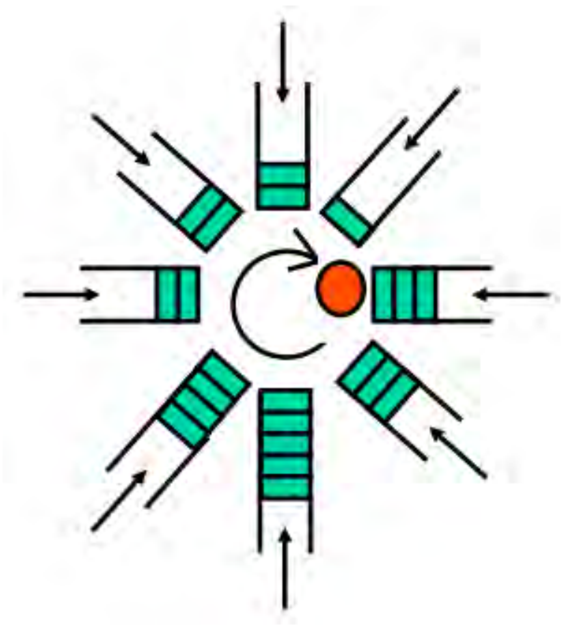
Closed-Form Waiting Time Approximation for Polling Systems with Simultaneous Batch Arrivals



Research Paper Business Analytics

June 2013

Peter van Bokhoven

Supervisor: prof. dr. R.D. van der Mei

VU University Amsterdam
Faculty of Sciences
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands

Closed-Form Waiting Time Approximations for Polling Systems with Simultaneous Batch Arrivals

Research Paper Business Analytics
June 2013
Peter van Bokhoven
Supervisor: prof. dr. R.D. van der Mei

VU University Amsterdam
Faculty of Sciences
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands

Preface

This research paper is written as part of the Master program Business Analytics at the VU University Amsterdam. The subject of this research paper is the derivation of a closed-form approximation for the expected delay in polling systems with simultaneous batch arrivals.

After attending the course Project Optimization of Business Processes where we made a simulation in Excel involving polling systems, I became enthusiastic about polling systems. The wide variety of polling system applications in the fields of computer-communication systems, production, robotics maintenance and manufacturing is attracted me and therefore, I decided to choose polling systems as a subject for this research paper.

I would like to thank my supervisor prof. dr. Rob van der Mei for his help during this research.
Furthermore, I would like to thank Irving van Heuven van Staereling for all the discussions we had.

Peter van Bokhoven
Amsterdam, June 2013

Abstract

In this research we study the delay of a customer in polling systems with simultaneous batch arrivals. Customers arrive according to a Poisson process. At any arrival epoch a batch of customers may arrive simultaneously at the different queues, according to a general joint batch-size distribution. The queues are visited by the server in cyclic order. The service times and the switch-over times are exponentially distributed and the service discipline is gated. The goal of this paper is to develop and evaluate a new closed-form expression for the expected delay at each queue that work well for arbitrary load values.

Key words: Polling systems; Approximation; Delay, Batch arrivals, Simultaneous arrivals, Heavy traffic, Light traffic.

Index

1 Introduction

In our daily real-life situations, queueing is all around us. Waiting lines in traffic situations, in supermarkets, at elevators or just at another ordinary store. Also at a more abstract level, such as communication systems, computer networks and at production factories queueing arises. To optimize or to simulate these situations queueing models are developed. This paper is about a special class of these queueing models called polling models.

1.1 Applications of polling systems

A polling system is a multi-queue single-server system in which the server visits the queues in some fixed order. Typically the server visits the queues in a cyclic order, also in this paper. The server visits the queues to serve the customers waiting at the queues, typically incurring some switch-over time to move from one queue to the next. Typical application fields of these polling models are computer communications systems, maintenance systems, traffic management and flexible manufacturing systems. Consider, for example, a Web server that needs to respond to numerous download request initiated by different users. A download from the web generally consists multiple files (e.g text, music, images), each of different size. Each of these files generates different download requests to the Web server. The server typically implements some schedule to determine the order in which these requests are handled. To implement such schedule the requests are buffered in separate queues, typically depending on the size of the download. The Web server continuously polls the different queues to check for waiting requests to be executed. In this example, the server represents the Web server and the customers represent the download requests.

Another situation, which we come across in real-life situations quite often, is a Local Area Network (LAN) in which different end users want to transmit a number of files. If such users want to transmit something over the network, transaction requests are placed into a buffer. When a user gets the right for transmission, one or more requests are handled/executed. In this example, the server represents the right for transmission, the queues represent the buffers and the customers represent the requests initiated by the end user.

The last example that will be discussed in this paper is traffic management. Waiting for a traffic light is always a cause of annoyance, impatience and loss of valuable time and money. At a traffic intersection, different cars arrive at different lanes for multiple directions. When the traffic light for one specific lane turns green, one ore more cars can pass the intersection and the other lanes must wait untill their trafficlight turns green. In this example, the server is represented by the trafficlight, the customers represent different cars and the queues represent the different driving lanes. More of these applications can be found in [3].

