Research Paper Mathematics

System Performance Measures Analysis for Heterogeneous Computing Network using Fuzzy Queue

G.Geetharamani	Department of Mathematics, Anna University, BIT Campus, Tiruchirappalli-620024, Tami Nadu, India.
B.Palpandi	Department of Mathematics, Anna University, BIT Campus, Tiruchirappalli-620024, Tami Nadu, India.
J.Arun Pandian	PG Scholar, Department of mobile and Pervasive Computing, Anna University, BIT Campus, Tiruchirappalli-620024, Tamil Nadu, India.

ABSTRACT

In this paper system performance measures analysis for heterogeneous computing network has been made by using fuzzy queuing system. Heterogeneous computing network consist of more than one kind of servers with different service rates. It is a multi-core system that not only provides performance by just adding cores, but also gives specialized processing capabilities to handle particular tasks. In real time most of the system parameters of the queuing models are vague. To handle vagueness, data arrival rates and different service rates of the servers are taken as trapezoidal fuzzy numbers. The basic idea is to convert the fuzzy inter arrival rate and service rate into crisp values by applying robust ranking technique. After that, numerical analysis has been made for performance measures of two server network with different service efficiencies.

KEYWORDS Heterogeneous computing network; Fuzzy sets (normal and convex); Trapezoidal fuzzy number; robust ranking technique.

1. INTRODUCTION

A heterogeneous computing network is a system in which there are many servers with different service efficiencies .The computers do not guarantee similar results for operations on floating point numbers. Such systems are mostly found in server-client computers. Heterogeneous System Architecture (HSA) utilizes multiple processor types (typically CPUs and GPUs), usually on the same silicon die, to give the best of both areas: GPU processing, apart from its well-known 3D graphics rendering capabilities, can also perform mathematically intensive computations on very large data sets, while CPUs can run the operating system and perform traditional serial tasks.

By the end of 2010, nearly all new desktop computers had multi-core processors, with dual-core and even quad core processors entering the mainstream of affordable computing. Still, multi-core processing posed some challenges of its own. Meanwhile, in the multi-core era there were some interesting developments in Graphics processing units (GPUs), which were growing in sophistication and complexity, spurred on by advances in semiconductor technology. GPUs have vector processing capabilities that enable them to perform parallel operations on very large sets of data – and to do it at much lower power consumption relative to the serial processing of similar data sets on CPUs. Thus GPUs have much capabilities such as incredibly realistic, multiple display stereoscopic gaming. Even though their value was initially derived from the ability to improve 3D graphics performance by offloading graphics from the CPU, they became increasingly attractive for more general purposes, such as addressing data parallel programming tasks.

The early efforts to leverage GPUs for general purpose computing coincided with a notable shift in consumer culture. There was a dramatic increase in the availability and quality of digital content, coupled with an increasing consumer appetite for rich visual experiences like video playback and viewing content in HD. At the same time, the emergence of mainstream operating system support for advanced multitasking began to require processing efficiency of an entirely new magnitude.

In real life queuing problems, heterogeneous-multi-server services (i.e., each of several servers provides service at a distinct rate) arise in many situations such as supermarket checkout lines, ticket offices, bank teller windows, and call centres .Though [4] have Analyzed infinite capacity M/M/2 and M/M/3 queuing systems in which the number of servers changes depending on the queue length, he only considered the servers with the equal service rates.[3] studied queue-dependent servers in an infinite capacity M/M/2 queuing system with two different types of service rates: in Type 1 operation the servers have identical service rates, while in Type 2 operation the service rates are different. They established a cost relationship to determine the optimal queue length at which the second server is brought into operation.

In the literature described above, arrival rates and service rates are required to follow certain probability distributions with fixed parameters. However, in many real-world applications, the parameter distributions may only be characterized subjectively; that is, the arrival and service patterns are typically described in everyday language summaries of central tendency, such as "the mean arrival rate is around 10 customers per minute" or "the mean service rate is approximately 8 to 10 units per minute," rather than with complete probability distributions. Thus, trapezoidal fuzzy numbers have been taken for arrival and service rates.

In this paper heterogeneous server fuzzy queuing system has been used in heterogeneous computing network for analysing performance measures of the network. Here data arrival rates and service rates are taken as trapezoidal fuzzy numbers. Even though Fuzzy queues are much more realistic, the values of the system characteristics should be crisp for taking optimum decisions. Thus we develop an approach that provides system characteristics in terms of crisp values for heterogeneous-server queues with three fuzzy variables: one fuzzified exponential arrival and two fuzzified exponential service distributions. Robust ranking technique has been used to convert the fuzzy variables into crisp values. After that using crisp values performance measures have been evaluated [2].

