Chapter 1

Analysis: Queuing Modelling

This chapter describes in detail the analytical model of Prefetching on Demand strategy (PonD), to estimate the average time to serve a demand miss. Firstly, it starts with describing the standard models and its operations. Secondly, it describes the model which can represent the clustering over the network and their solutions. Finally, it presents the preliminary results of the solved models and it concludes the chapter with describing the need to explore a space where QoS for streaming applications and demand misses could be satisfied.

1.1 Analysis

From the previous analysis, we can represent the system by two queues: the demand and the prefetch queue, as shown in Figure 1.1. Let λ_d be the rate at which demand requests are arriving at the demand queue and let λ_p be the rate at which prefetch requests are arriving at the prefetch queue. While serving, more than one request could be taken from both the queues, clustered into a network buffer which is then sent off to the server. This can be viewed as a type of *bulk service*.

This analysis attempts to answer the question: Given the arrival rates of the two queues, can we find a way to calculate the average waiting time experienced in the demand queue?

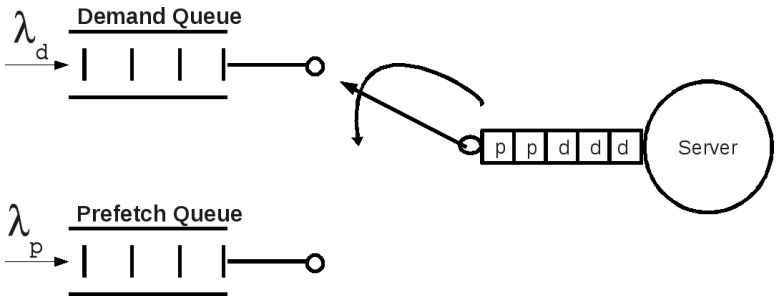

Figure 1.1: A model with a server serving two queues: Prefetch and Demand queues.

1.2 Literature

In order to answer the raised question, we looked at several polling models. A polling model is a system of multiple queues accessed in a cyclic order by a single server. In recent decades, polling models have been used to analyse the performance of a variety of systems. In the late 1950s, a polling model with a single buffer for each queue was used in an investigation of a problem in the British cotton industry involving a patrolling machine repairman [C., 1957a,b]. In the 1960s, polling models with two queues were used to analyse traffic signal control (see a survey by Stidham [S., 1969]). There were also some early studies from the viewpoint of queueing theory that were apparently independent of traffic analysis (e.g. Avi-Itzhak [Avi-Itzhak et al., 1965]). In the 1970s, with the advent of computer communication networks, an extensive study was carried out on a polling scheme for data transfer from terminals on multidrop lines to a central computer. Since the early 1980s, the same model has been revived by Bux [Bux, 1981] and others to study token passing schemes (e.g., the token ring and token bus) in local-area networks (LANs). It has also been used for resource arbitration and load sharing for multiprocessor computers [Wang and T., 1985]. A polling model was used in a non-technical article in Scientific American [LEISOWITZ, 1980] as an example of an interesting and important queuing system.

The usual objective in analysing polling models is to find the *message waiting time*, defined as the time from the arrival of a randomly chosen message to the beginning of its service. The mean waiting time plus the

mean service time is the mean message response time, which is the single most important performance measure in the most computer communication systems [Kleinrock, 1976].

Polling models are referred to in many survey articles and book chapters in data communication systems [[Bertsekas and Gallager, 1992], [Chu and Konheim, June 1972.], [Hayes and Sherman, November 1971], [R. and G.], [Kobayashi, Jan 1977], [KONHEIM, 1980], [lam, 1983], [PENNY, B. K., ANO BAGHOADI, A. A., 1979], [rei, 1982]].

The vast majority of the literature is concerned with the two traditional service disciplines, the *exhaustive* and *gated* policies. Exhaustive service means that a queue must be empty before the server moves on, whereas in case of gated service only those customers in the queue at the start of polling are served. Suggested references for readers who would like to pursue their study of the exhaustive and gated policies are [[tak, 1988], [tak, 2000], [Takagi, 1990]]. The main drawback of these traditional policies is the inability to prioritise among the different queues for improving total system performance. A more sophisticated service strategy offering this possibility is the *k-limited* service strategy. Under this K-limited strategy the server continues working at a queue until either a predefined number of k customers is served or until the queue becomes empty, whichever occurs first. Note that the case $k \longrightarrow \infty$ is equivalent to the exhaustive service strategy. In many applications of polling systems, the objective function typically depends not only the mean queue lengths, but on the complete marginal queue length distributions. Therefore, [van Vuuren and Winands, 2006] proposed to study the marginal queue length distributions in a continuous-time polling systems with k-limited service under the assumption of general arrival, service and set-up distributions.

