ISSN: 2277 – 9043
International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE)
Volume 5, Issue 7, July 2016

UNI Processor and Multi-processor Performance Comparison

Muhammad Haris[1], Naila Maqsood[1], UbaidUl Haq[1], Tahir Zaman[1], Muhammad Zubair[1]
[1]Department of Computer Science, Abdul Wali Khan University Mardan, Pakistan

Abstract- This research is about the comparison between the performances of single and multi-processing system. We will look that how jobs are processed in a uniprocessor and multiprocessor. We will work on process scheduling in both uni and multiprocessing systems. Some advantages and disadvantages are discussed here. Various workloads are tested to show what factors affecting their performance. Significant effort is made to make a fair comparison between them. The performance of various architectures is tested for uniprocessor system. Also much architectureis tested for the performance of multi-processor system.

Keywords: UNI, Multi-processor, architecture.

I. INTRODUCTION

Now a dayevery system is busy in supporting several programs. To handle these several programs a good processing system is required. There are two types of systems a Uni-processing system and multiprocessing system [1-4].

A uniprocessor system consists of a single processor which executes the program instructions as a single instruction per cycle. In uniprocessor system the memory is used only by a single processor [5].A multiprocessor system consists of more than one processor in a single chip which executes more than one instruction in a single cycle. In this system memory is shared among several processors.

In single processor the execution of a program takes more time than multi-Processing system because in multi-processing system the program is divided into parts and is executed parallel [6-9]. Parallelism is important because big problems can be split into smaller programs. That can be executed at the same time to reduce Execution time. In this research we are focusing on the workloads on these two architectures. The first objectives of this research are to undertake such a comparison. The comparison starts from the analysis of the uniprocessor and then leading towards the performance of multiprocessing system.

II. Univs. multiprocessor requirements

Uniprocessor standard architecture systems with well cached cpu do not use pipeline mode as frequently. For this reason the capability on p6 bus and of ram bus does not have a very significant performance. The architecture of a UNI processing system is shown in Figure 1.

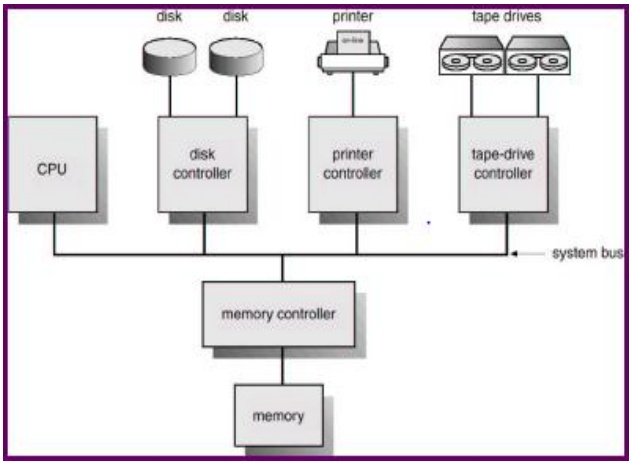


Fig1: Uni processing system

In a P2/450 uniprocessor PC, DRAM bus saturation typically hovers between 1 and 10%. By adding more processors, bus saturation can easily be driven beyond the bandwidth [10-19] capacity of 100 MHz SDRAM. To solve this problem, multiprocessor platforms may use multiple 64 bit buses, or increase the bus width to 128 bits or 256 bits. These configurations also allow servers to satisfy another important requirement – memory capacity. Large servers must usually ship with half a gigabyte of DRAM and be Configurable up to several gigabytes. This can be accomplished today using wide SDRAM configurations with registered DIMMs. But Rambus

All Rights Reserved © 2016 IJARCSEE

presents another problem in this area. Each Rambus interface channel allows only 32 memory Integrated circuits. Using 64 Mega bit DRAMs, the maximum configuration for a single channel is 256 Mega Bytes. Even in late 1999 or early 2000 when 256Mega bit components become available, capacity will still be constrained to 1 Gigabytes per channel. In order to meet platform requirements [20-27], multiple Rambus channels will be required, or repeater chips must be used to extend the capacity (and increase the latency) of each channel

