional switches, a vSwitch commutes packets between its ports but, unlike them, it operates as a software typically run by the hypervisor of the physical server. In general, a vSwitch has access to a set of logical CPU cores and to a set of physical and logical ports. Logical ports connect to ports of VMs hosted on the same physical machine while physical ports are associated with existing ports on the physical server. An example is given in Figure 2.1. For example, in a cloud computing context, a physical server may host several VMs that are interconnected using a vSwitch, which itself is executed by the hypervisor. In an SDN/NFV-based network, vSwitches (software running on commodity hardware) are replacing specialized hardware devices such as switches, firewalls, load-balancers, routers, and other middleboxes. In addition to commuting packets between their ports, vSwitches may also perform other operations like filtering, header editing, content encrypting and deep packet inspection. In fact, vSwitches may use all the headers from layer 2 up to layer 5 (and not just 2 as it is commonly the case for a traditional switch) so that they are also referred to as virtual multi-layer switches. Note that the physical machine running the vSwitch typically hosts VMs or containers so that the vSwitch has to share the physical resources with them.

The two prominent solutions to create a vSwitch in a hypervisor are OVS [B19] and FD.io VPP [B6]. Note that both are open-source projects and that VPP is the open-source version of Cisco's Vector Packet Processing. Aside from open-source implementations, a couple of proprietary virtual switch solutions have been released; e.g., by Cisco (Nexus 1000V) and VMware (vSphere Distributed Switch).

A vSwitch mainly comprises three types of components: (i) network interface cards (NICs) that altogether provide a total of N input/output (I/O) ports, (ii) a set of (logical) CPU cores that are in charge of processing the packets coming from the different I/O ports, and (iii) memory to store packets waiting for their processing, be it from the CPU cores or from the NICs. As a side note, note that packets are moved across these components through PCI buses.

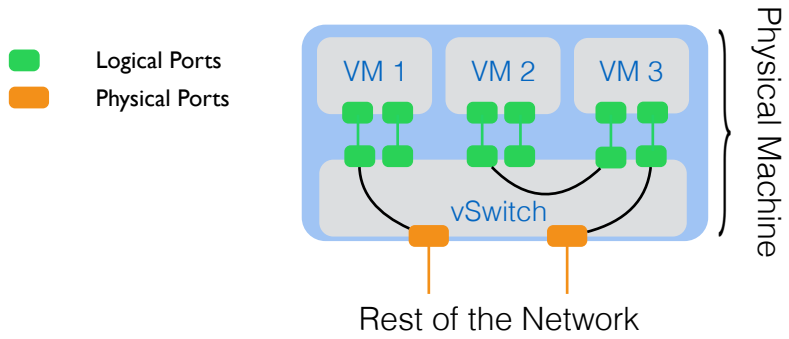


Figure 2.1: vSwitch connecting three VMs with two physical ports.

2.4.2 DPDK library

To let vSwitches deal with high rates of packets, different techniques, e.g., Netmap [B22], OpenOnload [B20], PacketShader [B21] and DPDK [B3], have been developed to provide a faster packet processing. DPDK is an open-source project and works as a specialized library for x86, ARM and PowerPC processors. It is developed by a consortium comprising companies like Intel and 6WIND. Note that DPDK is integrated to the most prominent vSwitch solutions, namely OVS and VPP, making it a de facto standard for vSwitches. Hence, we focus our study on a vSwitch equipped with the DPDK library [B3]. DPDK makes use of several means for accelerating the processing of packets.

No packets recopy

DPDK avoids the recopy of packets. Thanks to the use of a shared memory, CPU cores are able to process packets without recopying them in their associate memory. For a deeper understanding of these mechanisms, the reader can refer to the work of Scholz [B23].

Operations performed in the user space

Another means of DPDK for accelerating the packet processing is to run the associated operations in the user space and not within the kernel as is done by default. By doing so, DPDK avoids the overhead of CPU interrupts that result in additional delays in processing packets.

Balancing the load across all the CPU cores

Although DPDK allows various configurations in the polling of the several vSwitch ports by the multiple CPU cores, we consider here its standard configuration, using the so-called "Poll Mode Driver", which is known as the most versatile and efficient (unless in specific scenarios). First, one CPU core, aka the "master" core, is entirely dedicated to the control and management of the vSwitch while the other CPU cores are devoted to the packet processing. Let C denote the number of CPU cores devoted to the packet processing, i.e., not including the master core. Second, DPDK aims at uniformly distributing the load originating from each port across the C CPU cores. Said differently, each CPU core contributes to processing packets coming from each port. More precisely, modern NICs

perform load balancing by letting each of their ports dispatch incoming packets into C separate logical queues, called RX queues. This dispatching step is typically carried out through the application of a hash function on the packet headers (such as the Receive Side Scaling (RSS) used in DPDK) and aims at ensuring an even balance of the incoming packets among the RX queues as well as at accelerating packet processing by directing packets belonging to the same flow to the same RX queue. As a result, each RX queue is assigned to a single CPU core while each CPU core handles as many RX queues as the total number of ports in the vSwitch. We denote by K the RX queue size, i.e., the maximum number of packets that can be queued simultaneously in it. Then, once a CPU core ends up processing a packet, the packet is (logically) forwarded from its RX queue to a TX queue associated to the appropriate output port. At this stage, the packet is pending for transmission on the next link and does not need any further CPU core processing resource. Figure 2.2 illustrates this mapping between CPU cores, ports and RX queues.

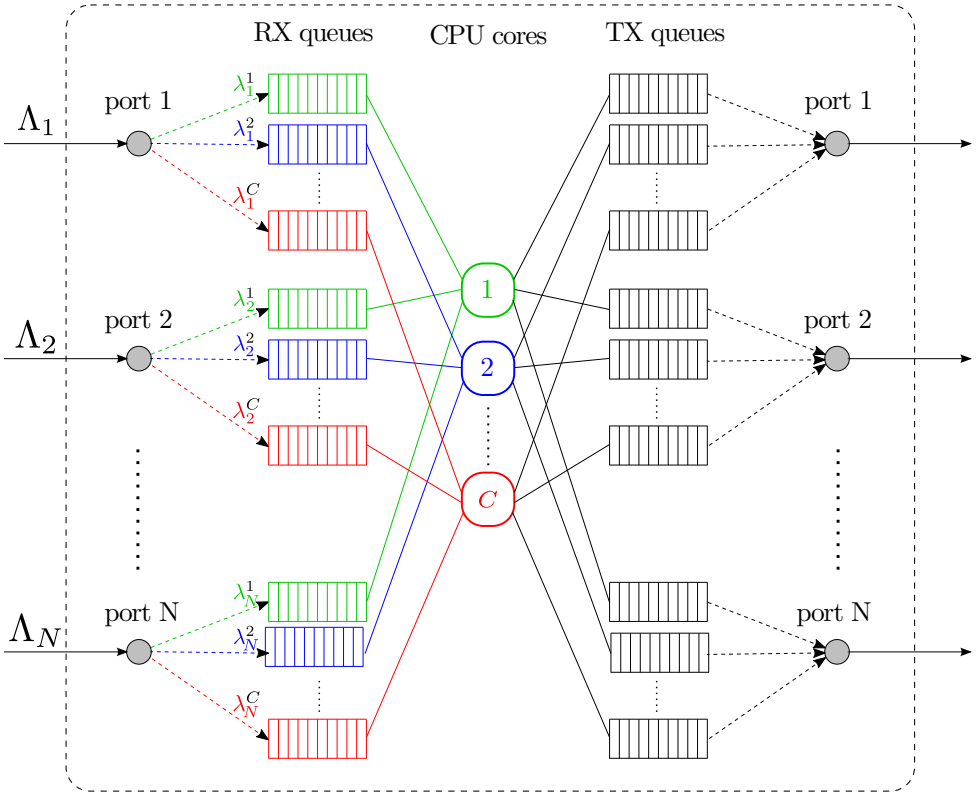


Figure 2.2: Internal architecture of a vSwitch with N I/O ports and C CPU cores.

Processing packets by batches

CPU cores poll their associated RX queues in a cyclic order (i.e., in a round-robin fashion). However, for the sake of performance, DPDK enables CPU cores to serve a batch of packets on the same RX queue before switching to the next RX queue. We denote by T_S the mean switch-over time taken by a core to switch from its current RX queue to the next one, and by M the maximum size of the batch. When the batch size is set to M, a CPU core can prefetch up to M packets on one RX queue and then it processes them in a run-to-completion manner. Note that packets that enter the RX queue while the CPU core has already started its service are not served in this round, and they have to wait until the

core revisits this queue. In the queueing theory literature, this discipline is known as a gated M-limited policy [B16].

Batching packets by groups of M packets tends to increase the overall efficiency of a vSwitch. Indeed, when a CPU core handles packets belonging to the same batch, chances are that the needed instructions are found in the CPU cache, which lowers the average processing time of a packet. We denote by T_H and T_M the average time needed by a CPU core to process a packet when the set of instructions is found (cache Hit), respectively not found (cache Miss), in the cache. Note that T_H is typically significantly smaller than T_M, say around an order of magnitude or so. Let T_R indicate the average time needed by the CPU core to forward a packet from an RX queue to a TX queue over a PCI bus (once the CPU core processing has ended). As a result, the total time needed by a core to serve a given packet is either $T_M + T_R$ (in case of a cache miss) or $T_H + T_R$ (in case of a cache hit). In general, the former case is more likely to occur if the considered packet is among the first packets of a batch (the cache is likely to be "cold") whereas the latter has more chances to happen for the subsequent packets of a batch as they will benefit from the cache information. In addition, processing packets by batches also increases efficiency by reducing the total number of switch-over times (as a CPU core does not switch to a different RX queue upon the completion of a single packet processing).

Despite the enhancements brought by DPDK, vSwitches are subject to performance issues, in particular if the incoming load is too large. Given the transmission speed of lines and the current transfer rates of PCI buses and memory (SDRAM), the bottleneck of a vSwitch, if any, is likely to occur during the processing of packets in RX queues due to the limited CPU resources. Therefore, we concentrate our modeling efforts on the interactions between the CPU cores, the RX queues and the ports.

2.4.3 Scenarios

Throughout this chapter, we consider three scenarios inspired by features of real vSwitches to demonstrate the accuracy and the abilities of our modeling approach. For the sake of simplicity and without loss of generality, we assume that the considered vSwitches comprise CPU cores running each at 3GHz, that RX queues are set to store up to $K = 128$ packets, that the mean packet size is 1000 bytes, and that the switch-over times T_S are equal to 1ns. We also assume that the dispatching function performed on incoming packets at the ports is well-behaved (see Section 2.4.2) so that every CPU core undergoes the same performance allowing us to restrict our analysis to only one of them. Note that this last assumption does not mean that ports are equally loaded. Note also that all the numerical values used to specify our three scenarios are derived from real-life experiments conducted in 6WIND lab.

Scenario 1 - Simple forwarding

In our first scenario, we consider a case where a vSwitch is simply forwarding incoming packets between its ports based on their link layer headers and does not provide any further services. Said differently, the vSwitch behaves similarly to a regular switch. We assume that the vSwitch operates on a small-scale network so that its flow table is relatively small. More precisely, we assume that there is a total of $N = 4$ ports and that 80 CPU cycles are enough for processing packets (i.e., looking up entries in the flow table) while 10 additional CPU cycles (resp. 200) are needed to access the information if the vSwitch

experiences a cache hit (resp. cache miss). Overall, given the speed of the CPU cores (3GHz), we have: $T_H = (80 + 10)/3 = 30$ns and $T_M = (80 + 200)/3 = 93.3$ns. We also assume that the batch size M is set to 4 packets and that PCI buses sustain 16 GBps so that $T_R = 1000/16 = 62.5$ns. Finally, we assume that ports are unevenly loaded as follows: Port 1 receives 15% of the whole traffic, Port 2 receives 20%, Port 3 receives 25% and Port 4 receives 40%.

Scenario 2 - Complex routing

Our second scenario pertains to a vSwitch whose flow table is large featuring numerous rules to handle different types of traffic with various destinations. Such a situation can occur for routers located in the core (backbone) network of a network operator. Here, we assume that, because of the size of the flow table, 800 CPU cycles are needed for the lookup operation while 10 additional CPU cycles (resp. 200) are needed to access the information if the vSwitch experiences a cache hit (resp. cache miss). Therefore, we obtain: $T_H = (800 + 10)/3 = 270$ns and $T_M = (800 + 200)/3 \simeq 333$ns. We choose a larger size of packet batch with $M = 8$ and we assume that PCI buses work at 32 GBps so that $T_R = 31.25$ns. Finally, we assume a total of $N = 5$ ports that are irregularly loaded as follows: Port 1 receives 10% of the whole traffic, Port 2 receives 15%, Port 3 receives 20%, Port 4 receives 25% and Port 5 receives 30%.

Scenario 3 - IPsec

In our last scenario, we consider a vSwitch applying IPsec encryption operations on incoming packets. Network architects typically deploy IPsec tunnels to provide security for data communication between pairs of distant nodes. The packets are encrypted at the ingress of the tunnel and decrypted at its egress using computationally intensive encryption algorithms implemented in IPsec. We assume that 8,000 CPU cycles are spent to perform the encryption process and that 10 additional CPU cycles (resp. 200) are needed to access the information if the vSwitch experiences a cache hit (resp. cache miss). Given the speed of CPU cores (3GHz), this leads to $T_H = (8000 + 10)/3 = 2670$ns and $T_M = (8000 + 200)/3 = 2733.3$ns. The size of batches is set to $M = 16$ packets. The speed of PCI bus is fixed to 8 GBps so that $T_R = 125$ns. The total number of ports is set to $N = 8$ ports that are unevenly loaded as follows: Port 1 receives 5% of the whole traffic, Port 2 receives 10%, Port 3 receives 15%, Port 4 receives 18%, Port 5 receives 22% and Port 6 receives 30%.

2.5 Modeling a vSwitch as decoupled queues with server vacation

2.5.1 System notation

We start this section by reminding the notation introduced so far. As stated in Section 2.4, C denotes the total number of CPU cores devoted to the packet processing and N represents the number of ports attached to the vSwitch. As a results, the total number of RX queues of the vSwitch is equal to $N \times C$. Each RX queue has a finite capacity expressed as a maximum of K packets. Each CPU core cyclically polls its associated RX queues and processes at most M packets from each RX queue before switching to the next one. M is referred to as the batch size. Let us also recall that the average time needed

Table 2.1: Principal notation.

Symbol	Description
C	Number of CPU cores devoted to the packet processing
N	Number of ports
K	Capacity of the RX queues
M	Size of packet batches
T_H	Average time needed by a CPU core to process a packet in case of a hit in the cache
T_M	Average time needed by a CPU core to process a packet in case of a miss in the cache
T_R	Average time needed by a CPU core to forward a packet to a TX queue
T_S	Average time needed by a CPU core to switch to the next RX queue
β	Switch-over rate
Λ_i	Packet arrival rate on port i, $i = 1, \ldots, N$
λ_i^j	Packet arrival rate dispatched to the j-th RX queue of port i, $j = 1, \ldots, C$ and $i = 1, \ldots, N$
Λ^j	Packet rate bound to the j-th CPU core, regardless of their incoming port ($j = 1, \ldots, C$)
μ_i^j	Service rate of the j-th CPU core when it is serving the i-th RX queue, $j = 1, \ldots, C$ and $i = 1, \ldots, N$
U^j	Utilization rate of the j-th CPU core, $j = 1, \ldots, C$
B_i	Blocking probability at port i, $i = 1, \ldots, N$
b_i	Loss rate at the entrance of queue i, $i = 1, \ldots, N$
$\bar{q}_i$	Average number of packets in queue i, $i = 1, \ldots, N$
$\bar{r}_i$	Average sojourn time in queue i, $i = 1, \ldots, N$

by a CPU core to process a packet is denoted by T_H in case of a hit in the cache, and by T_M in case of a miss. We use T_R to refer to the average time needed by a CPU core to forward a packet to a TX queue while we use T_S to denote the average time taken by a CPU core to switch from its current RX queue to the next one.

We use Λ_i to denote the packet arrival rate on port i ($i = 1, \ldots, N$) while λ_i^j refers to the rate of packets dispatched to the j-th RX queue of port i, and hence handled by the j-th core ($j = 1, \ldots, C$). This is illustrated by Figure 2.2. It follows that $\Lambda_i = \sum_{j=1}^{C} \lambda_i^j$ and, assuming the hash function dispatches equally across the RX queues, we have: $\lambda_i^j = \frac{\Lambda_i}{C}$. Finally, we denote by Λ^j the total rate of packets bound to the j-th core CPU core, regardless of their incoming port ($j = 1, \ldots, C$) so that $\Lambda^j = \sum_{i=1}^{N} \lambda_i^j$.

For the sake of our modeling framework we let μ_i^j denote the service rate of the j-th core when it is serving the i-th RX queue, while β denotes the switch-over rate. Note that by definition, we have, $\beta = 1/T_S$. We detail later how μ_i^j can be derived from T_H, T_M and T_R.

Table 2.1 summarizes the principal notation used in this chapter.

2.5.2 Performance parameters of interest

The objective of this chapter is to develop an accurate and scalable modeling framework to derive performance parameters of the vSwitch. These metrics may pertain to the RX queues or to the whole system itself. As for the i-th RX queue ($i = 1, \ldots, N$) attached to the j-th CPU core ($j = 1, \ldots, C$), performance parameters of interest include the blocking

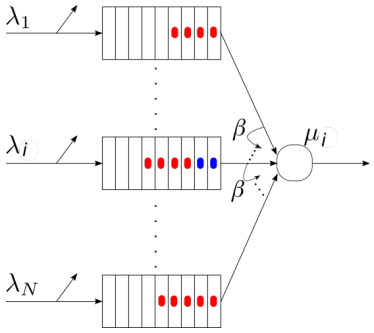


Figure 2.3: Illustration of a subsystem involving a single CPU core that polls N RX queues with a size of batch of $M = 2$. Blue packets are being processed while red packets are waiting for their turn.

probability (i.e., the loss rate) denoted by b_i^j, the mean sojourn time of a packet denoted by $\bar{r}_i^j$, as well as the buffer occupancy denoted by $\bar{q}_i^j$. Besides, for each CPU core j, we use U^j to indicate its utilization rate. Finally, the global performance of a vSwitch often derive from the former inner parameters. Thus, the global CPU core utilization rate, denoted by U, can simply be expressed as:

$$U = \frac{\sum_{j=1}^{C} U^j}{C} \tag{2.1}$$

Similarly, the packet blocking probability at port i, referred to as B_i, can be computed as:

$$B_i = \frac{\sum_{j=1}^{C} b_i^j \lambda_i^j}{\sum_{j=1}^{C} \lambda_i^j} = \frac{\sum_{j=1}^{C} b_i^j \lambda_i^j}{\Lambda_i} \tag{2.2}$$

Note that the global CPU utilization and blocking probability are performance parameters that capture and summarize the overall level of congestion in a vSwitch, and therefore represent metrics of direct interest for network operators.

2.5.3 Decomposition principle

The first step of the modeling framework is to break down the general switch architecture into C independent subsystems, each of them consisting of one CPU core that polls N independent RX queues. Recall that, as stated at the end of Section 2.4.2, we focus on the interactions between the CPU cores, the RX queues and the ports, and as a result we exclude from the model the transmission part of packets in TX queues. Every subsystem is identified with a distinct color in Figure 2.2 and is simply referred to as a *polling system*. In the rest of the chapter we only consider the model associated with a given CPU core j and its N related RX queues. Therefore, for the sake of clarity, we drop superscript j in all subsequent notations and equations. Figure 2.3 represents the polling system associated with the considered CPU core having a service rate μ_i when serving its i-th RX queue, and a switch-over rate β.

The second step of the general modeling framework consists in replacing each polling system with a set of N decoupled queues with server vacations. This decomposition step is illustrated in Figure 2.4. The buffer of queue i in the decomposed model represents the i-th RX queue associated with the considered CPU core. The server of the i-th queue in the decomposed model aims at reproducing the way packets of the i-th RX queue are

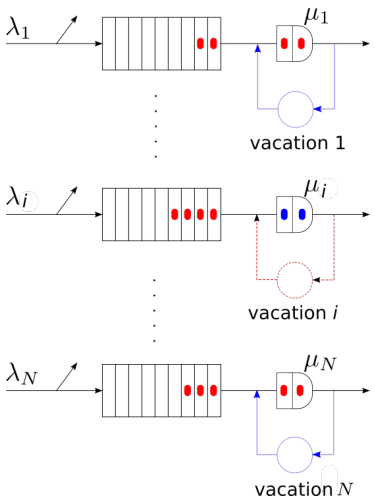


Figure 2.4: Decomposition of a subsystem into N separate queues with a size of batch of $M = 2$. Blue vacations are active while the red vacation is inactive.

processed by the CPU core. Because the core polls all its RX queues in-between the processing of two successive batches of (at most M) packets at a given queue i, there is an in-between time that corresponds to the processing of one batch of packets for all the other non-empty queues and N switch-over times. In the model, this total time will be referred to as a *vacation time*. As an illustration, in Figure 2.4, the server of queue i is in process, meaning that the CPU core is currently processing a packet of the current batch in RX queue i, and all other queues are in vacation. In this particular example, when queue i ends its processing, it goes in vacation, the remaining packets of queue i are put on a hold, and, at the same time, the switch-over time between RX queue i and RX queue $i+1$ starts. It is only after the completion of this switch-over time that queue $i+1$ ends its vacation and starts the processing of its first M in-line packets

2.6 Solution to the queues and their performance parameters

Having reformulated the initial problem of a polling system with N queues into a set of N separate queues with vacation time, we focus now on each of these queues.

2.6.1 Markovian assumptions

In order to derive tractable models, we make the following Markovian assumptions. We assume that the arrival of packets at the entrance of queue i follows a Poisson process of rate λ_i. We also assume that the processing time of one packet from queue i, the switch-over time, and the vacation times, are all exponentially distributed, respectively with rate μ_i, β and α_i.

2.6.2 Markov chain model associated with each RX queue

Under the markovian assumptions, we can associate with each queue i of the decomposed model, the continuous-time Markov chain depicted in Figure 2.5. The chosen state description has two dimensions and thus is made of two parts (k, CPU state). The left-hand side corresponds to the current number of packets in the queue, $k = 1, \ldots, K$, while the

right-hand side specifies if the CPU core is currently processing this queue (P), switching from this queue to the next one (S), or otherwise processing another RX queue or switching between the other RX queues (V).

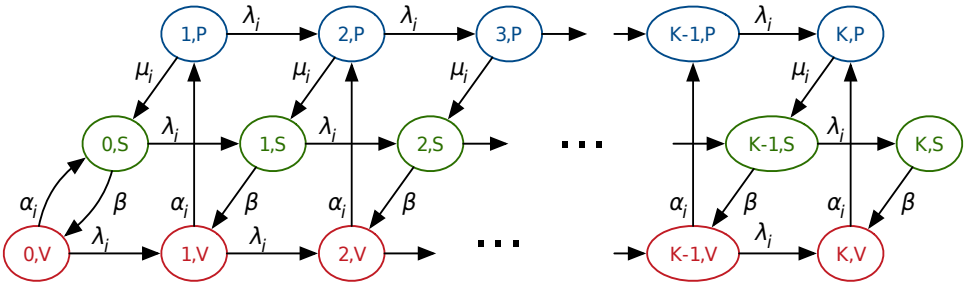


Figure 2.5: Continuous-Time Markov Chain associated with queue i.

In order to solve this chain corresponding to a particular queue i, we need to estimate three parameters, namely μ_i, β, and α_i (the other parameters λ_i and K are supposed to be known). Note that if all these parameters were to be known, then we can quickly and easily obtain the stationary probabilities of this Markov chain without resorting to any numerical technique (see [B2] for more details).

2.6.3 Estimating the service rate μ_i

In order to explain how we estimate the service rate at each RX queue μ_i (or equivalently, its mean processing time $1/\mu_i$), we take an example of a vSwitch having $N = 4$ ports and processing packets by batch of size $M = 16$. Let us consider that the CPU core at a particular round has to process a batch of 16 packets from RX queue 1 that is made of 4 packets destined to port 2, 7 packets destined to port 3, and 5 packets destined to port 4. We can reasonably assume that for the first packet destined to a given port, the information necessary for its processing (typically contained in the forwarding table) has to be fetched from the RAM, whereas for the subsequent packets destined to the same port, the information is present in the cache. If we make this assumption, the total time necessary to process all packets of the considered batch is thus $3T_M + 13T_H + 16T_R$, and the average processing time by packet is $\frac{3T_M+13T_H+16T_R}{16}$.

By generalizing this result, we first proposed in [B25] to estimate the average processing time of the considered CPU core when serving a packet from its i-th RX queue by the following simple equation:

$$\frac{1}{\mu_i} = \begin{cases} \frac{N-1}{M}T_M + \frac{M-(N-1)}{M}T_H + T_R & \text{if } M \geq N-1 \\ T_M + T_R & \text{if } M < N-1 \end{cases} \tag{2.3}$$

This estimation is realistic when $M >> N$ and when the system is heavily loaded. Indeed in this case there is a high chance that the CPU core processes full batches, i.e., batches made of M packets, and that in each batch there is at least one packet destined to each of the possible $N-1$ output ports. In this case, the processing time of the whole batch is $(N-1)T_M + (M-(N-1))T_H + MT_R$ and Eq. (2.3) becomes exact.

Note that in the special case where batches are of size $M = 1$, CPU cores are very unlikely to process consecutive packets belonging to the same flow, thereby essentially precluding any benefit from the cache. As a result Eq. (2.3) becomes simply:

$$\frac{1}{\mu_i} = T_M + T_R. \tag{2.4}$$

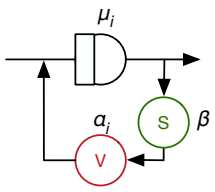


Figure 2.6: Representing the vacation times in our model.

In our work [B2], we refine the calculation of μ_i given by Eq. (2.3) to obtain a better estimate, especially when M is small or when the load is low.

2.6.4 Estimating the switch-over rate β

As for the switch-over rate, given a vSwitch with with batch size of M, we simply calculate β by spreading the delay incurred by the switch-over time T_S over the M packets that may compose the current batch. Therefore, we have:

$$\beta = \frac{M}{T_S}. \tag{2.5}$$

Note that if batches are of size $M = 1$, Eq. (2.5) turns to:

$$\beta = \frac{1}{T_S}. \tag{2.6}$$

2.6.5 Estimating the vacation rate α_i

We represent the vacation time as the sequence of the first switch-over time and aggregate all the remaining phases (including switch-over and processing times at the other RX queues). As shown in [B24], by doing this we drastically reduce the complexity of the model without significantly deteriorating its accuracy. This idea is illustrated in Figure 2.6. Following the processing stage of the server, the vacation starts by a switch-over time between RX queue i and RX queue $i+1$. The remaining of the vacation time is then aggregated into a single exponential phase with a given rate α_i (i.e., with a given mean duration $1/\alpha_i$), that has to be accurately estimated.

To estimate the value of α_i, we consider its inverse $1/\alpha_i$. This latter corresponds to the mean time between the end of switching from i-th to $(i+1)$-th RX queue (marking the time when the core is leaving queue i) and the end of switching from $(i-1)$-th to i-th RX queue (marking the time when the core is returning to queue i). Therefore, this time includes $N-1$ switch-over times, but also includes the processing of one packet for all non-empty RX queue j different from i. It follows that:

$$\frac{1}{\alpha_i} = (N-1) \times \frac{1}{\beta} + \sum_{j \neq i} (1 - \mathcal{E}_j) \times \frac{1}{\mu_j}. \tag{2.7}$$

In this expression, $\mathcal{E}_j$ represents the probability that RX queue j is empty when the core is returning to it, i.e., at the particular instant when the switch-over time from $(j-1)$-th to j-th RX queue ends. This parameters can be extracted from the Markov chain associated with RX queue j (equivalent to the one represented in Figure 2.5 but where i is replaced

by j). Indeed, $\mathcal{E}_j$ can be expressed as the ratio between the number of transitions from state $(0, V)$ to state $(0, S)$ by unit of time, and the total number of transitions from red states by unit of time, each of them correspondonding to the end of a vacation for RX queue j. Therefore, we have:

$$\mathcal{E}_j = \frac{\pi_j(0,V)\alpha_j}{\sum_{k=0}^{K} \pi_j(k,V)\alpha_j} = \frac{\pi_j(0,V)}{\sum_{k=0}^{K} \pi_j(k,V)}, \tag{2.8}$$

where the π_j are the stationary probabilities of the j-th Markov chain.