A feasible approximate approach for the queue length distribution in a k-limited polling system is the decomposition method, in which the polling system is decomposed into vacation systems, for which the vacation distributions are computed in an iterative approximate manner. At each step in the iteration the mathematical analysis focuses on one single queue, whereas the other queues in the system determine the length of the vacation period. This

decomposition method is adopted by the present research as well. We have to remark that these decompositions methods seem to be applicable to a wide variety of queueing systems (see, e.g.,[[dal, 1989], [ger, 2000], [vuu, 2005], [van Vuuren M, 2006]]). However, the main disadvantage of this method is that time and memory requirements are exponential functions of the number of queues.

In our work, we believe that the streaming applications access blocks at a constant rate. The number of blocks that needed to be fetch for the prefetch queue to provide required QoS can be controlled. Hence, we can reduce the model to a single queueing system based on demand requests but the service time for the demand blocks will include the cost of fetching prefetch blocks. We can further simplify the analysis by only prefetching when there are demand requests in the demand queue i.e. conservative prefetching as shown by Pei Cao [Cao et al., 1995]. This strategy is also justified as it uses the network more effectively.

1.3 Standard Approach (Partial Batch Model)

As a first step to analyse the average time to satisfy a demand request in the demand queue, we will use the partial batch model described in [Gross and Harris, 1998]. In this model a server can serve up to a maximum of K requests. If there are less than K requests in the system, the server begins service on these requests. Furthermore, when there are less than K requests being serviced new arrivals immediately enter service. The amount of time required to service requests, is an exponentially distributed random variable with mean $\frac{1}{\mu}$.

This model is represented in the Figure 1.2. Each state of the model is represented in terms of n and s. n is the total number of requests in the system and s is the number of requests currently being served. It can be seen from the Figure that any new arrival enters the service immediately as long as there are less than K number of requests being served and time taken to service those requests is exponentially distributed to a mean value of $\frac{1}{\mu}$.

A stochastic balance equation for the model can be written as:

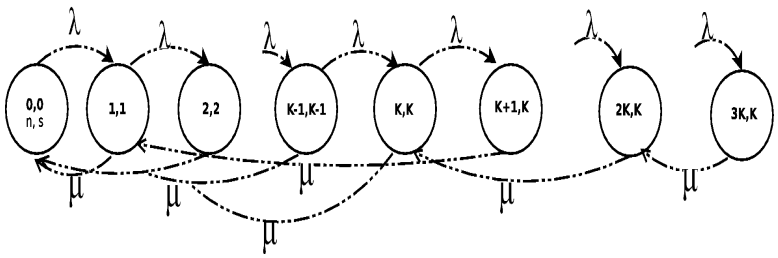

Figure 1.2: Partial Bulk Service model.

$$\begin{aligned} 0 &= -(\lambda+\mu)p_n + \mu p_{n+K} + \lambda p_{n-1} \quad (n \geqslant 1) \\ 0 &= -\lambda p_0 + \mu p_1 + \mu p_2 + \mu p_{K-1} + \mu p_K \end{aligned} \tag{1.1}$$

For $K = 1$;

$$\begin{aligned} 0 &= -(\lambda+\mu)p_n + \mu p_{n+1} + \lambda p_{n-1} \quad (n \geqslant 1) \\ 0 &= -\lambda p_0 + \mu p_1 \end{aligned} \tag{1.2}$$

The above equations are the basic equations for the M/M/1 queue. Hence, we can say the solution for PBM is same as the M/M/1:

$$P_n = P_0 r^n$$

By finding the root (r_0) of this equation that is between 0 and 1, one can work out mean queue length (L) and average waiting time (W) for the queue, using the equations below.

$$L = \frac{r_0}{1-r_0} \quad \text{and } W = \frac{r_0}{\lambda(1-r_0)} \tag{1.3}$$

The result presented in the Figure 1.5, showed that this approach is extremely accurate for very heavy traffic, since on these occasions the server will always be serving the maximum batch size. However, for lighter traffic loads the model is inaccurate because according to this approach new requests will immediately enter service, when the server is serving less than the maximum batch size which is not the case in our scenario. Here, the server only serves the number of people in the queue at its arrival, requests arriving after this point must be serviced in the next cycle regardless of whether or not the maximum batch size is being served in the current cycle. Hence, the scenario is gate-limited and not exhaustive-limited as seen in the partial batch model.

1.4 Proposed Gated-Limited Model

In this section we attempt to develop a more accurate model which could be used under operational loads. As shown in the Figure 1.3, the state of the model is defined by two variables i.e. n and s. n is the total number of requests in the system including the requests being served and s is the number of requests being served at any given time. Therefore, for the maximum batch size $s = d$, s goes from 0 to d, so when $s = 0$, the system is empty and when $s = K$ up to K requests are being served at a time. Also, excluding $(0, 0)$, this will give rise to the K different stages as shown in the Figure 1.3 with each stage having service rate depending on the number of blocks being served.