1.2 Previous research

The first literature about polling models is written in the early 1960s (cf. [1, 2] for overviews). In the past few decades, polling systems have received much attention. In many articles about these polling models it is assumed that customers arrive through an independent Poisson process. This means that at a particular queue exactly one customer arrives at a time.
However, in many applications customers arrive in batches at the same time at different queues. The correlation structure in the arrival processes may lead to improper performance predictions. In the examples above there could be batch arrivals that could lead to longer waiting times. Therefore polling models with simultaneously batch arrivals have received more attention since the 1990s. Levy and Sidi [4] study polling models with simultaneous batch arrivals and also van der Mei [5, 6]. For models with gated (the server serves the customers that are in the queue at an arrival point and then switches to the next queue) or exhaustive service (the server serves all customers that are in the queue until the queue is empty and then switches to the next queue) Levy and Sidi [4] derive a set of linear equations for the expected delay at each of the queues. They also provide a pseudo-conservation law (PCL) for the system with correlated arrivals. Boxma and Groenendijk [7] earlier derived a pseudo-conversation law for uncorrelated arrivals and Levy and Sidi added a term to the right hand side of this formula. This PCL is an exact expression for a specific weighted sum of the expected waiting times at the different queues. Van der Mei [5] derives a closed-form expression for the expected delay at each of the queues when the load tends to unity. This will be used in this paper together with the closed-form approximation for mean waiting times of Boon, Winands Adan and van Wijk [8]. The approximation of Boon et al. is for uncorrelated arrivals and must be extended for correlated batch arrivals.

Because of its simple form, the approximations of [8] and [5] are very suitable for optimization purposes. Although the mean waiting time of systems with gated or exhaustive services are studied, the results can easily be extended to higher moments and other service disciplines.

1.3 Goal of this paper

The goal in this paper is to extend the approximation for the mean waiting time for each queue of van der Mei [5] so that it can be used for arbitrary load values. Therefore known light traffic (LT) expressions from [8] will be used and combined together with the heavy traffic (HT) expressions of van der Mei [5]. The second goal is to build a simulation program that can test this approximation and produce numeric results together with an error rate of the approximation compared to the exact waiting time.
We consider a cyclic polling model with gated service and exponential service-time and switch-over time distributions. The correlation structure is modeled as follows. According to a Poisson process arrival points are generated. At each arrival point, batches of customers may arrive simultaneously at the different queues. These batches arrive according to a general joint batch-size distribution.
The focus is on the expected delay at each of the queues with arbitrary load values, denoted by ρ. Denoting the delay at queue i by W_i we focus on a linear combination of the light traffic limit explained in [8] and the heavy traffic limit explained in [5].

1.4 Outline

The remainder of this paper is structured as follows. In the next chapter the model description is explained together with the main result. After that we explain the simulation and produce some numerical results with error rates. In the last chapter we discuss some results and give information for further research.

2 Model

Because the approach taken is that of combining known models of [5] and [8], both models will be explained in this chapter. Consider a polling system consisting of $N \geq 2$ queues, $Q_2, \dots, Q_N$, with an infinite-size buffer. These queues are served by a single server that visits and serves the queue in cyclic order. Arrivals are generated according to a Poisson process with rate λ. At each arrival point, a random batch of size $\underline{\mathrm{K}} = (K_1, \dots, K_N)$ arrives at the queues, where K_i stands for the number of customers arriving at Q_i at an arrival point. The random vector $\underline{\mathrm{K}}$ is assumed to be independent of previous or future arrival points. Denote the joint batch-size distribution by $\pi(k_1, \dots, k_{n)} := Prob\{K_1 = k_1, \dots, K_N = k_n\}(k_i = 0, 1, \dots, for\ i = 1, \dots, N)$. Denote the arrival rate at Q_i by $\lambda_i := \lambda E[K_i]$, and let $K_{i,i} := E[K_i(K_i - 1)]$ for i = 1, … , N and $K_{i,j} := E\left[K_i K_j\right]$ for i ≠ j. Denote the total arrival rate by $\Lambda := \sum_{i=1}^{N} \lambda_i$. The (finite) k-th moment of the service time is defined by $b_i^{(k)}, k = 1, 2, \dots$. The load offered at Q_i is defined by $\rho_i = \lambda_i b_i^{(1)}$ which is $\lambda \mathrm{E}[K_i]\ b_i^{(1)}$. The total offered load of the system is equal to $\rho := \sum_{i=1}^{N} \rho_i$. Denote the k-th moment of the service time of an arbitrary customer by $b^{(k)} := \frac{1}{\Lambda} \sum_{i=1}^{N} \lambda_i b_i^{(k)}$, k = 1,2. Polling instants are defined as the epochs at which the server arrives at a queue to serve customers waiting at that particular queue. Often, two different types of service disciplines are considered in literature, gated and exhaustive. In this paper, only the gated service discipline will be handled. It is not difficult to extend the results in this paper for exhaustive and other service disciplines, but this is not in the scope of this paper. With gated service disciplines, only those customers will be served that are present at the polling instant at Q_i. Customers that arrive after this polling instant will not be served at that time, but at the next polling instant at Q_i. After serving the customers at Q_i the server immediately proceeds to the next queue. The duration of the switch-over period is exponentially distributed with mean r_i and (finite) second moment $r_i^{(2)}$. Denote the first moment of the total switch-over time by r := $\sum_{i=1}^{N} r_i$ and the second moment of the total switch-over time by $r_i^{(2)} := \sum_{i=1}^{N} r_i^{(2)} + \sum_{i \neq j} r_i r_j$. It is assumed that the interarrival times and service time are mutually independent of the state of the system. The system must be stable, i.e. $\rho < 1$, more information about this can be found in [9], and the system is in ready state. Let W_i be the delay of a customer at Q_i. We are interested in E[W_i], the expected delay at Q_i.

3 Approximation

The idea behind the approximation is that we first determine a heavy-traffic (HT) limit. After this we develop a light-traffic (LT) limit and interpolate between those two limits in the form of the following formula:

$$\mathrm{E}[W_{i,approximated}] = \frac{\alpha_i + \beta_i \rho}{1-\rho} \tag{1}$$

It is proven that capturing the behavior in an exact way, requires the (1- ρ) term in the denominator. This is not surprising, because nearly all mean waiting times of queueing systems show this behavior, except some special cases.