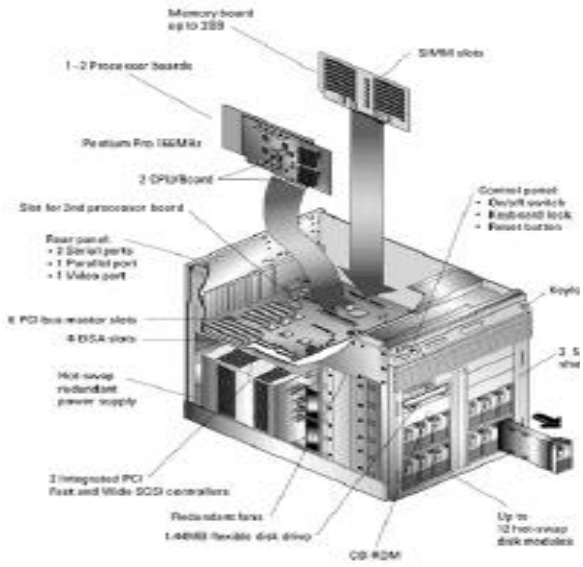


Fig2: Multiprocessing system

Advantages ofuni and multiprocessing system

1. multiprocessing system

As compared to the uniprocessor system multiprocessor systems have many advantages which are following,

- Better Performance

  Due to more than one processors the multi-processor system having a better performance by shorter response time and higher throughput than a single processor system. For example if there are two programs are to be run than a multiprocessor system is better than a single processor system because both programs will be run Simultaneously on different processors. Or if a single large program is run on both systems, so multiprocessor system will response quick then single processor because the program will divide in parts and every part will run parallel on different processors.

- Better reliability

  Multiprocessor system is more reliable than single processor system. In a properly designed multiprocessing system if one processor breaks down the other processes takes over the system load automatically, so there are less chances of complete failure.

From above discussion we can get the following advantages

- Increased reliability
- Increased throughput
- Economy of scale
- Higher performance due to parallel processing.
- If one CPU get failed other CPU are used to do that CPUs job

2. Uni processing system

- Main memory requirements

  A uni processing system having only one Processor so it can process only one job at a time so less memory is required for a single job. If we compare it with multiprocessing system than multiprocessing system requires more memory to process multi programs.

- Operating system scheduling

  For a single processor the job scheduling is very easy because there is only one processor for which the operating system will schedule jobs on the basis of some algorithms which are discussed later.

All Rights Reserved © 2016 IJARCSEE

Disadvantages of both systems

1. **multiprocessing system**

With advantages there are many disadvantages of multiprocessing system which are discussed following:

- Speed effect

  Multi-processing systems are designed for quick response so if one processor damaged than it will affect the speed of the system.

- Expensiveness

  In a multi-processing system more than one processors are connected with each other in single chip so for the designing of such a chip expensive tools and experts are required which are very expensive. Also there are many processors so every processor having own cost as more processors the system will be more expensive.

- Complex operating system

  As there are more than one processors for multi programs so this is a complex job to schedule the jobs and processors.

- Large main memory required

  As there are more than one jobs are running on different processors so a lot of memory is required for these jobs to run.

2. **Uni processing system**

The disadvantages of single processing system are given below:

- Complete breakdown of system

  As there is only a single processor so if it is damaged then the whole system will be breakdown.

- Low throughput

  Due to single processor only one job is processed so in a given interval of time only a single job is done or if the jobs are small then few jobs are completed so throughput is very low.

- Higher response time

  As only one processor handles the jobs so the complete Burdon of the Job is on single processor so it will take more time to complete the job.

III. Process scheduling in both systems

1. Uniprocessing system

In uni processing system we will study the basic concepts, scheduling criteria and algorithms.

- **Basic concepts**

Three types of scheduling is used in uni processing system:

1) Long term
   If there are multi proms for execution then long term scheduling decides that which program to be admitted to the system for execution.

2) Middle term
   If a program is running and make a i/o request then or if a program completes a request then it decides that when the program should be resume or suspend.

3) Short term
   Decides which process will be stopped; invoked upon
   – Interrupts, Operating system calls, Signals, etc.

All Rights Reserved © 2016 IJARCSEE

The first two (long term, middle term) control the degree of multi programming because there are more process and short amount of time but in this short amount of time all the processes will be executed .

We will discuss the short term scheduling here.

Short term scheduling (Sts)

The short term scheduling dispatch one program and starts another. So two dominating factors involve here

Switching context, selecting the new process to dispatch.