1.4.1 Simple Scenario

We have started looking at a simple scenario by restricting K equal to 2 i.e. $s = 2$, as shown in the Figure 1.4. Having K equal to 2 there can be only three stages: either server is serving 1 request or it is serving 2 requests or the queue is empty. This means that with the exception of the transition $(2, 2)$ to $(0, 0)$, each transition can only jump one stage at a time (i.e. 1 to 2 or 2 to 1) for e.g. $(3, 1)$ goes to $(2, 2)$ or $(3, 2)$ goes to $(1, 1)$. We will analyse each series individually starting with series 1.

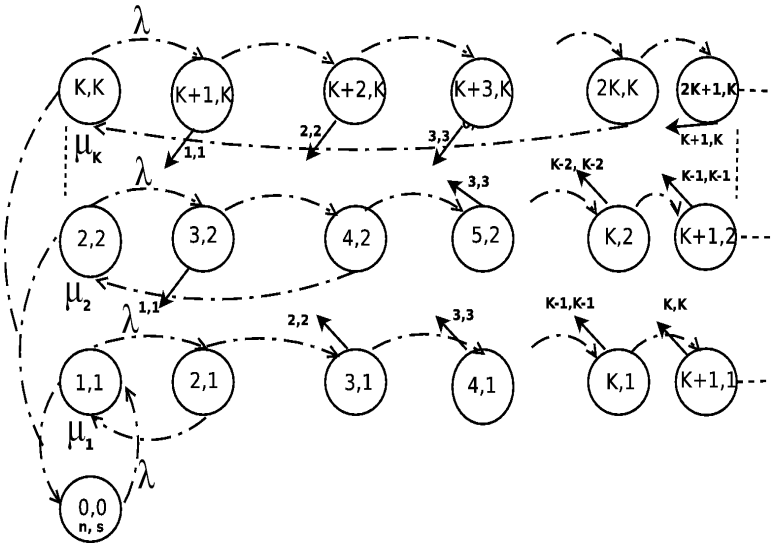

Figure 1.3: A model with a server which can serve up to K number of demand requests in a batch mode, n = the total number of requests in the system and s = the number of requests being served.

Considering Series One i.e. when $s = K = 1$

Let us consider series one of the Figure 1.4. In series one, $s = 1$ and for $n > s$ i.e. $n > 1$, we will have:

$$\lambda p_{n-1,1} = (\lambda + \mu_1) p_{n,1} \tag{1.4}$$

This implies that for any $n > 1$, in series one:

$$\begin{aligned} p_{n,1} &= \frac{\lambda}{(\lambda + \mu_1)} (p_{n-1,1}) \\ p_{n,1} &= (\frac{\lambda}{(\lambda + \mu_1)})^{n-1} (p_{1,1}) \end{aligned} \tag{1.5}$$

And for $n = s$, we have:

$$(\lambda + \mu_1) p_{1,1} = \lambda p_{0,0} + \mu_1 p_{2,1} + \mu_2 p_{3,2} \tag{1.6}$$

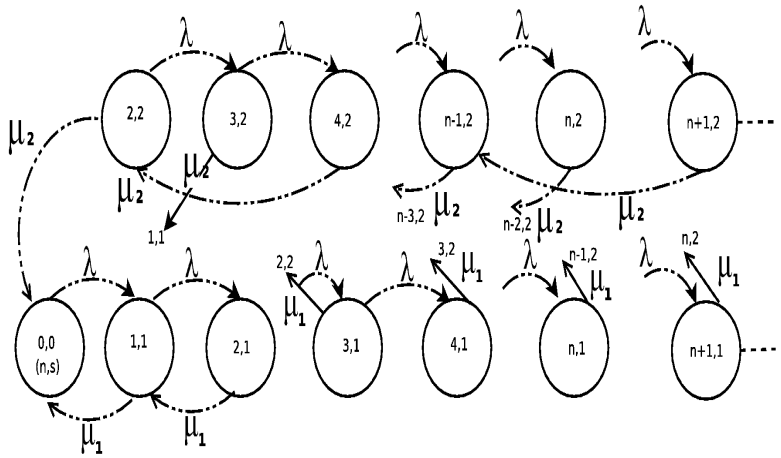

Figure 1.4: Two Stage Model, $K = 2$.

Finally, for $n = s = 0$, i.e. $p_{0,0}$ will be:

$$\lambda p_{0,0} = \mu_1 p_{1,1} + \mu_2 p_{2,2} \tag{1.7}$$

Considering the series two i.e. when $s = K = 2$

Similarly, for $s = 2$, we will derive equations for $n > s$ and $n = s$, using the Figure 1.4. when $n > s$, we have:

$$(\lambda + \mu_2)p_{n,2} = \lambda p_{n-1,2} + \mu_2 p_{n+2,2} + \mu_1 p_{n+1,1} \tag{1.8}$$

And for $n = s$, we have:

$$(\lambda + \mu_2)p_{2,2} = \mu_2 p_{4,2} + \mu_1 p_{3,1} \tag{1.9}$$

Now using the derived equations for the series 1 and the series 2, we will try to obtain an equation for stage 2 at point $p_{3,2}$[1]. We can find out the roots of