2.6.6 Fixed-point solution

As expected, the parameters of a Markov chain associated with a given queue i depend on the stationary solution of the other Markov chains (through Eq. (2.7) and (2.8)). As a result, the resolution of the model relies on a fixed-point iterative technique that is summarized by Algorithm 1. The main loop of the algorithm is repeated until a given convergence criterion is reached, e.g., the maximum relative difference of varying parameters between two successive iterations is very small (say less that 10^{-7}).

Algorithm 1: Fixed-point iterative technique

Input : System parameters K, μ_i, λ_i, β for each queue i
Output : Stationary probabilities π_i and performance metrics for each queue i
Initialize π_i, $\mathcal{E}_i$ for each queue i;
while *convergence criterion not satisfied* **do**
  foreach *queue* $i \in [\![1, N]\!]$ **do**
    Compute α_i using Eq. (2.7);
    Solve the Markov chain associated with queue i and compute the stationary probabilities π_i;
    Compute $\mathcal{E}_i$ using Eq. (2.8);
  end
end
Compute all performance metrics of interest from Eq. (2.9) to (2.11);

2.6.7 Computing the performance parameters

After convergence of our algorithm, we can derive the system performance parameters from the stationary probabilities of the Markov chains as follows. The average number of packets in queue i is given by:

$$\bar{q}_i = \sum_{k=1}^{K} k \times (\pi_i(k,P) + \pi_i(k,S) + \pi_i(k,V)). \tag{2.9}$$

We can compute the loss rate at the entrance of queue i as:

$$b_i = \pi_i(K,P) + \pi_i(K,S) + \pi_i(K,V). \tag{2.10}$$

The average sojourn time of an accepted packet in queue i is then obtained using Little's law:

$$\bar{r}_i = \frac{\bar{q}_i}{\lambda_i(1-b_i)}. \tag{2.11}$$

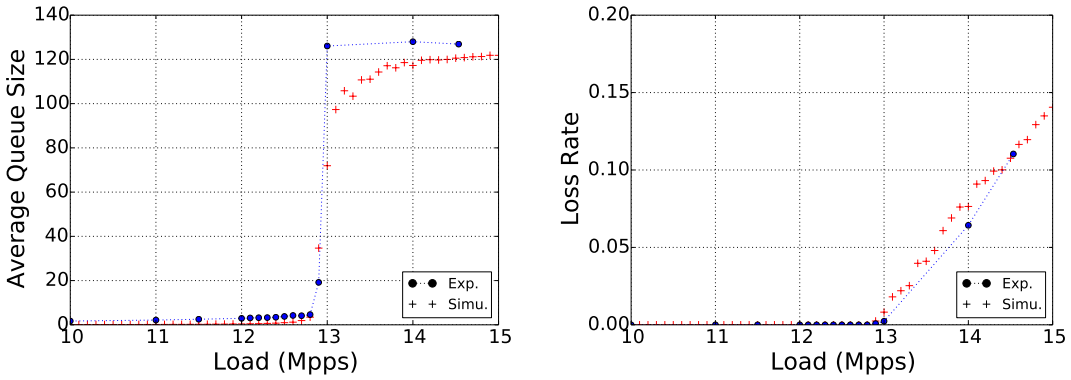
Note that once with these quantities are known, it is straightforward to obtain the performance for the vSwitch as a whole using Eq. (2.1) and (2.2).

2.7 Accuracy of the Proposed Approach

To study the accuracy of our modeling approach, we explore the three scenarios described in Section 2.4.3. We compare the results of our model to those of a home-made discrete-event simulator written in Java, whose code is made available [B4].

Unlike our model, the simulator precisely implements the behavior of a vSwitch as described in Section 2.4: i) Packets queued on an RX queue are processed by batches; ii) The time to process a packet is closely related to the presence or absence of the corresponding instructions in the cache, and not only averaged as this is the case in the model; iii) After processing a batch of packets on a given RX queue, the CPU core is assigned to the next RX queue. As such, the simulator does not proceed, as the model does, with a decomposition of the vSwitch architecture into decoupled queueing subsystems with vacation time.

Furthermore, we provide a brief validation of our simulator against real-life measurements to demonstrate its relevancy. For a given scenario, we compare the results provided by our discrete-event simulator [B4] that implements the behavior of a vSwitch as described in Section 2.4 with those collected on a real-life vSwitch implementing DPDK. The scenario is defined as follows. We consider a vSwitch with a total of $N = 32$ ports (equally loaded), RX queues of capacity $K = 128$ packets, CPU cores running at 2.3 GHz, a switch-over times T_S of 1ns, packet batch of size $M = 8$ and an average packet processing time of 178 CPU cycles corresponding to 77.43ns. The real-life vSwitch implements OVS with 6WINDGate and DPDK running on Ubuntu Linux with Intel Core i7 CPUs and Intel ixgbe NICs. Packets of 64 bytes were generated using IXIA. Figure 3.5 represents the corresponding results for the loss rate and the average queue size for a wide range of values of load. We observe that the results delivered by our simulator are in fair agreement with the experimental measurements.



(a) Average queue size.

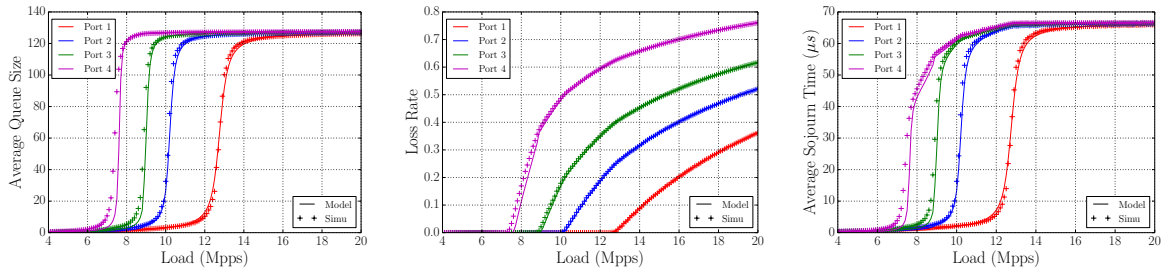
(b) Loss rate.

Figure 2.7: Simulator results against real-life measurements.

To validate the accuracy of our model, we run the simulator using seven independent replications of 50,000,000 packets completions each. The obtained estimated confidence intervals at 95 percent confidence level are so small that we use only the mid-point in our validation. In each scenario, we consider a wide range of values for the packet arrival rate, varying from a very low level of load up to a high level corresponding to a full saturation of the vSwitch.

Numerical results for Scenario 1, in which we consider the case of a simple forwarding

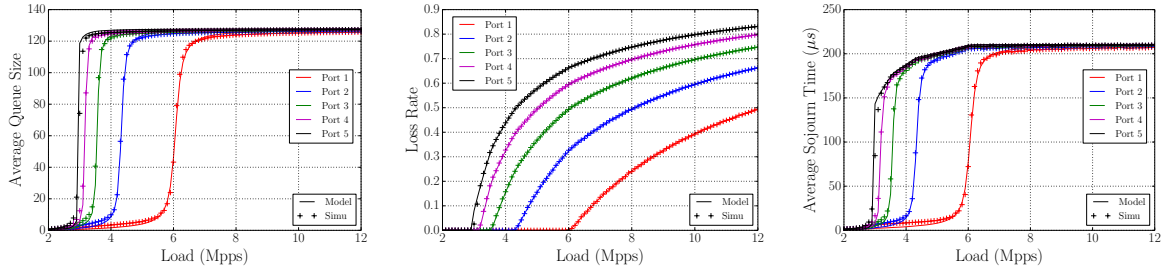
with $N = 4$ ports and batches of $M = 4$ packets, are presented in Figure 2.8. Refer to Section 2.4.3 for a complete list of the parameters. Figure 2.8a represents the average queue size (number of packets being buffered in RX queues) as a function of the load. We first notice that possible values of the average queue size vary from 0 for a low level of load up to 128 (corresponding to the capacity K of RX queues) when the load is high. As expected, we observe that the most loaded port, namely port 4 (receiving 40% of the total load), is the first to saturate when the load increases. For example, at a level of load of 8 Mpp, the queue associated with port 4 is almost full (i.e., close to 128 packets) while the queues associated with the three other ports are almost empty. Note also that the curves are significantly steep denoting a high sensibility to the actual level of load. In Figure 2.8b, we consider the loss rate (i.e., blocking probability) experienced by each port as a function of the load. We note that the most loaded port is again the first to undergo losses as soon as the load exceeds 7 Mpps. Finally, we study the evolution of the average sojourn time spent by a packet in an RX queue in Figure 2.8c. At heavy levels of load, the value found for each port converges to a common asymptotic value (corresponding to the average time necessary to process 128 packets of a given RX queue). Overall, Figure 2.8 shows the close agreement between the proposed model and simulation.



(a) Average queue size. (b) Loss rate. (c) Average sojourn time.

Figure 2.8: Accuracy of our approach on Scenario 1 - Simple forwarding.

Scenario 2 deals with the case of a vSwitch whose flow table is much larger and complex. Let us recall that, in this scenario, the number of ports is set to $N = 5$ while the size of batches equals $M = 8$ packets. Figure 2.9 presents the associated results. Figure 2.9c shows that, in the case of the average sojourn time, the asymptotic value for large levels of load differs from that found on Figure 2.8c. This gap results from the fact the mean time to process a packet is significantly larger in our second scenario. Here too, the performance parameters returned by our model closely match those delivered by the simulator.

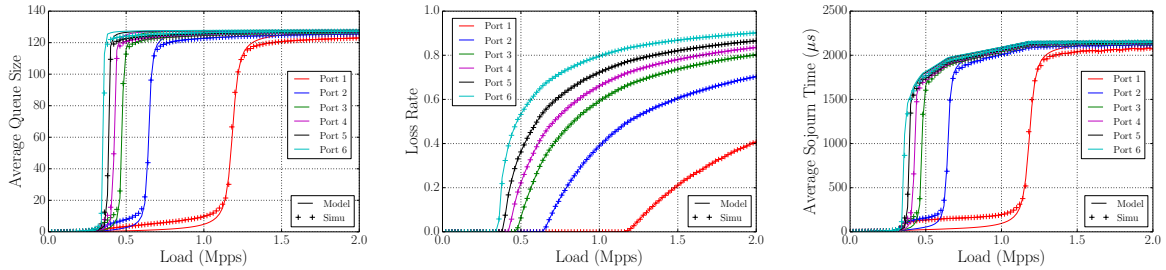


(a) Average queue size. (b) Loss rate. (c) Average sojourn time.

Figure 2.9: Accuracy of our approach on Scenario 2 - Complex routing.

Finally, Scenario 3, which addresses the case of a vSwtich performing IPsec functions and featuring $N = 6$ ports with a size of batches set to $M = 16$ packets, is handled in

Figure 2.10. We again observe that the performance obtained from our model are close to those delivered from the simulator at any level of load.



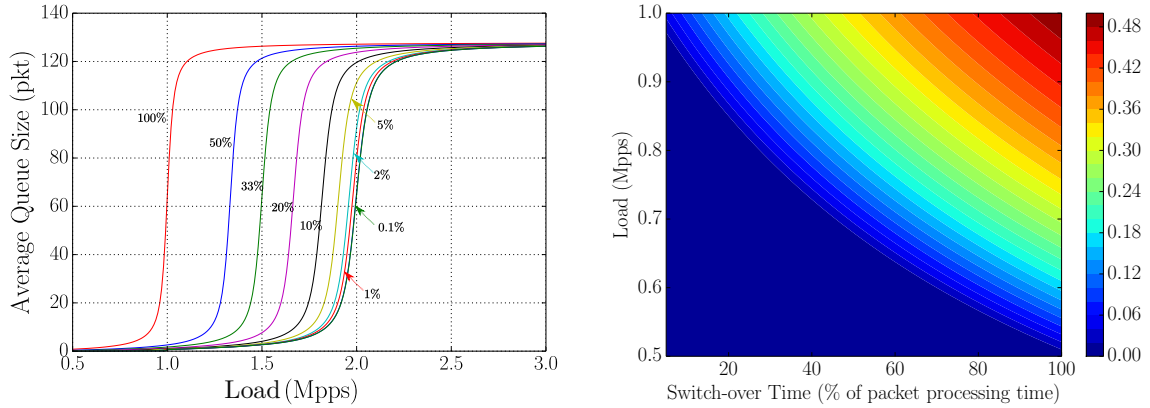
(a) Average queue size. (b) Loss rate. (c) Average sojourn time.

Figure 2.10: Accuracy of our approach on Scenario 3 - IPsec.

2.8 Examples of Application

Influence of switch-over time

We begin by studying the impact of the switch-over time on the average size of RX queues. We use parameters close to those of Scenario 2 and let T_S vary so that it ranges from a very low overhead (i.e., representing 0.1% of the average packet processing time), all the way to a massive overhead (i.e., 100%). Then, based on our model, we compute the average number of packets buffered in the RX queue of the most loaded port for different levels of load. The associated results are reported in Figure 2.11a. First, we notice that the relationship between T_S and the average queue size is far from being linear. Indeed, the deviation between an overhead of 100% and 50% is approximately twice smaller than that between 50% and 0.1%. Second, starting from a switch-over time representing approximately 2% of the packet processing time, all curves for subsequent smaller values tend to coincide. From a practical point of view, this suggests that, whenever the switch-over time represents an overhead less than, say 1% or 2%, they can be neglected in the performance analysis of a vSwitch.



(a) On the average size of an RX queue for Scenario 2 for various levels of load.

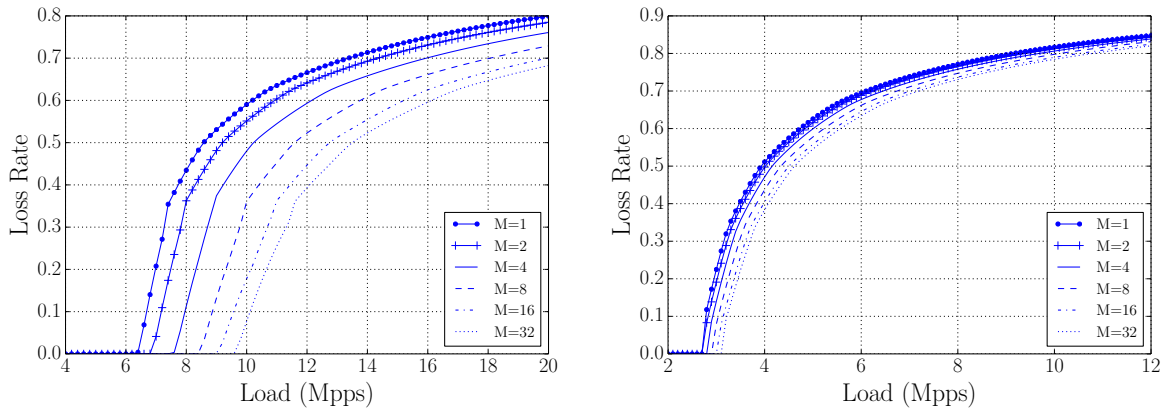
(b) On the loss rate experienced at an RX queue for Scenario 2 for various levels of load.

Figure 2.11: Influence of the switch-over time.

We continue this study by examining simultaneously the influence of the switch-over time and of the load, on the loss rate of an RX queue. Figure 2.11b shows the corresponding results with a varying switch-over time on the X-axis and a varying load on the Y-axis. As shown by the figure, when the load is low (under 0.5-0.6 Mpps), the value of the switch-over time overhead has virtually no influence on the loss rate that remains totally negligible, whereas when the load is high (close to 1 Mpps), the switch-over time overhead has a large impact on the loss rate.

Influence of batch size

In our second example, we investigate the influence of the size of packet batches M on the performance of a vSwitch. We begin our study by considering Scenario 1. We let M vary from a value of 1, in which at most one packet of each RX queue is processed before the CPU core moves to the next RX queue, up to a value of 32. We restrict our analysis on the loss rate experienced by the most loaded port, namely port 4. Figure 2.12a shows the corresponding results. We observe that there is a substantial gain in increasing the size of packet batches as the onset of packet losses is postponed from a load of 6.5 Mpps for $M = 1$ to a load of almost 10 Mpps for $M = 32$, which represents an improvement of more than 50%. In the same way, while a load of 9 Mpps results into a severe congestion for a vSwitch parameterized with $M = 1$ (with a loss probability approaching 0.55), setting M to 32 leads to virtually no packets being lost. It is worth noting that the marginal gain of incrementing the size of batches decreases quite rapidly with growing values of M.



(a) On the loss rate experienced at an RX queue for Scenario 1.

(b) On the loss rate experienced at an RX queue for Scenario 2.

Figure 2.12: Influence of the size of batches.

To develop a better understanding of the effect of M on the behavior of a vSwitch, we switch to Scenario 2 and we re-run our experiment. Let us recall that in this scenario the time for processing a packet varies much less between a cache hit and a cache miss. We represent the obtained results in Figure 2.12b. As is shown, the size of the packet batches does not affect much the values obtained for the loss rates. Indeed, the gain obtained by increasing M from 1 to 32 barely reaches 10%. This is in stark contrast with Scenario 1. This difference stems from the nature of the load. Indeed, unlike Scenario 1, here incoming packets belonging to the same batch are quite unlikely to access the same information in the CPU cache, and hence there is little benefit in batching their services.

Based on these two examples, it appears that increasing the size of packet batches may

significantly improve the performance of a vSwitch. However, the magnitude of the gain may vary widely depending on the characteristics of the packet processing times. Significant gains are expected when the average time needed to process a packet in case of a cache hit is significantly less than in the case of a cache miss.

Ensuring zero-loss policy

To illustrate the potential application of our model, we consider the problem of determining the proper number of CPU cores to meet a given QoS criterion. Operators often aim to size their networking devices so that packets exchanged over these devices are (almost) never dropped. We now describe how our model can help achieve this so-called "zero-loss" policy on a vSwitch.

We consider a vSwitch parameterized closely to that described in Scenario 3 but with $M = 1$. However, we let the number of allocated CPU cores, C, unspecified as it varies from 1 to 20. Then, for increasing values of the load submitted to the vSwitch, we compute the total loss rate experienced over all RX queues. Figure 2.13 shows the corresponding results. We observe that for a load of 8 Mpps, a minimum of $C = 10$ cores is required to ensure the zero-loss policy. This number grows to $C = 17$ cores if the load gets to 14 Mpps.

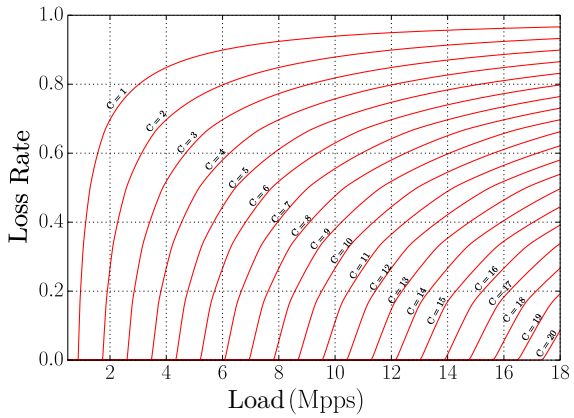


Figure 2.13: Ensuring zero-loss policy by adequately determining the number of allocated CPU cores.

Dimensioning curves, analogous to those depicted in Figure 2.13, are easily and quickly delivered by our model. We believe that the knowledge conveyed by these curves can provide useful information in order to avoid the under- or over-provisioning of networking devices.

2.9 Related Works

Many works have concentrated their efforts on evaluating the performance of DPDK-based vSwitch, either by conducting measurements or by using modeling techniques. In the following, we first present the approach adopted by DPDK's contributors to address this topic from an experimental point of view. We then review modeling attempts of vSwitch systems and, because polling is a key feature of DPDK systems, we study the related works of polling models.

Experimental approach

With the objective of achieving the best software switching performance, engineers contributing to the DPDK library developed internal performance measurement tools similar to monitoring probes. This approach has the advantage of giving a localized and precise analysis of low-level performance of a DPDK system, thus allowing developers to benchmark and validate any novel implementation or algorithmic optimization.

Nonetheless, the fact of reading and updating values in the system, typically when handling a counter, blocks the CPU cores for a while and so, reduces the overall system performance. About this, Vyatta/Brocade principal software architect stated during the 2015 Dublin DPDK Userspace forum that the act of "observing performance slows it down" [B8] To overcome this issue the performances of DPDK-based switching solutions are mainly assessed experimentally using dedicated test software such as the TestPMD application provided by Intel [B9]. Such applications are for instance used by Intel to publish bi-annual DPDK performance reports [B10]. Several works also attempted to experimentally characterize the open vSwitch performance. Most of them conducted measurements on an experimental testbed to demonstrate the enhancement provided by DPDK [B5]. They also measured the impact of the number of NIC, the offered load, and the packet size [B30]. Other works investigated the impact of active flow monitoring [B17], that might simply consist in sampling packets being forwarded across the vSwitch. Again, increasing the sampling rate to gain accuracy keeps consuming CPU resources, and in turn degrades the overall performance.

Modeling approach

By being non-intrusive in nature, modeling approaches can be used to overcome these measurement constraints and help in the performance evaluation of a DPDK system.

Non-DPDK-specific vSwitch modeling

Modeling the performance of vSwitch systems has been addressed for non-DPDK-specific systems. For instance, a model for estimating the packet loss probability and the average sojourn time of OpenFlow architectures is provided in [B11]. This model assumes that all the packets arrive at the same queue before being forwarded to the switch. Suksomboon et al. presented an optimal configuration selection algorithm for Linux-based software routers relying on an Erlang-k-based packet latency prediction model [B26]. Such prediction model uses measurements performed on two different router configurations to accurately predict the performance of all the configurations. It considers the case of systems with only a single CPU core.

These works cannot capture the effect of more sophisticated processing strategies such as polling used in the DPDK Poll Mode Driver or packet batch processing, that both highly contributes to a vSwitch performance enhancement [B7].

Polling system modeling

As polling is a key feature of DPDK systems, we review the works done in this regard. Reference surveys on polling models were published in the early 1990's by Takagi to provide a classification of polling systems and related research advances [B27], [B28].

Polling systems can be classified according to service policies, that might be exhaustive or gated, and unlimited or M-limited. Exhaustive: Once the server polls a given queue, it serves the queue until its complete exhaustion, and then it switches to the next queue. This implies that any request arriving in a queue while the server is currently processing another request of the same queue will be served before the server moves to the next queue. Gated: Unlike the exhaustive policy, the server does not process (in the current round) requests that may enter the queue while the server is already serving this same queue. Additionally, for both aforementioned policies, one can set an upper limit on the number of requests that the server can process for the same queue before switching to the next one. M-Limited: On each turn, the server can serve at most M requests for each queue. This corresponds to the case studied in this chapter. Unfortunately, the general solution to polling systems is not known. However, their analysis is no less important, and therefore, several approximations have been developed. Tran-Gia proposed an analytical framework for computing the performance of a gated 1-limited polling system with non-zero switch-over time [B29]. The modeling approach consists of solving a fixed-point problem to evaluate the state probabilities of an embedded Markov chain. It requires the computation of Laplace-Stieltjes transforms as well as the use of Laplace inversion procedures or two-moment approximation techniques. As stated by the authors, such model is accurate only for large switch-over times and small values of the queue capacities (less than 10 requests). The fixed-point approach developed by Tran-Gia has then been extended to the case of exhaustive M-limited systems in [B13]. In this work, the authors leverage the techniques provided by Lee to study $M/G/1/K$ queues with server vacation [B14], [B15]. It consists in decomposing the polling system in individual $M/G/1/K$ queues with server vacation. Each queue is then studied at polling instants. To reduce the number of modeling assumptions introduced in the previous works, a more general framework is presented in [B12]. When conducting the analysis of each queue, it eliminates the hypothesis that the busy period, i.e., the time the server is not on vacation and that the vacation times are independent. This approach relies on solving a system of several numerical equations. However, as stated by the author, the complexity of the involved expressions may require the use of a symbolic computation software.

In conclusion, most of these approaches address a different policy than that implemented in vSwitches, and/or they involve complex arithmetic operations that may not scale with the number of queues or with their capacity. Hence, they seem to be of little help when evaluating the performance of a DPDK-based vSwitch system.

2.10 Conclusions

Summary of Contributions

In this chapter, we present an analytical queueing model to evaluate the performance of a DPDK-based virtual switch with several CPU cores and network interface cards. Polling systems, in which a set of servers sequentially poll packets from a set of queues, with batch services, in which several packets are processed simultaneously, arises as an appropriate representation for modeling the behavior of a vSwitch. To circumvent the combinatorial growth of the state space associated with these models and their inherent complexity, we decouple the polling system associated with each CPU into several queues and we resort to servers with vacation to capture the interactions between queues. Our proposed solution is conceptually simple, easy to implement and computationally efficient.

We conduct tens of examples to assess the accuracy of our proposed model for various performance parameters such as the attained throughput, the packet latency, the buffer occupancy and the packet loss rate under various levels of loads. Comparisons against a discrete-event simulator show that our models typically deliver accurate estimates of the performance parameters. We illustrate how our models can help in determining an adequate setting of the vSwitch parameters using three real-life case studies, and derive some qualitative conclusions. For example, we find that increasing the size of packet batches may significantly improve the performance of a vSwitch, but only if a cache miss implies a much larger access time than a cache hit.

Encountered Difficulties

The main difficulty faced during this work was probably the lack of a technical documentation describing the inner workings of a DPDK-based vSwitch. Fortunately, we managed to collect all the required information thanks to 6WIND which brought us a sufficient understanding of the system to carry out our performance modeling. However, this learning process took time (almost two years) and implied many meetings, discussions, and exchanges of emails with 6WIND, as well as our own search for information on the web.

Strengths and Limitations

In my opinion, the main strength of our model is its simplicity. It is conceptually simple, scalable with the number of CPUs and NICs, easily implementable and its execution is fast.

On the other hand, the main limitation of our work probably lies in the lack of experimental data. Although we managed to describe realistic scenarios and to validate our simulator against real-life measurements, additional empirical study would probably have reinforced this work. Unfortunately, we relied on 6WIND to obtain real-life measurements and this proved to be more complex than they initially thought.

Possible Extensions

So far, we only investigated the performance of a DPDK-based vSwitch when setup in its standard configuration (known as the Poll Mode Driver). In this mode, each CPU core (with the exception of the master) contributes to processing packets from all the existing NICs. Other configurations, where a subset of CPU cores are devoted to a subset of NICs, could be worth exploring.

Potential Impact

As far as we know, we were the first to address the issue of performance modeling for DPDK-based vSwitches. Thus, our work may stimulate further research in this direction. As for potential impacts in the industry, our model may help engineers to better set up the resources and the parameters of vSwitches (e.g., number of allocated CPU cores and size of batches) before deploying them. Finally, the model can help in identifying the bottleneck limiting the performance of DPDK, and hence indicate where DPDK developers should focus their effort to overcome it.

Learned Experiences

Because of the inherent complexity of DPDK-based vSwitches and the many factors ruling their behavior, we adopted an incremental approach. Indeed, in our first attempt at modeling [B1], we neglected both the effects of packet batching and switch-over times. Then, we introduced the possibility of having switch-over times [B24], and the possibility of having packet batches [B25]. Finally, our last contribution [B2] includes realistic scenarios derived from real-life situations and experimental validations against real-life experiments. More generally, I believe that, in many cases, an incremental approach is an effective way of conducting research.

Additionally, this work also emphasized the importance of regularly writing progress reports and archiving all materials. Indeed, to address a reviewer request in our latest work [B2], we relied to measurements that we received from 6WIND two years earlier.

References for Chapter 2

[B1] G. Artero Gallardo, B. Baynat, and T. Begin. Performance modeling of virtual switching systems. In *IEEE MASCOTS*, 2016.

[B2] T. Begin, B. Baynat, G. Artero Gallardo, and V. Jardin. An accurate and efficient modeling framework for the performance evaluation of DPDK-based virtual switches. *IEEE Transactions on Network and Service Management*, 2018.

[B3] Data Plane Development Kit (DPDK). http://dpdk.org, 2017. Intel, 6WIND, etc.

[B4] DPDKSim - A simulator for DPDK-based Virtual Switches. https://github.com/garterog/DPDKSim, 2018.

[B5] P. Emmerich, D. Raumer, F. Wohlfart, and G. Carle. Performance characteristics of virtual switching. In *IEEE CloudNet*, 2014.

[B6] Fast data – Input/Output - Vector Packet Processing. https://wiki.fd.io/, 2017.

