Preface

Welcome to UKPEW 2015 at the University of Leeds. This is the 31st edition of UKPEW, and the third time UKPEW has been hosted by Leeds; the last time this happened was in 2009 for UKPEW 25th Anniversary. Other previous locations of UKPEW were:

2014 Newcastle
2013 Loughborough
2012 Edinburgh
2011 Bradford
2010 Warwick
2009 Leeds
2008 Imperial College, London
2007 Edge Hill University
2006 Poole
2005 Newcastle
2004 Bradford
2003 Warwick
2002 Glasgow
2001 Leeds
2000 Durham
2019 Bristol
2018 Edinburgh
2017 Ilkley (Bradford University)
2016 Edinburgh
2015 Liverpool John Moores
2014 Edinburgh
2013 Loughborough
2012 Imperial College, London
2011 Edinburgh
2010 Bradford
2009 Warwick
2008 Newcastle
2007 Edinburgh (Heriot-Watt)
2006 Edinburgh
2005 Edinburgh
2004 Bradford
2003 Edinburgh
2002 Bradford
2001 Edinburgh
2000 Edinburgh
2000 Leeds
2000 1st UKPEW, Edinburgh

UKPEW continues its mission as the leading UK forum for the presentation of all aspects of performance modelling and analysis of computer and communication systems. This year’s proceedings include 11 papers from various UK institutions including Bradford, Newcastle, and of course Leeds. We welcome back our colleagues from Germany, Iraq, Brazil, and Japan. The topics include Wireless Networks, Performance Modelling Techniques, Scheduling, energy efficiency, and Cloud Computing. Special thanks go to the referees who very kindly agreed to look through all the original submissions, as well as the programme committee consisting of Richard Kavanagh and Django Armstrong. Also, we would like to thank Judi Drew who helped with the local organisation.

We thank the steering committee of UKPEW who gave us the opportunity to hold this special event at Leeds this year:

Irfan Awan (Bradford)  Stephen Jarvis (Warwick)
Jeremy Bradley (Imperial)  Rob Pooley (Heriot-Watt)
Stephen Gilmore (Edinburgh)  Nigel Thomas (Newcastle)

We also thank the School of Computing at Leeds for hosting the event. We hope you will all enjoy this year’s UKPEW and encourage you to use this event for the free exchange of novel ideas and results in the evolving field of performance engineering.
Leeds, September 2015
The Workshop Chair & Programme Committee
Contents

1 Improving MCT Scheduling Algorithm to Reduce the Makespan and Cost of Workflow Execution in the Cloud by F. Llwah, N. Thomas and J. Cala  

2 PEPS2015 - Stochastic Automata Networks Software Tool by P. Fernandes, A. Sales and L. Zani  

3 Load Balancing and Device Dropout in Open Clusters of Heterogeneous M/M/m Queues by D. Thomas  

4 Performance analysis of the Trusted Cloud Computing Platform by S.N.S. Kamil and N. Thomas  

5 Large-Scale Data Center with Setup Time and Impatient Customer by T. Phung-Duc  

6 Resource Allocation in Cloud Computing Environment by A. Sheikh and N. Thomas  

7 Universal Performance Assessment of Cross-Platform Mobile Applications by A. Nitze and A. Schmietendorf  

8 Evaluation of Automated Static Analysis Tools for Malware Detection in Portable Executable files by A.P. Namanya, J.F. Pagnadisso and I. Awan  

9 Controllable Setup Queue for Energy-Aware server by T. Phung-Duc  


11 Service-based Inter domain Adaptive Autonomic Management architecture by A.K. Ramli and K. Djemame
Research Papers
Improving MCT scheduling algorithm
to reduce the makespan and cost of
workflow execution in the cloud

Faris Llwaah, Nigel Thomas, Jacek Cala*

Abstract

Cloud computing has become a modern paradigm that delivers IT resources as a service over the internet on the pay-per-use basis. Although scientific workflow applications can benefit from resources provided by cloud computing, the challenge is to find scheduling algorithms that can optimise workflow execution. This paper proposes MCT+, an extension to the Minimum Completion Time (MCT) algorithm that can reduce the makespan and cost of workflow execution. We evaluate our algorithm using WorkflowSim and two workflows from the Pegasus workflow management system. Our results show that MCT+ can improve MCT by about 35%.

Keywords: Cloud Computing, Workflow, Scheduling, Execution time, Cost

1 Introduction

Over the recent years cloud computing has proved to be an effective means to deliver IT resources for both commercial and scientific users. It also has played an increasingly important role in running scientific workflow applications. Using the cloud scientists can easily manage the amount of resources they want to use to run their analyses. However, when compared with HPC, in the cloud much more important is the effectiveness and scalability of users’ applications. The direct cost model of the cloud gives its users very clear message about how much they have spent to run their applications. Therefore, the benefit of virtually unlimited resources offered by the cloud can be guaranteed only if applications can effectively use them. Otherwise, using more resources merely generates more cost with little gains in the execution time.

One way to improve resource utilisation is to design scheduling algorithms that can find effective mapping of workflow tasks onto cloud resources. Following [1], we define workflow scheduling as a process that distributes the tasks in a workflow for execution to a suitable computing resource while observing the dependencies between tasks. The overall goal of scheduling is to optimise certain QoS criteria such as minimising the total task execution time (makespan) and/or minimising execution cost.

*School of Computing Science, Newcastle University, {f.llwaah, nigel.thomas, jacek.cala}@ncl.ac.uk
In this paper we show that a simple extension of the Minimum Completion Time (MCT) scheduling algorithm can generate a substantial benefit by reducing the makespan and cost of a workflow running in the cloud. This work is the initial stage in addressing more general optimisation issues with conflicting criteria, such as cost and reliability.

The remainder of this paper is structured as follows: the next section presents the representation of scientific workflow. Section 3 includes the background and related work. Section 4 presents the proposed extension which is followed by experimental and experimental results. Finally, conclusions and future work is presented in Section 6.

2 Scientific Workflow Representation

Workflow can be defined as a collection of atomic tasks which follow a specific order of processing to achieve a desired goal. Often, a workflow is modelled as directed acyclic graph \( \text{DAG} = (V, A) \), where set of vertices \( V = \{T_1, T_2, \ldots, T_n\} \) represents the tasks in the workflow and set of arcs \( A \) represents data dependencies between these tasks [2].

In the cloud each task of the workflow can be realised by one or more service instances. We denote a set of available service instances for task \( T_i \) as \( S_{im} = \{S_1^i, S_2^i, \ldots, S_m^i\} \). Then, for each service instance \( S_j^i \) running task \( T_i \) we consider three types of QoS parameters: its cost \( c_j^i \), execution time \( t_j^i \), and reliability \( r_j^i \) [3]. Our goal is to optimise the performance of the workflow in the cloud system by providing the user a way to express constraints to the above three QoS parameters, namely: deadline, budget and overall workflow reliability. Additionally, the user may want to specify optimisation objectives such as minimisation of makespan and cost, and maximisation of reliability. That leads to challenging workflow scheduling problems due to the trade off between these parameters.

3 Background and related work

Generally, the problem of scheduling tasks in a distributed execution environment is known to be NP-hard [4]. It means there are no algorithms that can generate the optimal solution in polynomial time and these which run in polynomial time can only produce approximate results. In this work we used one of the approximate algorithms: Minimum Completion Time.

3.1 MCT scheduling algorithm

MCT is an approach to assign a task to the resource with the minimum expected completion time for that task and in arbitrary order [5]. While simple to implement, the algorithm may result in some tasks being assigned to resources that do not have minimum execution time for them [6]. Therefore, we wanted to investigate how MCT can be improved to reduce the overall makespan and cost of the workflow execution.
3.2 Related Work

The scheduling of workflow applications onto cloud computing, or distributed execution environment, is a hard problem that has been studied extensively in the literature, see e.g. [7–9].

Yu & Buyya [10] presented the Min-Min algorithm to minimize the overall job execution time of the workflow application. But this algorithm does not take the processor speed (MIPS) into account. Our preliminary experiments ran in WorkflowSim showed that MCT+ outperforms the Min-Min algorithm.

Tsai et al. [11] proposed the hyper-heuristic scheduling algorithm (HHSA), a scheduling solution for cloud computing systems. The HHSA algorithm provides a significant reduction of the makespan when compared with other scheduling algorithms evaluated. In addition, HHSA has been implemented in cloud simulator (CloudSim [12]) and in real system (Hadoop [13]). The simulation included in the paper by Tsai et al. shows that HHSA is less effective in reducing makespan than Min-Min.

The Period ACO based scheduling algorithm (PACO) investigated by Sun et al. [14] is based on an Ant Colony Optimization algorithm. The simulation experiment results showed in the paper suggest that PACO is better than the Min-Min algorithm generating schedules with about 20% shorter makespan. Our MCT+ extension is better than Min-Min for about 25%.

Topcuoglu et al. [2] proposed the Heterogeneous Earliest Finish Time algorithm (HEFT). This algorithm finds the average of both the execution time for each task and communication time between the resources of two tasks. Then a rank function is used to order the tasks in the workflow, so that a task with higher rank value is given higher priority. In the phase of resource selection tasks are scheduled depending on their priorities where each task is assigned to the resource to obtain a minimum execution time. In contrast to proposed algorithm, the comparison experiment proves outperforms MCT+ algorithm in reducing the execution time.

4 Improving MCT

The basic MCT algorithm assigns tasks to VMs such that the first, arbitrary task is scheduled to the VM with the highest speed, the next task is assigned to the fastest VM in the remaining pool and so on until all tasks are assigned or all VMs have been allocated a task. If there are more tasks to allocate, the algorithm schedules the remaining tasks when one or more VMs become available.

Whilst MCT focuses only on the speed of VMs, our extension MCT+ takes into account also the size of the tasks, so that tasks with MCT+ are scheduled in order, from the largest to smallest. Therefore, the MCT+ algorithm schedules the ordered list of tasks onto the ordered list of VMs such that the longest task is scheduled to the fastest VM, second longest task is scheduled to the second fastest VM, and so on.

Figure 1 shows how the tasks of the Montage workflow are assigned to VMs by MCT+ and MCT alone.
5 Scheduling experiment

In our experiment we evaluated the the original MCT algorithm and our extension MCT+ to schedule selected workflow applications. The experiment was conducted in the WorkflowSim 1.1.0 framework [5] which is an extension of the CloudSim environment [12]. WorkflowSim can simulate workflow execution in a cloud simulated by CloudSim.

In our experiment the CloudSim framework consists of one data center with 5–15 VMs. Each VM has 512 MB of RAM and one CPU. The speed of VMs is assigned randomly from range 1000–2500. The relevant configuration parameters of the environment and related costs are included in Table 1.

Table 1: Configuration parameters and cost of resources used in the evaluation experiment.

<table>
<thead>
<tr>
<th>Configuration parameters</th>
<th>VM resource costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM number</td>
<td>CPU 3.0</td>
</tr>
<tr>
<td>CPU MIPS</td>
<td>Memory 0.0</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Bandwidth 0.0</td>
</tr>
<tr>
<td></td>
<td>Storage 0.0</td>
</tr>
</tbody>
</table>

As the workflow application, we use two scientific workflows available in WorkflowSim – Montage and CyberShake [15]. The Montage workflow is an astronomy application used to generate custom mosaics of the sky based on a set of input images; it contains 25 tasks. The CyberShake workflow is used to characterize earthquake hazards by generating synthetic seismograms. This workflow contains 30 tasks. Figure 2 and 3 present the structure of both workflows. Different colors denote different phases of the workflows; we run workflows with the clustering parameter turned on, which means that there is a synchronisation barrier between workflow phases. The figures show also the expected runtime of each task as defined in the XML workflow definition file.
Figure 2: The structure of the Montage workflow.

Figure 3: The structure of the CyberShake workflow.
To calculate the makespan of the workflows we used the following formulas:

\[
\text{makespan} = \text{Finish}(T_{last}) - \text{Start}(T_{first})
\]

where \(\text{Finish}(T_{last})\) and \(\text{Start}(T_{first})\) are functions that return completion time of the last task and start time of the first task scheduled, respectively. Then to calculate the total cost of the workflow execution \(C\) we used the following:

\[
c_j^i = \text{Cost}_{CPU}^j + \text{Cost}_{DataIn}^j + \text{Cost}_{DataOut}^j
\]

\[
C = \sum_{i=1}^{n} c_j^i
\]

where \(j\) is the number of the VM which ran task \(T_i\).

Figure 4a and b show the reduction in makespan and cost for the Montage workflow, whereas Figure 5a and b show the gains for the CyberShake workflow. In both cases we ran 10 tests for each data point and for the largest setup with 15 VMs the MCT algorithm is improved for about 35 ±1%.

6 Conclusion and Future Work

In this paper we presented an extension of the MCT scheduling algorithms and evaluated it using the WorkflowSim platform. The results show that MCT+, despite still very simple, can significantly reduce both makespan and cost when compared to the original MCT algorithm. This work allowed us also to investigate various scheduling policies available in a simulated environment; WorkflowSim proved to be very useful for this purpose.

In the future we would like to identify an efficient workflow scheduling algorithm that minimizes both the execution time and cost of workflow while meeting a specified reliability requirement. Additionally, we plan to extend WorkflowSim to allow us to simulate workflows designed in e-Science Central (e-SC) [16], our in-house built workflow management system. We expect that ability to simulate e-SC workflows will help us to select a scheduling algorithm that can best match the system.
Figure 5: Makespan (a) and total cost (b) of the CyberShake workflow scheduled with MCT and MCT+.

References


PEPS2015 - Stochastic Automata Networks
Software Tool

Paulo Fernandes¹, Afonso Sales¹, and Luis Zani¹

{paulo.fernandes, afonso.sales}@pucrs.br, luis.zani@acad.pucrs.br
Computer Science Department
PUCRS University - Porto Alegre - BRAZIL

Abstract. This paper presents the new version of PEPS software tool, designed for solving models expressed using Stochastic Automata Networks (SAN). The SAN formalism is historically the first Markovian structured formalism employing Tensor (Kronecker) Algebra. PEPS tool is a live project, therefore a short timeline of its previous versions, as well as the new features included in 2015 version, are presented, namely: compact and efficient reachable state space storage, efficient and powerful just-in-time function evaluation, optimized numerical solutions (indirect methods), and symbolic solution (direct solution). We also mention the tool capabilities to solve very large models.

1 Introduction

The Stochastic Automata Networks (SAN) formalism is a structured stochastic Markovian formalism proposed by Plateau [14]. It is defined as an analytical method of modeling systems and it can be applied in different areas e.g. software engineering, reliability, parallel programming, concurrent systems, and performance evaluation of diverse systems. It allows us to represent an entire system by splitting it into a set of subsystems, each with an independent behavior itself (local events), and also eventual interdependencies among each other (synchronizing events and functional rates). In that sense, the framework first proposed by Plateau establishes a structured and modular way to describe discrete and continuous-time Markovian models.

The PEPS project started in the late 80’s and its goal was to develop a software package capable of computing numerical solutions for the SAN formalism. The first version of the tool was presented in [13] and featured a basic vector-matrix multiplication, where the matrix columns were generated one by one in each iteration. Only the tensor formula of the Kronecker descriptor was stored, and the full matrix was never generated. Later on, another three official versions of the software tool were published.

Among several areas to which PEPS tool may be applied, it is convenient to cite the areas of computing and communication performance modeling, parallel and distributed systems, and finite capacity queueing networks. PEPS differs from other tools supporting extended automata, such as UPPAAL [17] and
KRONOS [18], in regards to scope, as the mentioned tools are more focused on modeling, validation and verification of real-time systems\(^1\).

Fig. 1. Overview of PEPS 2015 modules and data

The core contribution of this paper, in regards to PEPS latest version, are essentially a compact and efficient reachable state space storage, efficient function evaluation, and optimized numerical solutions and symbolic solutions. These features, as well as the techniques and algorithms used in the tool, such as application of Multi-Valued Decision Diagrams (MDD), Tensor Algebra, Shuffle and Split algorithms, and symbolic solution, are shown in Fig. 1.

This paper is organized as follows: Section I introduces the idea proposed in this paper. Section II contextualizes the reader with a brief explanation of Stochastic Automata Networks. Section III presents pre-existing features that have been implemented in PEPS previous versions so far. Section IV describes the new features proposed in this paper for the latest version of the tool, which is under development in 2015. Section V presents a comparison of PEPS current version with the previous ones. Section VI shows the usage of the tool itself along with some examples. Finally, in Section VII, the conclusion summarizes this paper contribution and points out future works.

\(^1\) Real-Time systems use networks of timed automata to model tasks that must be performed within strict time deadlines.
2 Stochastic Automata Networks

A SAN model can be seen as a collection of individual stochastic automata that operate almost independently of each other in order to represent different modules of a system. Each individual automaton consists of a number of states and rules that dictate the way it moves from one state to another. The state of the SAN is given by the current state of each of its component automata [14]. There can be any number of automata in a SAN model. In fact, if the model contains only one automaton, that would be merely a Markov Chain [9].

For example, the SAN model in Fig. 2 describes a Client-Server system as a composition of two subsystems. The first one, named Client, possesses 4 local states: Idle, Transmitting, Receiving, and Working; and the second one, called Server, has three local states: Idle, Transmitting, and Receiving, in order to represent the basic functioning of the respective components. There are transitions between states called local events, for they only change the state of one automaton at a time, without affecting any other. In the described model, the local events are proc, for “process”, more, for “more data to be transmitted”, no_more, for “nothing else to be transmitted”, and wait, for “wait for requests”. There are also events which are called synchronizing events, which occur simultaneously in more than one automaton, making all of the affected automata change their states at the same time. In this case, the synchronizing events are req, for “request”, and resp, for response. Note that they change states from “Transmitting” to “Receiving” on the client side, and vice-versa on the server side with only one occurrence.

![Fig. 2. Example of a SAN model](image-url)
Additionally, the functional transition rates determine that the average firing rate of the events is not a constant value. In fact, there is a function (in this case, $F1$) that determines the possible values for this rate, depending on the state of other automata. In the example depicted in Fig. 2, the functional transition rate $F1$ states that and the Server only changes states from “Transmitting” to “Idle” when the Client is at the state “Working”, meaning that while there is communication between the components, the server cannot go idle. The reader interested in further information about this subject may find more detailed material in [11].

3 Pre-existent Features

The previous version of PEPS software tool, PEPS2007 [2], contains features that have been developed and improved along the years, since the beginning of the project. This section presents each of these features with more details.

3.1 Textual Input (.san)

PEPS software tool takes files with the extension .san as an input. These files contain the detailed description of every component of a SAN model that is to be numerically resolved by the tool. The specification of a SAN is basically composed of five sections: identifiers, events, partial reachability, network, and results. An example of a .san file that represents the model depicted in Fig. 2 is presented on Section 5.3 of this paper.

Identifiers: The first section of a .san file is where the average firing rates for all the events in the model are defined. However, each rate is required to have a unique name, i.e., an identifier.

Events: In this section, all events in the model have to be described. Hence, for each event, its type (local or synchronizing) and name are specified, as well as which identifier its rate corresponds to. Each event rate is associated with the identifier for that specific rate or to a function that represents a functional transition rate.

Partial Reachability: This is the section where the starting state of each automaton is defined. Since not all global states are reachable, the combination of the starting states is specified to be surely reachable. In other words, it is guaranteed that at least this global state is known to be a reachable state.

Since it is a Markovian Formalism, SAN assumes that all rates represent the average of an exponential distribution. Hence, a constant rate stands for the average frequency of an exponentially distributed phenomenon [9].

---

---
Network: After defining the partial reachability of the model, all of the automata must be described. In this section of the file, the names, states, and the transitions along with its corresponding firing rates are defined for each automaton.

Results: The software tool resolves the SAN model given as input by means of numerical solution algorithms described in [5, 7, 8]. The output of the system is the probability of the states configuration to happen throughout the time. Hence, this section is where the states (either local or global) that one wants to know the probability of the model to be at are specified.

3.2 Automata Aggregation

Automata aggregation in SAN is meant to reduce the number of automata in the model, bringing some numerical benefits to the solution. Among these benefits, it is important to point out the theoretical and practical advantage, which is the decrease of total state space and reduced memory usage, respectively.

Alongside with the reduction of the total state space, additional benefits consequently come up, such as the elimination of some functional rates and unreachable states. It occurs because of the nature of the aggregation methods, which basically groups automata that are connected to the same state(s) into only one automaton.

There are essentially two aggregation methods for SAN formalism. The algebraic aggregation method uses properties of the generalized tensor algebra to reduce the number of automata. The semantic aggregation method is based on the relationship among replicated automata [1].

3.3 Just-in-Time Functions

PEPS also implements a just-in-time function evaluation, which basically creates individual files containing C++ coded functions for each functional transition rate defined in the SAN model. Then, the C++ function codes are compiled and linked with PEPS in compile time, in such a way that they can be called whenever they are needed, i.e., when a function is evaluated in runtime.

3.4 Stationary and Transient Numerical Solutions (Shuffle Algorithm)

The Vector-Descriptor product is one of the most important operations to achieve both stationary and transient solutions for models described by Kronecker structured formalisms using iterative methods. The algorithm that is usually used to perform this operation is called the Shuffle algorithm [8].

A thorough study about matrices reordering and generalized tensor algebra properties aiming to optimize the evaluation of functional rates is given in the literature [8].
The Shuffle algorithm basically consists in exploiting the property of decomposing a tensor product into the ordinary product of tensor products in a normal form.

In other words, the Shuffle algorithm consists in the successive multiplication of a vector by each normal factor. In fact, vector $v$ is multiplied by the normal factor, then the resulting vector is multiplied by the normal factor again, and so on until the last factor. The multiplication of a vector $v$ by the $i^{th}$ normal factor is equivalent to shuffle the elements of $v$ in order to assemble $\text{nleft}_i \times \text{nright}_i$ vectors of size $n_i$ and multiply them by matrix $Q^{(i)}$. So, assuming the matrix $Q^{(i)}$ is stored as a sparse matrix, the number of multiplications necessary to multiply a vector by the $i^{th}$ normal factor is:

$$n\text{left}_i \times n\text{right}_i \times n_{z_i}$$

(1)

where $n_{z_i}$ is the number of nonzero elements of the $i^{th}$ matrix of the tensor product term $Q^{(i)}$. Considering the number of multiplications to all normal factors of a tensor product term, the result is [5,8]:

$$\prod_{i=1}^{N} n_i \times \sum_{i=1}^{N} \frac{n_{z_i}}{n_j}$$

(2)

The reader may find extensive material in the literature [5] in regards to memory and CPU efficiency of the Shuffle algorithm.

### 3.5 Integration Functions

The final steps of the Shuffle algorithm calculate the probability of each global state to happen, as mentioned previously. In that sense, one of PEPS implemented features is the integration functions. The main idea of these functions is to sum the product of each row of the probability vector containing all the global states. Since not all global states will necessarily be reachable, some of these probabilities will be zero.

### 4 PEPS 2015 - New Features

The latest version of the software tool, PEPS2015, contains some additional features, aiming to achieve a better performance and memory-space usage. A more detailed explanation of each of these features can be found hereafter.

#### 4.1 C-like Functions

One of PEPS pre-existent features is the already mentioned *just-in-time* function evaluation, that creates separate files containing C++ code for the functions defined in the SAN file. With the goal of increasing performance of the application, this created C++ code is meant to be replaced by a new way of interpreting the functions described in the model.
The intention here is to change the way the functions are defined in the SAN file. Given that currently the compiler creates individual files with C++ coded functions to be linked with PEPS, it seems more reasonable to let the SAN specification be more of a C-like function. This way, instead of creating separate files with C++ code in compile time and then calling the created function codes whenever it is necessary in runtime, the compiler will be able to interpret proper C++ code that was specified directly in the SAN model, all in compile time. Thus, performance improvements are expected since the application will be able to run function evaluation without the need of accessing any side resource because the entire process is done beforehand.

4.2 Reachable State Space Efficient Generation

A Multi-valued Decision Diagram (MDD) [10] is a compact structure that allows us to store and to manipulate large sets of structured information in a compact format. The information in a model can be structured by $N$ components (or subsystems, i.e., in our case, automata) where these components have some independent behavior and occasional interdependency.

Basically a MDD is a directed acyclic edge-labeled multi-graph, where there are some specific properties, such as: (i) nodes are organized into $N+1$ levels, where $N$ represents the number of subsystems; (ii) level $N$ has only one single non-terminal node (known as the root), whereas levels $N-1$ through 1 have one or more non-terminal nodes; (iii) level 0 has two terminal nodes: 0 and 1; (iv) a non-terminal node $p$ at level $l$ contains $n_l$ arcs pointing to nodes at level $l-1$, where $n_l$ indicates the number of local states of $l$-th subsystem; (v) there are not duplicate nodes, i.e., nodes at a same level are unique.

Fig. 3 illustrates a MDD which represents the state space $S$, subset of a cross-product of a system splitted in four subsystems (i.e., $N = 4$), where $S^{(i)}$ represents the local state space of the $i$-th subsystem.

In Fig. 3, non-terminal nodes are depicted by circles and terminal nodes are depicted by squares. A given state $x$ of the system is composed by the combination of the local states of each subsystem, i.e., $x = x^{(N)} \ldots x^{(1)}$ of $S$ is element of a subset represented by a $N$-level MDD if and only if the path through the MDD, starting at the level-$N$ node, following downward pointer $x^{(l)}$ at level $l$, reaches terminal node 1. Dashed arcs and nodes are paths which lead only to terminal node 0 and in the rest of the paper, for reasons of clarity, will be omitted.

Also, MDD can be used to represent a set $S$ of integer tuples by storing the characteristic function $f_S$ of the set. Therefore sets can be manipulated using operations over MDDs on their characteristic functions (e.g., the operation union on sets corresponds to disjunction on MDDs).

Saturation-based state-space generation is a successful symbolic method based on MDDs applied to generating state spaces of structured models, e.g., Stochastic Petri Nets (SPN) models [3, 4, 12].

Sales and Plateau [15] presented an extension of the saturation method described in [4], which allows the use of functional transitions, i.e., the interaction
The main idea of the generation method is to compute the reachable space state (RSS) of the model by the successive firing of events from an initial state while the RSS is stored in a MDD. The method exploits the possibility of firing any event that affects a given MDD node and its descendant nodes, thus bringing the node to its saturated format. Besides, nodes are considered in a bottom-up fashion (i.e., when a node is computed, all its descendant nodes are already in the saturated format). A node is considered as saturated if it encodes a set of states which are a fixed point in regards to the firing of any event at its level or at a lower level.

Fig. 4 shows a small SAN model with three automata \( (N = 3) \) and the corresponding RSS of this model represented by a MDD, generated from initial state 000.

Functions are a powerful feature in the SAN formalism, since it allows us to represent very complex behaviors of a system in a very compact format. Moreover, the usage of MDDs to generate the RSS of a SAN model, which uses functions (having a small domain size), allows us to achieve the generation of large reachable state spaces with a small computational time, while keeping a low memory consumption.
4.3 Stationary and Transient Numerical Solutions (Split Algorithm)

SAN schemes that model real scenarios are naturally sparse: That’s because the tensor sum structures generally make the local portion of the descriptor very sparse. Also, the synchronizing events make the descriptor quite sparse, since they are mostly used to describe exceptional behaviors.

The Split algorithm [5] allows each tensor product of matrices to be partitioned into two different groups: one with more sparse matrices and the other with more nonzero elements. This way, the Sparse algorithm [7] could be applied to the first group generating Additive Unitary normal Factor (AUNF). An AUNF is composed of three elements: a scalar value obtained by multiplying one nonzero element of each matrix in the Sparse-like part by each other; an input slice, which is a part of the vector \( v \) identified by the line row coordinates \( i \) of the nonzero elements multiplied; and an output slice, as a part of the vector \( v \) identified by the column coordinates \( j \) of the multiplied elements [5]. Each of the generated AUNF must then be multiplied (tensor product) by the second group of matrices using the Shuffle algorithm, as Slice handles only the first matrix in this case.

Considering this concept, the goal is to split the tensor terms into two sets of matrices and deal with them in different ways, almost individually. Therefore, the Split algorithm is considered a generalization for the encompassing all possibilities from a pure Sparse approach until the Shuffle algorithm, once it follows the idea of breaking the system into distinct parts to be dealt with.

In regards of efficiency, a deeply detailed material may be found in the literature [5], explaining the memory and CPU efficiency of the Split algorithm, as well as Sparse and Slice algorithms. Additionally, some optimizations on the Split algorithm were proposed by Czekster, Fernandes, and Webber in [7], in case the reader is interested.
4.4 Symbolic Solution

The main idea of the proposed symbolic solution is to deal with the tensor representation of the infinitesimal generator by adding terms to perform Gauss-Jordan Elimination steps [9]. The concept of functional elements is highly important for our method to deal with particular cases of the tensor structure and keep the symbolic operations simple enough to be handled.

The first step for the proposed symbolic solution is to prepare the matrix by performing the Gauss-Jordan method to obtain the infinitesimal generator. Once this process is done, the tensor representation of the starting of the Gauss-Jordan method will be:

\[
Q^{(0)} = \bigoplus_{i=0}^{N-1} Q^{(i)} + \sum_{s \in S} \bigotimes_{i=0}^{N-1} Q^{(i)}_{s} + \sum_{s \in S} \bigotimes_{i=0}^{N-1} Q^{(i)}_{s} + \bigotimes_{i=0}^{N-1} Q^{(i)}_{lc}
\]

The second step is to perform the matrix triangularization, in which the elements below the diagonal are eliminated through the application of the Gauss-Jordan to each row. The row scaling operations correspond to include functional elements to perform the necessary scale operation to each product of the descriptor. Once the matrix triangularization is complete, it is computationally simpler to store the tensor structure and perform the final step: a backward substitution procedure to resolve the system of equations. More detailed information about the method itself is presented by Fernandes, Lopes and Yeralan in [9].