[1]Similar techniques can be used for different points of series 2, e.g. $p_{2,2}$, $p_{4,2}$, $p_{5,2}$

these equations as in the partial batch and thus will be able to find out the probability of being at each point in series 2.

$$(\lambda + \mu_2)p_{3,2} = \lambda p_{2,2} + \mu_2 p_{5,2} + \mu_1 p_{4,1} \tag{1.10}$$

From the Equation (1.5), $p_{4,1}$ can be expressed as $(\frac{\lambda}{\lambda+\mu_1})^3(p_{1,1})$. Further, from the Equation (1.7), $p_{1,1}$ can be expressed as $\lambda p_{0,0} - \mu_2 p_{2,2}$. Therefore,

$$p_{4,1} = (\lambda p_{0,0} - \mu_2 p_{2,2})(\frac{\lambda}{\lambda + \mu_1})^3 \tag{1.11}$$

Substituting the value of $p_{4,1}$ into the Equation 1.10, we get:

$$\begin{aligned}(\lambda + \mu_2)p_{3,2} &= \lambda p_{2,2} + \mu_2 p_{5,2} \\ &\quad + (\lambda p_{0,0} - \mu_2 p_{2,2})(\frac{\lambda}{\lambda + \mu_1})^3 \\ 0 &= -((\lambda + \mu_2)p_{3,2})\lambda p_{2,2} + \mu_2 p_{5,2} \\ &\quad + (\lambda p_{0,0} - \mu_2 p_{2,2})(\frac{\lambda}{\lambda + \mu_1})^3\end{aligned} \tag{1.12}$$

Now, we need to find the root r which is between 0 and 1 such that it solves the above equation, as in the Partial Batch model.

1.5 First Attempt to Solve Series Define in Section 1.4

Finding out the root (r) which will be between 0 and 1, we can find out the probability of being at each point in series 2 in terms of $p_{0,0}$. We will use the same approach as in M/M/1 queueing as well as the partial batch model

and so express $p_{n,2}$ in terms of $p_{0,0}$ as follows:

$$p_{n,2} = r^n p_{0,0} \tag{1.13}$$

Similarly, using the Equation (1.7), we can find out the probability of being at each point in series 1 in terms of $p_{0,0}$.

$$\begin{aligned}
\lambda p_{0,0} &= \mu_1 p_{1,1} + \mu_2 p_{2,2} \\
p_{1,1} &= \frac{(\lambda - \mu_2 r^2)}{\mu_1} p_{0,0} \quad \text{(substituting } p_{n,2} = r^n p_{0,0}\text{)} \\
p_{1,1} &= Const * p_{0,0} \quad \text{where Const} = \tfrac{(\lambda - \mu_2 r^2)}{\mu_1} \\
&\quad and \\
p_{n,1} &= (\frac{\lambda}{(\lambda + \mu_1)})^{n-1} * Const * p_{0,0} \\
&\qquad \text{(substituting value of } p_{1,1} \text{ in Equation 1.4)}
\end{aligned} \tag{1.14}$$

From the Equations (1.13) and (1.14), we can see that the probability at any point in the model can be known if $p_{0,0}$ is known. Also, the sum of all the probabilities at stages 1 and 2 should be equal to 1. Using this,

$$\left(\sum_{n=0}^{\infty} p_{n,1} + \sum_{n=0}^{\infty} p_{n,2} \right) = 1$$

Substituting values for $p_{n,1}$ and $p_{n,2}$ from Equations (1.13) and (1.14).

$$p_{0,0} = \frac{\lambda \mu_1 (1 - r)}{(\lambda + \mu_1)^2 * C(1 - r) + \lambda * \mu_1} \tag{1.15}$$

Once we know $p_{0,0}$, the total number of requests (L) in the system can be easily known using:

$$L = \sum_{n=0}^{\infty} n * p_{n,1} + \sum_{n=0}^{\infty} n * p_{n,2} \tag{1.16}$$

and the average time to serve a demand miss will be equal to W:

$$W = \frac{L}{\lambda} \tag{1.17}$$

1.5.1 Results of First Attempt

A simulation was developed to verify the analytical model results. In the experiment, the number of prefetch blocks (P) were kept constant ($P = 1$) and the arrival rate of the demand queue was varied. The analytical results were calculated using different points at stage 2, however, $p_{4,2}$ appeared to give the best results. The results estimates the average time to serve a demand miss (T_{D-NMS}) by the Partial Batch Model and Proposed Model, are shown in the Figure 1.5. The plotted points in the graph have 95% of confidence level with confidence interval of $\pm 5\%$. It shows that the results from the analytical model are in-line with the results obtain from the simulation and they are significantly better than the partial batch model.