- **Scheduling criteria for sts.**

CPU utilization: It keeps the CPU busy when there are processes to be executed.

Throughput: It means that number of process completed in the given interval of time.

Response time: This is the time when a process made a request to the completion of its execution. (Waiting time + Execution time).

Fairness: Watch priorities and to avoid starvation.

Over heading: when a lot of burden on the CPU occurs due to context switching and due to priorities.

Preemptive: The continued process will be continuous until it is finish or made an I/O request.

Non Preemptive: Currently running process may be interrupted and move to the ready queue. This preemption may be on priority base or time slice.

- **Algorithms for Sts.**

As we have studied that uniprocessor systems only process a single program. Here a question arises that what if several jobs are in the queue for process? What the operating system will do with the execution of several jobs? So here are some algorithms discussed so solve these questions. That algorithms are:

First come first served (FCFS)

Round robin

Shortest job first

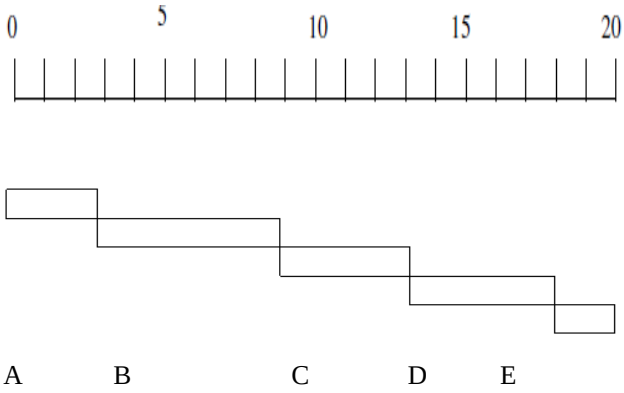
Priority based jobs first

Here we discussed all these techniques one by one.

First come first served (FCFS):

This algorithm process those jobs first which comes first for execution. This algorithm works on first in first out. This is a non-preemptive algorithm. Itmeans that if one job is in process, the other jobs will wait until its completion [30].

For example if there are some jobs for process A B C D and E. the execution time of job A is 3, job B is 6, job c is 4, job is 5 and job E is 2 seconds. Here the first job is A and the last is E. the shortest job is E with 2 seconds execution time.



Process	Arrival Time	Process time
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

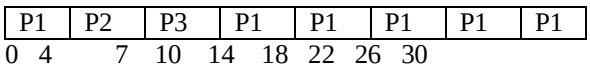
The drawback of this algorithm is that if a job has minimum execution time and comes after the large program so it will wait to finish the first program. As in the above example job E has 2 second time but it will wait for all the jobs to complete.

Round Robin:

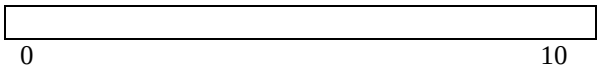
This algorithm is based on FCFS, but the preemption is involved here. The preemption is on the basis of time slice. The time slice is from 10 to 100 milliseconds. To implement RR scheduling, we keep

All Rights Reserved © 2016 IJARCSEE

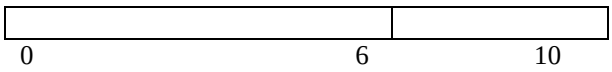
the ready queue as a FIFO queue of processes. New processes are added to the tail of the queue. The CPUscheduler picks a process from the ready queue and start the timer. If the process completes in this quantum then next process is places otherwise the first process is replaced by the second process and the first process is placed in the waiting queue. This process is known as context switching.If we have three processes to process P1, P2, and P3 having 24, 3 and 3 milliseconds respectively [31-39]. The slice is 4 milliseconds. The first process will start execution. It has the execution time of 24 milliseconds and the time quantum is 4ms. So the execution will terminate after 4 milliseconds and next job will be put for execution. It has the execution time of 3 milliseconds so it will complete its execution and will leave the CPU itself. Then job p3 will come for execution which has the quantum of 3 milliseconds, so it will also complete its execution and will leave the CPU itself. In this Round Robin scheduling algorithm, no process is allocated the CPU for more than 1 time quantum in a row until it is the only runnable process.



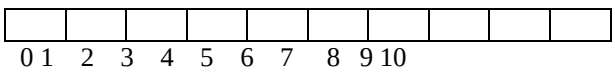
Round robin algorithm depends on time quantum which leads to the context switches. If the quantum is large then the context switches will be less. But if the quantum is small then there will be more context switching.
For example



Here the quantum is 10 so the no context switching.