5 Practical Capabilities

Table 1 shows some numerical results considering Product Space State (PSS) and type of model that the software tool can handle within reasonable time and space limits for PEPS current and previous versions.

<table>
<thead>
<tr>
<th>PEPS version</th>
<th>Model</th>
<th>PSS</th>
<th>Memory Used</th>
<th>Time to Solve</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>Unreliable Production Lines [23,24]</td>
<td>725,594,112</td>
<td>918.18 MB</td>
<td>133 hours</td>
</tr>
</tbody>
</table>
Based on the presented data, it is interesting to notice that the number of states in the network seems to be the driving factor for the final solution time. For the latest version of the tool, the execution time is fairly reasonable, for it resolves a model with a significantly larger number of states (725 million), using an acceptable amount of memory (less than 1 GB).

6 Software Tool

In order to make the way this tool works more explicit to the reader, this section presents an example of a .san file, as well as its compilation and execution processes. Regarding compatibility, the platforms that the software tool runs on are pointed out.

6.1 Environment

PEPS package has been tested and proven to be currently compatible with the following operating systems:

1. Linux - Ubuntu 14.04 64 bit
2. Mac OS X 10.8.2 - Mountain Lion
3. Mac OS X 10.9.2 - Mavericks
4. Mac OS X 10.10.2 - Yosemite

However, since the source code of PEPS is available upon request, the compilation and availability for other platforms is likely to be an easy task.

6.2 Example

The code shown in Code 1.1 is an example of a .san file, describing the SAN model depicted in Fig. 2, containing all components presented on Section 3.1 of this paper.

For this example, it was specified in the results section that the user expects to know the probability of the Client to be requesting, receiving, and processing. Also, the percentage of the time that the Client is transmitting data and the Server is receiving data and vice-versa. At last, the percentage of time that both Client and Server are idle.

6.3 Usage

An example of usage of the software tool is shown in this section. Both compilation and resolution of the .san file presented in the previous section can be seen in Fig. 5. For this demonstration, the compilation process does not make use of any aggregation method [1] and the resolution process uses the Power Method [11] to solve the model.

3 PEPS versions of 2003, 2007, and obviously 2015 also offer the possibility to perform stationary solutions using Arnoldi and GMRES methods, and also transient solution using Uniformization method [16] is available.
Code 1.1. Example of a SAN file

The compilation generates a report of produced intermediate files, and the time took to compile. In this toy example, the model was compiled in 0.2 milliseconds of user time. Analogously, the solution presents a report of how many iterations were necessary until achieving convergence (41 for this example), the time spent (0.05 milliseconds for this example) and the computation of all integration functions.

According to the output of PEPS 2015, depicted in the right hand side of Fig. 5, the probability of both Client and Server automata to be idle is less than fifteen percent, which means that, for these parameters, the model presents satisfying results in regards to occupancy, for the Server is far from overloaded.
Fig. 5. Compilation and resolution of the Client-Server SAN model

7 Conclusion

Fig. 1 summarizes the general vision of the PEPS 2015 software high level modules, as well as the data and information flow. In this figure is represented in the right hand side the input of PEPS 2015, a textual .san file describing an imagined SAN model, and the ultimate PEPS 2015 output the results computed by the application of integration functions over the computed probability vector, i.e., the quantitative estimations concerning the modeled reality. The three main modules of PEPS 2015 are the Compile, Solve and Integrate modules.

The Compile performing translation of a SAN model into a MDD description of the reachable state space, a tensor representation of transition matrix (descriptor), and the integration functions responsible to deliver the quantitative results. It comprises two specialized submodules, one responsible to generate the MDD representation of the reachable state space, and the other responsible to generate the just-in-time (jit) generated functions for the tensor representation and the integration functions.

The Solve module implements all solution iterative methods available (Power, Arnoldi, GMRES and Uniformization) offering Shuffle and Split vector-descriptor multiplications, but also the symbolic solution through Gauss-Jordan Elimination. In such way, this module is the more sensitive module in terms of computational efficiency, both in terms of CPU and memory usage.

Finally, the Integrate module is the simplest and more straightforward module, since it basically visits the reachable state space computing the integration functions and weighting the results by the computed probabilities.
PEPS software tool is a live project, hence there is ongoing work to add new features and keep improving it by seeking for even better techniques for vector-descriptor multiplication to speed up both transient and stationary numerical solutions, as well as more sophisticated structures for reachable state space manipulation. The new features proposed here led to a new version of PEPS as the main contribution of this paper. As new techniques are proposed in the future, they may result in even newer and optimized versions of PEPS tool. The interested researcher and practitioner may find more thorough information about PEPS in the webpage http://www.inf.pucrs.br/peg/peps/.

References


Load Balancing and Device Dropout in Open Clusters of Heterogeneous M/M/m Queues

David Thomas d.thomas926@btinternet.com

Abstract

This paper shows a simple method of calculating device Load up or Dropout points for open clusters of heterogeneous M/M/m queues. It extends the Balanced System (BS) method where the loads are balanced by ensuring that all the devices are equally utilised. This means that the device queue lengths are also equal and we show that the overall results are the same as if the devices were homogeneous, but with the same total capacity and arrival rate.

1 Introduction

There is an extensive literature on load sharing heterogeneous M/M/1 devices, (e.g. [1], [2], [3], [4], [7] and [8]). However, the emphasis has usually been on performance at high loadings, where device dropout may not improve overall response times. For Balanced Systems (BS) the utilisation of all (K) service centres is the same [3]. This means that the queue lengths are also equal [1]. This gives reasonable performance, obscuring the fact that improvements can be made by dropping out the slowest device(s) at low to medium loadings.

As the response times should be lower than Service Level Agreements (SLAs) at these loadings, the reduction in energy use to run (and cool) the servers may be more important than the performance improvements. Reduction in energy in IT is assuming greater importance. Nguyen et al [6] describe energy efficient dynamic server allocation for data centres and review the current literature for related work.

We will be dealing with open clusters of heterogeneous M/M/m queues, where ‘m’ is the same for all devices in the cluster. There is only need to calculate the queue length ($q_k$) at one device ‘k’. Multiplying this by the number of devices (K) gives the number (N) of jobs in the cluster. We can either start with the minimum number of the fastest devices where the capacity is greater than the arrival rate and increase the number until the number of jobs starts to increase. This is called the Load Up method. Alternatively, we could start with all the devices and drop out the slowest devices until the number of jobs starts to increase. This is called the Drop Out method.

The solutions are obtained in two stages. The first is to calculate the number of devices which gives the best overall results. For M/M/1 queues we show that it is possible to calculate bulk device dropouts. The second stage uses Little’s Law (2) below to calculate any required results such as the response time; both overall and at the individual devices. We extend the results from ‘m’ = 1 to ‘m’ = 2 and 4. The term heterogeneous means that the devices operate at different service rates, not that they are from different manufacturers or run under different operating systems.
2 Open Clusters

Open clusters are when work arrives via a source node, is sent to various devices via a load sharing algorithm and leaves via a sink node. In figure 1 we give an example of an open cluster of M/M/1 queues.

![Open cluster of M/M/1 queues](image)

The workload intensity is described by the transaction arrival rate (X) at the source node. For BS, the arrivals go to various nodes Q_1 to Q_k, where they receive service, such that their utilisation are equal as in the utilisation law (1) below.

\[ U_k = X_k \times S_k \text{ or } \frac{X_k}{\text{cap}_k} \]  

(1)

\( U_k \) is the utilisation, \( X_k \) is the arrival rate (or throughput), \( S_k \) is the service time per transaction and \( \text{cap}_k \) is the throughput capacity (or maximum service rate). \( S_k \) is the reciprocal of \( \text{cap}_k \). The terminology for this paper is taken from [3]. The authors state that “The utilisation Law is in fact a special case of Little’s Law”; the more general setting is shown below in (2).

\[ N = X \times R \]  

(2)

\( N \) is the number of jobs in the system and is the sum of the queue lengths at the various loaded devices. \( X \) is the overall arrival rate (or throughput) and \( R \) is the overall mean response time per job.

2.1 The basic model

In the first example the devices are modelled as M/M/1 queues, with Poisson arrivals and exponentially distributed service rates. The basic model has 5 devices with maximum service rates (or capacities), \( \text{cap}_1 = 100 \), \( \text{cap}_2 = 80 \), \( \text{cap}_3 = 60 \), \( \text{cap}_4 = 40 \) and \( \text{cap}_5 = 20 \). These are the reciprocals of the mean service times per arrival. The arrival rate (X) is 90 and as the total capacity (CAP) = 300. As we have added all the arrival rates and all the capacities, we have \( U = X / \text{CAP} \), we obtain the overall utilisation of...
0.3 as in (1). We will keep the service rates the same for the M/M/2 and M/M/4 examples, but the total capacities will be 600 and 1,200 respectively. In order to keep the utilisations the same (0.3), the arrival rates will be 180 and 360 respectively.

### 3 M/M/1 queues

#### 3.1 All devices (no dropout)

For the first example the loads are spread over all the devices, so that the utilisations are equal, ignoring the possibility of device dropout. As the utilisations are equal, the arrival rates at the individual devices are in the ratio of the capacities and sum to X. The individual arrival rates are \( X_k = X \times \frac{\text{cap}_k}{\text{CAP}} \). A well known result in queueing theory ([3], [4] and [5]) gives the response time for an open M/M/1 queue.

\[
R_k = \frac{S_k}{1 - U_k} \text{ or } \frac{1}{\text{cap}_k - X_k}
\]

(3)

\( R_k \) is the response time at \( Q_k \) and is the reciprocal of the remaining capacity (\( \text{cap}_k - X_k \)). Multiplying both sides by \( X_k \), from Little’s Law we obtain:

\[
q_k = \frac{U_k}{1 - U_k} \text{ or } \frac{X_k}{\text{cap}_k - X_k}
\]

(4)

The queue lengths per device are \( q_k \) and the results are in Table 1 below. Multiplying both sides by \( K \) we obtain (5) below.

\[
N = K \times \frac{X_k}{\text{cap}_k - X_k}
\]

(5)

If we multiply the numerator and denominator of the right hand side by \( \frac{\text{CAP}}{\text{cap}_k} \) we obtain:

\[
N = K \times \frac{X}{\text{CAP} - X}
\]

(6)

From Little’s Law we can write (6) as:

\[
R = \frac{K}{\text{CAP} - X}
\]

(7)

We can also see that the overall result is the same as if the devices were homogeneous as long as the total capacity is the same. We can obtain \( N \) (or \( R \)) without recourse to results at each of the individual devices. This also pertains to M/M/2 and M/M/4 devices below. We can thus obtain the overall results by treating the cluster as a single ‘composite’ device. The overall result for the example is \( N = 5 \times 90 / (300 - 90) = 15 / 7 = 2.1429 \) and \( R = 5 / 210 = 1 / 42 = 0.02381 \). In Table 1, for the individual devices the throughputs are calculated in proportion to the capacities, the queue lengths are \( N / K \) and the response times are calculated from Little’s Law.
Table 1 – M/M/1 Results for all devices

<table>
<thead>
<tr>
<th></th>
<th>Q_1</th>
<th>Q_2</th>
<th>Q_3</th>
<th>Q_4</th>
<th>Q_5</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>30</td>
<td>24</td>
<td>18</td>
<td>12</td>
<td>6</td>
<td>90</td>
</tr>
<tr>
<td>R</td>
<td>0.01429</td>
<td>0.01786</td>
<td>0.02381</td>
<td>0.03571</td>
<td>0.07143</td>
<td>0.02381</td>
</tr>
<tr>
<td>qL</td>
<td>0.4286</td>
<td>0.4286</td>
<td>0.4286</td>
<td>0.4286</td>
<td>0.4286</td>
<td>2.1429</td>
</tr>
</tbody>
</table>

3.2 Device dropout – stage 1

Table 2 – M/M/1 Results for K = 1 to 5

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>N</td>
<td>9</td>
<td>2</td>
<td>1.8</td>
<td>1.8947</td>
<td>2.1429</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>0.1</td>
<td>0.0222</td>
<td>0.02</td>
<td>0.02105</td>
<td>0.02381</td>
<td></td>
</tr>
<tr>
<td>qL</td>
<td>0.4286</td>
<td>0.4286</td>
<td>0.4286</td>
<td>0.4286</td>
<td>2.1429</td>
<td></td>
</tr>
</tbody>
</table>

3.3 Device dropout – stage 2

It can be seen from Table 2 that the best results are when the 3 fastest devices are loaded. The individual results for K = 3 are in Table 3.

Table 3 – M/M/1 Individual results for K = 3

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>37.5</td>
<td>30</td>
<td>22.5</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>0.016</td>
<td>0.02</td>
<td>0.0267</td>
<td>0.02</td>
<td></td>
</tr>
<tr>
<td>qL</td>
<td>0.6</td>
<td>0.6</td>
<td>0.6</td>
<td>1.8</td>
<td></td>
</tr>
</tbody>
</table>

3.4 Bulk drop-outs

All devices with a slower service time than the response time in Table 1 (i.e. Q_4 and Q_5) can be dropped out. Overall results for BS are the same as the User Equilibrium (UE) load sharing algorithm for M/M/1 queues [7], given the same number of loaded devices. The UE algorithm has equal response times for each device and from (3) the loaded devices have the same spare capacity and act as homogeneous devices. For BS, the overall cluster gives the same results as if the devices were homogeneous.

The easiest way to think of this for BS is by loading up devices in decreasing order of capacity as throughput increases. That is; starting with the fastest device and loading the next fastest when the response time reaches the service time of the second and so on. As cap_1 = 100 and cap_2 = 80, from equation 1 we see that when the throughput reaches 20 the spare capacity of device 1 is 100 – 20, which is the capacity of device 2. As they then have the same spare capacity any further increase in throughput goes to both in equal measure. This also means that, for BS, devices can be dropped out.
where their service time is less than the calculated response time. In Table 2 we show the results for $K = 1$ to $5$ which illustrates this. These show that the lowest number of jobs (and therefore the lowest response time) is for $K = 3$.

### 3.4.1 Mixed clusters (UE)

There will probably be few actual clusters where all the devices have different capacities. We extend the above example with several devices with a capacity of 60. This will not change the average capacity. At the point when $X = 60$, then $X_1 = 40$ and $X_2 = 20$. It is obvious that all devices with a capacity of 60 or more are loaded at this point as throughput increases, or dropped out as throughput decreases.

### 3.4.2 Mixed clusters (BS)

For BS we can check the total queue lengths for devices where the service rate = 60 i.e. for $K = 2, 3, 4$ etc. For $K = 2$ and $X = 60$ there is no load up for these devices. The utilisation is $60/180$ or $1/3$. From (3) we see that each device has a queue length of $(1/3)/(2/3)$ or $1/2$. This gives $N = 1$ as $K = 2$. If $K = 3$ the utilisation is $60/240$ or $1/4$. This gives a device queue length of $(1/4)/(3/4)$ or $1/3$. As $K = 3$ then $N = 1$ again. It can be easily seen that we can continue to do this for any number of devices which have a capacity of 60. As $N$ and $X$ are the same, $R$ is the same, from Little’s Law.

### 3.5 Clusters of clusters

There is a lot of interest in improving performance in this field, because of grid and cloud computing, with many academic conferences and journals where it is a hot topic. Several researchers (e.g. [2] and [8]) have looked at using BS because of its simplicity, robustness and good results, especially at high loadings. This is also a candidate for the above methods in order to solve even more simply and use (bulk) device dropout methods to improve performance and calculate the optimum number of loaded devices for BS or UE. Then, it will be possible to choose the more appropriate of these load balancing methods.

Table 4 shows some results from El-Zoghdy [2]. There are three clusters, with $4, 3$ and $5$ devices respectively, making $K = 12$. These include utilisations and response times; the latter calculated [2] using analytic ($R_a$) and simulation ($R_s$) methods. We note that by using $R = K / (CAP - X)$ (7), we are able to solve for response times simply e.g. $R = 12 / (1700 - 700) = 12 / 1000 = 0.012$.

#### Table 4 – Results for clusters of clusters

<table>
<thead>
<tr>
<th>$X$</th>
<th>$U$</th>
<th>$R_A$</th>
<th>$R_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>0.235924</td>
<td>0.009231</td>
<td>0.009431</td>
</tr>
<tr>
<td>700</td>
<td>0.411765</td>
<td>0.012000</td>
<td>0.012032</td>
</tr>
<tr>
<td>1600</td>
<td>0.941176</td>
<td>0.120000</td>
<td>0.119012</td>
</tr>
<tr>
<td>1690</td>
<td>0.994118</td>
<td>1.200000</td>
<td>1.201692</td>
</tr>
</tbody>
</table>

UKPEW 2015 – http://ukpew.org/
4 M/M/2 Model

4.1 Homogeneous devices

We double the arrival rates and capacities of the M/M/1 example, thus ensuring that the utilisations are the same. From Mitrani [5] we apply the following equation:

\[ q_l = 2 \cdot U_k / (1 - U_k^2) \]  \hspace{1cm} (8)

As \( \text{CAP} = 600 \) and \( X = 180 \), then the average for \( \text{cap}_k = 120 \), \( X_k = 36 \) and \( U_k = 0.3 \), which means that \( q_l = 0.65934 \) for all \( K = 5 \) devices and \( N = K \cdot q_l = 3.296703 \). From Little’s Law \( R = 0.018315 \).

4.2 Heterogeneous devices – no device dropout

The 5 devices have the following capacities: \( \text{cap}_1 = 200 \), \( \text{cap}_2 = 160 \), \( \text{cap}_3 = 120 \), \( \text{cap}_4 = 80 \) and \( \text{cap}_5 = 40 \). If we distribute the arrivals such that the utilisations are equal, then the overall results are as above and the individual results are as in Table 5.

Table 5 – M/M/2 Results for all devices

<table>
<thead>
<tr>
<th></th>
<th>( Q_1 )</th>
<th>( Q_2 )</th>
<th>( Q_3 )</th>
<th>( Q_4 )</th>
<th>( Q_5 )</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>60</td>
<td>48</td>
<td>36</td>
<td>24</td>
<td>12</td>
<td>180</td>
</tr>
<tr>
<td>( R )</td>
<td>0.010989</td>
<td>0.013736</td>
<td>0.018315</td>
<td>0.027473</td>
<td>0.054945</td>
<td>0.018315</td>
</tr>
<tr>
<td>( q_l )</td>
<td>0.659341</td>
<td>0.659341</td>
<td>0.659341</td>
<td>0.659341</td>
<td>0.659341</td>
<td>3.296703</td>
</tr>
</tbody>
</table>

4.3 Heterogeneous devices – device dropout

Table 6 – M/M/2 Results for \( K = 1 \) to 5

<table>
<thead>
<tr>
<th>( K )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( U_k )</td>
<td>0.9</td>
<td>0.5</td>
<td>0.375</td>
<td>0.321429</td>
<td>0.3</td>
</tr>
<tr>
<td>( q_l )</td>
<td>9.473684</td>
<td>1.333333</td>
<td>0.872727</td>
<td>0.716927</td>
<td>0.659341</td>
</tr>
<tr>
<td>( N )</td>
<td>9.473684</td>
<td>2.666667</td>
<td>2.618182</td>
<td>2.86771</td>
<td>3.296703</td>
</tr>
<tr>
<td>( R )</td>
<td>0.052632</td>
<td>0.014815</td>
<td>0.014545</td>
<td>0.015932</td>
<td>0.018315</td>
</tr>
</tbody>
</table>

Table 6 above shows the results for \( K = 1 \) to 5, while Table 7 shows the individual results for \( K = 3 \).

Table 7 – M/M/2 individual results for \( K = 3 \)

<table>
<thead>
<tr>
<th></th>
<th>( Q_1 )</th>
<th>( Q_2 )</th>
<th>( Q_3 )</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>75</td>
<td>60</td>
<td>45</td>
<td>180</td>
</tr>
<tr>
<td>( R )</td>
<td>0.011636</td>
<td>0.014545</td>
<td>0.019394</td>
<td>0.014545</td>
</tr>
<tr>
<td>( q_l )</td>
<td>0.872727</td>
<td>0.872727</td>
<td>0.872727</td>
<td>2.618182</td>
</tr>
</tbody>
</table>
It can be shown that the results for BS are superior to UE for balancing loads over heterogeneous M/M/2 queues for the same number of devices (or the same result for K = 1). You can drop out all devices where the overall response time is less than the service time and produce a better result using BS than UE.

Once you have done that it may still be the case that you can load more devices and produce better response time results. For UE the Load up point for the second fastest device is when R = S₂ = 0.0125. This happens when X = 89.442719, U₁ = 0.4472136 and q₁ = 1.118034. If we spread these arrivals over Q₁ and Q₂ such that the utilisations are equal (0.248452), then q₁ = q₂ = 0.529595. N = 1.05919 and R = 0.011842. However, energy conservation may be more important than performance.

5 M/M/4 Model

5.1 Without device dropout

The M/M/4 example model has double the capacities and arrival rates of the M/M/2 devices. The 5 devices have maximum service rates (or capacities) of cap₁ = 400, cap₂ = 320, cap₃ = 240, cap₄ = 160 and cap₅ = 80 respectively. The transaction arrival rate (X) is 360 and the total capacity (CAP) is 1,200 giving an utilisation of 0.3. The equations for M/M/m queues become more complex when m > 3 [5]. For our M/M/4 example we use a version of the generic method below to calculate the queue lengths.

5.2 Step 1

Calculate un-normalised queue length probabilities. This sets the probability of the queue length of zero to 1 and calculates the probabilities of qₙ - m to be p(m-1) * X / cap(m).

5.2.1

Set capacities for m = 1...m. For this example, these are 300, 600, 900 and 1,200 respectively.

Set p(0) = 1, then p(1) = 1 * 360/300 = 1.2
p(2) = 1.2 * 360/600 = 0.72
p(3) = 0.72 * 360/900 = 0.288
p(4) = 0.288 * 360/1200 = 0.0864

Calculate p(>=4) as p(4) / (1 – U) = 0.0864 / 0.7 = 0.123429

5.2.2 Step 2

Calculate the normalising constant ‘c’ and true probability p(4)

\[ c = 1 + 1.2 + 0.72 + 0.288 + 0.0123429 = 3.331429 \]
p(4) = 0.0864 / 3.31429 = 0.0259348

5.2.3 Step 3

Calculate numbers being served from m * U and numbers waiting in queue from p(4) * X / (CAP * (1 – U)²).

Numbers being served = 4 * 0.3 = 1.2.

Numbers waiting to be served = 0.0259348 * 360 / (1200 * 0.49) = 0.0158784. The total number is 1.2158784. This is true for all the devices and so N = 5 * 1.2158784 = 6.079392. We can drop out the slowest device(s) until N starts to rise.

5.3 With device dropout

For this we drop out a device in turn, recalculate the new utilisation and queue length. Then we multiply the number of devices by the queue length to obtain the number of jobs in the system. In Table 8 we do this for all devices. It can be seen that the best results are for K = 2.

Table 8 – M/M/4 results for K = 1 to 5

<table>
<thead>
<tr>
<th>K</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_k</td>
<td>0.9</td>
<td>0.5</td>
<td>0.375</td>
<td>0.321429</td>
<td>0.3</td>
</tr>
<tr>
<td>q_k</td>
<td>10.68988</td>
<td>2.173913</td>
<td>1.544751</td>
<td>1.30758</td>
<td>1.2158787</td>
</tr>
<tr>
<td>N</td>
<td>10.68988</td>
<td>4.347826</td>
<td>4.634254</td>
<td>5.230336</td>
<td>6.079392</td>
</tr>
<tr>
<td>R</td>
<td>0.02969</td>
<td>0.012077</td>
<td>0.012873</td>
<td>0.014529</td>
<td>0.016887</td>
</tr>
</tbody>
</table>

6 Conclusions

The realisation that load balancing of heterogeneous M/M/m queues via the balanced systems method gives the same overall results as if the queues were homogeneous leads to two improvements. The first is that there is only need to solve for one queue.

The other is that is easy to calculate the number of the slowest devices that can be dropped out for small to medium workloads. This improves the average response time and reduces the electricity used to run and (possibly) cool the devices. At low to medium loadings there may be no need to trade off between energy demand and response time SLAs.

The method could also be used to calculate workload balance quickly if servers break down or need to be taken out for repair and maintenance and it can also be used for clusters of clusters.

For M/M/1 queues it is possible to do bulk dropouts and then choose whether to use BS or UE load sharing methods, whichever is more appropriate.
7 Future work

It may be possible to send high priority work to faster devices with a lower response time. This could be utilised to attain Quality of Service objectives for multiple workloads instead of pre-emption of lower priority jobs.

There seems to be no obvious reason why this work cannot be extended for closed clusters of M/M/m queues, as it already pertains to single closed workloads of M/M/1 queues [7].

References


Performance analysis of the Trusted Cloud Computing Platform

Said Naser Said Kamil * Nigel Thomas *

Abstract

Cloud computing services have greatly increased the potential for organisations and individuals to outsource data storage and computation. A critical concern is how to maintain the confidentiality and the security of data whilst maintaining ease of access and quality of service. Different service models in cloud computing are exposed to a variety of security breaches. This paper will focus on examining systems that are concerned with the security matters at the level of IaaS service model, addressed by the Trusted Cloud Computing Platform (TCCP). The objective of this paper is to examine performance of the node registration protocol within the TCCP system.

Keywords. TCCP security protocol, Cloud Computing, PEPA.

1 Introduction

Over the recent years cloud computing has become one of the top technologies for both scientific research and commercial business use. Cloud computing offers a variety of solutions for manipulating and managing large scale data storage and analysis. Low-cost, on-demand access, scalability and availability are the essential features of cloud computing and have encouraged a significant number of companies to move some of their services and data to the cloud. Also, a number of computing technologies and notions such as virtualization, Service Oriented Architecture (SOA) and Web 2.0 with dependence on the Internet are integrated and employed by cloud computing [1]. Despite the various benefits offered, the adoption of cloud computing has also given rise to several security concerns, for instance outsourcing data storage, data migration, internal security integration and multi-tenancy [2].

According to Mell and Grance [3] there are three types of service models provided by cloud computing, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Understanding the dependencies and relationships between these three models of services is extremely important to understand the associated security challenges [4], whereby the dependencies between these types of services can expose different types of risks in each level. As IaaS is the backbone of the PaaS and SaaS cloud service models, any breach at any level of IaaS can have an impact on the other types of services. Several
security aspects such as risk of migration, level of information, network and application are challenging organizations and individuals [5]. Accordingly, ensuring the integrity of data and securing live migration of information between virtual machines (VMs) through the Internet networks are extremely important security challenges that faced cloud customers.

Hashizume et al [2] have presented a summary of several security vulnerabilities and threats that have been reported by several researchers and related to VMs. For instance, transferring VMs to an untrusted host and illegal data access during migration. In response to these threats Santos et al [6] presented Trusted Cloud Computing Platform (TCCP) techniques, which aims to mitigate the risks associated with the use of VMs. TCCP provides a secure execution environment for both lunching the VMs and the migration process. Clearly, if TCCP is to be successful it not only needs to provide these secure services, but it also needs to perform to a level which does not impede the execution of other services. This paper aims to investigate the performance of the node registration process in the Trusted Cloud Computing Platform TCCP using PEPA Eclipse Plug-in tool in order to get more insight into the performance of such secure systems.

The structure of this paper is organized as follows. The paper starts with an overview about the Trusted Cloud Computing Platform (TCCP) and concentrates on the node registration protocol. This is followed by the experiments and results section, which is divided into two main parts: varying the refresh rate results and varying the probability of trust and success. Finally, the conclusion and the outlook of our future work complete the paper.

2 Trusted Cloud Computing Platform TCCP

In the design of the trusted platform module (TPM)\(^1\) chip proposed by the Trusted Computing Group (TCG) [7] the TPM is identified uniquely by an endorsement private key (EK) and unmodifiable cryptographic functions. To guarantee the key validity and the chip correctness, the matching public key is signed by the respective manufacturers. TPM features are leveraged by a number of trusted platforms such as Terra [8] with the aim of enabling remote attestation. Although it is a valuable associated security mechanism, there is a limitation in this approach where the cloud manager can inspect or tamper the customers virtual machines at the backend of IaaS. As such, the TCCP method extends the approach of the trusted platform to include the whole backend of IaaS.

The Trusted Cloud Computing Platform (TCCP) [6] is designed based on trusted computing technologies. The proposed design of TCCP provides cloud service providers (IaaS), for instance Amazon EC2, with a closed box mechanism that ensures confidentiality and integrity of hosted VMs. This enables customers to decide if the service provided via the IaaS provider is secure or not, before the real launching of the VMs operation, through a secure remote attestation. Furthermore, the TCCP system uses the Trusted Virtual Machine Monitor (TVMM) to certify that the untrusted Cloud Manager (i.e. administrator) is secure.

\(^1\)The Trusted Computing Group (TCG) [7] defines TPM (Trusted Platform Module) as a computer chip (microcontroller) that can securely store artifacts used to authenticate the platform.
erator) at the backend of IaaS provider is not allowed to access or modify the VMs of customers, where the trusted nodes will be managed by the Trusted Coordinator (TC) which is hosted and maintained by a third party External Trusted Entity (ETE). The cloud manager (CM) has no privileges in the ETE. Fig. 1 illustrates the TCCP components including the untrusted Cloud Manager (CM), Trusted Nodes (TN), and a Trusted Virtual Machine Monitor (TVMM) that run in the backend and the Trusted Coordinator (TC).

![Figure 1: The Trusted Cloud Computing Platform components, including the untrusted cloud manager (CM), the trusted coordinator (TC) and a set of trusted nodes (N) [6].](image)

### 2.1 Node Registration

According to Santos et al [6] nodes within the TCCP must embed the Trusted Platform Module (TPM) chip and adopt the TCCP protocols in order to install the TVMM securely. The TC manages and maintains a list of trusted nodes, which need to run TVMM and are placed within a security boundary. Moreover, a set of events that relate to nodes are controlled by the TC, for example add nodes, remove nodes and shut down nodes. A set of protocols has been used by the TCCP to manage the trusted nodes and to secure any operations involving the management of VMs, namely launching a VM and migrating a VM. However, critical operations, for instance suspend and resume, have not been addressed by [6].