[B7] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle. Comparison of frameworks for high-performance packet io. In *ACM/IEEE ANCS*, 2015.

[B8] S. Hemminger. DPDK performance, Lessons learned in vRouter. In *DPDK Summit, Userspace 2015, https://dpdksummit.com/Archive/pdf/2015Userspace/DPDK-Userspace2015-StephenHemminger-Performance.pdf*, 2015.

[B9] Intel DPDK. The TestPMD Application. In *http://dpdk.org/doc/guides/testpmd_app_ug/index.html*.

[B10] Intel DPDK Validation team. DPDK Intel NIC Performance Report Release 18.02, March 2018. In *http://fast.dpdk.org/doc/perf/DPDK_18_02_Intel_NIC_performance_report.pdf*.

[B11] M. Jarschel, S. Oechsner, D. Schlosser, R. Pries, S. Goll, and P. Tran-Gia. Modeling and performance evaluation of an openflow architecture. In *ACM/IEEE ITC*, 2011.

[B12] D. Kofman. Block probabilities, throughput and waiting time in finite capacity polling systems. *Queueing Systems*, 1993.

[B13] M. Lang and M. Bosch. Performance analysis of finite capacity polling systems with limited-m service. In *ACM/IEEE ITC*, 1991.

[B14] T. T. Lee. *M/G/1/N* queue with vacation time and exhaustive service discipline. *Operations Research*, 32(4):774–784, 1984.

[B15] T. T. Lee. *M/G/1/N* queue with vacation time and limited service discipline. *Performance Evaluation*, 9(3):181–190, 1989.

[B16] H. Levy and M. Sidi. Polling systems: applications, modeling, and optimization. *IEEE Transactions on Communications*, 38(10), 1990.

[B17] V. Mann, A. Vishnoi, and S. Bidkar. Living on the edge: Monitoring network flows at the edge in cloud data centers. In *IEEE COMSNETS*, 2013.

[B18] N. McKeown, T. Anderson, H. Balakrishnan, G. M. Parulkar, L. L. Peterson, J. Rexford, S. Shenker, and J. S. Turner. OpenFlow: enabling innovation in campus networks. *Computer Communication Review*, 38(2), 2008.

[B19] Open vSwitch - An Open Virtual Switch. http://www.openvswitch.org, 2017.

[B20] OpenOnload. http://www.openload.org, 2017. Solarflare.

[B21] PacketShader. http://shader.kaist.edu/packetshader/, February 2011.

[B22] L. Rizzo. netmap: A novel framework for fast packet I/O. In *USENIX ATC*, 2012.

[B23] D. Scholz. A Look at Intel's Dataplane Development Kit. *Network*, 115, 2014.

[B24] Z. Su, B. Baynat, and T. Begin. A new model for DPDK-based virtual switches. In *IEEE NetSoft*, 2017.

[B25] Z. Su, T. Begin, and B. Baynat. Towards including batch services in models for DPDK-based virtual switches. In *IEEE GIIS*, 2017.

[B26] K. Suksomboon, N. Matsumoto, S. Okamoto, M. Hayashi, and Y. Ji. Erlang-k-based packet latency prediction model for optimal configuration of software routers. In *IEEE NetSoft*, 2017.

[B27] H. Takagi. Queuing analysis of polling models. *ACM Comput. Surv.*, 20(1):5–28, 1988.

[B28] H. Takagi. Analysis of finite-capacity polling systems. *Advances in Applied Probability*, 1991.

[B29] P. Tran-Gia and T. Raith. Performance analysis of finite capacity polling systems with nonexhaustive service. *Performance Evaluation*, 9(1):1 – 16, 1988.

[B30] D. Zhou, B. Fan, H. Lim, M. Kaminsky, and D. G. Andersen. Scalable, high performance ethernet forwarding with cuckooswitch. In *ACM CoNEXT*, 2013.

Chapter 3

Conflict graph-based modeling of IEEE 802.11 networks

Contents

3.1 Research Context **38**
3.2 Outline **39**
3.3 Motivations **39**
3.4 State of the Art **40**
3.5 System Description **41**
3.6 Model and its Solution **44**
3.6.1 Decomposing into subnetworks 44
3.6.2 Solving each subnetwork as one or more Markov chain(s) 45
3.6.3 Combining subnetwork solutions 49
3.7 Numerical Results **50**
3.7.1 Model validation 50
3.7.2 Modeling complexity 54
3.7.3 Possible application: Channel assignment 54
3.8 Conclusions **57**

Abstract

WLANs (Wireless Local Area Networks) based on the IEEE 802.11 standard have become ubiquitous in our daily lives. To extend their coverage and transmission capacity, network operators typically augment the number of APs (Access Points) composing the WLAN. This leads to network densification, which in turn demands some form of coordination between APs so as to avoid potential misconfigurations.

In this chapter, we describe a performance modeling method that can provide guidance for configuring WLANs and be used as a decision-support tool by a network architect, or as an algorithm embedded within a WLAN controller. The proposed approach estimates the attained throughput of each AP, as a function of the WLAN's conflict graph, the AP loads, the frame sizes, and the link transmission rates. Our modeling approach employs a Divide-and-Conquer strategy that breaks down the original problem into multiple sub-problems, whose solutions are then combined to provide the solution to the original problem. We conducted extensive simulation experiments using the *ns*-3 simulator that show the model's accuracy is generally good with relative errors typically less than 10%. We then explore the issue of assigning channels to APs when configuring a WLAN.

3.1 Research Context

During the 2010-2013 period, I was involved in an ANR (French National Research Agency) funded project called *RESCUE*. This project focused on wireless-based substitution networks [C29, C3]. As part of this project, we developed a modeling framework to theoretically evaluate the performance of multi-hop IEEE 802.11-based networks [C1, C2, C4]. In practice, we model each node belonging to a chain (or routing path) by a single server queue so that the full chain was modeled as a set of queues in series. The difficulty lied in adequately setting the values of the service times on each server. Indeed, the service time at each queue is a function of many factors, including the frame sizes, the link transmission rates, the collision probabilities, the throughputs of neighbor nodes, etc. We managed to come up with a closed-form expression for the service times in the case of small-scale multi-hop chains (up to 4 nodes). However, although our ultimate goal was to enable our approach to deal with large chains of networks (say 8 or 10), we realized that the tuning of some of the model parameters (e.g., the collision probability) will hinder the scalability of our approach.

A couple of years later, as part of a Master internship (co-supervized with Anthony Busson) that evolved into a Ph.D. that I am currently supervizing, I tackled again the issue of modeling IEEE 802.11 networks. Although the studied topic was not exactly the same (we were no more considering substitution networks), it had much in common. Therefore, based on my prior experience on IEEE 802.11, I really made sure that, since the very beginning and during the whole modeling process, we will design models that can scale up easily with the number of nodes.

This chapter details this contribution, which pertains to the modeling of IEEE 802.11-based wireless networks. It results from a joint-effort together with Anthony Busson (UCBL) and Marija Stojanova (UCBL). So far, it led to the publication of one conference paper [C31] as well as one journal paper (currently in revision) [C32].

3.2 Outline

The remainder of this chapter is organized as follows. In Section 3.3 we discuss the scientific context around this work as well as its motivations. Section 3.4 provides a review of the related works. In Section 3.5, we describe the considered system and the performance metrics of interest. Our modeling framework for IEEE 802.11-based WLANs is detailed in Section 3.6. In Section 3.7, we present the numerical evaluation of our model as well as a possible application to the configuration of a WLAN. Section 3.8 concludes this chapter.

3.3 Motivations

WLANs (Wireless Local Area Networks) have become ubiquitous and part of our daily lives. They are frequently offered in public places such as cafes, restaurants, hotels, shopping malls, museums, metro and train stations, airports, and often available in places like trains, planes, workplaces, domestic houses, educational institutions, etc. As for the devices connected to an Access Point (AP) of a WLAN, their variety has greatly expanded, comprising desktop and laptop computers, IP phones, smartphones, digital media players, etc.

WLANs are typically based on the IEEE 802.11 standard [C16] (commercially known as WiFi). In order to meet the increasing needs of WLAN users, IEEE 802.11 has undergone several amendments, mostly aimed at strengthening its performance and security. In particular, MAC (Medium Access Control) and PHY (Physical) functions have been enhanced. Indeed, transmission technologies, defining the PHY layer of IEEE 802.11, have significantly evolved over the years using e.g., wider channels, higher-order modulations, multiple-input multiple-output antennas (MIMO). Maybe to a lesser extent, the MAC layer has also undergone some transformations with the possibility of using the Request to Send / Clear to Send mechanism (RTS/CTS), smaller mandatory waiting periods before transmissions, as well as frame aggregation and block acknowledgment in the latest amendments of IEEE 802.11.

In order to extend the coverage and the available transmission capacity of WLANs, network architects may augment the number of APs within a WLAN. This network densification comes with a growing complexity in the WLAN management. Indeed, a WLAN with several APs requires some form of coordination between its APs so as to avoid potential misconfigurations that could lead to an inefficient use of radio resources, poor performance and/or unfairness between users. For instance, coordination efforts can pertain to the selection of a radio channel for each AP (for mitigating interferences from neighboring APs) as well as to the association of user devices with the APs (for balancing the load among APs). Some proprietary and commercial solutions implement such mechanisms. Among others, CAPWAP and 802.11v protocols, issued by IETF and IEEE respectively, enable APs (within the same WLAN) to exchange information about the network topology and radio environment to a central controller. However, the algorithms run by the controller and exploiting this knowledge are yet to be designed. Indeed, unlike PHY and MAC layers, coordinating the APs of a WLAN has attracted little attention so far.

In this chapter, we describe a performance modeling method that can provide guidance for configuring an IEEE 802.11-based WLAN composed of multiple APs. The method can be used as a decision-support tool by a network architect or as an algorithm embedded within

a WLAN controller. The proposed approach offers estimates of the attained throughput of each AP. These estimates are obtained in return for a WLAN description including its conflict graph, the AP loads, the frame sizes, and the link transmission rates. Our modeling approach employs a Divide-and-Conquer strategy in which we break down the complexity of the original problem by considering multiple sub-problems, whose solutions are then combined to provide the solution to the original problem. The proposed solution is conceptually simple, easily implementable, and can be fully automated. We conducted extensive simulation experiments using the *ns*-3 simulator to evaluate the accuracy of our solution. Numerical results show that its accuracy is generally good with relative errors typically less than 10%.

3.4 State of the Art

The different models that evaluate the performance of IEEE 802.11-based WLANs range over a wide spectrum of levels of abstraction. Bianchi [C5] as well as Cali, Conti, and Gregori [C8] model the network at a very fine level of abstraction. Both models take into account the behavior of every single frame transmission. In [C8] the authors analyze the ratio of the average frame size and its average transmission time in order to study the utilization of the network's capacity. Bianchi's seminal work [C5] introduced a model based on a two dimensional Markov chain. The Markov chain models the backoff process that takes place before every Distributed Coordination Function (DCF) frame transmission while the network is assumed to be fully-connected, i.e., all nodes are neighbors. A property shared by both models is that the networks they consider are saturated, meaning all nodes constantly have frames waiting to be sent.

Because the saturation assumption can be deemed too restrictive in some cases, many subsequent works are centered on relaxing it. Kosek-Szott [C20] as well as Gupta and Rai [C14] circumvent this barrier by adding one more state to the Markov chain proposed by Bianchi [C5]. This new state represents a node that has no frames to be sent. Note that both works deal only with fully-connected networks.

Another solution is proposed by Felemban and Ekici [C12], who have removed the condition of saturation by introducing the probability that a node has a frame waiting to be sent. They do so by creating a second Markov chain, embedded into Bianchi's original Markov chain. The embedded chain describes the current state of the channel, which can be either idle, in collision, or in successful transmission. The solution to their model is found by successively iterating between the two chains. Upon convergence, the found solution delivers the steady state transmission probability for each node, which can then be used to evaluate the network's performance. However, like Bianchi's original model [C5], the focus of this work is restricted to fully-connected networks.

To overcome the inherent complexity tied to a fine level of abstraction when the network grows in size, other works have developed modeling approaches that incorporate both a fine-level and a high-level of abstraction. Two such models are given in [C30] and [C4]. Both models analyze non-saturated multi-hop networks. In a multi-hop network, a packet from node A travels across relay nodes before arriving at its destination node B (as opposed to single-hop networks, where A and B directly exchange packets). Both papers present two-level modeling approaches of unsaturated multi-hop wireless networks, in which the low-level model is a version of Bianchi's original Markov chain, while the high-level model aims at capturing the inter-node dependencies in the network. In [C4], the high-level model consists of a set of $M/M/1/K$ queues, where each queue represents a given node

of the network. Although their modeling framework is designed to handle any number of nodes, setting the parameters of the models become a complex matter when the network comprises more than 4 nodes. In [C30], the high-level model is a separate Markov chain describing the channel's behavior. In practice, their analytical model leads to a large state space as the number of nodes increases, making it intractable for networks with more than 7 or 8 nodes.

Finally, at the other extreme of the spectrum, there are the modeling approaches that analyze the network from a high level of abstraction. These models do not take into account the behavior of every frame transmission, and instead, deal with the behavior of the entire network as a whole. In [C22, C11, C18], Markov chains are used to model a network based on its topology. The states of the chain describe the set of nodes that are transmitting in the current network state. Nardeli and Knightly [E17] rely on a Markov chain to derive a model that takes into account the errors due to collisions and hidden terminals for a single-hop network. The authors derive a closed-form expression for the throughput. Although the model accurately captures the behavior of CSMA/CA networks, it only deals with saturated networks. In [C11], a similar Markov chain is used to evaluate the fairness and spatial reuse in multi-hop, saturated networks with different carrier sensing and reception zones. More particularly, the authors study the spatial reuse in line-networks to show that CSMA/CA achieves maximal spatial reutilization as traffic intensity increases, at the cost of creating starvation in certain links. In [C7, C34] CSMA/CA networks are modeled as Markov chains and the model is then used to study the fairness of the network.

A novel approach in the modeling of non-saturated networks is introduced in [C19] and [C21], where the authors have chosen to map the idle time of a node to a longer backoff period. This approach keeps the simplicity of a saturated network model by not explicitly representing idle states, and yet allows the study of unsaturated nodes. Kai and Zhang [C19] propose a model that calculates the throughput of non-saturated CSMA/CA networks with arbitrary topologies. Laufer and Kleinrock [C21] use a similar model to estimate the throughput of a node in a fully-connected CSMA/CA network using the ratio between the transmitting and the backoff periods of that node, its probability of successful transmission, and the channel capacity. The result is then used in the analysis of a network's capacity region, based on nodes' throughputs, under stability conditions.

In this chapter, we study unsaturated, not fully-connected IEEE 802.11 wireless networks. We present a conflict graph-based modeling approach to discover the attainable throughput of each node. We apply a Divide-and-Conquer approach resulting in a series of Markov chains that together describe, at a high-level of abstraction, the current state of the entire wireless network. The conceptual simplicity of our model allowed us to fully automatize the procedure and to test it on networks of different sizes and topologies.

3.5 System Description

The system we consider is a Wireless Local Area Network (WLAN). Here, WLAN refers to any wireless network that implements the IEEE 802.11 standard in the Physical (PHY) and Medium Access Control (MAC) layers. IEEE 802.11 standards are accompanied by a series of amendments. Each amendment serves as an addition to the IEEE 802.11 standard and is developed to either modify the standard's PHY and MAC characteristics or offer additional functionalities not implemented in the basic standard. Examples of

these are the IEEE 802.11g that enhances the physical layer of the standard (and preceding amendments), and the IEEE 802.11i which offers additional security features.

In terms of the physical layer, every IEEE 802.11 standard amendment has a set of *transmission rates* that represent the physical rates at which a node can send data over the channel. The wireless channel is generally imperfect and highly affected by its environment. Therefore, when a transmission rate is chosen for a communication between two nodes the goal is to have the highest possible transmission rate while keeping a low error rate [C6]. A node can also choose which wireless channel it wishes to use. IEEE 802.11 standard amendments generally use two distinct frequency bands: the 2.4GHz and the 5GHz [C33]. In the 2.4GHz band, a node can choose from up to 14 wireless channels. Out of those 14 channels, only three occupy non-overlapping frequencies i.e., can be used simultaneously without collisions. In the 5GHz band, there can be up to 24 non-overlapping channels, meaning that as many as 24 transmissions can happen simultaneously in close proximity.

WLANs use the DCF in the MAC layer. DCF makes sure that a node that wishes to start a transmission senses an idle medium before the beginning of that transmission. This procedure is employed with the help of the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism in two distinct steps. First, before every transmission a node must sense the medium idle for the duration of a DIFS period whose length depends on the IEEE 802.11 standard amendment. Then, the node starts a countdown timer called the *backoff* period. The backoff has a random duration that helps desynchronize the beginnings of transmissions of neighboring nodes. Nevertheless, its average duration depends on the IEEE 802.11 amendment in use and we denote it by $T_{backoff}$. Additionally, for unicast frames only, DCF uses a short silent period called SIFS followed by an acknowledgment frame sent by the destination, serving as a confirmation that a frame was correctly received. Additional details on DCF can be found in the Appendix of [C32].

Each WLAN is composed of N nodes representing the Access Points (APs) and the user stations. A node's *Carrier-Sensing* (CS) zone contains all other nodes whose transmissions can be detected and a fortiori belong to the same channel. DCF, through its carrier sensing and backoff mechanisms, attempts to ensure that two nodes that belong to each others' CS zones do not simultaneously transmit, and instead, have to share the available transmission capacity. Nevertheless, collisions occur whenever two such nodes are simultaneously transmitting, potentially resulting in the loss of one or both transmissions. More precisely, a collision happens whenever the backoff countdowns of the two nodes finish (approximately) at the same time. CS zones play an important role in evaluating the WLAN's global transmitting capacity as they determine with whom and how nodes have to share the medium/channel capacity. Therefore, we can use the CS zones to represent a WLAN as a *conflict graph* in which two vertices share an edge if the corresponding network nodes belong to each others' CS zones. Although nodes can interfere even beyond their CS zones, Padhye *et al.* [C27] showed that the major source of collisions (interference) are nodes that belong to the CS zone. Note that the same authors also developed a pairwise interference measurement that can be used to discover a WLAN's conflict graph.

We refer to nodes that share a link in the conflict graph as *neighbors*. We denote by v_n the set of n's neighboring nodes (excluding n), referred to as n's neighborhood. In our work, we assume that the CS zones are symmetrical, meaning that if node m belongs to node n's CS zone, then node n belongs to node m's CS zone. Figure 3.1 shows the conflict graph of a four-node network. We notice that in this network nodes 1 and 4 can

simultaneously transmit as they do not detect each other transmissions. On the other hand, nodes 1 and 2 cannot transmit at the same time without causing a collision resulting in a potential loss of frame(s).

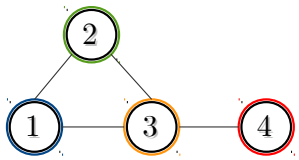


Figure 3.1: Conflict graph of a four-node network.

Note that, like in [C13], only network's APs are considered in the conflict graphs (user stations are not). Because of the traffic asymmetry where downloads from APs vastly outweigh uploads [C10], the set of APs provides a convenient, though approximate, description of a WLAN. We tested the validity of this assumption through simulations. Our results suggest that user stations can be disregarded in the conflict graph at the cost of a limited loss of accuracy (say around than 5%). For example, Fig. 3.1 may describe a network in which four APs are transmitting traffic to some user stations, which are mostly receiving traffic rather than generating traffic so that they can be dismissed from the graph.

Note also that a conflict graph involves only nodes that belong to the same channel. As a result, an average WLAN working in the 2.4GHz band (with three non-overlapping channels) is typically represented by three conflict graphs, each containing roughly a third of the total number of APs in the WLAN. With a maximal number of 24 non-overlapping channels in the 5GHz band, the corresponding conflict graphs would be even smaller.

We use x_n to denote the normalized *input rate* of node n. The higher x_n, the larger the demand of node n for throughput. On the other hand, we let y_n indicate the normalized *output rate* of node n. By definition, we have: $x_n \leq 1$ and $y_n \leq 1$, and, because in the long run the output rate of a node cannot exceed its input rate, it follows: $y_n \leq x_n$. Note that $y_n = 0$ indicates that node n never gets access to the channel (i.e., a starving node), while $y_n = 1$ signifies that the node permanently occupies the channel (be it in active transmission or with DCF overhead). Note also that y_n can be easily derived from the actual average throughput of a node (typically expressed in Mbps) by simply normalizing the latter by the maximum throughput achievable by the node (i.e., when all its neighbor nodes are silent). More precisely, we have:

$$y_n = \frac{t_n}{t_{n,max}} , \tag{3.1}$$

where t_n denotes the throughput achieved by node n, and $t_{n,max}$ is the maximum throughput node n can achieve, calculated as:

$$t_{n,max} = \frac{L \times 8}{T_{backoff} + T_{DIFS} + T_{PHY} + T_{FRAME} + T_{SIFS} + T_{PHY} + T_{ACK}} . \tag{3.2}$$

In Eq. (3.2) T_{PHY} is the duration of the physical layer's header, L is the mean payload length, T_{FRAME} is the total frame transmission time, including all headers brought by the MAC, Network, and Transport layers, $T_{backoff}$, T_{DIFS}, T_{SIFS}, T_{ACK} are all overhead times present in DCF. Note that the maximum throughput depends on the standard amendment, mean payload length, and transmission rate, while being independent of the network's topology or nodes' input rates.

System	
N	Total number of nodes
v_n	Node n's neighborhood
x_n	Input rate of n-th node, $x_n \in [0,1]$
y_n	Output rate of n-th node, $y_n \in [0,1]$
L	Mean payload length (in bytes)
$t_{n,max}$	Maximal throughput node n can achieve if all its neighbors are silent (in Mbps)
t_n	Achieved throughput of node n (in Mbps)

Model	
B	Set of possible subnetworks, $B = \{1,0\}^N$
b_i	i-th subnetwork, $b_i \in B$
$b_i(n)$	Regime of the n-th node in subnetwork b_i, $b_i(n) \in \{ON, OFF\}$
β_i	Occurrence probability of the i-th subnetwork
S	Set of possible sending states, $S \subseteq \{\boldsymbol{0,1}\}^N$
S_i	Set of sending states associated to subnetwork b_i, $S_i \subseteq S$
s_k	k-th sending state, $s_k \in S$
$s_k(n)$	State of the n-th node in sending state s_k, $s_k(n) \in \{\boldsymbol{1,0}\}$
$P_{k,\ell}$	Probability of the transition from sending state s_k to s_ℓ
w_n	Restricted set of neighbors of node n with blocked nodes removed
M_i	Number of irreducible Markov chains for the subnetwork b_i
c_i^m	m-th irreducible Markov chain of subnetwork b_i
S_i^m	Set of sending states associated to c_i^m, $S_i^m \subseteq S_i$
π_i^m	Steady-state probability distribution of c_i^m
π_i	Steady-state probability distribution of subnetwork b_i
ω_i^m	Occurrence probability of c_i^m

Table 3.1: Principal notation.

3.6 Model and its Solution

For the sake of clarity, when presenting our modeling framework, we resort to the sample network depicted in Fig. 3.1 to show its step by step execution.

3.6.1 Decomposing into subnetworks

In any network, nodes typically alternate their activity between ON and OFF periods. When in the ON regime, a given node n ($n = 1, \dots N$) has at least one frame to send, and thus has a non-empty buffer. In other words, an ON node is either transmitting or wishing to start a transmission. In the OFF regime, a node's buffer is empty. We consider that the nodes' regimes, and consequently their input rates, are independent of each other. In practice, a node may postpone the transmission of a frame because of the activity of its neighbors, thus extending its ON period. In order to keep the model tractable, we decided to omit the potential dependencies among the nodes' ON periods.

At any time, the state of the network activity can be described by a vector of length N, where the n-th element expresses the current regime of node n (be it ON or OFF). Thus, for a network with N nodes, there are 2^N such vectors that correspond to all the possible combinations of the two regimes over the N nodes.

In our work, we apply a Divide-and-Conquer approach by choosing to analyze the network not as a single complex network in which any node can alternate between ON and OFF, but rather as a collection of 2^N simpler networks in which every node is either ON or OFF. We refer to these new networks as the *subnetworks* and we denote them by $b_1, b_2, ..., b_{2^N}$. Hence $b_i(n)$ indicates the regime of node n in subnetwork b_i. We use B to designate the set that contains all subnetworks.

For the sample network of Fig. 3.1, as well as for any other four-node network, there is a total of 16 such subnetworks:

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{16} \end{bmatrix} = \begin{bmatrix} OFF & OFF & OFF & OFF \\ OFF & OFF & OFF & ON \\ \vdots & \vdots & \vdots & \vdots \\ ON & ON & ON & ON \end{bmatrix} \tag{3.3}$$

We refer to the probability that the current state of the network is subnetwork b_i as the *occurrence probability* of b_i and we denote it by β_i ($i = 1, \ldots, 2^N$). Note that a subnetwork's occurrence probability depends only on the nodes' input rates and can be calculated as:

$$\beta_i = \prod_{n|b_i(n)=ON} x_n \prod_{m|b_i(m)=OFF} (1 - x_m) \ . \tag{3.4}$$

For example, in our four-node network, one of the possible subnetworks is $b_{14} = [ON\ ON\ OFF\ ON]$. This subnetwork represents the case when nodes 1, 2, and 4 are in transmission or have a frame waiting to be sent, while node 3 has an empty buffer. Its occurrence probability is calculated as:

$$\beta_{14} = x_1 x_2 (1 - x_3) x_4. \tag{3.5}$$

Figure 3.2 shows a schematic representation of the entire solution where Stage 1 corresponds to breaking down the network into several subnetworks. We will now show how to solve each of the subnetworks separately and independently of the rest of the subnetworks.

3.6.2 Solving each subnetwork as one or more Markov chain(s)

We now detail how we analyze the behavior of each subnetwork using Markov chains. We start by defining the possible states and transitions of the corresponding Markov chains. Note that, in this subsection, the subject of study is any of the subnetworks b_i ($i = 1, \ldots, 2^N$) resulting from the network decomposition (see discussed above).

Defining the possible states for the subnetwork

While the regime (ON or OFF) of each node is set and fixed (for the considered subnetwork b_i), knowing the regime is not sufficient to determine if a node is currently sending a frame or not. Indeed, an ON node can be either transmitting or waiting for the medium to become idle. We eliminate this ambiguity by introducing the notion of sending states.