3.1 Heavy-traffic asymptotics

In [5] van der Mei determined the following closed-form expression:

$$\omega_i \coloneqq \lim_{\rho \uparrow 1} (1-\rho) E[W_i], \text{ for i = 1, ... , N .} \tag{2}$$

where for i ∈ G,

$$\omega_i = \frac{1+\hat{\rho}_i}{1+\sum_{m=1}^{N} \rho_m^2} \frac{b^{(2)}}{2b^{(1)}} + \frac{1}{2} r(1+\hat{\rho}_i) + \frac{\hat{\lambda}(1+\hat{\rho}_i)\sum_{j=1}^{N}\sum_{k=1}^{N} b_j^{(1)} b_k^{(1)} K_{j,k}}{2(1+\sum_{m=1}^{N} \rho_m^2)}$$

In the above expression for each variable *x* that is a function of ρ, the hat-notation is an indication for its value at $\rho = 1$. The above expression is a Heavy Traffic (HT) limit, and the rate at which E$[W_i]$ tends to infinity as $\rho \uparrow 1$ is ω_i. The proof of the expression above can be found in [5].

As said before, the goal of this paper is to extend the above expression so that it can be used for more arbitrary values of ρ. To accomplish this, in (1) we define β_i as ω_i minus the LT –limit that we will determine in the next section.

3.2 Light-traffic asymptotics

To approximate a light traffic limit we know that the system is empty when a batch of customers arrives. The number of customers a customer has to wait for is the number of customers before him in the queue and possibly the customers in the other queues depending on where the server will be at the arrival point of the batch. The idea for this approximation is to compute the average waiting time for an arbitrary customer arriving at Q_i, called a tagged customer. To do this we first compute the average number of customers that the tagged customer has to wait for in his own queue and then we compute the average number of customers that the tagged customer has to wait for in other queues. Thus, first we look at arriving customers at 1 queue. With probability p_i there will be arriving b_i customers, $i = 1, \dots, m$. So we have m different probabilities that different batch-sizes arrive at that queue. Then,

$$q_i = P(tagged\ customer\ is\ in\ batch\ type\ i) = \frac{p_i b_i}{\sum_{j=1}^{m} p_j b_j}$$

Moreover, it is easy to see that the expected number of customers before the tagged customer becomes:

E[# of customers before tagged customer]=
$\sum_{i=1}^{m} q_i E[\#\ customers\ before\ tagged\ cust.\ in\ batch\ type\ i]$

E[# customers before tagged customer in batch type i] $= \sum_{k=1}^{b_i} \frac{1}{b_i}(k-1) = \frac{b_i - 1}{2}$

Thus,

E[# customers before tagged customer] $= \sum_{i=1}^{m} q_i \frac{b_i - 1}{2}$

This is the expected number of customers that an arbitrary arriving customer has to wait for in his own queue. This was the first step, at the second step we are interested in the expected number of customers in the other queues that an arriving customer has to wait for.

We have the following joint batch-size distribution:

p_1	$(b_{11}, b_{21}, \dots, b_{N1})$
.	.
p_5	$(b_{15}, b_{25}, \dots, b_{N5})$
.	.
.	.
p_m	$(b_{1m}, b_{2m}, \dots, b_{Nm})$

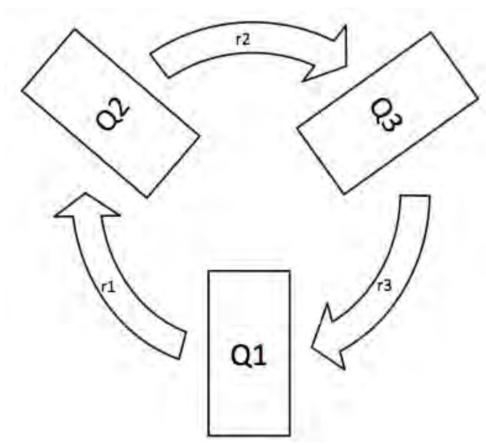
With probability p_k there will be arriving $b_{1k}, b_{2k}, \dots, b_{Nk}$ customers in $Q_1, Q_2, \dots, Q_N$ respectively. Then, again if a tagged customer is in Q_i

$$q_{ik} = P(tagged\ customer\ T_i\ in\ batch\ type\ k) = \frac{p_k b_{ik}}{\sum_{j=1}^{m} p_j b_{ij}}, \quad k = 1, \dots, m$$

E[# customers in Q_j | $tagged\ T_i$] = $\sum_{k=1}^{m} q_{ik} E[size\ of\ \ Q_j \mid type\ k]$

E[# customers in Q_j | $tagged\ T_i$] = $\sum_{k=1}^{m} q_{ik} b_{jk}$

If we take an example with 3 queues:



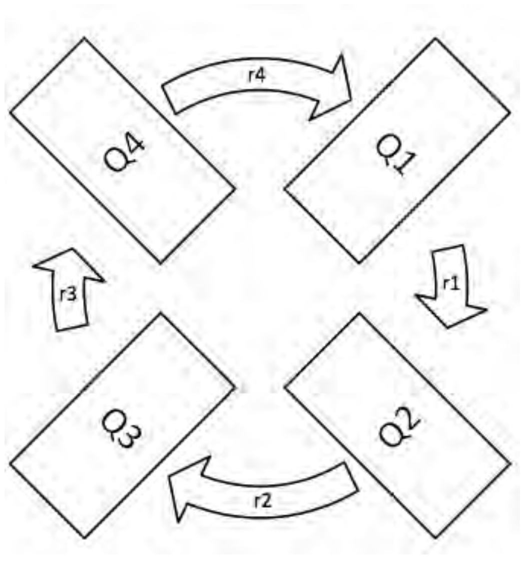
E[$W_{1,LT}$] =
$\frac{r_1+r_2}{r}\, b_3^{(1)}$E[# customers in Q_3 | T_1]

+ $\frac{r_1}{r}\, b_2^{(1)}$E[# customers in Q_2 | T_1]