The results from the first attempt show that the model appear to be very accurate at medium and high loads but inaccurate at lower loads. This is inadequate as it is unable to estimate average time to serve demand miss over the network when operating over a large operational range of λ_d. Hence, we attempt to come up with another solution based on the state of $P_{2,2}$ instead of $P_{0,0}$

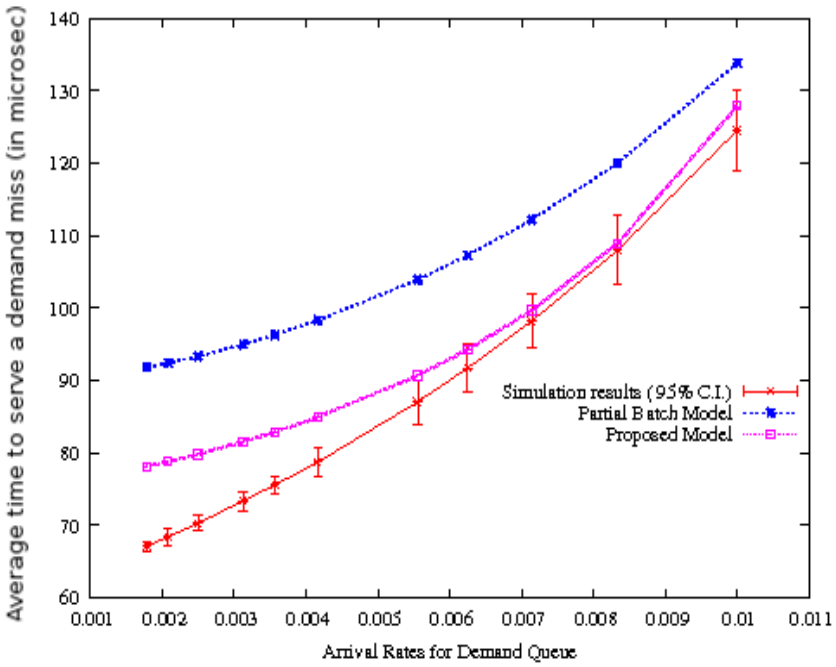

Figure 1.5: Estimates the average time to serve a demand miss (T_{D-NMS}) using Simulation, Partial Batch Model and Our approach.

1.6 Second Attempt to Solve Series Define in Section 1.4

In this section, we attempt to solve the Equation 1.12 based on the state $P_{2,2}$. First, we find out the roots of this equation using the same technique that was used in the Partial Batch Model. Thus the state probabilities of Chain 2 for $n >= 2$ can be given by:

$$p_{n,2} = r^{n-2} p_{2,2} \tag{1.18}$$

In order to solve the above equation, we imagine the second chain to be identical to a Partial Batch Model represented by Equation 1.12. This is shown in Figure 1.6. However, for states where $n > 2$, there is not real

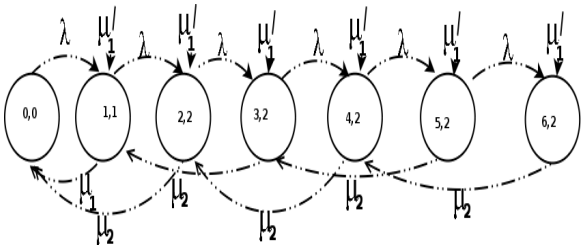

Figure 1.6: Imaginary Partial Batch Model for Chain 2

difference between the real or imaginary Chains as Equation (1.18) is valid in both scenarios. This means we can use the same approach taken in the Partial Batch Model to calculate r. Once this is done we can represent any state in the second Chain by the Equation 1.18. In addition, using the previous equations, it will also be possible to represent $p_{0,0}$ and Chain 1 in terms of $p_{2,2}$. A previous attempt by the authors to solve these equations revolved around solving other probabilities in terms of $p_{0,0}$ [Thakker et al., 2009]. However, the new approach taken here yields more accurate results.

Using Equation 1.7, we substitute for $\lambda p_{0,0}$ in Equation 1.6. In addition, we note that according to Equation (1.18): $p_{3,2} = rp_{2,2}$. Rearranging, we get: $p_{1,1} = C_{1,1}\ p_{2,2}$ where $C_{1,1}$ is given by the equation:

$$C_{1,1} = \mu_2(1+r)(\frac{\lambda+\mu_1}{\lambda^2}) \tag{1.19}$$

By substituting for $p_{1,1}$ in Equation 1.7, we can get an equation for $p_{0,0}$ in terms of $p_{2,2}$; i.e., $p_{0,0} = C_{0,0}\ p_{2,2}$ where $C_{0,0}$ is given by:

$$C_{0,0} = \frac{\mu_1 C_{1,1} + \mu_2}{\lambda} \tag{1.20}$$

1.6.1 Solving for $p_{2,2}$

The sum of the all the state probabilties must be equal to 1. Let S_1 be the sum of the state probabilities for Chain 1 and S_2 be the sum of the state

probabilities in Chain 2. So we can write:

$$p_{0,0} + S_1 + S_2 = 1 \tag{1.21}$$

where:

$$S_1 = \sum_{n=1}^{\infty} (\frac{\lambda}{(\lambda + \mu_1)})^{n-1} p_{1,1} \tag{1.22}$$

$$S_2 = \sum_{n=2}^{\infty} r^{n-2}\ p_{2,2} \tag{1.23}$$

For S_1, let $m = n - 1$ and substituting for $p_{1,1}$

$$S_1 = \frac{\lambda + \mu_1}{\mu_1} C_{1,1} p_{2,2} \tag{1.24}$$

Similarly for S_2, let $m = n - 2$

$$S_2 = \frac{1}{1 - r} p_{2,2} \tag{1.25}$$

Summing to one we get:

$$p_{2,2} = \frac{1}{C_{0,0} + \frac{\lambda+\mu_1}{\mu_1} C_{1,1} + \frac{1}{1-r}} \tag{1.26}$$

Using the value of $p_{2,2}$ in Equation 1.38, we can find values for $p_{1,1}$ and $p_{0,0}$.

The average number of people in the queue can be expressed as:

$$L = \sum_{n=1}^{\infty} n (\frac{\lambda}{(\lambda + \mu_1)})^{n-1} p_{1,1} + \sum_{n=2}^{\infty} n r^{n-2} p_{2,2} \tag{1.27}$$

We can further get an exact formula for L, as follows in the below section.

1.6.2 Further Solving for L

From Equation 1.27, we first solve for the first term on the Left Hand Side of the equation and then second term.

$$\sum_{n=1}^{\infty} n(\frac{\lambda}{(\lambda+\mu_1)})^{n-1} p_{1,1} \tag{1.28}$$

Let $q = \frac{\lambda}{\lambda+\mu_1}$

$$= \sum_{n=1}^{\infty} n q^{n-1} p_{1,1} \tag{1.29}$$

Now, $n * q^{n-1} = \frac{d}{dq} q^n$

$$\begin{aligned} &= \sum_{n=1}^{\infty} \frac{d}{dq} q^n p_{1,1} \\ &= \frac{d}{dq} \sum_{n=0}^{\infty} q^n p_{1,1} \\ &= \frac{d}{dq}(\frac{1}{1-q}) p_{1,1} \quad (\text{substituting } \sum_{n=0}^{\infty} q^n = \frac{1}{1-q}) \\ \sum_{n=1}^{\infty} n(\frac{\lambda}{(\lambda+\mu_1)})^{n-1} p_{1,1} &= \frac{1}{(1-q)^2} p_{1,1} \end{aligned} \tag{1.30}$$

Now solving the second term on the Left Hand Side of the Equation 1.27.

$$\sum_{n=2}^{\infty} n r^{n-2} p_{2,2} = \sum_{n=2}^{\infty} (n-1) r^{n-2} p_{2,2} + \sum_{n=2}^{\infty} r^{n-2} p_{2,2} \tag{1.31}$$

In order, to present the solution of the Equation 1.31 in the simple form,

we will again solve the terms in the Left Hand Side one by one, starting with the second term on the Left Hand Side of the Equation 1.31.

$$\begin{aligned}
&\sum_{q=2}^{\infty} r^{n-2} p_{2,2} \\
&\qquad = \sum_{q=0}^{\infty} r^q p_{2,2} \quad (\text{ substituting } q = n-2 \text{ }) \\
&\qquad = \frac{1}{1-r} p_{2,2} \quad (\text{substituting } \sum_{q=0}^{\infty} r^q = \frac{1}{1-r})
\end{aligned} \tag{1.32}$$

Now, solving the first term on the Left Hand Side of the Equation 1.31:

$$\begin{aligned}
&\sum_{n=2}^{\infty} (n-1) r^{n-2} p_{2,2} \\
&\qquad = \sum_{n=2}^{\infty} \frac{d}{dr} r^{n-1} p_{2,2} \quad (\text{substituting } (n-1)r^{n-2} = \tfrac{d}{dr} r^{n-1}) \\
&\qquad = \frac{d}{dr} \sum_{n=1}^{\infty} r^{n-1} p_{2,2} \\
&\qquad = \frac{d}{dr} \sum_{n=0}^{\infty} r^q p_{2,2} \quad (\text{substituting } q = n-1 \text{ }) \\
&\qquad = \frac{d}{dr} * \frac{1}{1-r} p_{2,2} \quad (\text{substituting } \sum_{n=0}^{\infty} r^q = \frac{1}{1-r}) \\
&\qquad = (\frac{1}{1-r^2}) p_{2,2} \quad (\text{substituting } \tfrac{d}{dr} \tfrac{1}{1-r} = \tfrac{1}{1-r^2})
\end{aligned} \tag{1.33}$$

