**Far East Journal of Electronics and Communications**

Published Online: September 2014
Available online at http://pphmj.com/journals/fjec.htm
Volume 13, Number 1, 2014, Pages 41-58

# QUEUING SYSTEM FOR DYNAMICALLY RECONFIGURABLE NETWORK PROCESSOR

**A. Satheesh, D. Kumar, P. Dharmalingam and T. K. S. Lakshmipriya**

Department of Computer Science and Engineering
Periyar Maniammai University
Vallam-613403, Thanjavur, Tamil Nadu, India
e-mail: asatheesh@pmu.edu

Department of Electronics and Communication Engineering
Periyar Maniammai University
Vallam-613403, Thanjavur, Tamil Nadu, India
e-mail: kumar_durai@pmu.edu

Department of Mathematics
Periyar Maniammai University
Vallam-613403, Thanjavur, Tamil Nadu, India
e-mail: dharmap4@gmail.com

Department of Printing Technology
Avinashilingam University
Coimbatore, Tamil Nadu, India
e-mail: tkslp.dr@gmail.com

## Abstract

Network processor (NP) is a programmable processor used for packet processing in network applications. Today, many vendors are offering different models of NPs for various applications. The major difference

Received: May 3, 2014; Revised: July 7, 2014; Accepted: July 21, 2014
Keywords and phrases: network processor, packet processing, microengine, dynamic reconfiguration, M/M/c queue, IXP2400 NP.

amongst them is the capabilities of NPs in terms of the number of microengines (MEs). The main objective of this work is to optimize the usage of MEs during network traffic fluctuation by dynamic reconfiguration. In this paper, packet queue length, which is a function of traffic fluctuation, is used to determine the number of MEs in use at a given point of time. Turning the MEs ON/OFF is done dynamically, when the packet queue length increases/decreases by *k*. This paper also proposes a new queuing system for this scenario, i.e., dynamically reconfigurable NPs during fluctuating traffic. Intel IXP2400 NP has been adopted for validating the queuing model for different traffic fluctuations. The simulation results closely matched with the analytical results.

## 1. Introduction

Multicore programmable network processors (NPs) had been being widely used in high-speed network routers, switches and various other network components [3]. In the fast-growing internet traffic, the present routers enabled new network protocols and are also flexible for different network traffic. However, the designers are required to pay utmost attention to both performance and flexibility of the router. The latest NP architecture is proficient enough to achieve such a high performance with equal flexibility through multiple microengines (MEs), which were processed either independently or in shared parallel operations. The programmable NPs have facilitated new applications of high performance rates at lower costs [4]. The major challenges for the NP designers were to design a low-power NP. In the NP architecture, the major power hungry component is the ME. In the previous work [1], the present authors had modeled a framework for self-configuring IXP2400 NP. The objective of that work was to transform the IXP2400 NP into a self-configurable environment. Self-configuration was the process of automatically altering the functionality of network system or network element. This automatic configuration described the way sophisticated networks could re-adjust themselves in the event of network traffic or service requirement, enabling the network processor to continue operation. The IXP2400 NP consists of eight MEs for packet processing.

This work involved the optimization of the usage of MEs based on network fluctuations at reduced energy consumption by NP. The control plane constantly monitored the traffic arrival rate in the system and based on the workload, the MEs were switched on/off without any packet loss. In this paper, a new queuing model developed for the above previous work [1] of dynamically reconfigurable NP is also discussed. We have used Intel IXP2400 NP architecture as stated earlier. The internal architecture of MEs is shown in Figure 1.

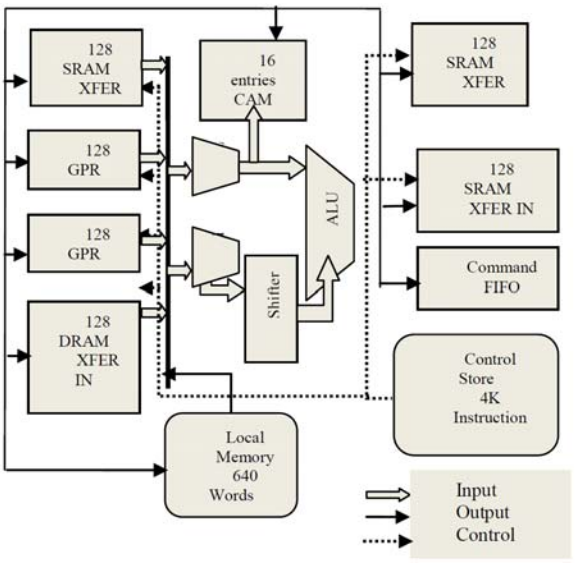


**Figure 1.** Architecture of IXP2400 NP microengine.

The Arithmetic Logic Unit (ALU), a shifter and numerous storage units are combined together to constitute the data path. The executable instructions are kept in control store. The Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM) utilize XFER (Transfer Registers) for transaction. For high-speed data storage, GPRs (General Purpose Registers) and XFERs along with local memory could be employed. Two adjacent MEs directly communicate in the data path with the help of 128 next-neighbor registers (not shown in Figure 1). Since the GPRs and the next-neighbor register are structurally same, they are treated alike in the simulation [6].