+ $b_1^{(1)}$ E[# customers before tagged in Q_1]

+ $\frac{r^{(2)}}{2r}$

This can also be calculated for the other queues. With 4 queues it will be:



E[$W_{1,LT}$] =

$\frac{r_1+r_2+r_3}{r}\, b_4^{(1)}$E[# customers in Q_4 | T_1]

+ $\frac{r_1+r_2}{r}\, b_3^{(1)}$ E[# customers in Q_3 | T_1]

+ $\frac{r_1}{r}\, b_2^{(1)}$ E[# customers in Q_2 | T_1]

+ $b_1^{(1)}$ E[# customers before tagged in Q_1]

+ $\frac{r^{(2)}}{2r}$

This can also be calculated for the other queues. To generalize this we introduce the following variables:

$$a_k = P(tagged\ customer\ must\ wait\ for\ Q_k)$$

In the previous example $a_4 = P(tagged\ customer\ must\ wait\ for\ Q_4) = \frac{r_1+r_2+r_3}{r}$

$$b_k^{(1)} = mean\ service\ time\ Q_k$$

$$A_i = \sum_{k=1}^{N} [a_k b_k E[\#\ customers\ in\ Q_k]], for\ k \neq i$$

The approximation of the light traffic limit becomes:

$$E[W_i] = A_i + b_i^{(1)} E[\#\ customers\ before\ tagged\ in\ Q_i] + \frac{r^{(2)}}{r}$$

We will use this light traffic approximation for α_i in (5) for the simulation.

4 Numerical validation of the approximation

The numerical validation of the approximation will be executed in Java. As you may know Java is a programming language and open-source computing platform that is very suited for the visualization and simulation of various kinds of models in mathematics, but also in computer science, business cases and even cell phones or websites. Because of its safety, efficiency and reliability Java is chosen for building the simulation for this paper. To save time, which we will need for the analysis and simulation, we will not build a completely new simulation. Petra Vis [10] has built a simulation program for polling systems that we will be using here. However, this simulation tool is not suited for simultaneous batch arrivals, so we have to make some adjustments to this program in able it to work properly.

Together with other adjustments to determine the values of the variables explained in the model, the main adjustment we had to make was the random arrivals of batches of customers to the system. As input we have multiple input parameters. The first is the general joint batch-size distribution, for example if we want to simulate 3 queues we could have the following batch size distribution: $\pi(1,2,4) = \frac{1}{2}, \pi(2,1,0) = \frac{1}{4}$ and $\pi(0,1,2) = \frac{1}{4}$. Which means that with probability ½ there will be 1 arrival at Q_1, 2 arrivals at Q_2 and 4 arrivals at Q_3. The same holds for the other, except with other probabilities and arrivals. The second and third parameters are $r_i^{(2)}, r_i^{(1)}, b_i^{(1)}$. The other input parameters are the number of queues, the squared coefficient of variation of the arrival distribution (always 1 in this paper in order to have exponential interarrival times), the overall λ, which is the arrival rate of the batches that arrive simultaneously, and the service policy, which is always FCFS. FCFS (First Come, First Served) is a service policy whereby the customers will be served according to the order that they arrived, without preferences. This is a standard policy for the processing of most queues in which people have to wait for service. After reading the input parameters, the variables such as $\rho, \lambda_i, \rho_i, E[K_i], K_{i,j}$ can be calculated. The squared coefficient of variation of the service time distribution is set to 1 in order to get exponential service times. The output of the simulation will be the expected waiting time of the approximation of this paper and the mean waiting time of the simulation, thus $E[W_{i,app}]$ and $E[W_i]$. The next step is the simulation of the arrivals at the different queues. In order to make different arrivals, switch moments and departures possible the program makes use of so called EventQueues, which are simply events waiting in queues to be handled. Different events are created, each event has a different starting time and has a service time which is a random variable drawn from the corresponding distribution, in this paper this will be the Exponential distribution. Each of the events are placed into the EventQueue, sorted such that it will be handled according to FCFS.

In order to check the results that will be obtained by the simulation we will be using the Pseudo Conversation Law.

As said earlier in the previous research section, Levy and Sidi provided an Pseudo Conversation Law (PCL) for polling systems with simultaneous batch arrivals by adding a part to the right hand side of the PCL provided by Boxma and Groenendijk in 1987. The PCL that must be used here is:

$$\sum_{i=1}^{N} \rho_i E[W_i] = \rho \frac{\sum_{i=1}^{N} \lambda_i b_i^{(2)}}{2(1-\rho)} + \rho \frac{r^{(2)}}{2r} + \frac{r}{2(1-\rho)}\left[\rho^2 - \sum_{i=1}^{N} \rho_i^2\right] + \sum_{i=1}^{N} E\left[M_i^1\right] + \frac{\frac{1}{2}\lambda}{1-\rho} \sum_i \sum_j b_i b_j K_{i,j}$$

Where $E\left[M_i^1\right]$ stands for the mean amount of work left behind at Q_i after service at Q_i. This depends on the service discipline, in this case Gated, so

$$E\left[M_i^1\right] = \rho_i = \rho_i^2 \frac{r}{1-\rho}$$

After programming this PCL, we can simulate a polling system with simultaneous batch arrivals. However, after running multiple simulations, the errors between the left hand side (LHS) and the right hand side (RHS) of the PCL where varying between 0.01 percent and 6 percent. The simulation time of the polling system and the warm-up time of the system caused this large variation in the error percentage. Also, in the program, multiple simulation runs are executed. This is because over these runs, a confidence interval is calculated. The confidence interval that will be used is 0.005 and is calculated as follows:

$$\mu_i = \frac{\sum_{l=1}^{\# runs} EW_i[l]}{\# runs}$$

$$\sigma_i = \sqrt{\frac{\sum_{l=1}^{\# runs} EW_i[l]^2 - \frac{\left(\sum_{l=1}^{\# runs} EW_i[l]\right)^2}{\# runs}}{\# runs - 1}}$$

$$\varepsilon_i = 1.96 \frac{\sigma_i}{\sqrt{\# runs}}$$

$$interval_l = \mu_i - \varepsilon_i \quad interval_r = \mu_i + \varepsilon_i$$

When $interval_r - interval_l > 0.005 * \frac{interval_l + interval_r}{2}$ another run will be executed. It is also possible to insert a maximum number of runs that can be executed. This number of runs, together with the warm-up and simulation time, could have an enormous influence on the error between the RHS and the LHS. In order to minimize this error and execute the program with the best parameters to get reliable results, the program is executed multiple times with different values of these constants. For now, it is tested with a 3 queues, and with the same batch size distribution as on the previous page.