Here the time quantum is 6 so the context switching will be twice.



Here the quantum is 1 so the context switching will be 10 times.

The last example leads the over heading due to more context switching.

Although the time slice should be large compared with the context switch time, it should not be too large. If the time quantum is too large, RR
Scheduling degenerates to FCFS policy. A rule of thumb is that 80 percent of the CPU bursts should be shorter than the time quantum.

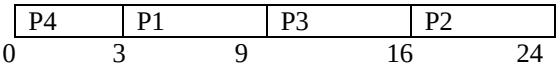
Shortest job first:

This is a different and non-preemptive approach from other algorithms because it prefers the job which has minimum execution time. When the CPU is free it is assign to the job which has the minimum CPU burst time. If the two jobs has the same execution time then FCFS is used to execute the jobs.

Here is an example of this algorithm, consider the following jobs with execution time in milliseconds

Process	CPU Time
P1	6
P2	8
P3	7
P4	3

Using SJF we will schedule the jobs as following:



The total waiting time for P1 is 3 Milliseconds, P2 has 16 Milliseconds, and P3 has 9 Milliseconds and P4 has 0Milliseconds. So the average waiting time is (3+16+9+0)/4=7 Milliseconds. If we compare this with FCFS then average waiting time for FCFS is 10.25 Milliseconds. So this SJF is better than FCFS.

This algorithm is better if we know the execution of the next job, but is not optimal when the execution time is not given.

Priority based algorithm:

In this algorithm the scheduler chooses a high priority job for execution over a low priority. The best example of this algorithm is SJF where the priority is on low burst time. In this algorithm priority is assigned to each process and then CPU is assigned to the high priority job. If the two processes are the same priority then FCFS is used to solve the problem.

All Rights Reserved © 2016 IJARCSEE

Here is an example of this algorithm, consider the following jobs with the Burst time and priority. The arrival time is same for all processes.

Process	CPU Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	5	2

Using priority scheduling we will schedule the jobs as following:

P2	P4	P1	P3
0 1	6	16	18

The average waiting time is 10.25 milliseconds.

Priorities can be defined as an internal or external. Internally defined priorities use some quantities to compute the priority of a process. For example, time limits, memory requirements, the number of open files, and the ratio of average I/O burst to average CPU burst have been used in computing priorities. External priorities are set by criteria outside the operating system, such as the importance of the process, the type and amount of funds being paid for computer use, the department sponsoring the work.
Priority scheduling can be preemptive or non-preemptive. The newly arrived job will be compared to the old jobs if the new job has larger priority than the old process will be terminated and the new job will be placed to execution. In case of SJF the priority is non preemptive.
A major problem with priority scheduling algorithms is **indefinite blocking,** or **starvation.** A process that is ready to run but waiting for the CPU can be considered blocked. A priority scheduling algorithm can leave some low priority processes waiting indefinitely.
For the above problem **aging** is introduced. In aging the low priority get some priority after some time spend.

2. Multi-processing system

So far we were discussing the scheduling criteria for single processing system. The multi-processing system helps in load sharing, but the process synchronization become more complex. Many possibilities have been tried, as we seen in single processing system no best solution occurs. Here we discuss several concerns with multi-processing system.

Approaches to Multiple-Processor Scheduling:

One approach is that one processor in multi-processing system gets the responsibility of I/O processing and other system activities, this processor is known as master processor. The other processors executes only user code. This approach is known as "asymmetric multiprocessing". This is a simple approach because only one processor access the system data structure and reducing the need of data sharing.

The other approach is "symmetric multiprocessing". Here every processor has to schedule itself. Each process has its own ready queue of processes or all processes may be a common ready queue. Regardless the scheduler of each processor examines the processes ready queue and select a process for execution. The operating system must ensure that the same process may not processed by two processors. Virtually all modern operating systems support SMP, including Windows XP, Windows 2000, Solaris, Linux, and Mac OS X.