In general, queuing theory considers a fixed number of servers [7, 8] and the performance metrics are queue length and waiting time. However, considering the fact that queuing system wherein the number of severs might change depending on the queue length, this paper assumes a queuing model with single server, namely, the ME. A new queuing model has been proposed, which allows optimized MEs usage by deploying additional ME only when required. This has resulted in a minimum number of MEs, sufficient to offer the required performance with high probability of optimizing the resource provisioning level.

As the $k$ packets arrive, the queue size increases and reaches a certain point. If $k+1$th packet arrives, then an additional ME is deployed. If more packets arrive, and the queue size increases to $2k$, then the third ME is deployed and so on. Such a queuing model is shown in Figure 2, which is the block diagram of queuing model for IXP2400 NP.

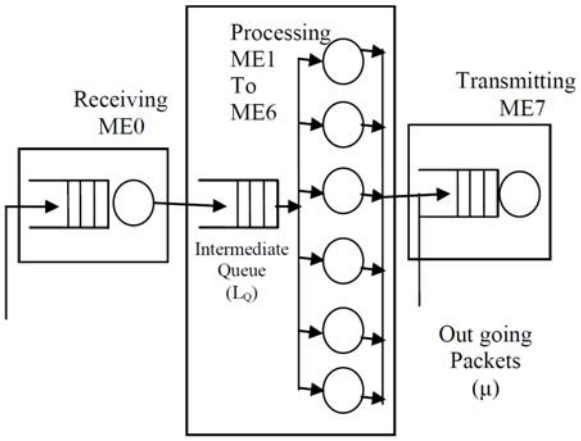


**Figure 2.** Queuing model for dynamic adaptive load sharing network processor.

M/M/c queuing model has been applied for calculating throughput (the number of packets processed per time unit), average response time (average time spent by the packet inside NP), and resource utilization (percentage of time, the MEs were busy processing the packets).

Intel IXP2400 network processor could operate in an environment with maximum network traffic of OC-48/4.1Gbps data rates [15]. Such a system is capable of transferring 400 packets in 3.278ms. Typical service rate is 54ns when four threads in each of the eight MEs are operational [2]. The MEs run at 600MHz, and execute each of the pipeline stages of the IP forwarding application. The MEs take 176 cycles for processing a single packet with the minimum size of 46 bytes [15]. Processing one packet is completed in 0.176 milliseconds.

In this paper, as a first level study, the arrival and departure rates ($\lambda$ and $\mu$) followed Poisson distribution. However, the packet processing revealed exponential distribution patterns. The queue service discipline was first-come-first served.

It was assumed that $n \geq c$, where $n$ and $c$ were the number of packets received and the number of servers used by the system.

The formula for identifying the traffic intensity, $\rho$ was

$$\rho = \frac{\lambda}{c\mu}. \tag{1}$$

When the system is with low level traffic (queue length was $k$), the queuing model is M/M/1. Then the traffic rate is slowly increased beyond the threshold values (queue length to $2k$), so that the system automatically switched on one additional ME for packet processing. Consequently, the system is in M/M/2 state. In the third case, the traffic rate was increased beyond the moderate level, then the system immediately switched on all the six MEs in it for packet processing, moving to M/M/6 state.

This paper is organized as follows:

- Section 2 - overviews of related works.
- Section 3 - a new queuing model with a variable number of servers for dynamically reconfigurable NP.
- Section 4 - the simulation results of dynamically reconfigurable IXP2400 NP.
- Finally, Section 5 gives the conclusion of the present study.

## 2. Related Works

The various performance aspects of NP systems were investigated by many studies. These investigations mainly focused on analytical model, real time experiments or simulations.

Thiele et al. in [9] described an analytic model of the NPs design space. However, the model was based on a high level of abstraction, where the goal was to quickly identify the interesting architectures, which might be subjected to a more detailed evaluation using simulation. The final output was three architectures, representing cost versus performance tradeoffs.