Like a subnetwork, *a sending state* is a vector of length N whose n-th element refers to the activity of the n-th node. However, unlike a subnetwork, a sending state indicates for

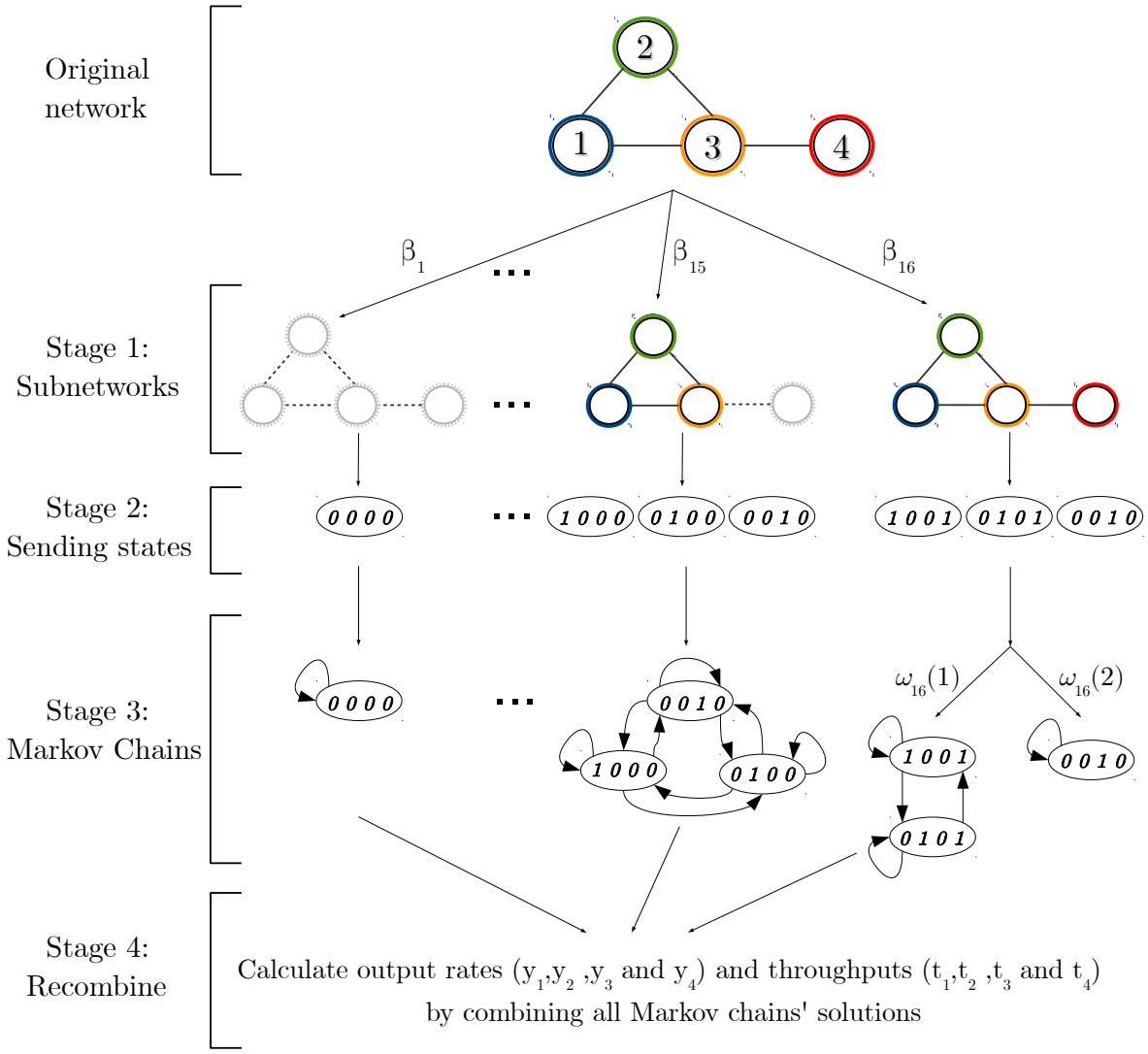


Figure 3.2: Schematic representation of the proposed solution.

each node n if the node is transmitting (marked $\boldsymbol{1}$) or not (marked $\boldsymbol{0}$). Let s_k denote the k-th sending state (with $k = 1, \ldots,$). Thus, if node n is currently transmitting we have $s_k(n) = \boldsymbol{1}$, and $s_k(n) = \boldsymbol{0}$ otherwise, for $n = 1, \ldots, N$. Note that $s_k(n) = \boldsymbol{0}$ means that node n is either OFF, or ON but waiting access for transmission. While in theory, the total number of sending states for each subnetwork is equal to 2^N, in practice this number is much smaller as we consider only a fraction of them to be possible. Let S denote the set of all possible sending states over all existing subnetworks. Each possible sending state must comply with a common property of CSMA/CA: neighboring nodes cannot transmit successfully at the same time, i.e., if the conflict graph contains an edge between nodes n and $n + 1$, then $s_k(n)$ and $s_k(n + 1)$ cannot both be equal to $\boldsymbol{1}$. Next, we designate by S_i the set of possible sending states associated to the subnetwork b_i. Note that we can easily determine S_i since S_i is a subset of S whose elements satisfy the following properties: (i) if $b_i(n) = ON$ and node n has no transmitting neighbors, then $s_k(n) = \boldsymbol{1}$; (ii) if $b_i(n) = OFF$, then $s_k(n) = \boldsymbol{0}$. Note that the rationale behind the second property is quite straightforward: a node that has no frames to be sent cannot be sending. The first property derives from a phenomenon studied in [C11]: CSMA/CA networks tend to increase the spacial reutilization of the medium by maximizing the number of simultaneous

transmissions. As a result, in our model, we enforce any node that is ON and senses an idle medium to be in transmission.

In the case of our sample network, the subnetwork $b_{16} = [ON\ ON\ ON\ ON]$ has three possible sending states, $s_1 = [\boldsymbol{1\ 0\ 0\ 1}]$, $s_2 = [\boldsymbol{0\ 1\ 0\ 1}]$, and $s_3 = [\boldsymbol{0\ 0\ 1\ 0}]$. Note that other sending states may exist but we consider them to be negligible in b_{16}. For example, the sending state $[\boldsymbol{1\ 1\ 0\ 1}]$ breaks the CSMA/CA condition, as nodes 1 and 2 are neighbors and cannot be simultaneously transmitting. The sending state $[\boldsymbol{1\ 0\ 0\ 0}]$ is deemed not possible since node 4 breaks the first condition. Indeed, b_{16} indicates that node 4 is ON, and because it has no sending neighbors, it should be sending its frames. This step of determining the sending state is illustrated by Stage 2 of Fig. 3.2.

Determining the possible transitions

The set of sending states found for the subnetwork b_i, namely S_i, will serve as the states of a Markov chain. We now detail how we decide which transitions are possible between those sending states. Our reasoning is based on the idea that, in a CSMA/CA network, the probability of two nodes starting (or ending) their transmission at the exact same time is negligibly small. We translate this CSMA/CA property into the following rule for our modeling purpose. Let s_k and s_ℓ be two possible sending states of S_i. The transition from sending state s_k to s_ℓ is deemed possible if and only if s_k and s_ℓ both verify that:

1. no more than one node alters from $\boldsymbol{1}$ in s_k to $\boldsymbol{0}$ in s_ℓ, and
2. no more than one node alters from $\boldsymbol{0}$ in s_k to $\boldsymbol{1}$ in s_ℓ.

Note that a self-transition on a given sending state s_k is always possible, as it implies no changes in the sending state.

For example, in our four-node network it is possible to go from sending state $[\boldsymbol{1\ 0\ 0\ 1}]$ to $[\boldsymbol{0\ 1\ 0\ 1}]$, as in this transition node 1 ends and node 2 starts a transmission. However, it is not possible to go from network state $[\boldsymbol{1\ 0\ 0\ 1}]$ to $[\boldsymbol{0\ 0\ 1\ 0}]$, as it implies both nodes 1 and 4 ending their transmissions at the exact same time. Figure 3.3 shows the existing transitions in our modeling framework between the possible sending states associated to the subnetwork b_{16}.

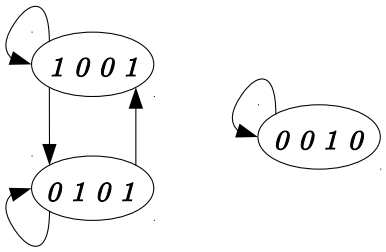


Figure 3.3: Possible sending states and corresponding existing transitions associated to the subnetwork $b_{16} = [ON\ ON\ ON\ ON]$.

Calculating the transition probabilities

We now explain how we determine the probability of the transitions between the possible sending states s_k composing our Markov chain. Note that non-possible transitions have

zero probability. To evaluate the non-zero transition probabilities, we need to introduce our definition of a *blocked node*. A node having at least one of its neighbors currently transmitting is said to be blocked as it is unable to start a collision-free transmission. For example, in the four-node network, node 3 can be blocked by the transmissions of any of the other three nodes.

We can now calculate $P_{k,\ell}$, the probability of the transition from sending state s_k to s_ℓ, as:

$$P_{k,\ell} = C \prod_{n|s_\ell(n)=\mathbf{1}} \frac{1}{1 + \sum_{m \in w_n} \mathbb{1}_{b_i(m)=ON}}, \tag{3.6}$$

where C is a normalizing constant such that $\sum_{\ell \geq 1} P_{k,\ell} = 1$, and w_n defined as $w_n = \{m \in v_n \backslash \{n\} \mid m \text{ is not blocked in } s_\ell \text{ by a node } \in v_m/\{n\}\}$ is the restricted neighborhood of node n, i.e., w_n contains all neighbors of n that are not blocked by some node different from node n. As an example, in the subnetwork $b_{16} = [ON\ ON\ ON\ ON]$ and the sending state $[\mathbf{1\ 0\ 0\ 1}]$, the restricted neighborhood of node 1 contains only node 2, as node 3 is blocked by node 4.

Note that the indicator function $\mathbb{1}_{b_i(m)=ON}$ returns 1 if $b_i(m) = ON$, and 0 otherwise. The underlying logic behind Eq. (3.6) is that all nodes that are ON (whether they are sending or not) compete with their neighbors for accessing the medium. We also consider them equally likely to gain the medium access. On the other hand, nodes that are OFF do not affect the transition probability because they do not compete for medium access.

For instance, when node 3 of the subnetwork b_{16} in Fig. 3.2 competes with nodes 1, 2, and 4, it has a $\frac{1}{4}$ chance of gaining the medium. However, in this same scenario node 4 competes with only one neighbor, so its chance of gaining access would be $\frac{1}{2}$.

Calculating the steady-state probabilities

At this stage, the Markov chain associated to subnetwork b_i is fully characterized and we can calculate its steady-state probabilities. Let us remind that a Markov chain is *irreducible* if from any of its states there is a way to reach any other state. Depending on the setting of the subnetwork under study, the corresponding Markov chain may or may not be irreducible. Should the Markov chain not be irreducible, we consider each irreducible Markov chain separately. We denote by M_i the number of irreducible Markov chains in subnetwork b_i. For example, as shown by Fig. 3.3, the subnetwork $b_{16} = [ON\ ON\ ON\ ON]$ contains two irreducible chains, i.e., $M_{16} = 2$ (since it is not possible to go from the sending state $[\mathbf{0\ 0\ 1\ 0}]$ to the other two sending states). We use c_i^m, $m \in [1, ..., M_i]$, to denote the m-th irreducible chain of subnetwork b_i. Hence the left-hand chain of Fig. 3.3 is denoted by c_{16}^1 and the right-hand chain is c_{16}^2.

We compute the steady-state probabilities of each irreducible chain c_i^m for the subnetwork b_i and we denote by π_i^m the vector containing the corresponding values. Note that we use S_i^m to refer to the set of sending states in chain c_i^m (while, as defined previously, S_i denotes the set of possible sending states associated to b_i). If the subnetwork has a single irreducible Markov chain ($M_i = 1$), then it follows that $S_i^1 = S_i$. Thus, the steady-state probabilities of the subnetwork's sending states are equivalent to those of the Markov chain c_i^1 and we have $\pi_i = \pi_i^1$, where π_i is the vector containing the steady-state probabilities of subnetwork b_i. In this case, we can skip the end of this section and directly proceed to Section 3.6.3.

Combining several irreducible Markov Chains

For subnetworks that contain more than one irreducible Markov chain, we need to combine the steady-state probabilities found for each Markov chain into the steady-state probabilities for the whole subnetwork b_i. We introduce a *weighting factor*, denoted by ω_i^m, to express the probability that this particular irreducible Markov chain c_i^m is initially chosen. For the sake of conciseness, we omit the calculation of these weighting factor (see [C32] for their computations).

To obtain the π_i (steady-state probabilities of the subnetwork b_i) from the π_i^m's (steady-state probabilities found for each of the M_i irreducible Markov chains associated with b_i) we simply proceed as follows:

$$\pi_i = [\pi_i^1 \times \omega_i^1,\ ...,\ \pi_i^{M_i} \times \omega_i^{M_i}]. \tag{3.7}$$

In other words, π_i is obtained as a weighted sum of π_i^m's.

Adjusting to the IEEE 802.11 parameters

To account for the value of the mean length of frames sent over the network, the transmission rates of wireless channels, as well as the particular amendment of IEEE 802.11 in use, we refine the computation of the weighting factors w_i^m. Although we skip this part here, we refer the reader to [C32] for its complete description.

3.6.3 Combining subnetwork solutions

So far we have divided the network into subnetworks and solved each one of them separately. The last phase of the model consists in combining the results obtained for different subnetworks and calculating the nodes' output rates. A node's output rate represents the portion of time when the node is occupying the medium, including the frame transmission itself and all the necessary DCF overhead. Thus, we calculate node n's output rate, y_n, as:

$$y_n = \sum_{i=1}^{|B|} \left\{ \mathbb{1}_{b_i(n)=ON} \times \beta_i \times \sum_{m=1}^{M_i} \left(\omega_i^m \times \sum_{k|s_k \in S_i^m} \left(\mathbb{1}_{s_k(n)=\mathbf{1}} \times \pi_i^m(k) \right) \right) \right\}, \tag{3.8}$$

Eq. (3.8) gives the sum of the stationary probabilities of all the sending states in which node n is sending, times the occurrence probabilities of all the irreducible Markov chains in which those states appear, times the occurrence probabilities of the subnetwork to which those chains belong. Otherwise stated, it is simply the sum product of the probabilities of all the *subnetwork* $\times$ *Markov chain* $\times$ *sending state* combinations in which node n is sending.

We can now transform node n's output rate into its obtained throughput, t_n. When all the network nodes use the same standard amendment, transmission rate, and mean payload length, then all nodes have an equal maximum achievable throughput, $t_{n,max}$, and we can calculate the throughput of node n as:

$$t_n = y_n \times t_{n,max}\ . \tag{3.9}$$

Note that when nodes have different maximum achievable throughput, $t_{n,max}$, Eq. (3.9) needs to be slightly readjusted. The exact formula can be found in [C32].

3.7 Numerical Results

We start this section by assessing the accuracy of the proposed modeling approach by comparing the model outcomes with those delivered by a discrete-event simulator under various scenarios. Then, we study the computational complexity of the modeling approach as a function of the network topology. At last, we explore a possible application of the model relating to the configuration and performance improvement of a WLAN.

3.7.1 Model validation

To evaluate the accuracy of the proposed model, we explore several scenarios with different values of various network parameters, such as the IEEE 802.11 standard, the mean frame length, the transmission rate, the topology and size of the network, and we compare the model's estimations to the simulation results delivered by the discrete-event network simulator *ns*-3 [C25].

Various network topologies and standard amendments

We begin by examining the proposed model's accuracy under different topologies. We consider two topologies: the six-node network depicted in Fig. 3.4a, and the ten-node network of Fig. 3.5a. Recall that the nodes of a conflict graph represents only the access points (APs) that belong to the same communication channel. Typically, the original WLAN contains several other APs operating on other channels. Besides, as discussed in Section 3.5, only APs (and not user stations receiving traffic from APs) appear in conflict graphs. Nonetheless, in the simulator, APs transmits traffic to associated user stations.

To account for the intrinsic uncertainty of the measured quantities in a simulator, we replicate each simulation 20 times and we calculate the 95% confidence intervals. However, given the length of the simulation runs and the number of replications, the computed confidence intervals are virtually indistinguishable from their mean values and we decided to not represent them in the following figures.

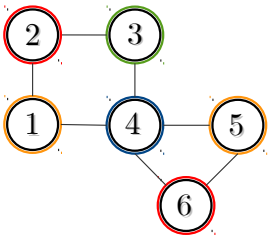
Scenario	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Six-node	0.5	0.4	0.6	0.3	0.7	0.9	/	/	/	/
Ten-node	0.3	0.6	0.8	0.4	0.7	0.3	0.5	0.4	0.9	0.7

Table 3.2: Input rates of nodes by default in scenarios.

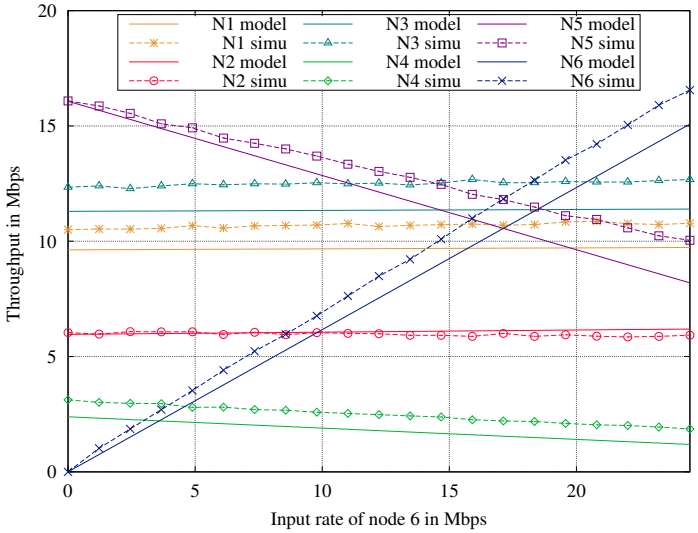
Six-node network Our first scenario deals with a network composed of 6 nodes and the IEEE 802.11n amendment whose transmission rate is set to 65Mbps. The network topology is depicted in Fig. 3.4a. Figure 3.4b shows the throughputs attained by each of the six nodes as the input rate of node 6 gradually increases from 0 to 1. Note that the input rates of the other nodes are given in Table 3.2.

Not surprisingly, we observe in Fig. 3.4b that the throughputs of nodes 4, 5, and 6 are the most affected by the increasing input rate of node 6, as all three belong to the same clique. Node 6 increases its throughput mostly at the expense of node 5, that loses more than a third of its original throughput. Node 4's throughput decays to a lesser extent, however its already small throughput is even further decreased. We also notice that nodes

1, 2, and 3 are not directly affected by node 6 and they keep an almost steady throughput regardless of the value of x_6.



(a) Conflict graph.



(b) Accuracy: varying the input rate of node 6, x_6.

Figure 3.4: Six-node network.

For the sake of completeness, we repeat the same scenario five other times but each time we keep x_6 steady to its default value (see Table 3.2) and we let the input rates of one of the other nodes vary from 0 to 1 by step of 0.05. This gives us a total of $21 \times 6 \times 4 = 756$ points, out of which we derived the statistics on the relative error shown in Table 3.3. We notice that in over 90% of the samples the relative error is less than 20% and that the mean relative error of less than 10%.

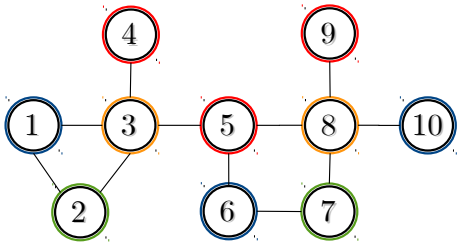
Scenario	**Mean**	**Median**	**<5%**	**<10%**	**<20%**	**<30%**	**>30%**
Six-node	9.80%	9.77%	21.10%	54.91%	97.75%	99.20%	0.80%
Ten-node	9.11%	7.47%	27.78%	68.12%	88.77%	99.36%	0.64%

Table 3.3: Distribution of the relative error for the throughput, t_n.

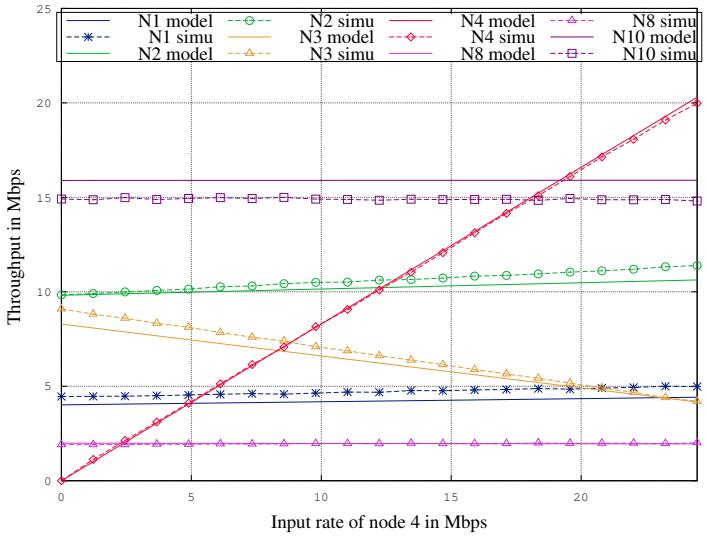
Ten-node network Our second scenario involves a network of 10 nodes (see Fig. 3.5a) using IEEE 802.11n. We study the throughput attained by all nodes as a function of node 4's input rate. Again, the input rates of the other nodes are given in Table 3.2.

Figure 3.5b shows the corresponding results. In order to keep the figure legible, we represent the attained throughput only for a subset of nodes. First, we observe that the variation of node 4's input rate causes its throughput to increase from 0 to approximately 20Mbps. On the other hand, as x_4 grows, the throughput of node 3 decays significantly (nearly halved). This agrees with the fact that node 3 is the only neighbor of node 4 (see Fig. 3.5a). Because of node 3's declining throughput, nodes 1 and 2 experience a slight gain in their throughput as x_4 grows. As for the nodes far from node 4 such as nodes 8 and 10, their attained throughput is almost not influenced by the variations in x_4. Finally, Fig. 3.5b shows that our model manages to capture all these behaviors with a good level of precision. Like in the two former scenarios, we repeat the same experiences letting the input rate of each node in lieu of x_4 vary from 0 to 1. This leads to a total

of 2100 samples that we use to compute the statistics shown in Table 3.3. Here also, we observe that the mean and median relative errors are both less than 10%. In the vast majority of examples (almost 90% of cases), the relative error made by our model is less than 20%.



(a) Conflict graph.



(b) Accuracy: varying the input rate of node 4, x_4.

Figure 3.5: Tend-node network.

Note that we obtain similar results on other common network topologies using different IEEE standard amendments (see [C32]).

Heterogeneous transmission rates

We now study the case where network nodes are heterogeneous with regard to their transmission rates. To do that, we reconsider the six-node node network but we assign a different transmission rate to every node as indicated by Table 3.4. Note that under these settings, node 2 has a transmission rate that is five times that of node 5. Analogously to the former scenarios, we let the input rate of each node vary from 0 to 1 while keeping the input rates of the other nodes to their default values (see Table 3.2). This gives us a total of 756 cases on which we calculated the estimated throughputs using our model and compare these values to those delivered by the simulator.

Scenario	**N1**	**N2**	**N3**	**N4**	**N5**	**N6**
Six-node	18 Mbps	54 Mbps	24 Mbps	12 Mbps	9 Mbps	12 Mbps

Table 3.4: Transmission rates for the nodes of the six-node network in Fig. 3.4a.

Table 3.5 presents the corresponding results. We notice that despite having nodes with significantly different transmission rates, our model is still able to deliver accurate estimations for the throughput. More precisely, the mean relative error of the model is 9% with 94% of the samples having an error less than 20%.

Scenario	**Mean**	**Median**	**<5%**	**<10%**	**<20%**	**<30%**	**>30%**
Six-node	9.31%	6.98%	20.29%	68.49%	94.13%	97.73%	2.27%

Table 3.5: Heteregenous transmission rates: distribution of the relative error for the throughput, t_n.

Frame aggregation in IEEE 802.11n

In our last scenario, we study the model's precision when the nodes implement the aggregation feature. When frame aggregation is enabled, multiples frames are concatenated into a single large frame before being transmitted. This tends to diminish the cost of the overhead introduced by the MAC protocol, thereby increasing the maximal achievable throughput.

We consider again the six-node network of Fig. 3.4a with the input rates given in Table 3.2. However, all six nodes now aggregate four MAC service data units (MSDUs) into a single frame at each transmission. While the simulator actually implements the aggregation features, in our model we simply extended by a factor of 4 the length of frames.

Figure 3.6 shows the attained throughputs of all nodes as a function of the input rate of node 6. We can assess the influence of the frame aggregation feature on this scenario by comparing Fig. 3.4b and Fig. 3.6. Although the trends exhibited by the throughputs are still comparable, we observe that the frame aggregation feature significantly increases (almost doubles) the attained throughput. Finally, we included in Table 3.6 the mean, median, and distribution of the relative error when we let another node than node 6 vary its input rate. Figure 3.6 along with Table 3.6 show that our modeling approach can successfully handle the frame aggregation and capture its effects.

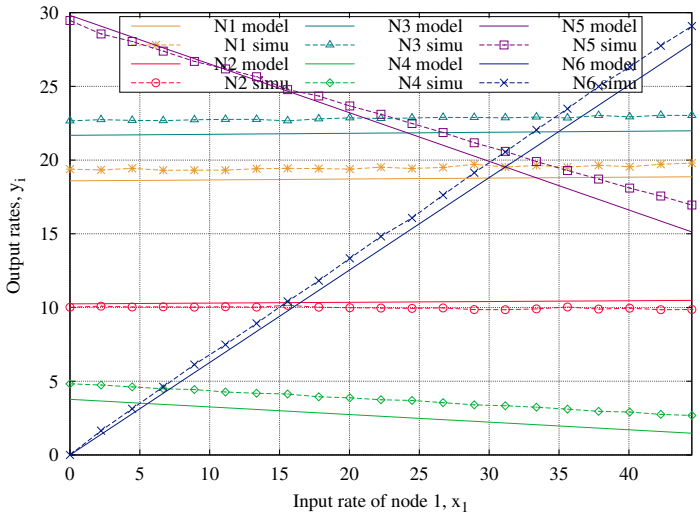


Figure 3.6: Frame aggregation on six-node network: varying the input rate of node 6.

Scenario	**Mean**	**Median**	**<5%**	**<10%**	**<20%**	**<30%**	**>30%**
Six-node	6.04%	5.07%	49.06%	92.03%	98.28%	99.22%	0.78%

Table 3.6: Frame aggregation: distribution of the relative error for the throughput, t_n.

3.7.2 Modeling complexity

In this section, we explore how the computational complexity of our modeling framework increases as the size of the WLAN under study grows. Unlike many existing modeling approaches [E17, C11, C19, C13] that make use of a single Markov chain to describe the whole network behavior, ours revolves around a Divide-and-Conquer approach. Indeed, our approach breaks the original problem into a set of smaller problems, each being solved individually thanks to the solution of a smaller Markov chain.

Unfortunately, we were not able to derive a closed-form expression (nor a tight upper bound) for the number of states in the Markov chains involved in our modeling approach. This exercise is made difficult as the exact value of the number of states depends significantly not only on the number of nodes in the network, N, but also on the network's density, aka the average node degree. We nonetheless provide an empirical study.

We randomly generate thousands of conflict graphs with size varying from $N = 5$ up to $N = 14$. We sort them into five groups based on their density: average node degree of less than 3, between 3 and 4, between 4 and 5, between 5 and 6, and between 6 and 7. Then, for each interval of network density, we calculate the mean number of (sending) states per Markov chain. Figure 3.7 shows the corresponding results for a number of nodes in the network varying from $N = 5$ to $N = 14$. As expected, the average number of sending states per subnetwork grows with increasing values of N. However, even for $N = 14$, the mean number of states per Markov chain tends to lie around 8, meaning that most involved Markov chains are very small.

Finally, for the sake of comparison, we included as a subplot in Fig. 3.7 the number of states in the Markov chain if one uses a classical description such as [E17, C11, C19, C13]. The actual number values were found using a previous work of ours [C31] that relies on a single large Markov chain to describe the whole network behavior. As expected, the mean number of states for the Markov chain is significantly larger (say two orders of magnitude) when using a single Markov chain as opposed to a series of smaller Markov chains, and can lead up to several hundreds of states when the number of nodes closes 14. Hence, we chose to have a large number of smaller Markov chains, keeping in mind that the last stage of our approach, aiming to combine the solutions found for each subnetwork, is a simple summation of the stationary probability distributions over all the subnetworks using the law of total probability [D18].

Overall, by splitting the original problem into many smaller problems, whose solutions can be easily parallelized, our Divide-and-Conquer strategy circumvents the dimensionality curse associated to large Markov chain for conflict graphs having up to a dozen or so nodes. In practice, with a non-optimized implementation, models are typically solved at a click-speed for N around 4 or 5, and within a couple of seconds for N near to 10. We remind that our conflict graphs contain only APs belonging to the same channel, and that, depending on the IEEE 802.11 standard amendment in use, there can be from three to 24 non-overlapping channels.

3.7.3 Possible application: Channel assignment

We illustrate a potential application of our model by studying the well-known issue of channel assignment in an IEEE 802.11 WLAN. In IEEE 802.11n and 802.11g networks, each AP can choose its channel among 14 different wireless channels in the 2.4GHz frequency range. However, out of these 14 channels, at most three can be chosen in a

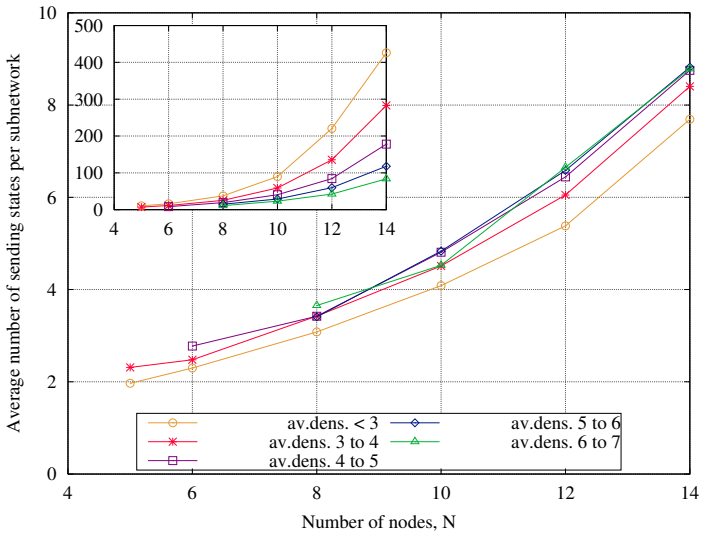


Figure 3.7: Number of (sending) states per Markov chain (subnetwork) as a function of the network's size and density.

manner that no channels have overlapping frequencies [C33]. Obviously, given the way APs share the channel, the choice of channel assignment considerably affects the network's performance.