The full overview of the results with different simulation time and runs can be found in Appendix I. As the results clearly show, when ρ comes closer to 1, the more important it becomes to have a larger number of runs and a longer simulation time. As the results show, when less runs are performed the probability of getting an outlier is higher (the outliers are colored grey in the results). The warm-up time does not have a significant impact on the error rate, however it may not be too low, it will be set on 500.000 time units.
To let the program be more time efficient we could produce a formula for the number of runs as a function of ρ, but this is not necessary for now, the simulation time will be set on 30.000.000. After analyzing these results and setting the variables to those values that produce the lowest error percentage, we can finally simulate the polling system and test our approximation (5).

To analyze these results to see if we have produced an accurate approximation, we define a relative error of the approximation, error %, by:

$$error_i \ \% := abs\ \left(\frac{E[W_i(app)]-E[W_i]}{E[W_i]}\right)\ x\ 100\ \% \qquad (3)$$

To answer the question for which values the approximation is accurate we first perform 3 numerical experiments. The first experiment is a symmetric 3-queue model. As said before the service discipline is gated for all queues. The switch-over times are exponentially distributed with mean 1.00. The service times are exponentially distributed with mean 1. The squared coefficient of variation of the service times is 1. At any arrival epoch, the joint batch-size distribution is as follows:
$\pi(1,2,4) = \frac{1}{2}, \pi(2,1,0) = \frac{1}{4}$ and $\pi(0,1,2) = \frac{1}{4}$. The mean batch sizes can be calculated as explained in the Model section. The second model is the same as the first experiment except the mean times are (1, 1.2, 0.75) for $Q_1, Q_2\ and\ Q_3$ respectively. The arrivals at each queue are twice the amount as in the first experiment. The third experiment is the same as the second except the arrival distribution is: $\pi(2,4,15) = \frac{1}{2}, \pi(4,2,8) = \frac{1}{4}$ and $\pi(0,1,8) = \frac{1}{4}$. The third experiment is executed such that ρ_i differs from each other.

Figure 1 shows the exact and the approximated values of van der Mei and this paper.

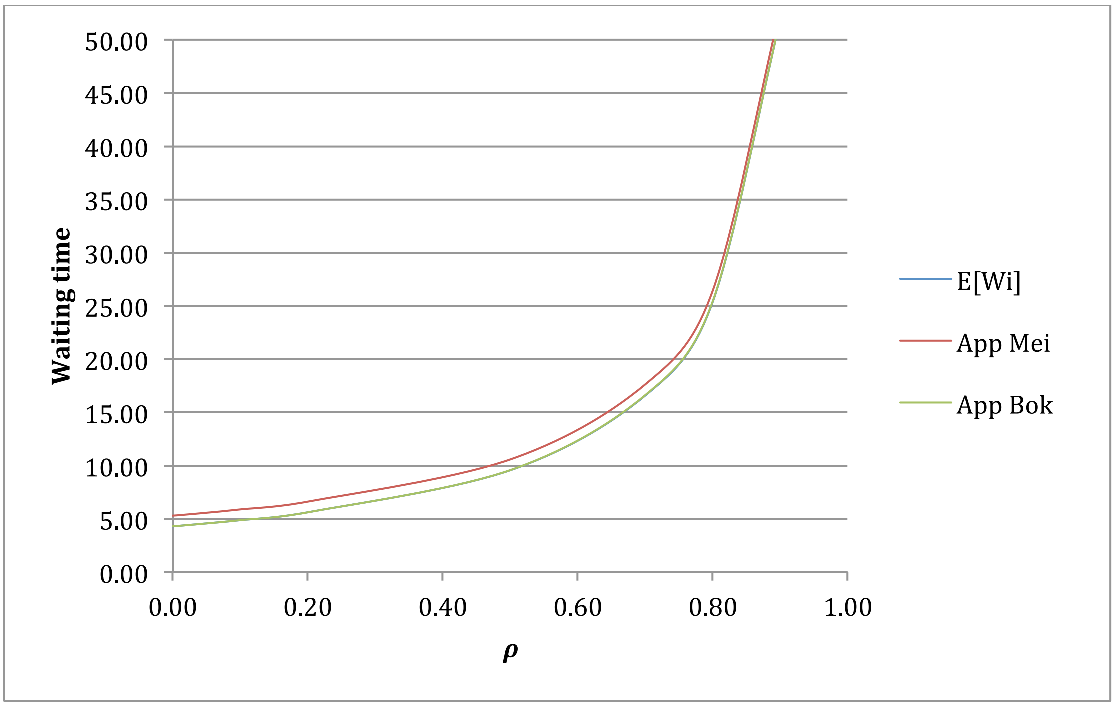


Figure 1: exact and approximated waiting times for numerical experiments

The first two numerical experiments showed the above behavior. The approximation of this paper is right on top of the exact waiting times in figure 1. The approximation of this paper is right on top of the exact waiting times in figure 1. If we look at the error rate defined in (6) we see that in the first two experiments it was varying between 0.6 and 3 %, which is better than expected. The next figure shows this error rate and compares it to the approximation of van der Mei (1).