Fig. 2 exhibits the protocol that a node must comply with to register with the TC, thus becoming a trusted node. In the first place, node N sends a nonce $n_N$ to the TC, and the TC replies with its bootstrap measurement list $ML_{TC}$, encrypted with the public endorsement key $EK_{TC}$ to guarantee the TC authenticity. The TC is trusted if the $ML_{TC}$ matches the expected configuration. In the second step, the TC attests to N by sending $n_{TC}$ to check the authenticity of N. Next the node N generates a key pair of trusted private key and trusted public key ($TK_p^N, TK_P^N$) and sends its public key $TK_P^N$ to the TC. If this is successful then the TC will add the $TK_P^N$ to its database. In step four the TC sends accepted message encrypted with $TK_P^N$, which confirms the node N is trusted. Finally, the TCCP ensures that the node trusted private key $TK_p^N$ only saved in the memory, as once the machine reboots the key will be lost and consequently, the node needs to start a new registration process.
3 Node Registration PEPA Model

In this section the TCCP node registration protocol has been modelled by means of Performance Evaluation Process Algebra (PEPA) modelling language. For those not familiar with PEPA, a formal definition can be found in [9]. The components describing the behaviour of node N and the Trusted Coordinator TC are shown in the PEPA model below.

The trusted coordinator TC components are shown below,

<table>
<thead>
<tr>
<th>N</th>
<th>sendNonce,N,r1).N_{1a}</th>
</tr>
</thead>
<tbody>
<tr>
<td>N_{1a}</td>
<td>(sendNonce,TC, r_{11})N_1 + (timeout, r_{13}).N_{fail}</td>
</tr>
<tr>
<td>N_{1}</td>
<td>(generateKeypairTprivate,N_{public},N,r_{5}).N_2</td>
</tr>
<tr>
<td>N_{2}</td>
<td>(sendPublicKey,N,p_{1}*r_{6}).N_{3}</td>
</tr>
<tr>
<td>N_{3}</td>
<td>(sendAccepted, r_{10}).N_{end} + (timeout2, r_{13}).N_{fail};</td>
</tr>
<tr>
<td>N_{end}</td>
<td>(end, r_{14}).N</td>
</tr>
<tr>
<td>N_{fail}</td>
<td>(reboot, r_{12}).N</td>
</tr>
</tbody>
</table>

The timeout and timeout2 actions within the Timer component have been used to give the system waiting time to complete some shared actions. Thus this will prevent any expected deadlocks if the TC finds that the node N is untrusted. Clearly it would be possible to extend this timer mechanism further by making the timeout Erlang distributed (and therefore more deterministic) if desired, although that is not considered here. In our model the timeout actions are triggered by the unsuccessful or untrusted actions of the TC. In practice any timeout at the node would be triggered solely by the elapsed time and would not involve any coordination with the TC. To achieve such a mechanism here would require timeout actions at the nodes and the TC, which would
unnecessarily complicate the model, as we are only interested in maintaining
the integrity of the model with respect to deadlock freeness.

\[ \text{Timer} \overset{\equiv}{=} (\text{untrusted}, r_7).\text{Timer}_1 +
\text{unsuccessful}, r_8).\text{Timer}_2 \]
\[ \text{Timer}_1 \overset{\equiv}{=} (\text{timeout}, r_{13}).\text{Timer} \]
\[ \text{Timer}_2 \overset{\equiv}{=} (\text{timeout}_2, r_{13}).\text{Timer} \]

As a final point, the system equation in the model of node registration exhibits
the cooperation between \(X\) instances of the node \(N\) and the Timer over the set
\(L\), which contains the \textit{timeout} actions. These two components cooperate with
trusted coordinator \(TC\) over the set \(M\), which contains the actions associated
with the messages sent between a node and the TC.

\[ \text{System} \overset{\equiv}{=} (N[X] \otimes \text{Timer}) \otimes_{\text{M}} \text{TC} \]

Where \(L = \{\text{timeout}, \text{timeout}_2\}\) and \(M = \{\text{sendNonce}_N, \text{sendNonce}_TC, \text{sendPublicKey}_N, \text{sendAccepted}, \text{untrusted}, \text{unsuccessful}\}\). The rates used in
this model are shown in the Table 1.

<table>
<thead>
<tr>
<th>Rate</th>
<th>Value</th>
<th>Rate</th>
<th>Value</th>
<th>Rate</th>
<th>Value</th>
<th>Rate</th>
<th>Value</th>
<th>Rate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(r_1)</td>
<td>1.0</td>
<td>(r_5)</td>
<td>0.55</td>
<td>(r_8)</td>
<td>0.65</td>
<td>(r_{11})</td>
<td>0.46</td>
<td>(r_{14})</td>
<td>0.1</td>
</tr>
<tr>
<td>(r_2)</td>
<td>0.4</td>
<td>(r_6)</td>
<td>0.75</td>
<td>(r_9)</td>
<td>0.65</td>
<td>(r_{12})</td>
<td>0.2</td>
<td>(p_1)</td>
<td>0.9</td>
</tr>
<tr>
<td>(r_3)</td>
<td>0.7</td>
<td>(r_7)</td>
<td>0.35</td>
<td>(r_{10})</td>
<td>0.73</td>
<td>(r_{13})</td>
<td>0.01</td>
<td>(p_2)</td>
<td>0.9</td>
</tr>
</tbody>
</table>

4 Experiments and Results

4.1 Node Registration by Varying Refresh Rate

In the figures which follow, the number of nodes, \(X\), has been fixed at 5. Fig.
3 shows the throughput of node registration model. As a sequence of different
actions will display exactly the same throughput in the model, it has been
decided that to choose the most significant actions, as shown in Table 2.

In addition to this, \(r_{14}\) in Fig. 3 is the rate of the end action which represents
the refresh rate in the node registration model. Thus varying \(r_{14}\) allows us to
consider different operating conditions. Hence, under a slow \(r_{14}\) rate it is less
likely that the node goes back and requests a new registration. As a result,
when it is very slow (e.g. \(r_{14} = 0.0001\)) the average throughput of all actions
is also very small. However, increasing the refresh rate \(r_{14}\) to be much faster
by varying its value from 0.0001 to 10 then the throughput of all actions will
increase, as seen in the Fig. 3. Ultimately, if \(X\) and \(r_{14}\) are large enough, the
load of requests becomes such that nodes will be forced to queue for service by
the TC. In Fig. 3 we can see that the throughput levels off when \(r_{14} = 0.1\),
showing that this is the maximum throughput can achieved as the TC saturates
at this point (utilisation is almost 1).

The action \textit{sendNonce}_\(N\) represents the initialisation of a node request to
register with the Trusted Coordinator (TC), while \textit{sendAccepted} represents the
Table 2: The actions that have the same throughput of the other actions.

<table>
<thead>
<tr>
<th>Action</th>
<th>Identical Throughput Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>sendNonce_N</td>
<td>bootstrapML_TC</td>
</tr>
<tr>
<td></td>
<td>encryptedPrivateKey_TC</td>
</tr>
<tr>
<td>isTrusted</td>
<td>generateKeypairTprivate_NTpublic_N_TC</td>
</tr>
<tr>
<td></td>
<td>sendNonce_TC</td>
</tr>
<tr>
<td></td>
<td>sendPublicKey_N</td>
</tr>
<tr>
<td>untrusted</td>
<td>timeout</td>
</tr>
<tr>
<td>unsuccessful</td>
<td>timeout2</td>
</tr>
<tr>
<td>sendAccepted</td>
<td>attestSuccessfully</td>
</tr>
<tr>
<td></td>
<td>end</td>
</tr>
</tbody>
</table>

completion of the node registration protocol. From Fig. 3 it can be observed that the throughput of untrusted and unsuccessful is low because the majority of time the probability of failure is small. Furthermore, there is a small difference between the throughputs of sendNonce_N, isTrusted and attestSuccessfully because some requests fail at the untrusted action and yet more fail at the unsuccessful action. Note that in this model the requests which fail are governed solely by probabilities, although clearly we could extend the model by explicitly representing misbehaving nodes.

Figure 3: The throughput of the Node Registration Model varied with refresh rate.

Fig. 4 shows the population of the node components, where the population of N is noticeably decreased influenced by the reduction in the refresh rate. In contrast, Nend is increased as r14 decreases; clearly as the refresh rate becomes slower the nodes spend most of their time at the state Nend. The performance of the Trusted Coordinator (TC) components is shown in the Fig. 5. Since there is only one TC component, the population here depicts the marginal probabilities that the TC is in each of its possible behaviours. Again we observe that if the refresh rate is low then the TC will spend most of its time idle, waiting for requests. In order to show this effect more clearly Fig. 6 illustrates just the initial state of TC and the successful completion of registration, TC5. Fig. 7 is used to show the performance of other behaviours of the TC that cannot easily
be seen in Fig. 5, thus making it easier to distinguish between them.

Figures 4 to 13 show the Ordinary Differential Equation (ODE) transient analysis that has been implemented on the Node Registration Model. These figures display the evolution of the system before it reaches its steady state and showing clearly the evolution behaviour of states of every single component. For the reasons described earlier it has been decided to split the resulted graphs into Node components (N) and Trusted Coordinator components (TC). In Fig. 8, the idle node population decreases from the initial value of 5 up to approximately 3, because the refresh rate is fast \( r_{14} = 0.1 \) and as a result nodes spend little time in \( N_{end} \) and therefore compete for access to the TC. Fig. 9 shows that the TC reaches its steady state behaviour much more quickly than the nodes. As we can see at this value of \( r_{14} \), the TC is rarely idle.

In the case shown in Fig. 10 and Fig. 11 \( r_{14} = 0.01 \) and immediately we see that this greatly affects the transient behaviour. Firstly we see that as nodes are spending more time in the behaviour \( N_{end} \), there is far less competition for access to the TC. Indeed, the number of nodes behaving as \( N \) rapidly drops to near zero. We also observe that the steady state behaviour of the nodes is reached much more rapidly, due to all the nodes completing registration and...
then spending time in $N_{end}$. The behaviour of the $TC$ component shown in Fig. 11 shows a rapid trend to steady behaviour, but then there is a curious jump at around 120 seconds, when the $TC$ becomes more likely to be idle. This jump corresponds to the completion of the initial registration of the 5 nodes and the consequent assuming of steady state.

The effects shown in Fig. 10 and Fig. 11 are even greater magnified in Fig. 12 and Fig. 13, where $r_{14} = 0.001$. Here we see that the average number of nodes behaving as $N_{end}$ rises even higher and steady state has not quite been reached at the end of the transient period shown. The $TC$ behaviour shows an even larger jump at the end of the initial node registrations, which takes place a little earlier due to a reduced effect of nodes requesting registration for a second time. Transient results have also been derived using stochastic simulation and these show very close agreement with the ODE results depicted here.
4.2 Node Registration Varying the Probability of Trust and Success

In the second set of experiments the probabilities of trust and success, $p_1$ and $p_2$, are varied from 0.1 to 0.9 in order to investigate the impact of failure on the performance of the model. Fig. 14 and Fig. 15 show the throughput of the Node Registration Model by varying the trust and success probabilities. Decreasing $p_1$ means a higher portion of requests coming from a node are untrusted by the Trusted Coordinator TC. This might happen for several reasons in practice, for instance, the node may have been incorrectly initiated, become compromised or an error has been made in transmitting, retrieving or computing the different values. Decreasing $p_2$ means a greater portion of unsuccessful registrations of trusted nodes, most likely due to errors in communication or the storage of keys. Clearly a decrease in $p_2$ will have a greater effect on the TC performance than a decrease in $p_1$, simply because an unsuccessful registration will have consumed a lot of TC effort, whereas an untrusted node will be detected relatively quickly. Thus a low value of $p_2$ will mean that the effective capacity of the TC may be significantly reduced.

Fig. 15 illustrates the case that the reboot action takes place and reaches its peak when decreasing the probability of trust and success to 0.1. Because 90% of requests at this point 0.1 will be untrusted and about 9% are unsuccessful (i.e. 90% of the 10% which are trusted). As a result, 99% are not going to succeed and there only 1% of requests will succeed. Conversely, increasing the probability of trust and success to be 0.9 will result in about 11% unsuccessful because 10% are untrusted and then 10% of the other 90% are unsuccessful. So, decreasing the probability of trust and success increases the probability of reboot. In addition, it is interesting to notice that the throughput of unsuccessful in Fig. 15 has a peak at 0.5 and then decline at the end. This is simply because so few requests are reaching the point at which success is determined and so the throughput of both the actions attestSuccessfully and unsuccessful will be small. This is consequence of the decision to make $p_1 = p_2$ in these experiments.

Fig. 16, Fig. 17 and Fig. 18 present the results of CTMC analysis and depict the population of Node Registration Model by varying the probability of trust and success rates ($p_1$ and $p_2$). For clarity the population analysis has been split into three graphs. It is interesting to note that, even when decreasing the probability of trust and success, the population of N in Fig. 16 decreased...
steadily up to the point $p_1 = p_2 = 0.5$, then starts to increase again. As the probability of success decreases, the TC will spend more time to process incoming requests. Besides, a small number of things getting to the $N_{end}$ as the probability of success decreased few of requests will successes. In Fig. 17 the population of $TC_1$ is increases progressively as $p_1$ and $p_2$ decrease. This is due to the fact that an increasing number of requests are untrusted, therefore the $TC$ is not often reaching behaviours further on in the protocol. The rise of $TC_1$ dominates Fig. 17 and it is hard to see what is happening to other behaviours. Hence Fig. 18 displays the same data as Fig. 17, except $TC_1$ has been removed. Thus we can see that $TC_3$ in Fig. 18 rises steadily along with the increase the probability of failure; as more requests will be unsuccessful and therefore later behaviours are less likely to be reached. This effect of $TC_3$ in Fig. 18 mirrors that of $TC_1$ in Fig. 17, albeit at a much lower intensity.
The ODE transient behaviour of the Node Registration Model by means of varying the probability of trust and success are displayed in Fig. 19, Fig. 20, Fig. 21, Fig. 22, Fig. 23 and Fig. 24. Again we have split the Node and TC into separate graphs. It is obvious that in the set of Node figures (Fig. 19, Fig. 21 and Fig. 23) the behaviour of the node is affected by decreasing the probability of trust and success. When \( p_1 = p_2 = 0.9 \) the population of \( N \) declines gradually from 5 to about 3 to reach its steady state as shown in Fig. 19. However, Fig. 21 and Fig. 23 illustrate that higher probabilities of \( p_1 \) and \( p_2 \) will make \( N \) finish its process quicker and thus steady state is reached sooner. Furthermore we observe a change in the population of \( N_1a \) and \( N_3 \) between the three graphs. This is because these two components will be in the state of timeout and timeout2 respectively waiting the TC to finish, which is more likely when the probability of failure is higher. The evolution in the TC performance before reaches its steady state is shown in Fig. 20, Fig. 22 and Fig. 24. The TC reaches steady state quickly when the probability of failure is small. However as the probability of failure increases, so does the variability in behaviour caused by the TC aborting requests and the associated nodes suffering timeouts and returning to request again at a later time. This transitory behaviour is shown to have a significant effect on TC1 when \( p_1 = p_2 = 0.1 \). However, when \( p_1 = p_2 = 0.5 \) all the actions show a definite transitory oscillation which dampens to reach steady state. The performance of the TC would clearly be highly unpredictable during this phase of operation.
Figure 19: ODEs analysis of Node Registration Model, showing only Node components (Ns) and the probability of trust and success (p1 and p2=0.9).

Figure 20: ODEs analysis of Node Registration Model, showing only Trusted Coordinator components (TCs) and the probability of trust and success (p1 and p2=0.9).

Figure 21: ODEs analysis of Node Registration Model, showing only Node components (Ns) and the probability of trust and success (p1 and p2=0.5).

Figure 22: ODEs analysis of Node Registration Model, showing only Trusted Coordinator components (TCs) and the probability of trust and success (p1 and p2=0.5).

Figure 23: ODEs analysis of Node Registration Model, showing only Node components (Ns) and the probability of trust and success (p1 and p2=0.1).

Figure 24: ODEs analysis of Node Registration Model, showing only Trusted Coordinator components (TCs) and the probability of trust and success (p1 and p2=0.1).
5 Conclusion

This paper has presented an approach to analysing the security protocol of node registration, which is the initial part of the Trusted Cloud Computing Platform TCCP mechanism. A model for node registration protocol has been created using the PEPA modelling language and a variety of analysis techniques offered by PEPA Eclipse Plug-in tool have been applied. Different scenarios for node registration PEPA model have been considered. In the first case we have investigated the load on the TC by varying the refresh rate. In the second case we have investigated the effect of failures on the performance of the TC. Although, the experiments are implemented on a small number of nodes, the results of these experiments show a clear behaviour of the system under examination. Even with very few nodes we are able to show the behaviour of the TC under high utilisation. We are also able to show that varying the failure rate not only has a predictable effect on steady state performance, but it also affects the transient behaviour before steady state is reached, which captures the behaviour of a newly initiated system or a system undergoing a sudden increase in activity.

Clearly further analysis is possible with this model and may provide additional insight as to the behaviour of node registration service. However, node registration is only one part of the TCCP, and indeed it is the simplest of the defined protocols. It is therefore of interest to follow up this study with an analysis of other functions provided by the TCCP and to consider the cumulative load effects on the TC. Modelling the situation where multiple requests of different types are being made to the TC is potentially challenging. However, knowing how to prioritise different services when secure resources are limited in order to maintain an acceptable level of service is clearly of practically benefit.

References


Large-Scale Data Center with Setup Time and Impatient Customer

Tuan Phung-Duc∗

Abstract

Data centers consume a large amount of energy in order to run and to keep cool. Saving a few percent of power-consumption has a big impact on the running cost of data centers. A simple and natural policy for power-saving data center is to turn off idle serves since these servers still consume about 60% of their peak processing jobs. However, servers should be turned on again when the work load increases. Servers need some setup time during which they consume energy but cannot process jobs. Furthermore, setup time also incurs in extra waiting time and then the abandonment of customers. The underlying Markov chain of these systems has a level-dependent QBD (quasi-birth and death) structure whose stationary distribution can be obtained using existing methods. However, due to the complexity of these methods, it is difficult to analyze the performance of large-scale data centers. In this paper, we propose an efficient method aiming at a quick performance evaluation of large scale-data centers. The complexity of the method is proportional to the number of states of the Markov chain and is significantly reduced in comparison with existent methods in the literature.

1 Introduction

1.1 Motivation

The core part of cloud computing is data center where a huge number of servers are available. These servers consume a large amount of energy. Thus, the key issue for the management of these server farms is to minimize the power consumption while keeping acceptable service level for users. It is reported that under the current technology an idle server still consumes about 60% of its peak processing jobs [2]. Therefore, the only way to save power is to turn off idle servers. However, off servers need some setup time to be active during which they consume energy but cannot process jobs. As a result, there exists a trade-off between power-saving and performance which could be analyzed by queueing models with setup time. Recently, motivated by applications in data centers, multiserver queues with setup times have been extensively investigated in the literature.

∗Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, tuan@is.titech.ac.jp
1.2 Related work

Recently, Gandhi et al. [5] analyze multiserver queues with setup times. They obtain some closed form approximations for the ON-OFF policy where any number of servers can be in the setup mode at a time. As is pointed out in Gandhi et al. [5], from an analytical point of view the most challenging model is the ON-OFF policy where the number of servers in setup mode is not limited. Gandhi et al. [6, 7] analyze the M/M/c/Setup model with ON-OFF policy using a recursive renewal reward approach. Phung-Duc [14] obtains exact solutions for the same model via generating functions and via matrix analytic methods. The algorithms in these works can compute the stationary distribution for models with a relatively small number of servers. It should be noted that Gandhi et al. [6, 7] and Phung-Duc [14] consider the infinite buffer model.

Although, the infinite model has been investigated [6, 7], results for systems with a large number (several hundreds) of servers are not obtained. This motivates us to develop new methods for the performance evaluation of large-scale server farms. Phung-Duc [16] presents a simple recursion for the stationary distribution of the M/M/c/K/Setup without abandonment. The computational complexity of the scheme is significantly reduced in comparison with that of direct methods [11, 12]. As a result, models with several hundreds of servers are easily analyzed. This allows us to explore new insights into the performance of large scale systems. Recently, in a closely related paper [8], Kuehn et al. suggest a recursive scheme for finite buffer model with threshold control. However, the stability of the numerical scheme is not proved and the abandonment of customers is not taken into account. In contrast to [8], we suggest here a new recursive scheme whose numerical stability is guaranteed. In all the work above, the abandonment of customers is ignored. However, since cloud users have impatient nature, it is important to consider a model with abandonment of customers. This motivates us to consider a model with abandonment and develop an efficient method for analyzing large-scale systems.

Some other related works are as follows. Mitrani [9, 10] considers models for server farms with setup costs. The author analyzes the models where a group of reserve servers are shutdown simultaneously if the number of jobs in the system is smaller than some lower threshold and are powered up simultaneously when it exceeds some upper threshold. Because of this simultaneous shutdown and setup, the underlying Markov chain in [10] has a birth and death structure which allows closed form solutions. The author investigates the optimal lower and upper thresholds for the system. The same author [9] extends their analysis to the case where each job has an exponentially distributed random timer exceeding which the job leaves the system. Schwartz et al. [17] consider a similar model to that in [9]. Single server models are investigated in [3, 4, 15, 18].

1.3 Organization of the paper

The rest of this paper is organized as follows. Section 2 presents the model in details while Section 3 is devoted to derivation of a recursion for the joint stationary distribution. Section 4 presents some numerical examples showing insights into the performance of the system. Concluding remarks are presented in Section 5.
2 Model

We consider a queueing system with \(c\) servers and a capacity of \(K\), i.e., the maximum of \(K\) customers can be accommodated in the system. Jobs arrive at the system according to a Poisson process with rate \(\lambda\). In this system, a server is turned off immediately if it has no job to do. Upon arrival of a job, an OFF server is turned on if any and the job is placed in the buffer. However, a server needs some setup time to be active so as to serve waiting jobs. We assume that the setup time follows the exponential distribution with mean \(1/\alpha\). Let \(j\) denote the number of customers in the system and \(i\) denote the number of active servers. The number of servers in setup process is \(\min(j-i, c-i)\). Under these assumptions, the number of active servers is smaller than or equal to the number of jobs in the system. Therefore, in this model a server is in either BUSY or OFF or SETUP. We assume that the service time of jobs follows the exponential distribution with mean \(1/\mu\). Furthermore, we assume that customers are of impatient nature and they abandon receiving service if the waiting time exceeds some threshold. In particular, we assume that a customer abandons the system after some exponentially distributed time with mean \(1/\theta\). We assume that waiting jobs are served according to a first-come-first-served (FCFS) manner. We call this model an M/M/c/K/Setup queue with abandonment.

Remark 1. The exponential assumptions for the inter-arrival, setup time and service time allow to construct a Markov chain whose stationary distribution is recursively obtainable. It should be noted that we can easily construct a Markov chain for a more general model with MAP arrival and phase-type service and setup time distributions. However, the state space of the resulting Markov chain explodes and thus the analysis is complex.

3 Analysis

In this section, we present a recursive scheme to calculate the joint stationary distribution. Let \(C(t)\) and \(N(t)\) denote the number of active servers and the number of customers in the system, respectively. It is easy to see that \(\{X(t) = (C(t), N(t)) ; t \geq 0\}\) forms a Markov chain on the state space:

\[ S = \{(i, j) ; 0 \leq i \leq c, j = i, i + 1, \ldots, K - 1, K\}. \]

See Figure 1 for transition among states for the case \(c = 2\) and \(K = 5\).

Let \(\pi_{i,j} = \lim_{t \to \infty} P(C(t) = i, N(t) = j) ((i, j) \in S)\) denote the joint stationary distribution of \(\{X(t)\}\). In this section, we derive a recursion for calculating the joint stationary distribution \(\pi_{i,j} ((i, j) \in S)\). The balance equations for states with \(i = 0\) read as follows.

\[
\begin{align*}
\lambda \pi_{0,0} &= \mu \pi_{1,1} + \theta \pi_{0,1}, \\
(\lambda + \min(j, c)\alpha + j\theta) \pi_{0,j} &= \lambda \pi_{0,j-1} + (j + 1)\theta \pi_{0,j+1}, \quad j = 1, 2, \ldots, K - 1, \\
(K\theta + c\alpha) \pi_{0,K} &= \lambda \pi_{0,K-1}.
\end{align*}
\]
leading to $\pi_{0,j} = b_j^{(0)} \pi_{0,j-1}$ where

$$b_j^{(0)} = \frac{\lambda}{\lambda + \min(j, c)\alpha + j\theta - (j+1)\theta b_{j+1}^{(0)}},$$

for $j = 1, 2, \ldots, K-1$, where

$$b_K^{(0)} = \frac{\lambda}{K\theta + c\alpha}.$$

**Theorem 3.1.** We have the following bound.

$$0 < b_j^{(0)} < \frac{\lambda}{j\theta + \min(j, c)\alpha}.$$

**Proof.** It is clear that Theorem 3.1 is true for $j = K$. Assuming that Theorem 3.1 is true for $j+1$, i.e.,

$$0 < b_{j+1}^{(0)} < \frac{\lambda}{(j+1)\theta + \min(j+1, c)\alpha} < \frac{\lambda}{(j+1)\theta}.$$

This inequality and the recursion (1) imply that Theorem 3.1 is also true for $j$. 

Furthermore, it should be noted that $\pi_{1,1}$ is calculated using the local balance equation in and out the set $\{(0, j); j = 0, 1, \ldots, K\}$ as follows.

$$\mu \pi_{1,1} = \sum_{j=1}^{K} \min(j, c)\alpha \pi_{0,j}.$$

**Remark 2.** We have expressed $\pi_{0,j}$ ($j = 1, 2, \ldots, K$) and $\pi_{1,1}$ in terms of $\pi_{0,0}$. 
Next, we consider the case \( i = 1 \).

**Lemma 3.2.** We have
\[
\pi_{1,j} = a_{j}^{(1)} + b_{j}^{(1)} \pi_{1,j-1}, \quad j = 2, 3, \ldots, K - 1, K,
\]
where
\[
a_{j}^{(1)} = \frac{(\mu + j\theta) a_{j+1}^{(1)} + \min(j, c) \alpha \pi_{0,j}}{s_{j}^{(1)}},
\]
\[
b_{j}^{(1)} = \frac{\lambda}{s_{j}^{(1)}},
\]
\[
s_{j}^{(1)} = \mu + \lambda + \min(j - 1, c - 1)\alpha + (j - 1)\theta - (\mu + j\theta) b_{j+1}^{(1)},
\]
for \( j = K - 1, K - 2, \ldots, 2 \) and \( a_{K}^{(1)} = \frac{c \alpha \pi_{0,K}}{\mu + (c - 1)\alpha + (K - 1)\theta}, \)
\[
b_{K}^{(1)} = \frac{\lambda}{\mu + (c - 1)\alpha + (K - 1)\theta}.
\]

**Proof.** We prove using mathematical induction. Balance equations are given as follows.
\[
(\lambda + \mu + \min(j - 1, c - 1)\alpha + (j - 1)\theta) \pi_{1,j} = \lambda \pi_{1,j-1} + (\mu + j\theta) \pi_{1,j+1} + \min(j, c) \alpha \pi_{0,j},
\]
\[\text{for } 2 \leq j \leq K - 1,
\]
\[
(\mu + \min(K - 1, c - 1)\alpha + (K - 1)\theta) \pi_{1,K} = \lambda \pi_{1,K-1} + c \alpha \pi_{0,K}.
\]
It follows from (5) that
\[
\pi_{1,K} = a_{K}^{(1)} + b_{K}^{(1)} \pi_{1,K-1},
\]
leading to the fact that Lemma 3.2 is true for \( j = K \). Assuming that Lemma 3.2 is true for \( j + 1 \), i.e., \( \pi_{1,j+1} = a_{j+1}^{(1)} + b_{j+1}^{(1)} \pi_{1,j} \). It then follows from (4) that Lemma 3.2 is also true for \( j \), i.e., \( \pi_{1,j} = a_{j}^{(1)} + b_{j}^{(1)} \pi_{1,j-1} \).

**Theorem 3.3.** We have the following bound.
\[
a_{j}^{(1)} \geq 0, \quad 0 \leq b_{j}^{(1)} \leq \frac{\lambda}{\mu + (j - 1)\theta + \min(j - 1, c - 1)\alpha},
\]
for \( j = 2, 3, \ldots, K - 1, K \).

**Proof.** We use mathematical induction. It is easy to see that the theorem is true for \( j = K \). Assuming that the theorem is true for \( j + 1 \), i.e.,
\[
a_{j+1}^{(1)} \geq 0, \quad 0 \leq b_{j+1}^{(1)} \leq \frac{\lambda}{\mu + j\theta + \min(j, c - 1)\alpha},
\]
for \( j = 1, 2, \ldots, K - 1 \). Thus, we have \( \mu b_{j+1}^{(1)} < \lambda \). From this inequality, (2) and (3), we find that
\[
b_{j}^{(1)} \leq \frac{\lambda}{\mu + (j - 1)\theta + \min(j - 1, c - 1)\alpha},
\]
and \( a_{j}^{(1)} \geq 0 \).
It should be noted that $\pi_{2,2}$ can be calculated using the local balance between the flows in and out the set of states $\{(i, j); i = 0, 1, j = i, i+1, \ldots, K\}$ as follows.

$$2\mu \pi_{2,2} = \sum_{j=2}^{K} \min(j - 1, c - 1)\alpha \pi_{1,j}.$$

**Remark 3.** We have expressed $\pi_{1,j}$ $(j = 1, 2, \ldots, K)$ and $\pi_{2,2}$ in terms of $\pi_{0,0}$.

We consider the general case where $2 \leq i \leq c - 1$. Similar to the case $i = 1$, we can prove the following result by mathematical induction.