2. HETEROGENEOUS SYSTEM ARCHITECTURE (HSA)

The computing industry is approaching a terrible obstacle course where anyone wishing to drive advances in computing technology must carefully negotiate several key trade-offs. First, reducing power consumption is increasingly critical across all segments of computing. Consumers want improved battery life, size, and weight for their laptops, tablets, and smart phones. [8] Likewise data center power demands and cooling costs continue to rise. At the same time, the researchers demand constantly improving performance to enable compelling new user experiences. User wants to access the devices through more natural interfaces (speech and gesture), and to manage ever-expanding volumes of data (home movies, pictures, and a world of content available in the cloud). To deliver these new user experiences, programmer productivity is another essential element that must be delivered. It must be easy for software developers to tap into new capabilities by using familiar, powerful programming models.

Finally, it is increasingly important that software be supported across a broad spectrum of devices. Developers cannot sustain today's trend of re-writing code for an ever expanding number of different platforms. To navigate this complex set of requirements, the computer industry needs a different approach – a more efficient approach to computer architecture. There is a need an approach that promises to deliver improvement across all 4 of these vectors: power, performance, programmability and portability.

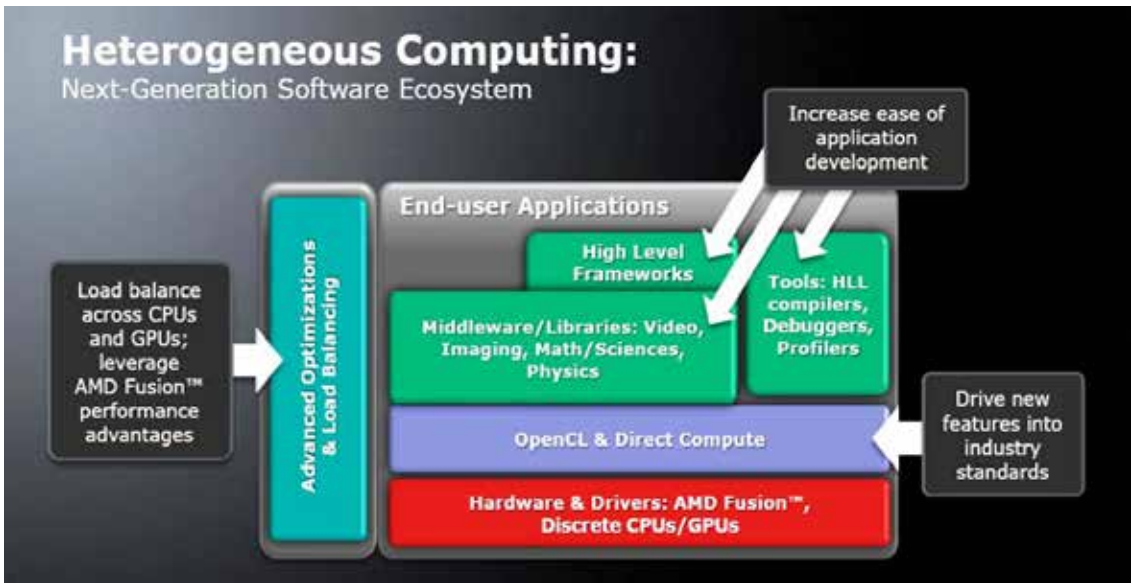

Figure 1. heterogeneous computing with ARM

Since their earliest days, computers have contained central processing units (CPUs) designed to run general programming tasks very well. But in the last couple of decades, mainstream computer systems typically include other processing elements as well. The most prevalent is the graphics processing unit (GPU), originally designed to perform specialized graphics computations in parallel. Over time, GPUs have become more powerful and more generalized, allowing them to be applied to general purpose parallel computing tasks with excellent power efficiency.

Today, a growing number of mainstream applications require the high performance and power efficiency achievable only through such highly parallel computation. But current CPUs and GPUs have been designed as separate processing elements and do not work together efficiently – and are cumbersome to program. Each has a separate memory space, requiring an application to explicitly copy data from CPU to GPU and then back again.