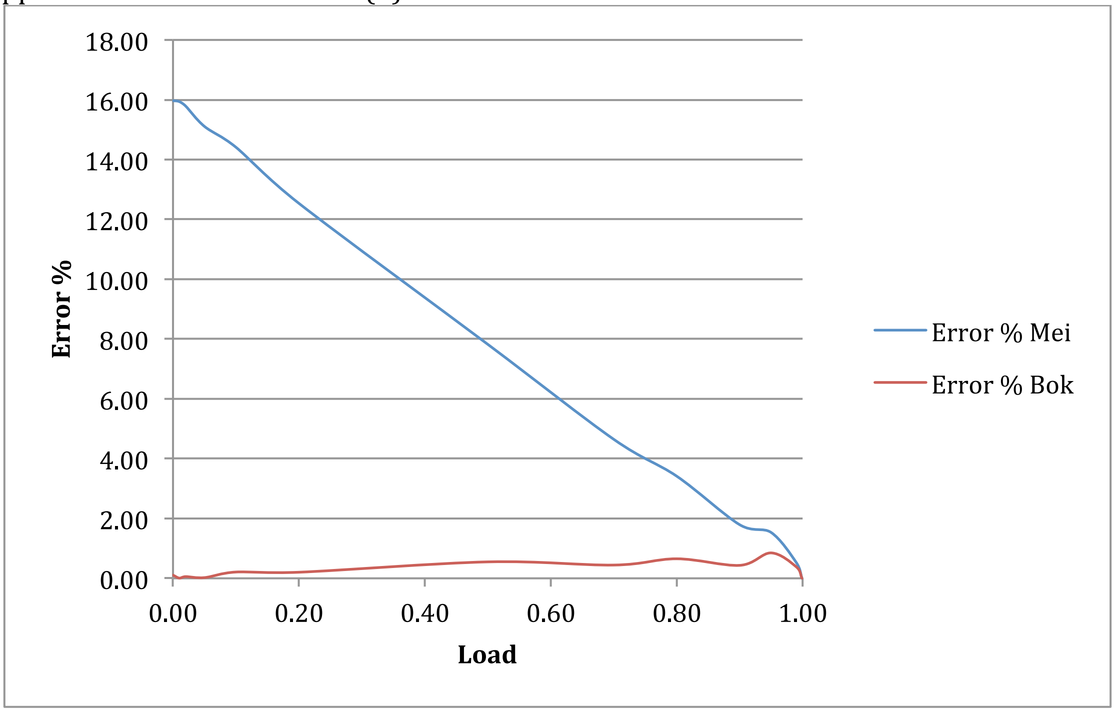


Figure 2: error rate (6) of approximation in this paper vs. van der Mei

Again, the first two experiments showed the above behavior where the error rates where significantly lower than those error rates from the approximation. As said before the error rate of the approximation (5) is low for arbitrary load values. The error rate of van der Mei is perfect for load values that are close to 1 but not for lower load values. For the third numerical experiment the error rate was varying between 2% and 9% for load values between 0.25 and 0.7. The third experiment performed not as well as the first two but still has accurate results comparing it to the approximation to van der Mei. Figure 3 shows the error rate of the third experiment compared to the error rate of van der Mei.

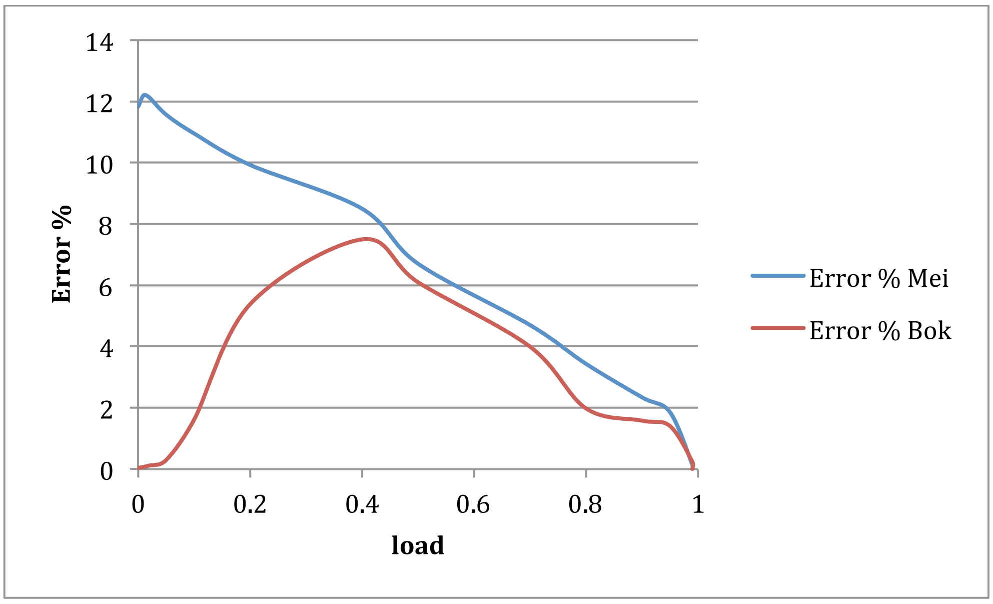


Figure 3: Error rate of the third experiment of the approximation (1) vs. the approximation of van der Mei.

As the results shows the simulation performs not as good as the first two experiments, this is due to the fact that this experiment is more asymmetric than the other two. After these experiments we tested 3 other variants: one with independent Poisson arrivals, one with independent batch Poisson arrivals and the last one with Simultaneous batch Poisson arrivals. The results are shown in table 1.