**Lemma 3.4.** We have

$$\pi_{i,j} = a_{j}^{(i)} + b_{j}^{(i)}\pi_{i,j-1}, \quad j = i + 1, i + 2, \ldots, K - 1, K,$$

where

$$a_{j}^{(i)} = \frac{(i\mu + (j + 1 - i)\theta)\pi_{j+1}^{(i)} + \min(c - i + 1, j - i + 1)\alpha \pi_{i-1,j}}{s_{j}^{(i)}} - (i\mu + (j + 1 - i)\theta)\pi_{j+1}^{(i)},$$

$$b_{j}^{(i)} = \frac{\lambda}{s_{j}^{(i)}} - (i\mu + (j + 1 - i)\theta)\pi_{j+1}^{(i)},$$

where $s_{j}^{(i)} = \lambda + \min(c - i, j - i)\alpha + i\mu + (j - i)\theta$ and

$$a_{K}^{(i)} = \frac{(c - i + 1)\alpha \pi_{i-1,K}}{(c - i)\alpha + i\mu + (K - i)\theta},$$

$$b_{K}^{(i)} = \frac{\lambda}{(c - i)\alpha + i\mu + (K - i)\theta}.$$

**Proof.** The balance equation for state $(i, K)$ is given as follows.

$$((c - i)\alpha + i\mu + (K - i)\theta)\pi_{i,K} = \lambda \pi_{i,K-1} + (c - i + 1)\alpha \pi_{i-1,K},$$

leading to the fact that Lemma 3.4 is true for $j = K$. Assuming that

$$\pi_{i,j+1} = a_{j+1}^{(i)} + b_{j+1}^{(i)}\pi_{i,j}, \quad j = i + 1, i + 2, \ldots, K - 1.$$

It then follows from

$$\lambda \pi_{i,j-1} + (i\mu + (j + 1 - i)\theta)\pi_{i,j} = \pi_{i,j+1} + \min(c - i + 1, j - i + 1)\alpha \pi_{i-1,j},$$

$j = K - 1, K - 2, \ldots, i + 1$,

that

$$\pi_{i,j} = a_{j}^{(i)} + b_{j}^{(i)}\pi_{i,j-1}.$$

\[\square\]

**Theorem 3.5.** We have the following bound.

$$a_{j}^{(i)} > 0, \quad 0 < b_{j}^{(i)} < \frac{\lambda}{i\mu + (j - i)\theta + \min(j - i, c - i)\alpha},$$

for $j = i + 1, i + 2, \ldots, K - 1, i = 1, 2, \ldots, c - 1$. 

---

UKPEW 2015 – http://ukpew.org/
Proof. We also prove using mathematical induction. It is clear that Theorem 3.5 is true for $j = K$. Assuming that Theorem 3.5 is true for $j + 1$, i.e.,

$$a_{j+1}^{(i)} > 0,$$

$$0 < b_{j+1}^{(i)} < \frac{\lambda}{\mu + (j + 1 - i)\theta + \min(j + 1 - i, c - i)\alpha},$$

for $j = i + 1, i + 2, \ldots, K - 1, i = 1, 2, \ldots, c - 1$. It follows from the second inequality that $i\mu b_{j+1}^{(i)} < \lambda$. This together with formulae (6) and (7) yield the desired result.

It should be noted that $\pi_{c+1,i+1}$ is calculated using the following local balance equation in and out the set of states:

$$\{ (k, j); k = 0, 1, \ldots, i; j = k, k + 1, \ldots, K \}$$

as follows.

$$(i + 1)\mu \pi_{c+1,i+1} = \sum_{j=i+1}^{K} \min(j - i, c - i)\alpha \pi_{i,j},$$

**Remark 4.** We have expressed $\pi_{i,j}$ ($i = 0, 1, \ldots, c - 1, j = i, i + 1, \ldots, K$) and $\pi_{c+1,i+1}$ in terms of $\pi_{0,0}$.

**Lemma 3.6.** We have

$$\pi_{c,j} = a_{j}^{(c)} + b_{j}^{(c)} \pi_{c,j-1}, \quad j = c + 1, c + 2, \ldots, K - 1,$$

where

$$a_{j}^{(c)} = \frac{(c\mu + (j + 1 - c)\theta)a_{j+1}^{(c)} + \alpha \pi_{c-1,j}}{\lambda + c\mu + (j - c)\theta - (c\mu + (j + 1 - c)\theta)b_{j+1}^{(c)}},$$

$$b_{j}^{(c)} = \frac{\lambda}{\lambda + c\mu + (j - c)\theta - (c\mu + (j + 1 - c)\theta)b_{j+1}^{(c)}},$$

and

$$a_{K}^{(c)} = \frac{\alpha \pi_{c-1,K}}{c\mu + (K - c)\theta}, \quad b_{K}^{(c)} = \frac{\lambda}{c\mu + (K - c)\theta}.$$

Proof. The global balance equation at state $(c, K)$ is given by

$$(c\mu + (K - c)\theta)\pi_{c,K} = \alpha \pi_{c-1,K} + \lambda \pi_{c,K-1},$$

leading to

$$\pi_{c,K} = a_{K}^{(c)} + b_{K}^{(c)} \pi_{c,K-1}.$$ 

Assuming that $\pi_{c,j+1} = a_{j+1}^{(c)} + b_{j+1}^{(c)} \pi_{c,j}$, it follows from the global balance equation at state $(c, j)$,

$$(\lambda + c\mu + (j - c)\theta)\pi_{c,j} = \lambda \pi_{c,j-1} + (c\mu + (j + 1 - c)\theta)\pi_{c,j+1} + \alpha \pi_{c-1,j},$$

$$(\lambda + c\mu + (j - c)\theta)\pi_{c,j} = \lambda \pi_{c,j-1} + (c\mu + (j + 1 - c)\theta)\pi_{c,j+1} + \alpha \pi_{c-1,j},$$

$$(\lambda + c\mu + (j - c)\theta)\pi_{c,j} = \lambda \pi_{c,j-1} + (c\mu + (j + 1 - c)\theta)\pi_{c,j+1} + \alpha \pi_{c-1,j},$$

that $\pi_{c,j} = a_{j}^{(c)} + b_{j}^{(c)} \pi_{c,j-1}$ for $j = c + 1, c + 2, \ldots, K$. □
Theorem 3.7. We have the following bound.

\[ a^{(c)}_j > 0, \quad 0 < b^{(c)}_j < \frac{\lambda}{c\mu + (j - c)\theta}, \]

\[ j = c + 1, c + 2, \ldots, K - 1. \]

Proof. We also prove using mathematical induction. It is clear that Theorem 3.7 is true for \( j = K \). Assuming that Theorem 3.7 is true for \( j + 1 \), i.e.,

\[ a^{(c)}_{j+1} > 0, \quad 0 < b^{(c)}_{j+1} < \frac{\lambda}{c\mu + (j + 1 - c)\theta}, \]

\[ j = c + 1, c + 2, \ldots, K - 1. \]

It follows from the second inequality that \((c\mu + (j + 1 - c)\theta)b^{(c)}_{j+1} < \lambda\). This together with formulae (8) and (9) yield the desired result.

We have expressed all the probability \( \pi_{i,j} \) \((i,j) \in S\) in terms of \( \pi_{0,0} \) which is uniquely determined by the normalizing condition.

\[ \sum_{(i,j) \in S} \pi_{i,j} = 1. \]

Remark 5. We see that the computational complexity order for \( \{\pi_{i,j}; (i,j) \in S\} \) is \( O(cK) \). A direct method for solving the set of balance equations requires the complexity of \( O(c^3K^3) \) while a level-dependent QBD approach needs the computational complexity of \( O(Kc^3) \).

Remark 6. The computational procedure for calculating \( \pi_{i,j} \) is numerically stable due to two reasons. First, we manipulate only positive numbers. Second, the quantities \( a^{(i)}_j, b^{(i)}_j \) are not big (See Theorems 3.1, 3.3, 3.5 and 3.7).

4 Performance Evaluation

4.1 Performance measures

Let \( P_B \) denote the blocking probability. We have

\[ P_B = \sum_{i=0}^{c} \pi_{i,K}. \]

Let \( \pi_i \) denote the stationary probability that there are \( i \) active servers, i.e., \( \pi_i = \sum_{j=i}^{K} \pi_{i,j} \). Let \( E[A] \) and \( E[S] \) denote the mean number of active servers and that in setup mode, respectively. We have

\[ E[A] = \sum_{i=1}^{c} i\pi_i, \quad E[S] = \sum_{i=0}^{c} \sum_{j=1}^{K} \min(j - i, c - i)\pi_{i,j}. \]

The power for the model with setup time is given by

\[ Cost_{on-off} = C_aE[A] + C_sE[S], \quad (10) \]
where $C_a$ and $C_s$ are the cost per a unit time for an active server and a server in setup mode, respectively.

For comparison, we also find the power for the corresponding ON-IDLE model, i.e., M/M/c/K without setup times. Letting $p_i$ ($i = 0, 1, \ldots, K - 1, K$) denote the stationary probability that there are $i$ customers in the system, we have

$$p_i = \frac{\lambda^i}{\prod_{j=1}^{i} \mu_j} p_0, \quad i = 1, 2, \ldots, K,$$

where $\mu_j = \min(j, c)\mu + \max(0, j - c)\theta$ and $p_0$ is determined by the normalization condition $\sum_{i=0}^{K} p_i = 1$. Let $\mathbb{E}[\hat{A}]$ denote the mean number of active servers, we have

$$\mathbb{E}[\hat{A}] = \sum_{i=0}^{K} \min(i, K) p_i.$$

The mean number of idle servers is given by $c - \mathbb{E}[\hat{A}]$. Thus, for this model, the power is given by

$$C_{\text{on-idle}} = C_a \mathbb{E}[\hat{A}] + (c - \mathbb{E}[\hat{A}])C_i. \quad (11)$$

where $C_i$ is the cost per a unit time for an idle server.

Let $\mathbb{E}[N]$ denote the mean number of customers in the system. We have

$$\mathbb{E}[N] = \sum_{i=0}^{c} \sum_{j=i}^{K} p_{i,j} \times j.$$

Let $\mathbb{E}[T]$ denote the mean response time of a customer, i.e., the time from arrival to either a departure or an abandonment. We have

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{\lambda(1 - P_B)},$$

for ON-OFF model and

$$\mathbb{E}[T] = \frac{\mathbb{E}[N]}{\lambda(1 - P_K)},$$

for ON-IDLE model.

### 4.2 Blocking probability

We consider the following parameter setting: $c = K = 400, 500$, $\mu = 1$. We consider the case where $\theta = 0.1$ meaning that the mean time to abandon is 10 times longer than the mean service time. Figure 2 represents the blocking probability against the traffic intensity $\rho = \lambda/(c\mu)$. As is expected we observe that the blocking probability for $c = 500$ is smaller than that of $c = 400$ keeping the traffic intensity the same.
4.3 Abandonment rate and throughput

Next, we investigate the effect of the traffic intensity on the abandonment rate and the throughput of the system. Parameter settings are the same as in Section 4.2. Figure 3 represents the abandonment rate against the traffic intensity. We observe an interesting phenomenon. In particular, we observe that the abandonment rate (which is linear to the mean number of waiting jobs) increases with the traffic intensity when the traffic intensity is small. However, the abandonment rate decreases when the traffic intensity is large enough. At the first look, this does not agree with intuition in queues without setup. It should be noted that in our model, an idle server is immediately turned off and it is turned on again when the workload increases. Thus, when the traffic intensity is small the number of active servers is small leading to the increases in the abandonment rate. However, when the traffic intensity is large, almost all the servers are likely active. As a result, the abandonment rate decreases with the traffic intensity.

Figure 4 shows the throughput against the traffic intensity. We observe that the throughput increases with the traffic intensity as is expected. We also observe that the throughput increases with the setup rate. This is because, a slow setup time incurs in the increase of both abandonment and losses.

4.4 Mean sojourn time

In this section, we present the mean sojourn time against traffic intensity. We consider the case $c = 40, 50, K = 500$. Figure 5 represents the mean sojourn time for the case $\theta = 0.01$ while Figure 6 shows the mean sojourn time for the case $\theta = 0.1$. We observe from Figure 5 that the mean sojourn time decreases with the traffic intensity when the traffic intensity is small. This is also counter-intuitive for models without setup time. It should be noted again that an idle server is immediately turned off and it is turned on again when a customer
Figure 3: Abandonment rate against $\rho$ ($c = 400, 500, \theta = 0.1$).

Figure 4: Throughput against $\rho$ ($c = 400, 500, \theta = 0.1$).
Figure 5: Mean response time against $\rho$ ($c = 40, 50, \theta = 0.01, K = 500$).

Figure 6: Mean response time against $\rho$ ($c = 40, 50, \theta = 0.1, K = 500$).
arrives. Under light traffic intensity, the effect of setup time is large because servers have chance to be turned on and off. As a result, increasing the traffic intensity incurs in the increase in the number of active servers.

However, when the traffic intensity is large enough, all the servers are likely ON all the time. Thus, our system behaves similarly to the corresponding system without setup time. Thus, we observe that the mean sojourn time increases with the traffic intensity as in the corresponding system without setup time.

5 Concluding remarks

We present a simple recursion to calculate the stationary distribution of the system state of an $M/M/c/K$/Setup queue with abandonment for data centers. The computational complexity order of the algorithm is only $O(cK)$ which is the same order as the number of states. The methodology of this paper can be applied for various variant models with setup time and finite buffer. In particular, the methodology of this paper can also be applied to the finite buffer $M/M/c$ queue with vacation and abandonment.

References


Acknowledgements

The author was supported in part by JSPS KAKENHI Grant Number 2673001. The author would like to thank the referee for constructive comments which improve the presentation of the paper.
Resource Allocation in Cloud Computing Environments

Abdullah Sheikh
Nigel Thomas
School of Computer Science
Newcastle University
{a.sheikh1|nigel.thomas}@newcastle.ac.uk

Abstract Cloud computing has been highly considered by both educational and business sectors. These considerations include the need of improving the Quality of Services (QoS) provided in different aspects such as time, best performance, and cost effective. This paper aims to investigate and define the considerable challenges facing trusted and cost-effective resource allocation in cloud computing environments. In addition, it reviews and critiques existing optimisation works for resource allocation in terms of solutions and approaches to address the existing challenges. After that, it will provide some investigations and results using CloudSim as a cloud simulator explaining some implications that have been found and how they affect the allocation processes. Also, it shows which allocation policies have been used and what configurations that have been set for each components to perform in these examinations. Finally, it will suggest some future work with possible modifications and settings that could help to inspect and what allocation policies that possible to apply to facilitate the allocation operations.

Keywords: Cloud Computing, Resource Allocation.

1. Introduction

Cloud computing is increasingly attracting both academic and industrial communities. Moreover, cloud computing has emerged with a range of different deployment models: public, community, private and hybrid [1]. Cloud architectures have been designed as Internet-based resource providers that offer services through different models: Infrastructure, Platform, and Software as services (i.e. IaaS, PaaS and SaaS) [2-4].

This variety of deployment and service models allow a flexible choice for businesses to meet their requirements and satisfy their needs [5, 6]. However, there are some limitations and issues using cloud computing resources lead to time cost and low performance and [6]. There are many works have been done in cloud computing in different areas such as security, data quality, and resource allocation policies.

Evaluating cloud environment has many difficulties such as deployment, testing, configuration, and setting users requirements [7]. To overcome these challenges, there are many simulation tools and applications that can be used to study a cloud environment [8], such as Cloudsim, D-Cloud, PreFail, and Distributed Load Simulation.

This paper will provide a general overview of cloud computing, basic cloud architecture and cloud features. Also, it will show some obstacles that facing cloud computing especially in resource allocation. Then it will present some tools used for testing and deploying cloud computing environment.
Moreover, this paper will review and critique some existing optimisation works for resource allocation in terms of solutions and approaches to address the existing challenges. After that, in related to provide an optimised strategy for resource allocation in cloud computing architectures which offers better trade-offs compared to the existing solutions it will present some examinations that have been done so far using CloudSim as a cloud simulator tool. Also, it will show some results and implications that have been found giving the possible causes and effects on performance, memory and CPU and the overall allocation process. Then to achieve an adaptive resource allocation solution that efficiently satisfies the clients’ requirements, match their needs, and maximizes the utilisation of the benefits provided by the cloud model.

This paper will facilitate to address the following questions: how to avoid creating too many multiple virtual machines (VMs)? What are ideal characterisations for deploying cloud components? Outcomes will expect that resource allocation obtaining better performance, time and cost effective, and less waste in resources. Finally, investigations and results included with some used cases that have been done to get better understand of the current status using some allocation policies.

2. Background

This section gives an overview and the research background with an overview of the cloud computing and its architecture. Also, it will present the most important elements that affect cloud computing behaviour. Then it illustrates some recent works in cloud computing resource allocation. Also, it will briefly present some methods and tools that used maintaining and managing cloud environment and what method will be used in this paper for simulation cloud environments in terms of testing and deploying.

2.1 Cloud Definition

There are a number of different definition for cloud computing, NIST [9] gives a very basic one of cloud computing as “a model for enabling convenient, on demand network access to a shared pool configurable computing resources (e.g. networks, servers, storage, application, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”.

According to NIST [9], a cloud architecture is a combination of five essential characteristics, three service models, and four deployment models (see Table 1). This architecture offers various alternatives in terms of establishing cloud services to each service provider and users to meet their desirable from each model. This allows service providers to launch reliable services, shared resources and software and information based on service level agreement (SLA) to users, via a cloud network [10].

As mentioned in [1, 11], each service models provide different capability, such as Software as a Services (SaaS) gives customers chances to use applications that running on the cloud. On the other hand, Platform as a Service (PaaS) lets users deploying and creating or obtaining applications that in implemented by programming languages and tools provides by services manager. In the Infrastructure as a Service model (IaaS), data storage, operations provisioning, resources management and networks can be controlled and supervised by users and providers. The four deployment models public, private, community and hybrid can be defined by the
availability of using cloud services [4], if the cloud services available only to particular group then it called private cloud. A cloud services that accessible and available to the general use called public cloud. Cloud community called on cloud services that shared between limited groups shared similar concerns. The hybrid called when there a multi cloud combinations for example joining services for private and public or community cloud [4, 11].

**Table 1: NIST Cloud Architecture.**

<table>
<thead>
<tr>
<th>Cloud Computing Architecture</th>
<th>Model Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Essential Characteristics</td>
<td>On demand self-service</td>
</tr>
<tr>
<td>Service Models</td>
<td>Software as a Service(SaaS)</td>
</tr>
<tr>
<td>Deployment Models</td>
<td>Private Cloud</td>
</tr>
</tbody>
</table>

To stand on issues and challenges that facing cloud computing resource allocation, there is a need to briefly present a general view of using cloud computing benefits for service provider and users. Also, to establish a cloud service there are three elements should be considered which are time, performance, and cost [12]. These elements are related to each other to measure cloud computing environment activities to meet providers and customers’ requirements.

Figure 1 shows cloud users concerns, which are how to get accessing to services with specific requirements needed in any convenient time, and provided in a good quality and operated in high performance, without any considerations to where the services hosted or how they provided [10, 12].

![Figure 1: Cloud Customer Requirements](image)

Against using a usual data centre as showed by [13], cloud computing has more advantages in aspects such as on-demand services, convenient resources, optional term use, cost-effective, higher utilisation in terms of using many cooperative
workloads, and less complexity operations.

2.2 Resource allocation

With all cloud computing features, providers must have very well-organized and cost-effective services [10]. That could lead to use several techniques to manage resources inside cloud environment to get high standard services [12]. One of this techniques used in cloud computing is resource allocation, which is defined by [10] as “the process of assigning available resources to the needed cloud applications over the internet.”

Resource allocation can use either static or dynamic resources management that can provide sufficient use of cloud resources to meet requirements even with some limitations exist such as cloud with a fixed number of resources, memory, virtual machines and applications.

Furthermore, using resource allocation techniques can avoid resource conflict, which means it can avoid accessing resources in the same time. Also, it can manage limited resources by handling the high demand and request.

Multiple software and tools used for testing and managing both cloud and self-hosting environments [8]. However, one of main differences is unlimited software resources used in cloud computing environment via software sharing features that does not exist in self-hosting environment [14]. Also, using software in cloud environment provides better performance in terms of scaling with loading trends high or low using dynamic techniques which is different than hosting self-hosting environment [8].

There are many different methods for testing and maintaining and implementing tools for cloud systems such as simulation, service mocking, test job parallelisation, and environment virtualisation [8]. As mentioned in [8], each method is different in the purpose of using, and what results are needed. Simulation as a technique used to decrease complications and gives better system testing quality. That would be more essential to focus on problems for each cloud features, and it allows to apply different cases to conduct a better cloud system.

Another method is service mocking, which is more capable to be used in service interface level to extend utilities from several providers. Test job parallelisation is a method that splits tasks and let them run individually, and that would be more efficient in time and cost but not for running the all service together. Last method is environment virtualisation, which helps to maintain and testing cloud environment by using virtual machines. That could be beneficial for fast processing and to get better testing cost.

Many simulation tools and frameworks have been implemented and developed for cloud computing such as CloudSim, which is a toolkit used for modelling and simulation of cloud computing environments and evaluation of resource provisioning algorithms [15]. That will help for investigating and testing and deploying a cloud computing before applying it in real environment. Then it will support to obtain a better standing on cloud performance and overall activities and understanding issues and challenges that need to be optimised.

CloudSim as a tool has been designed to simulate the cloud environment to help perform examinations and investigations in different situations [15]. Also, it is structured to several layers that useful to show operations for each cloud components and how they work and support provisioning and analysis all cloud elements (see Figure 2). Moreover, according to [16] CloudSim provides many opportunities and aspects that can cover most cloud activities as follow:
• Simulating the physical hardware like processor, memory, and bandwidth.
• Simulating virtual machine characteristics.
• Managing virtual machine, allocated physical resources based on different policies such as time sharing and space sharing policies.
• User programs executions and requested cloudlets can be simulated on the virtual machines.

One of challenges that facing cloud computing resource allocation is managing resources during the running time such as the allocated virtual machines [12]. This considered as a key element that cloud computing environment rely on, and it can affect services performance and time concerns [16].

2.3 Related Work

There are a various fields in cloud computing needs to be optimised and develop, which let most researchers thinking and developing to get providers and users expectations. As an example of works has been introduced in some areas in cloud computing such as security, Watson in [17] offered a multi-level security model of partitioning the workflows over federated clouds. Also, the paper explains the proposed model and how it could be more secure against any security breaches.

Another example of cloud computing area is data quality which is presented by [18]. The authors proposed a new schema that support data security with efficient operations such as read, delete, update and insert on the data. The last example is about resource allocation policies, which introduced by [19, 20]. The authors included a comparison of static and four heuristic dynamic policies, and showed some differences and presenting benefits and weaknesses of using each type in terms of using and managing cloud resources.

Qiang et al. [21] offered a solution for improving the performance by making resources virtual based on virtual machine, which makes all hardware resources in

![CloudSim Architecture](http://ukpew.org/)
public as shared space. In addition, they aim to regulate many resources utilisation of service level objective of applications SLOs. Jiayin et al. [2] propose an algorithm that adjusts resource allocation based on updating the actual task executions which helps to recalculate the finishing time that assigned to the cloud.

Walsh et al. [22] propose a solution about dividing architecture to two-layers (local and global). The local layer is responsible for calculating the utilities. Where, the global layer computes the near optimal configuration for resources based on result that provided by local layers. This solution is considered as static way of resource allocation and it implemented to fix the load balancing with the server cluster which also helps applications scalability [23].

Yazir et al. [24] present a new approach that has been introduced for dynamic independent resource management in cloud computing which consists of a distributed architecture of NAs. In addition, it performs resource configurations using Multiple Criteria Decision Analyse MCDA with the PROMETHEE method. This approach is more practicable with large amount of data centre [23].

According to Goudarzi et al. [20, 25], there is a problem considered of resource allocation, which is the optimising the total profits that gained from multi-dimensional for multi-tier application. Their aimed is to apply resource consolidation techniques to consolidate resources determining the active servers.

In [19, 20] the paper indicated one of most common static resource allocation policy. This policy in the allocation process starts with allocating jobs with the best feasible cost and load. Then it tries to repeat the allocation to reduce the cost for servers with switching technique to get the best cost. If it does not get the best cost it will remain on the current allocation, which means it the best cost and no need for further allocation.

Also, this paper included four dynamic heuristic policies which are (the average flow heuristic, the on/off heuristic, the queue size heuristic, and the load heuristic). All these heuristic polices are related but three of them are more testing to reduce the cost and one more for improving cloud allocation performance.

3. Problem and Justification

From the research point of view, there are some limitations that have been found in some approaches such low performance, time cost, users’ authentications, resources status, and loading time and balance in using cloud resources. For example in Tian’s system [26] issues and limitations could be as following:

First, the variables affect the systems are users’ authentication and resources status due to time waste. For example, the time needed to make the authentication process affects the performance and the load time. In addition, renewing the use of the allocated resources will affect the system performance as well.

Second, the system confounds between users data and connection managers and that can affect the performance and the cloud virtualisation. Finally, a user for each time needs to renew his request to use the allocated resources. Then the process will go over again which is time cost.

Furthermore, there are some approaches that based on the static solutions with fixed configurations of resource allocation and some are based on dynamics. Static solutions can help in some issues such as applications saleability and loading balance of the web server. Otherwise, the dynamical solutions have not discussed these issues because they need a minimum response time and high level of reliability from web applications [23]. In addition, dynamical solutions have discussed the cloud
virtualisation and reducing the time cost [23].

From [19, 20] the static allocation policy can be a simpler solution. The proposed four dynamic policies also considered in different scenarios especially with greatly loaded system. Three dynamic policies (the average flow heuristic, the on/off heuristic, the queue size heuristic) can be applied to save cost compared with the static heuristic policy. However, there are some points which need to be determined for the performance policy to optimise cloud resource allocation performance. Comparing all contributions in these fields presents there as on why the large number of cloud resources need to be efficiently managed for allocation, reallocation, and balancing resource access.

Desirable resource allocation efficiency is facing some inherent challenges such as unpredictable demands, on-demand resource provisioning, and dynamic availability considering time-variant and high energy costs of data centres. In order for improving resources allocation within clouds, such stated challenges and considerations should be addressed, towards achieving an optimised resource allocation model. Improving resource allocation should meet the changes in policies of resource management and works through the current network traffic situation. Hence, there is a need to pay more attention and spend extra efforts against the resource allocation challenges. Therefore, this paper is focusing on optimising trusted and cost-efficient resource allocation in cloud computing.

4. Experimentation and Results

As efforts of this paper that will be restricted to the investigation of resource allocation issues and the relevant optimisation aspects and heuristics in cloud computing. Also, it focuses on currying out architectural principles of enhanced resource allocation. This section will include investigations and results from experiments have been done so far with some used cases that in focus. The main aim of these investigations is how to avoid creating virtual machine that not necessary in the allocation process, and to try to determine the ideal characteristics for each component.

The experiments have been done so far used cloudsim as a simulator tool and uses time share and space share policies with fixed characteristics properties for VMs, cloudlets, file size, and dedicated RAM. Within cloudsim simulator, there are some components that perform specific task in the allocation process such as cloudlet, VM, broker, Datacentre [15]:

- Broker is a handler in middle that submit VMs to hosts and cloudlets to VMs, then it resetting all configurations to original state before the simulation process starts.
- Cloudlet is the requested application submitted via broker to run on the VM.
- VM is a space that used for provisioning and executing applications and cloudlets.
- Datacentre is a resource that contains all physical hosts, and used to monitoring and provisioning all operations.

Use Cases:

CloudSim has been used as a simulation tools for simulating and investigating. There are some cases have been considered as a result of several examinations using CloudSim. As mentioned before, examining policies such as time share and space share in terms of getting time saving, better performance and to be more cost
There are some objectives that have been considered in these investigations such as the number of VMs used in the allocation process, balance in allocation tasks between VMs, some allocation failure, and high performance and memory fault. The investigations cases as follow:

**First Case: Investigate Allocated VMs**

This examination started by investigating the basic allocated VMs that have been created when the service has been initiated, then to see how many VMs are used in the allocation operations. As a result, some VMs have not involved in the allocation process and it could be a reason of wasting some memory that used in the allocation process. This considered as a vital issue in cloud computing resource allocation and affects the overall operations which cause some costs by wasting resources (see Figure 3).

![Figure 3: VMs and allocated cloudlet](image)

**Case Two: Examine Allocation Effectiveness**

In this case, experiments have been proceeded to investigate the allocation effectiveness between the created VMs in the allocation operation. The main point of this experiment is to show that in some point during the allocation process, there was no balance in dividing task allocations for the assigned VMs. As a result, it could cause some performance, time, and cost issues that related to cloud resource allocation (see Figure 4).
Case Three: Allocation Failure

This test is about managing the size of VMs and requested cloudlets, and it has been found that if the requested clouds size is greater than the allocated VMs size will cause an allocation failure. This will cause some performance issues and delay of the cloud services.

Case Four: Performance and Memory Fault

This experiment used time share policy which tries to allocate the requested cloudlets to the available VM with a very large number of cloudlets requests. Also, the physical machine has been configured to the maximum performance level which helps to speed up the operation, and then setting the simulator process on high priority to perform the allocation process in appropriate time.

As a result of these examinations, that setting these configurations have caused some implications on performance, memory and CPU. In addition, the reason of getting low performance is the allocation process tries to allocate cloudlets to VM as fast as it can but with the large number requested it does not have enough space to use, and that causes an operation delay. That has led to conflict which makes the memory fault and high CPU usage.