Spalink et al. in [10] presented a model for building a software-based router using IXP1200 NP. Those works focused mainly on the queuing disciplines, which included port mapping, packet scheduling and queue contention. Many studies had been carried out with reference prototype architecture and IXP1200 NP. At the outset, those studies were focused on the queuing discipline of IXP1200 NP and discussed further technical details about where and how to queue packets. However, a study on dimensioning the queues in an NP system based on real world and synthetically generated traces, was not revealed in any of the literature surveyed.

Solovjev et al. [11, 12] in their paper had worked in the same line as that of the present paper. They had considered birth-death process with a system of parameters $\{\lambda_n\}$ and $\{\mu_n\}$ arrival intensity and service rate, respectively. The arrival rate was fixed with arbitrary service rate. In [12], Mazalov and Gurtov chose the example of Dallas-Fort Worth International Airport data. In that work, if the queue size exceeded every $k$ customer, then one more additional counter could be opened. The number of servers was infinite depending upon the customer arrival rate and the service counter could be opened at $n$ numbers as required.

LakshmiPriya and Parthasarathi [13] have proposed an architecture for a grid of network processors that could offer in-network services using active network technology. The concept had been demonstrated for the media streaming application.

In continuation with their work on exploring the capabilities of the network processor, LakshmiPriya and Parthasarathi [14] have proposed a network processor framework for application aware networks, which incorporated concepts such as self-configuration and active network technology in NPs, in order to form a coordinated framework in the network.

## 3. Queuing Model for Dynamic Reconfigurable NP

As mentioned earlier, the arrival and departure rates (λ and μ) followed Poisson distribution. However, the packet processing revealed exponential distribution patterns. The queue service discipline was first-come-first served. It was assumed that $n \geq c$, where $n$ and $c$ were the number of packets received and the number of servers used by the system. It was again assumed that $c$ is a random variable that could take natural values only.

It was assumed that each microengine (ME) processed maximum $k$ packets (queue length $k$). If the queue length increased to more than $k$, then the system would deploy the code to another ME would switch on. That would continue for the queue length of every $2k$, $3k$, etc. The common queue length at a time $t$ is denoted as $x(t)$. If the system as $c$ active MEs, then the following conditions hold at time $t$:

$k(c-1) < x(t) \leq kc$ for some natural $c$. Let $P_i(t)$, $i = 0, 1, \ldots$ be the probability that $i$ packets were in queue at a given time $t$. Kolmogorov equations for this model were written as follows:

$$P'_{kc}(t) = -(\lambda + c\mu)P_{kc}(t) + \lambda P_{kc-1}(t) + (c+1)\mu P_{kc+1}(t),$$

$$P'_{kc+j}(t) = -(\lambda + (c+1)\mu)P_{kc+j-1}(t) + (c+1)\mu P_{kc+j+1}(t)$$

with $P_{-1}(t) = 0$ for $c = 0, 1, \ldots, j = 1, \ldots, k-1$.

The above set of equations may be regarded as a particular case of the system of equations related to birth-death process:

$$P'_l(t) = \lambda_{l-1}P_{l-1}(t) - (\lambda_l + \mu_l)P_l(t) + \mu_{l+1}P_{l+1}(t).$$

For $l = 0, 1, ...,$ where $\lambda_{l-1} = \mu_0 = 0,\ \lambda_l = \lambda,\ l = 1, 2, ..., \mu_{kc+j} = (c+1)\mu.$

For $c = 0, 1, ..., j = 1, ..., k-1$ and $\mu_{kc} = c\mu$ for $c = 1, 2, ....$

The necessary and sufficient conditions for such a process to have steady state were

$$\sum_{l=0}^{\infty} \theta_l < \infty, \quad \sum_{l=0}^{\infty} \frac{1}{\lambda_l \theta_l} = \infty, \tag{2}$$

where

$$\theta_0 = 1, \quad \theta_l = \frac{\lambda_0 \cdots \lambda_{l-1}}{\mu_1 \cdots \mu_l}, \ \ l \geq 1.$$