	Independent Poisson		Independent Batch Poisson		Simultaneous Batch Poisson	
Load	$E[W_i]$	$E[W_i, app]$	$E[W_i]$	$E[W_i, app]$	$E[W_i]$	$E[W_i, app]$
0.01	2.03	2.03	3.20	3.21	5.46	5.46
0.02	2.06	2.06	3.25	3.25	5.52	5.53
0.05	2.16	2.16	3.38	3.39	5.74	5.74
0.10	2.33	2.33	3.64	3.63	6.10	6.11
0.20	2.74	2.75	4.22	4.22	6.99	6.99
0.50	5.00	4.99	7.38	7.39	11.79	11.78
0.70	9.00	8.99	13.02	13.03	20.19	21.28
0.80	13.98	14.00	20.01	20.07	30.90	31.91
0.90	29.00	28.99	41.28	41.20	62.74	63.80
0.95	59.27	59.00	84.14	83.47	127.31	130.58
0.99	300.10	298.99	424.01	421.62	617.66	620.83

Table 1: Expected waiting time vs. the approximated expected waiting time

As the results clearly show the approximation works well for all variants. All of the results above show that the approximation can be applied for all kind of different models and variants. Only the asymmetric models performed slightly worse, but still has accurate results.

5 Discussion & further research

As said in the previous chapter, the approximation was not as accurate for all load values in an asymmetric model. This could be due to the fact that we did not perform enough runs or did not used enough simulation time. However this seemed not to be true, because our results show that for the error between the LHS and RHS of the pseudo conversation law the number of runs and simulation time did not have significant affection above a certain value. The chosen number of runs and simulation time where depended on the computer capabilities of the writer of this paper. Higher number of runs and longer simulation time where simply not possible, perhaps with a computer with more capabilities a higher number of runs and longer simulation time could be accomplished. In general longer simulation leads to better reflection of reality. In fact, if we should simulate an infinite amount of time, the simulation is equal to reality.

The second point we would like to discuss is the formula of Boon et al. defined in [8]. It showed an approximation for the expected waiting time. This formula is proven for individual Poisson arrivals in [8] and because of its functional form it could improve the approximation in this paper especially for highly asymmetric models. Of course it should first be extended so that it can be used for polling models with simultaneous batch arrivals. This extension is not trivial and can cost a lot of work. If this is added to the approximation in this paper, the error percentage defined in (3) could decrease even more.

The last point for further research is that the simulation used for this paper could be extended for the use of other service disciplines (exhaustive) and other distributions. This paper only handles the exponential distribution but it can easily be extended to others.

6 Bibliography

[1] Takagi, H. (1990). Queueing analysis of polling models: an update. In: Stochastic Analysis of Computer and Communication Systems , ed. H. Takagi (North-Holland, Amsterdam), 267– 318.
[2] Takagi, H. (1997). Queueing analysis of polling models: progress in 1990– 1994. In: Frontiers in Queueing : Models and Applications in Science and Technology , ed. J.H.
[3] Roubos, A (2007). Polling Systems and Their Applications.
[4] H. Levy and M. Sidi, Polling systems with simultaneous arrivals, IEEE Trans. Commun. 39 (1991) 823-827.
[5] R.D. van der Mei, Polling systems with simultaneous batch arrivals, Stochastic Models. 17 (2001)
[6] R.D. van der Mei, Waiting-Time Distributions in Polling Systems with Simultaneous Batch Arrivals, In: Annals of Operations Research 113, 155-173, 2002.
[7] O.J. Boxma and W.P. Groenendijk, "Pseudo-conservation laws in cyclic queues", J. Appl. Prob., vol. 24, pp. 949-964, 1987.
[8] M.A.A. Boon, E.M.M. Winands, I.J.B.F. Adan and A.C.C. van Wijk, Closed-Form Waiting Time Approximations for Polling Systems, 2009.
[9] Cooper, R.B., Niu, S.-C. and Srinivasan, M.M. (1997). Setups in polling models: does it make sense to set up if no work is waiting? J. appl. Prob. 36, 585-592.
[10] P. Vis, Waiting-time distributions in polling systems with non-FCFS service policies, 2012.

Appendix I

Testing the PCL error rate between the LHS and the RHS with different # of runs and different simulation time. The outliers are colored grey, an outlier is defined as error rate > 0.5. Tested model is described in section 4.

	Simulation time: 1,000,000							
	# runs							
ρ	**5**	**10**	**20**	**30**	**50**	**100**	**250**	**500**
0.1	0.218	-0.031	0.091	0.046	-0.028	0.045	0.082	-0.013
0.2	-0.001	-0.003	-0.117	-0.071	0.022	0.011	-0.123	0.057
0.3	-0.401	0.049	0.035	0.011	0.008	-0.224	0.073	0.105
0.4	0.311	0.119	-0.041	0.100	-0.150	0.010	0.113	0.063
0.5	-0.533	0.094	0.007	0.188	0.247	0.079	-0.179	-0.214
0.6	0.267	-0.370	0.184	-0.103	0.024	-0.170	0.054	-0.169
0.7	0.644	0.325	0.242	0.091	0.189	0.009	0.065	-0.039
0.8	-0.066	-0.321	0.077	0.216	0.231	-0.071	0.056	-0.042
0.85	-0.373	0.291	-0.696	-0.622	0.046	0.228	-0.105	0.042
0.9	-1.479	1.004	0.328	0.371	-0.749	-0.771	-0.258	-0.019
0.95	0.437	1.140	0.092	-2.595	-1.936	0.419	-0.164	0.356
0.975	9.207	-6.794	-5.345	-3.928	0.649	-0.050	-0.726	-0.721