Processor Affinity:

What is the role of cache memory when a processor processes a process? The most recently used data of the process populates the cache for the processor, this results a successive data process between processor and cache. Now what will happen if a process is transferred from one processor to another processor in execution? The data of cache memory must be invalidated from that processor which the data is transferred from. And to which processor the data is transferred the cache is re populated. Because of the high cost of invalidating and re-populating caches, most SMP systems try to avoid migration of processes from one processor to another and instead attempt to keep a process running on the same processor. This is known as **processor affinity,** meaning that a process has an affinity for the processor on which it is currently running. Processor affinity takes several forms. When an operating system has a policy of attempting to keep a process running on the same processor—but not guaranteeing that it will do so— we have a situation known as **soft affinity.** Here, it is possible for a process to migrate between processors. Some systems
—such as Linux—also provide system calls that support **hard affinity,** thereby allowing a process to specify that it is not to migrate to other processors.

Load balancing:

All Rights Reserved © 2016 IJARCSEE

On semantic multi programming systems, it is important to keep the workload balanced between all processors to completely utilize the benefits of having multiple processors.
Otherwise, one or more processors may sit idle while other processors have high workloads along with lists of processes awaiting the CPU. On semantic multi programming system**loadbalancing** tries to keep the workload distributed among all the processors. It is important to know that load balancing is important only when every processor has its own queue of processes to execute.
For a common queue load balancing is often unnecessary because when a processor become idle, it gets another process from the running queue.

There are two techniques used for load balancing, **Push migration** and **pull migration**.

With push migration a task is performed which checks the load on each processor, if an unbalance is found it push processes from overloaded to idle processor or from overloaded to less using processor.

The pull migration, an idle processor pulls a waiting process from a busy processor.

Push and pull migration need not be mutually exclusive and are in fact often implemented in parallel on load-balancing systems.
For example, the Linux scheduler and the ULE scheduler available for FreeBSD systems implement both techniques. Linux runs its balancing algorithm every 200 milliseconds (push migration) or whenever the run queue for a processor is empty (pull migration).

Symmetric Multithreading:

Symmetric multi-processing systems allow several threads to run concurrently by providing multiple physical processors.
An alternative strategy is to provide multiple logicalprocessors rather than physicalprocessors.Such a strategy is known as symmetric multithreading (or SMT).It has also been known as "hyper threading technology"on Intel processors.
The idea behind SMT is to create multiple logically processors on the same physical processor. It represent multiple logical processor to the operating processor, also on a system having single processor.
Each logical processor has its own "architecture state", which includes general-purpose and machine-state registers. Here interrupts handling is done by these logical processor rather than physical processor. Otherwise, each logical processor shares the resources of its physical processor, such as cache memory and buses.

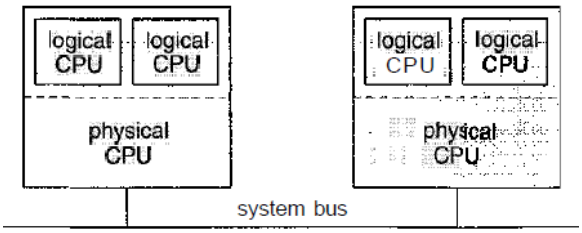


Figure 5.8 A typical SMT architecture

The above figure shows two physical processor which having 2, 2 logical processors. For the operating system there are four processors in this system.

IV. CONCLUSION

This research was about the comparison of multi processer and uni processor performance. We had worked on advantages and disadvantages of both systems. We also studied that how operating system assigns jobs to uniprocessor and multiprocessor systems. In uni processor scheduling we studied different algorithms which include FCFS, Round robin, and shortest job first and priority jobs. We also studied their waiting time and compared with other algorithms. For multiprocessing systems we studied different scheduling criteria's.
At the end we concluded that multiprocessing system is best due to short response time and large throughput. We can run different jobs at a time which will run on different processors in the same system.
In the future all the work may be on the multiprocessing system to improve their performance and to reduce the complication of operating system job scheduling.

REFERENCES

[1]. Khan, F., & Nakagawa, K. (2013). Comparative study of spectrum sensing techniques in cognitive radio networks. In *Computer and Information Technology (WCCIT), 2013 World Congress on* (pp. 1-8). IEEE.

[2]. Khan, F., Bashir, F., & Nakagawa, K. (2012). Dual head clustering scheme in wireless sensor networks. In *Emerging Technologies (ICET), 2012 International Conference on* (pp. 1-5). IEEE.

[3]. M. A. Jan, P. Nanda, X. He, and R. P. Liu, "A

All Rights Reserved © 2016 IJARCSEE

Lightweight Mutual Authentication Scheme for IoT Objects," *Future Generation Computer Systems (FGCS)*, "Submitted", 2016.