Here, the values $\theta_l$ took the form

$$\theta_{kc+i} = \frac{\rho^{kc+i}}{(c!)^k (c+1)^i}, \ \ i = 1, ..., k; \ \ c = 0, ...,$$

where $\rho = \frac{\lambda}{\mu}$ was the system load. It was easy to see that the condition (2) for the existence of steady-state holds, since the series

$$\sum_{c=0}^{\infty} \sum_{i=1}^{k} \theta_{kc+i} = \sum_{c=0}^{\infty} \left( \frac{\rho^c}{c!} \right)^k \sum_{i=1}^{k} \left( \frac{\rho}{c+1} \right)^i$$

converges and

$$\frac{1}{\lambda} \sum_{c=0}^{\infty} \sum_{i=1}^{k} \frac{1}{\theta_{kc+i}} = \frac{1}{\lambda} \sum_{c=0}^{\infty} \left( \frac{c!}{\rho^c} \right)^k \sum_{i=1}^{k} \left( \frac{n+1}{\rho} \right)^i = \infty,$$

$$n! = \left( \frac{n}{e} \right)^n \sqrt{2\pi c} \left( 1 + O\left( \frac{1}{c} \right) \right).$$

It was obtained that a system of equations for the state $(P_i(t) = P_i = \text{const})$ contains several groups of similar recurrent equations.

The first group of $k$ equations:

$$P_1 = \rho P_0,\ P_{i+1} = (1+\rho)P_i - \rho P_{i-1},\ i = 1, ..., k-1.$$

The second set of equations:

$$P_{k+1} = \frac{1+\rho}{2}P_k - \frac{\rho}{2}P_{k-1},\ P_{k+i+1} = \left(1+\frac{\rho}{2}\right)P_{k+i} - \frac{\rho}{2}P_{k+i-1},\ i = 1, ..., k-1.$$

The general formula for $P_i$:

$$P_{kc+i} = \frac{\rho^{kc+i}}{(c!)^k (c+1)^i} P_0,\quad i = 1, ..., k,\quad c = 0, 1, .... \tag{3}$$

It was found that $P_0$ from the condition

$$\sum_{i=0}^{\infty} P_i = 1.$$

From (3), it was derived:

$$P_0 + \sum_{c=0}^{\infty}\sum_{i=1}^{k} P_{kc+i} = 1$$

or

$$P_0\left(1 + \rho\sum_{c=0}^{\infty}\sum_{i=1}^{k}\frac{\rho^{kc+i}}{(c!)^k (c+1)^i}\right) = 1.$$

On simplifying,

$$P_0 = \left(1 + \rho\sum_{j=1}^{\infty}\frac{\rho^{(j-1)k} j^k - \rho^k}{(j!)^k\, j - \rho}\right)^{-1}.$$

Now we can calculate all the $P_j$s and consequently all the main characteristics of the queuing system.

The average number of MEs in the system was an important factor

$$E\{c\} = \sum_{c=0}^{\infty} \sum_{i=1}^{k} (c+1) P_{kc+i}.$$

On simplifying,

$$\begin{aligned} E\{C\} &= \sum_{c=0}^{\infty} (c+1) \sum_{i=1}^{k} P_{kc+i} \\ &= \sum_{c=0}^{\infty} \sum_{i=1}^{k} \frac{\rho^{kc+i}}{(c!)^k (c+1)^{i-1}} P_0 \\ &= \rho \sum_{c=0}^{\infty} \sum_{i=1}^{k} \frac{\rho^{kc+i-1}}{(c!)^k (c+1)^{i-1}} P_0 \\ &= \rho \left[ \sum_{c=0}^{\infty} \sum_{i=0}^{k-1} P_{kc+i} \right] \\ &= \rho \left[ \sum_{c=0}^{\infty} \sum_{i=1}^{k} P_{kc+i} + \sum_{c=0}^{\infty} P_{kc} - \sum_{c=0}^{\infty} P_{kc+k} \right] \\ &= \rho \left( 1 - P_0 - \sum_{c=0}^{\infty} P_{k(c+1)} + \sum_{c=0}^{\infty} P_{kc} \right) = \rho. \end{aligned}$$

From the above, it could be easily established that the average number of MEs given $k$, was equal to the load of the system. However, it was found that the variance of $c$ is dependent on $k$. First, it was concluded that

$$\begin{aligned} E\{C^2\} &= \sum_{c=0}^{\infty} (c+1)^2 \sum_{i=1}^{k} P_{kc+i} \\ &= \sum_{c=0}^{\infty} (c+1)^2 \left[ P_{kc+1} + \sum_{i=2}^{k} P_{kc+i} \right] P_0 \end{aligned}$$

$$= \sum_{c=0}^{\infty} (c+1)^2 \left[ \frac{\rho^{kc+1}}{(c!)^k (c+1)^i} + \sum_{i=2}^{k} \frac{\rho^{kc+i}}{(c!)^k (c+1)^i} \right] P_0$$