	Simulation time: 5,000,000							
	# runs							
ρ	**5**	**10**	**20**	**30**	**50**	**100**	**250**	**500**
0.1	0.188	0.016	0.080	-0.015	0.013	0.076	-0.105	0.000
0.2	-0.027	0.015	-0.004	0.100	0.012	0.113	0.052	0.052
0.3	-0.120	0.068	0.100	0.046	0.028	0.364	-0.036	0.053
0.4	0.146	-0.023	-0.251	-0.077	-0.213	-0.343	0.109	0.117
0.5	0.021	-0.060	0.019	-0.029	0.052	0.080	0.111	-0.029
0.6	0.006	-0.050	-0.166	-0.030	-0.056	-0.249	0.048	-0.065
0.7	0.045	0.238	0.007	-0.096	-0.158	-0.112	0.170	-0.063
0.8	-0.300	0.227	-0.104	0.203	0.250	-0.169	-0.094	0.210
0.85	-0.080	-0.126	-0.244	-0.181	0.031	0.058	-0.248	0.116
0.9	0.473	-0.870	-0.363	0.533	0.030	0.635	-0.210	0.176
0.95	0.067	-0.798	0.321	0.659	-0.553	-0.406	-0.086	0.214
0.975	-5.229	-0.950	2.532	-0.712	-0.291	0.941	-0.176	-0.577

ρ	Simulation time: 10,000,000							
	# runs							
	5	10	20	30	50	100	250	500
0.1	0.009	0.048	-0.100	-0.041	-0.098	-0.024	-0.132	0.0282
0.2	-0.131	0.020	-0.037	0.112	-0.012	0.098	0.191	0.0705
0.3	-0.035	0.028	0.002	-0.045	0.069	0.076	-0.062	-0.1566
0.4	0.127	-0.032	0.045	0.100	-0.032	-0.029	0.127	-0.0200
0.5	-0.001	0.054	0.068	-0.135	-0.084	0.091	0.142	-0.0716
0.6	-0.084	-0.059	0.089	-0.145	0.017	0.123	-0.347	0.1388
0.7	0.019	-0.113	-0.107	-0.059	-0.094	0.045	-0.042	-0.0047
0.8	-0.260	0.290	0.319	-0.025	0.048	-0.094	-0.104	0.0239
0.85	-0.650	0.250	0.118	-0.004	0.151	-0.043	-0.039	0.2020
0.9	0.204	0.198	0.207	0.061	-0.029	-0.572	-0.019	0.0384
0.95	-0.267	1.094	-0.094	-0.405	-0.526	-0.221	-0.586	0.0980
0.975	1.363	0.134	0.836	-0.764	0.156	0.537	0.204	-0.5286

ρ	Simulation time: 15,000,000							
	# runs							
	5	10	20	30	50	100	250	500
0.05	0.010	-0.001	0.073	0.072	-0.105	-0.019	0.010	-0.066
0.1	0.028	-0.070	-0.032	0.007	-0.097	0.096	-0.086	0.003
0.3	0.007	-0.015	-0.081	-0.021	0.016	-0.022	-0.124	-0.065
0.5	-0.067	0.005	-0.051	0.158	-0.011	0.034	0.005	-0.104
0.7	-0.016	-0.014	-0.116	-0.025	0.191	-0.147	-0.076	0.034
0.8	-0.164	0.056	0.145	0.108	-0.042	-0.095	0.167	-0.001
0.9	-0.607	-0.072	0.056	-0.064	-0.012	0.068	-0.030	-0.088
0.95	0.301	1.235	0.050	0.016	0.503	0.025	-0.128	0.092
0.975	1.346	-0.798	-0.431	-0.238	-0.345	-0.157	0.328	-0.033
0.99	-5.952	-3.804	-0.485	-3.320	-0.976	0.232	0.442	0.505

ρ	Simulation time: 20,000,000							
	# runs							
	5	10	20	30	50	100	250	500
0.05	0.064	0.007	0.065	0.042	0.055	0.000	-0.067	0.068
0.1	-0.033	0.001	-0.172	-0.066	0.057	0.031	-0.084	-0.063
0.3	0.012	0.010	-0.034	0.042	-0.061	-0.012	0.086	0.032
0.5	0.018	0.075	-0.080	0.046	-0.051	-0.031	-0.019	0.084
0.7	-0.040	-0.151	0.040	0.167	-0.223	0.081	-0.032	0.094
0.8	0.378	0.005	-0.039	-0.213	-0.073	-0.001	0.235	0.070
0.9	-0.037	0.059	-0.131	-0.223	-0.087	0.151	0.277	0.190
0.95	0.394	0.516	0.135	-0.494	-0.064	0.036	-0.100	0.073
0.975	-0.263	1.393	0.834	0.192	0.245	-0.029	-0.019	0.036
0.99	-6.524	-0.777	-0.656	0.086	-0.840	-0.706	0.431	-0.042

ρ	Simulation time: 30,000,000							
	# runs							
	5	**10**	**20**	**30**	**50**	**100**	**250**	**500**
0.05	0.003	0.006	0.089	0.036	-0.032	0.108	0.089	-0.038
0.1	-0.009	0.017	-0.030	0.029	0.024	-0.004	-0.009	-0.018
0.3	-0.108	-0.010	-0.046	-0.055	0.003	0.026	-0.003	-0.030
0.5	-0.072	0.000	0.056	-0.018	0.049	-0.125	0.029	-0.037
0.7	0.087	-0.080	-0.018	-0.011	-0.035	0.114	-0.023	-0.099
0.8	0.000	0.123	0.291	-0.092	0.014	-0.071	0.108	-0.185
0.9	-0.410	-0.165	0.112	0.219	-0.077	-0.051	-0.073	0.276
0.95	0.110	0.271	-0.325	0.084	-0.138	0.335	-0.031	0.210
0.975	-0.688	-1.321	-0.634	-0.155	0.162	-0.117	0.104	0.013
0.99	-0.284	1.339	-0.886	0.417	0.167	0.044	0.269	-0.038