The results expressed in the Equations 1.32 and 1.33 showed that the

Equation 1.31 can be expressed as :

$$\begin{aligned}
\sum_{n=2}^{\infty} nr^{n-2}p_{2,2} &= ((\frac{1}{1-r}) + (\frac{1}{1-r^2})) * p_{2,2} \\
&= ((\frac{1}{1-r}) + (\frac{1}{(1-r)^2})) * p_{2,2} \\
&= ((\frac{1-r+1}{(1-r)^2}) * p_{2,2} \\
\sum_{n=2}^{\infty} nr^{n-2}p_{2,2} &= ((\frac{2-r}{(1-r)^2}) * p_{2,2} \qquad (1.34)
\end{aligned}$$

And the Equation 1.27 can be expressed as:

$$L = \frac{1}{(1-q)^2}p_{1,1} + ((\frac{2-r}{(1-r)^2}) * p_{2,2} \quad (\text{ From Equations } 1.30 \text{ and } 1.34 \text{ }) \qquad (1.35)$$

The average waiting time in the demand queue, $W_d = \frac{L}{\lambda_d}$

1.6.3 Results of Second Attempt

We have used values measured from the NMS simulation to investigate the analytical model presented. Simulation results for $p = 1$ and $d = 2$ were obtained for different demand miss rates. The simulation results are then compared with results from analytical model. This is shown in Figure 1.7.

The two results are quite close in value over a wide operational range. This indicates that the model will be useful in developing practical algorithms for high-performance network-based servers. It should be noted that the model is approximate as it depends on which state of the imaginary Chain is used to calculate r. This is because the solution for r varies slightly depending on which state is used. The best results were obtained using the state $2, 2$ which, in this case, is equal to K, the maximum batch size. Whether this holds for other values of K is being investigated.

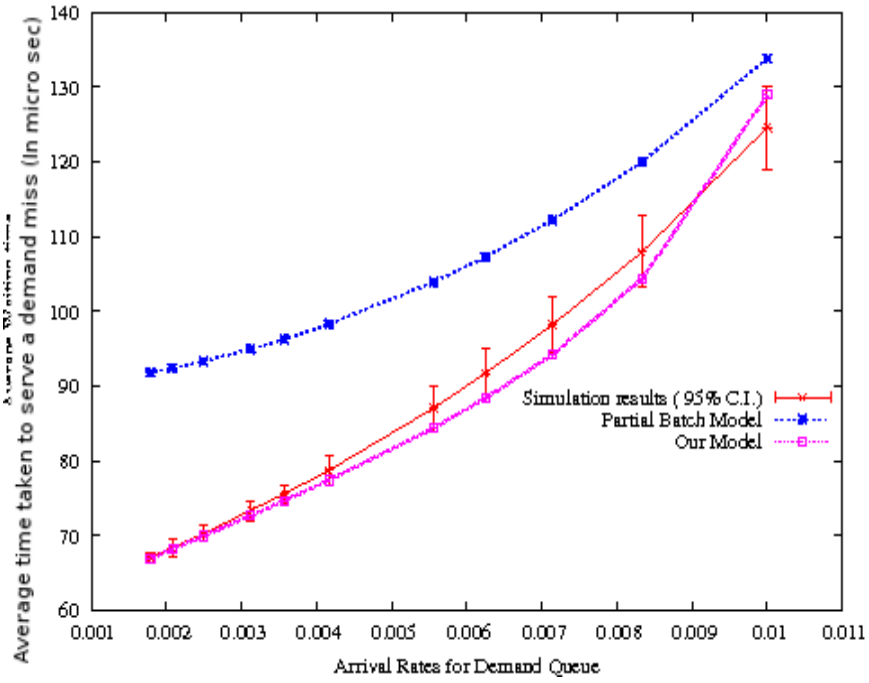

Figure 1.7: Average time to serve a demand miss (T_{D-NMS}) using Simulation, Partial Batch Model and Our approach.

1.7 Towards a General Solution

In this section, we seek to extent the method used for $K = 2$ to a general value of K. So a gate-limited model, where K is equal to maximum number of requests can be served at any moment, can be represented by a gate model of K chains. Furthermore, we can express the average number of requests in each chain, $L_{n,n}$, in terms of the first element of that chain, $P_{n,n}$ and thus we get the sum:

$$L = \sum_{n=1}^{K} *Ln, n = \sum_{n=1}^{K} \frac{n - (n-1) * r_n}{(1 - r_n)^2} P_{n,n} \tag{1.36}$$

For n ¡ K,

$$r_n = \frac{\lambda}{\lambda + \mu_n} \tag{1.37}$$

For $n = K$, we use the imaginary PBM technique to solve for r_K. Furthermore, we have

$$p_{K,K} = \frac{1}{C_{0,0} + \sum_{n=1}^{K=n-1} \frac{\lambda + \mu_n}{\mu_n} C_{n,n} + \frac{1}{1 - r_k}} \tag{1.38}$$

where $P_{n,n} = C_{n,n} P_{K,K}$. To get the general technique, we need to find the value of $C_{n,n}$ and we can do so using the equations for the state of $P_{n,n}$ in our model. This can be done by solving a series of simultaneous equations, using matrix techniques. This is not further persuade in this thesis but the chapter shows that it is possible to get a fairly accurate waiting time results based on this analytical model.