$$= \sum_{c=0}^{\infty} \frac{(c+1)^2 \rho^{kc+1} P_0}{(n!)^k (c+1)^i} + \rho^2 \sum_{c=0}^{\infty} \sum_{i=2}^{k} \frac{\rho^{kc+i-2}}{(c!)^k (c+1)^{i-2}} P_0$$

$$= \sum_{c=0}^{\infty} \frac{\rho^{kc+1}(c+1)}{(c!)^k} P_0 + \rho^2 \sum_{c=0}^{\infty} \sum_{i=0}^{k-2} P_{kc+i}$$

$$= \sum_{c=0}^{\infty} \frac{\rho^{kc+1}(c+1)}{(c!)^k} P_0 + \rho^2 \left[ 1 - \sum_{c=0}^{\infty} P_{k(c+1)} - 1 \right]$$

$$= \rho^2 + \sum_{c=0}^{\infty} \left[ \frac{\rho^{kc+1}(c+1)}{(c!)^k} - \rho^2 \frac{\rho^{k(c+1)-1}}{(c!)^k (c+1)^{k-1}} \right] P_0$$

$$= \rho^2 + \left( \rho + \sum_{c=1}^{\infty} \frac{\rho^{kc+1}(c+1)}{(c!)^k} - \sum_{c=0}^{\infty} \frac{\rho^{k(c+1)+1}}{(c!)^k (c+1)^{k-1}} \right) P_0$$

$$= \rho^2 + \rho P_0 + \left( \sum_{c=1}^{\infty} \rho^{kc+1} \frac{(c+1)}{(c!)^k} - \sum_{c=0}^{\infty} \frac{\rho^{k(c+1)}(c+1)}{(c!)^k (c+1)^k} \right) P_0$$

$$= \rho^2 + \rho P_0 + \sum_{c=1}^{\infty} \rho^{kc+1} P_0 \left( \frac{c+1-c}{(c!)^k} \right)$$

$$= \rho^2 + \rho P_0 + P_0 \sum_{c=1}^{\infty} \frac{\rho^{kc+1}}{(c!)^k}$$

$$= \rho^2 + \rho P_0 + \left[ \sum_{c=1}^{\infty} \frac{\rho^{kc+1}(c+1)}{(c!)^k} - \sum_{c=1}^{\infty} \frac{\rho^{kc+1} \cdot c}{(c!)^k} \right] P_0$$

$$= \rho^2 + \rho P_0 + \sum_{c=1}^{\infty} \frac{\rho^{kc+1}}{(c!)^k} P_0$$

$$= \rho^2 + \sum_{c=0}^{\infty} \frac{\rho^{kc+1}}{(c!)^k} P_0.$$

The variances of the number of active MEs were found to be

$$Var\{c\} = E\{c^2\} - (E\{c\})^2,$$

$$Var\{c\} = P_0 \rho \sum_{c=0}^{\infty} \left( \frac{\rho^c}{c!} \right)^k.$$

The relationship showed that the variance was a decreasing function of $c$. It was an important characteristic in practice, since it showed the variance of the necessary number of MEs around the mean value. The larger variance indicated the requirement of additional MEs for packet processing at a point of time.

The other important characteristics such as average queue length, average waiting time in the queue were useful for deciding the optimum $c$.

As the average number of packets in the system is less than $k\rho$, then

$$E\{q\} = \sum_{c=0}^{\infty} \sum_{i=1}^{k} (kc + i) P_{kc+i} = k\rho - \sum_{c=0}^{\infty} (k - i) P_{kc+i} \leq k\rho.$$

It was required to find an upper bound on the mean waiting time. It was important that the number of active MEs could change depending on the current queue length. Then, during the packet processing, the number of active MEs would gradually decrease and then expected waiting time for the packet would be

$$\frac{i}{(c+1)\mu} + \frac{k}{c\mu} + \frac{k}{(c-1)\mu} + \cdots + \frac{k}{\mu} = \frac{i}{(c+1)\mu} + \frac{k}{\mu}\left(1 + \frac{1}{2} + \cdots + \frac{1}{c}\right).$$

The maximum waiting time for the packets in the queue for processing was given by

$$E\{t\} = \sum_{c=0}^{\infty} \sum_{i=1}^{k} \left( \frac{i}{(c+1)\mu} + \frac{k}{\mu} s_c \right) P_{kc+i},$$

which was an upper bound for waiting time, where $s_c = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$.

## 4. Implementation

Earlier the same authors of this paper [1] had implemented the dynamic reconfiguration technique in real time environment. They had used ENP2611 chipset for Intel IXP2400 NP. Intel IXP2400 Developers Workbench a simulation tool, which helped to execute the ME code, and obtained the performance measures for the application program. Microcode assembly language for ME implementation and embedded C programming for X-scale implementation had been used to implement the system. The packets arrival was assumed to follow Poisson distribution.