5. Conclusion

This paper highlighted the growing of using cloud computing for both education and industrial sectors. Then it showed the cloud computing architecture clarifying each component. Next, it explained why resource allocation in cloud environment is an issue to service providers to be considered and presented some recent work that related to the resource allocation.

Then it shows what possible tools and applications used to simulate cloud environment. After that, some investigations and results have been introduced with some used cases defining what allocation policies have been used such as time share and space share policies that used in examined simulations. Results from cases such
as creating VMs that not needed and not involved in the allocation process should be considered in terms of saving cloud resources.

Also, to get a better service performance there is a need to pay attention to balancing on dividing requests to the available VMs for more effectiveness allocations. Finally, pushing the allocation process to high performance level can cause some implications such as memory fault and conflict in allocation process which can delay the operations.

6. Future Work

For future work, experiments and investigation will be continued in terms of finding an optimised solution that simplifies the current implications. For further improvement in allocation process suggestions will be considered as follow:

- Characteristics and configurations for each component involved in the allocation process might be change such as VMs and cloudlets and memory that dedicated for allocation process.
- Performance level and process priority of the simulator are possible to set to proper level for testing investigations.
- Applying other allocation policies such as using switching policy to balance allocated cloudlets between VMs which mentioned by [20] in terms of optimising the allocation process that could help with current situations of resource allocation.

References


Universal Performance Assessment of Cross-Platform Mobile Applications

Andre Nitze∗; Andreas Schmietendorf†

Abstract
Mobile applications (apps) are being used more and more often to enable or support important business processes. The performance efficiency of these apps is pivotal to the user experience and has a tangible impact on revenues. But mobile devices are very diverse in terms of operating system versions, hardware features and ecosystems. This aggravates software quality assurance and in particular performance testing. Hence, in this paper an universal approach for the performance assessment of mobile business applications is presented. It is based on the decomposition of typical processes into atomic steps and their measurement on source-code-level.

1 Motivation and Aims

Many business processes can be improved or even enabled by use of mobile applications (apps). The ubiquity of mobile devices enables instant and convenient interaction between employees, business partners and customers. Many businesses have already employed mobile apps for various tasks in their supply chain. The impact of software on the core processes of an enterprise can be considered extremely high. With mobile apps being increasingly used to support these processes, the importance of high quality mobile applications is obvious.

The quality of mobile apps is –besides functionality– primarily defined by the user experience. The user experience in return is to a large extent dependent on the performance of the application. Important tasks and work-flows are suspended if the respective apps do not respond in a timely manner. This results in decreased process quality and efficiency, decreased customer satisfaction and eventually less revenues. For quality assurance this means an increased demand for thorough performance testing of the software product to ensure that the performance is appropriate to the use case and user expectation.

In previous research [8, 7] the need for a universal benchmark has been stated. In this paper we follow that call and propose an approach to measure mobile application performance across platforms and benchmark results of different endpoints. The key question is how mobile applications can be tested for performance bottlenecks which could potentially detract business users from accomplishing the task at hand.

*Faculty of Computer Science, Otto-von-Guericke University Magdeburg, andre.nitze@hwr-berlin.de
†Faculty of Computer Science, Otto-von-Guericke University Magdeburg, schmiete@ivs.cs.uni-magdeburg.de
In practice, tool support for performance management is readily available and in comparison to other mobile development tools disproportionately mature. There are on-premise and Cloud-based load testing solutions, etc. Some analytic tools focus on marketing and user behaviour and lack insights for developers and product managers on performance criteria. However, in general, existing tools and solutions from the performance management practice can be used for measuring durations, number of requests, memory consumption etc. The measurement and instrumentation details are therefore not part of this paper.

2 Related Work

The basics of mobile application development have been described extensively by Salmre [9]. Although the mobile market and its technologies are changing rapidly, the basic software engineering tasks and methods remain similar.

According to Wasserman, performance is one of the most important factors for the perceived quality of mobile applications (s. [12]). This point is supported with an empirical evaluation in the next section.

In [5] Kurbel et al. state the importance of mobile solutions in business environments and discuss a rudimentary multi-tier architecture for mobile enterprise resource planning (ERP) applications. In this paper, the same business context and architectural and technical preconditions are assumed.

According to the “Performance Golden Rule” following Souders’ analyses (s. [10]) ‘80–90% of the end-user response time is spent on the front-end.’ Although Souders analysed web sites, the implications should be comparable to the ones of mobile apps due to the convergence of both application types in recent past and the notion of multi-tier architectures in general.

Table 1: Exemplary component-based metrics for measuring mobile application performance according to [3]

<table>
<thead>
<tr>
<th>Component</th>
<th>Monitoring Parameters (Not Exhaustive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>CPU usage</td>
</tr>
<tr>
<td></td>
<td>Load</td>
</tr>
<tr>
<td></td>
<td>Process time</td>
</tr>
<tr>
<td></td>
<td>Bytes total</td>
</tr>
<tr>
<td></td>
<td>User time</td>
</tr>
<tr>
<td></td>
<td>Packets sent/received</td>
</tr>
<tr>
<td>Network</td>
<td>Packets and bytes sent</td>
</tr>
<tr>
<td></td>
<td>Packets and bytes received</td>
</tr>
<tr>
<td></td>
<td>Average delay</td>
</tr>
<tr>
<td></td>
<td>Packet drops</td>
</tr>
<tr>
<td>Device</td>
<td>CPU and memory usage</td>
</tr>
<tr>
<td></td>
<td>Method level profiling</td>
</tr>
<tr>
<td></td>
<td>Web application component level performa nce</td>
</tr>
<tr>
<td></td>
<td>Response times</td>
</tr>
<tr>
<td>Transaction</td>
<td>Response times</td>
</tr>
<tr>
<td></td>
<td>Throughput</td>
</tr>
</tbody>
</table>

Gawande and Rudagi categorized the requirements for mobile application
testing and how to recreate these conditions in the testing environment (s. [3]). They also identified the required components to analyze for an end-to-end performance assessment of mobile apps. Table 1 shows these components and corresponding metrics to assess performance. On the network level, for example there are several factors like geographic origin of traffic, network type (2G/3G/WiFi etc.) and connection quality or network load (e.g., 50% bandwidth utilization).

Many works have been published on the low-level networking aspects (bandwidth, latencies, ad-hoc networks etc.) and how to optimize those, e.g., in [6]. Thompson et al. proposed a model-driven approach to change the architecture of mobile applications and then estimate power consumption and performance using instrumented emulated code (s. [11]). The authors focus on power consumption due to network access which is a low-level consideration needed for higher-level approaches like the one stated in this contribution.

3 Empirical Evaluation

3.1 Others’ results

There have been several reports and surveys from mobile vendors concerning mobile quality aspects, especially performance. In 2011 a Compuware survey with 4,014 participants found that mobile usage reached a critical mass with online retailers and other companies starting to provide more mobile-friendly services (s. [2]). A Google online survey from 2012 with 1,018 respondents focuses on mobile website performance and states that users are dissatisfied with non-mobile-optimized or slow mobile websites (s. [4]). Both reports found that the abandonment rate increases with the page load time due to the fact, that there is always urgency in mobile access which is leading to very low user patience. The World Quality Report 2013 also found performance efficiency to be the highest priority while testing mobile applications (s. [1]).

3.2 Own results

In a survey conducted by the authors [7], 144 mobile application users expressed their expectations on app performance. When participants were asked (unsupported) for what they associate with a high-quality app, the responses referred to usability (41%), performance (33%), robustness (30%), functionality (23%), look and feel (21%), and data privacy (11%) (s. Figure 1).

To assess the expectations of mobile users in regard to performance, another question was directed at the amount of time users are willing to wait for an application to respond to a user request. Half of the participants (45%) selected a response time of 3 to 5 seconds with another quarter of the respondents willing to wait longer than 5 seconds (s. Figure 2). 13% of the respondents were less generous. – They would only wait 1 or 2 seconds for a response.

These results capture a high-level view. The willingness to wait for certain tasks will probably differ from case to case in every app and therefore could be evaluated more nuanced. But the results support the need for performance assessments in general.
Figure 1: Quality expectations of mobile application users

Figure 2: Response time expectations of mobile application users
4 Concept of a Universal Performance Assessment

In this section, a concept of a universal assessment framework for cross-platform mobile applications is presented. After stating the requirements of such an approach, exemplary mobile test cases are provided before the measurement method is described in detail.

4.1 Requirements

The requirements for the assessment of mobile application performance can be stated as follows:

- Technology independence
- User-oriented assessment
- End-to-end measurement

An independence of technology in terms of platforms, programming languages and hardware is necessary to ensure consistent modeling, long-term comparability and applicability of performance assessments.

An user-oriented assessment requires a process consideration in form of use cases, user stories or the like to incorporate the users’ perspective on the app performance. These well-defined interactions must be detailed enough to be measurable in the source code and large enough to be meaningful to the user.

An end-to-end measurement approach is needed to enable holistic development and management decisions on quality improvement measures on all architectural levels, e. g., to split back-end services into smaller units to improve response times for specific mobile endpoints which require only a small part of the provided data.

A basic assumption is the technical measurability of the desired metrics in a practical and efficient way. Although this can be hard at times, it generally can be considered a wise investment having appropriate means of monitoring and reporting performance metrics.

4.2 Typical Test Cases

It is necessary to provide a basic set of common test cases that can be applied to any mobile platform. These test-cases can be specific to the organization and may be derived from user-centered requirements documents. Well-written requirements will already include acceptance criteria regarding performance behavior. If there are no thresholds for corresponding performance metrics (response times, throughput etc.), these must be defined explicitly beforehand.

Due to platform-specific differences the test cases may vary for the mobile target platforms. The test cases may also be specified with different performance objectives based on the empirical data (see above).

The more generic –yet useful– the test cases are worded, the more they can be used for benchmarking and comparison. But in the most simple form of the approach this can be omitted for the sake of easy implementation.
4.3 Process Compositions

The interactions specified by the requirements can be subdivided into transactions which can be used to compose more complex interaction processes. These transactions are atomic in the way that further subdivision would not increase the precision of the performance assessment.

Prior to the composition, minimal processes like fetching a resource from a server or conducting a calculation or transformation must be identified and formally specified within a model. Modeling in software engineering helps to abstract from the implementation details like the programming language and focus on the longer-lasting concepts like service-oriented architectures. For the modeling of mobile interactions with regard to performance measurement, sequence diagrams seem to be an adequate mean. Sequence diagrams are not only easy to understand but will in most projects already be available in some form.

Figure 3 shows a sequence diagram with an example interaction of the user with the mobile device and the necessary method calls. The transaction involves the validation of login credentials using a third party service and shows.

![Sequence diagram of an end-to-end mobile interaction (user login)](image)

Figure 3: Sequence diagram of an end-to-end mobile interaction (user login)

5 Mapping Process

The mapping of requirements with the source code via sequence diagrams is shown in Figure 4. The mappings must be formally specified and stored for
later comparison with the desired performance objectives and further analysis of the effectiveness of quality assurance efforts.

The sequence diagrams contain the single interaction steps and to provide an end-to-end-view on the complete mobile interaction from front-end dialogs to middleware and back-end servers.

On the lowest level the performance is measured in a debugging-like manner (performance profiling). This can be done by instrumentation of the source code via annotations or automatic static analysis with existing tools.

Figure 4: Conceptual approach of the universal performance assessment from requirements to source code

6 Conclusion and Future Work

Enterprise users depend on the quality of the provided mobile applications and in particular the performance of these apps. With the presented approach, performance measurement can easily be improved by mapping the mobile interactions via requirement documents, sequence diagrams and performance measurement on source-code-level.

The benefits of the approach are its’ long-time applicability due to the technology independence and the sequence modeling approach. It can be used for benchmarking across multiple platforms and technologies, e.g. to compare different technologies and frameworks, but also performance optimization techniques like minification or cross-compilation.

The major challenge seems to be the automation and collection of source code performance measurements in the diverse mobile platforms to form a consistent
view of the performance efficiency of the applications.

A feasibility study with a prototypical implementation in an industry scenario is in preparation as of writing.

References


UKPEW 2015 – http://ukpew.org/

Evaluation of automated static analysis tools for malware detection in Portable Executable files

Anitta Patience Namanya*, Jules Pagna-Disso†, Irfan Awan*

Abstract

The rate at which malware samples are released is much higher than the rate at which malware analysts can fully investigate. CNN reported that nearly one million malware threats are released daily. It is however known that most of the new samples discovered are repackaged or simply variants of already known malware. With a list of known fingerprints of malware, detection of the variants would reduce the number of unknown malware, increase detection rates thus reducing the impact of cybercrime. Whilst detecting malware may require both static and dynamic analysis, efficient static analysis remains safer and a key first step. In order to understand what characteristics are required to detect malicious files, we analysed 2269 samples of malware using three of the most popular open source malware static analysis tools: PEFrame, Pyew and mastiff. Whilst the comparison of the tools will inform us what tool is best depending on the given scenario, the file feature extraction will allow us to obtain fingerprints against which files can be tested to determine whether they are malicious or not. This paper presents our findings of the work described above.

1 Introduction:

CNN reported earlier this year that nearly one million malware threats [1] are released each day. We expect this number to keep increasing as cybercrime takes advantage of increased dependency on technology due to the big drive towards the Internet of Things. There is a need for faster detection to reduce the impact of malware if people are to benefit from the evolution of the Internet of Things. Since more than 90% [2] of computer users still use Windows Operating Systems based computers, this work focuses on Windows executable files and detection of malicious files in the windows environment is normally performed on the executable file. Static malware analysis allows for malware to be analysed without the need to execute it[3] and this is always the best starting point for malware analysis and therefore detection. For one to perform analysis and extract as much information as possible, there is need to understand the structure of the file format to be analysed.

Malware detection requires both static and dynamic analysis and the work by Egele et al [4] covers most of the present dynamic analysis techniques and discusses the different analysis programs and tools that use these techniques. There is a need to explore the static analysis techniques especially with the recent release of numerous automated static analysis tools which have given the cyber security community a much needed boast towards efficient static malware analysis. Although Ligh et al [5] provides insight into many of these tools, there is still a need for a detailed evaluation of some of the more recent prominent tools available today.

* School of Electrical Engineering and Computer Science, University of Bradford. anamanya@student.bradford.ac.uk, I.U.Awan@bradford.ac.uk
† Nettitude Intelligent Cyber Security and Risk Management. jpagnadisso@nettitude.com
The main contribution of this paper is to provide an evaluation of three prominent, open source malware static analysis tools focusing mainly on the analysis of Portable Executable files. It provides an overview of the automated static analysis tools in the first section, the test environment in the second section and the 9 test scenarios with emphasis on PE features to perform a comparison of the tools and extract information from the samples. We then present the findings of the scenarios and the tool feature comparisons and present a brief summary of the fingerprints found which can be used as indicators of compromise in executable files.

2 Overview of automated Static Analysis tools.

This section provides summarised overviews of the automated static analysis tools that are being evaluated in this study.

2.1 Peframe

Peframe is an open source, command line based static malware analysis tool written by Gianni Amato that extracts information from Portable Executable files [6]. It is written in python and uses the pefile module written by Ero Carrera [6] and the Anti-Virtual Machine Signatures written by Joxean Koret [6]. In its folder called ...

Figure 1: Hex-Dump of entrypoint of file md5-a3c5e50c55c901767b0c3b7749a48c9b

The file is analysed as a hex-dump and the signature highlighted in Figure 1 is identified to be identical to the signature in the userdb.txt and in the results, it returns that it has identified the presence of the packer UPX 2.93 as shown in the extract of the report below:

```
Packer matched [3]
```

```
Packer
UPX 2.93 (LZMA)
```

Figure 1: Hex-Dump of entrypoint of file md5-a3c5e50c55c901767b0c3b7749a48c9b

The file is analysed as a hex-dump and the signature highlighted in Figure 1 is identified to be identical to the signature in the userdb.txt and in the results, it returns that it has identified the presence of the packer UPX 2.93 as shown in the extract of the report below:

```
Packer matched [3]

Packer
UPX 2.93 (LZMA)
```

Code 1: Peframe Report Extract of the file

Although the results are described in the command line, it provides the option of printing the results to a text file which can be manipulated further for analysis.

2.2 Pyew

Pyew is a static analysis framework written by Joxean Koret that performs file and code analysis on PE, PDF, ELF and OLE2 file types [7]. It is mainly command line based although the tool bokken which has a GUI can use it as a backend to provide
better user interaction. Pyew uses the standard Pefile.py module to read the file contents of the PE File. This enables it to read PE structure and display the contents depending on the command used. It also provides debugger properties without the need to install a debugger. The scripts of interest for a malware analyst are found in the folders:

...\pyew\plugins

Vmdetect.py contains some signatures of known anti-VM tricks which are used to detect the presence of these tricks in the file.

Virstotal.py is the script that searches the virustotal website for a report on the file being analysed using the MD5 and prints out the report it retrieves.

UserDB.txt: It is a copy of the PEiD packer detection signatures used by Packer.py to detect the presence of a packer.

...\pyew

gcluster.py: A new module that is not seen in any of the other tools which uses the graph vertices to compute the similarity between the call graphs.

graph.py: The script that retrieves the call graph of the file and an example of the extract can be seen in the Figure 2

Figure 2: Call Graph of a sample malware
Using Pyew analysis is limited as it requires additional scripting to allow for automated multi analysis and requires for changes to be applied to the modules to get output logs that can be further manipulated as the information saved in the SQLite database is very limited and not of great value during the analysis.

2.3 Mastiff

Mastiff is a framework developed in python that performs static analysis of files and is command line based. It is designed to extract the characteristics of the file by automatically identifying the type of file being analysed and using the right techniques to analyse it. Mastiff relies on plugins which makes it easy to be extended for further use and can be used as a building block in the design of other frameworks.

Figure 3: Mastiff Work Flow [8]
Mastiff functions and workflow as seen in Figure 3 are further documented by Hudak [8] with details on how the different modules work together to provide an in-depth analysis of the file. Mastiff supports the analysis of different file formats but the plugins of interest are found in the folders ...

...\mastiff\plugins\analysis\EXE and ...

...\mastiff\plugins\analysis\GEN:

EXE-peinfo.py which is the script that extracts and dumps information the PE header and the structure of the executable.

EXE-resources.py extracts the information on the PE Resources directory

EXE-sig.py extracts the PE digital Signatures in the file.
EXE-Singlestring.py extracts single byte strings found in the file.
GEN-fuzzy.py which extracts the fuzzy hash of the file and files in the sample that match.
GEN-String.py extracts and decodes any of the embedded strings found in the code.

Mastiff provides an option of using the Virus Total API so that the files uploaded can be analysed by the virus total website and the report generated downloaded.

3 Test Environment

This section discusses the steps taken to setup the analysis environment in order to ensure validity and reliability of the results. It also describes the evaluation and analysis approach used in the study.

3.1 Environment Setup

Table 1 provides the details of the different machine specification, environment, files and tools used.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Specifications/ Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer</td>
<td>Dell 745, CPU – Intel duo core @ 2.13GHz, RAM 2GB. Hard Disk – 150GB</td>
</tr>
<tr>
<td>Host Machine OS</td>
<td>Windows 7 Professional N Service Pack 1 (64 bit)</td>
</tr>
<tr>
<td>Virtual Machine Manager</td>
<td>Oracle VM VirtualBox Manager Version 4.3.20 r96997</td>
</tr>
<tr>
<td>Virtual Appliance</td>
<td>Honey Drive 3 – Royal Jelly installed with 80 GB vmdk space</td>
</tr>
</tbody>
</table>
| Tools under analysis | Pyew version 3X  
Mastiff version 2.0  
Peframe as seen on the Github accessed April 2015. |
| Data set        | 2620 samples of malicious PE files were downloaded from  
http://www.nothink.org/honeypots/malware-archives/ |
| Data management tools | SQLite Studio version 3.0.6.  
Python IDLE version 2.7.9 |

Table 1: Experiment Setup Specifications

3.2 Comparison, Analysis and Evaluation Approach

The dataset was downloaded and analysed to pick only PE type files. Since the dataset included many malware samples which could only be analysed one at a time by any of the tools it was necessary to write scripts to automate this process. With all the malware files saved into a single folder we were then able to analyse each file and dump the results as a text file into another folder.

![Figure 4: Pictorial representation of study Approach.](image-url)
By modifying the script written by tekdefense in [9] different scripts were written to enable auto multi-file analysis for Mastiff, Pyew and Peframe. The script used for Pyew also utilised its call graph module and call graph clustering module in order to obtain results necessary for this evaluation. Data analysis was performed on the information retrieved in order to obtain meaningful results. Figure 4 shows the study approach taken during this study.

4 Test Scenarios

The test scenarios were formulated based on the information required to be extracted from PE files in order to detect any suspicious characteristics and the additional features that the tools provide that add value to the detection process. This section discusses the test scenarios chosen and the reasons to why they were deemed important.

4.1 File Identification:

Fields that identify the files such as the Filename, File size, MD5, SHA and SHA256 are important to extract because these can be used to check the integrity of the file downloaded. In addition to being used as identifiers when performing further analysis, they can be used to query existing databases for known malware.

4.2 Detection of Obfuscation Techniques

Obfuscation is a major characteristic of many malware as they try to evade detection or slow down analysis. Detection of some of the signatures that show that an application is using one or more of some of the obfuscation techniques may lead to detecting a malicious file. For this test scenario, we will look at three common methods applied in the binary obfuscation techniques as seen in malicious files:

- Packers

Most malware writers apply packers or even multiple packers to produce different variants of the same malware code. Perdisci, et al [10] states that more than 80% of the new malware discovered are actually packed versions of already existing malware. Packers compress the file into a smaller size and sometimes encryption is applied to the compressed version of the file to make the unpacking process more difficult. "The packer program automatically transforms an executable into a syntactically different but semantically equivalent representation" [4] as seen in Figure 5. Some packers are custom built by malware writers and these can be used to actually detect that the file is malicious without the need for further analysis while commercial packers that are readily available online are seen in many malware variants.

**Figure 5: Structure of a Packed PE File[11]**

**Figure 6: Call Graph (G) Structure***
A debugger is a program that allows one to observe the rendered code as it runs and
the most basic features include the ability to set breakpoints and trace through the
executable code[12]. This is because most programs are too complex for a human to be
able to trace and predict all the possible execution paths. To frustrate such efforts,
malware is written to detect the presence of debuggers and then either give the wrong
output or unexpected events. Common debuggers are IDAPro, OllyDbg, Immunity
Debugger and WinDbg among others. Methods employed by malware writers to detect
debuggers are further discussed in [5]and [10].

Running malware in a virtual machine is a common and safe method used to analyse
the behaviour of the malware as it theoretically infects the virtual machine and never
the host. Virtualisation also enables faster analysis times than installing new hosts
every time one needs to examine a new malware sample. Since malware analysis is
done in an isolated environment, malware have found a way to detect whether they are
running in a virtual setting. Some of these tricks are discussed in [13-15]
With the tricks known, analysts have developed signatures that detect specific packers,
anti- VM and anti-debug obfuscation techniques. So the results obtained from the tools
will be analysed to determine the ability of the tool to detect; packers, Anti-VM and
Anti- debug tricks in the file information. The tools are also expected to identify the
specific packer type where possible which should assist in the next step where
unpacking is required.

### 4.3 Analysis of APIs

The API calls extracted from a PE file can highlight the expected behaviour and
characteristics of the file[12]. Some APIs may indicate that the file employs some
obfuscation technique and how the file would interact with the system.

Some of the APIs used for anti-debug detection for example are [16]

- **IsDebuggerPresent**: This function checks whether the application is being
debugged by returning a non-zero.
- **NtQueryInformationProcess**: Returns internal Operating System structures
related to the process passed. This function is no longer available for newer
versions of Windows but it can still be seen in some malware samples
- **CheckRemoteDebuggerPresent**: Returns a non-zero value if the process
passed to the function is being debugged.
- **OutputDebugString**: The function sends the debugger a string to display.

Some other notable characteristics are if the file expects to connect to the internet, how
it will execute and/or how it accesses memory. Depending on the APIs clustered, files
can be partially grouped into clusters of predicted behaviour so extraction of APIs is a
very important feature of an analysis tool. For this field, two sub-definitions shall be
considered; the extracted suspicious APIs according to the tool and the general
extracted APIs.

### 4.4 PE Header analysis

Many of the PE file fields have specific standards set by Microsoft [17]so changes in
these standards might indicate that the file is suspicious[5], [11], [15]. Some of the
features that can be used to detect suspicious characteristics are:

- **TimeDateStamp**: Some malicious files are known to have this field changed
[5] to reflect an invalid date and time.
- **Address Entry Point**: The address Entry point not pointing to an offset found
in the .text or .code sections might indicate a suspicious file.
- **SizeofUnInitialisedData.** This field is set for compressed sections which shows the presence of a packer. Otherwise, the values are normally set to zero.

- **SizeofRawData.** Malicious files which are packed have this set to zero.

- **Checksum.** In some malicious files, the checksum value given in the file header is not the same as the checksum calculated.

- **Digital Signature.** Presence of a non-zero value for the size of the IMAGE_DIRECTORY_ENTRY_SECURITY field indicates an embedded signature. Extracting and analysing this field can tell a lot about the validity of the certificate used in the file.

- **Section names:** Malicious files tend to have section names that are not the valid PE file section names sometimes. Packed PE files also rename the section names for example, UPX renames the sections UPX0 and UPX1.

- **Section Entropy.** Entropy is used to describe the randomness of data in a data block and the values ranges from 0 to 8. Very high or very low entropy may indicate the presence of a packer or even code obfuscation.

- **Characteristic Flags.** The section characteristic flags dictate whether the section is executable, can be read or written to. Evidence of having the characteristic flag(s) of many sections set to executable may indicate that the file is suspicious.

In this scenario, we will be testing to see how much of the PE file field information is extracted by the tools or if used, whether the tool provides suspicious alerts for the fields that have invalid information in the fields.

### 4.5 SSdeep hashing and Malware clustering

Cryptographic hashing computes a hash value on a data block and any changes in the data block produces different hash value. If one file has the same cryptographic hashing value as a known malware sample, then it is concluded that the file is a copy of the known malware. Since malware variants tend to change some bit values of the original malware code, normal cryptographic hashing like the MD5 fails to detect the similarity of such files. So to counter this limitation, fuzzy hashing uses a rolling window to produce a continuous stream of hash values. These hash values can be compared to produce a percentage score of similarity between the files compared which enables malware analysts to detect malware variants. Many hashing algorithms; ssdeep, imphash and others have been investigated in different works [18]–[20] to show how they improve the accuracy of detection when used. This scenario looks at analysing what file hashes are computed and how they are used in file clustering. One limitation worth noting is that using hashing has a high false positive rate for packed malware.

### 4.6 Call Graph Extraction and Comparison

A call graph is a directional graph with nodes (N) that represent the functions that are interconnected with function calls represented by edges (E(i, j) where i to j define the execution path taken. Extraction of a call graph represented by $G = (N, E)$ provides a graphical representation of the execution process of the program.

Call graphs have been used in different research works to show how they can predict the behaviour of the file [29], [30]. The work carried out by Kinable [21] shows that call graph matching and clustering can be used to detect malware variants. Availability of such information from a tool provides a way to increase the accuracy of the malware detection. The fields analysed in this scenario look at the extraction of code call graphs and comparison capabilities of code call graphs performed by the tool.
4.7 String Analysis

Strings obtained from a file during static analysis when not encrypted can provide a lot of information for a malware analyst, like URLs, executable files, registry key paths, command line options, passwords and IP addresses. Similarity in the information across different samples may be used to correlate them. However, it is important to note that sometimes, the malware obfuscates the strings and may also provide misleading information in the strings.

4.8 Third Party Plugin

We will analyse the ability of the tools being evaluated to support the integration of third party tool as this enables the additional of new features during the analysis of malware.

4.9 Usability

Although this is not a metric that would lead to determining how malicious a file might be, the ease of use for an analysis tool is an important feature to consider. This section shall be broken down into 2 subsections:

- User Interface: Command line, Graphical User interface and online analysis presence are what are considered as the measure for these tools.
- Output data management: During analysis, the most important factor is how easy it is to handle to information one gets from the analysis tool.

We will discuss what was observed as we used these tools to analyse the malware samples collected; the technical skill level required and availability of tools that supplement the analysis of the output files.

5 Feature Comparison, Analysis and Evaluation.

Once the dataset of malware were analysed by the tools, the output data is analysed and results from the three tools are compared against each other including the tool features and usability as observed during the experiments.

5.1 Tool Feature Comparison

This section shows the comparison of the kind of information that the tools extracted and the additional features of the 3 tools. Table 2 provides the summary that would important to know when deciding which tool would be best depending on the depth of static analysis required to be done on a file sample. This information was collected by observation during the analysis phase of this study.