[4]. Khan, F., Kamal, S. A., &Arif, F. (2013). Fairness improvement in long chain multihop wireless ad hoc networks. In *2013 International Conference on Connected Vehicles and Expo (ICCVE)* (pp. 556-561). IEEE.

[5]. Khan, F. (2014). Secure communication and routing architecture in wireless sensor networks. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)* (pp. 647-650). IEEE.

[6]. M. A. Jan, P. Nanda, X. He and R. P. Liu, "PASCCC: Priority-based application-specific congestion control clustering protocol" Computer Networks, Vol. 74, PP-92-102, 2014.

[7]. Khan, S., & Khan, F. (2015). Delay and Throughput Improvement in Wireless Sensor and Actor Networks. In *5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)* (pp. 1-8).

[8]. Khan, F., Jan, S. R., Tahir, M., Khan, S., &Ullah, F. (2016). Survey: Dealing Non-Functional Requirements at Architecture Level. *VFAST Transactions on Software Engineering*, *9*(2), 7-13.

[9]. Khan, F., & Nakagawa, K. (2012). Performance Improvement in Cognitive Radio Sensor Networks. *the IEICE Japan*.

[10]. Khan, F., Khan, S., & Khan, S. A. (2015, October). Performance improvement in wireless sensor and actor networks based on actor repositioning. In *2015 International Conference on Connected Vehicles and Expo (ICCVE)* (pp. 134-139). IEEE.

[11]. M. A. Jan, P. Nanda, X. He and R. P. Liu, "A Sybil Attack Detection Scheme for a Centralized Clustering-based Hierarchical Network" in Trustcom/BigDataSE/ISPA, Vol.1, PP-318-325, 2015, IEEE.

[12]. Jabeen, Q., Khan, F., Khan, S., & Jan, M. A. (2016). Performance Improvement in Multihop Wireless Mobile Adhoc Networks. *the Journal Applied, Environmental, and Biological Sciences (JAEBS), 6(4S)*, 82-92.

[13]. Khan, F. (2014, May). Fairness and throughput improvement in multihop wireless ad hoc networks. In *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on* (pp. 1-6). IEEE.

[14]. Khan, S., Khan, F., Arif, F., Q., Jan, M. A., & Khan, S. A. (2016). Performance Improvement in Wireless Sensor and Actor Networks. *Journal of Applied Environmental and Biological Sciences*, *6(4S)*, 191-200.

[15]. Khan, F., & Nakagawa, K. (2012). B-8-10 Cooperative Spectrum Sensing Techniques in Cognitive Radio Networks. *電子情報通信学会ソサイエティ大会講演論文集*, *2012*(2), 152.

[16]. M. A. Jan, P. Nanda, X. He, and R. P. Liu, "A Lightweight Mutual Authentication Scheme for IoT Objects," *Elsevier Future Generation Computer Systems (FGCS)*, "Submitted", 2016.

[17]. Khan, F., Jan, S. R., Tahir, M., & Khan, S. (2015, October). Applications, limitations, and improvements in visible light communication systems. In*2015 International Conference on Connected Vehicles and Expo (ICCVE)*(pp. 259-262). IEEE.

[18]. Jabeen, Q., Khan, F., Hayat, M. N., Khan, H., Jan, S. R., &Ullah, F. (2016). A Survey: Embedded Systems Supporting By Different Operating Systems. *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Print ISSN*, 2395-1990.

[19]. Jan, S. R., Ullah, F., Ali, H., & Khan, F. (2016). Enhanced and Effective Learning through Mobile Learning an Insight into Students Perception of Mobile Learning at University Level. *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Print ISSN*, 2395-1990.

[20]. Jan, S. R., Khan, F., &Zaman, A. The perception of students about mobile learning at University level.

[21]. M. A. Jan, P. Nanda, X. He, and R. P. Liu, "A Sybil Attack Detection Scheme for a Forest Wildfire Monitoring Application," *Elsevier Future Generation Computer Systems (FGCS)*, "Accepted", 2016.

[22]. Jan, S. R., Shah, S. T. U., Johar, Z. U., Shah, Y., & Khan, F. (2016). An Innovative Approach to Investigate Various Software Testing Techniques and Strategies. *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Print ISSN*, 2395-1990.