Figure 3 shows that when the traffic got increased from low to medium, the minimum resource would not be able to manage the workload, so the queue length got increased. Once again, heavy traffic was injected into the NP when maximum resource is dynamically provisioned to handle the network traffic. Thus, the MEs are allocated based on the network traffic, the queue did not overflow, and resources were well-utilized. The arrival rate of the system was 400 packets per milliseconds, the queue size was 10 and it used three MEs (one ME for ingress, one ME for egress and one ME for packet processing). The queue length was increased from 10 to 20 when an additional ME was deployed for packet processing. At another instant, when the arrival rate was increased from 400 to 500 packets per milliseconds, the queue length increased by more than 20, and hence the system deployed all eight MEs.

Similarly, the simulation parameters were set as, 250MHz the MEs clock frequency, MEs took 4ns to execute single instruction and each packet was executed in 25 instructions inside an ME. These parameters were based on IXP2400 NP architecture. Therefore, each packet needed 100ns for instruction execution. The values of 10, 60 and 100 were the access latencies for Static Random Access Memory (SRAM) and Dynamic Random Access Memory (DRAM). For every 100 packets, the averages of interarrival time were calculated. It varied from 400ns to 1400ns. The traffic intensity was periodically changing high and low. For the simulation, it was chosen that the parameters $\lambda = 400$ and $\mu = 54$.

A queuing simulation using C language was developed. The settings used were: arrival rate of 400packets/ms, the number of servers was eight and the inputs were, the arrival rate ($\lambda$) was 400ms, the service rate ($\mu$) was 54ns. The NP architecture was IXP2400. So the default number of MEs was 8. The $k$ values varied from 1 to 30. Figures 4-6 show the simulation results for the queuing system using dynamically reconfigurable queue simulation. This simulation was developed by using C language. Figure 4 shows the necessary number of MEs versus the parameter $k$ (maximum queue length per ME). Figure 5 shows the average waiting time in milliseconds versus the parameter $k$. Figure 6 shows the reconfiguration time in milliseconds versus the parameter $k$. As seen in the figures, if the parameter $k$ was fixed as 18 (the maximum queue length per ME), then the IXP2400 NP used only one number of active MEs. Then the average waiting time in the queue for a packet would be 0.14ms and the reconfiguration time would be 0.11ms. This result satisfied the following two conditions:

(i) The average number of active MEs would be equal to traffic intensity ($\rho$). Here, the result of traffic intensity ($\rho$) was 0.93 and the output of average number of active MEs was 0.93.

(ii) The average number of packets in the system was less than $k\rho$. In the output, the average number of packets in the system was 7. This value was less than $k\rho$.