A program running on the CPU queues work for the GPU using system calls through a device driver stack managed by a completely separate scheduler. [9] This introduces significant dispatch latency, with overhead that makes the process worthwhile only when the application requires a very large amount of parallel computation. Further, if a program running on the GPU wants to directly generate work-items, either for itself or for the CPU, it is impossible today.

To fully exploit the capabilities of parallel execution units, it is essential for computer system designers to think differently. The designers must re-architect computer systems to tightly integrate the disparate compute elements on a platform into an evolved central processor while providing a programming path that does not require fundamental changes for software developers. This is the primary goal of the new HSA design.

HSA creates an improved processor design that exposes the benefits and capabilities of mainstream programmable compute elements, working together seamlessly.[10] With HSA, applications can create data structures in a single unified address space and can initiate work items on the hardware most appropriate for a given task. Sharing data between compute elements is as simple as sending a pointer. Multiple compute tasks can work on the same coherent memory regions, utilizing barriers and atomic memory operations as needed to maintain data synchronization (just as multi-core CPUs do today).

The HSA team at AMD group analyzed the performance of Haar Face Detect, a commonly used multi-stage video analysis algorithm used to identify faces in a video stream. The team compared a CPU/GPU implementation in OpenCL™ against an HSA implementation. The HSA version seamlessly shares data between CPU and GPU, without memory copies or cache flushes because it assigns each part of the workload to the most appropriate processor with minimal dispatch overhead. [11] The net result was a 2.3x relative performance gain at a 2.4x reduced power level. This level of performance is not possible using only multi-core CPU, only GPU, or even combined CPU and GPU with today's driver model. Just as important, it is done using simple extensions to C++, not a totally different programming model.

3. PRELIMINARIES

Zadeh in 1965 first introduced Fuzzy set as a mathematical way of representing impreciseness or indistinctness in everyday life [1].

Definition: A fuzzy set is characterized by a membership function mapping elements of a domain space or universe of discourse X to the unit interval [0,1]. i,e.

$$A = \{(x, \mu_A(x); x \in X\},$$

here μ_A: X→[0,1] is a mapping called the degree of membership function of the fuzzy set A and $\mu_A(x)$ is called the membership value of $x \in X$ in the fuzzy set A.These membership grades are often represented by real numbers ranging from [0,1].

Definition: A fuzzy set A of the universe of discourse X is called a normal fuzzy set implying that there exist atleast one $x \in X$ such that $\mu_A(x) = 1$.

Definition: The fuzzy set A is *convex* if and only if, for any $x_1, x_2 \in X$, membership function of A satisfies the inequality $\mu_A\{\lambda x_1 + (1-\lambda)x_2\} \geq \min\{\mu_A(x_1), \mu_A(x_2)\}$, $0 \leq \lambda \leq 1$.

Definition: (Trapezoidal fuzzy number)

For a Trapezoidal number A(x), it can be represented by A (a, b, c, d; 1) with membership function μ(x) given by

$$\mu(x) = \begin{cases} \frac{x-a}{b-a}, a \leq x \leq b \\ 1, b \leq x \leq c \\ \frac{d-x}{d-c}, c \leq x \leq d \\ 0, otherwise \end{cases}$$

Definition: (α-cut of a fuzzy number)

The α-cut of a fuzzy number A(x) is defined as

$A(x) = \{x: \mu(x) \geq \alpha, \alpha \in [0, 1]\}$

Addition of two Trapezoidal fuzzy numbers can be performed as

$(a_1, b_1, c_1, d_1) + (a_2, b_2, c_2, d_2) = (a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2)$.

4. FUZZY QUEUE WITH HETEROGENEOUS SERVERS

Consider an M/M/2/∞ queuing system in which the service rates of the two servers, denoted as μ_1 and μ_2 respectively, are distinct. It is assumed that customers arrive according to Poisson process with arrival rate λ. Arriving customers form a single waiting line and are served in the order of their arrival (i.e., first come first served). It is further assumed that each server may serve only one customer at a time, and that the service is independent of the arrival of new customers. If both servers are busy, arriving customers must wait until one is available. If both servers are idle, an arriving customer proceeds immediately to any server.