We consider the 12-node network ($N = 12$) depicted in Fig. 3.8a with three non-overlapping channels. The input rates of nodes are given in Table 3.7. and low-demanding nodes whose input rates are below 0.5. Let a be a vector of length N that represents one possible allocation of the three channels among the N APs. We denote by $y(a) = \{y_1, y_2, \ldots, y_N\}$ the set of output rates obtained when implementing the channel assignment a. Remind that y_i can be viewed as a measure of the normalized throughput attained by node i.

We consider four different performance metrics to evaluate the performance of the network:

1. The global satisfaction rate, GSR, or the proportion of the network's general throughput demand that has been met, calculated as: $GSR(y(a)) = \frac{\sum_{n=1}^{N} y_n}{\sum_{n=1}^{N} x_n}$.

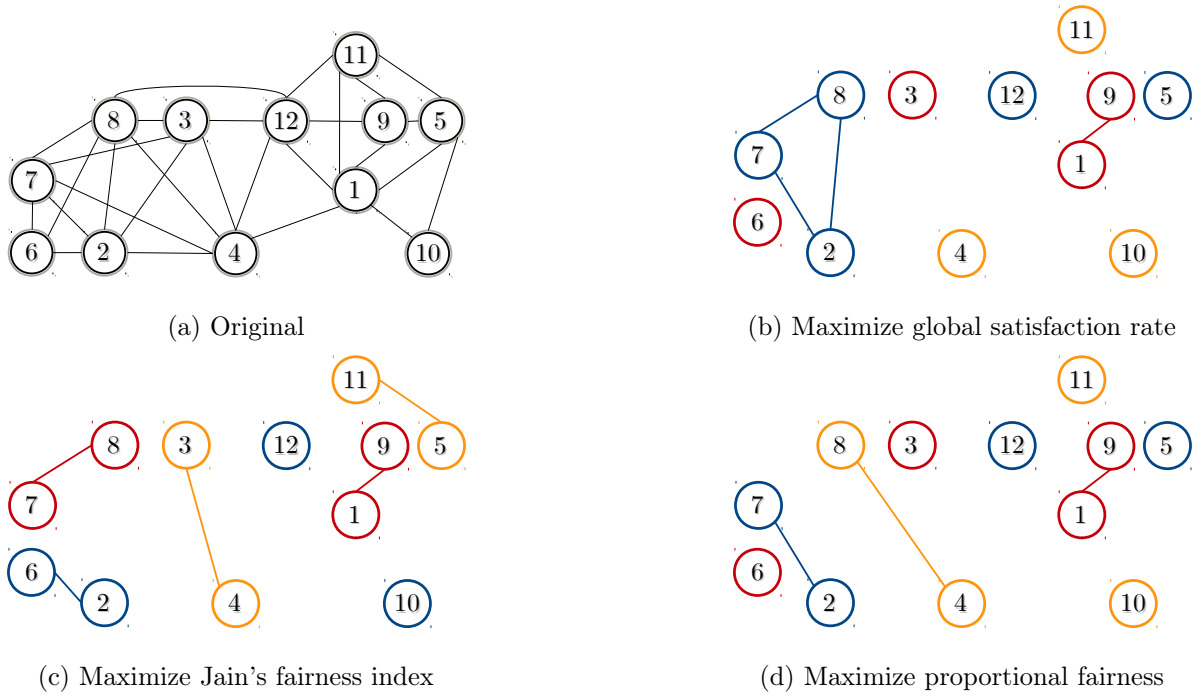
2. The Jain's fairness index [C17], J, that measures how fairly the throughput was divided among the nodes. Jain's index is a quantity in the interval $[0, 1]$, where 1 represents the highest fairness, meaning all nodes get an equal share. It is calculated as: $J(y(a)) = \frac{\left(\sum_{n=1}^{N} y_n\right)^2}{\sum_{n=1}^{N} y_n^2}$ Additionally, we can calculate the Normalized Jain's index, NJ. The normalization refers to accounting for the nodes' input rates when calculating Jain's index: $NJ(y(a)) = \frac{\left(\sum_{n=1}^{N} \frac{y_n}{x_n}\right)^2}{\sum_{n=1}^{N} \frac{y_n}{x_n}^2}$.

3. The proportional fairness, PF, that is a trade-off between GSR and J as it tries to maximize both fairness and throughput by giving more throughput to nodes with higher demands: $PF(y(a)) = \sum_{n=1}^{N} log\frac{y_n}{x_n}$.

Scenario	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
12-node	0.2	0.4	0.9	0.7	0.8	0.9	0.1	0.3	0.2	0.6	0.8	0.3

Table 3.7: Input rates of the 12-node network in Fig. 3.8a.

In practice, our model could be used jointly with existing solutions in the field of channel allocation, such as [C9, C28]. A classical way of finding (sub)optimal channel allocations is to start from a given allocation, and then iteratively improve it with regard to some network performance parameters until convergence is found. In this regard, our model could be used to quickly evaluate the performance parameters of interest at each iteration (rather than relying on long simulations). However, for the sake of simplicity and given the size of the network, we choose to explore all of the $3^{12} \simeq 530,000$ possible allocations and retain the ones maximizing one of the criteria given above. Figures 3.8b, 3.8c, and 3.8d illustrate the channel assignment that maximize GSR, J, and PF, respectively. For each of these three channel assignments, we also indicated in Table 3.8 their score over the other performance metrics.



(a) Original

(b) Maximize global satisfaction rate

(c) Maximize Jain's fairness index

(d) Maximize proportional fairness

Figure 3.8: Different channel allocations for a randomly-generated 12-node network.

Performance metric	GSR	J	NJ	PF
Fig. 3.8b Maximize GSR	96%	0.725	0.983	-1.27
Fig. 3.8c Maximize J	73%	0.796	0.955	-3.30
Fig. 3.8d Maximize PF	95%	0.735	0.987	-1.08

Table 3.8: Evaluation of the proposed channel allocations.

When maximizing the global satisfaction rate, GSR, the retained solution maximizes the overall throughput obtained in the network and leads a GSR of 96%. Interestingly, we notice in Fig. 3.8b that all the high-demanding nodes (whose input rate is over 0.5) do not share the channel with any other node, thereby enabling them to obtain the highest possible throughput. On the other hand, when maximizing Jain's index, we observe in

Fig. 3.8c that almost all nodes have a neighbor with whom they share the medium. In fact, with the exception of the pair of nodes 6 and 2, all the other pairs involves two nodes belonging to the same class (be it low-demanding or high-demanding nodes). As a consequence high-demanding nodes get lower output rates, as they have to share the medium with other high-demanding nodes. The optimal solution for the Jain's index increases its score from 0.725 to 0.796, at the expense of over 20% loss in the GSR. The last optimal solution maximizes the proportional fairness, PF. In Fig. 3.8d we observe that the only difference between the PF solution and the GSR solution lies in the selected channel of node 8. This similarity can be understood by the fact that proportional fairness, unlike Jain's fairness index, takes into consideration not only the output rate of each node but also its input rate. Overall, in this example, the optimal solution for Proportional fairness coincides with the optimal solution for Normalized Jain's index, and appears as a good trade-off between maximizing throughput or fairness, as it offers both a GSR value and J value that are remarkably close to their optimal values.

3.8 Conclusions

Summary of Contributions

We have presented a modeling framework for IEEE 802.11-based WLANs. Our approach accounts for WLANs composed of multiple APs assuming their conflict graph is known. Our framework assumes any levels of load in the APs, arbitrary sizes for frames and arbitrary transmission rates for links, as well as recent amendments to IEEE 802.11 such as 802.11n. The proposed solution revolves around a Divide-and-Conquer approach to split the initial problem into many sub-problems, each being of much lower complexity.

We studied several hundreds of examples to assess the accuracy of our modeling framework comparing its results with those delivered by the *ns*-3 simulator. We considered several network topologies with the number of APs ranging from 3 to 10, different amendments of IEEE 802.11, various levels of the load on each AP, different transmission rates on the APs, as well as examples where APs implement the aggregation feature so that multiples frames are concatenated into a single large frame before being transmitted. Overall, in our examples, our model was able to forecast with a reasonable degree of precision (typically within 10% of relative errors) the mean throughput attained by each AP of the network.

Encountered Difficulties

Our progress in the development of our approach has been hampered by a peculiar difficulty regarding the use of a discrete-event simulator. To validate a theoretical model for IEEE 802.11-based networks, one commonly relies on performance comparisons with a discrete-event simulator such as *ns*-2 [C24], *ns*-3 [C25], OMNeT++ [C26], and NetSim [C23]. Although discrete-event simulators cannot capture the whole complexity of a real-life IEEE 802.11-based network, they appear as a good alternative to real-life experiments. Simulations are easier to handle, less expensive, reproducible and often regarded as accurate.

Our initial choice was to use the simulator *ns*-2 because its use seemed quicker than that of the other simulators. As we made progress in our research, we realized that *ns*-2 had some limitations, in particular relating to newer amendments of IEEE 802.11 and the properties of the physical and MAC layers. In particular, *ns*-2 does not natively

support IEEE 802.11g or n. More importantly, it poorly implements phenomena like the capture effect that, in reality, enables a node receiving two signals simultaneously to often demodulate successfully the stronger of the two signals. In *ns*-2, this feature is not realistically implemented, and the node will often undergo a collision for each transmission.

In order to assess the potential impact of this omission, we compared the results delivered by *ns*-2 with those of *ns*-3, which implements the capture effect. Our results show that, in many scenarios, neglecting the capture effect may significantly bias the performance of a network. Therefore, we made the decision to abandon *ns*-2 and to move to *ns*-3. To give an idea of how wide the gap may be, we represent in Figure 3.9 the achieved throughput by a node in a simple three-node topology as predicted by *ns*-2 and *ns*-3, respectively. The difference may be close to zero when the node has a moderate demand for load but it may grow up to 20% for higher level of loads.

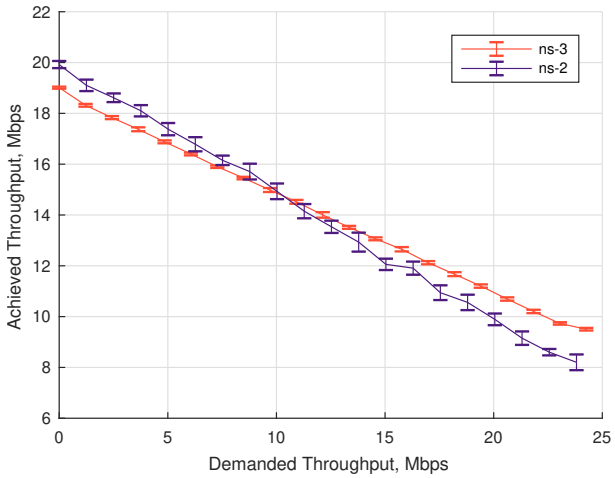


Figure 3.9: Potential discrepancy between *ns*-2 and *ns*-3 for the achieved throughput of a node in a three-node topology.

Another faced obstacle is that the performance of an IEEE 802.11-based network are very sensitive to its underlying conflict graph and to its node activities. Thus, even two seemingly similar networks may lead to significantly different behavior. This sensitivity has hardened the step of validating our model accuracy as we had to go through many examples to gain a statistically meaningful metrics of accuracy.

Strengths and Limitations

I believe that a key strength of our model is found in its simplicity. In fact, not only does its "Divide-and-Conquer" approach facilitates its scalability with the number of nodes in a WLAN, but it also makes it a good candidate for parallelization as any sub-problems are independent of each others. Another of its strengths is its readability (traceability) that provides insights on the many relationships and dependencies that exist among the various elements composing an IEEE 802.11-based WLAN.

In my opinion, the main limitation to our model approach is that we were not able to clearly evaluate its computational complexity. While the number of state in each Markov chain is probably a relevant metric to its complexity, we were not able to derive a closed-form expression of it. The derivation of such an expression does not appear easy as the number of states is deeply tied to the particular conflict graph underlying the considered

wireless network. Therefore, instead of a theoretical complexity analysis, we relied on a much less convincing empirical study.

Possible Extensions

I envision mostly two extensions for our work. First, to better account for frames with different lengths and links with different transmission rates, we could replace the use of discrete-time Markov chains by continuous-time Markov chains. Continuous-time Markov chains would offer a natural way of having different time transmission at each node.

Second, we validated our model against the discrete-event simulator *ns*-3. Although this latter aims at capturing most effects taking place in a real WLAN and is often described as a realistic simulator, an experimental validation using real-life measurements is still a missing piece to our work. However, the lack of time and adequate equipment and the complexity have postponed our efforts in this direction. I discuss these aspects in further details in Chapter 5.

Potential Impact

With the increasing size of WLANs, some form of coordination among their APs is desired to improve the WLAN overall behavior. A case in point is the issue of allocating radio channel to each AP. The search for the optimal solution is NP-hard and existing tools often rely on iterative heuristics. They start from a given allocation scheme, and then iteratively improve it (with regard to some network performance parameters such as the overall throughput, the fairness, the channel utilization) until convergence is found. In this regard, our model can help the heuristics by providing quick and fair estimates of the current setting of the WLAN at each iteration (rather than relying on long simulations).

Learned Experiences

Because of their seemingly complexity, simulators may appear as trustworthy at first glance. However, as discussed above in this section (see Encountered difficulties), this complexity does not prove, by any means, that the simulator delivers realistic and accurate results for a given scenario. I could sum up my experience in this regard as *Don't trust a simulator blindly unless you know all its details!*

References for Chapter 3

[C1] T. Abreu, B. Baynat, T. Begin, and I. Guérin Lassous. Hierarchical modeling of IEEE 802.11 multi-hop wireless networks. In *ACM MSWIM*, 2013.

[C2] T. Abreu, B. Baynat, T. Begin, I. Guérin Lassous, and H.-N. Nguyen. Modeling of IEEE 802.11 multi-hop wireless chains with hidden nodes. In *ACM MSWIM*, 2014.

[C3] T. Abreu, N. Nguyen, T. Begin, I. Guérin Lassous, and B. Baynat. Substitution Networks: Performance Collapse due to Overhead in Communication Times. In *AdhocNets*, 2012.

[C4] T. Begin, B. Baynat, I. Guérin Lassous, and T. Abreu. Performance analysis of multi-hop flows in IEEE 802.11 networks: A flexible and accurate modeling framework. *Performance Evaluation*, 96:12–32, 2016.

[C5] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, 2000.

[C6] S. Biaz and S. Wu. Rate adaptation algorithms for IEEE 802.11 networks: A survey and comparison. In *IEEE ISCC*, 2008.

[C7] R. Boorstyn, A. Kershenbaum, B. Maglaris, and V. Sahin. Throughput analysis in multihop CSMA packet radio networks. *IEEE Transactions on Communications*, 35(3):267–274, 1987.

[C8] F. Cali, M. Conti, and E. Gregori. IEEE 802.11 wireless LAN: Capacity analysis and protocol enhancement. In *IEEE INFOCOM*, 1998.

[C9] S. Chieochan, E. Hossain, and J. Diamond. Channel assignment schemes for infrastructure-based 802.11 WLANs: A survey. *IEEE Communications Surveys & Tutorials*, 12(1):124–136, 2010.

[C10] *Cisco Visual Netwotking Index*. `https://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html`.

[C11] M. Durvy, O. Dousse, and P. Thiran. Self-organization properties of CSMA/CA systems and their consequences on fairness. *IEEE Transactions on Information Theory*, 55(3):931–943, 2009.

[C12] E. Felemban and E. Ekici. Single hop IEEE 802.11 DCF analysis revisited: Accurate modeling of channel access delay and throughput for saturated and unsaturated traffic cases. *IEEE Transactions on Wireless Communications*, 10(10):3256–3266, 2011.

[C13] M. Garetto, T. Salonidis, and E. W. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. In *IEEE INFOCOM*, 2006.

[C14] N. Gupta and C. Rai. New analytical model for non-saturation throughput analysis of IEEE 802.11 DCF. In *CNC*, 2014.

[C15] M. Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[C16] IEEE Computer Society LAN MAN Standards Committee and others. Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. *IEEE Standard 802.11-2016*, 2016.

[C17] R. Jain, D. M. Chiu, and W. R. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer system. *Digital Equipment Corporation*, 1984.

[C18] L. Jiang and J. Walrand. A distributed CSMA algorithm for throughput and utility maximization in wireless networks. *IEEE/ACM Transactions on Networking*, 18(3):960–972, 2010.

[C19] C. Kai and S. Zhang. Throughput analysis of CSMA wireless networks with finite offered-load. In *IEEE ICC*, 2013.

[C20] K. Kosek-Szott. A comprehensive analysis of IEEE 802.11 DCF heterogeneous traffic sources. *Ad Hoc Networks*, 16:165–181, 2014.

[C21] R. Laufer and L. Kleinrock. The capacity of wireless CSMA/CA networks. *IEEE transactions on Networking*, 2015.

[C22] B. Nardelli and E. W. Knightly. Closed-form throughput expressions for CSMA networks with collisions and hidden terminals. In *IEEE INFOCOM*, 2012.

[C23] *NetSim - Network Simulator & Emulator*. https://www.tetcos.com/.

[C24] *The Network Simulator ns-2*. http://www.isi.edu/nsnam/ns/.

[C25] *The Network Simulator ns-3*. https://www.nsnam.org/.

[C26] *OMNeT++ Discrete Event Simulator*. https://www.omnetpp.org/.

[C27] J. Padhye, S. Agarwal, V. N. Padmanabhan, L. Qiu, A. Rao, and B. Zill. Estimation of link interference in static multi-hop wireless networks. In *ACM SIGCOMM*, 2005.

[C28] C. Peng, H. Zheng, and B. Y. Zhao. Utilization and fairness in spectrum assignment for opportunistic spectrum access. *Mobile Networks and Applications*, 11(4):555–576, 2006.

[C29] T. Razafindralambo, T. Begin, M. Dias De Amorim, I. Guérin Lassous, N. Mitton, and D. Simplot-Ryl. Promoting quality of service in substitution networks with controlled mobility. In *AdHocNow*, 2011.

[C30] Z. Shi, C. Beard, and K. Mitchell. Analytical models for understanding space, back-off, and flow correlation in CSMA wireless networks. *Wireless Networks*, 19(3):393–409, 2013.

[C31] M. Stojanova, T. Begin, and A. Busson. Conflict graph-based Markovian model to estimate throughput in unsaturated IEEE 802.11 networks. In *IEEE WiOpt*, 2017.

[C32] M. Stojanova, T. Begin, and A. Busson. Conflict graph-based model for IEEE 802.11 networks: A Divide-and-Conquer approach. *Performance Evaluation*, 2018.

[C33] L. Verma, M. Fakharzadeh, and S. Choi. WiFi on steroids: 802.11 ac and 802.11 ad. *IEEE Wireless Communications*, 20(6):30–35, 2013.

[C34] X. Wang and K. Kar. Throughput modelling and fairness issues in CSMA/CA based ad-hoc networks. In *IEEE INFOCOM*, 2005.

Chapter 4

Reduced State Description

Contents

4.1 Research Context **64**
4.2 Outline **65**
4.3 Motivation and Related Work **65**
4.4 Model with Infinite Buffer **67**
4.5 Model with Finite Buffer and State Dependencies **69**
4.6 Numerical results **71**
4.6.1 Accuracy for the Mean Number in the queue 72
4.6.2 Loss Probability with finite buffers 73
4.6.3 Wait Probability 74
4.6.4 Speed of Asymptotic Convergence for queues with unrestricted buffer . 75
4.6.5 Speed of Convergence of the Fixed-Point Iterative Solution 75
4.6.6 Model with State Dependencies 75
4.6.7 Computational Complexity 76
4.6.8 Example of comparison with simple approximations 77
4.7 Conclusions **78**
4.8 Appendix **82**

Abstract

Ph/Ph/c and *Ph/Ph/c/N* queues can be viewed as a common model of multi-server facilities. We propose a simple approximate solution for the equilibrium probabilities in such queues based on a reduced state description in order to circumvent the well-known and dreaded combinatorial growth of the number of states inherent in the classical state description. The number of equations to solve in our approach increases linearly with the number of servers and phases in the service time distribution. A simple fixed-point iteration is used to solve these equations. Our approach applies both to open models with unrestricted buffer size and to queues with finite-size buffers.

The results of a large number of empirical studies indicate that the overall accuracy of the proposed approximation appears very good. For instance, the median relative error for the mean number in the queue over thousands of examples is below 0.1% and the relative error exceeds 5% in less than 1.5% of cases explored. The accuracy of the proposed approximation becomes particularly good for systems with more than 8 servers, and tends to become excellent as the number of servers increases.

4.1 Research Context

For years I established a regular scientific collaboration with Pr. Alexandre Brandwajn (UCSC, USA). I visit his lab every year for a couple of weeks, and vice versa. Together we have explored several research topics such as the influence of distributional properties of inter-arrival and service times on the performance of queueing systems, the development of a high-level modeling approach, as well as the design of simple and computationally efficient solutions for queueing systems.

Our joint-interest in simple and computationally efficient solutions for queueing systems started in 2009, when, for the sake of another work (high-level modeling), we were in need of a fast and scalable solution to queues with multiple servers and non-exponential service times. Unfortunately, with the exception of very particular cases, it is a complex problem and existing solutions tend to be either inaccurate, or complex and unscalable with the number of servers (these aspects are discussed in detail in Section 4.3). Thus, we developed a new solution for queues with multiple servers where the arrival process can be regarded as general (namely, an arbitrary Coxian distribution [D14]) whereas the service process at each server is restricted to a specific subclass of distribution (namely, a two-stage Coxian distribution). Although this solution filled our need, it never got fully published per se. Nonetheless, it stirred our interest in this domain, and in the following years, we proposed solutions to various queueing systems including single server queues with general service and inter-arrival times [D9, D10] as well as an accuracy study for several existing approximations for multi-server queues [D3]. However, it was not until 2014 that, following Alexandre Brandwajn proposal, we managed to develop an original approach that differs from the state-of-the-art solutions by its simplicity and scalability. In essence, our solution makes a clever use of the assumption of dealing with homogeneous servers (i.e., statistically identical). We applied this idea to two classes of queues with multiple servers and general service times. First, in the case of Poisson arrivals [D11], and then with general inter-arrivals times [D12]. In this chapter, I present the latter since it includes the particular case of Poisson arrivals.

4.2 Outline

This chapter is organized as follows. The next section discusses the motivation for this approximation and the related work. Section 4.4 is devoted to the approximate solution of the $Ph/Ph/c$ queue with an infinite buffer. In Section 4.5 we consider a queue with a finite buffer and state-dependent distributions of interarrival times and service times. Section 4.6 presents numerical results to illustrate the accuracy of the proposed approximation. Finally, Section 4.7 concludes this chapter.

4.3 Motivation and Related Work

A number of areas of computer applications and systems offer examples of multi-servers facilities. For instance, many-core CPUs with 32 or more cores are around the corner [D8]. Parallel Access Volumes in mainframe storage [D23] provide a potentially large number of "exposures" for simultaneous access to information. Call centers with hundreds of agents [D16] are an element of everyday life. The number of Virtual Machines hosted on Physical Machines can easily exceed 16. In the area of fiber optical cables, WDM multiplexing allows over a hundred simultaneous signals on a single fiber.

Such systems can be naturally represented as multi-server queues in which requests arrive, queue for service if all servers are found busy, and eventually leave the system after receiving service from one of the servers. Unfortunately, if one realistically assumes general distributions of times between request arrivals and general service times, the resulting $G/G/c$ queueing model does not possess a known analytical solution except in some special cases [D19, D15, D4, D1]. Additionally, under higher loads, realistic models must account for finite buffer space (queueing room) which may prevent requests from joining the queue when the buffer is full.

A common approach is then to replace the "general" distributions by so-called "phase-type" distributions [D27, D6, D24] as any distribution can be approximated arbitrarily closely by a phase-type distribution (e.g., [D27]). This has the distinct advantage of leading to a system that, in steady-state, can be described by familiar balance equations. Generally speaking, these balance equations can be obtained using one of two possible state descriptions involving the current number of requests in the system, the current phase of the arrival process and a vector to represent the state of the servers. The first one uses the vector of the current number of servers in each phase of the service process. In the second possible description, the vector is that of the current phases for each server (note that the servers are assumed to be homogenous but they are not synchronized). This latter state description is generally less thrifty than the first one and rarely, if ever, used. Both descriptions exhibit combinatorial growth as the number of phases and the number of servers grow. We discuss this point further in Section 4.6.7. In queueing terms, the $G/G/c$ queue is replaced by the $Ph/Ph/c$ queue. The latter can be solved numerically using a direct iteration [D26] or matrix geometric methods [D25, D22, D5]. This approach works great as long as the number of servers and/or phases in the arrival and especially service process is not too large. However, as mentioned above, the number of servers in many realistic examples varies from several tens to many hundreds, and the traditional phase-type approach is known to suffer from the "dimensionality curse" in that the number of states (and, hence, equations to solve in the linear system) grows combinatorially as the number of servers and /or phases increases. This precludes the direct use of this approach in many interesting and important areas.

In the area of approximate solutions to such systems, several authors attempt to summarize the properties of general distributions in $G/G/c$ queues by their first 2 (rarely, 3) moments [D18]. Although the resulting approximations are usually simple to implement and their execution is fast, unfortunately, they fail to account for the intrinsic dependence of the $G/G/c$ queue on higher-order properties of the distributions involved [D17, D3]. We provide an example in Section 4.6.8. Fluid queues represent another avenue for approximation based on the fact that, as the number of requests in a queueing system tends to infinity, one can consider the flow of discrete requests as a continuous flow and hence apply fluid mechanics equations to describe the system. These methods have been applied, for example, to represent call centers [D21, D16] and the $G/G/c/N$ queue [D29]. By their principle, these approximations appear best suited for the study of limiting processing capacities of such queues.

In the particular case when the arrivals can be treated as generated by a Poisson or quasi-Poisson source, there has been recent progress in obtaining computationally manageable approximations applicable to systems with hundreds of servers. Khazaei et al. [D20] propose to use an adaptation of the embedded Markov Chain method in the case of a pure Poisson arrival process. They show the good accuracy of their numerical results for service time coefficients of variation not exceeding 1.4. The finite buffer size in their numerical results is relatively small and kept at less than half the number of servers. Their approach does not seem easy to apply to systems with state dependencies or more general arrival processes. In this regard, our contribution with Prof. Brandwajn [D11] introduces an approximation based on a reduced state description and demonstrate the accuracy of their approach for much larger range of coefficients of variation of the service time distribution (up to 7) and buffer sizes.

Clearly, in many situations the arrival process cannot be viewed as Poisson or quasi-Poisson. Therefore, in this chapter we present an approximate solution based on the reduced state description that deals with systems having phase-type distributions for the time between arrivals. The proposed approximation works both for open queues (i.e., queues with unlimited buffer size) and queues with finite buffer. While no human-made system possesses a truly unrestricted buffer size, such models are of practical interest when the physical buffer size is relatively large and the server utilization is not too close to saturation. In such cases the use of an open model may result in computational saving over a finite-buffer-size model. Interestingly, in open queues, our approximation happens to tend to the correct asymptotic rates of request arrivals and service rates as the number of requests tends to infinity.

To avoid arbitrary truncation in the open model, we transform the balance equations for the reduced state into equations for the conditional probabilities of the state of the arrival process and the reduced state of the service given the current number of requests. We then exploit the convergence of such conditional probabilities to their asymptotic values so as to enumerate the states only up to the practical asymptotic convergence point.

The use of conditional probabilities partitions the state space into independently normalized subspaces, which may contribute to numerical stability, and we employ them also in the case of finite buffers. We propose a simple fixed-point iteration to solve the conditional probability equations. Although we do not have a theoretical proof of convergence to a unique solution, the proposed iteration has never failed in the large number of cases explored.