[23]. Khan, I. A., Safdar, M., Ullah, F., Jan, S. R., Khan, F., & Shah, S. (2016). Request-Response Interaction Model in Constrained Networks. *In International Journal of Advance Research and Innovative Ideas in Education, Online ISSN-*

All Rights Reserved © 2016 IJARCSEE

2395-4396

[24]. Azeem, N., Ahmad, I., Jan, S. R., Tahir, M., Ullah, F., & Khan, F. (2016). A New Robust Video Watermarking Technique Using H. 264/AAC Codec Luma Components Based On DCT. *In International Journal of Advance Research and Innovative Ideas in Education, Online ISSN-2395-4396*

[25]. Rehman, Ateeq Ur, Lie-Liang Yang, and LajosHanzo. "Performance of Cognitive Hybrid Automatic Repeat reQuest: Stop-and-Wait." *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*.IEEE, 2015.

[26]. Jan, S. R., Khan, F., Ullah, F., Azim, N., &Tahir, M. (2016). Using CoAP Protocol for Resource Observation in IoT. *International Journal of Emerging Technology in Computer Science & Electronics, ISSN: 0976-1353*

[27]. Azim, N., Majid, A., Khan, F., Jan, S. R., Tahir, M., &Jabeen, Q. (2016). People Factors in Agile Software Development and Project Management. *In International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)* ISSN: 0976-1353

[28]. Azim, N., Majid, A., Khan, F., Tahir, M., Safdar, M., &Jabeen, Q. (2016). Routing of Mobile Hosts in Adhoc Networks. *In International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)* ISSN: 0976-1353.

[29]. *Azim, N., Khan, A., Khan, F., Majid, A., Jan, S. R., &Tahir, M. (2016) Offsite 2-Way Data Replication toward Improving Data Refresh Performance. In International Journal of Engineering Trends and Applications,* ISSN: 2393 – 9516

[30]. Tahir, M., Khan, F., Jan, S. R., Azim, N., Khan, I. A., &Ullah, F. (2016) EEC: Evaluation of Energy Consumption in Wireless Sensor Networks. . *In International Journal of Engineering Trends and Applications,* ISSN: 2393 – 9516

[31]. Rehman, A. U., Yang, L. L., &Hanzo, L. (2016, May). Performance of Cognitive Hybrid Automatic Repeat reQuest: Go-Back-N. In *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)* (pp. 1-5).IEEE.

[32]. M. A. Jan, P. Nanda, M. Usman, and X. He, "PAWN: A Payload-based mutual Authentication scheme for Wireless Sensor Networks," *Concurrency and Computation: Practice and Experience*, "accepted", 2016.

[33]. Azim, N., Qureshi, Y., Khan, F., Tahir, M., Jan, S. R., &Majid, A. (2016) Offsite One Way Data Replication towards Improving Data Refresh Performance. *In International Journal of Computer Science Trends and Technology,* ISSN: 2347-8578

[34]. Safdar, M., Khan, I. A., Ullah, F., Khan, F., & Jan, S. R. (2016) Comparative Study of Routing Protocols in Mobile Adhoc Networks. *In International Journal of Computer Science Trends and Technology*, ISSN: 2347-8578

[35]. Tahir, M., Khan, F., Babar, M., Arif, F., Khan, F., (2016) Framework for Better Reusability in Component Based Software Engineering.*In the Journal of Applied Environmental and Biological Sciences (JAEBS), 6(4S),* 77-81.

[36]. Khan, S., Babar, M., Khan, F., Arif, F., Tahir, M. (2016). Collaboration Methodology for Integrating Non-Functional Requirements in Architecture. *In the Journal of Applied Environmental and Biological Sciences (JAEBS), 6(4S),* 63-67

[37]. Jan, S.R., Ullah, F., Khan, F., Azim, N., Tahir, M., Khan, S., Safdar, M. (2016). Applications and Challenges Faced by Internet of Things- A Survey. *In the International Journal of Engineering Trends and Applications,*ISSN: 2393 – 9516

[38]. Tahir, M., Khan, F., Jan, S.R., Khan, I.A., Azim, N. (2016). Inter-Relationship between Energy Efficient Routing and Secure Communication in WSN. *In International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)* ISSN: 0976-1353.

All Rights Reserved © 2016 IJARCSEE