Let E[N] represent the average number of customers in the heterogeneous server M/M/2/∞ system and E[W] represent its mean response time (i.e., the expected waiting time in the system). Because the process is Markovian , E[N] and E[W] can be obtained in a simple way as follows:

$$E(N) = \frac{\lambda(\lambda + \mu_2)(\mu_1 + \mu_2)^3}{(\mu_1 + \mu_2 - \lambda)[(\mu_1 + \mu_2 - \lambda)(\mu_1 + \mu_2 + 2\lambda)\mu_1\mu_2 + \lambda(\lambda + \mu_2)(\mu_1 + \mu_2)^2]}$$

$$E(W) = \frac{(\lambda + \mu_2)(\mu_1 + \mu_2)^3}{(\mu_1 + \mu_2 - \lambda)[(\mu_1 + \mu_2 - \lambda)(\mu_1 + \mu_2 + 2\lambda)\mu_1\mu_2 + \lambda(\lambda + \mu_2)(\mu_1 + \mu_2)^2]}$$

In steady-state, $0 < \frac{\lambda}{\mu_1+\mu_2} < 1$

Suppose the arrival rate λ and service rates μ_1 and μ_2 are approximately known and can be represented by the fuzzy numbers $\tilde{\lambda}$, $\tilde{\mu_1}$, and $\tilde{\mu_2}$ respectively.

Let $\mu_{\tilde{\lambda}}(x)$, $\mu_{\tilde{\mu_1}}(y)$, $\mu_{\tilde{\mu_2}}(z)$ denote the membership functions of $\tilde{\lambda}$, $\tilde{\mu_1}$, and $\tilde{\mu_2}$ respectively. We then have the following fuzzy sets:

$\tilde{\lambda} = \{(x, \mu_{\tilde{\lambda}}(x)/ x \in X\}$

$\tilde{\mu_1} = \{(y, \mu_{\tilde{\mu_1}}(y)/ y \in Y\}$

$\tilde{\mu_2} = \{(z, \mu_{\tilde{\mu_2}}(z)/ z \in Z\}$ Where,

X, Y, and Z are the crisp universal sets of the corresponding arrival and service rates.

5. ROBUST RANKING TECHNIQUE-ALGORITHM

For solving the problem we defuzzify the fuzzy numbers into crisp ones by a fuzzy number ranking method [5] .Robust ranking technique which satisfies compensation, linearity, and additive properties and provides results which are consistent with human perceptions. Give a convex fuzzy number ã, the Robust ranking index is defined by

$R(\tilde{a}) = \int_0^1 0.5(a_\alpha{}^L + a_\alpha{}^U)d\alpha$,

Where $(a_\alpha{}^L, a_\alpha{}^U)$ is the α –level cut of the fuzzy number ã [6].

In this work, robust ranking technique has been used for ranking the fuzzy numbers. The Robust ranking index R(ã) gives the representative value of the fuzzy number ã. It satisfies the properties linearity, area compensation and additively.

6 NUMERICAL EXAMPLE

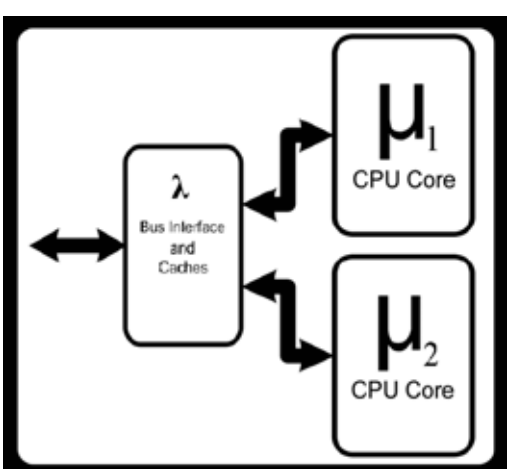

Figure 2 heterogeneous computing with two servers

An example, motivated by familiar real-life systems based on, is considered to demonstrate the practical use of the proposed solution.Suppose heterogeneous computing network has two servers (

Figure 2), with different efficiencies. After using these two servers for two months, the manager wants to know system characteristics, such as the expected number of jobs in the system and the mean response time. The job stream can be modelled as a Poisson process with fuzzy arrival rate $\tilde{\lambda}$ jobs/minute, and the service times can be modelled as exponential distributions with fuzzy service rates $\widetilde{\mu_1}$ and $\widetilde{\mu_2}$ jobs/ minute respectively. Trapezoidal fuzzy numbers are used to illustrate fuzzy numbers for all observations.