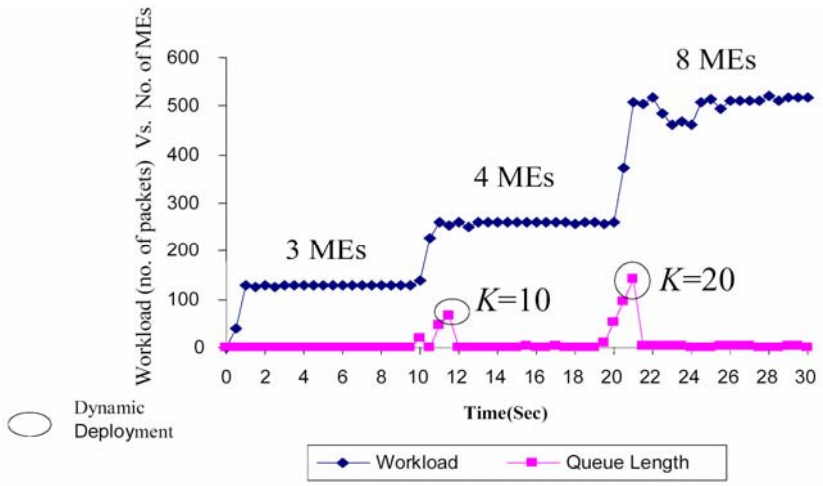


**Figure 3.** Workload, queue length and the number of MEs used in dynamic reconfiguration.

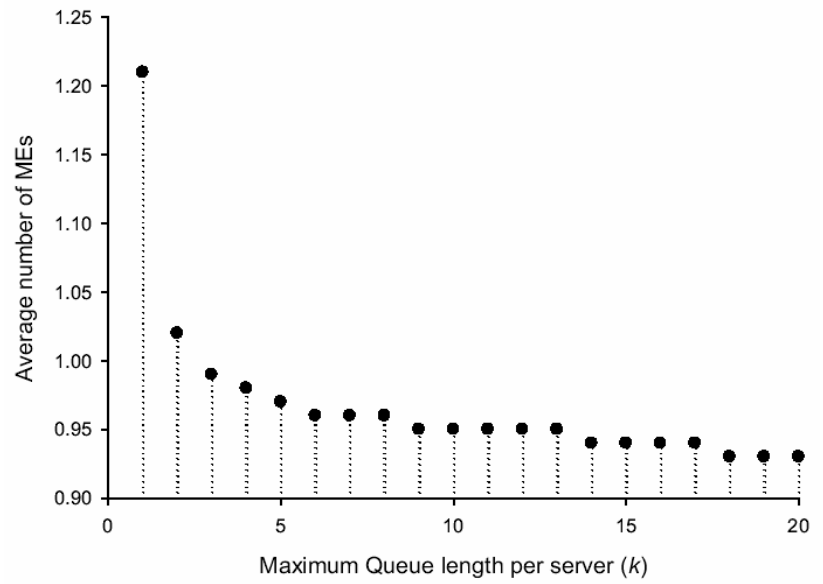


**Figure 4.** The necessary number of MEs versus the parameter $k$ (maximum queue length per server).

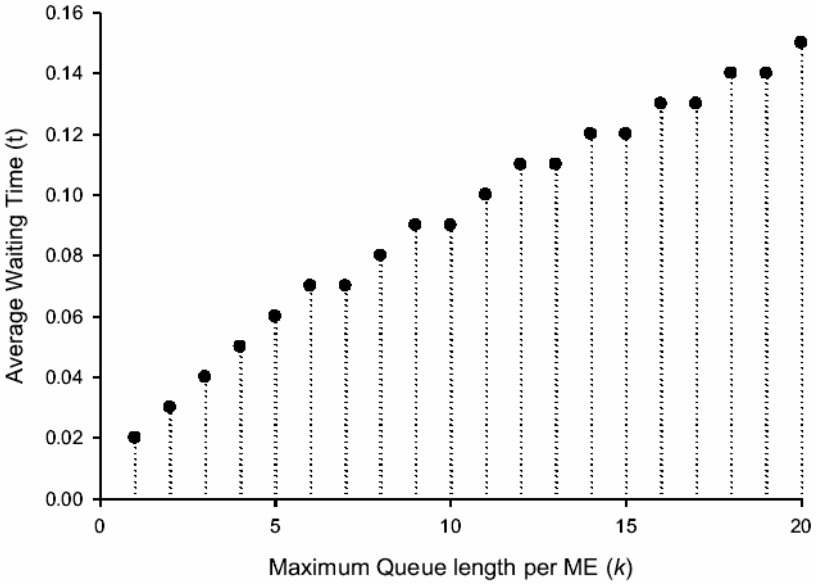


**Figure 5.** Average waiting time in milliseconds (ms) versus the parameter *k* (maximum queue length per server).

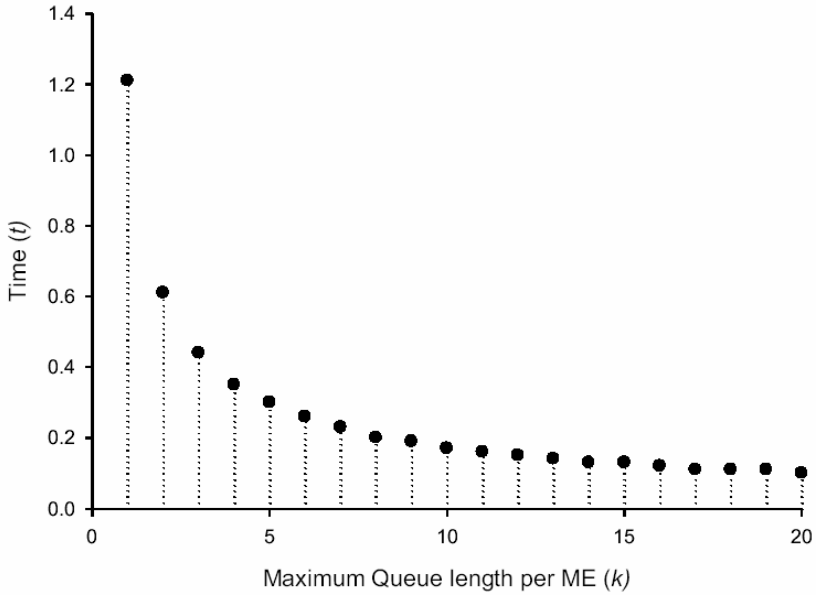


**Figure 6.** Reconfiguration time in milliseconds (ms) versus the parameter *k* (maximum queue length per server).

## 5. Conclusion

In the proposed dynamically reconfigurable queuing system, the number of MEs would dynamically adjust based on the queue length. This paper analyzed the dynamically reconfigurable queuing model in IXP2400NP using *Kolmogorov* differential equations and obtained upper bounds on the number of MEs, queue length, average waiting time and reconfiguration time.

This proposed system could model the dynamic reconfiguration queuing system used in IXP2400 NP. The simulation model showed a close match with analytic results using real world data. This model permitted energy saving by optimizing the number of MEs while providing an acceptable waiting time for the packet processing with high probability.

In this paper, a particular queuing mechanism in IXP2400 NP has been considered. It allowed to simplify the problem and formulate it using just one parameter $k$, the queue segment length. This had led to find a solution analytically in closed form for the application of result in practice. The work presented in this paper is a first level model, which has to be subsequently refined by including more parameters.

## References

[1] A. Satheesh, S. Krishnaveni and S. Ponkarthick, Self-configurable environment for the Intel IXP2400 network processor, International Journal of Computers and Applications 31(4) (2009), 268-273.

[2] Jing Fu, Olof Hagsand and Gunnar Karlsson, Queueing behavior and packet delays in network processor systems, Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007, MASCOTS'07, 15th International Symposium on October 24-26, 2007, pp. 217-224.

[3] Charlie Wiseman, Jonathan Turner, Michela Becchi, Patrick Crowley, John Dehart, Mart Haitjema, Shakir James, Fred Kuhns, Jing Lu, Jyoti Parwatikar, Ritun Patney, Michael Wilson, Ken Wong and David Zar, A remotely accessible network processor-based router for network experimentation, ANCS'08, November 6-7, 2008, pp. 20-29.