4.4 Open Model (Infinite Buffer) and its Solution

We start by considering a classical *Ph/Ph/c* queue with an infinite buffer [D18]. We assume that the c servers are homogeneous, i.e., statistically identical, but not synchronized. As shown in Figure 4.1, the distribution of the times between arrivals comprises a exponential phases and the service time distribution has b exponential phases. We denote by σ_i the probability that service starts in phase i, by μ_i the completion rate for phase i $(i = 1, \ldots, b)$ of the service process, and by $\hat{q}_i$ the probability that the service process completes after phase i. We denote by τ_j, λ_j and $\hat{r}_j$ the corresponding quantities for phase j $(j = 1, \ldots, a)$ of the arrival process.

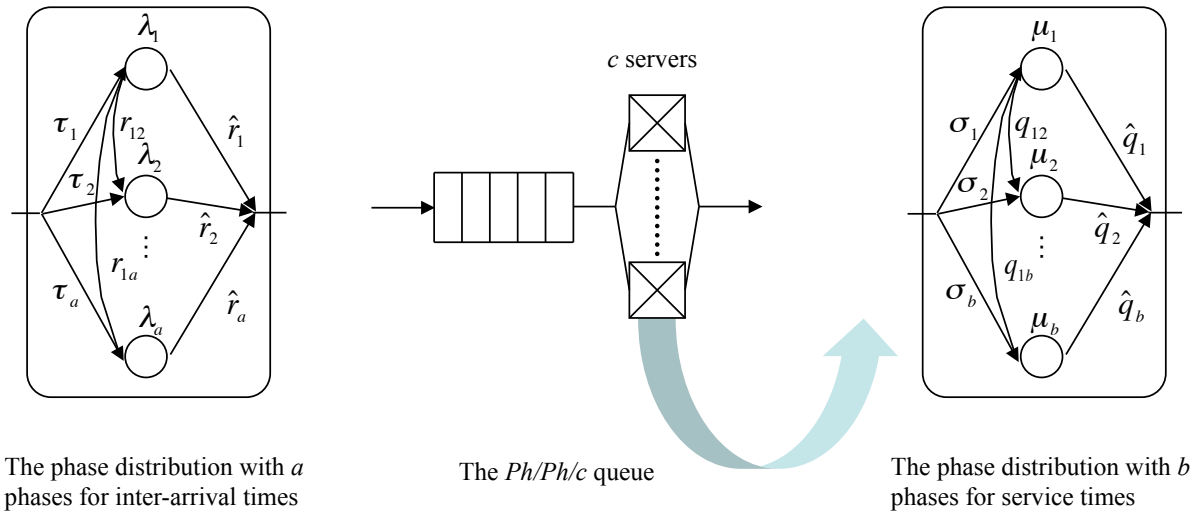


Figure 4.1: *Ph/Ph/c* queue with unrestricted buffer.

The classical approach to derive the steady-state probability of the number of requests (customers) in such a system is to consider a state description that includes the current number of requests in the system (n), the current phase of the arrival process (j), and the vector of the current number of requests in each phase of the service process $(\overrightarrow{m} = m_1, \ldots, m_b)$. It is clear that (for each value of n) such a full state description results in a combinatorial explosion of the number of balance equations one has to solve as the number of servers and service phases increases, compounded by the number of arrival phases (see Section 4.6.7 for more details).

As we are focusing on systems with large numbers of servers, to escape this issue, we use a reduced state description comprising the current number of users (n), the current phase of the arrival process (j) and the current phase of the service process for an arbitrarily selected server (i). Since for $n < c$ the selected server may be idle, we use the value $i = 0$ to denote its idle state. Let $p(n, j, i)$ be the corresponding steady-state probability where $n = 0, 1, \ldots$, $j = 1, \ldots, a$, $i = 0, \ldots, b$. Denote by $p(n)$ the steady-state probability that there are n requests in the system, and by $p(j, i|n)$ the conditional probability of the current phase of the arrival process and the current service phase for the selected server given the current number of requests in the system. For $n = 0$, all servers are idle and only the arrival phase $j = 1, \ldots, a$ is of significance.

Using this reduced state description, it is a straightforward matter to obtain the balance equations for the probabilities $p(n, j, i)$. These equations involve the parameters of the arrival process and of the service distribution, as well as the conditional rate of completions

of requests by servers other than the chosen server given the retained state description, which we denote by $\nu(n, j, i)$. Note that these balance equations, whose form is illustrated by Eq. (4.12) in Appendix, are exact in the sense that their solution would produce the exact steady-state probabilities $p(n, j, i)$ if we knew the exact values for $\nu(n, j, i)$. This latter quantity, however, is not known explicitly and will be evaluated only approximately. Therein lies the approximation of our proposed solution.

The conditional rate of request completions by the **selected server** given the current number in the system can be expressed as

$$\psi(n) = \sum_{j=1}^{a} \sum_{i=0}^{b} p(j, i|n) \mu_i \hat{q}_i. \tag{4.1}$$

Since the c severs are homogeneous, the overall conditional rate of request completions given n is simply

$$u(n) = c\psi(n). \tag{4.2}$$

Let $m = \min(n, c)$ be the number of busy servers with n customers in the system. To obtain an approximation for the unknown conditional rate of completions by other servers $\nu(n, j, i)$ we assume that the latter depends primarily on the current number of requests in the system n and not on the current phase of the arrival process or service at the selected server, when it is active, so that we have

$$\nu(n, j, i) \simeq \frac{m-1}{m} u(n) = (c - \frac{c}{m})\psi(n) \quad \text{for} \quad j = 1, \ldots, a \quad \text{and} \quad i = 1, \ldots, b. \tag{4.3}$$

For $n < c$ and $i = 0$, i.e., when the selected server is not active, we use simply

$$\nu(n, j, 0) \simeq u(n) = c\psi(n). \tag{4.4}$$

This is the main approximation in our approach. In essence, we neglect the dependence on the current state of the arrival and service process in $\nu(n, j, i)$ so that the latter becomes a function only of the current number of requests in the system n. Intuitively, the neglected state information matters most when the number of servers is small. With larger numbers of servers, the knowledge of the current state of just one out of many servers or of the current phase of the arrival process conveys little information about the state of other servers. This is why we expect the approximation in formula (4.3) to improve as the number of servers increases.

The conditional rate of request arrivals given the current number in the system can be expressed as

$$w(n) = \sum_{j=1}^{a} \sum_{i=0}^{b} p(j, i|n) \lambda_j \hat{r}_j. \tag{4.5}$$

Hence, the steady-state probability that there are n requests in the system can be obtained in terms of $w(n)$ and $u(n)$ as

$$p(n) = \frac{1}{G} \prod_{k=1}^{n} \frac{w(k-1)}{u(k)}, \quad n = 0, 1, \ldots, \tag{4.6}$$

where G is a normalizing constant.

Using the fact that $p(n,j,i) = p(j,i|n)p(n)$ together with formula (4.6) in the balance equations for $p(n,j,i)$ we can transform the latter into equations for the conditional probabilities $p(j,i|n)$. We provide the corresponding equation for $n > c$ (see Eq. (4.13) in Appendix). Because the quantities $w(n)$, $u(n)$ and $\nu(n,j,i)$ are expressed in terms of system parameters and of the probabilities $p(j,i|n)$, we obtain a self-contained system of equations for the conditional probabilities $p(j,i|n)$. These probabilities must be normalized for each value of n so that we must have

$$\sum_{j=1}^{a}\sum_{i=0}^{b} p(j,i|n) = 1, \quad n = 0,1,\ldots \tag{4.7}$$

With an unrestricted buffer size, the resulting system of equations is in theory infinite since there is no upper bound on the value of the number of requests n. Assuming the system is ergodic, the equations for $p(j,i|n)$ must tend to a limit as n tends to infinity $\lim_{n\to\infty} p(j,i|n) = \widetilde{p}(j,i)$ and similarly for the conditional rate of arrivals and completions $\lim_{n\to\infty} w(n) = \widetilde{w}$ and $\lim_{n\to\infty} u(n) = \widetilde{u}$ (as well as $\lim_{n\to\infty} \nu(n,j,i) = \widetilde{\nu}(j,i)$). This is in agreement with the well-known result that the probability $p(n)$ is asymptotically geometric [D28]. We assume and our empirical results confirm that this is also the case when using the approximation given by (4.3) and (4.4). Denote by $\widetilde{n}$ the value of n for which $p(j,i|n)$ and hence also $w(n)$ and $u(n)$ become sufficiently close to their respective asymptotic values, i.e., $|w(n) - \widetilde{w}| < \delta$ and $|u(n) - \widetilde{u}| < \delta$ where $\delta > 0$.

Thus, for $n = 0,\ldots,\widetilde{n}-1$ we use the equations for $p(j,i|n)$ and for $n \geq \widetilde{n}$ we use the corresponding equations for the asymptotic conditional probabilities $\widetilde{p}(j,i)$, i.e., the limit for $n \to \infty$ of the equations for $p(j,i|n)$. Both sets of equations can be readily solved using a simple fixed-point iteration (see [D12] for more details). Hence, we obtain the values of $w(n)$ and $u(n)$ for $n = 0,\ldots,\widetilde{n}-1$, and the asymptotic values $\widetilde{w}$ and $\widetilde{u}$, which allows us to compute the steady-state probability $p(n)$ using formula (4.6) as

$$p(n) = \begin{cases} \frac{1}{G}\prod_{k=1}^{n} \frac{w(k-1)}{u(k-1)}, & n < \widetilde{n} \\ \frac{1}{G}\big(\prod_{k=1}^{\widetilde{n}} \frac{w(k-1)}{u(k-1)}\big)\cdot\big(\frac{\widetilde{w}}{\widetilde{u}}\big)^{n-\widetilde{n}}, & n \geq \widetilde{n} \end{cases} \tag{4.8}$$

Clearly, for this approach to be of practical interest, such asymptotic convergence must happen for values of $\widetilde{n}$ that are not excessively large. Fortunately, as shown by our numerical results, these values tend to be reasonable. Note that the value of $\widetilde{n}$ is determined dynamically in our method. In a practical implementation, we dimension the data structures to hold the quantities $p(j,i|n)$, $w(n)$ and $u(n)$ according to the results presented in Section 4.6.4, and we monitor dynamically the iteration process for asymptotic convergence so that the enumeration of values of the number of requests in the system n can be stopped. Note also that our approximation produces the correct values for the asymptotic arrival and service rates $\widetilde{w}$ and $\widetilde{u}$. The next section is devoted to a model with finite buffer space and state dependencies.

Table 4.1 summarizes the principal notation used in this chapter.

4.5 Model with Finite Buffer and State Dependencies

In this section we consider a model with a finite buffer and possible state dependencies. We denote by N the maximum number of requests (including those in service) that can

Table 4.1: Principal notation.

Symbol	Description
c	Number of servers
N	Buffer size, i.e., maximum of requests in the system (queued and in service)
n	Total current number of requests in the system
b	Number of phases for the service time distribution
σ_i	Probability that service of a request starts in phase i, $i = 1, \ldots, b$
μ_i	Completion rate for phase i of service process, $i = 1, \ldots, b$
q_{il}	Probability that service process continues in phase l upon completion of phase i, $i, l = 1, \ldots, b$
$\hat{q}_i$	Probability that service process ends (request departs the system) upon completion of phase i, $i = 1, \ldots, b$
a	Number of phases for the inter-arrivals distribution
τ_j	Probability that arrival of a request starts in phase j, $j = 1, \ldots, a$
λ_j	Completion rate for phase j of arrival process, $j = 1, \ldots, a$
r_{jk}	Probability that arrival process continues in phase k upon completion of phase j, $j, k = 1, \ldots, a$
$\hat{r}_j$	Probability that arrival process ends (request enters the system) upon completion of phase j, $j = 1, \ldots, a$
$p(n, j, i)$	Probability that there are n requests in the system, the current phase of the arrival process is j, and the current phase of the service process at the selected server is i
$\psi(n)$	Conditional rate of request completions by the selected server given the current number in the system is n
$u(n)$	Overall departure rate from the set of c servers given that the current number in the system is n
$\nu(n, j, i)$	Conditional rate of completions of requests by servers other than the chosen server given that the current number in the system is n, the current phase of the arrival process is j, and the current phase of the service process at the selected server is i
$w(n)$	Conditional rate of request arrivals given the current number in the system is n
$p(n)$	Marginal probability that there are n requests in the system

be present in the system at any time. As before, we assume that the distribution of the times between arrivals comprises a exponential phases and the service time distribution has b exponential phases. The completion rates (intensities) of these exponential phases, as well as the probabilities of initial phase selection, moving from phase to phase and of completing after a phase may depend on the current number of users in the system, $n \leq N$. Thus, we let $\mu_i(n)$ be the completion rate for phase i ($i = 1, \ldots, b$) of the service process, and $\hat{q}_i(n)$ the probability that the service process completes after phase i. We also let $\lambda_j(n)$ and $\hat{r}_j(n)$ be the corresponding quantities for phase j ($j = 1, \ldots, a$) of the arrival process.

As before, we use a reduced state description $p(n, j, i)$ where n is the current number of users in the system, j is the current phase of the arrival process and i is the current phase of the service process for an arbitrarily selected server. We denote by $p(n)$ the steady-state probability that there are n requests in the system, and by $p(j, i|n)$ the conditional probability for the current phase of the arrival process and of the service process at the selected server given n.

With state-dependent phase intensities and transition probabilities, the conditional rate of request completions by the selected server given the current number in the system

becomes

$$\psi(n) = \sum_{j=1}^{a} \sum_{i=0}^{b} p(j,i|n)\mu_i(n)\hat{q}_i(n). \tag{4.9}$$

Similarly, the conditional rate of request arrivals given n becomes

$$w(n) = \sum_{j=1}^{a} \sum_{i=0}^{b} p(j,i|n)\lambda_j(n)\hat{r}_j(n). \tag{4.10}$$

As before, the overall conditional rate of request completions given n is $u(n) = c\psi(n)$ and we use approximation (4.3) and (4.4) for the unknown conditional rate of completions by other servers. The steady-state probability that there are n requests in the system can be computed from formula (4.6) once we have obtained the conditional rates of arrivals and of completions $w(n)$ and $u(n)$. To compute these two quantities, just like previously, we use the identity $p(n,j,i) = p(j,i|n)p(n)$ and (4.6) in the balance equations for $p(n,j,i)$ to transform the latter into equations for the conditional probabilities $p(j,i|n)$.

However, unlike in the case of an open queue (unrestricted buffer space), the number of values of n to consider is finite (and thus there is no asymptotic convergence), and we have a particular equation for $n = N$. There are several ways in which a physical system may behave when the buffer space is full, e.g. the source of arrivals may become blocked until there is space in the buffer or the source may continue to generate requests which are simply lost. We consider the latter case in this chapter although our method can be readily adapted to handle the case of blocked arrivals as well. The corresponding particular equation for $n = N$ are given in [D12].

With this assumption on system behavior when the buffer is full, it is important in some applications to determine the probability that an arriving request will be lost. This loss probability can be expressed as

$$p_{loss} = \frac{w(N)p(N)}{\sum_{n=0}^{N} w(n)p(n)} \tag{4.11}$$

and thus easily calculated once we have the conditional rates of arrivals $w(n)$ and the steady-state probabilities $p(n)$.

The next section is devoted to the accuracy and other aspects of the numerical behavior of the proposed approximation.

4.6 Numerical results

We study the accuracy of the proposed approach for a spectrum of values of system parameters. We present results for the following numbers of servers: c=8,16,32,64,128 and $c = 256$. (For numbers of servers below 8, exact numerical solutions using the full state description [D25, D22, D5] are manageable and there is little need for approximations.) The offered load per server, i.e., the ratio of the submitted traffic to the number of servers, varies in steps of 0.1 from 0.5 to 0.9 for queues with unrestricted buffer, and from 0.5 to 1.5 for queues with finite buffer. We consider 5 buffer sizes for the latter, expressed in relation to the number of servers as $N = 1.5c$, $2c$, $2.5c$, $3c$ and $N = 4c$ (recall that N is the maximum number of requests queued and in service that can be present in the

system). We use distributions of the times between arrivals and of service times with four phases and coefficients of variation ranging from 0.5 to 4 in steps of 0.5. We utilize the method proposed by Bobbio et al. [D6] to generate the four-phase distributions with specified coefficients of variation. The behavior of the reduced state approximation with exponential inter-arrival times ($M/Ph/c$ queue) has been studied in prior work [D11] and the proposed approach produces exact results in the case when the service time distribution is exponential. Therefore, we skip the value 1 for the coefficient of variation of both inter-arrival and service times.

Overall, we explored 1,470 examples for the open model, and 16,170 examples with finite buffers. The performance measures considered include the mean number of requests in the system, the wait probability (probability that an arriving request finds all servers busy), as well as the loss probability in the case of a finite buffer. We use the absolute value of the relative error (expressed as a percentage) between the exact and approximate values as measure of approximation accuracy. The "exact" values come from a numerical solution of the full balance equations for $c = 8$ and $c = 16$. For larger number of servers, we use discrete-event simulation with 15 independent replications of 50,000,000 completions each. Our choice of these simulation parameters, while somewhat arbitrary, is rooted in the idea that with independent replications it may be worthwhile to have larger values for the number of completions per replication to minimize "warm-up" effects. With the values chosen, the estimated confidence intervals at 95% confidence level are generally so small that we use only the mid-point value. Given the difficulty of estimating small quantities in simulations, we only consider relative errors for wait and loss probabilities when the "exact" values exceed 0.01. To summarize the accuracy of our approach, we consider the distribution of relative errors, as well as the median error values.

We now describe the numerical results obtained for the quantities considered.

4.6.1 Accuracy for the Mean Number in the queue

We start by considering the accuracy of the method for the mean number of requests in the system. Figure 4.2 shows the distribution of errors and the median error as a function of the number of servers for queues with unrestricted buffer. We observe that the percentage of cases in which the error exceeds 5% falls rapidly from 46% with 8 servers to 6% with 64 servers and all the way to 0% with 256 servers. At the same time, the percentage of observed errors exceeding 15% decreases from 25% with 8 servers down to 0.4% with 64 servers and 0% for 128 or more servers. It is worthwhile noting that errors exceeding 15% observed with 8 servers tend to occur for larger values of the coefficient of variation of the service time and higher server utilizations. Thus, as expected on intuitive grounds, the accuracy of the proposed method increases rapidly as the number of servers grows. The median error decreases from 4% with 8 servers to less than 0.4 % with 16 or more servers.

Figure 4.3 summarizes the results obtained for the mean number of requests in the system for queues with finite buffer. In the large set of examples explored (over 16,000) we found virtually no cases in which the relative error exceeds 10%. We observe that the infrequent larger errors tend to occur for smaller numbers of servers and when the coefficient of variation of the time between arrivals is small and the coefficient of variation of the service time is large. Overall, in some 99% of cases, the relative error remains below 5%. The median error remains below 0.1% throughout all cases explored.

It is interesting to examine the accuracy of the proposed approximation as a function of

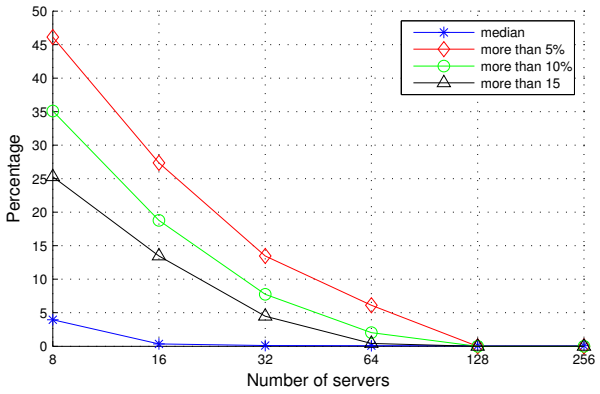


Figure 4.2: Distribution of the percentage relative errors in the approximate solution for the mean number of requests in a $Ph/Ph/c$ queue.

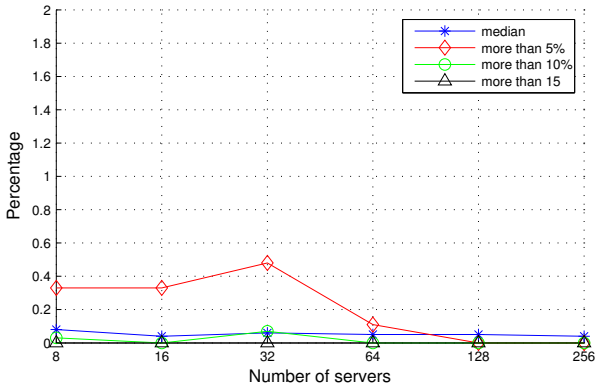
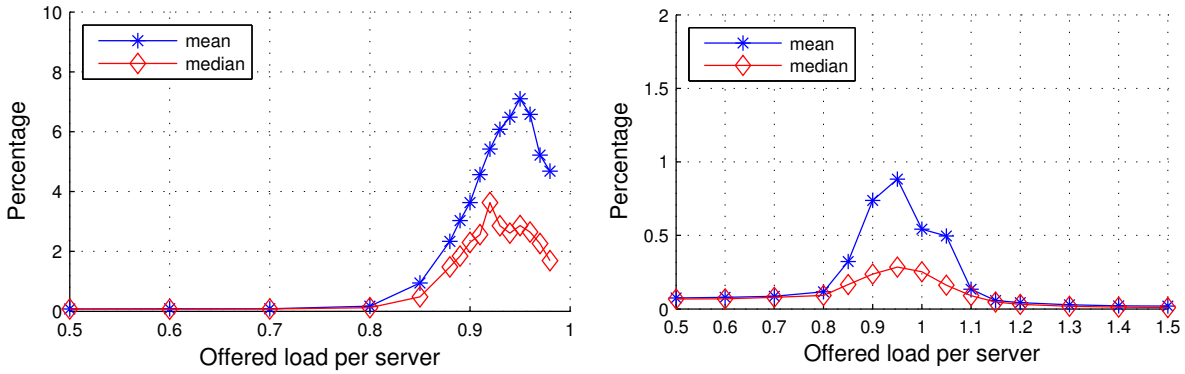


Figure 4.3: Distribution of the percentage relative errors in the approximate solution for the mean number of requests in a $Ph/Ph/c/N$ queue.

the offered load per server. As an example, we show in Figure 4.4 the results obtained with servers and other parameters spanning the spectrum of values for the coefficients of variations of the distributions of the time between arrivals and of the service time, as well as buffer sizes for finite-buffer queues, described above. We observe that the mean relative error for the mean number of requests in the system tends to peak when the offered load per server is around 0.95 (we believe that queues with unrestricted buffer approach their limit of practical validity when the load exceeds 0.9). In our results, for the open queue with 64 servers, the mean relative error peaks at some 7%, while for queues with a finite buffer the corresponding mean relative error peaks at less than 1%. The median values of the relative error peak at around 4% for the open queue and less than 0.5% for finite-buffer queues.

4.6.2 Loss Probability with finite buffers

Figure 4.5 illustrates the accuracy of the proposed approach for the loss probability in the case of queues with finite buffers. We observe that the percentage of cases in which the error exceeds 5% decreases from some 8% for 8 servers to 3% with 32 servers and all the way to about 1% with 256 servers. The median error remains well below 0.5% in all the cases studied.



(a) Case of a $Ph/Ph/c$ queue.

(b) Case of a $Ph/Ph/c/N$ queue.

Figure 4.4: Percentage relative errors in the approximate solution for the mean number of requests with c=64 servers as a function of the offered load per server.

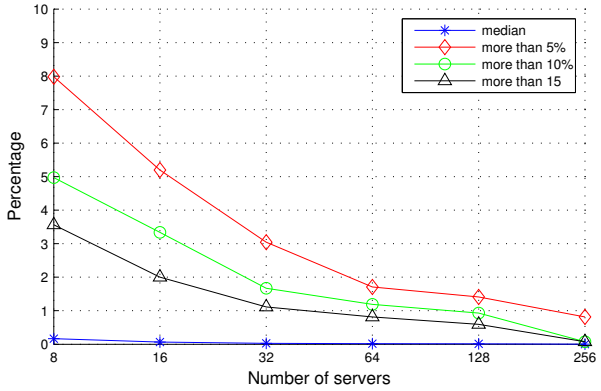


Figure 4.5: Distribution of the percentage relative errors in the approximate solution for the loss probability in a $Ph/Ph/c/N$ queue.

4.6.3 Wait Probability

Table 4.2 summarizes the results obtained for the probability that an arriving request has to wait in the case of queues with unrestricted buffer. In the over 1,400 examples studied, the relative errors remain below 5% in over 96% of the cases explored. The mean error is around 1% and the median error is below 0.5%.

Average	**Median**	**<5%**	**<10%**	**<15%**	**>15%**
1.02%	0.22%	96.05%	98.33%	99.20%	0.80%

Table 4.2: Distribution of the percentage relative errors in the approximate solution for the wait probability in a $Ph/Ph/c$ queue.

Table 4.3 displays analogous results for queues with finite buffers. Here we observe that the relative error remains below 5% in over 98% of the 16,170 cases studied. The mean error is below 1% and the median error is less than 0.1% confirming the impressive accuracy of the proposed approach in the case of finite buffer space.

Average	**Median**	**<5%**	**<10%**	**<15%**	**>15%**
0.60%	0.08%	98.23%	99.5%	99.75%	0.25%

Table 4.3: Distribution of the percentage relative errors in the approximate solution for the wait probability in a *Ph/Ph/c/N* queue.

4.6.4 Speed of Asymptotic Convergence for queues with unrestricted buffer

As discussed in Section 4.4, in the case of unrestricted buffer size our approximation relies on the convergence of the conditional probabilities $p(j,i|n)$ to their asymptotic values $\widetilde{p}(j,i)$ in order to transform an infinite set of equations into a finite one without arbitrary truncation. Table 4.4 shows the mean and the median values of the ratio $\widetilde{n}/c$ with a rather stringent $\delta = 10^{-11}$. Recall that δ corresponds to the point at which we consider that asymptotic convergence has been achieved for the conditional rates $w(n)$ and $u(n)$. We observe that the mean and median values of $\widetilde{n}$ grow less than linearly as the number of servers increases. Clearly, the value of $\widetilde{n}$ increases as the number of servers increases, but relative to the number of servers, the rate of growth slows down markedly as c increases.

	$c=8$	$c=16$	$c=32$	$c=64$	$c=128$	$c=256$
Mean	33.4	22.3	16.0	12.6	9.9	8.0
Median	35.9	22.4	15.9	12.8	9.9	8.2

Table 4.4: Mean and median values found for the ratio $\widetilde{n}/c$.

4.6.5 Speed of Convergence of the Fixed-Point Iterative Solution

The proposed solution of the equations for $p(j,i|n)$ (which is fully presented in [D12]) is a rather straightforward fixed-point iteration, presented as a simple "proof of concept". Nonetheless, it may be interesting to examine the number of iterations required to attain convergence. The latter varies with specific queue parameters but, on average, seems to depend mostly on the number of servers in the system. Interestingly, the median number of iterations needed is sufficiently close for queues with unrestricted and with finite buffer that they can be displayed together. Table 4.5 summarizes the median number of iterations relative to (i.e., divided by) the number of servers with $\epsilon = 10^{-8}$.

	$c=8$	$c=16$	$c=32$	$c=64$	$c=128$	$c=256$
Median	143.1	143.3	136.3	127.6	118.2	109.5

Table 4.5: Median number of iterations before convergence is found.

We observe that, while the number of iterations clearly grows with the number of servers, the rate of increase tends to be less than linear, especially for larger number of servers.

4.6.6 Model with State Dependencies

The good accuracy of the proposed approximation appears to extend to the case when the intensity of the arrivals and the service phases depend on the current number of requests in the system. As an example, Figure 4.6 compares the results obtained for the mean

number of request in the system using our approximation and using an exact numerical solution of the full balance equations for several levels of the offered load in the queue. This example corresponds to a system with $c = 32$ servers and a finite queueing room of $N = 128$. The service process has a coefficient of variation $c_s = 3$. The completion rate of each service phase is a function of the current number of requests in the system such that the service rate decreases linearly from full speed for a single user down to 50% of its full speed as the number of users reaches full system capacity N. The arrival process is quasi-Poisson with rate $\lambda(n) = \phi.(1 - \frac{n}{3N})$, i.e., it represents a set of $3N$ identical exponential request sources. Figure 4.6 illustrates the good accuracy of the approximation for the mean number of requests in the system as a function of the maximum offered load ϕ.