The expected number of jobs in the heterogeneous-server system (E[N]

The arrival and service rates are represented with Trapezoidal fuzzy numbers by

$\widetilde{\lambda}$ = [2,3,4,5], $\widetilde{\mu_1}$= [1,2,3,4], and $\widetilde{\mu_2}$ = [2,4,6,8].

First, it is easy to find that the membership functions for $\widetilde{\lambda}, \widetilde{\mu_1}, and\ \widetilde{\mu_2}$ are:

$$\mu_{\widetilde{\lambda}}(x)=\begin{cases}\frac{x-2}{1}, 2 \le x \le 3\\ 1, 3 \le x \le 4\\ \frac{5-x}{1}, 4 \le x \le 5\\ 0, otherwise\end{cases}$$

$$\mu_{\widetilde{\mu_1}}(y)=\begin{cases}\frac{y-1}{1}, 1 \le y \le 2\\ 1, 2 \le y \le 3\\ \frac{4-y}{1}, 3 \le y \le 4\\ 0, otherwise\end{cases}$$

$$\mu_{\widetilde{\mu_2}}(z)=\begin{cases}\frac{z-2}{2}, 2 \le z \le 4\\ 1, 4 \le z \le 6\\ \frac{8-z}{2}, 6 \le z \le 8\\ 0, otherwise\end{cases}$$

The α –cut of the fuzzy numbers $\widetilde{\lambda}, \widetilde{\mu_1}, and\ \widetilde{\mu_2}$ are given by

$[x_\alpha{}^L, x_\alpha{}^U]$ = [2+ α, 5- α], $[y_\alpha{}^L, y_\alpha{}^U]$ = [1+ 4α, 4-α], $[z_\alpha{}^L, z_\alpha{}^U]$ = [2+ α, 8- α]

$R(\widetilde{\lambda})$=R(2, 3, 4, 5)= $\int_0^1 0.5(a_\alpha{}^L + a_\alpha{}^U)d\alpha = \int_0^1 0.5(7)d\alpha = 3.5$

Proceeding similarly, the robust ranking indices for the fuzzy numbers $\widetilde{\mu_1}$, $\widetilde{\mu_2}$ are calculated as:

$R(\widetilde{\mu_1}) = R(1,2,3,4)=2.5$; $R(\mu_2)$ =R(2,4,6,8)=5

Hence λ=3.5 ; $\mu_1 = 2.5; \mu_2 = 5$

From traditional queuing theory formulas

The expected number of jobs in the heterogeneous server M/M/2/∞ system

$$E(N) = \frac{\lambda(\lambda + \mu_2)(\mu_1 + \mu_2)^3}{(\mu_1 + \mu_2 - \lambda)\left[(\mu_1 + \mu_2 - \lambda)(\mu_1 + \mu_2 + 2\lambda)\mu_1\mu_2 + \lambda(\lambda + \mu_2)(\mu_1 + \mu_2)^2\right]}$$

$$= \frac{3.5(3.5 + 5)(2.5 + 5)^3}{(2.5 + 5 - 3.5)[(2.5 + 5 - 3.5)(2.5 + 5 + 2(3.5))2.5(5) + 3.5(3.5 + 5)(2.5 + 5)^2]}$$

=1.5401customers.

The mean response time in the heterogeneous-server system E(W)

$$E(W) = \frac{(\lambda + \mu_2)(\mu_1 + \mu_2)^3}{(\mu_1 + \mu_2 - \lambda)[(\mu_1 + \mu_2 - \lambda)(\mu_1 + \mu_2 + 2\lambda)\mu_1\mu_2 + \lambda(\lambda + \mu_2)(\mu_1 + \mu_2)^2]}$$

$$= \frac{(3.5 + 5)(2.5 + 5)^3}{(2.5 + 5 - 3.5)[(2.5 + 5 - 3.5)(2.5 + 5 + 2(3.5))2.5(5) + 3.5(3.5 + 5)(2.5 + 5)^2]}$$

=0.44 minutes.