[4] Matthias Gries, Chidamber Kulkarni, Christian Sauer and Kurt Keutzer, Comparing analytical modeling with simulation for network processors: a case study, Presented at the Proceedings of the Conference on 2003 Design, Automation and Test in Europe Conference and Exposition, Munich, Germany, March 3-7, 2003, pp. 20256-20261.

[5] Christoforos Kachris and Stamatis Vassiliadis, A dynamically reconfigurable queue scheduler, Proceedings of the 2006 International Conference on Field Programmable Logic and Applications (FPL), Madrid, Spain, August 28-30, 2006, pp. 1-4.

[6] Intel, Intel ixp2xxx network processors hardware reference manual, Intel Corporation, 2005.

[7] R. Hassin and M. Haviv, To Queue or Not to Queue: Equilibrium Behaviour in Queueing Systems, Springer, 2002.

[8] L. Kleinrock, Queueing Systems, Volume I: Theory, Wiley Interscience, 1975.

[9] L. Thiele, S. Chakraborty and M. Gries, Design space exploration of network processor architectures, Proc. of 1st Workshop on Network Processors, held in conjunction with the 8th International Symposium on High-performance Computer Architecture, Cambridge, MA, Vol. 1, 2002, pp. 30-41.

[10] T. Spalink, S. Karlin, L. Peterson and Y. Gottlieb, Building a robust software-based router using network processors, Proc. of the 18th ACM Symposium on Operating Systems Principles (SODP), Banff, Alberta, Canada, October 2001, pp. 216-229.

[11] A. D. Solovjev, A problem of optimal queuing, Technical Cybernetics 5 (1970), 40-49.

[12] Vladimir V. Mazalov and Andrei Gurtov, Queuing system with on-demand number of servers, Mathematica Applicanda 40(2) (2012), 1-12.

[13] T. K. S. LakshmiPriya and Ranjani Parthasarathi, Architecture for an active network infrastructure grid - the iSEGrid, The International Workshop on Active Networks - IWAN05, France, LNCS 4388, 2009, pp. 38-52.

[14] T. K. S. LakshmiPriya and Ranjani Parthasarathi, Coordinated support for application-aware networks, Special issue on new technologies, mobility and security, Ubiquitous Computing and Communication Journal Volume: NTMS – Special Issue, 2008.

[15] Intel IXP2400 Network Processor: Flexible, high performance solution for access and edge applications, 2003. Available from: http://www.intel.com/design/network/papers/ixp2400.htm.