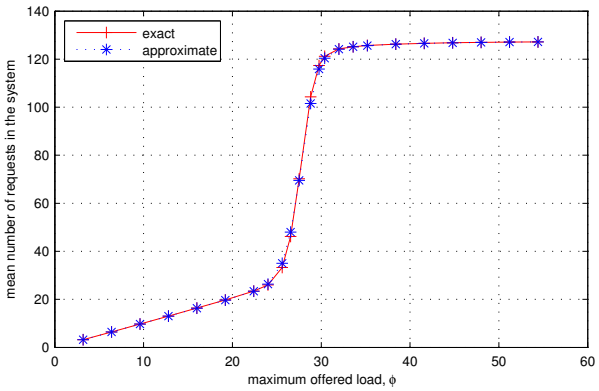


Figure 4.6: Mean number of requests in $Ph/Ph/c/N$ queue with $c = 32$, $N = 4c$, $c_s = 3$ and state dependencies for the service process and the arrival process for various levels of workload.

Overall, in the many examples explored, the accuracy of the proposed method appears excellent, especially for larger numbers of servers.

4.6.7 Computational Complexity

We now consider the issue of computational complexity of the proposed approach. As mentioned in the introduction, there are essentially two possible full state descriptions for an $Ph/Ph/c/N$ queue. The first description consists of the number of requests, the current phase of the arrival process and the vector of the current number of servers in each phase of the service process. The total number of states for this description is given by

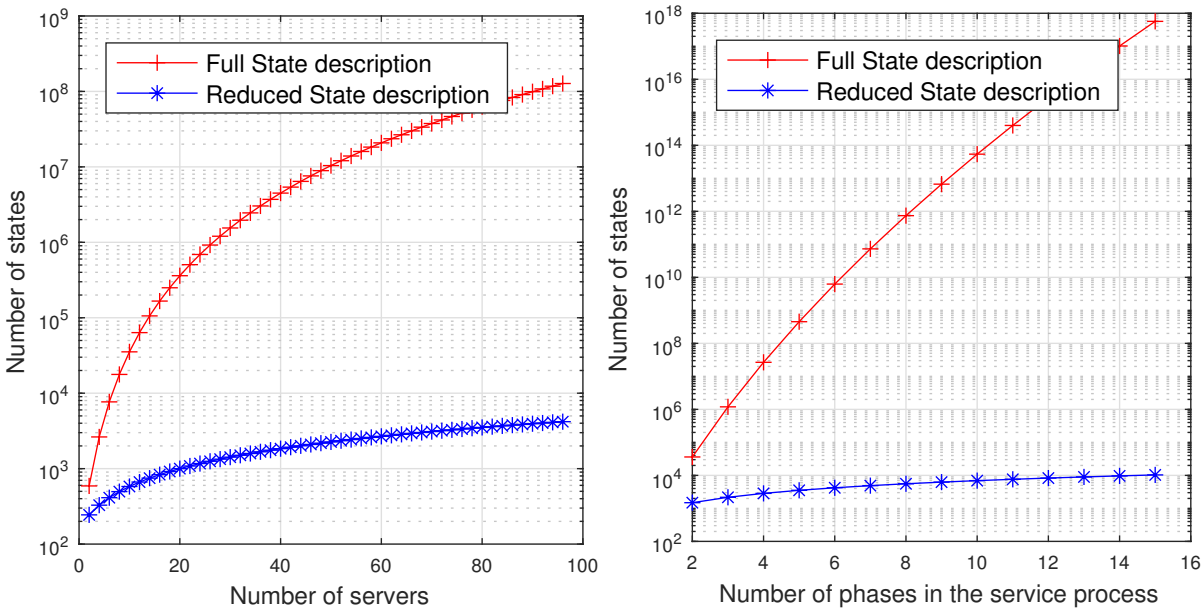
$$a\left(1 + \sum_{n=1}^{c-1} \binom{b+n-1}{n} + (N-c+1)\binom{b+c-1}{c}\right).$$

The second description involves the current number of requests in the system, the current phase of the arrival process and the vector of the current phases for each server. The total number of states in this state description is given by

$$a\left(1 + \sum_{n=1}^{c-1} \binom{c}{n} b^n + (N-c+1)b^c\right).$$

As for our approximation, the number of states in the proposed reduced state description is given by

$$a\Big(1 + (c-1)(b+1) + (N-c+1)b\Big) = a(c+Nb).$$



(a) As a function of the number of servers c given $b = 4$ phases for the service time process.

(b) As a function of the number of phases in the service process b given $c = 64$ servers.

Figure 4.7: Number of states induced by our reduced state description versus a full state description for a $Ph/Ph/c/N$ queue with a buffer of size $N = 5c + 20$ and a number of phases in the arrival process $a = 2$.

Figure 4.7 compares the computational complexity in the number of states between the first full description and our reduced state description. We do not include the second full state description in our figure since the number of states is systematically higher than for the first one. It is obvious from Figure 4.7 that, as the number of servers and phases increases, there is a difference of many orders of magnitude between the complexity of the full state description and our proposed reduced state description. Even with a relatively small number of phases in the service process (say 4) and only 64 servers, the complexity of the full state description results in about 25,000,000 states while the reduced state description involves less than 3,000 states (see Figure 4.7a). The difference gets even more dramatic for larger numbers of phases. With 8 phases and 64 servers, there are almost 1 trillion states in the full description compared with less than 5600 states for our method, amounting to 8 orders of magnitude difference (see Figure 4.7b).

4.6.8 Example of comparison with simple approximations

As mentioned in the introduction, the few existing simple approximations fail to account for potentially important distributional dependencies in the $G/G/c$ queue. Figure 4.8 illustrates the resulting relative error for the mean number of requests in the system in a queue with unrestricted buffer with $c = 16$ servers, the coefficient of variation of the time between arrivals $c_a = 3$ and the coefficient of variation of the service time $c_s = 0.5$ for two simple approximations: the Allen-Cunneen approximation and the approximation proposed by Kulbatzki [D7].

We observe that both simple approximations produce significant errors as the load per server approaches 0.8 while the relative error in our approximation remains small. Similar large errors occur in many other examples (e.g., see [D3]).

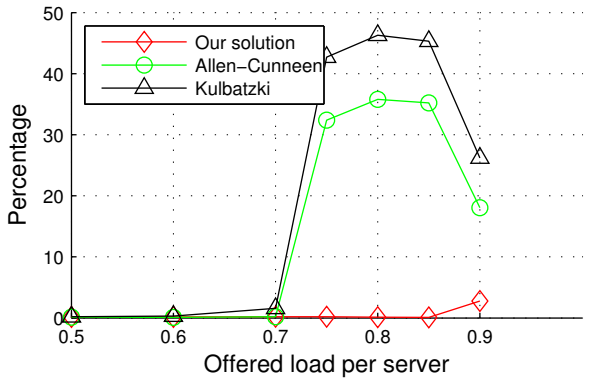


Figure 4.8: Relative error in the mean number of requests in $Ph/Ph/c$ queue with $c = 16$, $c_s = 0.5$, and $c_a = 3$ for various levels of workload.

4.7 Conclusions

Summary of Contributions

We propose a simple approximate solution for $Ph/Ph/c$ queues, which uses a reduced state description in order to circumvent the combinatorial growth of the number of states inherent in the classical state description used for such queueing systems. The number of equations to solve in our approach grows only linearly with the number of servers and phases in the distributions of times between arrivals and of service times. A simple fixed-point iteration is used to solve these equations. Although we do not have a theoretical proof of convergence of the fixed-point iteration, in the very many examples explored, it never failed to converge within a reasonable number of iterations.

We used common performance indices such as the mean number of requests in the system, the wait probability, as well as the loss probability in the case of queues with finite buffers to assess the accuracy of our approximation. Our results indicate that the accuracy of the approximation is generally very good for 8 or more servers and tends to improve rapidly as the number of servers grows. The median relative error for the mean number in the queue over the totality of examples considered (17,640 distinct scenarios) is below 0.1% and the relative error exceeds 5% in less than 1.5% of the scenarios considered. For the wait probability and the loss probability, the median error is below 0.5% and the observed errors exceed 5% in less than 4% of the cases explored. Overall, the accuracy of the proposed approximation appears very good.

Encountered Difficulties

We encountered some difficulties in obtaining the numerical results. While the solution to our approximation remained fast throughout all the explored examples, running the discrete-event simulator on some examples proved to be very lengthy. In particular, the simulator took several hours to run certain examples with a large number of servers (say more than 64), a high or low variability of their service and arrival process (say a coefficient of variation of less than 0.5, or above 3), and a relatively high offered load. Because we explored thousands of examples in our validation, and while most of this exploration was automatized and performed on computing servers, the overall process of validation still

took a couple of weeks (due to these lengthy simulations).

Strengths and Limitations

In my opinion, the use of the reduced state description provides two advantages to the proposed approximation. First, it provides a fair (though partial) description of the system, thereby limiting the loss of accuracy. Second, its complexity (number of states) only grows linearly with the number of servers and phases in the service time and inter-arrival distributions.

Like with any approximation, it is important to understand its accuracy and its domain of applicability. Unfortunately, we have not been able to obtain theoretical bounds for the errors of our approximation. This is why we undertook a systematic empirical study of its behavior over a large spectrum of examples attempting to cover cases that tend to cause problems.

Possible Extensions

In my opinion, the reduced state description that is the cornerstone of this approximation could probably be applied, with some adjustments, to other systems, too. In particular, systems of large dimensions in which many components can be viewed as homogeneous (statistically identical) are natural candidates.

More generally, I believe that several classical problems in queueing theory, if addressed using a full and exact system description, become so complex that their solution, if any, requires approximations. In some cases, a better option may be to consider a possibly approximate state description for which the solution can be found easily and involves little or no approximation. Said differently, for certain systems, it might be preferable to introduce a small modeling error that leads to a small solution error as opposed to starting with an exact model whose approximate solution ensues significant errors or limitations. In this regard, together with Pr. Brandwajn, we propose a new solution to the multi-server queue with the First-Come-First-Serve discipline, based on an original state description, that stands out by its simplicity (currently in revision) [D13].

Potential Impact

In order to promote the use of our work, we developed a website[1] that allows anyone to run our approximation on their own examples. This tool was presented at the conference [D2] and it features the solutions to several classical queueing systems. It attracts each month hundreds of visitors from all around the world.

In my opinion, our work on the design of simple and computationally efficient solutions for queueing systems has attracted modest interest so far despite being favorably received in well-established journals (e.g., *Performance Evaluation*, *Computers & Operations Research*, *Journal of Applied Probability*). I believe that this relative lack of visibility has to do with the relative "coldness" of the topic (see discussion in Section 1.3). Unlike other research areas such as SDN, 5G, IoT, there seems to be less excitement around the queueing theory at the moment. However, its impact may grow in the long run.

[1] `http://queueing-systems.ens-lyon.fr/`

Learned Experiences

Skill-wise, in addition to queueing theory, I improved my abilities in simulations. Especially when facing lengthy and complex ones, automation becomes mandatory but it must come with a fine examination of the obtained results.

More generally, it surprised me that no ones ever made a full use of the fact that, in a multi-server queue, servers are typically assumed to be homogeneous (statistically identical). In other words, this works convinced me that *Even on a long-standing and complex problem, there may still be space for new approaches.*

References for Chapter 4

[D1] S. Asmussen and J. R. Møller. Calculation of the steady state waiting time distribution in *GI/PH/c* and *MAP/PH/c* queues. *Queueing Systems*, 37(1-3):9–29, 2001.

[D2] T. Begin and A. Brandwajn. A tool for solving *Ph/M/c* and *Ph/M/c/N* queues. In *ACM QEST*, 2012.

[D3] T. Begin and A. Brandwajn. A note on the accuracy of several existing approximations for *M/Ph/m* queues. In *HSNCE*, 2013.

[D4] D. Bertsimas. An analytic approach to a general class of *G/G/s* queueing systems. *Operations Research*, 38(1):139–155, 1990.

[D5] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains*. Oxford University Press on Demand, 2005.

[D6] A. Bobbio, A. Horváth, and M. Telek. Matching three moments with minimal acyclic phase type distributions. *Stochastic Models*, 21(2-3):303–326, 2005.

[D7] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[D8] S. Borkar and A. A. Chien. The future of microprocessors. *Communications of the ACM*, 54(5):67–77, 2011.

[D9] A. Brandwajn and T. Begin. Performance evaluation of a single node with general arrivals and service. In *ASMTA*. Springer, 2011.

[D10] A. Brandwajn and T. Begin. An approximate solution for *Ph/Ph/1* and *Ph/Ph/1/N* queues. In *ACM/SPEC ICPE*, 2012.

[D11] A. Brandwajn and T. Begin. Reduced complexity in *M/Ph/c/N* queues. *Performance Evaluation*, 78:42–54, 2014.

[D12] A. Brandwajn and T. Begin. Breaking the dimensionality curse in multi-server queues. *Computers & Operations Research*, 73:141–149, 2016.

[D13] A. Brandwajn and T. Begin. First-Come-First-Served Queues with Multiple Servers and Customer Classes. *Performance Evaluation*, 2018.

[D14] D. R. Cox. A use of complex probabilities in the theory of stochastic processes. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 51, pages 313–319. Cambridge University Press, 1955.

[D15] J. H. De Smit. The queue *GI/M/s* with customers of different types or the queue *GI/H m/s*. *Advances in Applied Probability*, 15(2):392–419, 1983.

[D16] N. Gans, G. Koole, and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing & Service Operations Management*, 5(2):79–141, 2003.

[D17] V. Gupta, M. Harchol-Balter, J. Dai, and B. Zwart. On the inapproximability of *M/G/K*: why two moments of job size distribution are not enough. *Queueing Systems*, 64(1):5–48, 2010.

[D18] M. Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[D19] A. Ishikawa. On the equilibrium solution for the queueing system *GI/Ek/m*. *TRU Mathematics*, 15(1):47–66, 1979.

[D20] H. Khazaei, J. Misic, and V. B. Misic. Performance analysis of cloud computing centers using *M/G/m/m+ r* queuing systems. *IEEE Transactions on Parallel and Distributed Systems*, 23(5):936–943, 2012.

[D21] G. Koole and A. Mandelbaum. Queueing models of call centers: An introduction. *Annals of Operations Research*, 113(1-4):41–59, 2002.

[D22] G. Latouche and V. Ramaswami. A logarithmic reduction algorithm for quasi-birth-death processes. *Journal of Applied Probability*, 30(3):650–674, 1993.

[D23] A. S. Meritt, J. A. Staubi, K. M. Trowell, G. Whistance, and H. M. Yudenfriend. z/OS support of the IBM TotalStorage enterprise storage server. *IBM Systems Journal*, 42(2):280–301, 2003.

[D24] T. Osogami and M. Harchol-Balter. Closed form solutions for mapping general distributions to quasi-minimal PH distributions. *Performance Evaluation*, 63(6):524–552, 2006.

[D25] V. Ramaswami and D. M. Lucantoni. Algorithms for the multi-server queue with phase type service. *Stochastic Models*, 1(3):393–417, 1985.

[D26] L. Seelen. An algorithm for *Ph/Ph/c* queues. *European Journal of Operational Research*, 23(1):118–127, 1986.

[D27] M. Taaffe and M. Johnson. The denseness of phase distributions. *School of Industrial Engineering Purdue University Research Memorandum*, pages 88–20, 1988.

[D28] Y. Takahashi. Asymptotic exponentiality of the tail of the waiting-time distribution in a *Ph/Ph/c* queue. *Advances in Applied Probability*, 13(3):619–630, 1981.

[D29] W. Whitt. A diffusion approximation for the *G/GI/n/m* queue. *Operations Research*, 52(6):922–941, 2004.

4.8 Appendix

Balance Equations and Equations for Conditional Probabilities

Denote by r_{jk} the probability that phase j of the arrival process is followed by phase k, and by q_{il} the probability that phase i of the service process is followed by phase l.

The balance equations for the reduced state description $p(n, j, i)$ considered in this paper have the following form. For $n > c$, $j = 1, \ldots, a$, $i = 1, \ldots, b$

$$\begin{aligned} p(n,j,i)[\lambda_j + \mu_i + \nu(n,j,i)] = \quad & \textstyle\sum_{k=1}^{a} p(n-1,k,i)\lambda_k \hat{r}_k \tau_j + \sum_{l=1}^{b} p(n,j,l)\mu_l q_{li} \\ & + \textstyle\sum_{k=1}^{a} p(n,k,i)\lambda_k r_{kj} + p(n+1,j,i)\nu(n+1,j,i) \\ & + \textstyle\sum_{l=1}^{b} p(n+1,j,l)\mu_l \hat{q}_l \sigma_i. \end{aligned} \tag{4.12}$$

The corresponding balance equations for other values of n can be found in [D12]. Note that these balance equations are exact in the sense that, if we knew the exact values for the conditional completion rates $\nu(n, j, i)$, the solution of these balance equations would produce the exact steady-state probabilities $p(n, j, i)$. The approximation in our method stems from the fact that we compute an approximate value for $\nu(n, j, i)$.

We present below the conditional probability equations used in the solution of the open model. For $n > c$, $j = 1, \ldots, a$, $i = 1, \ldots, b$

$$\begin{aligned} p(j,i|n)[\lambda_j + \mu_i + \nu(n,j,i)] = \quad & \textstyle\sum_{k=1}^{a} p(k,i|n-1)\lambda_k \hat{r}_k \tau_j u(n)/w(n-1) \\ & + [p(j,i|n+1)\nu(n+1,j,i) + \textstyle\sum_{l=1}^{b} p(j,l|n+1)\mu_l \hat{q}_l \sigma_i] w(n)/u(n+1) \\ & + \textstyle\sum_{l=1}^{b} p(j,l|n)\mu_l q_{li} + \sigma_{k=1}^{a} p(k,i|n)\lambda_k r_{kj}. \end{aligned} \tag{4.13}$$

Here also, the conditional probability equations for other values of n as well as for $n = N$ in the case of a finite buffer are given in [D12].

Chapter 5

Conclusions

Contents

5.1 "Missing" Contributions **84**

5.2 The Importance of Accuracy in Modeling **85**

5.3 Work methods and Good practices **87**

5.4 Scientific Challenges and Prospects for the Future **89**

5.1 "Missing" Contributions

The last three chapters have attempted to provide a broad and critical overview of three of my main contributions. I decided not to describe the other contributions for the sake of conciseness. However, these latter have all been presented in publications (see Section 1.4 of Chapter 1 for a full list).

In this section, I would like to briefly discuss the contributions that I attempted to obtain but unfortunately failed to. I refer to those as the "missing contributions". Some were caused by refusals to fund research projects. For example, a couple of years ago, I participated in a project proposal introducing aspects of machine learning for the parameterization of wireless networks. The project was submitted to an international call between Hong-Kong and France. Although the project was graded as "Excellent" by the experts, it ultimately got refused two years in a row. This restrained us to tackle this issue, which has now become a very hot topic for computer networks.

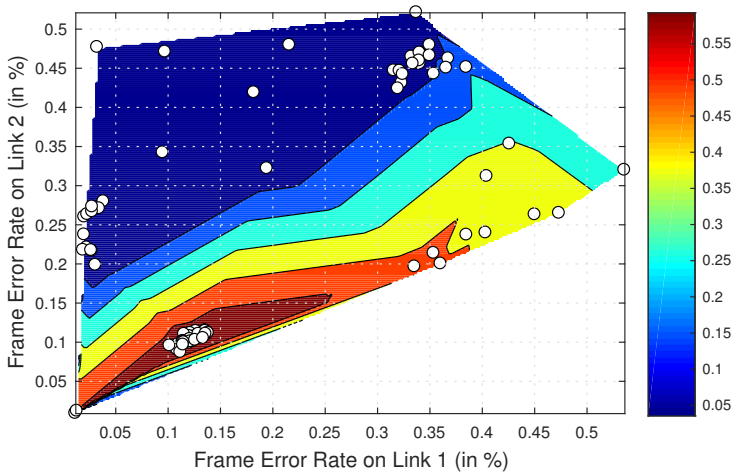


Figure 5.1: Maximum Attained Throughput (in Mbps) along an IEEE 802.11b Three-node Chain with Transmission Rate at 1 Mbps as a function of the Frame Error Rates on both links.

Other missing contributions may be due to the lack of time and adequate equipment as well as technical obstacles. In this regard, together with Pr. Guérin Lassous, we share this ambition of validating theoretical models with real-life experiments (and not only with a discrete-event simulator). Therefore, we developed and deployed an IEEE 802.11 testbed in our lab using standard machines and coding our own scripts. Our pursued objective was to collect performance measurements and to compare them with those delivered by a priorly developed theoretical model [G4], [H1, H2]. Our efforts were hindered by many difficulties including that our equipment was too old to support newer amendments of IEEE 802.11 (potentially making the obtained results viewed as outdated), and that the used machines had heterogeneous network interface card (NIC) causing unexpected behaviors. We managed though to get measurements of performance parameters for an IEEE 802.11 network in several configurations (using the 802.11b amendment with low transmission rates). Figure 5.1 shows an excerpt of our results in which we measure the maximum attained throughput of a flow routed along a chain made of three nodes and two links. Note that x-axis (resp. y-axis) corresponds to the Frame Error Rate (FER) experienced over the fist link (resp. second link). We observe that broadly speaking the largest value of attained throughput are obtained for small values of FER on both link. Unfortunately, it appears that we had trouble at evenly sampling the space of FER combinations as many measurement points fall in the same areas. Overall, the

measurement process revealed to be technically complex and lengthy, and although we spent much energy on it, we viewed the results as too incomplete to be included in a publication.

As for the other missing contributions, I believe that they are mostly due to insufficient skills in some fields. For example, although I had the opportunity to familiarize myself with some optimization techniques (e.g., Derivative-Free Optimization technique), I often met optimization issues that postponed or deviated my research direction.

5.2 Discussion on the Importance of Accuracy in Modeling

Since the introduction of my first theoretical models, I have faced the recurrent question of their accuracy, often expressed as a concern by external referees. Although I do not question their motives, I'd like to discuss some thoughts of mine on this matter.

How Accurate is Accurate?

Assessing the accuracy of a theoretical model is a complex matter. In fact, I find the following questions hardly answerable: *How accurate should be a model? Is an accuracy of 1%, or 5%, or 10% enough? Or does it even matter?* Before going any further in this discussion, it is worth reconsidering what are the modeling motives. As discussed in Chapter 1, a model can essentially help to explain a system behavior, to understand the functionality of its components, and to forecast its performance. Additionally, a model can provide qualitative or quantitative insights.

For instance, I don't believe the Erlang C model [E11] was ever designed to closely fit real call centers. However, it became famous and useful as it captures, in a very simple way, the main phenomena taking place in a call center (e.g., call arrival rate, average call-holding time, number of telephone lines). I believe that the same can be said for several renowned models (e.g., central server model proposed by Buzen [E6], packet-switched network model by Kleinrock [E15]). All these models would have been probably poorly received if they had been evaluated solely by their level of accuracy. Indeed, in my opinion, their value lies more in their qualitative abilities than in their quantitative predictions.

On the other hand, when quantitative purposes outweigh qualitative ones, the accuracy level (namely, the relative error) becomes a much more relevant criterion for assessing a model. But how accurate should it be? As far as I know, it seems that the desired (or required) level of accuracy is generally higher from the academic point of views than from practitioner's. In fact, I was told a couple of times that, in the industry, an accuracy of about 20% is generally viewed satisfactorily. On the other hand, in my experience, academics tend to be more stringent, sometimes even expecting an accuracy below 5%[1]. Unfortunately, I believe that such a requirement often comes at the expense of the model simplicity. Additionally, one needs to keep in mind that the tool used for validation (be it a simulator or real-life experiences) may not provide this level accuracy anyway.

Overall, in my opinion, when modeling a system, the ultimate goal is not necessarily to come up with a very accurate model. Instead, the objective is to find an adequate trade-off between simplicity and accuracy. Unfortunately, I don't believe there is a simple and general rule to determine a right level of accuracy as this latter greatly depends

[1]Note that relative errors may not be an adequate choice when values are close to 0.

on the actual uses of the model (i.e., mostly qualitative or quantitative). Nonetheless, broadly speaking, I would tend to agree with practitioners who look favorably at a level of accuracy around 10 or 20% for a quantitative model.

Ethical Considerations

The issue of validating a model accuracy also raises ethical questions. Indeed, given the generally large size of the space of parameters, the accuracy of a model may widely vary with the value of its parameters. For example, in the case of an IEEE 802.11-based network, the number of parameters that can potentially affect its performance are almost countless: the number of nodes, the node locations, the offered load at each node, the channel transmission rates, the amendment of 802.11, the buffer sizes, the routing, and the transport protocols to name a few. Therefore, regardless of the real value of a proposed model or solution, I believe it would always be possible to find some parameter combinations in which excellent results will be delivered. The same goes in queueing theory where proposed approximations intended to work for general distributions of the service or inter-arrival times are sometimes validated with not so variable distributions. For example, if a value of barely 1.4 is used for the coefficient of variation of the service time distributions, I believe that a solution consisting in simply replacing the actual service time distribution by an exponential distribution will probably do just fine. Other examples include cases where multi-server queues approximations have been validated at utilization levels where basically all forms of queues behave similarly. Although I am deeply convinced that researchers are overwhelmingly willing to be fair and critical in their scientific advancements and contributions, the growing pressure for publishing rapidly and continually, coined in the phrase *Publish or perish*, may lead to the over-representation of successful cases.

Statistical Approaches and Benchmarking

Computer science is not the only scientific discipline to face issues related to the validation of a model, a hypothesis or a solution. A case in point is polls and surveys where a limited number of individuals are sampled from the whole population in order to estimate the population opinion on a given issue. Although each sample taken separately does not reveal much information on the population opinion, a well-chosen set of samples does. Analogously, I believe that, in order to evaluate a model accuracy or a solution efficiency, computer scientists would benefit from applying such statistical approaches. In practice, a fair approach could consist of exploring numerous examples that together browse a large range of values for the system parameters, including favorable and unfavorable examples. Then, the obtained results can be presented as a distribution of the level of accuracy, together with the mean and median values.

Another avenue to address the issue of validating a model accuracy could be benchmarking. In the field of image processing, researchers have agreed to use a standard test image introduced in 1973. This picture, referred to as the Lenna image, contains a nice mixture of detail, flat regions, shading, and texture that make it well-suited for testing different image processing algorithms. Since then, other test images have been included to form a broader benchmark. Similarly, the numerical linear algebra community is using a large set of sparse matrices known as the University of Florida Sparse Matrix Collection [E10] for the development and performance evaluation of sparse matrix algorithms. In my opinion, the computer network and queueing theory communities would both benefit from

defining analogous standard scenarios. These latter would not only help the validation process but also enable clear comparisons of different solutions on a common ground. Of course, defining these standard scenarios is not a simple task since many aspects must be taken into account when setting them up. The other side of the coin is that an evaluation through standard scenarios may lead to tailor-made approximations and ad-hoc solutions that may underperform when placed in other contexts. However, I believe that a well-rounded set of scenarios can mitigate this risk.

Note that some elements in the computer network discipline may be viewed as steps in this direction. For instance, in the case of the computer network community, the topology suggested by Chen [E9] for representing the backbone of a North American network operator has been subsequently re-used by a number of researchers including myself. Another example is the set of anonymized Internet traces provided by CAIDA [E7].

5.3 Work methods and Good practices

In this section, I shortly discuss some work methods and good practices built up from my own experience. By no means, I assume that these latter apply to anyone, but at least they work for me.

A Rigorous Approach

Everyone would probably agree that rigor is a big asset, if not an absolute necessity, for conducting scientific research. Indeed, any form of research requires extremely thorough and careful analysis. I would also add that a rigorous approach can represent a significant source of work efficiency. Although rigor may sometimes be viewed as a hindrance to progress in the short-run, I believe that research is a long process that builds upon convincing proofs and sound materials. In the long-run, a rigorous approach is the best means to avoid repeating the same tasks (e.g., double-checking due to inconsistencies) or to reuse materials from previous projects (as is often needed due the incremental nature of research). Christopher Thompson formulated this idea in a funny way for coders: *Sometimes it pays to stay in bed on Monday, rather than spending the rest of the week debugging Monday's code.*

Start Small and Test Extensively

When tackling issues such as modeling a system, implementing a simulator, or collecting real-life measurements, I believe that a sensible approach is to start with a very simple version, which does not involve all details, and to check its good operation right away. This approach is analogous to the use of unit testing when developing large software systems. More generally, I advocate incremental methods where additional complexity is gradually introduced with systematic tests and validations. By doing so, insights and experience will be progressively gained and, equally importantly, potential flaws or errors will be detected at early stage saving time and effort. Lately, a colleague who was in charge of collecting simulation results for a joint-work spent nearly 3 months designing an elaborate simulator before testing it (upon our request). It turns out that, despite its efforts, the simulator was unable to produce correct results on the simplest possible example. Much time and energy were wasted on this work.