1.8 Conclusion

This chapter presented an analytical model which could be used to estimate the average time to serve a demand miss (T_{D-NMS}) for a given demand arrival rate and prefetch rate. Comparison of the results from the analytical model and simulation results showed that the results estimated by the analytical model are at 95% confidence level with confidence interval of $\pm 5\%$. Hence, the model can be used at run time to estimate the average time to serve the demand misses for a given scenario. Using the results from the analytical model and simulation, we would like to explore the space where QoS could be provided for streaming applications and demand misses.

Bibliography

Approximate analysis of transfer lines with unreliable machines and finite buffers, volume 34, Sep 1989. doi: 10.1109/9.35807.

A decomposition method for analyzing inhomogeneous assembly/disassembly systems, volume 93, 2000. Annals of Operation Research.

*Tr-88 Multiple Access Protocols**, 1983. Prentice Hall.

Performance evaluation of data communications systems., 1982.

Analysis and Application of Polling Models, London, UK, 2000. Springer-Verlag. ISBN 3-540-67193-5.

Queuing analysis of polling models, volume 20, New York, NY, USA, 1988. ACM. doi: http://doi.acm.org/10.1145/62058.62059.

Performance analysis of multi-server tandem queues with finite buffers and blocking., volume 27, Charleston, 2005. OR Spectrum.

B. Avi-Itzhak, W. L. Maxwell, and L. W. Miller. Queuing with Alternating Priorities. *OPERATIONS RESEARCH*, 13(2):306–318, 1965. doi: 10.1287/opre.13.2.306. URL `http://or.journal.informs.org/cgi/content/abstract/13/2/306`.

Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, second edition, 1992.

W. Bux. Local-area subnetworks: A performance comparison. *Communications, IEEE Transactions on*, 29(10):1465–1473, Oct 1981. ISSN 0090-6778.

Mack C. The efficiency of N machines uni-directionally patrolled by one operative when walking time is constant and repair times are variable. pages 173–8, 1957a.

Mack C. The efficiency of N machines uni-directionally patrolled by one operative when walking time is constant and repair times are constants. *J Roy Stat Soc Ser B*, pages 166–172, 1957b.

Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li. A study of integrated prefetching and caching strategies. In *SIGMETRICS '95/PERFORMANCE '95*, pages 188–197, New York, NY, USA, 1995. ACM Press. ISBN 0-89791-695-6. doi: http://doi.acm.org/10.1145/223587.223608.

W. W. Chu and A. G. Konheim. On the analysis and modeling of a class of computer-communications system. *IEEE Trans. Commun.*, vol. COM-20 (2):645–660, June 1972.

Donald Gross and Carl M. Harris. *Fundamentals of Queueing Theory (Wiley Series in Probability and Statistics)*. Wiley-Interscience, February 1998. ISBN 0471170836. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0471170836`.

J. F. Hayes and D. N. Sherman. A Study of Data Multiplexing Techniques and Delay Performance. *Bell Syst. Tech. J.*, 51(9):1983–2011, November 1971. doi: 10.1287/opre.13.2.306.

L. Kleinrock. *Queueing Systems, Volume 2: Computer Applications*. Wiley, 1976.

A. Kobayashi, H.; Konheim. Queueing Models for Computer Communications System Analysis . volume 25, pages 2 – 29, Jan 1977.

A. G. KONHEIM. Mathematical models for computer data communication. In Case Studies in Mathematical Modeling. pages 256–334, 1980.

M. A. LEISOWITZ. Mathematical models for computer data communication. In Case Studies in Mathematical Modeling. pages 256–334, 1980.

PENNY, B. K., ANO BAGHOADI, A. A. Survey of computer communications loop networks., journal = Computer Communications. 2(4):165–180 and 224–241, 1979.

KAYE A. R. and RICHARDSON T. G. A performance criterion and traffic analysis for polling systems. *INFOR, Can. J. Oper. Res. Inf. Process.*, 11 (2):93–112.

STIDHAM S. Optimal control of a signalized intersection. Technical report, Cornell Univ, Ithaca, NY, 1969.

H Takagi. Queueing analysis of polling models: An update, Stochastic Analysis of Computer and Communication Systems. pages 267–318, 1990.

D Thakker, G Mapp, and O Gemikonakli. Modelling mixed access-patterns in Network-Based Systems. March 2009.

M van Vuuren and E Winands. Iteractive approximation of k-limited polling systems. May 2006.

Adan IJBF van Vuuren M. Performance analysis of assembly systems. pages 89–100, Charleston, 2006. Proceedings of the Markov anniversary meeting.

Yung-Terng Wang and Morris R. J. T. Load sharing in distributed systems. *IEEE Trans. Comput.*, 34(3):204–217, 1985. ISSN 0018-9340. doi: http://dx.doi.org/10.1109/TC.1985.1676564.