5.2 Analysis and Evaluation

5.2.1 File Identification:

The tools considered in this study provide different information that can be used for file identification. While mastiff logs the file results in a folder identified by the file name, it does not provide the MD5 or other hash values for the whole file. It instead computes the MD5 for each section and appends it at the beginning of each section. The Pyew module provides the option for the tool user to request for the filename, hash values and file name using a script called runme.py that can be edited to automate the request for each analysis. Peframe provides the most information for file identification. The comparison of the file identification information provided by the tool is shown in Table 2 in the general File details section. This shows that Peframe and Pyew are the
stronger of the tools when there is need to immediately get file identification information upon analysis. Detection of Obfuscation Techniques.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Peframe</th>
<th>Pyew</th>
<th>Mastiff</th>
</tr>
</thead>
<tbody>
<tr>
<td>General File Details</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Filename</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>File size</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MD5</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SHA</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SHA256</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Obfuscation Technique Detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Packer</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Anti-Vm</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Anti-Debug</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>APIs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>General APIs Extraction</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Suspicious API extraction</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calculated Hashes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSDeep hash</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>imphash</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>File Clustering based on hash</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>PE File Details</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sections</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Section Entropy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Exports</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Imports</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hex-Dump</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Call Graph</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generation</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Cluster Comparison</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User interface</td>
<td>Command line</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Graphical User Interface</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Online Analysis option</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Output Data</td>
<td>.txt files</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>.json files</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>.db output</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Additional Features</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>String Extraction</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Virus Total API utilisation</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Disassemble</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>File metadata</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Possibility of tool Extension by plugin</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Static Analysis Tool Feature Comparison
For this scenario, we only consider Pyew and Peframe. Pyew detects packers and anti-vm using known signatures while Peframe detects packers, anti-vm and anti-debug based on the signatures provided in the signature folder. Peframe allows for better extension options than Pyew especially for anti-vm and anti-debugging techniques as the text files used as comparison signatures can be edited. Pyew’s signatures save for
the packer signatures are hard-coded in the module. We analyse the results obtained from the tools based on the dataset and the comparison of the detection percentages are presented in the Figure 7.

5.2.2 Analysis of APIs

All the three tools extract the APIs that are identified in the file during analysis. While Peframe allows for only APIs deemed as suspicious based on the signature saved in its comparison file, Mastiff returns a file `string.txt` that contains all the strings identified in the analysed file which contains the APIs and Pyew returns a list of all the APIs when the command to return imports and exports are called. There is a variation in the top APIs detected by the different tools which can be explained by the fact that Peframe is heavily affected by packing where some packers led to the tool throwing an error and it returns only APIs it has matched to be suspicious in its signature database. Pyew and mastiff extract all the APIs and Pyew is able to perform deep code analysis to retrieve more APIs than Mastiff. For this scenario, in the future work, we will analyse a set of benign executables to extract the API calls and perform a comparison on these functions.

![Figure 7: Comparison of Obfuscation Detection](image1)

![Figure 8: Sample Compile time analysis](image2)

5.2.3 PE header analysis

Here, we analysed the information given by the tools that can be used to detect if a file is suspicious or not. For example Peframe extracts the compile time and Figure 8 shows that the compilation year analysis of the malware samples. An extract from a report of a malware sample analysed 6ec7e5c29b87c724735fea3c98b10288 shows that the file has an invalid date and it is also a good example abnormal section names. Code 2, Code 3 and Code 4 show the various report samples from the three different tools of the same sample.

By analysing the three reports, the information provided by the various tools defers in the level of detail. Pyew provides the shortest file report with the section names, the section addresses and sizes which are important when deeper analysis is required. Peframe provide detailed information about the header with the important field of the compile time and also flags it because it detects that the time given is indeed invalid, it then also flags the 3 sections that have names that are unknown together with the hash values. These hashes can be used for hash matching during further detailed analysis. The mastiff report provides findings in more detail even providing alerts based on the discrepancies it has detected. For example like the file having an address
of Entry Point that lies outside the section’s boundaries which a known indicator that the file is malicious, a rawdata size that is larger than the actual file size.

File Name          6ec7e5c29b87c724735fea3c98b10288
Suspicious Sections discovered [3]

Code 2: Peframe report extract for sample- 6ec7e5c29b87c724735fea3c98b10288
PE Information
Sections:
  .k$k2hx_  0x1000  0xbb6  1189
  .tgni4t_  0x2000  0x1e8  0
  .nljhct_  0x3000  0x7f8  0
  .rsrc___  0x4000  0xfa9a  64154
  .reloc__  0x14000  0x3eb00000  0
  .rdata__  0x3eb14000  0x26000  134144

Code 3: Pyew Report
Number of Sections: 6
Section Name    Entropy  Flags
--------------------------------------------------------
.k$k2hx         7.1709   IMAGE_SCN_MEM_EXECUTE, IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_WRITE, IMAGE_SCN_CNT_CODE, IMAGE_SCN_MEM_READ
.tgni4t_        0.0      IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_WRITE, IMAGE_SCN_MEM_READ
.nljhct_        0.0      IMAGE_SCN_CNT_INITIALIZED_DATA, IMAGE_SCN_MEM_WRITE, IMAGE_SCN_CNT_INITIALIZED_DATA

Suspicious NumberOfRvaAndSizes in the Optional Header.
Normal values are never larger than 0x10, the value is: 0x5c515225

Suspicious flags set for section 0. Both IMAGE_SCN_MEM_WRITE and IMAGE_SCN_MEM_EXECUTE are set. This might indicate a packed executable.
Error parsing section 3. SizeOfRawData is larger than file.
Error parsing section 3. PointerToRawData points beyond the end of the file.
Error parsing section 3. PointerToRawData should normally be a multiple of FileAlignment, this might imply the file is trying to confuse tools which parse this incorrectly.
Too many warnings parsing section. Aborting.
AddressOfEntryPoint lies outside the sections' boundaries. AddressOfEntryPoint: 0x3eb32000
Error parsing the import directory at RVA: 0x3eb290ac
Error parsing the resources directory. The directory contains 23387 entries (>4096)

Code 4: Mastiff Report
The mastiff report also provides more warning details like the section characteristic flag warnings, section field warnings, section entropy and directory warnings that can be used by analyst to deduce that a file is malicious based on PE header analysis than
the other two tools. However the information from all the three tools is equally important in order to improve detection accuracy.

5.2.4 SSdeep hashing and Malware clustering

Mastiff is the only tool of the three that calculates the Fuzzy Hash of the file and compares the hash against the hashes of the files already in the database to give similarity percentage in the files. Code 5 shows an extract of one of the reports.

Fuzzy Hash:
3072:YTleUJFD7UNGyjFAxUgCGWk?puc6TKkKpxdQpah72Tf1K7cVMIR
2:YRBjJFUnsEFAxUgWk702WKpzd2zTF5z
This file is similar to the following files:

<table>
<thead>
<tr>
<th>MD5</th>
<th>Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>ae8710006c5033404f4f7a8f8d9f63724</td>
<td>96</td>
</tr>
<tr>
<td>469d0f049b904af5560dd46c695237f4</td>
<td>97</td>
</tr>
<tr>
<td>2ceae1e12031ae595e3bac7d801d809e</td>
<td>97</td>
</tr>
<tr>
<td>54a13fd680ae7d4e281b1756d36c1d7</td>
<td>85</td>
</tr>
<tr>
<td>9f96ec93120e1531b53836af841c627</td>
<td>96</td>
</tr>
<tr>
<td>cc9d6b5d42c15dbc305bad46b2fcee3</td>
<td>97</td>
</tr>
<tr>
<td>dfc4b5f3559fbbcaa7d003fbf5577fk</td>
<td>97</td>
</tr>
<tr>
<td>4623c454b08d8eccc8e664637ab3c7771</td>
<td>85</td>
</tr>
<tr>
<td>78d9013678a334b52a93b02f24680a2d</td>
<td>74</td>
</tr>
<tr>
<td>b1d1d0f596ce8d6b464bc2c7e2e92f</td>
<td>97</td>
</tr>
<tr>
<td>l3a4b1e4e2b23d19b12b6c6a3e07878e</td>
<td>97</td>
</tr>
<tr>
<td>b1934f6f8f8f2c784495553413b4a2e4</td>
<td>97</td>
</tr>
</tbody>
</table>

Code 5: Mastiff Fuzzy Hashing results

5.2.5 Call Graph Extraction and Comparison

Only Pyew has the modules that are responsible for generating call graphs and call graph comparison. However, the tool was unable to produce call graphs for packed malware and could only do so once the malware was unpacked.

In order to measure if the call graphs produced are good enough, we analysed a number of call graphs generated by this tool and the ones generated by IDA Pro. The nodes identified by Pyew are fewer, however, it still produces a call flow that can be used to classify the malware based on the graph clustering module in the tool. This module was analysed next to measure the accuracy of its findings against the results obtained using SSdeep hashing. Using the report shown in Code 5, the graph clustering module was used to analyse the original file against 3 of the files in the extract.

<table>
<thead>
<tr>
<th>File MD5</th>
<th>Mastiff Expert</th>
<th>Pyew Alist</th>
<th>Primes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfc4b5f3559fbbcaaf7d003fbf5577f4</td>
<td>97</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>4623c454b08d8eccc8e664637ab3c7771</td>
<td>85</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>78d9013678a334b52a93b02f24680a2d</td>
<td>74</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3: Comparision of Mastiff Similarity detection vs Pyew cluster graph similarity analysis.

Analysis of the similarity detection was further performed using the file identified with MD5: 5f232bc729328b46855ddcd8d86a01 and its’ fuzzy hash matches from Mastiff. The files were uploaded in the graph clustering similarity module in pyew and Figure 10 shows the comparison of the results achieved. The results obtained from the module in Pyew gave 100% similarity across all the files for expert, Alist and Primes as shown in Table 3 and then in Figure 10 which argues against the module’s accuracy in graph cluster matching.

Mastiff gives more defined answers than pyew results although the difference can be seen in one file. While the graph clustering module in Pyew is a very good tool to be
used in conjunction with other information extracted, it is not best to use it as a reliable tool but rather combine it with other tools to build a framework with better detection accuracy.

5.2.6 String Analysis

Using Peframe, URLs were extracted and a count was used to see how many times each URL appears in the dataset and the top 10 are listed in Table 4. The URLs extracted can be used as fingerprints to detect if a file is malicious and they can also be used to group what kind of malware they are.

5.2.7 Third Party Plugins

Using the virus total plugin, the detection across the samples gives good results because the samples are older than one year.

<table>
<thead>
<tr>
<th>Top 5 urls</th>
<th>No.</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://ocsp.thawte.com">http://ocsp.thawte.com</a></td>
<td>43</td>
</tr>
<tr>
<td><a href="http://nsis.sf.net/NSIS_Error">http://nsis.sf.net/NSIS_Error</a></td>
<td>40</td>
</tr>
<tr>
<td><a href="http://crl.thawte.com/ThawteTimestampingCA.crl">http://crl.thawte.com/ThawteTimestampingCA.crl</a></td>
<td>39</td>
</tr>
<tr>
<td><a href="http://ts-aia.ws.symantec.com/tss-ca-g2.cer">http://ts-aia.ws.symantec.com/tss-ca-g2.cer</a></td>
<td>39</td>
</tr>
<tr>
<td><a href="http://ts-crl.ws.symantec.com/tss-ca-g2.crl0">http://ts-crl.ws.symantec.com/tss-ca-g2.crl0</a></td>
<td>39</td>
</tr>
</tbody>
</table>

Table 4: Top URLs extracted

However from the analysis, even given that the samples are old, Table 5 has a list of files where the Anti-Virus engines that Virus Total uses return a detection rate of 0% which shows that even some malware are not detected by a collection of anti-virus engines and therefore there is a need to fill this gap. The overall detection analysis obtained from virus total is shown in Figure 9.

<table>
<thead>
<tr>
<th>Sample MD5</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>12d69c64ff6e98debc05243213</td>
<td>e5561bf6</td>
</tr>
<tr>
<td>6c55b3c4d59420b26f198b2b</td>
<td>2ea32d25</td>
</tr>
<tr>
<td>7c7dcb713a160ad00f3a717a</td>
<td>9ae3cca</td>
</tr>
<tr>
<td>77cfb9a441eb8516943da23d</td>
<td>bd035cbaf</td>
</tr>
<tr>
<td>e97143bf1c63cafldb8e4a3ca</td>
<td>086c3834</td>
</tr>
</tbody>
</table>

Table 5: Some of the analysis results

Vigna[22] notes that relying on Anti-viruses to detect malware is not a good solution even a year after malware has been discovered. He states that some engines would not
detect the malicious files while detection of the malware on the day of discovery is limited to 51% of the engines sometimes and there are some cases where it takes up to 2 days before the anti-viruses can even detect a new malware sample. These statistics don’t favour the reliance on the protection offered by anti-virus engines and particularly for virus total. There is always a time delay of 10 minutes between upload and retrieval of the report. However, the information offered by Virus total is a great addition for malware analysts when detecting malware in systems.

6 Conclusion and future work

In this paper we evaluate 3 new static analysis tools that provide better advanced static analysis statistics today. Although most of the tools provide the same information, mastiff is more detailed than the others and Pyew introduces new modules can improve the detection and clustering rates while Peframe provides a simple but straightforward report. During the experimentation, we faced challenges that allowed us to develop scripts that allow for better automation of the analysis process and also analysis output log manipulation and management. We also noticed some limitations since some of the indicators of compromise in a PE file have not been fully explored. This is the focus of our future works. We are testing a framework that utilises the positives of these tools while extending their functionality and bridging the gap.

The goal is use the fingerprints extracted from the information obtained from static analysis to model semantic signatures that can be used to detect malicious files without having to extend resources required for dynamic analysis. Some of the fingerprints identified have been classified as PE field indicators, File and section hash values and extracted file information and Call graph sequencing. We are looking at using a combination of all these fingerprints obtained from in-depth static analysis and similarity matching and classification methods to provide high accuracy in malware detection.

7 References


Controllable Setup Queue for Energy-Aware Server

Tuan Phung-Duc*

Abstract

In order to save energy, a server is switched to a sleep mode as soon as becoming free. However, when there are some waiting jobs, the server should be switched to the normal mode in order to reducing the delay. The server needs some setup time to return to the normal mode. A quick setup is desired when the number of waiting jobs exceeds some threshold. In this paper, we propose such a model for which we obtain exact results for the joint distribution of the number of jobs in the system and the state of the server as well as the generating functions from which the mean queue length and the mean power consumption are derived. Numerical results are provided to show the energy-performance tradeoff.

1 Introduction

Vacation queue is the one in which the server may be unavailable for a random period of time upon being idle. The time that the server is away from service is called the vacation. Vacation is resulted from many factors. In some cases, vacation is resulted from post service processing or maintenance. In some other cases, vacation corresponds to sleep mode where the server is turned off in order to save energy in communication systems. Sometime vacation is caused by server breakdown where the vacation period corresponds to the repair time. A vacation is also interpreted to some secondary service. In general, server vacations are useful for systems where we wish to utilize the idle time of the server for some other purpose. Therefore, vacation models are more applicable for real world system than the ordinary one without vacation. Research of vacation queues is an active area with numerous references in the literature [11].

Two simple vacation policies are single vacation and multiple vacation. In the former, the server takes one vacation and returns to normal mode even if there is no customer in the system. In the latter case, upon the completion of a vacation, if there are not jobs present, the server takes another vacation otherwise it starts serving waiting jobs exhaustively. This paper focuses on the latter case where the server takes consecutive vacations if the system is empty upon returning from vacations. Working vacation is proposed by Servi and Finn [3] in which the server works at a different speed during vacation.

Recently, efficient usage of energy in ICT systems is paid much attention. Especially in data centers with a huge number of servers, saving energy consumption is very important. Many works on this direction have been carried

∗Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, tuan@is.titech.ac.jp

K. Djemame, R. Kavanagh, D. Armstrong (Eds.): UKPEW 2015, University of Leeds, pp. 96–109
out [7, 8, 10, 9, 6]. Phung-Duc [7, 8, 10] and Mitrani [6] consider power saving policies for data centers where jobs are stored in a common buffer if all the servers are busy. In these work, a server is turned ON and OFF according to the number of jobs present at the common buffer. Maccio and Down [5] consider a power-saving policy for a single server in which the server is switched OFF after being empty for sometime and it is turned to the normal mode when the number of jobs in the system reaches some threshold. It is also assumed by Maccio and Down [5] that the server takes sometime to switch from the OFF state to the normal mode.

In this paper, we extend the model in [5] by allowing service at the sleep mode and different setup rates according to the congestion level. In particular, the server has a power saving mode (sleep mode) and a normal mode where the service rate of the latter is larger than that of the former. When the server becomes empty, it is switched to the sleep mode. When some customer arrives the server is switched to the normal mode. However, it takes some setup time to change from the sleep mode to the normal mode. The main characteristics of our model is that the setup time can be controlled according to the number of customers in the system. The setup rate is larger when the number of customers in the system reaches some threshold. This generalizes the working vacation model of Servi and Finn [3] in which the setup rate does not depend on the number of customers in the system. For this model, we obtain explicit expressions for the joint stationary distribution of the state of the server and the number of customers in the system and its generating functions from which we can derive some performance measures such as the mean queue length and the mean power consumption in explicit form.

It should be noted that the setup time and service speed can be adjusted by power consumption [1]. In particular, fast setup time requires more power consumption. The speed of the server can also be changed using dynamic DVFS (voltage and frequency scaling) [4].

The rest of this paper is organized as follows. Section 2 presents the model and the queue length analysis. Numerical results are presented to show the energy-performance tradeoff in section 3. A model with working vacation interruption is presented in section 4. Concluding remark is presented in section 5.

2 Controllable setup time

2.1 Model

We consider an M/M/1 queueing system with working vacation. It should be noted that in the single server context, vacation and setup time are the same. Customers arrive at the system according a Poisson process with rate $\lambda$ and request for an exponentially distributed service with mean $1/\mu$. The server is switched to a sleep mode as soon as it becomes free in which the service time of the server is exponentially distributed with mean $1/\mu_0$. In the sleep mode, if there are some waiting customers, the server is switched to the normal mode. However it takes some setup time (switching time) which depends on the number of jobs in the system. Let $L(t)$ denote the number of jobs in the system at time $t \geq 0$. We assume that if $L(t) < N$ the setup time is exponentially distributed with mean $1/\gamma_0$ while the setup time is exponentially distributed with mean...
1/γ if \( L(t) \geq N \). Our model generalizes the M/M/1/WV by Servi and Finn [3] where the case \( γ_0 = γ_1 \) is considered.

2.2 Analysis

Let \( S(t) \) denote the state of the server,
\[
S(t) = \begin{cases} 
0 & \text{setup mode,} \\
1 & \text{normal mode.} 
\end{cases}
\]

It is easy to see that \( \{(S(t),L(t)); t \geq 0\} \) forms a Markov chain on state space
\[
\{(0,j); j \in \mathbb{Z}_+ \} \cup \{(1,j); j \in \mathbb{N} \},
\]
where \( \mathbb{Z}_+ = \{0, 1, 2, \ldots\} \) and \( \mathbb{N} = \{1, 2, \ldots\} \). See Figure 1 for the transitions among states. Let \( π_{i,j} \) denote the steady state probability that the system is in state \((i,j)\).

Balance equations for states \((0,j), j \in \mathbb{Z}_+ \) are given by
\[
(\lambda + \mu_0 + γ)π_{0,N} = \lambda π_{0,N-1} + \mu_0 π_{0,N+1},
\]
(1)
\[
(\lambda + \mu_0 + γ)π_{0,j} = \lambda π_{0,j-1} + \mu_0 π_{0,j+1}, \quad j \geq N + 1.
\]
(2)

Letting \( \Pi_0(z) = \sum_{j=0}^{\infty} π_{0,j}z^j \) and transforming (1) and (2) to the generating function domain yields
\[
(λ + μ_0 + γ)Π_0(z) = λπ_{0,N-1}z^N + λzΠ_0(z) + \frac{μ_0}{z}[Π_0(z) − π_{0,N}z^N],
\]
leading to
\[
((λ + μ_0 + γ)z − λz^2 − μ_0)Π_0(z) = λπ_{0,N-1}z^{N+1} − μ_0 π_{0,N}z^N.
\]
Let \( 0 < z_0 < 1 < z_1 \) denote two distinct roots of the quadratic equation in the left hand side. We have
\[
z_0 = \frac{λ + μ_0 + γ - \sqrt{(λ + μ_0 + γ)^2 - 4λμ_0}}{2λ},
\]
\[
z_1 = \frac{λ + μ_0 + γ + \sqrt{(λ + μ_0 + γ)^2 - 4λμ_0}}{2λ}.
\]
Putting $z = z_0$ into the above equation yields,

$$\pi_{0,N} = \beta_N \pi_{0,N-1},$$

where $\beta_N = \lambda z_0 / \mu_0$. After some arrangement we obtain

$$\Pi_0(z) = \frac{\pi_{0,N} z^N}{1 - \frac{z}{z_1}}.$$ 

Furthermore, we also have

$$(\lambda + \mu_0 + \gamma_0) \pi_{0,j} = \lambda \pi_{0,j-1} + \mu_0 \pi_{0,j+1}, \quad j = 1, 2, \ldots, N - 1.$$ 

Together with the relation $\pi_{0,N} = \beta_N \pi_{0,N-1}$, we obtain

$$\pi_{0,j} = \beta_j \pi_{0,j-1}, \quad j = N, N - 1, \ldots, 1,$$

where

$$\beta_j = \frac{\lambda}{\lambda + \mu_0 + \gamma_0 - \mu_0 \beta_{j+1}}, \quad j = N - 1, N - 2, \ldots, 1.$$

Similarly, now we consider the case where the server is in the regular mode. The balance equations are given as follows.

$$(\lambda + \mu) \pi_{1,N} = \lambda \pi_{1,N-1} + \mu \pi_{1,N+1} + \gamma \pi_{0,N},$$

$$(\lambda + \mu) \pi_{1,j} = \lambda \pi_{1,j-1} + \mu \pi_{1,j+1} + \gamma \pi_{0,j}, \quad j \geq N + 1.$$ 

Let $\Pi_1(z) = \sum_{j=0}^{\infty} \pi_{1,j} z^j$. Transforming the above balance equations to the generating function domain yields,

$$[(\lambda + \mu) z - \lambda z^2 - \mu] \Pi_1(z) = \lambda \pi_{1,N-1} z^{N+1} - \mu \pi_{1,N} z^N + \gamma z \Pi_0(z).$$ 

Substituting $z = 1$ into the above equation, we have $\mu \pi_{1,N} = \lambda \pi_{1,N-1} + \gamma \Pi_0(1)$. Thus,

$$(z - 1)(\mu - \lambda) \Pi_1(z) = \lambda \pi_{1,N=1} z^{N}(z - 1) + \gamma z \Pi_0(z) - \gamma \Pi_0(1) z^N.$$ 

After some algebra we obtain

$$\Pi_1(z) = \frac{\lambda \pi_{1,N-1} z^N}{\mu - \lambda z} + \frac{\gamma \pi_{0,N}}{1 - \frac{1}{z_1} (1 - \frac{z}{z_1}) (\mu - \lambda z)} z^N.$$ 

Dividing both sides by $z^N$ and put $z = 0$ in both sides yields

$$\pi_{1,N} = \alpha_N \pi_{1,N-1} + \varphi_N,$$

where

$$\alpha_N = \frac{\lambda}{\mu}, \quad \varphi_N = \frac{\gamma \pi_{0,N}}{\mu (1 - 1/z_1)}.$$ 

We also have balance equations as follows.

$$(\lambda + \mu) \pi_{1,j} = \lambda \pi_{1,j-1} + \mu \pi_{1,j+1} + \gamma_0 \pi_{0,j}, \quad j = 2, 3, \ldots, N - 1.$$
Thus, we have
\[ \pi_{1,j} = \alpha_j \pi_{1,j-1} + \varphi_j, \quad j = N, N - 1, \ldots, 2, \]
where
\[ \alpha_j = \frac{\lambda}{\lambda + \mu - \mu \alpha_{j+1}}, \quad \varphi_j = \frac{\mu \varphi_{j+1} + \gamma_0 \pi_{0,j}}{\lambda + \mu - \mu \alpha_{j+1}}, \quad j = N - 1, N - 2, \ldots, 2. \]

Furthermore, we also have
\[ (\lambda + \mu) \pi_{1,1} = \mu \pi_{1,2} + \gamma_0 \pi_{0,1}. \]

Thus, we obtain
\[ \pi_{1,1} = \frac{\mu \varphi_2 + \gamma_0 \pi_{0,1}}{\lambda + \mu - \mu \alpha_2}. \]

Therefore, now we have expressed every probabilities \( \pi_{i,j} \) in terms of a single unknown probability \( \pi_{0,0} \). This unknown probability will be determined using the normalization condition
\[ \sum_{j=0}^{N-1} \pi_{0,j} + \Pi_0(1) + \sum_{j=1}^{N-1} \pi_{1,j} + \Pi_1(1) = 1. \]

The steady state probability \( \pi_{0,N+i}, \pi_{1,N+i} (i \in \mathbb{Z}_+) \) are explicitly obtained as follows.
\[ \pi_{0,N+i} = \frac{\pi_{0,N}}{z_1^i}, \quad \pi_{1,N+i} = \frac{\varphi N A}{z_1^i} + (\alpha_N \pi_{1,N-1} + \varphi_N B) \rho^i, \quad i \in \mathbb{Z}_+, \]
where
\[ A = \frac{1}{1 - \rho z_1}, \quad B = \frac{\rho z_1}{\rho z_1 - 1}. \]

We have
\[ \Pi_0(z) = \frac{\pi_{0,N} z^N}{1 - \frac{z}{z_1}}, \quad \Pi_1(z) = \alpha_N \pi_{1,N-1} \frac{z^N}{1 - \rho z} + \varphi_N \left( \frac{A z^N}{1 - \frac{z}{z_1}} + \frac{B z^N}{1 - \rho z} \right) \]

Differentiating these generating functions yields
\[ \Pi'_0(1) = \pi_{0,N} \frac{1}{z_1} + N(1 - \frac{1}{z_1}) \frac{1}{(1 - \frac{1}{z_1})^2}, \]
\[ \Pi'_1(1) = \varphi_N A \frac{1}{z_1} + N(1 - \frac{1}{z_1}) \frac{1}{(1 - \frac{1}{z_1})^2} + (\alpha_N \pi_{1,N-1} + \varphi_N B) \frac{1}{(1 - \rho)^2}. \]

### 2.3 Performance measures

Because the power consumption is different among states, we classify the state spaces as in Table 1. Thus, we have the following expressions.
Table 1: Classification of states.

<table>
<thead>
<tr>
<th>Notation</th>
<th>States</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>(0,0)</td>
<td>$P_{idle}$</td>
</tr>
<tr>
<td>LOW SETUP</td>
<td>{(0,j); j = 0,1,\ldots,N-1}</td>
<td>$P_{L\text{setup}}$</td>
</tr>
<tr>
<td>HIGH SETUP</td>
<td>{(0,j); j \geq N}</td>
<td>$P_{H\text{setup}}$</td>
</tr>
<tr>
<td>NORMAL MODE</td>
<td>{(1,j); j \geq 1}</td>
<td>$P_{H\text{setup}}$</td>
</tr>
</tbody>
</table>

- Probability that the server is idle:
  $$P_{idle} = \pi_{0,0}.$$  

- Probability that the server is at low setup state:
  $$P_{L\text{setup}} = \sum_{j=1}^{N-1} \pi_{0,j}.$$  

- Probability that the server is at high setup state:
  $$P_{H\text{setup}} = \Pi_{0}(1).$$  

- Probability that the server is at normal state:
  $$P_{\text{normal}} = \sum_{j=1}^{N-1} \pi_{1,j} + \Pi_{1}(1).$$  

- The mean number of customers in the system is given by
  $$E[L] = \sum_{j=1}^{N-1} j\pi_{0,j} + \sum_{j=1}^{N-1} j\pi_{1,j} + \Pi'_{0}(1) + \Pi'_{1}(1).$$  

- Next, we consider the power consumption of the system. We assume that the power consumptions at the sleep mode are $C_0$ and $C_1$ at low and high setup rates, respectively. Furthermore, the power consumption at normal mode is $C_2$. Thus, the mean power consumption is given by
  $$E[P] = C_0 \left( \sum_{j=1}^{N-1} \pi_{0,j} \right) + C_1 \Pi_{0}(1) + C_2 \left( \sum_{j=1}^{N-1} \pi_{1,j} + \Pi_{1}(1) \right).$$  

- We also consider a cost function taking into account both the queue length and the power consumption [5] as follows:
  $$f_{\text{cost}} = E[L] \times E[P].$$
Remark 1. We consider a special case where $\gamma \to \infty$ meaning that the setup time is instantaneous. In this case it is easy to see that $\lim_{\gamma \to \infty} z_0 = 0$ and $\lim_{\gamma \to \infty} z_1 = \infty$ leading to $\pi_{0,N} = 0$ and then $\Pi_0(z) = 0$. We also have

$$\Pi_1(z) = \frac{\lambda \pi_{1,N-1} z^N}{\mu - \lambda z}.$$ 

Thus our system reduces to a queueing system with vacation interruption [2].

Remark 2. For a special case where $\gamma_0 = 0$, our system reduces to the M/M/1 queues with N-policy and working vacations [13].

![Figure 2: Probability vs. threshold.](image)

3 Numerical results

We fix some parameters as follows $\gamma = 10, \gamma_0 = 0.1, \mu = 1, \mu_0 = 0.01$ and change the value of $\lambda = 0.9, 0.7, 0.5, 0.3$. Figures 2–5 represent the probabilities of server states (idle, low setup, high setup, normal) against the threshold. It should be noted that the power consumption of the server on these states are different.

Under the current setting, the in the states $(0, j) \ (j \in \mathbb{Z}_+)$, the service rate is very low leading to the fact that almost all customers are served in normal mode. As a result, $P_{normal}$ is almost $\lambda/\mu$ according to the Little’s law. On the other hand, $P_{Lsetup}$ and $P_{idle}$ increase with the threshold as is expected. Furthermore, $P_{Hsetup}$ decreases with the threshold.

Figure 7 shows the mean number of customers against the threshold. We observe that the mean number of customers in the system increases with the threshold as is expected. So, we may think that it is better to setting the
threshold by a small number. However, if the power consumption for high setup states is large, setting a small threshold may lead to a large power consumption.

Thus, we need to have a closer look at the power consumption. To this end, we investigate a scenario where $C_2 = 1, C_0 = 0.01, C_1 = 100$ and $\lambda = 0.9, 0.7, 0.5, 0.3$. In particular, we assume that the server at normal mode consumes $C_2$ unit energy at a unit time while that at slow and high setup modes are $C_0 = 0.01$ and $C_1 = 100$, respectively.