Similarly, expected number of jobs in the system, expected waiting time of a job in the system for different fuzzy arrival rate and service rates are calculated as follows:

Table 1

DIFFERENCE BETWEEN SERVERS EFFICIENCY =2 TIMES							
λ		μ_1		μ_2		$E(N)$	$E(W)$
Fuzzy arrival rate	Arrival rate	fuzzy service rate	Service rate	fuzzy service rate	Service rate		
(2,3,4,5)	3.5	(1,2,3,4)	2.5	(2,4,6,8)	5	1.5410	0.44
(7,8,9,10)	8.5	(6,7,8,9)	7.5	(12,14,16,18)	15	0.9435	0.111
(12,13,14,15)	13.5	(11,12,13,14)	12.5	(22,24,26,28)	25	0.9315	0.069

Table 2

DIFFERENCE BETWEEN SERVERS EFFICIENCY =4 TIMES							
λ		μ_1		μ_2		$E(N)$	$E(W)$
Fuzzy arrival rate	Arrival rate	fuzzy service rate	Service rate	fuzzy service rate	Service rate		
(2,3,4,5)	3.5	(1,2,3,4)	2.5	(4,8,12,16)	10	1.246	0.356
(7,8,9,10)	8.5	(6,7,8,9)	7.5	(24,28,32,36)	30	1.14750	0.315
(12,13,14,15)	13.5	(11,12,13,14)	12.5	(44,48,52,56)	50	0.6751	0.05

Table 3

DIFFERENCE BETWEEN SERVERS EFFICIENCY = 6 TIMES							
λ		μ_1		μ_2		$E(N)$	$E(W)$
Fuzzy arrival rate	Arrival rate	fuzzy service rate	Service rate	fuzzy service rate	Service rate		
(2,3,4,5)	3.5	(1,2,3,4)	2.5	(6,12,18,24)	15	0.7352	0.21
(7,8,9,10)	8.5	(6,7,8,9)	7.5	(36,42,48,54)	45	0.6811	0.08
(12,13,14,15)	13.5	(11,12,13,14)	12.5	(66,72,78,84)	75	0.6211	0.046

7. CONCLUSION

In this work, fuzzy queuing system with heterogeneous servers has been used to evaluate the system performance measures of heterogeneous computing network. Since data arrival rate and service rates of servers are fuzzy numbers, the system is more realistic and general in nature. As results of system characteristics are crisp values, the system manager can take optimum decisions. From table 1, 2 and 3 conclusion has been made that when difference between efficiency of two server increases, the system attains better performance measures. Though installation of servers with high efficiency is costly, compare to system length and data waiting time; installation of high efficiency servers will be optimum for system managers.

REFERENCES

1. 1.L.A .Zadeh, "Fuzzy sets", Information and control ,Vol.8 ,pp. 338-353, 1965. | 2. R. Nagarajan and A.Solairaju, "Computing improved fuzzy optimal Hungarian assignment problems with fuzzy costs under robust ranking techniques, "International Journal of Computer Applications", Vol. 6,(4), pp.6-13,2010. | 3. R. L Garg and P. Singh, "R Queue-dependent servers queuing system", Microelectronics and Reliability,Vol. 33, ,pp. 2289–2295, 1993. | 4. V. P. Singh, "Queue-dependent servers, Journal of Engineering Mathematics", Vol.7, pp.123–126, 1973. | 5. P. Fortemps and M. Roubens , " Ranking and defuzzification methods based on area compensation", Fuzzy Sets and Systems, Vol. 82, pp.319–330, 1996. | 6. R. R. Yager, "A procedure for ordering fuzzy subsets of the unit interval", Information Sciences, Vol.24, pp.143–161, 1981. | 7. Jau-Chuan Ke, Hsin-I Huang , Chuen-Horng Lin, "Analysis on a queue system with heterogeneous servers and uncertain patterns", Journal of Industrial and Management Optimization, Vol 6, (1), pp. 57–71, 2010. | 8. Kirill Mechitov, Gul Agha, "Building Portable Middleware Services for Heterogeneous Cyber-Physical Systems" IEEE journal, pp. 31 – 36, 2012. | 9. Alvin Ting, David Chieng, Kae Hsiang Kwong, Ivan Andonovic, K. D. Wong, "Dynamic Backhaul Sensitive Network Selection Scheme in LTE-WiFi Wireless HetNet", IEEE 24th International Symposium on Personal, Indoor and Mobile Radio Communications, pp. 3061-3065, 2013. | 10. Rajiv Chechi, Rajesh Khanna, "QoS BASED HANDOVER IN HETROGENEOUS NETWORKS", IEEE journal, pp.5021-5024, 2011. | 11. Chang Kyung Sung and Iain B. Collings, "Lifetime Maximization for Sensor Networks with Hetrogeneous Nodes", IEEE journal, vol 70, pp.1-5, 2009. |