Stay Open-Minded

I believe that in general there are several effective ways to address a research problem. Therefore, when discussing research activities (be it through a review process for a journal, at a conference, in an office or at the coffee room), it is always worth paying great attention to any feedback coming from colleagues. A critical assessment of a research work, even if not stated in a constructive manner, may provide useful information. More generally, I would advocate having numerous scientific exchanges, including with researchers in different communities as they may provide a new outlook. In my experience, seemingly trivial discussions may ultimately lead to original ideas for research.

Seek Simplicity

As discussed in Chapter 1, my research activities have in common that I seek simple solutions for complex problems. In practice, as discussed above, I typically start with a simple solution and build on that, introducing complexities only when I cannot find any workaround. In my opinion, complexity may sometimes result from a wrong, or at least not sharp enough, understanding of a problem. On the other hand, I believe that the simplicity of some scientific contributions does not necessarily come easily. Indeed, in many cases, once an easy solution has been found, the latter became obvious (e.g., reduced state description where one only of the many homogeneous servers is fully described [E4, E5]). This idea was nicely captured by Steve Jobs: *Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple.*

Helping Ph.D. students

So far I have co-supervized 4 Ph.D. students and I am currently supervizing the 5th. Although each case is a different one, I have come to establish a certain set of practices that can help Ph.D. students get through their research study. Clearly, these practices are based on my own experience and they may not apply to anyone. I list here some of the most important ones.

- Having regular meetings, say weekly;
- Asking for regular progress reports;
- Exchanging ideas and points of views (listening to students as much as talking to them);
- Encouraging them to present their research activities whenever possible;
- Avoiding overloading and multi-tasking;
- Building a relationship of trust;
- Giving them enough autonomy and making them responsible;
- Informing them about the research community and career prospects.

5.4 Scientific Challenges and Prospects for the Future

I will now discuss how I envision my research activities for the next years to come. However, I would like first to add the following disclaimer. The content of this section is largely a forecast and is only indicative. Who knows how it will turn out? In other words, to paraphrase our politicians, I'd like to make it clear that: *"Ces écrits n'engagent que ceux qui les lisent"*.

Machine Learning for Computer Networks

As Feamster, Rexford and Zegura explained in their 2013 paper [E12], computer networks have become increasingly complex and difficult to manage. They involve many kinds of equipment, from routers and switches to middleboxes such as firewalls, network address translators, server load balancers, and intrusion-detection systems. Routers, switches and access points typically run complex, distributed control software that is typically closed and proprietary. Furthermore, communications between nodes involve several protocols from different layers in the OSI model with potentially ineffective interactions. Because of the large dimensions and undocumented aspects of computer networks, using a constructive approach for performance modeling has become significantly more complex over the last years [E20].

On the other hand, remarkable progress has been achieved in the area of Artificial Intelligence (AI) and Machine Learning (ML) but they have not yet fully percolated through the networking community. The following quote is an excerpt from the website of the 2018 Workshop on AI in Networks (WAIN) jointly organized with IFIP WP Performance[2]: *AI and ML are currently being exploited in almost every scientific fields. However, computer networks have still a limited development and deployment of these techniques.* I believe that computer networks may be a fruitful field for the application of ML techniques. Indeed, these latter are particularly useful when addressing high-dimensional and large-scale problems as is often the case in computer networks. While Supervised learning techniques mostly aim at recognizing certain rules for correlating inputs to outputs based on a training set (samples of data), Unsupervised learning provides tools to find structures and patterns in the data themself. Most existing works where ML techniques have been applied to computer networks deal with issues such as node localization for indoor wireless networks (e.g., [E13, E21]), the spectrum utilization in cognitive radios [E3], or maybe to a lesser extent, the detection of intrusion (e.g., [E16]). As far I know little has been done in the context of performance evaluation.

Overall Strategy and Planned Projects

In a nutshell, my plan is to use techniques of ML to help the design of performance models for computer networks, or to guide the parameterization/configuration of existing networking solutions (e.g., protocols). I started to get interested in this subject about 4 or 5 years ago. I enrolled and completed a couple of Massive Open Online Courses (MOOC) dealing with AI and ML. This confirmed my desire to orient my research activities in this direction. Besides, during my Ph.D. and the supervizion of a Ph.D. student I had the opportunity to study nonlinear regression and techniques from unsupervized learning like k-means clustering.

[2] `https://performance2018.sciencesconf.org/page/wain`

Aside from ML techniques for computer networks, I intend to keep working on performance aspects of Network Function Virtualization (NFV) as well as on the design of simple and computationally efficient solutions for queueing systems. Without further ado, I now describe three planned projects that revolve around ML techniques.

A - Neural Network-based Performance Modeling of IEEE 802.11

Not only are IEEE 802.11 performance difficult to measure but they are hardly predictable. Indeed, their value depends on a vast number of parameters such as the used amendment of 802.11, the channel transmission rate, the number of competing nodes, the Frame Error Rate (FER), the offered load, and the transport protocol to name a few. This high-dimensionality contributes to hinder the design of robust and versatile models.

This prospective project would explore a new approach by letting a dataset of measurements determine a learned (data-driven) model of performance. This dataset, which will contain IEEE 802.11 measurements of throughput under various networking configurations, will serve as the training set for the model. Clearly, this project relies on a descriptive approach of modeling [E20] (as opposed to a constructive approach). The ultimate goal of this project is to provide an easy to use tool for predicting IEEE 802.11 performance given a particular network configuration.

In my opinion, the largest difficulties pertain to the definition of the training set. This requires to identify all parameters of an IEEE 802.11 network that may significantly affect its performance. Additionally, the process of collecting the performance values together with the corresponding network parameters (output and inputs) will prove to be complex and lengthy. Therefore, to begin with, and as a proof of concept, I intend to start by using discrete-event simulator (in place of real-life experiments) and considering a network made of only a couple of Access Points (APs).

On the bright side, I have a good understanding of IEEE 802.11 and I acquired experience with real-life 802.11 measurements (see Section 5.1). Last but not least, during my Ph.D., I developed a descriptive approach for performance modeling [E2] that involves concepts such as gradient descent, feature scaling, mean normalization, training, cross validation and test sets, overfitting and regularization factor.

This problem belongs to the class of Nonlinear Regression problems, and because of its large number of dimensions, I expect to use Artificial Neural Networks (that are known to deal easily with hundreds of dimensions).

B - Supervized Learning Approach for IEEE 802.11-based Multi-Hop Wireless Networks

Note that the ideas presented here are the results from joint-reflexions with Isabelle Guérin Lassous and Bruno Baynat.

The Distributed Coordination Function (DCF) technique employed by the Medium Access Control (MAC) protocol in IEEE 802.11 is known to be potentially unfair and poorly efficient when applied to Multi-Hop Wireless Networks (MWNs). A number of solutions have been proposed to address this issue by adapting some inner 802.11 parameters like its contention window (e.g., [E1, E17]) However, none of them seems able to perform well in any configuration of an MWN due to their lack of adaptability [E18]. In practice, some solutions may work just fine in some MWN configurations, and not in other configurations.

In this project, the main idea is to let each node of an MWN discover their environment so as to select the best adaptation of their parameters. The plan proceeds as follows. First, we will identify so-called "unitary scenarios", each representing a specific performance issue arising in MWNs (e.g., cell, symmetrical hidden node, asymmetrical hidden node, flow in the middle) [E8, E14]. Second, we will build a training set that includes MWNs in which we label some nodes to be in certain unitary scenarios (e.g., based on our understanding of 802.11 , or on networking parameters such as the number of one-hop neighbor nodes, medium utilization, idle periods, frame error rates, queueing delay). Third, we will use this set to train a classifier that would allow labeling the most likely unitary scenario for other nodes. The potential use of this project would be to enable nodes of an MWN to self-assess their environment, and to adapt the values of their MAC parameters accordingly.

In my opinion, the biggest difficulty lies in the definition of the unitary scenarios that aim together at covering all possible states of a node with overlapping as little as possible. To this aim, I will benefit from my previous experiences on IEEE 802.11 as well as those of my research team colleagues.

As such, this project is formulated as a Multi-Class (Semi-)Supervized Learning problem for which neural networks seem a natural candidate.

C - Graph Signal Processing

This third project is actually a recently started work with Marija Stojanova and Paulo Gonçalves. Instead of revolving around ML techniques, this work relies on a Graph Signal Processing (GSP) approach that, however, shares several aspects.

Graphs are a natural abstraction for representing a variety of systems, including computer and communication networks. Recently, GSP has emerged as a promising field for analyzing and modeling signals that are supported on a graph. GSP aims at extending classical signal processing concepts and tools such as Fourier transform, filtering and spectral response to data residing on graphs [E19].

While designing our performance modeling approach for IEEE 802.11-based WLAN (presented in Chapter 3), we wondered if the GSP toolbox could provide a much easier way of addressing this issue. Cast as a GSP problem, the considered signal on the graph representing the weights on the vertices (viz. 802.11 nodes) is bivariate. It should encode both the level of the load at this Access Point and the selected rate of channel transmission. Then, using a graph-based Moving Average filter (MA), we explore how this latter can accurately forecast the throughput attained by each node (filter output) of the network given the conflict graph (viz. network topology) between the nodes as well as the level of load and channel transmission rate on each node (filter inputs). The advantage of this approach, aside from its simplicity, is that it intrinsically captures the whole conflict graph as it involves the adjacent matrix.

The difficulty here lies in the coefficients of the MA filter. Capping the filter at a low order may lead to an inaccurate modeling of the data (bias) while considering a high order may result in overfitting the data.

Although it is too early to draw any conclusion on the effectiveness of this approach, our preliminary results appear to be encouraging.

Disseminating Knowledge

Aside from these scientific projects, I will keep disseminating the knowledge and skills generated from them. So far this has taken the form of lecture classes at the university (e.g., Master courses on SDN and NFV following research projects on these fields) or explanatory websites (e.g., `http://queueing-systems.ens-lyon.fr/` that illustrate some of our contributions in queueing theory).

References for Chapter 5

[E1] A. Aziz, D. Starobinski, and P. Thiran. Understanding and tackling the root causes of instability in wireless mesh networks. *IEEE/ACM Transactions on Networking (TON)*, 19(4):1178–1193, 2011.

[E2] T. Begin, A. Brandwajn, B. Baynat, B. E. Wolfinger, and S. Fdida. High-level approach to modeling of observed system behavior. *Performance Evaluation*, 67(5):386–405, 2010.

[E3] M. Bkassiny, Y. Li, and S. K. Jayaweera. A survey on machine-learning techniques in cognitive radios. *IEEE Communications Surveys & Tutorials*, 15(3):1136–1159, 2013.

[E4] A. Brandwajn and T. Begin. Reduced complexity in *M/Ph/c/N* queues. *Performance Evaluation*, 78:42–54, 2014.

[E5] A. Brandwajn and T. Begin. Breaking the dimensionality curse in multi-server queues. *Computers & Operations Research*, 73:141–149, 2016.

[E6] J. P. Buzen. *Queuing Network Models of Multiprogramming Ph. D.* PhD thesis, Thesis, Harvard University, Cambridge, MA, 1971.

[E7] CAIDA - Center for Applied Internet Data Analysis. http://www.caida.org.

[E8] C. Chaudet, D. Dhoutaut, and I. G. Lassous. Performance issues with IEEE 802.11 in ad hoc networking. *IEEE Communications magazine*, 43(7):110–116, 2005.

[E9] S. Chen and K. Nahrstedt. On finding multi-constrained paths. In *IEEE ICC*, volume 2, pages 874–879. IEEE, 1998.

[E10] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[E11] A. K. Erlang. Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges. *Post Office Electrical Engineer's Journal*, 10:189–197, 1917.

[E12] N. Feamster, J. Rexford, and E. Zegura. The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, 44(2):87–98, 2014.

[E13] C. Figuera, J. L. Rojo-Álvarez, M. Wilby, I. Mora-Jiménez, and A. J. Caamaño. Advanced support vector machines for 802.11 indoor location. *Signal Processing*, 92(9):2126–2136, 2012.

[E14] M. Garetto, T. Salonidis, and E. W. Knightly. Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 16(4):864–877, 2008.

[E15] L. Kleinrock. *Message delay in communication nets with storage*. PhD thesis, Massachusetts Institute of Technology, 1963.

[E16] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis. Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset. *IEEE Communications Surveys & Tutorials*, 18(1):184–208, 2016.

[E17] B. Nardelli and E. W. Knightly. Closed-form throughput expressions for CSMA networks with collisions and hidden terminals. In *IEEE INFOCOM*, pages 2309–2317. IEEE, 2012.

[E18] B. Nardelli, J. Lee, K. Lee, Y. Yi, S. Chong, E. W. Knightly, and M. Chiang. Experimental evaluation of optimal CSMA. In *IEEE INFOCOM*, pages 1188–1196. IEEE, 2011.

[E19] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[E20] K. Salamatian and S. Fdida. A framework for interpreting measurement over Internet. In *MoMeTools*, pages 87–94. ACM, 2003.

[E21] H. Zou, X. Lu, H. Jiang, and L. Xie. A fast and precise indoor localization algorithm based on an online sequential extreme learning machine. *Sensors*, 15(1):1804–1824, 2015.

Chapter 6

Co-Authored Publications

International Journals

[G1] Tulin Atmaca, Thomas Begin, Alexandre Brandwajn, and Hind Castel-Taleb. Performance evaluation of cloud computing centers with general arrivals and service. *IEEE Transactions on Parallel and Distributed Systems*, 27(8):2341–2348, 2016.

[G2] Thomas Begin, Bruno Baynat, Guillaume Artero Gallardo, and Vincent Jardin. An accurate and efficient modeling framework for the performance evaluation of DPDK-based virtual switches. In *IEEE Transactions on Network and Service Management*, pages 1–14, 2018.

[G3] Thomas Begin, Bruno Baynat, Alexandre Brandwajn, and Francis Sourd. A DFO technique to calibrate queueing models. *Computers & Operations Research*, 37(2):273–281, February 2009.

[G4] Thomas Begin, Bruno Baynat, Isabelle Guérin Lassous, and Thiago Abreu. Performance analysis of multi-hop flows in IEEE 802.11 networks: A flexible and accurate modeling framework. *Performance Evaluation*, 96:12–32, 2016.

[G5] Thomas Begin and Alexandre Brandwajn. Higher-order distributional properties in closed queueing networks. *Performance Evaluation*, 66(11):607–620, November 2009.

[G6] Thomas Begin, Alexandre Brandwajn, Bruno Baynat, Bernd Wolfinger, and Serge Fdida. High-level approach to modeling of observed system behavior. *Performance Evaluation*, 67(5):386–405, May 2010.

[G7] Alexandre Brandwajn and Thomas Begin. A recurrent solution of $Ph/M/c/N$-like and $Ph/M/c$-like queues. *Journal of Applied Probability*, 49(1):84–99, March 2012.

[G8] Alexandre Brandwajn and Thomas Begin. Reduced complexity in M/Ph/c/N queues. *Performance Evaluation*, 78:42–54, August 2014.

[G9] Alexandre Brandwajn and Thomas Begin. Breaking the dimensionality curse in multi-server queues. *Computers & Operations Research*, 78:141–149, September 2016.

[G10] Alexandre Brandwajn and Thomas Begin. Multi-server preemptive priority queue with general arrivals and service times. *Performance Evaluation*, 115:150–164, 2017.

[G11] Alexandre Brandwajn and Thomas Begin. First-Come-First-Served Queues with Multiple Servers and Customer Classes. *Performance Evaluation*, 2018.

[G12] Alexandre Brandwajn, Thomas Begin, Hind Castel-Taleb, and Tulin Atmaca. A study of systems with multiple operating levels, probabilistic thresholds and hysteresis. *IEEE Transactions on Parallel and Distributed Systems*, 29(4):748–757, April 2018.

[G13] Paulo Gonçalves, Shubabrata Roy, Thomas Begin, and Patrick Loiseau. Dynamic resource management in clouds: A probabilistic approach. *IEICE Transactions on Communications, special session on Networking Technologies for Cloud Services*, 95(8):2522–2529, 2012. Invited paper.

[G14] Hossein Soleimani, Thomas Begin, and Azzedine Boukerche. Safety message generation rate adaptation in lte-based vehicular networks. *Computer Networks*, 128:186–196, 2017.

[G15] Marija Stojanova, Thomas Begin, and Anthony Busson. Conflict graph-based model for IEEE 802.11 networks: A Divide-and-Conquer approach. *Performance Evaluation*, 2018.

Articles in Proceedings of International Conferences with Program Committee

[H1] Thiago Abreu, Bruno Baynat, Thomas Begin, and Isabelle Guérin Lassous. Hierarchical modeling of IEEE 802.11 multi-hop wireless networks. In *Proceedings of the 16th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM MSWIM'13*, pages 143–150, Barcelona, Spain, November 2013.

[H2] Thiago Abreu, Bruno Baynat, Thomas Begin, Isabelle Guérin Lassous, and Huu-Nghi Nguyen. Modeling of IEEE 802.11 multi-hop wireless chains with hidden nodes. In *Proceedings of the 17th International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM MSWIM'14*, pages 159–162, Montréal, Canada, September 2014.

[H3] Thiago Abreu, Nghi Nguyen, Thomas Begin, Isabelle Guérin Lassous, and Bruno Baynat. Substitution Networks: Performance Collapse due to Overhead in Communication Times. In *Proceedings of the 4th International Conference on Ad Hoc Networks, AdhocNets'12*, pages 1–16, Paris, France, October 2012. Invited paper.

[H4] Doreid Ammar, Thomas Begin, and Isabelle Guérin Lassous. A new tool for generating realistic internet traffic in $NS-3$. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques, IEEE SIMUTools'11*, pages 81–83, Barcelona, Spain, March 2011.

[H5] Doreid Ammar, Thomas Begin, Isabelle Guérin Lassous, and Ludovic Noirie. Evaluation and comparison of MBAC solutions. In *Proceedings of the 36th Conference on Local Computer Networks, IEEE LCN'11*, pages 215–218, Bonn, Germany, October 2011.

[H6] Doreid Ammar, Thomas Begin, Isabelle Guérin Lassous, and Ludovic Noirie. KBAC: Knowledge-Based Admission Control. In *Proceedings of the 37th Conference on Local Computer Networks, IEEE LCN'12*, pages 537–544, Miami, Florida, October 2012.

[H7] Doreid Ammar, Thomas Begin, Isabelle Guérin Lassous, and Ludovic Noirie. Traffic-aware flow admission control. Demo at Alcatel Lucent, Open Days, May 2012.

[H8] Doreid Ammar, Julien Brochet, Thomas Begin, Isabelle Guérin Lassous, and Ludovic Noirie. Knowledge-Based Admission Control: A real-time performance analysis. Demo at the 37th Conference on Local Computer Networks, IEEE LCN 2012, October 2012.

[H9] Guillaume Artero Gallardo, Bruno Baynat, and Thomas Begin. Performance modeling of virtual switching systems. In *Proceedings of the 24th IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS'16*, pages 125–134, London, England, September 2016.

[H10] Thomas Begin and Azzedine Boukerche. A note on the causes degrading communication between RSUs and vehicles in overloaded conditions. In *Proceedings of the 13th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, ACM PE-WASUN'16*, pages 27–31, Valletta, Malta, November 2016. Invited paper.

[H11] Thomas Begin and Alexandre Brandwajn. A Tool for solving Ph/M/c and Ph/M/c/N queues. Proceedings of the 9th ACM International Conference on Quantitative Evaluation of SysTems, QEST12, September 2012.

[H12] Thomas Begin and Alexandre Brandwajn. A note on the accuracy of several existing approximations for M/Ph/m queues. In *Proceedings of the 4th IEEE International Workshop on High-Speed Network and Computing Environment, IEEE HSNCE'13*, pages 730–735, Kyoto, Japan, July 2013.

[H13] Thomas Begin and Alexandre Brandwajn. Predicting the system performance by combining calibrated performance models of its components - a preliminary study. In *Proceedings of the 7th ACM/SPEC International Conference on Performance Engineering, ICPE'16*, pages 95–100, Delft, Netherlands, March 2016.

[H14] Thomas Begin, Alexandre Brandwajn, Bruno Baynat, Bernd Wolfinger, and Serge Fdida. High-level approach to modeling observed system behavior. In *SIGMETRICS Performance Evaluation Review*, volume 35, pages 34–36, Cologne, Germany, October 2007. Presented as a Poster at Performance 2007.

[H15] Thomas Begin, Alexandre Brandwajn, Bruno Baynat, Bernd Wolfinger, and Serge Fdida. Towards an automatic modeling tool for observed system behavior. In *Proceedings of the 4th European Performance Engineering Workshop, EPEW'07*, pages 200–212, Berlin, Germany, September 2007. LNCS.

[H16] Thomas Begin, Anthony Busson, Isabelle Guérin Lassous, and Azzedine Boukerche. Video on Demand in IEEE 802.11p-based Vehicular Networks: Analysis and Dimensioning. In *Proceedings of the 21st International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, ACM MSWIM'18*, pages 1–8, Montréal, Canada, October 2018.

[H17] Alexandre Brandwajn and Thomas Begin. A note on aspects of workload characterization in parallel access volumes. In *Proceedings of 19th the Computer Measurement Group, CMG'09*, pages 1–6, Dallais, US, December 2009.

[H18] Alexandre Brandwajn and Thomas Begin. A note on the effects of service time distribution in the M/G/1 queue. In *Proceedings of the Standard Performance Evaluation Corporation Benchmark Workshop, SPEC'09*, pages 138–144, Austin, Texas, January 2009.

[H19] Alexandre Brandwajn and Thomas Begin. Preliminary results on a simple approach to $G/G/c$-like queues. In *Proceedings of the 16th International Conference on Analytical and Stochastic Modelling Techniques and Applications, ASMTA'09*, pages 159–173, Madrid, Spain, June 2009.

[H20] Alexandre Brandwajn and Thomas Begin. Performance evaluation of a single node with general arrivals and service. In *Proceedings of the 18th International Conference on Analytical and Stochastic Modelling Techniques and Applications, ASMTA'11*, pages 85–98, Venice, Italy, June 2011.

[H21] Alexandre Brandwajn and Thomas Begin. An approximate solution for $Ph/Ph/1$ and $Ph/Ph/1/N$ queues. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE'12*, pages 57–62, Boston, Massachusetts, April 2012.

[H22] Jean-Baptiste Delavoix, Shubabrata Roy, Thomas Begin, and Paulo Gonçalves. Demonstrating a Versatile Model for VoD Buzz Workload in a Large Scale Distributed Network. Demo at the 1st International Conference on Cloud Networking, IEEE CloudNet 2012, November 2012.

[H23] Sébastien Doirieux, Bruno Baynat, and Thomas Begin. On finding the right balance between fairness and efficiency in WiMAX scheduling through analytical modeling. In *Proceedings of the 17th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems, MASCOTS'09*, pages 1–10, London, England, September 2009.

[H24] Huu-Nghi Nguyen, Thomas Begin, Anthony Busson, and Isabelle Guérin Lassous. Approximating the end-to-end delay using local measurements: a preliminary study based on conditional expectation. In *Proceedings of the International Symposium on Networks, Computers and Communications, IEEE ISNCC'16*, pages 1–6, Hammamet, Tunisia, May 2016. Invited paper.

[H25] Huu-Nghi Nguyen, Thomas Begin, Anthony Busson, and Isabelle Guérin Lassous. Evaluation of an end-to-end delay estimation in the case of multiple flows in SDN networks. In *Proceedings of the 3rd International Workshop on Management of SDN and NFV Systems, ManSDN/NFV'16*, pages 336–341, Montréal, Canada, November 2016.

[H26] Huu-Nghi Nguyen, Thomas Begin, Anthony Busson, and Isabelle Guérin Lassous. Towards a passive measurement-based estimator for the standard deviation of the end-to-end delay. In *Proceedings of the Network Operations and Management Symposium, IEEE/IFIP NOMS'16*, pages 632–637, Istanbul, Turkey, April 2016.

[H27] Van Dan Nguyen, Thomas Begin, and Isabelle Guérin Lassous. Multi-constrained routing algorithm: a networking evaluation. In *Proceedings of the 4th IEEE International Workshop on High-Speed Network and Computing Environment, IEEE HSNCE'13*, pages 719–723, Kyoto, Japan, July 2013.

[H28] Tahiry Razafindralambo, Thomas Begin, Marcelo Dias De Amorim, Isabelle Guérin Lassous, Nathalie Mitton, and David Simplot-Ryl. Promoting quality of service in substitution networks with controlled mobility. In *Proceedings of the 10th International Conference on Ad Hoc Networks and Wireless, AdHocNow'11*, pages 248–261, Paderborn, Germany, July 2011.

[H29] Shubabrata Roy, Thomas Begin, and Paulo Gonçalves. A complete framework for modelling and generating workload volatility of a VoD system. In *Proceedings of the 9th International Wireless Communications & Mobile Computing Conference, IWCMC'13 - 4th International Workshop on TRaffic Analysis and Classification, IEEE TRAC'13*, pages 1168–1174, Cagliari, Italy, July 2013.

[H30] Marija Stojanova, Thomas Begin, and Anthony Busson. Conflict graph-based markovian model to estimate throughput in unsaturated ieee 802.11 networks. In *Proceedings of the 15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, IEEE/IFIP WiOpt'17*, pages 1–8, Paris, France, May 2017.

[H31] Zidong Su, Bruno Baynat, and Thomas Begin. A new model for dpdk-based virtual switches. In *Proceedings of the 3rd Conference on Network Softwarization, IEEE NETSOFT'17*, pages 1–5. Bologna, Italy, July 2017.

[H32] Zidong Su, Thomas Begin, and Bruno Baynat. Towards including batch services in models for dpdk-based virtual switches. In *Proceedings of the 9th Conference on Global Information Infrastructure and Networking Symposium, IEEE GIIS'18*, pages 37–44. Saint Pierre, France, October 2017.

Articles in Proceedings of National Conferences with Program Committee

[I1] Doreid Ammar, Thomas Begin, Isabelle Guérin Lassous, and Ludovic Noirie. Contrôles d'admission basés sur des mesures : Evaluation et comparaison de solutions. In *Proceedings of the 15th Colloque Francophone sur l'Ingénierie des Protocoles, CFIP'11*, pages 1–4, Sainte Maxime, France, May 2011. Hermès.

[I2] Doreid Ammar, Thomas Begin, Isabelle Guérin Lassous, and Ludovic Noirie. Contrôle d'admission basé sur un plan de connaissance. In *Proceedings of the 14th Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications, ALGOTEL'12*, pages 1–4, La Grande Motte, France, May 2012.

[I3] Thomas Begin, Bruno Baynat, Alexandre Brandwajn, Serge Fdida, Safia Kedad, and Francis Sourd. Génénaration automatique de modèles calibrés. In *Proceedings of the 12th Colloque Francophone sur l'Ingénierie des Protocoles, CFIP'06*, pages 74–86, Tozeur, Tunisia, November 2006. Hermès.

[I4] Thomas Begin and Alexandre Brandwajn. Note sur la simulation d'une file $M/G/1$ selon la distribution du temps de service. In *Proceedings of the 12th Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications, ALGOTEL'10*, pages 1–4, Belle Dune, France, June 2010.

[I5] Thomas Begin and Alexandre Brandwajn. Une solution approchée pour les files $Ph/Ph/1$ et $Ph/Ph/1/N$. In *Proceedings of the 13th Rencontres Francophones sur les Aspects Algorithmiques de Télécommunications, ALGOTEL'11*, pages 1–4, Cap Estérel, France, Mai 2011.

[I6] Mehdi Bezahaf, Thomas Begin, Bruno Baynat, and Serge Fdida. Note sur les performances de TCP dans un environnement sans-fil multisaut. In *Proceedings of the 14th Colloque Francophone sur l'Ingénierie des Protocoles, CFIP'09*, pages 1–3, Strasbourg, France, October 2009.

[I7] Alexandre Brandwajn and Thomas Begin. Note sur les temps de service résiduels. In *Proceedings of the 13th Colloque Francophone sur l'Ingénierie des Protocoles, CFIP'08*, pages 6–18, Les Arcs, France, March 2008.

[I8] Shubabrata Roy, Thomas Begin, and Paulo Gonçalves. An MCMC procedure for calibrating a VoD workload model. In *Proceedings of the 24th colloque Gretsi , GRETSI'13*, pages 1–4, Cagliari, Italy, July 2013.