Figure 6 shows the mean power consumption against the threshold. We observe that the mean power consumption decreases with the threshold. Thus, there exists some trade-off. Figure 8 represents the cost function against the threshold. We observe that there exists some threshold at which the cost function is minimized.

4 Model with vacation interruption

4.1 Model

We consider in this section an M/M/1 queueing system with working vacation and vacation interruption. Customers arrive at the system according a Poisson process with rate $\lambda$ and request for an exponentially distributed service with mean $1/\mu$. The server is switched to a power saving mode as soon as it becomes free in which the service time of the server is exponentially distributed with mean $1/\mu_0$. In the power saving mode, if there is some waiting customers, the serve is switched to regular mode. However the server needs some setup time to change from the working vacation mode to the normal mode. We assume that the setup time is exponentially distributed with mean $1/\gamma_0$. Furthermore, we assume that if a customer arrives and the number of customers in the system reaches some threshold ($N$), the server interrupts the working vacation and switches to the
Figure 4: Probability vs. threshold.

Figure 5: Probability vs. threshold.
Figure 6: Mean power consumption vs. threshold.

Figure 7: Queue length vs. threshold.
normal mode immediately. This model generalizes the non-working vacation mechanism in [2]. It should be noted that this model is not the one obtained by letting $\gamma \rightarrow \infty$ in the previous model.

4.2 Analysis

Let $S(t)$ denote the state of the server,

$$S(t) = \begin{cases} 
0 & \text{serving,} \\
1 & \text{setting up,} 
\end{cases}$$

Let $L(t)$ denote the number of customers in the system. It is easy to see that $(S(t), L(t))$ forms a Markov chain on state space

$\{(0, j); j = 0, 1, \ldots, N - 1\} \cup \{(1, j); j \in \mathbb{N}\}$.

See Figure 1 for the transitions among states. Let $\pi_{i,j}$ denote the steady state probability that the system is in state $(i, j)$. 

---

Figure 8: Cost function vs. threshold.

Figure 9: Model with working vacation interruption.
Balance equations for states \((0, j), j = 0, 1, \ldots, N - 1\) are given as follows
\[
(\lambda + \mu_0 + \gamma)\pi_{0, 1} = \lambda\pi_{0, 0} + \mu_0\pi_{0, 2},
\]
\[
(\lambda + \mu_0 + \gamma)\pi_{0, j} = \lambda\pi_{1, j-1} + \mu_0\pi_{0, j+1}, 1 \leq j \leq N - 2,
\]
\[
(\lambda + \mu_0 + \gamma)\pi_{0, N-1} = \lambda\pi_{0, N-2}.
\]
We have
\[
\pi_{0, N-1} = \alpha_{N-1}\pi_{0, N-2},
\]
where \(\alpha_{N-1} = \lambda/(\lambda + \mu_0 + \gamma)\). Furthermore, from the above balance equations, it is easy to see that
\[
\pi_{0, j} = \alpha_j\pi_{0, j-1}
\]
where
\[
\alpha_j = \frac{\lambda}{\lambda + \mu_0 + \gamma - \mu_0\alpha_{j+1}}, \quad j = 1, 2, \ldots, N - 1.
\]
Thus, once \(\pi_{0, 0}\) is known, all other \(\pi_{0, j}\) are known too.

Next, we consider \(\pi_{1, j} (j \in \mathbb{N})\). First, taking the cut before state \((1, N)\) yields
\[
\mu\pi_{1, N} = \lambda\pi_{1, N-1} + \lambda\pi_{0, N-1},
\]
or
\[
\pi_{1, N} = \beta_N\pi_{1, N-1} + \varphi_N,
\]
where
\[
\beta_N = \frac{\lambda}{\mu}, \quad \varphi_N = \frac{\lambda\pi_{0, N-1}}{\mu}.
\]
Second, we have
\[
(\lambda + \mu)\pi_{1, j} = \lambda\pi_{1, j-1} + \mu\pi_{1, j+1} + \gamma\pi_{0, j}, \quad 2 \leq j \leq N - 1,
\]
which yields,
\[
\pi_{1, j} = \beta_j\pi_{1, j-1} + \varphi_j,
\]
where
\[
\beta_j = \frac{\lambda}{\lambda + \mu - \mu\beta_{j+1}}, \quad \varphi_j = \frac{\mu\varphi_{j+1} + \gamma\pi_{0, j}}{\lambda + \mu - \mu\beta_{j+1}}.
\]
We also have
\[
(\lambda + \mu)\pi_{1, 1} = \gamma\pi_{0, 1} + \mu\pi_{1, 2}
\]
which together with \(\pi_{1, 2} = \beta_2\pi_{1, 1} + \varphi_2\) yield
\[
\pi_{1, 1} = \frac{\gamma\pi_{0, 1} + \mu\varphi_2}{\lambda + \mu - \beta_2\mu}.
\]
Letting \(\Pi_1(z) = \sum_{j=N}^{\infty} \pi_{0, j}z^j\), using the same arguments as in Section 2, we obtain
\[
\Pi_1(z) = \frac{\pi_{1, N}z^N}{1 - \rho z},
\]
where \(\rho = \lambda/\mu\). Now, everything are expressed in terms of \(\pi_{0, 0}\) which will be determined by the normalization condition
\[
\sum_{j=0}^{N-1} \pi_{0, j} + \sum_{j=1}^{N-1} \pi_{1, j} + \Pi_1(1) = 1.
\]
4.3 Variant models
The methodology of this paper can be applied for various models with state-dependent structure. In particular, we may consider a general model with similar transition structure where the transition rates are state-dependent when the number of customers in the system is smaller than the threshold. This includes the multiserver model with synchronous vacation [14] or the model with $(e, d)$ setup time [12]. A straightforward extension may be the system where the setup rate depends on multiple thresholds. It is also possible to analyze extensions of our models where the server stays idle waiting for customers for some time before being switched to the sleep mode.

5 Concluding remarks
In this paper, we have presented the generating function methodology for a model with controllable setup time. We have obtained explicit expressions for the partial generating functions for the number of customers in the system. These generating functions have been used to show the energy-performance tradeoff. Our results include the ones in the literature as special cases. We have considered some numerical examples where the power-consumption for each state is given. However, since these parameters depend on the system, from a practical point of view, a careful validation is needed.

References
Acknowledgements

The author was supported in part by JSPS KAKENHI Grant Number 2673001. The author would like to thank the referee for constructive comments which improve the presentation of the paper.
Abstract

The IEEE 802.11 family of protocols for wireless local area networks have been used for many years. In this paper we present a PEPA model for 802.11b operating under a number of scenarios which highlight different performance issues. In particular we study the issue of fairness when there is competition for access to the transmission channel.

Keywords. Performance analysis, IEEE 802.11b, WLAN, fairness.

1 Introduction

In the past decade, the 802.11 family of protocols have been the standard for wireless local area networks [1, 11]. IEEE 802.11 is categorised as a set of protocols, 802.11 a/b/g/n/ac, with very similar structure, but different operating ranges (power, data rate, message length etc). Determining the optimum characteristics for transmission in a specific usage scenario prior to deployment, is clearly a problem of considerable practical relevance. There are many simulations techniques and packages which can be used to build or analyse models of WLAN and mobile environments. While simulations can support a detailed representation of protocol actions, the approach may suffer from excessively long run times, making parameter optimisation infeasible in general. A typical solution to this problem is to employ some form of stochastic modelling technique to create an abstract representation of the system which can solved analytically or numerically to derive measures of interest, which can then be verified using simulation as necessary. Both simulation and mathematical modelling can suffer from problems of lack of behavioural insight and lack of modelling reusability (as a new bespoke model potentially needs to be created for every new scenario). Formal modelling techniques, such as stochastic Petri nets, stochastic automata and stochastic process algebra, seek to overcome these issues by providing a high level modelling paradigm which can used to reason about the model behaviour and to derive numerical solutions to predict performance.

Performance modelling has been employed successfully to evaluate the performance of (current and future) networking systems for many decades (see [16] for a general overview). There have been many attempts to model different aspects of IEEE 802.11 using a wide variety of methods. The majority of these studies have used simulation, which can give a good indication of predicted performance, but provide limited insight on the behaviour which leads to this performance. Formal modelling techniques, such as stochastic process algebra, allow the modeller to reason about properties of a model via explicit naming of components and actions. However, constructing large process algebra models with complicated internal component behaviour is a difficult task. Despite this there are a small number of published studies which have used process algebra to model aspects of IEEE 802.11, see [2, 14, 18].

The aim of this paper is to investigate the fairness of IEEE 802.11b using the stochastic process algebra PEPA [12]. The approach is based on that of Kloul and Valois [14], extended
to consider additional analysis that is available now in the latest tools for PEPA and in considering additional scenarios. Initially we develop a model where there are a number of pairs of nodes, with communication within each pair. The pairs may be within signal range of each other, so must compete for access to the medium. This scenario is used to better understand the behaviour of the protocol and to establish a baseline for further analysis. The derived performance metrics of interest include the utilisation of the medium and the throughput of each pair of nodes.

This paper is organized as follows. In Section 2 we discuss a background and related work to summarise some knowledge about IEEE 802.11 and PEPA. In Section 3, we give the model that we used in PEPA. Section 4 is for parameters and experimental set up. The results and figures with explanations are given in Section 5. Then, Section 6 is for conclusion and future works.

2 Background and Related work

2.1 IEEE 802.11

IEEE 802.11 categorised to a/b/g/n/ac and etc. Each of them has a specific performance, transmission speed and signal range, and they are a realistic standard in WLAN. Many researcher has been worked on IEEE 802.11 in term of rate adaptation scheme and performance of IEEE 802.11 MAC layer, and performance metric systems. Zhai et al. [21] attempted to characterize the probability distribution model of the MAC layer packet service time. They have argued, it has been based on deriving and creating function of the probability mass function of the inter-departure interval, but, it seems, they have worked on delay mean, throughput at several traffic loads and evaluate a performance metric systems only. In [7] explained that to access the medium for any devices the capability and fairness are most important for reaching great effectiveness in numerous wireless devices and traffics. However, other research studied on backoff algorithm on ad hoc and multi-hop ad hoc, they have been described performance and fairness concerns as a part of 802.11 MAC protocol. Razafindralambo and Valois [17] explored a symmetric hidden terminal scenario to analyse three pairs scenario for four backoff algorithms, and for understanding the performance of backoff algorithms in multi-hop ad hoc, they have evaluated the performance of each backoff algorithms from efficiency point of view and when possible from a fairness by using PEPA. However, it seems they did not consider the retry limit, reducing and increasing process for fairness performance metrics. Short-term and long-term fairness have investigated performance evaluation of WLAN protocol of fairness for accessing channel [14] for the communicating pairs in term of medium utilization and throughput. We have studied on two pairs and three pairs scenario based on [14], to demonstrate performance modelling of WLAN protocols with IEEE 802.11 to better understanding the protocol behaviour.

Many simulations techniques used to analyse WLAN models. While simulations can support a detailed representation of protocol actions, the approach may suffer from excessively long run times, making parameter optimisation infeasible in general. A typical solution to this is to employ some form of stochastic modelling technique (see for example [8, 15] to create an abstract representation of the system. PEPA [12] is a compositional algebraic modelling formalism used to efficiently specify and analyse models of computer systems, multimedia applications and communication systems. According to Hillston [12], PEPA was "developed to investigate how the compositional features of process algebra might impact upon the practice of performance modelling. From a PEPA specification it is possible to analyse a module using a Continuous Time Markov Chain (CTMC), system of ordinary differential equations (ODEs) or by stochastic simulation.

Argent-Katwala et al. [2] studied on wireless network protocols and performance models of the 802.11 in term of its QoS based on PEPA. They argued that most of the technologies have been developed to enhance the reliability of computer networks. In wireless communications protocols security is mandated needs in exchanging data, which must be delivered within a specific time. Moreover, they used PEPA to find properties which it can not be easy to find manually in term of computing quantitative, passage time and increase higher probability for performance demands in wireless communication. Sridhar and Ciobanu [18] by using PA and PEPA focused on DCP within IEEE 802.11, which it uses (CSMA/CA) and backoff mechanism. They described handoff mechanism, quantitative analysis and channel mobility. Likewise, they assessed handoff mechanism and improved π-calculus with numerator in WLAN, which is data can be pass in always by allowed mediums to another one. Kloul
and Valois [14] studied on performance analysis of the 802.11b protocol by using performance modelling. Particularly, they investigated unfairness scenario in MANET, as they studied on three different scenarios. Three pairs scenario is one of these scenarios. However, this scenario has been studied for the first time in [4]. In addition, they have interested on system behaviour to measure and investigate the performance of 802.11b protocol. Similarly, we have studied on three pairs scenarios, then we shows that there are uncertainty of fairness for accessing the channel for two pairs and unfairness for three pairs scenario in term of medium utilization and throughput.

Finally, Duda [6] has argued that the issue of unfairness, as highlighted in the above research, is a consequence of the manner in which 802.11b was implemented on early switches and that modifications made to later switches alleviate this problem. However, this does not seem to have been shown empirically by any researchers. By implementing a model which shows unfairness as reported by Kloul and Valois [14] and then modifying this as described by Duda [6], it will be possible to further investigate this claim.

3 The model

3.1 Basic access mechanism

Initially, in WLAN the node listens the channel, to make sure that the channel is free to send. If the channel is busy, the node waits for a random time (backoff) in the range $[0, CW]$, where, $CW$ is the contention window. $CW_{\text{min}}$ is [31], and it doubles up to the $CW_{\text{max}}$ [1023], under 802.11b (see [6, 10]). $CW$ can return to the initial value after each successful transmission. If two nodes transmit simultaneously, then a collision occurs. When collisions are detected the transmitting nodes will backoff before attempting to retransmit. Furthermore, in 802.11b after each frame transmission, in case of collision, the Inter-Frame Space ($IFS$) is applied. The minimum fixed and shortest interval of time is called Short-IFS ($10\mu s$). Distributed-IFS ($50\mu s$) when the channel is idle, and the packet is communicated if the backoff is equal to zero. After the node notices a transmission and if the channel is free to use, the backoff decrements. Finally, the decrementation starts again through a $DIFS$, while the channel stays as an idle. Nevertheless, before sending next frame, instead of $DIFS$ an Extended-IFS ($364\mu s$) will apply, whenever, the node notices a signal for the duration of the backoff. Also, when the transmission positively completed, the receiver sends an $ACK$ after $SIFS$. Figure 1 shows basic access mechanism, and Table 1 typical values of the IEEE 802.11b protocol.

![Figure 1: RTC-CTS and Data-ACK scheme.](image)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Typical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CW_{\text{min}}$ and $CW_{\text{max}}$</td>
<td>31, and 1023</td>
</tr>
<tr>
<td>Slot time</td>
<td>20$\mu$s</td>
</tr>
<tr>
<td>$SIFS$</td>
<td>10$\mu$s</td>
</tr>
<tr>
<td>$DIFS$</td>
<td>50$\mu$s</td>
</tr>
<tr>
<td>$EIFS$</td>
<td>364$\mu$s</td>
</tr>
</tbody>
</table>

Table 1: Default attribute values of IEEE 802.11b protocol.
3.2 Scenarios with a PEPA

3.2.1 The two pairs scenario (scenario 1)

In our research we started with a two pairs scenario see Figure 2, then a three pairs scenario see Figure 3. Essentially, in scenario 1 (A and B) has been used as two symmetric and independent node pairs. Once one node attempts to transmit, the other one in the pair waits to receive.

The model in the scenario 1 is interacting between three components (Pair A, Pair B, and Medium F). Pair A and Pair B are equally occupying the medium (Medium F). During sending the packet Pair A draw backoff to Pair A0, then Pair A0 starts to count the DIFS to Pair A1 or stay in the queue with Pair A5, and Pair A5 it waits with Pair A4. Similarly, all SIFS and EIFS will count till the backoff end, then the packet can transmit in Pair A2 then successfully will ACK in Pair A6.

Sequential Component Process of Pair A and Pair B component.

\[
\begin{align*}
Pair_A & \overset{\text{draw backoff, r}}{\equiv} (\text{draw backoff, r}).Pair_A \\
Pair_A0 & \overset{\text{count_difs, µ_difs}}{\equiv} (\text{count_difs, µ_difs}).Pair_A0 + (\text{queue, } \top).Pair_A5 \\
Pair_A1 & \overset{\text{count_backoff, µ_backoff}}{\equiv} (\text{count_backoff, µ_backoff}).Pair_A1 + (\text{end_backoff, µ_backoff}).Pair_A2 + (\text{queue, } \top).Pair_A5 \\
Pair_A2 & \overset{\text{transmit, µ_data}}{\equiv} (\text{transmit, µ_data}).Pair_A3 + (\text{queue, } \top).Pair_A5 \\
Pair_A3 & \overset{\text{count_sifs, µ_sifs}}{\equiv} (\text{count_sifs, µ_sifs}).Pair_A6 \\
Pair_A4 & \overset{\text{count_difs, µ_difs}}{\equiv} (\text{count_difs, µ_difs}).Pair_A1 + (\text{count_sifs, µ_sifs}).Pair_A1 + (\text{queue, } \top).Pair_A5 \\
Pair_A5 & \overset{\text{wait, µ_data}}{\equiv} (\text{wait, µ_data}).Pair_A4 \\
Pair_A6 & \overset{\text{ack, µ_backoff}}{\equiv} (\text{ack, µ_backoff}).Pair_A \\
Pair_B & \overset{\text{draw backoff, r}}{\equiv} (\text{draw backoff, r}).Pair_B \\
Pair_B0 & \overset{\text{count_difs, µ_difs}}{\equiv} (\text{count_difs, µ_difs}).Pair_B0 + (\text{queue, } \top).Pair_B5 \\
Pair_B1 & \overset{\text{count_backoff, µ_backoff}}{\equiv} (\text{count_backoff, µ_backoff}).Pair_B1 + (\text{end_backoff, µ_backoff}).Pair_B2 + (\text{queue, } \top).Pair_B5 \\
Pair_B2 & \overset{\text{transmit, µ_data}}{\equiv} (\text{transmit, µ_data}).Pair_B3 + (\text{queue, } \top).Pair_B5 \\
Pair_B3 & \overset{\text{count_sifs, µ_sifs}}{\equiv} (\text{count_sifs, µ_sifs}).Pair_A6 \\
Pair_B4 & \overset{\text{count_difs, µ_difs}}{\equiv} (\text{count_difs, µ_difs}).Pair_B1 + (\text{count_sifs, µ_sifs}).Pair_B1 + (\text{queue, } \top).Pair_B5 \\
Pair_B5 & \overset{\text{wait, µ_data}}{\equiv} (\text{wait, µ_data}).Pair_B4 \\
Pair_B6 & \overset{\text{ack, µ_backoff}}{\equiv} (\text{ack, µ_backoff}).Pair_B
\end{align*}
\]

The complete system: The shared medium is cooperated with by both pairs equally. If the first one starts to use the medium then the second one stops trying to transmit and vice versa. All components are interacting with this cooperation sets:

\[
\text{scenario1} \overset{\text{def}}{=} ((\text{Pair}_A \bowtie Medium_F) \bowtie Pair_B)
\]
where \( K = \{ \text{transmit, ack, queue, count diffs, count backoff} \} \).
\( L = \{ \text{transmit, ackB, queueB, count diffsB, count backoffB, count eifsB} \} \).

**Component of Medium F:**

\[
\begin{align*}
\text{Medium}_F & \triangleq (\text{transmit, } \top).\text{Medium}_{F+}\triangleq (\text{transmitB, } \top).\text{Medium}_{F1} \\
& + (\text{count diffs, } \top).\text{Medium}_{F+}(\text{count backoff, } \top).\text{Medium}_{F2} \\
& + (\text{count backoff, } \top).\text{Medium}_{F+}(\text{count eifs, } \top).\text{Medium}_{F3} \\
& + (\text{count diffsB, } \top).\text{Medium}_{F+}(\text{count backoffB, } \top).\text{Medium}_{F4} \\
& + (\text{end backoffB, } \top).\text{Medium}_{F+}(\text{count eifsB, } \top).\text{Medium}_{F5} \\
\text{Medium}_{F1} & \triangleq (\text{ackB, } \top).\text{Medium}_{F+}(\text{queue, } \lambda_{\text{loc}}).\text{Medium}_{F1} \\
\text{Medium}_{F2} & \triangleq (\text{transmit, } \top).\text{Medium}_{F+}(\text{ack, } \top).\text{Medium}_{F2} \\
& + (\text{queueB, } \lambda_{\text{loc}}).\text{Medium}_{F+}(\text{count diffs, } \top).\text{Medium}_{F2} \\
& + (\text{count backoff, } \top).\text{Medium}_{F+}(\text{count backoffB, } \top).\text{Medium}_{F2} \\
& + (\text{count eifs, } \top).\text{Medium}_{F2} \\
\text{Medium}_{F3} & \triangleq (\text{ack, } \top).\text{Medium}_{F+}(\text{queueB, } \lambda_{\text{loc}}).\text{Medium}_{F3} \\
& + (\text{count diffs, } \top).\text{Medium}_{F+}(\text{count backoff, } \top).\text{Medium}_{F3} \\
& + (\text{end backoffB, } \top).\text{Medium}_{F+}(\text{count eifsB, } \top).\text{Medium}_{F3} \\
\end{align*}
\]

### 3.2.2 The three pairs scenario (scenario 2)

This scenario has four components (the node pairs \( A, B \) and \( C \)), and the central one has been unfairly disadvantaged as it is been out competed by externals.

The external pairs cannot hear each other as they lie outside each others transmission range, but the central pair can hear both of them. If one of the external pairs is transmitting then the central one cannot; it is queuing till the channel is free to use. However, the second external pair cannot see the busy channel and so both the external pairs may transmit simultaneously. Thus, in this scenario, the central pair has less chance to access the medium and hence this scenario is called the unfairness scenario. Moreover, the central one has been unfairly disadvantaged as it has been out competed by externals.

**PEPA model:** In this model all pairs are collaborated with a medium. Either the medium is occupied by any external or central pairs.

**Sequential Component Process Pair A/C Component (Externals)**

\[
\begin{align*}
\text{Pair}_A & \triangleq (\text{draw backoff}, r).\text{Pair}_A0 \\
\text{Pair}_A0 & \triangleq (\text{count diffs, } \mu_{\text{diffs}}).\text{Pair}_A1+\text{(queue, } \top).\text{Pair}_A5 \\
\text{Pair}_A1 & \triangleq (\text{count backoff, } \mu_{\text{backoff}}).\text{Pair}_A1+(\text{end backoff, } \mu_{\text{end_backoff}}).\text{Pair}_A2 \\
& + \text{(queue, } \top).\text{Pair}_A5 \\
\text{Pair}_A2 & \triangleq (\text{transmit, } \mu_{\text{data}}).\text{Pair}_A3+(\text{end, } \top).\text{Pair}_A5 \\
\text{Pair}_A3 & \triangleq (\text{count eifs, } \mu_{\text{eifs}}).\text{Pair}_A6 \\
\text{Pair}_A4 & \triangleq (\text{count diffs, } \mu_{\text{diffs}}).\text{Pair}_A4+(\text{count eifs, } \mu_{\text{eifs}}).\text{Pair}_A1 \\
& + \text{(queue, } \top).\text{Pair}_A5 \\
\text{Pair}_A5 & \triangleq (\text{wait, } \mu_{\text{data}}).\text{Pair}_A4 \\
\text{Pair}_A6 & \triangleq (\text{ack, } \mu_{\text{ack}}).\text{Pair}_A \\
\text{Pair}_B & \triangleq (\text{draw backoff}, r).\text{Pair}_B0 \\
\text{Pair}_B0 & \triangleq (\text{count diffs, } \mu_{\text{diffs}}).\text{Pair}_B1+(\text{queueB, } \top).\text{Pair}_B5 \\
\text{Pair}_B1 & \triangleq (\text{count backoff, } \mu_{\text{backoff}}).\text{Pair}_B1+(\text{end backoff, } \mu_{\text{end_backoff}}).\text{Pair}_B2 \\
& + \text{(queueB, } \top).\text{Pair}_B5 \\
\text{Pair}_B2 & \triangleq (\text{transmitB, } \mu_{\text{data}}).\text{Pair}_B3+(\text{queueB, } \top).\text{Pair}_B5 \\
\text{Pair}_B3 & \triangleq (\text{count eifs, } \mu_{\text{eifs}}).\text{Pair}_A6 \\
\text{Pair}_B4 & \triangleq (\text{count diffs, } \mu_{\text{diffs}}).\text{Pair}_B4+(\text{count eifs, } \mu_{\text{eifs}}).\text{Pair}_B1 \\
& + \text{(queueB, } \top).\text{Pair}_B5 \\
\text{Pair}_B5 & \triangleq (\text{wait, } \mu_{\text{data}}).\text{Pair}_B4 \\
\text{Pair}_B6 & \triangleq (\text{ackB, } \mu_{\text{ack}}).\text{Pair}_B \\
\end{align*}
\]

**The complete system:** This section shows how this model can communication. Component \( A \) can communicate with \( B \) through the medium \( F \), but both external pairs cannot interact.
with each other, this symbol $\parallel$ shows that, here is the cooperation sets that are defined as:

\[
\text{scenario}_2 \overset{\text{def}}{=} ((\text{Pair}_A \parallel \text{Pair}_C) \otimes M_{\text{Medium},F} \otimes (\text{Pair}_B)
\]

where $K = \{\text{transmit, ack, queue, count_{dif}s, count_{backoff}, end_{backoff}, count_{eifs}}\}$.

\[L = \{\text{transmit, ackB, queueB, count_{dif}sB, count_{backoffB}, end_{backoffB}, count_{eifsB}}\}.
\]

**Component of Medium F:**

\[M_{\text{Medium},F} \overset{\text{def}}{=} (\text{transmit, } T).M_{\text{Medium},F} + (\text{transmitB, } T).M_{\text{Medium},F}
\]

\[+ (\text{count_{dif}s, } \top).M_{\text{Medium},F} + (\text{count_{backoff}, } T).M_{\text{Medium},F}
\]

\[+ (\text{count_{eifs}, } T).M_{\text{Medium},F} + (\text{end_{backoff}, } T).M_{\text{Medium},F}
\]

\[+ (\text{end_{eifs}, } T).M_{\text{Medium},F} + (\text{count_{eifsB}, } T).M_{\text{Medium},F}
\]

\[M_{\text{Medium},F1} \overset{\text{def}}{=} (\text{ackB, } \top).M_{\text{Medium},F} + (\text{queue, } \lambda_{\text{oc}}).M_{\text{Medium},F1}
\]

\[M_{\text{Medium},F2} \overset{\text{def}}{=} (\text{ack, } \top).M_{\text{Medium},F3} + (\text{ack}, T).M_{\text{Medium},F}
\]

\[+ (\text{queueB, } \lambda_{\text{oc}}).M_{\text{Medium},F2} + (\text{count_{dif}s, } T).M_{\text{Medium},F2}
\]

\[+ (\text{count_{backoff}, } T).M_{\text{Medium},F2} + (\text{end_{backoff}, } T).M_{\text{Medium},F2}
\]

\[+ (\text{count_{eifs}, } T).M_{\text{Medium},F2}
\]

\[M_{\text{Medium},F3} \overset{\text{def}}{=} (\text{ack, } T).M_{\text{Medium},F3} + (\text{ack}, T).M_{\text{Medium},F3}
\]

\[+ (\text{count_{dif}s, } T).M_{\text{Medium},F3} + (\text{count_{backoff}, } T).M_{\text{Medium},F3}
\]

\[+ (\text{count_{eifs}, } T).M_{\text{Medium},F3}
\]

4 Parameters

We have used 0.5 as a probability values of $p$ and $q$ as an assumption ($q = 1 - p$). The values of $\lambda_{\text{oc}}$ and $r$ are 100000 and 200000 respectively; these are the same values that have been used in [14]. According to the IEEE 802.11b definition and PHY standards, the data rate per stream are (1, 2, 5.5, and 11) Mbit/s [20], equal to (125000, 250000, 687500 and 1375000) Bytes/s respectively. We have applied these rates with packet payload size (700, 900, 1000, 1200, 1400 and 1500) Bytes. And, the packets per time unit for arrival and departure rate are ($\lambda$ and $r$) respectively. In this model $\mu_{\text{ack}}$ is the rate of $\text{ACK}$ of packets, $\mu_{\text{ack}} = \text{Channel throughput} ÷ (\text{Ack length} = 1\text{Byte})$. Also, $\mu_{\text{data}}$ is a rate of waiting action for packages, as it is calculated by channel throughput ÷ Packet payload, after multiplying with $10^{-6}$ it changes to Bytes per second. WLAN is used the CSMA/CA for collision avoidance, it is crucially used for performance improvement and precisely for sharing the medium equally. It is using three main techniques ($IFS$, $CW$ and $ACK$).

4.1 Inter-Frame Space (IFS)

802.11 is a fundamentally giant system of timers. Afterward each frame transmission on the medium, if nosiness are occur, the require Inter-Frame Space (IFS) is used in 802.11 protocol. Possibly, when transmitting of any particular frame ends then another one starts the IFS can apply as smallest number when the channel have to stay clear. It is an idle and essential period of time. The main crucial of an IFS is to supply waiting time during each frame transmission in a particular node, then, it allows the transmitted signal to another node. 802.11 protocol has deal with $SIFS$, $DIFS$, $EIFS$ and Slot time, see [5, 6].

4.1.1 Short Inter-Frame Space (SIFS)

SIFS is the minimum Inter-Frame time and highest priority transmissions used with DCF. It is a fixed and shortest value, and it is measured by micro seconds, $SIFS$ is an important in 802.11 to better process a received frame. It is equal to 10µs in 802.11b/g/n.

4.1.2 DCF Inter-Frame Space (DIFS)

DIFS is a medium priority waiting time after SIFS to monitor the medium. If the channel is idle again, the node is practicing DIFS. Usually the DIFS is longer than SIFS, after the node defer the idle of the channel for a specific of time (DIFS) then it waits for another period of time (backoff).

\[DIFS = SIFS + (2 \times (\text{Slot time} = 20\mu s \text {in 802.11b/g/n}))\]
4.1.3 Extended Inter-Frame Space (EIFS)
When the node can detect the signal but the DIFS is not functioning for sending next frame during collision, the transmission node is using EIFS instead of DIFS. It is the longest of the IFS, but, it is lowest priority after DIFS. EIFS (in DCF) can derive by:
\[
\text{EIFS} = \text{SIFS} + \text{DIFS} + \text{transmission time of Ack frame at lowest basic rate.}
\]

4.2 Contention Window (CW)
After a node has experimental an idle channel with appropriate IFS, the node is waiting because of any collisions. Before sending any frame the node waits randomly. In CSMA/CA it is called backoff, it is selected by node from a Contention Window (CW). Faster backoff needs less time to spend, then transmission will be faster too. Backoff is chosen over \([0, \text{CW}]\), and \(\text{CW}=\text{CW}_{\text{min}}\) for all station or nodes if a node successfully transmits a packet and then receives an ACK. But in the case of not transmitted successfully, the node is dealing another (backoff), then the CW size is increased exponentially till it is obtained to the \(\text{CW}_{\text{max}}\). Finally, the CW is reset to \(\text{CW}_{\text{min}}\) when the packet is received properly. CW and backoff can shows as follows:
\[
\text{CW}_{\text{min}} = 31, \text{CW}_{\text{max}} = 1023. \text{ And } \text{CW}_{\text{min}} \text{ augmented by } 2^n - 1 \text{ on each retry}
\]

\[
\text{Backoff Time} = (\text{Random}()) \mod (\text{CW}+1) \times \text{Slot Time}.
\]

If \(\text{BackoffTimer} = b\), when \(b\) is a random integer, also \(\text{CW}_{\text{min}} < b < \text{CW}_{\text{max}}\)

We have used the mean of CW to calculate \(\mu_{\text{bck}}\) by 
\[
\mu_{\text{bck}} = 10^{-6} \div \text{Mean of CW} \times \text{Time Slot}
\]

4.3 Data Rates
ACK send by receiver when it gets the packet successfully, it is precautions action when collisions occur. ACK in 802.11b protocol is deal with data rate (1, 2, 5.5, and 11) Mbit/s, each \(\mu_{\text{ack}}\) is equal to (1644.74, 3289.5, 9046.125 and 18092.25) Bytes/s respectively. Then, \(\mu_{\text{data}}\) can obtain for each of them by \(\mu_{\text{ack}} \times \text{packet payload size}\).

5 Results and Figures
5.1 Performance results of the two pairs scenario (scenario1)
After we have run PEPA to analyse 802.11b performance, we have obtained different results. We have used \((r, \lambda_{oc}, \mu_{difs}, \mu_{sifs}, \mu_{eifs}) (200000, 100000, 20000, 100000, 2747.3)\) respectively, to measure the utilisation and throughput, and to better understand the model behaviour.

The given formula for the channel utilisation.
\[
\text{Channel utilisation} = P[\text{Medium} \land (\text{Pair} A2 \text{ or Pair} B2)] + P[\text{Medium} \land 1] + P[\text{Medium} \land 2]
\]

\textbf{Figure 4} shows the channel utilization rate increases if the packet payload size increases, for the data rate (1, 2, 5.5, and 11) Mbit/s. Because the occupied channel time increases as the packet payload size increases. In 11Mbit/s the packet can send faster for actual rate transmission. Hence, we can see the channel utilization rate in (1Mbit/s) is increasing while the packet payload size is increasing for the same speed and similarly, it is the same for all speeds, accordingly we can see that the actual transmission rate will faster. Likewise, the probability transmission for the channel utilisation increases once the packet payload size increases also, see the \textbf{Figure 5}. However, the channel throughput decrease when the packet payload size increase, this is because the channel occupancy time always is increasing with increasing the packet payload size from 700 -1500 bytes, see the \textbf{Figure 6}. Finally, in throughput if we have faster backoff, we need less time to transmit. Which means we will get faster transition in
fewer time. Once, the backoff ends successfully, then the medium can use equally by each pair A and B for transmitting data, formally the sender receives an ACK.

Figure 5: Probability transmission for the channel utilization (scenario 1).

Figure 6: Total throughput for both pairs A and B (scenario 1).

In this scenario, the obtained results of each pairs are equal as they are symmetric. They are equally occupying the channel. Figure 7 is shown the channel utilisation for pair A (it is similar for pair B), for each speed it is increasing when the packet payload size is increasing, as a channel occupancy time is increasing too. Hence, each pair can access the medium equally. Finally, we can call this scenario as a fairness scenario.

Figure 7: Channel utilisation rate for pair A in (scenario 1).

5.2 Performance results of the three pairs scenario (scenario 2)

The three pairs scenario is another scenario to demonstrate effects of unfairness. In this system both external pairs (A/C) are fully independent. And they are using the channel equally. But, the behaviour of the central pair (B) are not the same as external pairs. The rate declarations and parameters in this scenario are the same as previews scenario. Finally, measuring the probability of transmit of this model has calculated by this formula:

\[
\text{Probability of transmission} = \Pr(\text{Medium} F \text{ and } (\text{Pair}_A2, \text{Pair}_B2 \text{ or } \text{Pair}_C2))
\]

Firstly, we have studied on the probability of transmission. It is equal to 78% for 1Mbit/s then acknowledge it, also it is increasing by increasing the packet payload size. But the probability of transmission decreases for (1, 2, 5.5, and 11) Mbit/s respectively. It seems the all pairs are compete to access the medium, see the Figure 8.
Additionally, we have studied on the channel utilisation for externals, it increases if the packet payload size increases. As long as both External pairs are transmitting at the same time without collision. This is caused by the channel use time increasing, because of the packet size is increasing at the same time, they can occupy the channel equally as two symmetric pairs. See the given formula and the obtained results in Figure 9.

\[
\text{Channel utilisation} = \frac{\text{Total utilisation} \times \text{Throughput A}}{\text{Total throughput (AckA and AckB)}}
\]

Moreover, the channel utilisation of the central is similar to externals, but, it is much lower than the externals, because of the central has very limited to access the channel, most time the channel is occupied by the externals, see the Figure 10.

\[
\text{Channel utilisation} = \frac{\text{Total utilisation} \times \text{Throughput B}}{\text{Total throughput (AckA and AckB)}}
\]

As the channel utilisation rate increase we understand that the unfairness unusual, so for fastest transmit we need to increase the packet size or transmit at lower throughput.

Finally, the channel throughput decreases as the packet payload size increases. But, it is not like the channel utilisation rates, because the channel occupancy time. The fastest channel in transmitting packet will occupy less time in this channel, Figure 11 shows externals and Figure 12 central. However, accessing the channel by the central pair is limited compare to the external pairs. In term of throughput, this scenario is unfairness and it is not significant for all pairs. Clearly, the central pair is out competed by others and it is unfairly disadvantaged but the external one is fairly advantaged.
6 Conclusion

WLAN is commonly used by users around the world, and easily installation support users to make a connection between two or more nodes without using cable. Today, numerous WLAN devices are based on IEEE 802.11 protocols, and nowadays many researchers have studied on the performance of IEEE 802.11 protocols. Specially, medium access methods (PCF and DCF). In this paper we have concentrated on the DCF mechanism, and we have studied and analysed the performance of IEEE 802.11b protocol in term of the throughput and utilization using PEPA. We studied scenarios with two and three transmitting node pairs to better understand fairness. In the scenario 1, once the packet payload size increases, the utilization of both pairs increases too or slowing transmission are staying longer. The speed of packet is effected, when the speed of the transaction are increasing, and the packet can send faster for actual rate transmission, this scenario is a totally fair scenario because each pairs can access the channel equally. However, the scenario 2 shows the channel access and medium sharing is unfair, because the central pair has less chance to access the channel compared to the externals. The central one cannot compete the other pairs, hence, we can understand the central pair is unfairly disadvantaged, as it waits most of time to use the medium while the external pairs have access. Finally, in term of throughput with the faster backoff, the transmission are complete in a shorter time, then the medium can be used by each pairs and the sender receives an ACK.

In the future, we will expand our experiment to investigate and study on hidden node scenario and four pairs scenario. In the hidden node scenario, transmitting nodes cannot hear each other but the destination nodes can hear more than transmitter, hence collisions can occur. The four pairs scenario is an extension of the three pairs scenario (scenario 2) considered here, where there are two central pairs. In this scenario the external pairs will gain less advantage and the access to the medium should be fairer. The next step beyond these extended scenarios, is to consider more recent versions of the 802.11 protocol, namely 802.11g and 802.11ac, which have replaced 802.11b in modern deployments.

7 Acknowledgements

The financial support by Kurdistan Regional Government is acknowledged. And the authors are an extremely grateful to Leila Kloul for her help in the early stages of this work.
References


UKPEW 2015 – http://ukpew.org/


An Inter domain Adaptive Management architecture for Internet Service Providers (ISPs)

Ahmad Kamal Ramli and Karim Djemame*

Abstract

Internet Service Providers (ISPs) provide Internet access to their subscribers with agreed terms and conditions. Quality of Service (QOS) is the mechanism to appraise subscribers satisfaction on their provided services. The current business model between ISPs has increased competition on the bandwidth prices, while the negotiations between them are exclusive among themselves and are inaccessible to the public knowledge. Users are becoming active subscribers in line with attractive offers from available suppliers. To mitigate this situation, a Service Level Agreement (SLA) is a contract signed embedded as the legal element within the QOS framework to monitor the fulfillment and violation of terms of running services. The novel contribution of this paper is an adaptive QoS provision and monitoring feedback inter - ISPs with the support of SLAs, bandwidth management and user profiling using an autonomic computing ecosystem. Early experiments show an improvement in terms of throughput, Internet Protocol (IP) traffic drop, and application response time in the autonomous environment as compared to a non-autonomous environment. These outcomes are an example of substantial performance of enterprise routing, client-server application and application profiles throughout inter ISPs architecture.

1 Introduction

Internet access is the vital catalyst for online users, and the number of mobile subscribers is predicted to grow from 6.7 Billion in Q2 2013 to 9.3 Billion by the year 2019 [1]. To ensure demand is within the bandwidth capacity of current Internet Service Providers (ISPs), a mechanism is needed to ensure user satisfaction for their services. An ISP has the opportunity of getting more revenue by imposing extra cost for services through peering inter domain networks such as metro Ethernet, Enhanced Interior Gateway Routing Protocol (EIGRP) and Border Gateway Protocol (BGP). However, ISPs have their own limitations such as tiers, available bandwidth and business models to serve current and potential end users and corporate customers. This scenario leads to an ongoing market price issue and at present there is no concrete solution to undertake that effectively. To streamline the existing architecture running on private or

*School of Computing, University of Leeds, scakr@leeds.ac.uk , K.Djemame@leeds.ac.uk
public networks [2], an adaptive solution regarding the policies of getting the right agreement and negotiation in decision making must be available within the ISPs neighborhood or inter domain networks. An adaptive architecture inspired by autonomic computing will ensure the process is transparent between autonomic elements in inter-domain operations. The key of this architecture is the extension of self-* capabilities [9] designed to have self-management over user behavior, available network resources and interactions with neighboring ISPs [6]. This enterprise feature is considered, to some extent, in organic computing, autonomous Software Defined Networking, Software Defined Networking and will address the problem statement between inter ISPs services. A Service Level Agreement (SLA), is the utmost legal catalyst to monitor any contract violation between both end users and ISPs and is embedded with Quality of Service (QoS). It will strengthen and advance the quality of control over the application and network resources and can be further stretched out by leveraging into ensuring permitted time in mission-critical applications, better user experience and manage the costs of the resources efficiently. A considerable amount of literature has been published on the development of autonomous approaches [13], [18] and inter domain architecture [7] [13]. However, little attention is given to self-management features with the Monitor, Analyse, Plan and Execute (MAPE) Model [9] to support the services running through inter ISPs environments. Recently, an adaptive framework for link prices at the maximum of two ISPs and inter ISP connection using agents rather than autonomic elements was proposed. As for industrial environments, CISCO proposed Application Centric Architecture [22], a robust approach and limited to the local network and is famously known within Software Defined Networking (SDN). Lastly, Autonomic IT Management as a service for cloud computing is presented in [20] but left the inter domain architectures for further investigation.

Among the substantial input of this paper is to fill in this missing gap by providing a novel adaptive self-management architecture for ISP services. The core contributions are:

1. Introduce a novel adaptive architecture for managing global and local policies such as Service Level Agreements, Quality of Service, Bandwidth Management and User Profiling.

2. Introduce a novel autonomic element as a broker to identify a number of ISPs with their offers and proceed with comprehensive negotiation to perform service deployment between ISPs.

3. Introduce a novel negotiation element for inter ISPs to support service deployment that matches the established criteria, such as pricing, routing distance, and category service quality.

The computing context will impact the achievement of the fully autonomous system. It requires ISPs negotiation as the continuous element to ensure ability to exercise any definite impact between the parties. In the same way, the usage context reflects end-users and external systems that connect with autonomous system and places where the interaction will be executed. A combination of computing and usage context is the key of outlining the entire architecture. Through early experiments, we demonstrate the capability of an autonomous environment in handling network routing, client-server applications, and user
profiles with different regions using Border Gateway Protocol (BGP) or non-BGP routing. Three core metrics are used for evaluation; throughput, IP traffic dropped, and Application Response Time which are a combination of elements in the environmental context an administration in the computing environment [21]. The remaining of this paper is structured as follows: the related work is presented in Section 2, whereas section 3 discusses research challenges and methods to address the research problem. In section 4, we present the proposed architecture for this work, details of negotiation, brokerage and autonomic elements. The simulation and early experiment findings are discussed in section 5 and finally, in section 6, we draw the conclusion and future work.

2 Related Work

In the perspective of bandwidth management allocation, the research in [7] focuses on the implementation of bandwidth management in the intra domain environments. Autonomic management is the key to this exercise to ensure the availability of the bandwidth with sufficient justification between network resources. Although the result grants that it can control bandwidth allocation from one edge router to another end within the small network, further research should examine the large scale of networks with various routing protocols. The reason is to ensure the autonomic management with robust enhancement can operate all the differences in routing technologies. Alcaraz et al [15] explored another method for supporting bandwidth reservation. Four main elements are highlighted: Primary Network (PN), Secondary Network (SN), Primary Users (PU), and lastly Secondary Users (SU) as the perimeter that will be the input into the Markov Reward Model. A bandwidth reservation scheme is proposed by which the PN keeps a set of adjacent channels free of PU transmissions. These retained channels merely accommodate PU traffic when all the non-reserved channels are used, and the SU simply occupies the available channel within the engaged spectrum. In this theory, secondary users are not limited to persons; this can be extended to appliances, running algorithms such as Bayesian and multichannel access. The results show that, in non-congested PN with activity coming from Secondary Users, the interference reduction capability of Bandwidth Reservation increases the comprehensive capacity of the PN compared to not using Bandwidth Reservation. In the latest development by Peter Vrancx et al, [19] the research differs in terms of contributions and extensive approaches using autonomic computing rather than partial reinforcement learning to manage the major issues which are identically addressed in the introduction to this paper. Research on the subject has been mostly restricted to the limited comparison such as; One ISP learns in a stationary environment; One ISP learns in a nonstationary environment, and Two ISPs learn simultaneously. Instead of autonomic computing self-features, the approaches applied to this research are based on Learning Automata (LAs) to understand and adapt the best link prices to associate with. The assessment of the work done by Valancius and Lumezanu [16] is different on the tiered pricing with the transit market issues. A recent and thorough review made in [8] shows that adaptation in techniques with MAPE architecture are widely implemented in various applications and among are the real time applications. It is a strong argument how this solution within this proposal can be forecast in terms of future development ecosystems.
and support from different types of applications.

3 Research Challenge

The Internet is the combination of various tier inter networks, which supply bandwidth as their core services among the ISPs. Globally, there are connected from different regions and identified by a unique autonomous system controlled by the ISPs using their core routers, this architecture available in figure 1. The router can be a single or a group of autonomous systems depending on the setup and architecture of each ISP. In this research, the core challenge is to have an adaptation mechanism to react on the available resources that cross over inter-domain connections between ISPs. Four issues are identified and require research and assessment to be incorporated into the enterprise approach using autonomic computing, which are;

- Quality of Service
- Service Level Agreement
- Bandwidth Management
- User Profiling

Figure 1: ISPs Tiers Architecture.

QoS is the standard mechanism in the networking connection to assess how well the network establishment from one point to another with designated metrics. QoS is commonly deployed when network bandwidth is limited and is sometimes needed to review network performance in the event of congestion. The concept of QoS itself can perform classification, marking, shaping, policing and queuing. In the ISP environment, the steps for QoS execution are engineered as below;
Recognize Application Traffic (Classification and Marking)
• Prioritize (Queuing and Shaping)
• Buffer Tuning
• Throttle Traffic (Policing and Weighted Random Early Detection)

Corporate customers are equipped with software or devices such as traffic shaper or bandwidth management to monitor and configure provided services from an ISP. However, the solution is limited to the enterprise, and these are not applicable to the entire ISPs ecosystem. An inter-domain ISPs connection has different costs and depends upon the connection types. At this stage, ISPs are motivated by their business model to choose which offers match which their requirements. Furthermore, prices of the routing are different for ISPs, depending on the tiers and quality. There are three core categories available such as gold, silver or bronze. On the other hand, end users always aim for the low-cost price and good service quality. As per this model, some ISPs with an overloaded network will have problems in meeting their end-users satisfaction, whereas ISPs with the low load network will offer better prices to attract the active customers, this situation will be continuous and jeopardize the transit market price. SLA is the key to bind all the terms and agreement between ISPs and end-users. However, with the conventional process, the adaptive model in observing negotiation and monitoring on the violation of the SLA via brokerage activities is not present and requires an urgent attention with the growing number of ISPs. A successful model will enhance the way of controlling bandwidth and performing user profiling to ensure all bandwidth activities are well presented to the end users to satisfy Service Level Objectives (SLO). Table 1 summarizes the research challenges and relations to current ISPs operation.

3.1 Method

Adaptation has attracted considerable research interests in various disciplines, including computing with a focus on self-Adaptation [8]. The adaptation control can be classified into three core areas such as approach, adaptation decision criteria and degree of decentralization. Figure 2 shows the information about the self-adaptation taxonomy.

The central question for the adaptation time is how long it can tailor to the situation. It is actually the input from user’s perspective to evaluate and justify the adaptation with regards to all the arguments connected to their situations. This approach can be realistic with the usage of MAPE (Monitoring, Analysis, Planning and Execute) [9]. In the early phase, there are two processes, which are monitor and execute. This is due to human interference playing a major role [14]. Once the adaptive architecture is developed, analysis and planning will be the next. Human interference will take place to develop a knowledge foundation on the frequent decision making. By causing this, the adaptive system, with an outstanding knowledge base will be guided by growing artificial knowledge.

This research will establish autonomic elements that will be contained within the adaptable management architecture. The elements will be the policy exchanges between the local and global autonomic managers, whereby the overall management will be controlled by enterprise adaptive architecture. Inter-
Table 1: Comparison of QoS Research Contributions and Relation to Provision.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Current Research Problem</th>
<th>Proposed Novel Solution</th>
<th>Benefit after completion of research activity</th>
</tr>
</thead>
</table>
| Inter-ISPs QoS provision | No back to back agreements exists with tier issues in ISPs                                 | Architecture of Adaptive Enterprise Management to incorporate the elements of ISPs provision | • Transparent agreements and terms of the end-users  
• Better network Management  
• A system able to react adaptively to the available resources |
| QoS load balancing | The load balancing feature available for intra domain                                        | Inter domains, load balancing resource management                                          | • Load balancing in using available resources between ISPs |
| QoS Services  | Issues of Integrated services and differentiated services.                                 | Adaptive Service Level Agreements mechanism to ensure end-users satisfaction               | • Transparent Service Level Agreements between users and ISPs. Users are allowed to have adaptive terms in the best effort and guaranteed services |
| QoS performance | Issues with an uncertainty of network performance either in intra networks or entire networks. | Bandwidth Management supported through usage based module on the profiling basis          | • Transparent billing over the convergence of ISPs network with the ability to place their preferences on the usage of applications |

connection latency as a QoS provision element will support the bandwidth management framework.

4 Proposed Architecture

A QoS Broker is introduced to keep the successful negotiation, document or known as Service Level Agreement to be appraised by the Local Autonomic Manager within the same tier on the cost-saving, effective routing that concludes as the latency issue and will access by the enterprise autonomic manager for the result and the governance of the autonomic computing ecosystem within this high level architecture.
The connection between the local autonomic manager and the enterprise autonomic manager over autonomic computing main architecture is addressed in figure 4. This enterprise autonomic manager and the local autonomic manager consist of the same K-MAPE model design for autonomic computing. With the abilities to minimize expert’s intervention by providing managed elements, it will then develop betterment of the knowledge base which runs recursively in the iteration procedure. Within this architecture, K-MAPE will evaluate the exchanging elements either from local and enterprise process as well as system response. The system will have four core functions, which are monitor, analyze, plan and execute to ensure the environments are running smoothly and intelligently updated with every new alert.

The QoS Broker as per figure 5, contains two more sub-elements, which are the vital to this operation. First, one is the trailing of user profile between brokers, and secondly the negotiation over the terms and services with other ISPs. User Profiles model solver inherits three main databases such as user profile metadata, SLA commitments and Internet Service Providers Metadata. By having databases, it will develop concrete user profiles and will carry credential data to be further evaluated during the negotiation process. The technical decision of the bandwidth management model solver is in the response to the information supplied by QOS negotiators through the negotiation process. As discussed earlier, the key for valuation is bandwidth, and will focus heavily on the latency issues. Offers from the Internet Service Provider will ensure the establishment of a connection between ISPs with the support of autonomic computing ecosystem.

The final proposed architecture sub-component is the Meta negotiation architecture as illustrated in Figure 6. This is component inspired by the one proposed in [18] and is divided into six negotiation layers.
5 Early Evaluation

The main aim of this early assessment is to have a model of autonomous environment evaluated through simulation. Self-properties are the ultimate metrics to ensure the MAPE model is able to simulate the considered scenarios. The OPNET software [23] selected to ensure inter ISPs environment with various backbones routing protocols can be seamlessly executed with three major metrics as addressed earlier.

5.1 Objectives of the experiments

There are four main objectives, which have been identified for this evaluation;

- To demonstrate that Border Gateway Protocol (BGP) with autonomous system is relevant to the high-level architecture described in Figures 3 and 4.
- To justify the transmission of digital data applicable and measurable in the autonomous environment.
- To ensure the autonomic elements as per Figure 6 are available for the BGP connection.
- Following this early experiment to drive future research activities in autonomic computing environments.
5.2 Setup

In this experiment, there are two scenarios, one with without Border Gateway Protocol (GBP) and another one with BGP environment. Border Gateway Protocol chosen as the protocol to justify the autonomous concept unoccupied in the autonomous system uniquely present for each server and how later it can connect within the same autonomous system number to create a group of neighborhood ISPs or with different neighboring routers. The following Table 2 show the scenario information.

5.3 Results

The evaluation will be based on three metrics which are throughput, delay and response time. Below are the criteria for the assessment;

i. Throughput.
   As for this part, it will have two logical subnets from each region as per Figures 7 and 8; each of them is an individual scenario. In the case of the scenario without BGP, the output will be straightforward because it will be measured from one region to another region through one ISP. Whereas, for the next scenario, the environment will be in the BGP mode and the connection from one region to the Internet connection will be based on the dedicated autonomous system and neighborhood concept. All of this assessment will be for the outgoing packets rather than incoming packets. The outgoing packets will ensure the connectivity from one point...
Table 2: Scenario Parameters for Simulation Exercises.

<table>
<thead>
<tr>
<th>No</th>
<th>Description</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Internet Service Providers</td>
<td>2 (Non BGP) and 3 for BGP</td>
</tr>
<tr>
<td>2</td>
<td>Country within ISP</td>
<td>One Country with two regions</td>
</tr>
<tr>
<td>3</td>
<td>Logical Subnet</td>
<td>Two Subnets</td>
</tr>
<tr>
<td>4</td>
<td>Router</td>
<td>2 (for Non BGP) and 10 for BGP</td>
</tr>
<tr>
<td>5</td>
<td>Number of Nodes</td>
<td>One group of workstation for both subnets. Three servers located within one of the logical subnet</td>
</tr>
<tr>
<td>6</td>
<td>Cloud Connection</td>
<td>Using Cloud32. It is worldwide internet connection and connected via PPP DS3. It is also known as T3 line and the signal transmission up to 45 MB per seconds.</td>
</tr>
<tr>
<td>7</td>
<td>Application Supported Profiles</td>
<td>Using two Application Servers, Email and Streaming. Every workstation will have profiles that identical to both of the application server and with additional http browsing activities</td>
</tr>
</tbody>
</table>
Figure 6: Meta Negotiation Architecture with multiple brokers.

Figure 7: OPNET Academic Modeller BGP Environment Design.
to another from distinctive routers will be carefully measured to produce the designated finding.

- Results of the simulation software

  In Figure 9, the outcome based on the normal network connectivity using non BGP environment from one region to another is shown, whereas for Figure 10, it is a BGP environment with a dedicated autonomous number from one router to another. Figures 11 and 12 are the results of connection from one region with multiple BGP and it is within multiple BGP neighborhoods. Results are measured by quantum of 10 minutes and evaluates in the number of bits per seconds. In Figure 8, it took slightly a duration of two minutes before the performance reaches the 62,000 bits per second and the amount of packets, and this graph is really much different with BGP environments as presented in Figures 10-12, whereas the increments for a bits per second rocketed at 0.5 minutes. In this early feedback, we can conclude by using non BGP environment, the throughput will be efficient at the start of the simulation after 4 minutes, whereas using BGP, network performance is better as seen in Figures 11 and 12.
Figure 10: Point to point throughput in West Malaysia using BGP.

Figure 11: Point to point throughput in West Malaysia uses multiple BGP (A).

Figure 12: Point to point throughput in West Malaysia uses multiple BGP (B).
ii. Response Time (Application)

Upload response time and download response time will be the indicator for this simulation to measure the performance of both scenarios. It will simulate one email server with two groups of workstations from different regions running email activities. Each of the activities has been profiled using application profile feature within OPNET application.

- Results of the simulation software

In Figure 13, we can see the performance of average response time for BGP environment is constantly at 0.024 seconds from beginning of 2 minutes simulation until end to the simulations time. This situation is different without BGP environment, and the results are inconsistent with high processing loads. On another note, the average of uploads response time for email activities in Figure 14 is slightly inverse from one to another between BGP and without BGP. Performance of BGP is better with the beginning of 0.07 seconds until the end simulation with 0.15 seconds. With this result, it shows in BGP environment, the application runs smoothly within the autonomous environment and is capable of sharing the load between ISPs neighborhoods.

iii. IP Traffic Dropped

The last indicator for this simulation will measure the quality of internet
protocol traffic dropped between subnets while executing different types of applications over the networks.

- Result of the simulation software

Figure 15 shows the evaluation of IP traffic dropped while running all the applications and services with and without BGP. At the start of the simulation, BGP environment scored lower 1 packet per second dropped compared to non BGP and at minute 1, both reach the peak dropped, where the non BGP still have the highest number of dropped packets per second. This result remains consistent for BGP from minute 4 till the end of simulation, whereas a lower number of packets are dropped against non BGP.

6 Conclusion

This paper has proposed a novel ISPs architecture to provide QoS. The solution is designed for ISP architecture to integrate with additional elements of SLA, Bandwidth Management and User Profiling in the adaptive ecosystem of autonomic computing. Global and Local Autonomic Managers are the autonomic elements to manage ongoing negotiations between ISPs to ensure the establishment of services as per signed contract between agreed parties.

Preliminary simulation experiments show that the autonomic computing approach executed smoothly in the BGP environment, which is backed up with specific metrics to support this statement. To date, there have been continuous and promising algorithms available in binomial heaps, Bayesian network and reinforcement learning as suitable options for QoS provision. Figure 16 presents...
the plan of continuing research to enhance existing ISPs architecture.

Adaptive technologies are promising research domains and have grown tremendously to academic and industrial disciplines. Future work on the adaptive architecture introduced in this paper includes:

- Synchronization of Software Defined Network (SDN) at the local and within an enterprise network with the ISPs architecture to produce robust services and automation of application, routing coming from every preferred layer within the networked environment.

- To streamline adaptive approaches with existing TCP/IP technologies. ISPs because they cannot proceed with the advancement of routing technologies due to limited resources in the hardware investment, such as appliance for IPv6, Metro Ethernet and Routing protocol. Due to these limitations, ISPs are using emulators to emulate various devices with different technologies to provide services as agreed with end-users.

References


UKPEW 2015 – http://ukpew.org/


[23] Chen Jiongcong, Pan Wei, Hu Chunchao, Ma Kai, Cai Zexiang and Ouyang Jun, Research on networking strategy and optimization simulation of process level network in smart substation based on OPNET, Electricity Distribution (CICED), 2014 China International Conference on, pp. 700-705, IEEE
Index

Abdullah, C.O., 110
Awan, I., 81
Cala, J., 0
Djemame, K., 122
Fernandes, P., 9
Kamil, S.N.S., 33
Llwaah, F., 0
Namanya, A.P., 81
Nitze, A., 72
Pagna-Disso, J.F., 81
Phung-Duc, T., 47, 96
Ramli, A.K., 122
Sales, A., 9
Schmieterdorf, A., 72
Sheikh, A., 61
Thomas, D., 24
Thomas, N., 0, 33, 61, 110
Zani, L., 9